EPS
Escola Politècnica Superior
UdG

Treball final de grau

Estudi: Grau en Enginyeria Electrònica Industrial i Automàtica

Títol: Implementació d'arquitectura ROS al robot BigBot

Document: 1. Memòria

Alumne: Jordi Hortal Garí

Director/Tutor: Albert Figueres i Xavier Cufí
Departament: EEEA / ATC
Àrea: ESA / ATC

Convocatòria (mes/any): Setembre 2014

Índex

1 INTRODUCCIÓ

Es planteja implementar l'arquitectura software ROS en una nova placa-PC que serà incorporada al robot de rescat BigBot. Així el robot disposarà de les capacitats suficients que li permetran ser més autònom. El robot Bigbot és una plataforma mòbil que està destinada a treballar conjuntament amb un gos ensinistrat en operacions de cerca de persones en situacions de rescat. Fruit del resultat del treball desenvolupat en aquest TFG el robot ha de disposar d'un grau d'autonomia més elevat.

1.1 ANTECEDENTS

La universitat disposa d'una flota de robots de rescat desenvolupada en el laboratori de sistemes intel·ligents de l'EPS. L'arquitectura software del robots (BIGBOT) estava realitzada en LABVIEW, depenent d'un sistema de control extern, actualment, s'està adaptant per treballar amb ROS (Robot Operating System), ja que està esdevenint un estàndard per a diferents plataformes robotitzades. El paquet ROS permet compartir esforços de diferents grups de treball relacionats en desenvolupar arquitectura software per a diferents robots mòbils, atès que és un software de codi lliure.

El sistema ROS proporciona els serveis generals d'un sistema operatiu, com són l'abstracció hardware, el control de baix nivell de dispositius i la comunicació entre dispositius a través d'un protocol de comunicació natiu de ROS.

1.2 OBJECTE

L'objectiu del projecte és dotar d'una més gran autonomia el robot BigBot, implementant l'arquitectura ROS i una nova placa-PC nova placa de control basada en un PC industrial miniaturitzat, escollida entre les existents en el mercat i compatible amb els sensors existents com poden ser: la càmera de visió, Làser Range Finder, Unitat de Mesura Inercial (IMU).

1.3 ESPECIFICACIONS I ABAST

L'abast d'aquest projecte serà la incorporació completa de la nova placa-PC al robot. Aquesta serà escollida amb les prestacions necessàries per comunicar tots els sensors que disposa actualment el robot per a la seva navegació i implementar l'arquitectura ROS.

2 PROJECTE MATE

El projecte de recerca en robòtica de rescat (MATE) es centra en la recerca de robòtica per rescat. La proposta de projecte de recerca MATE es basa en la utilització d'un robot mòbil per a donar suport als equips de rescat, formats per persones expertes i gossos ensinistrats, en la recerca de persones en entorns pre-urbans i urbans on s'ha produït una catàstrofe.

Hi han diverses metodologies utilitzades avui en dia en la recerca de persones degut a que hi ha diverses situacions diferents com terratrèmols, inundacions o esfondraments, entre d'altres. Per aquest motiu, bombers i/o serveis de rescats han d'actuar davant de circumstancies de perill elevat, per minimitzar riscos i danys els robots de rescat podrien ajudar.

Un dels mètodes més utilitzats en el rescat de persones és la utilització de gossos ensinistrats, aquests animals disposen d'un molt desenvolupat sentit de l'olfacte, i amb les seves dimensions i agilitat, són capaços d'endinsar-se en zones on una persona no podria entrar-hi. En aquest punt és on entra el projecte MATE.

La principal idea en la que parteix aquest projecte, és la de realitzar un robot mòbil capaç de cooperar i interactuar amb un gos ensinistrat i seguir-lo dins una situació de recerca o rescat de persones en entorns pre-urbans o urbans. El robot, sempre supervisat per un operari expert remotament, incorpora un gran nombre d'aplicacions que juntament amb les característiques del gos, incrementarien l'eficiència dels grups de rescat davant un gran nombre de situacions catastròfiques diferents.

Figura 1. Projecte MATE

L'altra part del projecte, el que s'està assajant actualment, és la implementació del robot i la seva comunicació amb experts humans de l'equip de rescat. Aquest és un dels punts a

destacar del MATE, ja que a part de millorar la recerca i/o rescat de persones, s'aconsegueix aïllar l'ensinistrador del gos de la zona de risc situant-lo en una zona de control i monitorització fora de perill.

Per realitzar-ho, s'està implementant un robot mòbil autònom, capaç de seguir al gos amb la capacitat d'evitar obstacles, però al mateix temps, emetent informació auxiliar a l'equip de rescat a través dels diferents sensors que incorpora, com la càmera RGB-D o el sensor de gasos. Permeten així realitzar una comunicació entre l'ensinistrador i el gos utilitzant el robot d'intermediari.

Un cop aconseguit això, el robot estarà acompanyant al gos ensinistrat i permetrà que els operadors experts humans puguin interactuar amb el gos ensinistrat, sense posar-se en perill, mitjançant el robot mòbil, i a més, aquest ha de poder recollir informació auxiliar de la zona d'operació per a transmetre-la cap a les sales de control de les operacions de rescat o altres robots treballant en el mateix rescat.

Per dur a terme aquest projecte, es parteix de la flota de robots de rescat realitzada al laboratori ARLAB, fabricats anteriorment. Els quals s'adaptaran per poder utilitzar-los com a primer prototip d'enllaç entre un expert i el gos.

Les tasques de desenvolupament inicial del projecte es realitzen en 2 treballs final de carrera (un d'ells es aquest treball final de grau) i una tesi de màster. Per tal garantir una bona cooperació, es van repartir les tasques.

L'estudiant de màster realitza la navegació del robot utilitzant la càmera RGB-D, sent capaç el robot de seguir a una distancia de seguretat a un objecte en moviment, reconeixent el color vermell de l'objecte i la forma d'aquest.

Un altre projectista de treball final de grau, s'ocupa de la implementació i integració de nous sensors com són el sensor de gasos de CO2 i O2, una càmera tèrmica, una càmera RGB-D usada per l'estudiant de màster i la eliminació dels dispositius obsolets que disposa el robot actualment.

Aquest treball final de grau s'ocupa de dotar de més autonomia al robot incorporant una nova placa-PC, adaptant el sistema ROS al robot per treballar en perfecte harmonia el projecte de navegació i el d'incorporació de nous sensors.

A més a més, dóna noves capacitats de comunicació per veu i de transmissió de dades, que permetrà no només comunicar-se amb el gos o altres equips de rescatament, si no que permetrà un contacte directe amb la víctima que es pugui trobar en l'escenari de rescat amb els experts dels equips de rescat.

També dotarà d'una fàcil interacció entre operari i robot amb possibilitat de comunicació remota al robot des de ordinadors externs i la fàcil instal·lació d'aquest sistema en nous robots.

3 SOFTWARE NECESSÀRI

L'objectiu principal és utilitzar el sistema ROS, una arquitectura software, dissenyada per sistemes robòtics, que permet la comunicació e integració de diferents dispositius i comença a ser usat a nivell mundial com un estàndard.

Aquest sistema va ser creat per la universitat Standford dels EEUU, amb la intenció d'alliberar el codi font i permetre que amb la col·laboració d'altres universitats o grups d'investigació es pugui fer créixer aquest projecte.

El projecte es va iniciar el 2007 i ha aconseguit ser acollit per milers d'usuaris d'arreu del món i diferents grups d'investigació que a poc a poc el van fent créixer aportant "packages" nous.

Robot Operating System (ROS) és una col·lecció de software per al desenvolupament de software robòtic. Bàsicament, ROS proporciona els serveis estàndards d'un sistema operatiu com el control de dispositius a baix nivell, l'abstracció de hardware, la comunicació entre processos i el manteniment de "packages".

Aquest sistema està basat en una arquitectura de "graphs" on el processament es du a terme en els nodes. Els nodes estan combinats entre ells amb una transmissió periòdica i constant de missatges. Aquests, són processos computacionals que poden rebre, enviar i multiplexar missatges de sensors, estats, actuadors, planificacions, etc.

Per exemple, un node controla la càmera RGB-D, un altre node controla les rodes dels motors, un altre controla la ruta de navegació, un altre proporciona visió gràfica d'un procés, etc. Tots aquests nodes queden identificats amb un "graph" intern de ROS que els diferencia de la resta del sistema. Per exemple, /kinec_node.

La comunicació entre nodes es fa mitjançant tòpics. Un tòpic és un sistema on diferents nodes li envien missatges periòdicament, els emmagatzema i altres nodes es subscriuen a aquest tòpic per poder llegir, rebre i tractar aquests missatges.

El tòpic, per tant, és un intermediari d'informació entre nodes. Un tòpic seria, per exemple, l'equivalent a un "hashtag" de Twitter on publicant a través d'aquest "hashtag" els usuaris poden seguir els missatges publicats. Aquest sistema permet estandaritzar la comunicació

interna entre els nodes i no dependre d'un mateix llenguatge de programació per a tots els nodes del robot.

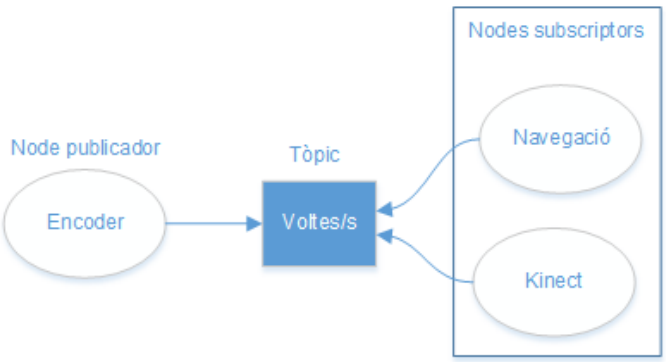

Figura 2. Comunicació per nodes i tòpics

ROS es complementa amb ROS-pkg o "packages" que són paquets de llibreries i codi sense compilar on estan definits els nodes i tòpics, aquest paquets serien petits programes i/o drivers. Els ROS-pkg són desenvolupats per usuaris i compartits normalment sota llicència BSD (Berkeley Software Distribution) de software lliure la qual permet un ús comercial i d'investigació.

ROS disposa d'una gran font d'informació, en una web en format wiki (http://wiki.ros.org/), on els desenvolupadors dels "packages" creen tutorials per tal d'ensenyar a utilitzar-los. Des de la versió de ROS Groovy els "packages" estan allotjats a la web GitHub (https://github.com/) de forma pública l'evolució es podrà seguir a través d'aquest portal.

GitHub és un espai web dedicat a emmagatzemar projectes d'aquestes característiques per a difondre'ls, de forma pública o privada, a diferents usuaris. És una "forja" de projectes com ho poden ser SourceForge, GForce o Google code. Els avantatges de GitHub són una pàgina web per a cada projecte, amb una wiki i gràfiques de les actualitzacions que aporten els desenvolupadors dels projectes, tenir seguidors del projecte i les interaccions socials que ells generen.

ROS va ser dissenyat inicialment per a treballar sobre la distribució de Ubuntu Linux. No obstant això, ROS s'està desenvolupant, en fase Beta i mantingut per usuaris, per poder-se fer servir sobre altres sistemes operatius, com són: Mac OSX, Windows 7/8, altres distribucions populars de Linux o altres sistemes operatius menys populars o derivats de BSD.

Tot i tenir una versió per a Windows, s'escull la versió de ROS per a Ubuntu, perquè és el sistema operatiu més estable per a ROS i l'únic sistema operatiu que li garanteix una bona compatibilitat.

Un altre motiu és que Ubuntu disposa de més material de codi compilat distribuït per internet al ser un sistema operatiu OpenSource. És un sistema operatiu més obert que Windows, cosa que permetrà modificar aspectes interns si fos necessari. Concretament s'ha escollit la versió Ubuntu 12.04LTS.

3.1 LINUX UBUNTU 12.04 LTS

La distribució Ubuntu deriva de la distribució Debian i és una de les més famoses per la seva senzillesa i per estar enfocat a un usuari final i no a un servidor. Ubuntu està recolzat per el kernel de Linux i el seu entorn gràfic pot ser escollit però per defecte ve amb Unity des de la versió 10.10 que va reemplaçar la versió GNOME desktop environment.

Linux va ser creat l'any 1991 per Linus Torvald amb la idea de fer un sistema operatiu UNIX de codi lliure. El sistema operatiu Unix va ser creat per l'empresa americana AT&T per facturar totes les trucades dels seus clients l'any 1969. Linus Torvald va agafar Minix, un sistema operatiu creat per ensenyar a alumnes d'informàtica les parts d'un sistema operatiu, per crear el kernel Linux. El kernel és un programa que gestiona el hardware de l'ordinador, decideix quin programa fa ús de quin recurs o dispositius i durant quant de temps.

Linux també és conegut com GNU/Linux per que quan Linus Torvald va llençar Linux, el projecte GNU ja portava molt de temps en marxa i oferia eines bàsiques com: un command-line, biblioteca C i un compilador, però aquest projecte tenia un nucli (un kernel anomenat Hurd) que tenia molts errors. Per tal d'omplir aquest forat que tenia el sistema operatiu de GNU es va fer servir el nucli de Linux i per això es coneix com a GNU/Linux. Així el nucli del sistema operatiu és Linux i la part fonamental d'interacció entre hardware i usuari són les eines del projecte GNU.

Així Ubuntu utilitza el kernel de Linux. El seu patrocinador es Canonical, una empresa britànica que ofereix el sistema de manera gratuïta i es finança mitjançant serveis vinculats al sistema operatiu. Ubuntu va néixer del codi base de Debian amb l'objectiu de fer una distribució fàcil d'utilitzar i entendre per usuaris finals.

Ubuntu disposa de 2 tipus de versions estables. Una versió LTS (Long Term Support) i una no LTS. La diferència entre les dues es que Canonical, l'empresa desenvolupadora d'aquesta popular distribució de Linux, ofereix ajuda tècnica i actualitzacions de seguretat durant més o menys temps. Des de la versió 13.04, 9 mesos per les versions no LTS i des de la versió 12.04LTS, 5 anys de suport per les versions LTS. Abans tenien un suport de 18 mesos i 3 anys respectivament.

Quan finalitza el suport per una versió d'Ubuntu s'eliminen els repositoris, dels que disposa aquella distribució, de tots els servidors, tant en els servidors principals, com en els servidors mirall repartits per tot el món. Això implica que no es podrà descarregar nous arxius per aquella versió.

Canonical llença cada 6 mesos una versió d'Ubuntu per escriptori. D'aquestes versions, cada 2 anys, una d'elles, és una versió LTS. És a dir, cada 4 versions llançades, una d'elles, és una versió LTS i les altres 3 no LTS. Fins que no es torna a llançar una versió LTS conviuen dues versions d'Ubuntu. La numeració de les versions no segueix un ordre de desenvolupament de codi. Els 1ers dígits indiquen l'any que va ser llançada la versió, i les següents dues xifres indiquen el mes. Per exemple, la versió 12.04 va ser llançada l'abril de l'any 2012.

Sabent això, s'escollirà una versió LTS per obtenir un sistema operatiu que estarà actualitzat durant més temps. D'altre manera, en menys d'un any, la versió del sistema operatiu estaria obsoleta. Des de l'abril, la versió més actual és la versió 14.04.

Però com que encara no ha sigut llançada cap versió compatible de ROS amb aquesta versió de Ubuntu es continuarà amb la versió 12.04LTS, quina encara disposarà de 3 anys de suport i actualitzacions.

Aproximadament, cada 8 mesos apareix una nova versió de ROS. Les versions intenten estar sincronitzades amb les versions d'Ubuntu LTS, donant suport d'una versió de ROS per les següents 3 versions posteriors d'Ubuntu. Per tant, ROS Groovy és compatible amb la versió Ubuntu Precis (12.04LTS), la Ubuntu Quantal Quetzal (12.0) i la Ubuntu Saucy salamander (13.04).

3.1.1 Intèrpret de comandes BASH

Una de les particularitats de tots els sistemes Linux és el gran ús que es fa amb les terminals. La terminal és un intèrpret de comandes, en anglès s'utilitza la paraula SHell (sh), fa d'interfície entre l'usuari i el sistema operatiu. El nom del Shell de Ubuntu és bash (acrònim de Bourne Again SHell).

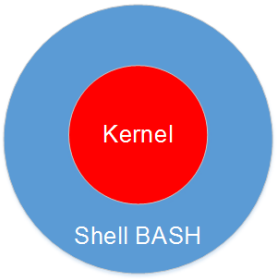

Figura 3. Nivells del sistema Linux

La terminal disposa d'un indicador de línia d'ordres (en anglès es coneix com prompt), si ets un usuari normal l'indicador es mosta amb el caràcter $ i si ets súper usuari (administrador) amb #. En Ubuntu tindria aquest aspecte: usuari@equip:directori_actual$. El caràcter ~ que es podria veure en algun moment com a part del directori és una mena d'abreviatura de la ruta /home/usuari.

Per accedir a una terminal es pot fer de dues maneres, la primera seria sortir de l'entorn gràfic i entrar en el mode text i una altre seria fent ús d'emuladors com Gnome-terminal. Un cop s'executa una terminal es llegeix l'arxiu .bashrc i es carreguen totes les variables d'entorn i configuracions (es fa un login).

Una variable que es genera en una terminal és local, no global, per fer-la global s'hauria de llistar en .bashrc. Això fa que un cop es tanqui la terminal tots els canvis no tinguin efectes en el sistema, tampoc, tots els canvis que es facin en el sistema, posteriorment després d'obrir la terminal, tampoc es vegin reflectits en la terminal oberta, a no ser que es cridi la comanda:

```
$sudo source ./.bashrc
```

La comanda obliga a tornar a carregar la configuració de bashrc, i la comanda sudo dona permisos d'administrador. S'ha de tenir en compte aquest aspecte, si es carregues una nova connexió, la terminal no veuria el canvi fins que es reobris la terminal o es recarregues la

sessió BASH. Per editar l'arxiu bash hem d'introduir en una terminal que un editor de text (en aquest cas gedit) obri i ens permeti editar l'arxiu.

```
$sudo gedit ./.bashrc
```

Des de la terminal podem controlar tot el sistema d'arxius i els processos executats o per executar de l'ordinador. Tot es realitza mitjançant comandes, una comanda pot tenir arguments o flags que la complementin i li indiquin paràmetres o funcions que ha de realitzar. Els arguments o flags van seguits de la comanda. La diferencia d'un flag amb un argument es que l'argument dóna valor a una variable normalment i un flag activa una funció o paràmetre que s'ha de tenir en compte.

Tots els sistemes Unix comparteixen algunes comandes però d'altres són diferents o no existeixen. Seguidament es mostra una taula amb les comandes de BASH més útils. Tots aquests paràmetres es poden veure amb la comanda "man", la qual ensenya un manual de cada funció, on també es mostren tots els diferents flags que es poden utilitzar.

man	Manual de cada comanda
ls	Llista de directori
cd	Cambi de directori
echo	Mostrar un text per pantalla
sudo	Donar permisos d'administrador
read	Per llegir una entrada (com podria ser un teclat)
chmod	Modificar permisos d'un arxiu o carpeta
>	Afegir text a un arxiu sobreescrivint
>>	Afegir text al final de l'arxiu sense modificar res més
sleep	Un delay en segons
egrep	Es un cercador de caràcters
cut	Permet dividir strings
ping	Permet comprovar una connexió
cat	Mostra el contingut d'un fitxer
less	Mostra el contingut de forma tabulada
rm	Elimina un arxiu
cp	Copia un arxiu
diff	Compara dos arxius
clear	Neteja la terminal
export	Assigna valor a una variable d'entorn
alias	Assigna un nom simbòlic a una comanda
bg	Mostrar processos en segon pla
killall	Permet finalitzar processos

Taula 1. Comandes de l'entorn BASH

No s'han inclòs totes les comandes ni molt menys sinó les que més s'han utilitzat en el projecte. Les comandes poden introduir-se una per una o es pot fer una seqüència, inclús que una depengui de l'altre o el resultat d'una afecti la següent. Per il·lustrar aquest aspecte es mostra una petita taula amb les diferents separacions.

;	un cop hagi finalitzat la comanda seguirà la següent, sense importar la anterior
&&	Un cop hagi finalitzat i no hagi donat cap error la comanda seguirà la següent
|	El resultat de la primera comanda serà una entrada de la següent
&	Al final de la comanda, farà que la comanda s'executi però en segon pla

Taula 2. Separadors de comandes

Aquests nexes de comandes permeten crear programes complexes amb relativa facilitat. I la opció "&" permet que una terminal no es quedi bloquejada quant executa un programa, ja que al quedar en segon pla la terminal queda "alliberada" pots seguir treballant des d'ella sense haver d'obrir-ne una de nova.

3.1.2 Generació dels scripts

L'entorn Ubuntu Linux permet la programació d'Scripts en BASH code, un llenguatge només compatible en shells BASH. Un Script és un programa que executa comandes i permet automatitzar processos. També es podria crear un script amb llenguatge python o amb SHell, però el codi BASH permet més compatibilitat amb el sistema.

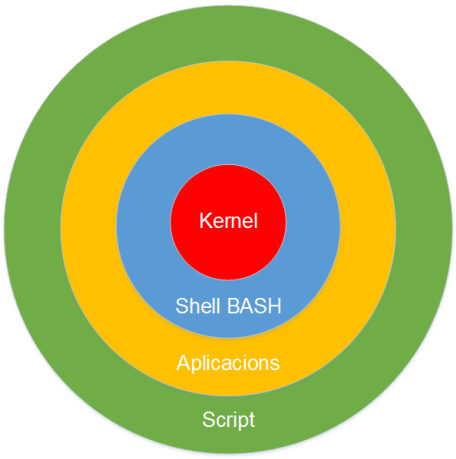

Figura 4. Nivells des d'on treballa un script

La programació d'scripts en BASH es molt completa i permet condicions lògiques. La semàntica del llenguatge BASH es bastant diferent de C. Per exemple, per comparar valors

numèrics s'utilitza "–eq", ja que un "=" compara strings només. El codi bash es bastant sensible a espais o caràcters, per tant s'ha de programar amb bastanta exactitud.

El codi BASH permet programar: if, while, for, casos, funcions i cerca (amb les comandes grep, awk o sort). Les variables no s'han de declarar, automàticament són reconegudes. Per facilitar el reconeixement de les variables dins del codi es solen posar en majúscules com a norma general.

Tot script en BASH ha d'incorporar en la primera línia la següent comanda per tal que el Shell reconegui que és un script.

```
#!/bin/bash
```

Per tal que l'arxiu sigui executable s'han de donar permisos d'execució a l'arxiu o no es podrà executar. Per aconseguir-ho només cal introduir a la terminal la següent comanda:

```
$chmod +x NOM_SCRIPT
```

Per al projecte es creen una sèrie d'scripts que ajudaran a controlar el robot i automatitzar molts de processos. En total s'han generat sis scripts: script_gen, inici, inici_remot, stream_audio, install_files, i sortir.

Per tal de fer més fàcil la creació d'scripts, ja que és una eina molt útil i que s'ha utilitzat molt en el projecte, s'ha creat un script per generar nous scripts. El generador d'scripts pot crear scripts per BASH, Shell, Python i Expect.

L'script crea un script, amb el nom que se li dona, a la carpeta /home amb permisos d'execució, afegeix a la primera línia de l'arxiu la línia necessària per que el Shell reconegui l'Script, sigui en el llenguatge que sigui, i obre el programa Geany (cal tenir-lo instal·lat) amb l'script creat per començar a treballar. El programa Geany es pot instal·lar fàcilment amb la comanda facilitada.

```
$sudo apt-get install geany
```

Usant la funció Alias es pot crear una comanda que executi directament l'script sinó caldria introduir:

```
$sh script_gen.sh
```

La funció Alias permet assignar a una paraula un string amb una comanda, fins i tot amb els seus arguments, només cal editar l'arxiu bashrc i afegir l'alias com:

```
alias scrip_gen="cd ~ && sh scrip_gen.sh"
```

L'script "script_gen" permet fàcilment crear un script i està preparat per què l'usuari pugui entrar de diferents maneres el nom de l'script o en cas de no recordar els passos ser guiat, pas a pas, per el mateix script, amb avisos per pantalla.

Es pot executar donant el nom i la extensió de l'arxiu, donant el nom i el llenguatge que s'utilitzarà, donant el nom complet amb l'extensió adjuntada, donant només el nom o executar simplement l'script.

Seguidament es mostren dos exemples de com s'executaria l'script, el primer donant el nom i l'extensió adjunta i el segon el nom i el llenguatge que s'utilitzarà.

```
$script_gen NOM_SCRIPT.sh
```

```
$script_gen NOM_SCRIPT python
```

En cas d'equivocar-se l'script detectarà l'error i a no ser que sigui molt greu, demanarà corregir-lo. L'script detecta si hi ha caràcters estranys o espais en blanc en el nom de l'arxiu nou i en cas afirmatiu demanarà reintroduir l'arxiu per no donar conflictes amb el sistema operatiu, ja que alguns caràcters especials es reserven per ús intern del sistema operatiu.

Aquest script és una eina molt útil que permetrà fer instal·lacions de tot el projecte a nous robots o noves actualitzacions a robots amb una sola comanda. Per realitzar la actualització del sistema creat en aquest projecte es podrà realitzar remotament a través de connexió SSH usant la comanda scp que pot ser usada per enviar arxius de l'ordinador local al remot o pot ser usada per rebre arxius de l'ordinador remot a l'ordinador local.

3.1.3 Inici

L'script inici s'encarrega d'iniciar tots els processos necessaris per poder controlar el robot amb teleoperació. Aquest script treballa amb col·laboració de l'script inici_remot, el qual està programat en expect. Expect és un codi d'esperes que permet automatitzar processos esperant una resposta del procés que s'indica. L'script inici remot només serveix com a pont per connectar-se amb el robot de manera remota.

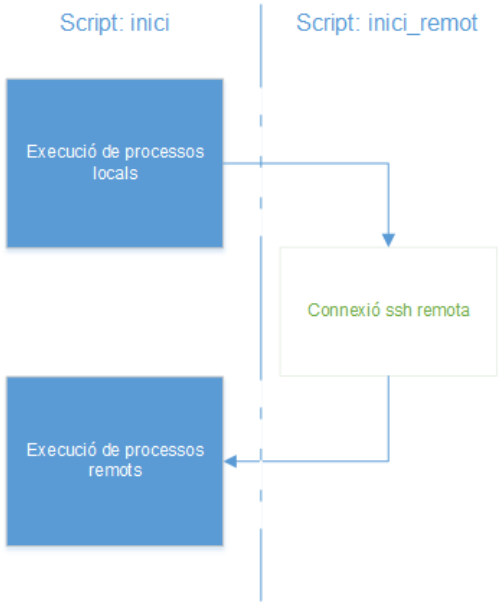

Figura 5. Cooperació d'scripts

La inicialització comença executant la comanda inici o start a la terminal. En aquest punt l'script té dos modes de funcionament, com a mestre o com esclau, aquest mode se li pot indicar a través del primer argument de la comanda inici, i es configurarà l'ordinador com a master o com slave (mestre o esclau).

Per tal de no generar una gran quantitat d'arxius separats s'ha intentat ajuntar i reutilitzar els scripts tantes vegades com es pugui. D'aquesta manera enlloc de fer una inicialització del robot i una per l'ordinador remot, s'ha ajuntat tot en un arxiu. Això també facilita en el moment de treballar en el sistema. Per tant, la configuració esclau està enfocada a ser la configuració per el robot i la configuració mestre per a l'ordinador remot.

Un cop inicialitzat l'script, si s'ha iniciat com a mestre, començarà a buscar la direcció IP de la connexió WiFi i configurarà amb aquesta IP les variables ROS_HOSTNAME i ROS_MASTER_URI de la sessió BASH.

Per fer això, s'utilitzarà la comanda expect. Amb això fem que aquestes variables d'entorn només variïn de valor en aquesta terminal. Si es volgués es podria editar l'arxiu bashrc i posar de forma permanent les direccions IP a aquestes variables. En cas de no detectar cap connexió d'internet, avisarà en forma d'error.

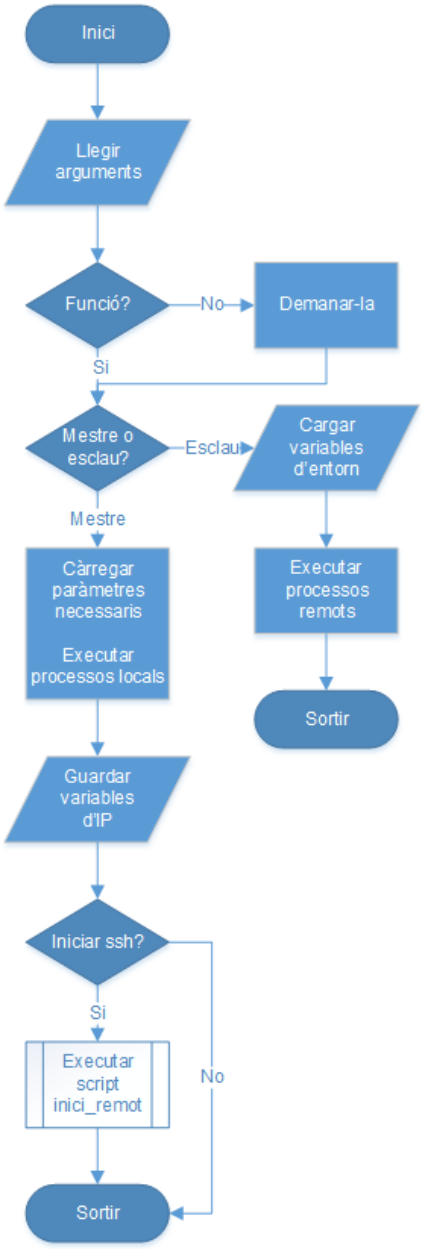

Figura 6. Diagrama de flux de l'script inici

Un cop tot està apunt, s'inicien en aquesta ordre la seqüencia de processos: roscore, els nodes del joystick (rosrun joy joy_node i rosrun joy2twist joy2twist.py), i el procés del camshift per començar. Finalment s'iniciarà, si es confirma, una connexió remota amb el robot mitjançant una connexió SSH (Secure SHell).

S'ha optat per identificar l'ordinador com a mestre, ja que el roscore s'inicia en l'ordinador de control i no en el robot, l'ordinador serà qui durà a terme el processament de càlcul de la imatge i el robot com a esclau per què serà qui enviarà tota la informació dels sensors.

Un cop es configura tot l'ordinador com a mestre s'executa i es configura l'script inici_remot amb les variables de IP de mestre i la IP de l'esclau. L'script usa un codi anomenat expect, aquest codi, permet fer una espera fins que un procés (cridat amb la comanda spawn) mostri per la terminal cert text, indicat amb la comanda expect. En aquest cas s'espera veure "d: " (quant es demana la contrasenya es mostra password: per pantalla).

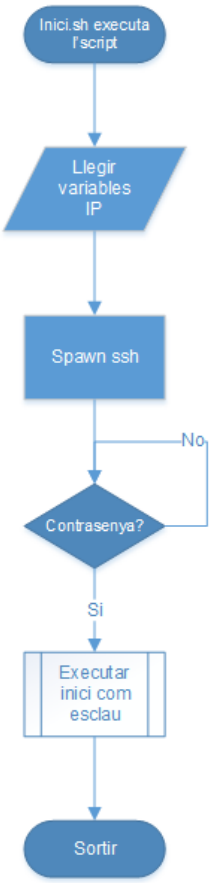

Figura 7. Diagrama de flux de l'script inici_remot

Un cop mostrat, s'enviarà la contrasenya i des de l'ordinador de control estarem connectats remotament al robot. S'executarà un altre cop l'script inici, però aquest cop s'executarà en el robot, no en l'ordinador de control i gràcies a la connexió SSH podrem visualitzar des de la terminal de l'ordinador de control tot el procés. Aquest cop s'executarà l'script inici indicant la configuració esclau s'enviaran les IP del mestre i de l'esclau per coincidir amb les del mestre.

S'ha de fer ús de dos scripts diferents per que els llenguatges usats són diferents, el codi BASH no permet aquest tipus d'esperes. Per altre banda, no s'ha programat tot amb expect per que el llenguatge expect no està preparat per codis complexes.

En el robot s'executaran els processos del node de la base de control, el node de la càmera RGB-D i càmera tèrmica i començarà a fer stream de l'àudio del robot.

3.1.4 Comunicació per SSH

La comunicació remota es fa a través de xarxes inal·làmbriques WiFi sense la necessitat de connectivitat a internet treballa sota el protocol TCP i el protocol SSH. SSH és el nom del protocol i del programa que l'implementa i serveix per permetre una comunicació remota entre dos equips a través d'una xarxa.

El protocol SSH s'utilitza moltíssim per controlar servidors de forma remota, ja que proveeix una connexió estable i segura entre dos equips. Permet iniciar sessions en servidors remots (login), executar comandes remotament, copiar arxius entre diferents hosts, executar aplicacions X11 remotament i realitzar túnels IP xifrats. SSH no només és un reemplaçament de Telnet, ftp, rlogin, rexec entre d'altres, en una sola aplicació, sinó que també permet assegurar, en tot moment, una comunicació segura (xifrada) entre client i servidor, sent el robot el servidor i l'ordinador de control el client.

SSH permet executar aplicacions gràfiques, és a dir, obrir una finestra a l'ordinador remot i veure-la a l'ordinador que ha iniciat la connexió. A diferencia de programes com Teamviewer o Logmein, SSH no permet una connexió remota entre ordinadors d'altres xarxes.

Per iniciar una connexió remota hem d'introduir el nom d'usuari i la direcció IP de l'ordinador al que ens connectarem, un cop introduïts els valors s'haurà d'introduir la contrasenya i veurem la ruta de la terminal varia i es converteix en la ruta de l'ordinador remot.

```
$ssh NOM_USUARI@DIRECCIÓ_IP
```

Per executar directament una comanda, es situaria la comanda després de la direcció i si es volgués executar de manera gràfica s'afegiria el flag –X abans del nom i la direcció. Aquest flag fa referència al sistema de finestres X, normalment es fa referència a la versió 11 d'aquest protocol (X11). El sistema de finestres és un software desenvolupat als anys 80 en el MIT per

dotar d'una interfície gràfica als sistemes Unix. Aquest protocol permet una interacció gràfica en xarxa entre un usuari i una o més computadores.

La comunicació entre finestres de les màquines es realitza mitjançant el protocol Xprotocol, que consisteix en una sèrie de bytes interpretats com comandes bàsiques per generar finestres, posicionar-les o controlar events. Els clients X accedeixen al Xprotocol utilitzant la llibreria Xlib que ajuda al programador a manipular el codi binari de l'Xprotocol. Però la decoració de la finestra no està inclosa en la llibreria. X no és un gestor de finestres i necessita d'un per controlar les finestres. En el cas d'Ubuntu fa servir Compiz com a gestor de finestres.

Tant el programa SSH, com el protocol X11, explicats venen instal·lats per defecte a Ubuntu 12.04. I no es requereix de cap instal·lació prèvia normalment. El protocol X11 està inclòs en el programa SSH.

A través de la connexió SSH s'aconsegueix activar i desactivar tots els processos al robot. Quant s'executa inicialment l'script inici, mitjançant la connexió SSH es criden els processos de la càmera RGB-D i executa el node de ROS per controlar la placa de control. S'ha de tenir en compte que els processos s'han de cridar en segon pla o quant finalitzi aquest procés la sessió SSH finalitzarà. Al mateix succeeix quant s'obra una nova terminal o una pestanya dins la terminal, la nova terminal no te la sessió SSH activa. No obstant, si obre una nova terminal a través d'un script si que manté la sessió activa en la nova terminal.

3.1.5 Streaming d'àudio

El robot, com a robot teleoperat remotament, necessita d'un sistema que li permeti poder-se comunicar verbalment i escoltar el seu entorn. Tot i tenir un paquet de ROS, anomenat audio_common, que permet un stream d'àudio a través de TCP/IP, no s'utilitzarà de moment, per que no ofereix una comunicació estable. Després d'una estona d'estar enviant àudio a internet, el paquet Audio_common comença a perdre paquets de dades i va acumulant errors que a la llarga provoquen que s'acabi perdent la comunicació. Un altre inconvenient d'aquest paquet era que no permet una comunicació bidireccional, només permet escoltar o parlar.

Com a solució s'ha optat utilitzar una llibreria de conversió d'àudio que prové del projecte FFmpeg, el qual aporta les llibreries libavcodec i libavformat, la primera permet codificar diferents arxius d'àudio o vídeo i la última transcodificar els arxius a altres formats. Però Ubuntu incorpora la seva versió que han creat ells que prové del projecte FFmpeg. Es van

separar del projecte degut a conflictes d'interessos i Ubuntu va crear Libav. I poc a poc estan intentant que els usuaris de distribucions Debian es passin al nou sistema.

Ubuntu a dia d'avui utilitza una versió pròpia del projecte FFmpeg que és Libav i ha reanomenat la comanda ffmpeg per la comanda avconv per distanciar-se més del projecte FFmpeg. Durant aquest període de transició encara es pot utilitzar la comanda ffmpeg per cridar la llibreria Libav però sortirà un avís dient que aquesta comanda no serà mantinguda en pròximes versions i demana utilitzar avconv millor. Sabent això s'ha escollit la comanda avconv per defecte. El funcionament de ffmpeg i avconv és idèntic, els paràmetres que pots modificar són els mateixos i té l'avantatge de no haver de fer cap instal·lació prèvia.

Avconv	Comanda per convertir arxius d'àudio vídeo
Avplay	Per reproduir arxius d'àudio o vídeo
-f	Indica el format de l'arxiu d'entrada o sortida
-i	Nom de l'arxiu d'entrada
-acodec	Codificació de l'àudio
-ab	Bitrate (bit/s)
-ar	Freqüència de mostreig del senyal
-aq	Qualitat de la sortida de l'àudio
-re	Llegir entrada al frame rate natiu
rtp	Real time protocol
-map	Designa una o mes streams d'entrades o fitxers
-ac	Designa el numero de canals

Taula 4. Opcions avconv

Però aquesta llibreria a part de permetre codificar arxius de vídeo o àudio permet transmetre'ls per xarxes a través de protocols RTP o RTSP. Amb aquesta llibreria aconseguim agafar directament el senyal de l'entrada del micròfon mostrejar-la a la freqüència desitjada, codificar-la a mp3 i enviar-la per stream a la velocitat seleccionada amb el protocol RTP (real time protocol) a través d'una direcció IP.

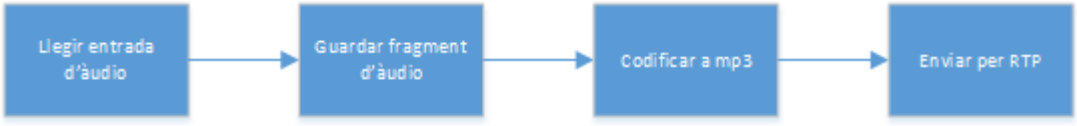

Figura 8. Stream d'àudio amb Libav

Des d'un altre dispositiu connectat a la mateixa xarxa es pot reproduir aquest stream, ja sigui a través de programes com VLC o amb l'ús de la llibreria Libav i la comanda avplay. Per

aconseguir realitzar tot això només caldrà programar les comandes de la llibreria de la manera desitjada.

```
$avconv -f alsa -i hw:0,0 -acodec libmp3lame -ab 32k -ar 16000 -re -f rtp
rtp://234.5.0.5:1234
$avplay -i rtp://234.5.0.5:1234
```

Per tant en un ordinador s'executaria la primera comanda mostrada seguidament i en un altre (el que escolta) la segona comanda. La primera comanda s'ha programat per que agafi el driver de so, en aquest cas el driver de so ALSA (Advanced Linux Sound Architecture, Arquitectura de so avançada per Linux), i seleccioni el dispositiu de gravació/reproducció, enumerat per aquest driver com a "0", de la targeta de so, enumerada per aquest driver internament com a 0 (Podria donar-se el cas que un ordinador disposes de més d'una tarja de so). Per saber de quins dispositius disposa Ubuntu per reproduir o enregistrar àudio, es poden utilitzar les comandes:

```
$arecord -l
$aplay -l
```

El següent paràmetre de la comanda és el tipus de codificació, amb la llibrera libmp3lame es podrà codificar a mp3 i podrem configurar aquesta codificació amb els paràmetres –ab i –ar. Segons ISO-11172-3 que regula la transmissió del mp3 s'escollirà un bitrate de 32kbit/s.

S'escull aquest paràmetre per què és la velocitat més baixa que permet aquesta llibreria, s'escull la més baixa possible per optimitzar al màxim l'ample de banda que requereix aquest streaming, es dóna per suposat que no es necessita una qualitat d'àudio òptima i la qualitat a 48kbits/s és suficientment acceptable, i una freqüència de mostreig de 16kHz per què s'enviarà només veu.

Tot aquest contingut s'envia a través del protocol RTP. Aquest protocol es utilitzat en transmissions d'informació en temps real i és un dels punts claus de les trucades VoIP (per exemple l'Skype). Utilitza la capa UDP per enviar les dades i va de la mà amb els protocols RTCP i RTSP. El protocol RTP proporciona confidencialitat, autentificació de missatges i protecció de reenviaments de fluxos d'àudio i vídeo de forma constant.

Tot aquest àudio s'envia a la direcció 234.5.0.5 per el port 1234. Les direccions IP, d'acord amb el RFC 3171, de la 224.0.0.0 fins la 239.255.255.255 estan reservades al multicast,

aquestes direccions estan pensades per grups de receptors, l'emissor envia un sol datagrama i el router s'encarregarà de fer copies i enviar-les als receptors.

En el laboratori només es disposen d'11Mb/s i a aquesta baixa velocitat s'ha aconseguit que tot funcionés amb fluïdesa. En el cas de l'àudio només hi ha un retard d'1,3 segons. En un principi la baixa velocitat suposa una gran pèrdua de paquets i s'ha optimitzat al màxim possible la transmissió de d'àudio ocupant el mínim de banda ampla sense perdre gran qualitat d'àudio. S'han realitzat diferents proves d'àudio en diferents bitrates i freqüències de mostreig per aconseguir una transmissió òptima i fluida.

El sistema, per tant, envia l'àudio gravat per el micròfon a internet i el router s'encarrega de enviar l'àudio al seu destinatari.

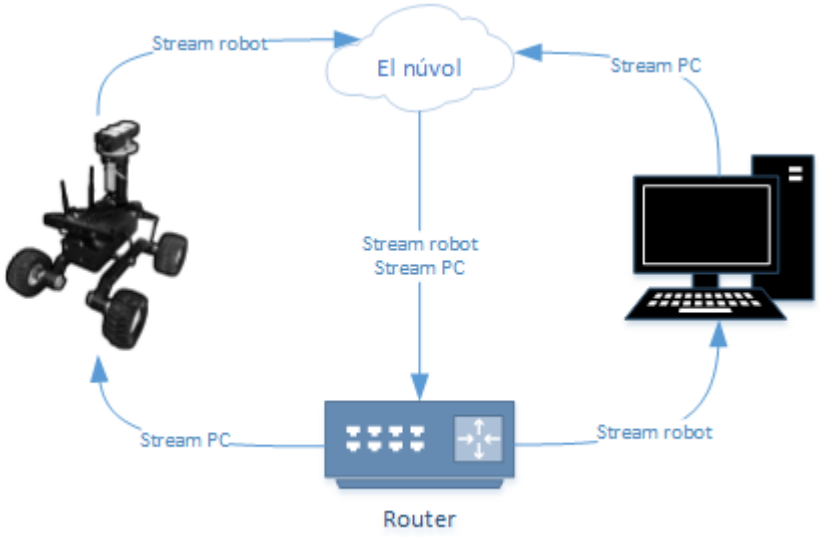

Figura 9. Integració d'àudio al projecte

Aquest sistema permetrà que el robot disposi d'unes "orelles" per escoltar i adonar-se de què està passant al seu voltant, en un radi de 360º, no només frontalment com permeten les càmeres instal·lades. I dota el robot de la capacitat de poder transmetre l'àudio rebut remotament.

Per tal de fer més fàcil aquestes tasques i automatitzar aquest , s'ha creat un script que executa l'stream d'àudio tant en el robot com en l'ordinador remot. Una part de l'stream d'àudio es executada en la configuració d'esclau de l'script inici, en l'inici s'executa en el robot la

comanda per començar a grabar l'àudio del micròfon del robot i enviar-lo a una direcció IP d'internet.

En aquest nou script anomenat stream_audio s'inicia l'escolta de l'àudio del robot des de la IP 234.5.1.5:1234, s'inicia la gravació de l'àudio de l'ordinador i s'executa l'inici_remot per iniciar una connexió SSH amb el robot. Es reutilitza aquest arxiu per tal de no crear una gran quantitat d'arxius.

L'inici_remot funciona de manera semblant a quant es cridat per l'script inici, però en aquest cas, s'executa la variable case amb un valor diferent, amb valor 1, així l'script reconeix que ha sigut cridat per l'stream_audio i no per l'script inici i executa la escolta des del robot a l'stream d'àudio de l'ordinador a la IP 234.5.5.5:1234.

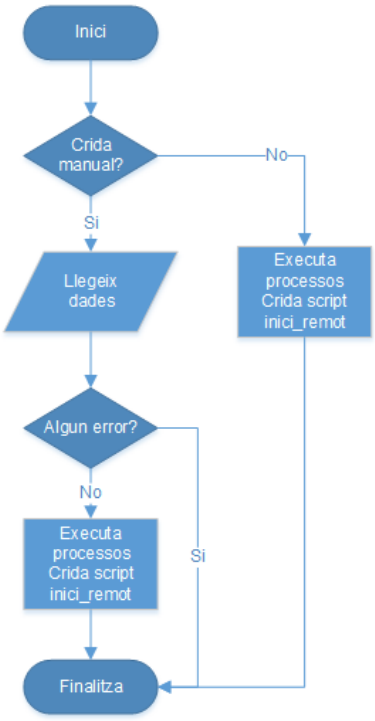

Figura 10. Diagrama de flux de l'script stream_audio

S'ha previst que l'equip de control no estigui emetent àudio tota l'estona així que s'ha creat una tecla (F12) que permet silenciar o reactivar el micròfon de l'ordinador. Així evitem saturar l'ample de banda amb missatges que no són necessaris i evitar que s'escolti tot el que es digui des de la zona de control, el robot no ha de convertir-se en un lloro. Quant es prem la tecla F12 s'executa la comanda amixer set Capture toggle.

Per executar l'script stream_audio i iniciar la comunicació de veu només cal introduir a la terminal la següent comanda:

```
$stream_audio
```

L'script disposa d'un sistema que en cas de no detectar cap connexió a internet o no reconèixer correctament la IP de l'ordinador, finalitza sense iniciar l'stream d'àudio i retorna un error per la terminal. Aquest script, a més a més, s'ha deixat preparat per si s'automatitza el procés d'executar l'stream d'àudio.

Al ser un streaming multicasting, permetrà que qualsevol usuari que conegui la direcció IP i pugui reproduir rtp, com per exemple, des de el mòbil amb aplicacions com VLC per mòbils, podrà escoltar aquest stream. Això permetria que equips de rescats que estiguessin per la zona i no a la zona de control puguin escoltar a la víctima en temps real.

3.2 ROS GROOVY

ROS Groovy Galapagos és la sisena distribució llançada el 31 de desembre de 2012. En aquesta versió l'equip de desenvolupament s'ha centrat en el nucli de la infraestructura de ROS per intentar aconseguir fer-lo més modular i més fàcil per l'usuari.

Les principals millores que aporta respecte anteriors versions és la migració massiva de codi a GitHub, un nou sistema de compilació catkin, la eliminació d'stacks, un nou sistema de llançament de nous paquets (bloom), la millora de la interfície gràfica rqt, el dynamic_reconfigure s'ha separat de rqt-rviz i ha rebut una millora d'interfície gràfica molt gran, junt amb una nova API disponible.

Pluginlib i class_loader (llibreries per la creació de plugins en c++) han sigut reescrites i permeten retro compatibilitat, amb noves millores de seguretat.

El canvi més important es la migració de rosbuild a catkin dels arxius anteriors, però com que la versió de ROS Groovy encara permet la compilació en rosbuild de moment es mantindrà catkin en segon pla i es seguirà utilitzant rosbuild per no complicar les coses i no posar problemes on no són necessaris.

La instal·lació i configuració de ROS Groovy és molt senzilla primerament s'ha de d'actualitzar l'arxiu sources.list activar la clau i actualitzar l'índex d'Ubuntu i començar la instal·lació de ROS Groovy Galapagos. Un cop finalitzada s'inicia carregar les dependències.

```
$sudo sh –c 'echo "deb http://packages.ros.org/ros/ubuntu precise main" >
/etc/apt/sources.list.d/ros-latest.list' && wget http://packages.ros.org/ros.key -O
- | sudo apt-key add - && sudo apt-get Update && sudo apt-get install ros-groovy-
desktop-full && sudo rosdep init && rosdep Update
```

En cas de ser una nova instal·lació s'haurà de modificar l'arxiu de la sessió de BASHi recarregar-la per aplicar els canvis i si hi ha més d'una distribució instal·lada carregar l'entorn ROS amb el setup.bash

```
$echo "source /opt/ros/groovy/setup.bash" >> ~/.bashrc && source ~/.bashrc &&
source /opt/ros/groovy/setup.bash
$sudo apt-get install python-rosinstall
```

Un cop tot instal·lat s'ha de procedir a crear una zona de treball per el ROS que no és més que una carpeta on s'instal·laran els paquets, dependències i arxius que s'aniran instal·lant. En el robot aquesta carpeta s'anomena ros_workspace. Es pot descobrir fàcilment la ubicació de la carpeta amb la variable d'entorn ROS_PACKAGE_PATH.

El sistema ROS aporta moltes comandes per facilitar l'ús a l'usuari, moltes d'elles tenen semblança amb les normalment usades en UNIX. Les més útils es llisten seguidament.

Roscd	Cd a la ruta origen del workspace
rospack	Permet gestionar els paquets
rosnode	Permet gestionar els nodes
rostopic	Permet gestionar els tòpics
rosrun	Executa un node
roslaunch	Executa una sèrie de processo
roscore	Executa el sistema ROS

Taula 5. Comandes bàsiques de ROS

La comanda per iniciar el sistema ROS és la comanda roscore, és la primera comanda que s'hauria d'executar. Les altres comandes només funcionen completament amb el roscore actiu, sense el roscore actiu podrien no funcionar. En algunes ocasions un roslaunch pot activar el roscore ell mateix si no està activat prèviament, però els roslaunch que s'utilitzen en aquest projecte, cap d'ells activa el roscore, encara que no estigui actiu.

3.2.1 Nodes

Des de l'script inici es criden un total de 36 nodes dels quals 21 són nodes per controlar la càmera RGB-D, 4 per controlar la teleoperació, 2 per la visualització de la imatge de les càmeres, 1 node per controlar la càmera tèrmica, 2 nodes per controlar i monitoritzar la placa de control, 2 per iniciar streaming d'àudio en el robot i 1 últim per els càlculs de seguir un objecte.

En l'altre script (stream_audio) es criden els nodes que falten, un total de 3 nodes més, dos per l'àudio i un per el seguidor. En total el sistema ROS ha de gestionar 39 nodes. En qualsevol moment es pot utilitzar la comanda "rosnode list" per visualitzar els nodes que disposa el robot.

A part d'aquest nodes, s'ha creat un de nou on es publiquen les dades enviades per la placa de control a través d'internet i estan publicades en diferents tòpics. El codi ha estat programat amb C++, però fàcilment podria ser construït en python. El codi generat per fer funcionar el node és bastant senzill, utilitza la llibreria std_msg.h i simplement es subscriu als tòpics indicats, a una freqüència d'un segon o mig segon, i crida la funció del missatge per mostrar les dades. En total, es subscriu a 12 nodes: temps, error, velocitat roda 1, 2, 3 i 4, control de rodes 1, 2, 3 i 4, intensitats rodes 1, 2, 3 i 4, nivell de bateries, voltatge de les bateries, posició X posició Y, posició Z i angle.

El codi del node es pot dividir en dues parts, les funcions que guarden la informació i la mostren per pantalla i la part on les variables es subscriuen a tòpics i n'extreuen la informació. Seguidament es mostra un exemple que mostra el voltatge de la bateria del robot.

```
Void BatCallback(const std_msgs::Float64::ConstPtr& msg2)
{
  ROS_INFO("Bateria: [%f]V", msg2->data);
}
int main(int argc, char **argv)
{
  ros::init(argc, argv, "listener");
  ros::NodeHandle n;
  ros::Subscriber sub12 = n.subscribe("Bateria", 1000, BatCallback);
  ros::spin();
  27etorn 0;
}
```

Les comandes ROS_INFO són similars a fer un printf per consola, ros::Subscriber serveix per subscriure's a un node, sub12 indica la variable que agafarà els valors del tòpic, i n.subscriure serveix per llegir el tòpic i entre parèntesis es posen els paràmetres desitjats, el nom del tòpic, la freqüència de lectura i la funció que s'executarà un cop llegit el tòpic.

La funció ros::spin crear un loop infinit i repeteix el codi fins que detecta un Ctrl-c premut (Ctrl-c en una terminal unix s'utilitza per finalitzar processos). Una vegada el detecta indicarà al mestre (roscore) que l'apagui.

En l'exemple anterior, primer s'ha definit la funció que es cridarà quant es llegeixi el tòpic i després s'ha subscrit al tòpic indicant la variable que agafarà el valor de la variable del tòpic, el tòpic que volem, la freqüència en que es llegirà el tòpic i la funció que s'ha de cridar després.

3.2.2 Tòpics

Els tòpics es poden veure fent ús de "rostopic list" i es pot veure la freqüència en que es publica alguna cosa nova o literalment la informació exposada afegint a la comanda rostopic els arguments echo o hz.

El sistema ROS que s'ha adaptat, gestiona un total de 203 tòpics, dels quals la gran majoria són de la càmera RGB-D, concretament 170 tòpics. Aquest fet es degut a que s'envien les imatges en diferents formats, per exemple, envia la imatge raw en format theora i en format compressed.

S'utilitzen tan el format compressed, com el format theora, per tant, no podem reduir el nombre tòpics actual, ja que també són necessaris.

En aquest projecte s'utilitzen les dades en compressed per tenir una bona recepció. En el format compressed aconseguim una visualització més fluida que no es té en el format theora, el format theora provoca retrassos (d'entre 3 a 5 segons) en les imatges rebudes i dificulta la navegació teleoperada remotament.

La resta de tòpics donen informació de la resta de nodes. En el cas de la placa de control tenim 21 tòpics on podem extreure informació de les intensitats que circulen pels motors, velocitats i posicions del robot, 2 nodes per la càmera tèrmica (un per la configuració actual

de la càmera i l'altre per el vídeo), 5 per el procés de camshift, 1 per el rosout (tòpic intern de ROS que sempre ha d'estar actiu) un últim per el joystick.

En el cas del node listener agafem valors dels 21 tòpics de la placa de control per mostrar-los per pantalla. Els tòpics poden ser llegits des de qualsevol ordinador connectat a la xarxa i amb la direcció del rosmaster editada per actuar com esclau del mestre.

4 HARDWARE DEL ROBOT BIGBOT

El robot BIGBOT disposa d'un hardware pre-instal·lat que no compleix amb els requisits que es demanen en els objectius d'aquest projecte i necessita una actualització i millora.

Aquesta millora s'ha realitzat afegint una nova placa-PC i actualitzant components antics per altres de nous. Amb la actualització aplicada es pot separar i referenciar en dos blocs el hardware del robot, com a Placa-PC i Base de control. El primer bloc ja estava pre-instalat i se l'hi ha aplicat actualitzacions i el segon bloc és completament nou.

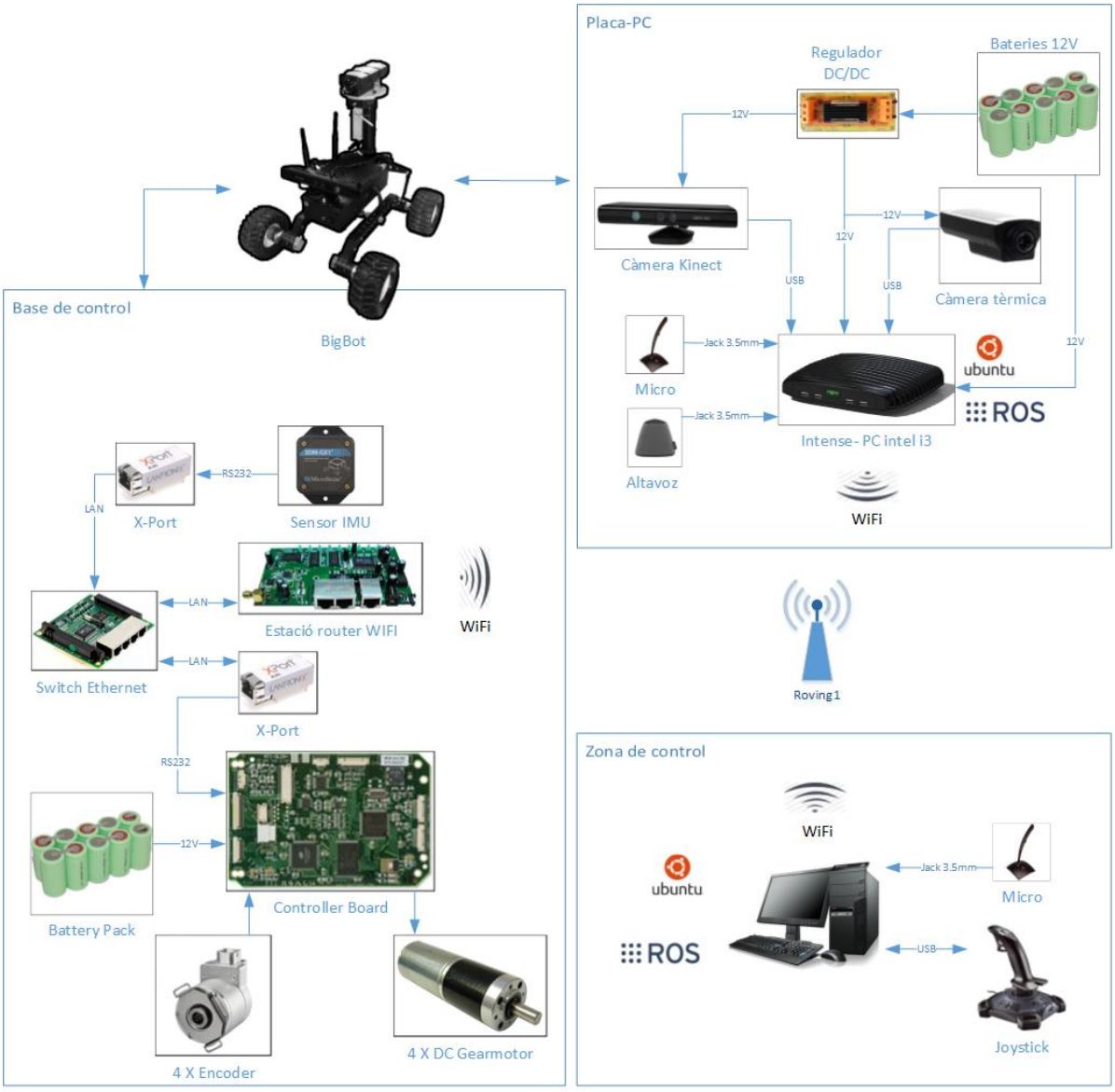

Figura 11. Esquema elements Bigbot

En el bloc corresponent a la base de control, el robot disposa d'uns encoders i uns motors de corrent continua per cada roda, una placa controladora amb un dsPIC que gestiona diferents PID's de regulació de velocitat, encoders i la etapa de potència per moure les rodes. Aquesta etapa de potència està formada per 4 ponts en H que permeten un control en paral·lel de les 4 rodes simultàniament, oferint diferents velocitats i sentits de gir de les rodes. A més a més, aquest microcontrolador envia una trama de dades a través de WIFI amb valors de consums, velocitats i posicions aconseguits dels sensors IMU i dels encoders, la descripció més detallada d'aquesta trama es pot veure en l'annex B.

En un principi, el robot, disposava d'una càmera de visió i un Làser range finder. Però aquests dispositius s'han retirat del robot, i s'ha afegit una càmera RGB-D. D'aquesta manera, el robot segueix disposant de les prestacions que oferien aquests sensors d'una manera més eficient i amb més característiques afegides.

El robot també disposa d'un sensor IMU, que permet saber la posició X i posició Y respecte un punt inicial. El dispositiu IMU ofereix connectivitat RS-232, però per problemes de compatibilitat dels drivers amb el sistema operatiu Ubuntu LTS12.04 no podem fer una connexió directe amb la placa-PC, la connexió la fem a través d'ethernet connectant l'IMU a un XPORT del switch d'ethernet. D'aquesta manera, les dades del sensor IMU, s'envien per internet a través de la trama de la placa de control i queden recopilades en els tòpics corresponents a posició, X, Y, Z i l'angle de posició.

En el bloc corresponent a Placa-PC i alimentat per un grup de bateries diferent a l'altre bloc. Aquest bloc està constantment comunicat amb el primer bloc mitjançant la connexió WiFi roving1 (la connexió del laboratori). Té com a nucli principal la placa-PC, intense PC IPC C2340V-WB-FM4U, un ordinador que disposa d'un microprocessador d'intel d'arquitectura x64 i 4GB de memòria RAM.

En aquesta placa mitjançant USB es connecta la càmera RGB-D i mitjançant connexió ethernet la càmera tèrmica, alimentades les dues càmeres per el mateix grup de bateries que la placa-PC. S'han incorporat un micròfon per proves d'àudio amb una connexió per jack 3.5mm i un altaveu amb connexió per jack de 3.5mm o bluetooth i disposa d'una bateria incorporada recarregable per USB.

Els dos grups de bateries que alimenten els dos blocs són de 12V i 10A. El grup de bateries encarregat d'alimentar el bloc Placa-PC no l'alimenta directament, prèviament un regulador

DC/DC estabilitzar el voltatge per donar un senyal més net i en casos en que la bateria estigués a nivells baixos, seguiria aportant el mateix voltatge.

4.1 Placa-PC

Per tal de poder adaptar i donar autonomia al robot, s'ha buscat una placa-PC de poc consum però suficient potència de càlcul. Un equilibri entre aquests dos requeriments per no consumir les bateries de manera molt ràpida, ni quedar-se curt en velocitat de procés, potència de càlcul i fluïdesa de funcionament. Un equilibri difícil, ja que els processadors ARM ofereixen un consum molt baix però una potència de càlcul una mica limitada, en canvi els processadors d'ordinadors d'escriptori ofereixen una gran potència de càlcul però uns nivells de consum molt elevats.

Com que ROS es un sistema que funciona sobre un sistema operatiu i s'ha decidit que el sistema operatiu serà Linux. Necessita d'un processador, amb una arquitectura compatible amb aquest sistema. S'ha descartat directament utilitzar o dissenyar les plaques amb cor de FPGA, de microcontrolador o microprocessadors ARM, ja que el sistema operatiu Ubuntu no soporta aquestes configuracions. Per tant, hi ha la necessitat d'escollir una placa-PC amb un processador d'intel o amd, ja sigui d'arquitectura x86 o x64, és a dir, processadors de 32 bits o 64 bits que si són compatibles amb les versios d'Ubuntu d'escriptori.

Una altre dificultat alhora d'escollir una placa on allotjar el sistema Ubuntu i elsistema ROS són les dimensions reduïdes del robot BigBot, no es pot incorporar qualsevol placa-PC. Ha de ser una placa-PC amb tots els components per funcionar però en unes dimensions que permeti incorporar-se adequadament al robot. Un ordinador portàtil es massa gran i la pantalla es innecessària. Els portàtils petits coneguts amb el nom de netbooks, tenen processadors de molt poc consum però no disposen d'un processador prou potent, i segueix apareixent el mateix problema de disposar d'una pantalla innecessària. Una placa mare d'ordinador d'escriptori amb tots els seus components segueix ocupant molt d'espai i fa impossible la seva integració òptima en el robot.

En resum, la plaça-PC ha de ser de mida reduida, de baix consum o autonomia pròpia, una potència de processament alta i que permeti executar el sistema Ubuntu Linux de forma completa i disposi d'un disc dur intern.

En el mercat estan apareixen uns ordinadors petits anomenats HTPC fàcils d'ensamblar i molt compactes, la casa intense PC ofereix un ordinador HTPC industrial en format molt reduït. Les mides són: 190x160x40mm. Pot incorporar diferents processadors a escollir entre i3, i5 o i7. Aquest processadors incorporen una targeta gràfica integrada (de baix rendiment). El sistema d'emmagatzematge és amb disc durs SSD a escollir entre diferents capacitats.

Figura 12. Intense PC cara davantera i posterior.

S'ha escollit un processador i3 de 2ona generació, el i3-3217UE, per què comparant-lo amb els altres processadors ofertats per aquesta empresa no hi ha molta diferencia. El successor d'aquest, el de 3era generació, camparant les puntuacions dels benchmarks no s'aprecien grans diferències (una puntuació de 2200 per el de 2ona generació). S'ha escollit un disc dur de 120 GB, suficients per incorporar un sistema operatiu (màxim dos, si és volgués disposar de Windows convivint amb l'Ubuntu) i per als arxius que es vulguin posar.

Mides	190x160x40mm
Potència de consum	17 Watts
Voltatge d'alimentació	De 10 a 16V
CPU	Intel core i3 3217UE 64bits 1.6GHz
Targeta gràfica	HD Graphics 4000 Intel
Memòria	2x2Gb DDR3-1066, 64-bits (ampliable a 16Gb)
Ports USB	6x USB 2.0 a 480Mbit/s i 2x USB 3.0 a 5Gbit/s
Port sèrie	Mini RS232
Ethernet	2x RJ45
WLAN WIFI	802.11b/g/n, dues antenes, 150Mbit/s
Disc dur	120GB (S-ATA)
Àudio	Entrada stereo 7.1 S/PDIF, jack 3,5m;m; sortida mono

Taula 6. Especificacions de la placa-PC

Un dels avantatges d'aquesta placa-PC és que incorpora 2 connexions RJ45 i 6 USB, la qual cosa permetrà ampliar futurament el robot i no quedar-se petit de connectivitats i haver d'ampliar ports amb accessoris. Tant la memòria RAM com de disc dur es poden ampliar

futurament fins els valors fixats en la taula 6. En el cas del disc dur es podrà connectar un disc dur d'estat sòlid (SSD) a través de S-ATA.

Aquesta placa-PC s'alimenta a 12V i té un consum de 17W. És la part més interessant d'aquesta placa, que ofereix unes prestacions molt bones amb un consum elèctric molt baix, i no descarregarà les bateries ràpidament aconseguint així l'equilibri buscat.

4.2 Placa de control i motors

La placa de control ha sigut dissenyada per la UdG amb l'objectiu de controlar els motors dels robots de la flota MATE a través de ponts en H com a drivers de control. La placa té com a nucli un microcontrolador PIC, concretament el model dsPIC 33FJ256MC.

Arquitectura	16 bit – 40MHz
Memòria de programa	256 kB
Memòria RAM	30720 Bytes
Encapsulat	100 pin TQFP (85 i/o pins)
Control d'execució	PBOR, POR, WDT
Canals de memòria DMA	8
Entrades analògiques	24 x 12-bit @ 500 (ksps) 2-A/D
Comunicacions digitals	2 x UART, 2 x SPI, 2 x ECAN, 2 x I2C
Canals per control de motors per PWM	8 (16 bit resolution)
Canals Input-Capture	8
Timers	9 x 16 bit, 4 x 32 bit
Interfície QEI	1

Taula 7. Especificacions microcontrolador

En aquest xip se l'hi volca un programa en C que s'encarregarà de totes les tasques de control de baix nivell. A part del control dels quatre motors en llaç tancat, es controlen altres prestacions com la lectura dels polsos de l'encoder (mitjançant comptadors d'alta velocitat), es realitza la lectura de les intensitats consumides per cada motor a través de cel·les hall i es llegeix el nivell de les bateries constantment ja que des d'aquesta placa de control (que disposa de fonts commutades de 3,3V i 5V) s'alimenten altres dispositius.

La comunicació entre el PIC i el PC es fa amb trames de bytes variables sobre el protocol TCP/IP, que poden ser consultades en l'annex B. Aquesta comunicació pot ser a través de cable ethernet o a través de connectivitat inal·làmbrica. S'ha escollit la connectivitat inal·làmbrica per evitar conflictes amb la connexió de la càmera tèrmica que es cablejada i deixar lliure d'aquesta manera el port ethernet de la placa-PC per futurs usos.

La placa de control, amb a una estació router WIFI, te accés a xarxes inal·làmbriques que permeten una comunicació amb la placa-PC i accés al sistema ROS. Aquesta estació router té una IP fixe (192.160.1.8) i proporciona a la placa de control una IP fixe (192.168.1.18).

La placa-PC envia i rep dades de la placa de control, d'aquesta manera el sistema ROS pot controlar els motors i el consums detectats per les cel·les Halls i la velocitat que els encoders llegeixen. Aquesta comunicació es pot veure en l'annex B.

Tensió nominal	12 V
Parell nominal	3,4 N·m
Velocitat en buit	71 rpm
Corrent en buit	330 mA
Corrent màxim (eix bloquejat)	16,9 A
Diàmetre de l'eix	6,4 mm

Taula 8. Especificacions dels motors

La placa de control disposa de quatre ponts en H per tal de subministrar la potència demanada a cada motor de forma independent. Degut a les dimensions del robot, els motors necessiten un par motor elevat per poder moure'l. Per aconseguir augmentar el parell els motors disposen d'uns reductors amb engranatges per reduir-ne la velocitat de l'ordre de 65,5 a 1 obtenint un parell nominal de 3,4Nm.

Els encòders, encarregats de llegir les velocitats de les 4 rodes disposen de dos canals per llegir dos sentis de gir. Disposen d'una resolució de 500 CPR (cicles per revolució), és a dir, per cada volta l'encoder envia 500 polsos, És una resolució adequada per a les velocitats que aconsegueix el robot que són de l'ordre de 30cm/s.

La placa de control pot conèixer el consum elèctric del motor de cada roda i aplicar diferents controls de velocitat PI. Amb aquesta informació es pot preveure el temps d'autonomia de les bateries i informació sobre el terreny on està operant el robot.

4.3 Unitat de mesura inercial

Una unitat de mesura inercial o IMU és un dispositiu electrònic que mesura i permet saber la velocitat lineal i angular, l'acceleració lineal i angular (forces gravitacionals) i la seva posició utilitzant la combinació d'acceleròmetres i giroscopis.

El giroscopi és un sensor que permet determinar la posició angular del robot respecte el camp magnètic del terra. També permet obtenir l'acceleració i la velocitat angular en els tres eixos. L'acceleròmetre és un sensor que mesura les forces gravitatòries i sabent aquestes es coneix l'acceleració lineal.

L'IMU està connectat a través d'ethernet a la placa de control i la placa de control amb aquesta informació determina la posició X, Y i Z. Un dels grans desavantatges d'utilitzar aquest sensor per la navegació és l'error acumulatiu que es va generant amb el temps. El sistema va fent estimacions de la posició i això deriva en petits errors. El sistema calcula la velocitat i la direcció i fa un estimació de la posició actual, és a dir, si durant 5 segons el robot s'ha dirigit al nord a 20cm/s estima la posició actual i dedueix que està 100cm al nord respecte abans. Seria el mateix cas que si una persona tingués els ulls embenats i s'hagués estat movent en diferents direccions, si se li pregunta després a on es troba, ell només sabrà fer una estimació.

Per corregir aquest error la majoria d'aquests sistemes incorporen d'altres sensors per eliminar l'error acumulatiu com sensors de gravetat, GPS, sensors de velocitat externs, sistemes baromètrics per corregir l'altitud o brúixoles magnètiques. El robot disposa d'altres sensors que li permeten corregir aquests errors com són els encòders (sensors de velocitat) i la placa-PC que disposa de GPS si està connectat a internet.

4.4 Càmera RGB-D

Una càmera RGB-D, és una càmera RGB (permet grabar en color) que incorpora informació de profunditat (Depth) de les imatges que està gravant. Per aconseguir-ho fa ús d'un sensor d'infrarojos amb un rang de 4 metres (el mateix que el làser range finder) en un radi de 58.5º en horitzontal i 45.6º en vertical.

Per controlar la càmera RGB-D ROS utilitza els drivers del paquet camshift i openni, aquests publiquen tota la informació en tòpics, on posteriorment usant la llibreria imatge_view es poden visualitzar les imatges de la càmera en diferents formats.

La càmera RGB-D Kinect, va ser desenvolupada per Microsoft l'any 2010 com a accessori de la consola XBOX 360. Però degut al potencial de combinar una càmera RGB i sensors de profunditat per projecció d'infrarojos s'ha convertit en un dels sensors visuals més utilitzats en el món de la robòtica.

La càmera Kinect serà la usada en el robot al tenir molta informació disponible i paquets de ROS operatius que permeten controlar la càmera.

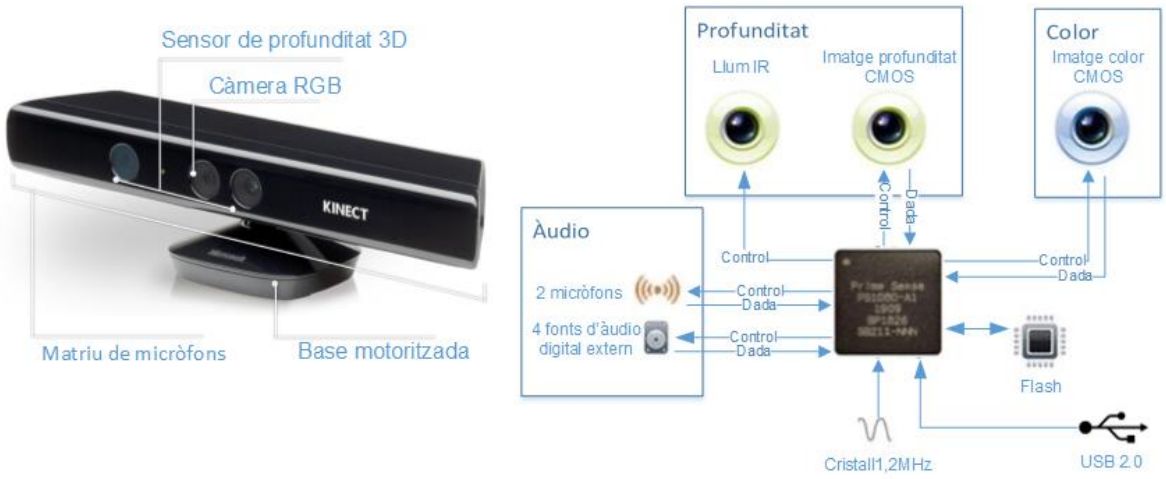

Figura 13. Esquema càmera Kinect 1.0

Per determinar la profunditat de la imatge, la càmera RGB-D incorpora un sensor de projecció per punts d'infraroig. Aquesta projecció emet un seguit de punts a l'espai formant un patró de molts punts per trobar la profunditat de d'una figura respecte a un fons del mateix color. A partir d'aquest càlculs amb l'ajuda de la càmera RGB s'utilitzen càlculs trigonomètrics per la detecció dels objectes i calcular la profunditat de l'objecte respecte el seu fons.

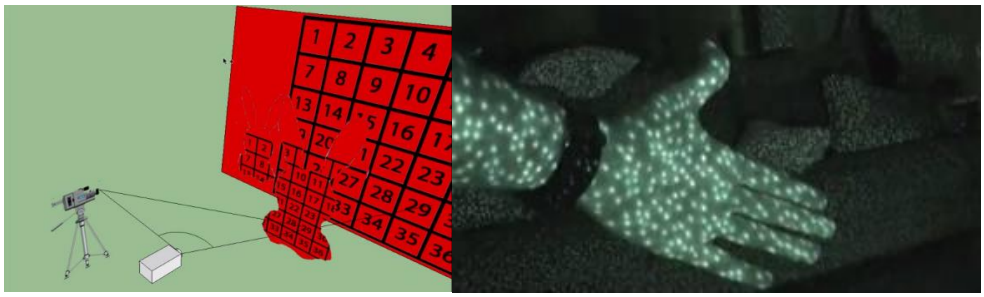

Figura 14. Càlcul trigonomètric de la Kinect

La càmera Kinect identifica cada punt com una posició d'una matriu projectada en un fons. Quant un objecte està entre el fons i la càmera mitjançant les dues càmeres de profunditat (de distància fixe i coneguda) es tracen dues línies que formaran un triangle respecte les càmeres i l'objecte (els catets), amb l'ajuda de relacions trigonomètriques es reconeix la distancia (la hipotenusa). Cada posició de la matriu està numerada per la càmera d'infrarojos, d'aquesta manera els objectes poden ser diferenciats sense importar el color. Amb l'ajuda d'aquest punts també es pot aconseguir determinar l'angle (la posició) de l'objecte.

El patró de punts es casi aleatori però com que la distancia entre càmeres es sempre exacta i fixe, un cop es reflecteix la llum dels sensors d'infrarojos se sap la distància de la càmera a l'objecte. Aquesta tecnologia te l'inconvenient que en espais molt il·luminats la projecció de punts queda difusa i perd exactitud.

Les resolucions de les càmeres es poden escollir són diferents entre la de color i les de profunditat. En el cas de la de color es pot escollir entre 80x60, 320x264, 640x480 i 1280x960 píxels i pels sensors de profunditat les mateixes exceptuant la de 1280x960 píxels però en aquest projecte només s'escolliran resolucions de 640x480 per aconseguir una bona qualitat d'imatge sense necessitar molt ample de banda.

Totes les imatges que s'agafen dels tòpics de la càmera RGB-D i es mostren en pantalla es comprimeixen en format compressed, enlloc de RAW, amb un buffer de 10MB per aconseguir una fluïdesa més òptima d'imatge.

4.5 Autonomia del robot Bigbot

L'autonomia actual del robot és d'uns 70min, usant 2 grups de bateries de 10 cel·les cada un. D'aquesta manera s'aconsegueix donar 10A·h i 12V per cada grup de bateries, les cel·les de les bateries són de Níquel-Metall Hidrur de mida estàndard (LR20) de 10000mA·h i 1,2V col·locades en sèrie aconseguint els 12V.

Aquestes bateries presenten una taxa d'autodescàrrega d'un 30% mensual, més elevat que les bateries de NiCd que només tenen una taxa del 20%. Per això per consums continus s'utilitzen les de Níquel-Metall Hidrur i per usos de llargs períodes (com a comandes a distància, llums d'emergència...) s'utilitzen les de NiCd. De moment les bateries de Níquel-Metall Hidrur són usades en la gran majoria de vehicles de propulsió elèctrica.

Un dels grups de bateries s'utilitza per alimentar la placa de control, la placa que proporciona WIFI a la placa de control, els quatre motors amb es seus respectius encòders i cel·les hall i el sensor IMU.

L'altre grup de bateries alimenta la placa-PC, la càmera RGB-D i la càmera tèrmica. Aquest grup de bateries està connectat a un regulador de voltatge per regular el nivell de tensió proporcionat a la placa-PC a un nivell estable i fixe.

La tensió de les bateries es llegeix mitjançant un conversor AD del dsPIC de la placa de control. Aquest valor es transmet a ROS i es pot visualitzar en tot moment a través del seu tòpic.

Les bateries permeten ser recarregades amb carregadors de bateries de 12V convencionals a través d'un jack de DC. El temps promig de càrrega de les bateries és d'unes 6 hores. El temps de vida útil d'aquestes bateries oscil·la entre els 1000 cicles de carga segons fabricant, un cop sobrepassat les bateries podrien perdre la seva capacitat d'emmagatzematge.

5 RESULTATS

Quant executem en una terminal "inici" s'inicia l'script que executarà el sistema ROS amb tots els nodes, tòpics i programes necessaris per el funcionament del BigBot com a robot seguidor i teleoperat.

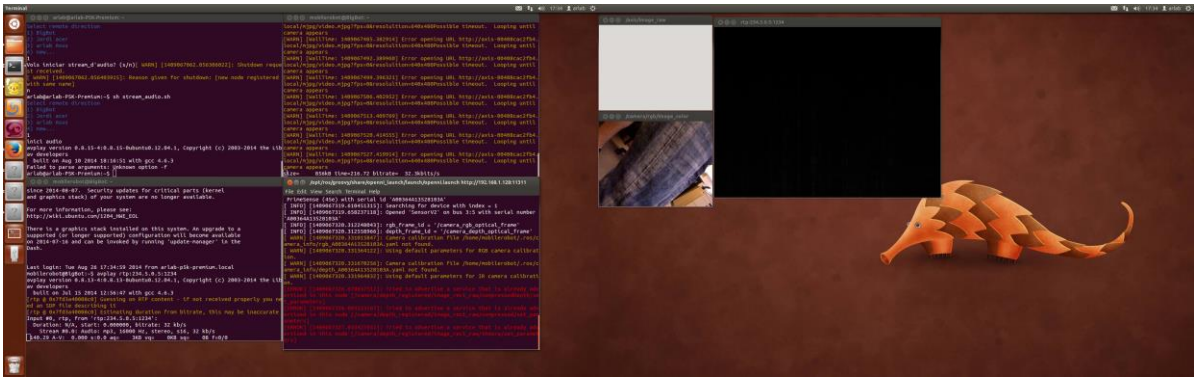

Figura 15. Escriptori a doble pantalla

Si tot ha anat bé, el robot podrà ser teleoperat amb el joystick. I quant s'indiqui en la pantalla del camshift quina figura i color s'ha de seguir, l'ordinador remot començarà a fer els càlculs de trajectòries per ser processats posteriorment per la navegació del robot. Tot aquest procés està comunicat a través dels tòpics del sistema ROS. El robot envia les imatges de la càmera RGB-D a través de tòpics i l'ordinador de control tracta aquestes imatges i el robot rep aquesta informació que després tracta i calcula la nova trajectòria.

També amb l'script inici queda oberta una finestra que mostra les imatges de la càmera tèrmica, aquestes imatges no poden ser modificades, es a dir, no pots ampliar el rang de zoom ni disminuir-lo i tampoc pots canviar els colors ni la escala de grisos.

Figura 16. Imatge càmera tèrmica

S'observarà que quedaran obertes dues finestres on es mostra la informació de la càmera RGB-D. Una es la de la camshift, fàcilment identificable per contenir un requadre seguint la figura demanada i l'altre només amb la imatge de la càmera.

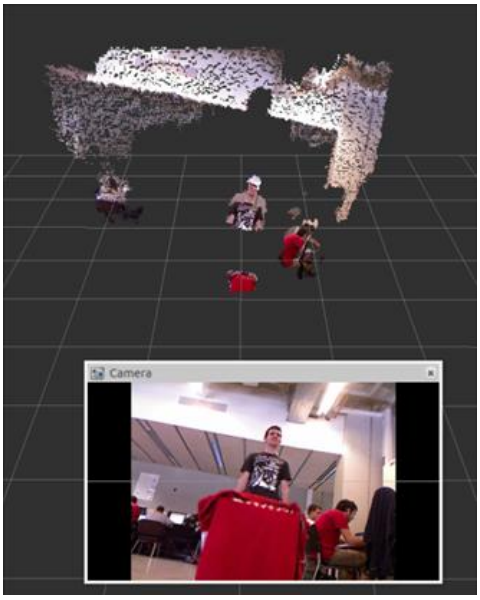

Figura 17. Visualització de Camshift

La diferencia entre elles es que la última el format del vídeo és "compressed" un format que permet una molt bona qualitat de vídeo amb molt poc ample de banda. La imatge d'aquesta última es veu molt més fluida i permet una navegació per teleoperació més còmode.

Figura 18. Imatge de càmera RGB-D

La imatge de la càmera RGB-D a diferencia de la càmera tèrmica, permet ser modificada, tant en mida de buffer, mida màxim de les imatges i permet escollir quines dades es volen mostrar,

ja siguin imatges en blanc i negre, imatges de profunditat o imatges en color RGB. Tots aquest diferents tipus d'imatge es poden veure en diferents formats, oferint alguns millor qualitat i altres oferint millor fluïdesa.

Aquests es un dels motius de tenir el de tenir dues finestres obertes i no només una. La camshift al necessita calcular colors, necessita millor qualitat (el mateix paquet camshift no dona la possibilitat d'escollir un altre format) però ofereix imatges amb un retard d'entre 3 i 5 segons. Aquest retard en navegació provoquen molts problemes, per tant s'ha optat per obrir una altre finestra amb una imatge més fluida que té un retard de milisegons.

Amb aquest script també s'inicia una part de la comunicació d'àudio, el robot comença a retransmetre, però ningú l'escolta fins que algú es connecti a aquella IP o s'executi l'script stream_audio. Un cop iniciat la comunicació d'àudio ja es completa entre dos o més equips.

L'àudio, com ja s'ha explicat anteriorment, es retransmet en mp3 per aconseguir una fluïdesa i qualitat acceptables. Amb aquest sistema s'ha aconseguit mantenir una comunicació oberta durant més de 24 hores seguides. En temps real es pot visualitzar el senyal rebut, sigui visualitzant les ones de so o l'espectròmetre de freqüències.

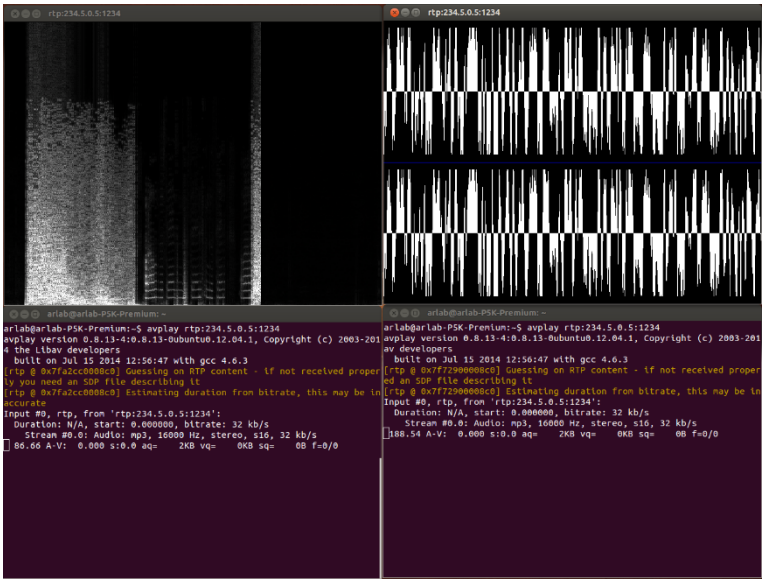

Figura 19. Espectròmetre i senyal de so

El retard que ofereix aquest sistema és d'1 segon o menys si hi ha bona cobertura de WiFi. Es pot visualitzar en la figura superior que fins que no s'ha pres F12 no s'ha començat a sentir àudio.

6 RESUM DEL PRESSUPOST

El pressupost total per dur a terme aquest projecte inclou les hores de treball i tot el material necessari per el complet funcionament del robot Bigbot.

El pressupost total ascendeix a cinc mil sis-cents vint-i-cinc euros amb quaranta-tres cèntims, IVA no inclòs.

7 CONCLUSIONS

Una vegada realitzada la implementació al robot, es pot observar que les especificacions s'han assolit completament.

Per començar, s'ha trobat una placa-PC que disposa de tots els requeriments mínims demanats i disposa, a més a més, d'altres funcionalitats, com possibilitat d'incorporació de nous sensors, ús total del sistema operatiu Linux, tots els avantatges de l'ús de processadors d'arquitectura x64, gran capacitat d'emmgatzament, etc.

Segonament, s'ha incorporat el sistema ROS en aquesta placa-PC sense cap disminució de prestacions i permet una perfecte comunicació entre el hardware antic del robot i altres robots o ordinadors de control que es puguin arribar a connectar.

Tercerament, s'ha creat un petit programa-menú que permet a un usuari iniciar tots els processos de comandament i comunicació del robot i monitoritzar l'estat del robot.

Finalment, s'ha aconseguit incorporar un sistema de comunicació entre el robot i un altre robot o ordinador de control, des de qualsevol dispositiu (robot o ordinador remot) es pot enviar veu i el robot la podrà reproduir i des del mateix dispositiu es podrà rebre veu procedent del micròfon del robot. Inclús s'ha incorporat d'un mecanisme que silencia els micròfons i es podria fer una comunicació semblant a la "push-to-talk".

A més, s'ha pogut conservar feina ja realitza anteriorment d'altres projectes realitzats paral·lelament amb aquest, amb les mínimes modificacions, que són la incorporació de nous sensors i la navegació del robot.

Caldria destacar les dificultats que han anat apareixent al llarg del desenvolupament del projecte, ja que el sistema Linux és un sistema operatiu amb molts anys de millores continues i molta gent darrera fent créixer aquest projecte poc a poc, cosa que fa que en poc temps apareixen moltes millores i d'altres quedin obsoletes. Tot i poder disposar de molta informació referent a l'ús i programació del sistema, s'ha de saber triar entre la informació recent i la obsoleta. A data d'agost de 2014 ha aparegut una nova versió LTS d'Ubuntu, però no s'ha utilitzat al no disposar un sistema ROS compatible amb aquesta nova versió.

El mateix ha succeït amb el sistema ROS, es disposa de noves versions de ROS però no s'ha utilitzat la ultima versió per què degut a un retràs de llançament, no està disponible la última versió.

Resta assenyalar el treball futur. Aquesta implementació només és la continuació natural del projecte, el següent pas seria la millora de la navegació del robot, dotant al robot la capacitat d'esquivar obstacles i incorporar nous sensors que millorin la seva tasca.

La conclusió final és que al ser un projecte en continu desenvolupament no es pot donar per acabat i es poden realitzar futures millores pel que fa al robot i a la seva navegació i control.

Jordi Hortal Garí
Graduat en Enginyeria Electrònica Industrial i Automàtica

Girona, 18 d'agost de 2014

8 RELACIÓ DE DOCUMENTS

Aquest projecte consta de cinc documents independents. Aquests són la memòria, els plànols, el plec de condicions, l'estat d'amidaments i el pressupost.

9 BIBLIOGRAFIA

Dreamsyssoft, script per SHELL BASH, (http://www.dreamsyssoft.com/unix-shell-scripting/, 2 de juny de 2014)

MARTINEZ, AARON i FERNANDEZ, ENRIQUE. Learning ROS for Robotics Programming. Packt publishing, 1a Edició. Birmingham (2013)

Libav, documentació de Libav, (https://libav.org/, 15 de juny de 2014)

McGraw-Hill, TCP/IP: Arquitectura, Protocolos e Implementacion, Madrid, Espanya, 2004

OpenNI, OpenNI Org., (http://www.openni.org/, 15 de març de 2014)

ROS, Ros, (http://www.ros.org/wiki/, 23 de març de 2014)

GOEBEL, R. PATRICK. ROS by exemple Vol.1 Groovy. PI robot production, 1a edició. Aarhus (27 d'agost de 2013)

StackOverflow, codi BASH, (http://stackoverflow.com/, 22 d'abril de 2014)

UdG, Flota De Robots De Rescat, Girona, Espanya, 2006

UdG, Resum de Hardware, Girona, Espanya, 2006

10 GLOSARI

ALSA: Advanced Linux Sound Architecture

API: Application Programming Interface

BASH: Born Again SHell

GPS: Global Positioning System

HTPC: Home Theater Personal Computer

IMU: Inertial Measurement Unit

LAN: Local Area Network

LTS: Long Term Support

PIC: Peripheral Interface Controller

ROS: Robotic Operational System

RTP: Real Time Protocol

RTSP: Real Time Streaming Protocol

SSH: Secure SHell

TCP/IP: Transmision Control Protocol/ Internet Protocol

VoIP: Voice over Internet Protocol

WiFi: Wireless Fidelity

A CODI DISPOSITIUS

En aquest Annex s'incorporaran tots els codis de programa que s'han fet servir en aquest projecte o han sigut modificats per adaptar el codi al projecte.

A.1 NODE MONITORITZACIÓ

```
#include "ros/ros.h"
#include "std_msgs/String.h"
#include "std_msgs/Int8.h"
#include "std_msgs/Int16.h"
#include "std_msgs/Float64.h"
#include "std_msgs/Float32.h"

#define SLOW 1000
#define FAST 500

/**
 *  !FUNCIONS DELS SUBSCRIPTORS!
 */
void tempsCallback(const std_msgs::Int8::ConstPtr& msg2)
{
  ROS_INFO("temps: [%d]", msg2->data);
}

void ErrorCallback(const std_msgs::Int8::ConstPtr& msg2)
{
  ROS_INFO("Error: [%d]", msg2->data);
}
//Velocitats
void Vel_FLCallback(const std_msgs::Int8::ConstPtr& msg2)
{
  ROS_INFO("Velocitat Roda davant-esquerra: [%d]", msg2->data);
}
void Vel_FRCallback(const std_msgs::Int8::ConstPtr& msg2)
{
  ROS_INFO("Velocitat Roda davant-dreta: [%d]", msg2->data);
}

void Vel_BLCallback(const std_msgs::Int8::ConstPtr& msg2)
{
  ROS_INFO("Velocitat Roda darrera-esquerra: [%d]", msg2->data);
}
```

```
void Vel_BRCallback(const std_msgs::Int8::ConstPtr& msg2)
{
  ROS_INFO("Velocitat Roda darrera-dreta: [%d]", msg2->data);
}
//Senyals de control
void Cont_FLCallback(const std_msgs::Int8::ConstPtr& msg2)
{
  ROS_INFO("Control Roda davant-esquerra: [%d]", msg2->data);
}
void Cont_FRCallback(const std_msgs::Int8::ConstPtr& msg2)
{
  ROS_INFO("Control Roda davant-dreta: [%d]", msg2->data);
}

void Cont_BLCallback(const std_msgs::Int8::ConstPtr& msg2)
{
  ROS_INFO("Control Roda darrera-esquerra: [%d]", msg2->data);
}

void Cont_BRCallback(const std_msgs::Int8::ConstPtr& msg2)
{
  ROS_INFO("Control Roda darrera-dreta: [%d]", msg2->data);
}
//Intensitats
void Int_FLCallback(const std_msgs::Float64::ConstPtr& msg2)
{
  ROS_INFO("Intensitat Roda davant-esquerra: [%f]", msg2->data);
}

void Int_FRCallback(const std_msgs::Float64::ConstPtr& msg2)
{
  ROS_INFO("Intensitat Roda davant-dreta: [%f]", msg2->data);
}

void Int_BLCallback(const std_msgs::Float64::ConstPtr& msg2)
{
  ROS_INFO("Intensitat Roda darrera-esquerra: [%f]", msg2->data);
}

void Int_BRCallback(const std_msgs::Float64::ConstPtr& msg2)
{
  ROS_INFO("Intensitat Roda darrera-esquerra: [%f]", msg2->data);
}

//Nivell de bateria
void BatCallback(const std_msgs::Float64::ConstPtr& msg2)
```

```
{
  ROS_INFO("Bateria: [%f]V", msg2->data);
}

void VccCallback(const std_msgs::Float64::ConstPtr& msg2)
{
  ROS_INFO("Vcc: [%f]V", msg2->data);
}

//Posicio
void PosXCallback(const std_msgs::Float32::ConstPtr& msg2)
{
  ROS_INFO("PosX: [%f]m", msg2->data);
}

void PosYCallback(const std_msgs::Float32::ConstPtr& msg2)
{
  ROS_INFO("PosY: [%f]m", msg2->data);
}
void ThetaCallback(const std_msgs::Float32::ConstPtr& msg2)
{
  ROS_INFO("Theta: [%f]°", msg2->data);
}

msg2->data

int main(int argc, char **argv)
{
  ros::init(argc, argv, "listener");

  ros::NodeHandle n;

  ros::Subscriber sub2 = n.subscribe("Temps", SLOW, tempsCallback);
  ros::Subscriber sub3 = n.subscribe("Error", SLOW, ErrorCallback);
  ros::Subscriber sub4 = n.subscribe("Velocitat_FL", FAST, Vel_FLCallback);
  ros::Subscriber sub5 = n.subscribe("Velocitat_FR", FAST, Vel_FRCallback);
  ros::Subscriber sub6 = n.subscribe("Velocitat_BL", FAST, Vel_BLCallback);
  ros::Subscriber sub7 = n.subscribe("Velocitat_BR", FAST, Vel_BRCallback);
  ros::Subscriber sub8 = n.subscribe("Intensitat_FL", SLOW, Int_FLCallback);
  ros::Subscriber sub9 = n.subscribe("Intensitat_FR", SLOW, Int_FRCallback);
  ros::Subscriber sub10 = n.subscribe("Intensitat_BL", SLOW, Int_BLCallback);
  ros::Subscriber sub11 = n.subscribe("Intensitat_BR", SLOW, Int_BRCallback);
  ros::Subscriber sub12 = n.subscribe("Bateria", SLOW, BatCallback);
  ros::Subscriber sub13 = n.subscribe("Vcc", SLOW, VccCallback);
  ros::Subscriber sub14 = n.subscribe("Control_FL", SLOW, Cont_FLCallback);
  ros::Subscriber sub15 = n.subscribe("Control_FR", SLOW, Cont_FRCallback);
```

```
  ros::Subscriber sub16 = n.subscribe("Control_BL", SLOW, Cont_BLCallback);
  ros::Subscriber sub17 = n.subscribe("Control_BR", SLOW, Cont_BRCallback);
  ros::Subscriber sub18 = n.subscribe("Pos_X", FAST, PosXCallback);
  ros::Subscriber sub19 = n.subscribe("Pos_Y", FAST, PosYCallback);
  ros::Subscriber sub20 = n.subscribe("Theta", FAST, ThetaCallback);

  ros::spin();

  return 0;
}
```

A.2 CODI STREAM_AUDIO

```
#!/bin/bash
#ENCODING: UTF-8
#Created by Jordi Hortal
if [ $# -eq 0 ]; then
            echo "$(tput setaf 4)Select remote direction"
            echo "1) BigBot"
            echo "2) Jordi acer"
            echo "3) arlab Asus"
            echo "4) new...$(tput sgr 0)"
            read NUM

            case $NUM in
                  1)    USER="mobilerobot"
                        HOST="192.168.1.101"
                        ERR=false
                        ;;
                  2)    USER="jordi"
                        HOST="192.168.1.146"
                        ERR=false
                        ;;
                  3)    USER="arlab"
                        HOST="192.168.1.100"
                        ERR=false
                        ;;
                  4)    read -p "Introduce IP: " HOST
                        read -p "Introduce name of the user: " USER
                        ERR=false
                        ;;
                  *)    echo "$(tput setaf 1)ERR: wrong select$(tput sgr 0)"
            esac
```

```
	USER_HOST="$USER@$HOST"

	MESTRE=$(ifconfig wlan0 | grep 'inet addr:' | grep -v '127.0.0.1' | cut -d: -
f2 | awk '{ print $1}')
	if [ "$MESTRE" = "" ]; then
		MESTRE=$(ifconfig wlan0 | grep 'inet:' | grep -v '127.0.0.1' | cut -
d: -f2 | awk '{ print $1}')
		if [ "$MESTRE" = "" ]; then
		echo "$(tput setaf 1)ERR: No internet conection detected$(tput sgr 0)"
		exit 1
		fi
	fi
	#depenent del pc pot ser inet o inet addr

elif [ $# -eq 3 ]; then
	MASTER="$1"
	HOST="$2"
	USER_HOST="$3"
else
	echo "$(tput setaf 1)Incorrect parameters$(tput sgr 0)"
	exit 1
fi
#es deixa preparat per si es crida des de un altre script

avplay rtp:234.5.0.5:1234&
echo "inici audio"
sleep 2
gnome-terminal -e avconv -f alsa -i hw:0,0 -acodec libmp3lame -ab 32k -ar 14500 -
re -f rtp rtp://234.5.1.5:1234&
sleep 2
gnome-terminal -e ./inici_remot.exp\ $USER_HOST\ $MESTRE\ $HOST\ 1
exit 0
```

A.3 CODI INICI

```
#!/bin/bash
#ENCODING: UTF-8
#Created by Jordi Hortal v1.2
AUTO=false
if [ $# -eq 0 ]; then
	read -p "Start as: master or slave? " SCRIPT
elif [ $# -eq 1 ]; then
	SCRIPT="$1"
elif [ $# -eq 3 ]; then
```

```
	SCRIPT="$1"
	MASTER="$2"
	SLAVE="$3"
	AUTO=true
else
	echo "$(tput setaf 1)Incorrect parameters$(tput sgr 0)"
	exit 1
fi

echo "Creating configuration as $SCRIPT"
if [ "$SCRIPT" = "master" ]; then
	#read -p "ip of the PC or master: " MESTRE
	#MESTRE=192.168.1.100
	MESTRE=$(ifconfig wlan0 | grep 'inet addr:' | grep -v '127.0.0.1' | cut -d: -
f2 | awk '{ print $1}')
	if [ "$MESTRE" = "" ]; then
		MESTRE=$(ifconfig wlan0 | grep 'inet:' | grep -v '127.0.0.1' | cut -
d: -f2 | awk '{ print $1}')
		if [ "$MESTRE" = "" ]; then
		echo "$(tput setaf 1)ERR: No internet conection detected$(tput sgr 0)"
		exit 1
		fi
	fi
	#depenent del pc pot ser inet o inet addr

	echo "ip of the PC or master: $MESTRE"
	export ROS_HOSTNAME=$MESTRE
	export ROS_MASTER_URI=http://$MESTRE:11311
	sleep 1

	roscore&
	sleep 2
	#rosparam set joy_node/dev "/dev/input/js1"
	#Si dona error en el joynode llavors has de mirar en quin js# s'ha posat
	rosrun joy joy_node&
	rosrun joy2twist joy2twist.py&
	sleep 1

	read -
p "$(tput setaf 4)Do you want to connect to a robot remotely (yes/no): $(tput sgr 0
) " SSH
	ERR=true
	if [ $SSH = "s" ] || [ $SSH = "si" ] || [ $SSH = "SI" ] || [ $SSH = "S" ] ||
[ $SSH = "yes" ] || [ $SSH = "y" ]; then
		echo "$(tput setaf 4)Select remote direction"
		echo "1) BigBot"
```

```
            echo "2) Jordi acer"
            echo "3) arlab Asus"
            echo "4) new...$(tput sgr 0)"
            while [ $ERR != "false" ]
            do
            read NUM
            case $NUM in
                  1)    USER="mobilerobot"
                        HOST="192.168.1.101"
                        ERR=false
                        ;;
                  2)    USER="jordi"
                        HOST="192.168.1.146"
                        ERR=false
                        ;;
                  3)    USER="arlab"
                        HOST="192.168.1.100"
                        ERR=false
                        ;;
                  4)    read -p "Introduce IP: " HOST
                        read -p "Introduce name of the user: " USER
                        ERR=false
                        ;;
                  *)    echo "$(tput setaf 1)ERR: wrong select$(tput sgr 0)"
            esac
            done

            USER_HOST="$USER@$HOST"
            gnome-terminal -e ./inici_remot.exp\ $USER_HOST\ $MESTRE\ $HOST\ 0
            sleep 1


      rosrun image_view image_view image:=/axis/image_raw _image_transport:=compres
sed &
            sleep 2

      rosrun image_view image_view image:=/camera/rgb/image_color _image_transport:
=compressed &
            sleep 10

            read -p "Vols iniciar stream_d'audio? (s/n)" AUDIO

            if [ $AUDIO = "s" ]; then
                  ./stream_audio.sh\ $MESTRE\ $ESCLAU\ $USER_HOST
            elif [ $AUDIO = "n" ];then
                  exit 0
```

```
            else
                  echo "$(tput setaf 1)ERR: Wrong type"

      echo "Pots iniciar manualment l'stream amb sh stream_audio.sh$(tput sgr 0)"
                  exit 1
            fi

      else
            exit 1
      fi

elif [ "$SCRIPT" = "slave" ]; then
      if [ $AUTO = "true" ]; then
            MESTRE="$MASTER"
            ESCLAU="$SLAVE"
      else
            read -p "ip of the PC or master: " MESTRE
            read -p "ip of the robot or slave: " ESCLAU
      fi

      export ROS_HOSTNAME=$ESCLAU
      export ROS_MASTER_URI=http://$MESTRE:11311
      gnome-terminal --title "kinect" -x bash -
c "roslaunch openni_launch openni.launch "&
      sleep 2

      echo "executa rosrun project bigbot"
      rosrun projectbigbot base_controller &
      sleep 3
      export ROS_NAMESPACE=axis
      rosrun axis_camera axis.py _hostname:=axis-
00408cac2fb4.local _password:=ferran &

      echo "executa rosrun axis_camera"
      sleep 3
      avconv -f alsa -i hw:0,0 -acodec libmp3lame -ab 32k -ar 14500 -re -
f rtp rtp://234.5.0.5:1234
      echo "enviant audio per rtp:234.5.0.5:1234"
      sleep 2
      exit 0
else
      echo "$(tput setaf 1)ERR: Wrong type$(tput sgr 0)"
      exit 1
fi
echo "$(tput setaf 2)Configurate as $SCRIPT. End program. $(tput sgr 0)"
#printf "es com echo pero sense fer un \n"
```

```
#roscore
#rosnode kill audio_play
#rosrun audio_capture audio_capture
#rosnode kill audio_capture
#rosrun audio_play audio_play
#gnome-terminal --title "rosrun axis_camera" -x bash -
c "rosrun axis_camera axis.py _hostname:=axis-
00408cac2fb4.local _password:=ferran "&
exit 0
```

A.4 CODI INICI REMOT

```
#!/usr/bin/expect
#ENCODING: UTF-8
#Created by Jordi Hortal

set USER_HOST [lindex $argv 0]
set MASTER [lindex $argv 1]
set SLAVE [lindex $argv 2]
set case [lindex $argv 3]

spawn ssh -o GSSAPIAuthentication=no $USER_HOST -X

expect -timeout 7 "*(yes/no)?" { send "yes\r";exp_continue }
expect -timeout 10 "*d: "

send "mobilerobot\n";
expect "*$ "
sleep 1

if { $case == 0 } {
       send "./inici.sh slave $MASTER $SLAVE\n";
} elseif { $case == 1 } {
       send "avplay rtp:234.5.0.5:1234\n";
       sleep 2
} else {
       put "error d'opcio";
       }
sleep 1
interact
```

A.5 CODI GENERACIÓ D'SCRIPTS

```
#!/bin/bash
echo ""
echo $(tput setaf 4)scrip per la creacio de scripts by jordi Hortal V1.3.1
echo scrips de bash, expect o python$(tput sgr 0)
echo ""

if [ $# -eq 0 ]; then
read -p "introdueix el nom de l'script-> " SCRIPT
EXTENSIO=false
elif [ $# -eq 1 ]; then
SCRIPT="$1"
EXTENSIO=false
       if [ "${SCRIPT}" = "$(echo ${SCRIPT} | egrep '\.+')" ]; then
       EXT=$(echo $SCRIPT | cut -d "." -f2)
       SCRIPT=$(echo $SCRIPT | cut -d "." -f1)
       EXTENSIO=true
       fi
elif [ $# -eq 2 ]; then
SCRIPT="$1"
EXT="$2"
EXTENSIO=true
else
echo $(tput setaf 1)Parámetros incorrectos
echo el nom no pot contenir espais buits$(tput sgr 0)
exit 1
fi

while [ "${SCRIPT}" != "$(echo ${SCRIPT} | egrep '^[a-z0-9_.]*$')" ]
do
echo $(tput setaf 1)nombre no valido: $SCRIPT
echo "No pot contenir caracters. Nomes numeros i/o lletres amb l'excepció de _"
read -p "introdueix el nom de l'script-> $(tput sgr 0)" SCRIPT
EXTENSIO=false
done

ERR=true
if [ $EXTENSIO = "false" ]; then
echo "$(tput setaf 4)Tipus d'Scripts que es poden generar: shell, bash, exp, py"
read -p "especifica llenguatge o extensio de l'script: $(tput sgr 0)" EXT
fi

while [ $ERR != "false" ]
do
```

```
case $EXT in
       bash)  EXT="sh"
                     RUTA="#!/bin/bash"
                     ERR=false
                     ;;
       expect|.exp|exp) EXT="exp"
                     RUTA="#!/usr/bin/expect"
                     ERR=false
                     ;;
       python|.py|py) EXT="py"
                     RUTA="#!/usr/bin/python"
                     ERR=false
                     ;;
       .sh|sh) EXT="sh"
                     RUTA="#!/bin/bash"
                     ERR=false
                     ;;
       shell) EXT="sh"
                     RUTA="#!/bin/sh"
                     ERR=false
                     ;;
       *)     echo $(tput setaf 1)extensio/llenguatge no reconeguda
              echo $SCRIPT y $EXT
              read -
p "especifica llenguatge o extensio de l'script: $(tput sgr 0)" EXT
              ;;
esac
done
cd $HOME && touch $SCRIPT.$EXT && chmod +x $SCRIPT.$EXT
echo $RUTA > $SCRIPT.$EXT && echo '#ENCODING: UTF-
8' >> $SCRIPT.$EXT && echo '#Created by Jordi Hortal' >> $SCRIPT.$EXT
echo $(tput setaf 2)script amb nom $SCRIPT.$EXT creat a: $HOME$(tput sgr 0)
sleep 1
geany ./$SCRIPT.$EXT&
exit 0
#ENCODING: UTF-8
#V1.2.1
#add line 17
#v1.3.1
#add option script.py as a name
# $# variable de bash que diu quants arguments hi han (si hi ha alguna cosa despres
 de la comanda)
# -eq sirve para comparar valores numericos es como un =
#echo -
e activa la interpretació de cadenas (ex \n com salt de linia \r retorn de carro
```

```
#egrep busca el que li demanis de un string el ^es per negar i el *$ es per indicar
 que pot apareixer algun cop i que no el tingui en compte
#per seguretat no es pot introduir caracters per si s'escriu per error directament
nom.sh
```

B COMUNICACIÓ ENTRE PIC I PLACA-PC

A continuació es descriuran tots els bytes de comunicació entre el PIC i el PC. Primer s'explicaran els bytes del PC cap al PIC i després els bytes de resposta del PIC cap al PC.

Al tractar-se de comunicació sèrie no es poden tractar les dades com "paquets" de forma literal, tot i que aquí es descriuen d'aquesta forma. La mida del paquet de dades sempre ha de concordar perfectament amb allò descrit pels flags per tal que es mantingui el sincronisme de les transmissions.

B.1 PAQUET PC A PIC

A continuació es detalla byte a byte el significat de les comandes per enviar al robot.

byte 0	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
IMU_pkg	Conf_PID	Err_ad	--	Corr_T	Corr_pos	T_order	Pos_order

Aquest byte defineix la mida del paquet total de dades enviades.
Aquest byte es obligatori en totes les transmissions.

Pos_order Determina si el paquet de dades conté una consigna de posició
0: No s'envia
1: Sí que s'envia

T_order Determina si el paquet de dades conté una consigna d'angle final
0: No s'envia
1: Sí que s'envia

Corr_pos Determina si el paquet de dades conté una correcció de posició
0: No s'envia
1: Sí que s'envia

Corr_T Determina si el paquet de dades conté una correcció de l'orientació
0: No s'envia
1: Sí que s'envia

Err_ad Determina si el paquet de dades conte un nou valor per l'error admissible durant el control de posició
0: No s'envia
1: Sí que s'envia

Conf_PID Determina si el paquet de dades conté nous paràmetres per al control de velocitat PID

0: No s'envia

1: Sí que s'envia

IMU_pkg Determina si el paquet conté únicament la correcció d'angle de l'IMU (cas especial de paquet de 3 bytes). L'activació d'aquest bit implica que el flag Corr_T hagi d'estar també activat.

0: No s'envia, el paquet és estàndard.

1: Sí que s'envia, el paquet només conté el yaw.

byte 1	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Encóders		Diff. tick	Ctl. Sig.	Current	Position	Batery	Speed

Aquest byte defineix quines dades ha de retornar el dsPIC en el seu missatge.
Aquest byte es obligatori en totes les transmissions (excepte si IMU_pkg = 1)

Speed Determina si el dsPIC envia les velocitats de cada roda

0: No s'envia

1: Sí que s'envia

Batery Determina si el dsPIC envia l'estat de la bateria i de la font de 5 Vdc

0: No s'envia

1: Sí que s'envia

Position Determina si el dsPIC envia la posició del robot calculada per odometria

0: No s'envia

1: Sí que s'envia

Current Determina si el dsPIC envia els corrents mesurats en cada roda

0: No s'envia

1: Sí que s'envia

Ctl. Sig Determina si el dsPIC envia els senyals de control aplicats a cada roda

0: No s'envia

1: Sí que s'envia

Diff. Tick Determina si el dsPIC envia les diferencies de gir mesurades entre la roda de davant i la de darrere

0: No s'envia

1: Sí que s'envia

Encóders Determina si el dsPIC envia les lectures dels encóders

0: No s'envia

1: Sí que s'envia

byte 2	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
					Ang_rate	Acceleration	Euler_ang

Aquest byte defineix quines dades ha de retornar la Litestation dins el seu missatge. És un byte només interpretat per la Litestation (pel cas de comunicacions indirectes) i ignorat pel dsPIC.

Aquest byte és obligatori en totes les transmissions (excepte si IMU_pkg = 1)

Euler_ang Determina si la Litestation envia els angles de l'IMU

0: No s'envia

1: Sí que s'envia

Accelerations Determina si la Litestation envia les acceleracions de l'IMU

0: No s'envia

1: Sí que s'envia

Ang_rate Determina si la Litestation envia les velocitats angulars de l'IMU

0: No s'envia

1: Sí que s'envia

byte 3	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
PID_sel	Set_offset		Led_2	Led_1	Mode_PID_1	Mode_PID_0	Ctl_Speed

Aquest byte defineix configuracions generals del robot

Aquest byte és obligatori en totes les transmissions (excepte si IMU_pkg = 1)

Ctl_Speed Determina si es fa control de velocitat

0: PID desactivat (llaç obert)

1: PID activat (llaç tancat)

Mode_PID Determina si el PID s'evalua independentment per cada roda o per parelles

00: PID per parelles (dreta i esquerra) (velocitat)

01: PID independent per cada roda (velocitat)

10: PID independent per cada roda (corrent)

Led_x Controla els leds d'alta intensitat per al sistema de localització (opcional)

0: Led desactivat

1: Led activat

Set_offset Actualitza el valor d'offset de les corrents al valor actual

0: No fa res

1: Actualitza amb la última lectura

PID_sel Selecciona sobre quin PID aplicar les configuracions

0: PID de velocitat (depèn del mode de PID)

1: PID de corrent (depèn del mode de PID)

Byte 4	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Order_FL							

byte 5	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Order_FR							

byte 6	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Order_RL							

byte 7	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Order_RR							

Aquests 4 bytes defineixen la consigna de velocitat per cada roda

Aquests bytes són obligatoris en totes les transmissions (excepte si IMU_pkg=1)

Order_xy Consignes de velocitat

Si Ctl_Speed = 0: Senyal de control [-100% ~ +100%]

Si Ctl_Speed = 1: Consigna de velocitat [-125cm/s ~ +125cm/s]

byte 8	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Order_X_cm							

byte 9	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Order_Y_cm							

byte 10	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Order_X_m							

byte 11	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Order_Y_m							

Aquests 4 bytes permeten donar una consigna de posició al robot. Sempre que la consigna de velocitat ho permeti, el robot es mourà fins arribar al punt donat a la consigna.
Aquests bytes són opcionals en funció del bit Pos_order

Order_x_cm Consignes de posició per als eixos X i Y, fracció de centímetres [-99cm ~ +99cm]
Order_x_m Consignes de posició per als eixos X i Y, part entera [-125m ~ +125m]

byte 12	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Order_T							

Aquest byte permet donar una consigna de posició angular. Si aquest byte es combina amb una consigna de posició, el robot es traslladarà al punt de consigna i seguidament aplicarà la consigna d'angle al final.
Aquest byte és opcional en funció del bit T_order.

Order_T Consignes de posició angular proporcional a 0º~360º [0 ~ 255]

byte 13	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Corr_X_cm							

byte 14	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Corr_Y_cm							

byte 15	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Corr_X_m							

byte 16	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Corr_Y_m							

Aquests 4 bytes permeten donar una correcció de la posició del robot en els eixos de coordenades.
Aquests bytes son opcionals en funció del bit Corr_pos.

Corr_x_cm Correcció de posició per als eixos X i Y, fracció de centímetres [-99cm ~ +99cm]
Corr_x_m Correcció de posició per als eixos X i Y, part entera [-125m ~ +125m]

byte 17	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Correction_T_Hi							

byte 18	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Correction_T_Lo							

Aquest byte permet donar una correcció de la posició angular del robot. Si el bit yaw_to_PIC es troba actiu, la Litestation inserirà en aquests bytes la lectura de l'IMU. En cas contrari es podrà inserir una correcció des del sistema de control.
Aquest byte és opcional en funció del bit Corr_T.

Correction_T Correcció de posició angular proporcional a 0º~360º x 100.

byte 19	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Error_ad							

Aquest byte permet fixar la precisió amb la qual es realitza el control de posició del robot.
Aquest byte és opcional en funció del bit Err_ad.

Eorror_ad Error de posició admissible (distància absoluta) [0cm ~ 100cm]

byte 20	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Kp_Hi							

byte 21	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Kp_Lo							

byte 22	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Ki_Hi							

byte 23	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Ki_Lo							

byte 24	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Kd_Hi							

byte 25	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Kd_Lo							

Aquests 6 bytes permeten modificar els paràmetres Kp, Ki i Kd del control de velocitat. Aquests valors s'han d'enviar en format de coma flotant de 16 bits dividits en 2 bytes. Quan s'apliquen nous valors, es genera un codi d'error específic per informar que s'ha fet correctament el canvi.
Aquests bytes son opcionals en funció del bit Conf_PID.

Kp_x Constant proporcional del PID
Ki_x Constant integral del PID (Ki=0 per control PD)
Kd_x Constant derivativa del PID (Kd=0 per control PI)

B.2 PAQUET PIC A PC

La resposta del robot igualment està formada per una part fixe del paquet, és a dir, que sempre s'envia, i per una part variable. Les dades que inclou la part variable del paquet depenen de les dades que s'han sol·licitat mitjançant el paquet de comandes.

byte 0	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Encóders	Sensors	Diff. tick	Ctl. Sig.	Current	Position	Batery	Speed

Aquest byte defineix la mida del paquet total de dades enviades.
Aquest byte és obligatori en totes les transmissions.

Speed Determina si s'envien les velocitats de cada roda
0: No s'envia

1: Sí que s'envia

Batery Determina si s'envia l'estat de la bateria i de la font de 5 Vdc

0: No s'envia

1: Sí que s'envia

Position Determina si s'envia la posició del robot calculada per odometria

0: No s'envia

1: Sí que s'envia

Current Determina si s'envien els corrents mesurats en cada roda

0: No s'envia

1: Sí que s'envia

Ctl. Sig Determina si s'envien els senyals de control aplicats a cada roda

0: No s'envia

1: Sí que s'envia

Diff. Tick Determina si s'envien les diferencies de gir mesurades entre la roda de davant i de darrera

0: No s'envia

1: Sí que s'envia

Sensors Determina si s'envien les lectures dels sensors infrarrojos

0: No s'envia

1: Sí que s'envia

Encóders Determina si s'envien les lectures dels encóders

0: No s'envia

1: Sí que s'envia

byte 1	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
					Ang_rate	Acceleration	Euler_ang

Aquest byte defineix si el paquet conté dades afegides per la Litestation. Aquest byte no té cap efecte si es fa servir el mètode de comunicació directe.
Aquest byte sempre s'envia

Euler_ang Determina si el paquet conté les orientacions llegiedes per l'IMU

0: No s'envia

1: Sí que s'envia

Accelerations Determina si el paquet conté les acceleracions

0: No s'envia

1: Sí que s'envia

Ang_rate Determina si el paquet conté les velocitats angulars

0: No s'envia

1: Sí que s'envia

byte 2	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Time_stamp							

Aquest byte marca la base de temps de les dades rebudes. El receptor ha de ser capaç de detectar el pas per zero. D'aquesta manera podrà interpretar correctament la base de temps.

Aquest byte sempre s'envia.

Time_stamp Base de temps [0 ~ 255ms] (quan passa de 255 torna al valor 0)

byte 3	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Error_code							

Aquest byte dona informació sobre mals funcionaments del robot. Cada codi correspon a un determinat missatge.

Aquest byte sempre s'envia.

001 Overflow T6 (RL)
002 Overflow T7 (FL)
003 Overflow T8 (FR)
004 Overflow T9 (RR)
005 Error flag driver M1 (FL)
006 Error flag driver M2 (FR)
007 Error flag driver M3 (RL)
008 Error flag driver M4 (RR)
011 Overflow en Con_FL
012 Overflow en Con_FR
013 Overflow en Con_RL
014 Overflow en Con_RR
016 Overflow en Speed FL (Speed value is max or min)
017 Overflow en Speed FR (Speed value is max or min)

018 Overflow en Speed RL (Speed value is max or min)
019 Overflow en Speed RR (Speed value is max or min)
020 PID parameters changed succesfully
021 Error I2C2 bus (sensors) lost data
022 Error I2C2 bus (sensors) was busy
023 Comunications timeout (causes stop)
026 Overflow on position correction (x)
027 Overflow on position correction (y)
028 Batery overvoltage
029 5V source overvoltage
030 Overflow Pos_x (Pos_x will be 0)
031 Overflow Pos_y (Pos_y will be 0)
032 Overflow on Int_FL (Int is max or min)
033 Overflow on Int_FR (Int is max or min)
034 Overflow on Int_RL (Int is max or min)
035 Overflow on Int_RR (Int is max or min)
036 Overflow on Sens_1 value (Sens is max)
037 Overflow on Sens_2 value (Sens is max)
038 Overflow on Sens_3 value (Sens is max)
039 Overflow on Sens_4 value (Sens is max)
040 Overflow on tic_R
041 Overflow on tic_L

byte 4	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
--	--	--	--	--	Arrived	Les_2_st	Led_1_st

Aquest byte dóna informacions generals.
Aquest byte sempre s'envia.

Led_x_st Informa sobre l'estat dels Led's
0: Apagat
1: Encès

Arrived En cas de treballar en mode de control de posició, informa sobre si s'ha assolit la posició final
0: El robot es troba en marxa
1: El robot ha arribat al destí

byte 5	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Speed_FL							

byte 6	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Speed_FR							

byte 7	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Speed_RL							

byte 8	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Speed_RR							

Aquests bytes donen la velocitat de cada roda del robot i s'envien en funció del bit Speed

Speed_xy Velocitat de la roda [-125cm/s ~ +125cm/s]

byte 9	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Batery							

byte 10	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Vcc							

Aquests bytes donen informació sobre l'estat de les alimentacions.

Aquests bytes s'envien en funció del bit Batery

Batery Nivell de la bateria [0 ~ 160 V·10]

Vcc Nivell de la font de 5V [0 ~ 200V·100] (Al valor s'han de sumar 3.5V per obtenir el nivell)

byte 11	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Pos_X_cm							

byte 12	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Pos_Y_cm							

byte 13	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Pos_X_m							

byte 14	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Pos_Y_m							

byte 15	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Pos_T							

Aquests 5 bytes donen la posició del robot en els eixos cartesians calculada per odometria i s'envien en funció del bit Position

Pos_x_cm	Posició per als eixos X i Y, fracció de centímetres [-99cm ~ +99cm]
Pos_x_m	Posició per als eixos X i Y, part entera [-125m ~ +125m]
Pos_T	Posició angular [0~256] equivalent [0º~360º]

byte 16	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Current_FL_Hi							

byte 17	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Current_FL_Lo							

byte 18	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Current_FR_Hi							

byte 19	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Current_FR_Lo							

byte 20	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Current_RL_Hi							

byte 21	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Current_RL_Lo							

byte 22	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Current_RR_Hi							

byte 23	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Current_RR_Lo							

Aquests 8 bytes donen les lectures de corrent de cada motor. Els valors estan representats com a enters de 16 bits fraccionats en dos bytes.
Aquests bytes són opcionals en funció del bit Current.

Current_xy_z Valor de corrent. [-5000mA ~ +5000mA]

byte 24	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Control_FL							

byte 25	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Control_FR							

byte 26	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Control_RL							

byte 27	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Control_RR							

Aquests 4 bytes donen la senyal de control aplicada a cada roda del robot
Aquests bytes s'envien en funció del bit Control

Control_xy Senyal de control [-100% ~ +100%]

byte 28	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Diff_tick_R							
byte 29	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Diff_tick_L							

Aquests 2 bytes donen la diferència de gir entre la roda davantera i la de darrere de cada costat en polsos dels encóders.
Aquests bytes s'envien en funció del bit Diff_tick.

Diff_tick_x Diferència de gir per les rodes de la dreta i l'esquerra en valor absolut [0 ~ 255 polsos]

byte 30	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Temp_Hi							

byte 31	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Temp_Lo							

byte 32	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Sens_1							

byte 33	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Sens_2							

Aquests 4 bytes donen la lectura de quatre sensors analògics.
Aquests bytes s'envien en funció del bit Sensor.

Temp_x Lectura del sensor de distància [0~4096] equivalent a [0~100ºC]
Sens_x Sensor de distància – actualment no implementat --

byte 34	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Enc_L_Hi							

byte 35	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Enc_L_Lo							

byte 36	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Enc_R_Hi							

byte 37	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Enc_R_Lo							

Aquests 4 bytes donen la lectura dels encóders. Cada valor correspon a la mitjana dels valors de les rodes del costat dret i esquerre. Els valors s'expressen com un enter de 16 bits fraccionat en dos bytes. S'interpreten com un increment respecte l'últim valor enviat.
Aquest byte s'envia en funció del bit encóder.

Enc_x_y Lectura de l'encóder. [-32512 ~ +32512 polsos]

byte 38	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Roll_Hi							

byte 39	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Roll_Lo							

byte 40	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Pitch_Hi							

byte 41	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Pitch_Lo							

byte 42	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Yaw_Hi							

byte 43	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Yaw_Lo							

Aquests 6 bytes donen la lectura d'orientació obtinguda amb l'IMU. Aquests bytes no es poden obtenir en mode de comunicació directa.
Aquest byte s'envia en funció del bit Euler_ang

Roll_x Angle Roll [0º~360º]
Yaw_x Angle Yaw [0º~360º]
Pitch_x Angle Pitch [0º~360º]

byte 44	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Accel_X_Hi							

byte 45	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Accel_X_Lo							

byte 46	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Accel_Y_Hi							

byte 47	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Accel_Y_Lo							

byte 48	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Accel_Z_Hi							

byte 49	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Accel_Z_Lo							

Aquests 6 bytes donen la lectura d'acceleracions en els 3 eixos obtinguda amb l'IMU. Aquests bytes no es poden obtenir en mode de comunicació directa. S'envien en funció del bit Accel

byte 50	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Ang_rate_X_Hi							

byte 51	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Ang_rate_X_Lo							

byte 52	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Ang_rate_Y_Hi							

Byte 53	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Ang_rate_Y_Lo							

byte 54	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Ang_rate_Z_Hi							

byte 55	bit: 6	bit: 5	bit: 4	bit: 3	bit: 2	bit: 1	bit: 0
Ang_rate_Z_Lo							

Aquests 6 bytes donen la lectura de velocitat angular en els tres eixos obtinguda amb l'IMU. Aquests bytes no es poden obtenir en mode de comunicació directa. S'envien en funció del bit Ang_rate