U SCIENCE TECH
FACULTAT DE CIÈNCIES
I TECNOLOGIA
UVIC-UCC

Treball de Fi de Màster

Desenvolupament d'un robot mòbil terrestre educatiu per a la introducció al ROS

Marc Genevat Travesa

Màster en Robòtica

Tutor/a: Dr. Carlos J. Rosales Gallegos

Vic, Setembre de 2016

U SCIENCE TECH
FACULTAT DE CIÈNCIES
I TECNOLOGIA
UVIC-UCC

RESUM TREBALL FINAL DE GRAU

GRAU EN ENGINYERIA MECATRÒNICA

Títol: Desenvolupament d'un robot mòbil terrestre educatiu per a la introducció al ROS

Paraules clau: ROS, SLAM, RViz, URDF, Gazebo, opensource, Fusion 360, render, modelització, simulació, Arduino, Raspberry Pi 3, IMU, Kinect.

Autor: Marc Genevat Travesa

Tutor: Dr. Carlos J. Rosales Gallegos

Data: 26 de Setembre de 2016

Resum

Aquest Treball Final de Master consisteix en el desenvolupament d'un robot mòbil terrestre de baix cost, *opensource i openhardware* per tal de que estudiants de robòtica el puguin fer servir per introduir-se a ROS, un dels sistemes operatius per a robots més utilitzats actualment.

El projecte s'ha dividit en tres parts, repartint-les entre en Juan Pedro López, l'Oriol Orra i en Marc Genevat. La part d'aquest projecte que es presenta en aquesta memòria s'ha centrat en el disseny i fabricació de la plataforma, la modelització del sistema en URDF i la implementació del model en l'entorn de simulació Gazebo.

Gràcies a una bona organització, esforç i les noves tecnologies de fabricació, la majoria dels requeriments que es van proposar que complís la plataforma s'han complet i alguns d'ells amb resultats sorprenents. El cost de fabricació ha resultat ser molt més baix de l'esperat i el pes de la plataforma i la seva capacitat de càrrega també han resultat ser molt bons.

U SCIENCE TECH
FACULTAT DE CIÈNCIES
I TECNOLOGIA
UVIC-UCC

MASTER FINAL PROJECT ABSTRACT

MASTER ON ROBOTICS

Title: Development of an educational mobile ground robot for ROS introduction

Key words: ROS, SLAM, RViz, URDF, Gazebo, open source, *open hardware*, Fusion 360, render, modelization, simulation, Arduino, Raspberry Pi 3, IMU, Kinect.

Author: Marc Genevat Travesa

Tutor: Dr. Carlos J. Rosales Gallegos

Date: 26 September, 2016

Abstract

This Master Final Project consists of the development of a low-cost educational mobile ground robot, open source and open hardware because robotics students can introduce themselves into ROS, one of the most used robotics operation systems.

The project is divided in three main parts, distributing them among Juan Pedro López, Oriol Orra and Marc Genevat. This report is focused on the part centralized on the platform design and Building, the URDF System modelization and the model implementation in the simulation environment Gazebo.

Thanks to a good organization, effort and the new technologies of fabrication, the most part of requirements that were suggested to be accomplished by the platform have been so and some of them with surprising results. The cost of production has resulted to be much lower of what was expected and the platform's weight and its payload have also turned out to be good.

Agraïments

Ha estat un procés d'uns tres quatre mesos molt intens, exprimint el temps fins al final per tal d'arribar a assolir el màxim possible. Tots els esforços i tota la dedicació finalment han valgut la pena. I tot això no hagués estat possible sense la col·laboració dels meus dos companys Juan Pedro López i Oriol Orra, l'ajuda d'en Jordi Serra, cap dels tallers de la Universitat de Vic (UST), durant tot el procés de fabricació i evidentment la paciència i el suport constant d'en Carlos Rosales, tutor del projecte. No voldria oblidar el recolzament continu de la família i amics, que m'han ajudat a tirar en tot moment.

CONTINGUTS

Índex de figures

Índex de taules

Capítol 1

Introducció

1.1. Motivació

La idea d'aquest projecte, extens i ambiciós, és fruit de que tant l'Oriol Orra, en Juan Pedro López i jo mateix, en Marc Genevat, hàgim cursat la primera edició del màster en robòtica de la UVic-UCC i Ascamm-Eurecat i hàgim trobat certes mancances en la part pràctica de robòtica mòbil terrestre. Tot i disposar d'un parell de plataformes força adequades pel món de la recerca i educació i tenint en compte el seu cost, les pràctiques que es van dur a terme durant una part del segon semestre a les instal·lacions d'ASCAMM van deixar molt que desitjar pel que fa a la preparació prèvia, al contingut i guiatge de les pràctiques i a l'aportació i enriquiment final de coneixement per part de l'alumnat. Aquest fet ens va motivar per desenvolupar des de zero la nostra pròpia plataforma basada en ROS (concepte que s'explicarà al capítol següent) i de baix cost i que comptés amb la seva pròpia implementació en un entorn simulat.

El robot, que més endavant va passar a anomenar-se Canyonero, i el projecte en sí, estarien totalment dissenyats per assistir a universitats i professors en introduir ROS als seus estudiants de manera pràctica i equitativa gràcies a l'entorn simulat. A més, es prepararia de tal manera que seria fàcilment adaptable per a la recerca millorant les prestacions actuals o bé incorporant-hi més components.

El temps i els recursos amb què s'ha volgut fer la plataforma han estat tot un repte, però gràcies a una bona organització, un bon tutoratge i moltes ganes de crear i aprendre s'ha pogut tirar endavant el projecte des del primer moment en què es va plantejar i amb un progrés força constant.

1.2. Objectius

L'objectiu d'aquest treball final de Màster és, principalment, la creació d'un robot mòbil terrestre educatiu de baix cost i basat en ROS. Aquest robot, anomenat Canyonero, ha d'estar completament integrat tant a nivell de *hardware*, *software* i simulació perquè qualsevol estudiant d'enginyeria robòtica o investigador pugui treballar amb la plataforma, ja sigui afegint-hi components o testejant els seus propis algoritmes. Així doncs, la plataforma Canyonero pretén ser accessible per a universitats i grups de recerca perquè serveixi tant per educació en l'àmbit de la robòtica mòbil com per recerca i desenvolupament.

Concretament, els objectius plantejats per aquest projecte es podrien desglossar de la següent manera:

- Disseny, fabricació i muntatge d'un robot mòbil terrestre basat en ROS.
- Modelització de la plataforma en URDF
- Simulació del sistema robotitzat en l'entorn Gazebo.

A continuació es presenten els requeriments que ha d'assolir aquest projecte en concret:

Requeriments

Concepte	Requeriment/Desig	Descripció
Funcions	R	Capacitat de moviment per interior i exteriors
	R	Capacitat de càrrega: 20Kg
	D	Carrosseria hermètica i impermeable
	R	Estructura modular
	R	Estructura amb quatre rodes motoritzades
	R	*Hardware opensource*
	R	*Hardware* accessible
	R	Implementació en ROS
	R	Implementació del model URDF
	R	Visualització del model URDF en Rviz
	D	Simulació del model URDF en Gazebo
Pes i dimensions	R	Pes màxim: 20 Kg
	R	Dimensions màximes: 0,5 m^3
Moviments	R	Moviment diferencial
	R	2 DOF
Material	R	Material lleuger, resistent i econòmic com alumini, plàstic, ...
Senyals i control	R	Integració d'una càmera 2D/3D
	R	Integració d'encoders (odometria)
	D	Integració de sensor inercial
	D	Integració de GPS
	R	Teleoperació amb *joystick* sense cables
	D	Teleoperació amb *Leap Motion*
Comunicació	R	Comunicació amb un PC extern
Autonomia	R	20 minuts
Vida útil	R	6 mesos

Taula 1. Especificacions del projecte

1.3. Metodologia

El projecte Canyonero ha estat des de l'inici un projecte ambiciós i amb objectius massa amplis perquè un sol estudiant el pugui portar a terme individualment en tres mesos. Així que unint esforços entre tres estudiants de *backgrounds* diferents s'ha pogut dividir el projecte en tres subprojectes on la quantitat de feina ja seria més assumible per cada un. A continuació, a la Figura 1 es mostra els membres i l'organització del projecte:

Figura 1. Equip Canyonero, divisió de tasques

Una vegada clarificat quina és la part que ocupa aquesta memòria, es parlarà sobre l'estructura d'aquesta. Primer de tot es vol fer una petita introducció a ROS, s'entén que el lector d'aquesta memòria ja coneix aquest sistema operatiu per a robots però com que aquesta memòria i subprojecte vol tenir un caire educatiu, es repassarà com va sorgir i els beneficis que va aportar a la robòtica, es mostrarà l'estat de l'art dels robots mòbils basats en ROS més similars al robot que es vol desenvolupar en aquest projecte i finalment es repassaran els conceptes bàsics i les eines més utilitzades dins del món ROS, la majoria de les quals s'han utilitzat en força profunditat en el projecte Canyonero. Seguidament s'explicarà com s'ha dut a terme el desenvolupament del robot pel que fa a nivell de sistema, de disseny i construcció, tot analitzant les parts més crítiques de la plataforma i el cost de fabricació. Una vegada explicada la part mecànica es passarà a parlar sobre com s'ha integrat el model Canyonero a ROS. S'explicarà com s'ha modelitzat el sistema en URDF i la seva implementació dins de l'entorn de simulació Gazebo.

Finalment, es conclourà la memòria amb unes reflexions sobre tot el procés i les línies de futur que podria tenir aquest projecte, les quals seran una part fonamental ja que servirà a futurs estudiants de robòtica aprendre dels errors o de les bones pràctiques realitzades dins d'aquest treball i a més veuran propostes per com continuar-lo i dur-lo més enllà.

Al final de la memòria es poden trobar un parell d'annexos dedicats als plànols de les peces que s'han dissenyat i dels codis que s'han implementat, els quals es poden trobar a GitHub.

Capítol 2

ROS – *Robot Operation System*

Per tal de que el lector pugui comprendre perfectament el projecte i seguint amb el seu caràcter educatiu i divulgatiu, és convenient explicar en què consisteix el sistema operatiu per robots anomenat ROS, amb el qual està basat el robot desenvolupat en qüestió. Així doncs, aquest capítol parlarà primer de tot sobre com va sorgir ROS i els beneficis que va aportar la seva creació al món de la robòtica. Seguidament es presentaran els robot més destacables que hi ha actualment compatibles amb ROS dins dels que són similars al robot que s'ha desenvolupat en aquest projecte, per la qual cosa són d'especial interès. I per acabar, s'explicaran els conceptes bàsics i les eines més utilitzades dins d'aquest entorn per a robots per les quals el fa ser tan popular entre els desenvolupadors de plataformes robòtiques.

2.1. Introducció

El *Robot Operating System*, àmpliament conegut com ROS, és bàsicament un *framework* flexible per desenvolupar *software* per a robots. Es basa en una col·lecció d'eines, llibreries i convencions que tenen com a objectiu simplificar i estandarditzar tasques complexes per a una, cada cop més extensa, varietat de plataformes robòtiques mitjançant una política de *software* obert o *open- source*.

Figura 2. Logotip oficial de ROS (http://www.ros.org/about-ros/)

ROS va néixer fa 9 anys, al 2007, a mans d'una incubadora californiana plena de visionaris de la robòtica, Willow Garage. Tot i ser una empresa privada, manté un fort vincle amb Stanford University i tot el *software* i *hardware* que desenvolupen mitjançant dos robots, el PR2 i el Turtlebot, dels quals se'n parlarà més endavant, és alliberat sota la llicència *BSD license*, perquè tothom pugui gaudir del coneixement que generen i es continuï innovant. Segons ells, només sumant esforços s'aconseguirà que els robots arribin a les nostres cases.

Des de l'inici, ROS va estar desenvolupat per múltiples institucions i per a múltiples robots. Actualment, l'ecosistema ROS compte d'una comunitat de desenes de milers d'usuaris d'arreu del món, desenvolupant des de projectes personals fins a grans sistemes industrials automatitzats, sempre contribuint i alliberant el que es genera, garantint així el creixement constant de ROS,

motiu pel qual és un dels sistemes operatius per a robots més populars.

2.2. Beneficis que ROS aporta al món de la robòtica

Abans de que comencessin a aparèixer els sistemes operatius per a robots, el desenvolupament de plataformes robòtiques era summament difícil fer-ho per un mateix. Les diferents disciplines que abasten la robòtica com la mecànica, electrònica i programació integrada, feia que es necessitessin grans equips d'experts en les matèries per tal de dur a terme projectes en l'àmbit de la robòtica, havent de desenvolupar sempre el seu propi *software* pel *hardware* específic del que disposaven.

La principal idea dels sistemes operatius per a robots era que no es reinventés la roda contínuament, és a dir, compartint lliurament les experiències i coneixements obtinguts en altres projectes per tal de que els següents a desenvolupar alguna cosa similar no haguessin de desenvolupar els seus propis algoritmes des de zero i tinguessin certs punts de partida, podent així arribar molt més lluny. A més, com qualsevol sistema operatiu per ordinadors, sistemes operatius per a robots com ROS permeten a l'usuari gestionar i tenir un control de baix nivel de tot el *hardware* que composa el robot.

Així doncs, es podria dir que ROS ha afegit un valor molt considerable als projectes de robòtica i a les seves aplicacions. Tot gràcies a ser dissenyat de la manera més distribuïda i modular possible, permetent a l'usuari escollir quines parts aprofita de la comunitat de ROS i quines parts desenvolupa ell mateix, aportant-ho així al ecosistema ROS.

2.3. Estat de l'art

En aquest apartat es fa un recull de les plataformes de robòtica mòbil terrestre basades en ROS més aplicables en el món educatiu actualment. Tot i que el número de plataformes desenvolupades dins d'aquest àmbit és molt extens, no n'hi ha masses que tinguin una empresa o un equip d'enginyers darrera que donin suport i mantinguin el robot amb les últimes actualitzacions tant a nivell de *software* com de *hardware*.

2.3.1. Clearpath

Clearpath és una de les empreses líders en el sector de la robòtica mòbil autònoma i la robòtica de serveis en general. Els seus inicis es remunten al 2009, quan quatre amics apassionats per la robòtica van començar a construir els seus propis vehicles autònoms al soterrani de la University of Waterloo, Canadà. Ofereixen un ampli ventall de possibilitats dins de la robòtica autònoma, distribuint robots tant de terra com mar i aire. La llista d'acontinuació es centrarà únicament amb

les plataformes terrestres que s'assemblen més a la plataforma desenvolupada dins d'aquest projecte:

- **Husky**: Plataforma robòtica de mida mitjana (990 x 670 x 390 mm) àmpliament utilitzada en recerca i en robòtica de serveis. El pes del robot és de 50Kg i asseguren una capacitat de càrrega de 75Kg, ideal per integrar-hi qualsevol tipus de sensor o enginy mecatrònic capdamunt. Pel que fa a la velocitat, pot arribar a 1m/s i la seva autonomia, segons la pàgina principial de Clearpath, és de 3hores fent un ús típic. La versió bàsica del Husky, tot i que molt personalitzable, no disposa de cap sensor integrat ni ordinador d'abord, únicament encoders als motors, controladora de motors i bateria.

Figura 3. Robot Husky de Clearpath (http://www.clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/)

  Val a dir que és una plataforma molt utilitzada en el món de la recerca gràcies a la seva capacitat de càrrega, robustesa i versatilitat.

- **Jackal:** És la plataforma terrestre més petita (508 x 430 x 250 mm) que ofereix Clearpath. El pes del robot és de 17Kg i la seva capacitat de càrrega de 20Kg. Gràcies al seu tamany i la seva lleugeresa és el doble de ràpid que el Husky, pot arribar als 2m/s i la seva autonomia s'allarga a les 4 hores. Al igual que el Husky, la versió bàsica només disposa de encoders pel que fa a sensors, controladora de motors i bateria.

Figura 4. Robot Jackal de Clearpath (https://www.clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/)

2.3.2. Omron Adept Technologies, Inc.

Una altra empresa capdavantera en aquest àmbit és **Omron Adept Technologies, Inc.**, amb el nom d'Adept MobileRobots, Inc. en els seus inicis el 1995. Aquell any van llençar el primer robot de la família Pioneer, família que des de llavors no ha parat de créixer.

Adept MobileRobotics va ser adquirida per un dels líders mundials en automatització, Omron Corporation, el 2015 i des de llavors la divisió de recerca i educació d'Omron Adept ha continuat proveint les plataformes robòtiques més utilitzades en el món de la recerca de robòtica mòbil.

Els sistemes comercials i industrials de robòtica mòbil d'Omron Adept Technologies està

revolucionant la manera de treballar en llocs com fàbriques, magatzems, hospitals i laboratoris, entre altres. La seva família de robots Pioneer és globalment referent gràcies a la seva versalitat, eficiència i durabilitat, factors que la fan ideal per la recerca en robòtica mòbil. Tots els robots Pioneer inclouen un controlador de motors dedicat totalment programable amb encoders i còmode per utilitzar la plataforma tant per missions autònomes com teleoperades, ja sigui en recerca o educació, tot comptant amb el seu propi *software*, el Pioneer SDK. A continuació, es presenten un parell dels seus robots més interessants a tenir en compte en aquest projecte:

- **Pioneer 3-DX:** Plataforma mòbil diferencial compacta de dues rodes totalment ensamblada amb un controlador de motors integrat amb encoders de 500 polsos, rodes de 190mm de diàmetre, xassís d'alumini, 8 sensors d'ultrasons al davant i bateria. Acaba sent un kit de desenvolupament de *software* força complet on només cal afegir un ordinador perquè estigui totalment a punt.

Figura 5. Robot Pioneer 3-DX de Omron Adept Technologies, Inc (http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx)

La base del robot pesa 9Kg i mesura 381x455x237mm amb una capacitat de càrrega de 17Kg. La seva autonomia amb 3 bateries és de 8-10 hores i la velocitat lineal màxima que pot arribar és de 1,2m/s.

Igual que el Husky de Clearpath, el Pioneer 3 DX és perfectament personalitzable amb una gran varietat d'accessoris i suficientment robust per aguantar anys en recerca o ensenyament.

- **Pioneer 3-AT:** Plataforma mòbil molt versàtil de quatre rodes amb encoders de 100 polsos més potent que l'anterior i molt popular en projectes de flotes de robots per a exteriors o terrenys complicats.

Figura 6. Robot Pioneer 3-AT de Omron Adept Technologies, Inc. (http://www.mobilerobots.com/ResearchRobots/P3AT.aspx)

La base del robot pesa 12Kg i mesura 497x508x277mm amb una capacitat de càrrega de 12Kg. La seva autonomia amb 3 bateries és de 2-4 hores i la velocitat lineal màxima que pot arribar és de 0,7m/s.

El P3-AT ofereix la possibilitat de tenir un ordinador integrat permetent l'opció a realitzar processament d'imatge abord, comunicacions Ethernet, utilització de làser, DGPS i altres funcions comuns en robòtica mòbil autònoma.

2.3.3. Innok Robotics

L'empresa alemana **Innok Robotics** també és un clar exemple de desenvolupadors i distribuidors de plataformes robòtiques terrestres aplicades a la robòtica de serveis. Col·laborant amb la Regensburg University of Applied Sciences, tenen com objectiu crear robots per facilitar i assumir tasques perilloses de la feina dels seus clients.

Estan especialitzats amb robots terrestres d'assistència en exteriors. La seva oferta es basa en dos models, el **Innok TX** que destaca per la seva veocitat i agilitat fins i tot en els terrenys més complicats i el **Innok Heros** que destaca per la seva versalitat i la ratio "price-performance", ja que és totalment personalitzable segons l'aplicació a realitzar. Tot seguit es presenten els dos models de manera més detallada:

- **Innok Heros**: Plataforma robòtica modular capaç de moure's per tot tipus de terrenys d'interior i exterior. Està disponible amb 3 i 4 rodes i disposa d'una gran quantitat de mòduls per poder adaptar-se a la necessitat del client. Les seves dimensions van de 900 a 1500mm d'allargada i de 620 a 800mm d'amplada. L'alçada del robot és de 410mm. El pes de la plataforma pot variar entre 70 i 140Kg i la seva capacitat de càrrega, depenent de la configuració mecànica, va de 70 a 400Kg. La seva velocitat i autonomia variarà notablement depenent de la seva configuració mecànica i aplicació.

Figura 7. Robot Innok Heros de Innok Robotics (http://www.innok-robotics.de/en/products/heros)

  Gràcies a la seva versalitat i personalització, fa que Innok Heros sigui una plataforma molt interessant i que caldria tenir en compte per aplicacions de robòtica de serveis en sectors

com agricultura, logística, recerca, etc.

- **Innok TX:** Plataforma robòtica de sis rodes expressament dissenyada per treballar en exteriors. El robot està equipat de suspensions independents per a cada roda i pot arribar als 10m/s a camp obert. La seva personalització a nivell de *hardware* resulta molt senzilla gràcies a la seva estructura de perfils d'alumini estandarditzats. El control del robot es pot fer a través d'un Bus CAN ja integrat a la plataforma que permet accedir a diferents dades.

Figura 8. Robot Innok TX de Innok Robotics (http://www.innok-robotics.de/en/products/tx)

2.3.4. Enova Robotics

Enova Robotics va néixer el 2014 després d'una dècada d'experiència en educació i recerca dins del camp de la robòtica. És una empresa especialitzada en el desenvolupament de robots mòbils i en projectes d'I+D sobre robòtica. Sent de Tunísia, és la primera empresa a Àfrica i dins del món aràbic que dissenya i produeix els seus propis robots.

Entre els tres robots que han desenvolupat des dels seus inicis, n'hi ha un parell que resulten molt interessants i presenten força similituds amb el Canyonero, un per la seva aplicació educativa, el Mini-Lab, i l'altre pel seu disseny estructural i amb quatre rodes motoritzades, el PearlGuard.

- **Mini-Lab**: robot mòbil de mida mitjana que segons Enova Robotics ofereix la millor relació robustesa-preu del mercat. El Mini-Lab ha estat concebut per aplicar-se en recerca i educació.

Figura 9. Robot Mini-Lab de Ennova Robotics (https://www.enovarobotics.eu/index.php/products/mini-lab)

La plataforma educativa d'Enova Robotics està destinada a realitzar aplicacions en interiors, les seves dimensions són de 409x364x231mm i el seu pes és de 11,5 Kg. El seu sistema motriu és diferencial, compost per dues rodes amb motor i encoder a cada una d'elles. Pot suportar una petita càrrega de 3Kg i arribar a una velocitat de 1,5 m/s, comptant amb una

autonomia de 4 hores. A més, disposa de la càmera de profunditat Asus Xtion Live Pro i 5 sensors d'ultrasons i 5 sensors de infraroig per tal de poder navegar autònomament.

- **PearlGuard**: Robot de quatre rodes monitoritzades dedicat a la vigilància i la seguretat amb una autonomia de 8 hores. Per realitzar aquests tipus d'operacions, el robot disposa d'un conjunt de càmeres infraroges repartides per la plataforma per tal de poder cobrir completament tot el seu entorn, una càmera tèrmica, un sistema d'adquisició de so omnidireccional i un sistema d'alarma sonor i lluminós. A més, disposa també de GPS per localitzar-se geogràficament i un sistema làser per detecció.

Figura 10. Robot PearlGuard de Ennova Robotics (https://www.enovarobotics.eu/index.php/products/pearlguard)

El robot ve acompanyat d'una interfície remota que permet a l'usuari controlar el robot manualment i a distància, accedir a les dades de tots els seus sensors i planificar rutes o trajectòries de moviment mitjançant punts de referència GPS, apart de monitoritzar l'estat del robot, com el nivell de càrrega de la bateria, la seva velocitat, estat de la senyal de comunicació, etc.

Les dimensions de la plataforma són de 1210x866x771mm i els seu pes és de 120Kg. La càrrega màxima que pot suportat és de 10Kg i és capaç d'arribar als 3m/s amb una acceleració de $1m/s^2$.

2.3.5. Robotnik

Per últim, es parlarà d'una empresa de robòtica de serveis de l'estat espanyol, anomenada **Robotnik**. L'empresa valenciana es va fundar l'any 2002, dedicant-se inicialment a la robòtica industrial i a l'automatització fins acabar centrant la seva activitat empresarial a la robòtica de serveis. És una empresa força coneguda a l'estat espanyol pel seu èxit empresarial, la seva capacitat d'innovació i amb una alta formació tecnològica i professional.

Actualment, ofereixen serveis com a desenvolupadors de projectes o aplicacions robòtiques però també distribuïdors sistemes robòtics. Tenen disponible una gran varietat de tipologies de robots i models basats en ROS, des del petit humanoide OP2, passant per manipuladors i braços robots fins a plataformes mòbils, que algunes d'elles seran les que es veuran a continuació amb més

detall:

- **Summit XL**: Plataforma mòbil basada en ROS de cinemàtica diferencial amb quatre motors *brushless* i amb encoders en cada una de les seves rodes.

  Aquest *rover* té dues possibles configuracions cinemàtiques. La configuració omnidireccional que serveix úniament per moure's en interior i disposa de quatre rodes mecanum muntades en un sistema de suspencions independents i la configuració d'exteriors-interiors que es canviarien les rodes mecanum per unes convencionals com les de la figura dónant-li més versatalitat pel que fa al terreny però menor maniobrabilitat.

Figura 11. Robot Summit XL de Robotnik (http://www.robotnik.es/robots-moviles/summit-xl/)

  Les dimensions de la plataforma són de 722x610x392mm, té un pes de 45Kg i capacitat de càrrega de 20Kg. Asseguren una autonomia per un ús estàndard en el laboratori de20 hores però de 5 hores per un ús de moviment continu. La velocitat lineal màxima és de 3m/s.

  El robot pot navegar autònomament o ser teleoperat mitjançant una càmera de dos eixos (*Pan-Tilt*) que transmet vídeo a temps real. Els accessoris estàndars incloen un escàner làser Hokuyo i kits RTK-DPGS per localització i navegació. Tanmateix, disposa de connectivitat interna (USB, RS232 i GPIO) i externa (USB, RJ45, alimentació de 12V) per poder acoplar fàcilment tot tipus de components que requereixin les aplicacions o recerca a desenvolupar.

- **TurtleBot 2:** Plataforma successora del robot TurtleBot de la qual tant Robotnik com Clearpath són integradors i distribuidors, entre altres. Presumeix de ser la plataforma robòtica més econòmica del mercat que suporta arquitectura ROS per utilitzar-se en recerca i educació. Aquest robot mòbil de cinemàtica diferencial es pot utilitzar en múltiples aplicacions com AAL (*Ambient Assisted Living)*, HRI (*Human Robot Interface*), generació de mapes, localització i navegació autònoma, sistemes multi-robot, etc. gràcies al gran ventall de paquets disponibles en ROS.

Figura 12. Robot Turtlebot 2
(http://www.robotnik.es/robots-moviles/turtlebot-2/)

La plataforma està basada amb la base Kobuki, que és la utilitzada pel robot de neteja Roomba. Les dimensions de la base bàsica són de 315,430,347mm, tot i que és expandible cap amunt en format sandvitx. El pes de la base inicial és de 5Kg i té una capacitat de càrrega també de 5Kg. L'autonomia del robot amb la bateria gran és de 7 hores i amb la petita de 3 hores, podent arribar a una velocitat lineal màxima de 0,65 m/s.

Els sensors dels que disposa la plataforma inicialment, tot i que és fàcilment ampliable, són encoders de 11.5 polsos/mm, càmera 3D que pot ser la ASUS Xtion PRO LIVE o la Kinect de Microsoft i un sensor giroscopi de tipus *rate gyro* de 110 deg/s.

Tot i que possiblement no seria un robot que serviria per aplicacions de servei finals, gràcies a la seva versalitat, baix cost i la gran quantitat de *software* ja implementat obert a la comunitat, el Turtlebot 2 s'ha convertit en un dels robots més utilitzats per iniciar-se en ROS o per fer recerca en robòtica mòbil autònoma.

2.4. Fonaments de ROS

La funció bàsica que ha de fer un sistema operatiu per a robots és córrer un gran número d'executables en paral·lel que requereixin un intercanvi de dades entre ells de manera síncrona o asíncrona. Per exemple, un sistema operatiu per a robots haurà d'adquirir dades dels sensors del robot a una certa freqüència, tractar-la, passar-la a algoritmes de processament ¨(reconeixement de veu, visió artificial, etc.) i per acabar controlar els motors en conseqüència. Tot aquest procés o similars es van generant contínuament i en paral·lel dins del sistema.

A continuació s'explicarà el conjunt de conceptes que formen la xarxa computacional de ROS, la qual fa possible tot el descrit anteriorment.

2.4.1. Conceptes bàsics

- ***ROS master*:** ROS comença amb el *ROS Master*. El *master* permet que tots els altres "components" de *software* de ROS (*Nodes*) es reconeguin i es comuniquin entre ells. Sense el *master*, els nodes no es podrien trobar entre ells, intercanviar missatges o cridar serveis. D'aquesta manera no cal indicar cada vegada que es vulgui adquirir dades per exemple com "Enviar les dades d'aquest sensor a l'ordinador a 127.0.0.1. Simplement es demana que el Node 1 enviï un missatge al Node 2. Això els nodes ho fan a través dels *topics*.

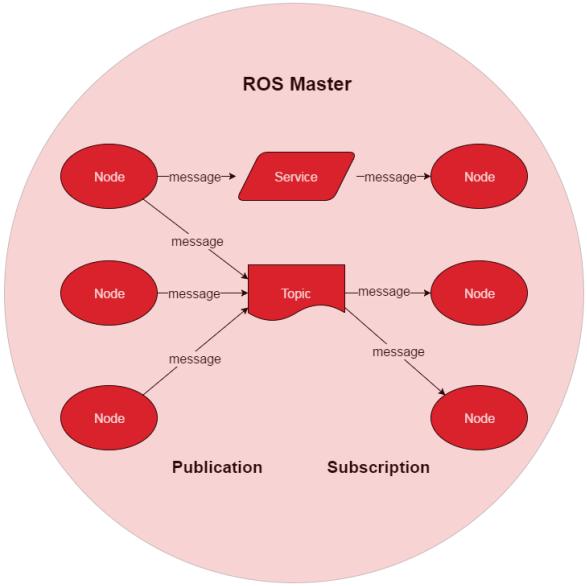

Figura 13. Xarxa computacional ROS
Font: autor

Així doncs, el *ROS Master* proporciona noms de registre i cerca per la resta de la xarxa computacional de ROS. A més, el *ROS Master* inclou un component molt utilitzat anomenat *Parameter Server* o Servidor de paràmetres que, com el nom indica, consisteix en una base de dades centralitzada on els nodes hi poden emmagatzemar dades i alhora compartir paràmetres de tot el sistema.

- ***Nodes*:** Els nodes són bàsicament processos computacionals. ROS està dissenyat per ser completament un sistema modular i el sistema d'un robot normalment està compost de molts nodes. Per exemple, un node es podria ocupar de controlar un sensor làser o en anglès un *lidar* o els motors de les rodes, també es podria encarregar de realitzar la planificació de trajectòries o proporcionar un gràfic visual de tot el sistema, etc. Un node està escrit utilitzant llibreries de client de ROS, com les roscpp (C++) o rospy (Python).

- ***Messages*:** Els nodes es comuniquen entre ells enviant-se *messages* o missatges. Un missatge és simplement una estructura composada de dades. Un missatge comprèn una combinació de tipus "primitius" (cadenes de caràcters, Booleans, enters, reals,...) i missatges (un missatge és una estructura recursiva). Per exemple, un node podria representar un servomotor d'un robot que enviaria al sistema la seva posició angular dins d'un missatge com a número enter, la seva temperatura o velocitat com a número real, etc.

- ***Topics*:** Els missatges són conduits via un sistema de "transport" de tipus publicació/subscripció (*publish/subscribe*). Un node envia un missatge publicant-lo a un *topic* en concret. El *topic* és un nom que s'utilitza per identificar el contingut del missatge. Un node que està interessat a un cert tipus de dades es subscriurà al *topic* apropiat. Poden haver-hi múltiples publicacions i subscripcions per a un únic *topic* i un únic node pot publicar i/o subscriure's a múltiples *topics*.

- ***Services*:** El model *publish/subscribe* és un tipus de comunicació molt flexible, però n'és només un i sovint moure informació en una sola direcció no és apropiat per a interaccions de *request/reply*, que acostumen a ser necessàries en sistemes distribuïts. *Request/reply* es fa via els *services* o serveis, que estan definits per un parell d'estructures de missatges: una per la *request* i l'altra per el *reply*. Un node que pugui proporcionar informació pot oferir un servei amb un cert nom i un *client* l'utilitza per enviar el missatge *request* i esperar el *reply*.

- ***Bags*:** Els *bags* són un format per emmagatzemar i reproduir dades de missatges ROS. Els *bags* són un mecanisme molt utilitzat per guardar dades adquirides per sensors per exemple, que poden ser complicades d'obtenir per alguna raó però són necessàries per desenvolupar i testejar algoritmes. A més es poden reproduir tantes vegades com es vulgui i també s'utilitza molt per fer *debugging* d'un sistema després d'un succés.

2.4.2. Eines i llibreries més utilitzades

- **RQT**: Interfície gràfica amb un conjunt d'eines per interactuar o visualitzar dades provinents del robot o de l'algoritme que s'està executant.

- **TF**: Paquet de ROS entre molts altres però amb una gran importància ja que permet a l'usuari localitzar els sistemes de coordenades en el temps. TF manté la relació entre els sistemes de coordenades i les estructures d'arbre (*tree structure*) durant el temps, i permet a l'usuari transformar punts, vectors, etc. entre qualsevol parell de sistemes de coordenades en qualsevol punt que es vulgui en el temps.

- **RViz**: Entorn de visualització 3D que permet a l'usuari veure el que el robot veu, pensa o fa en temps real o no. Hi ha dues maneres principals d'introduir dades dins del món RViz. La primera és que RViz mostri informació de sensors i de l'estat del sistema, com per exemple d'escàners làsers, núvol de punts, càmeres, sistemes de coordenades, etc. La visualització d'aquestes dades és totalment configurable al gust de l'usuari. La segona manera tracta dels "visualization markers", que permeten a l'usuari introduir formes primitives de colors com cubs, fletxes o línies amb total llibertat. La combinació de la visualització de les dades dels sensors i marcadors visuals personalitzats converteix a RViz en una eina molt potent per desenvolupar i fer recerca en el camp de la robòtica.

Figura 14. Visualitzador 3D RViz de ROS (http://sdk.rethinkrobotics.com/wiki/Rviz)

- **Gazebo**: Simulador 3D per a robots essencial en un *toolbox* de robòtica com és ROS. Permet a l'usuari testejar algoritmes, dissenyar robots i realitzar proves utilitzant escenaris realistes.

Figura 15. Simulador 3D Gazebo (http://gazebosim.org/)

- **OpenCV:** Llibreria inclosa a ROS pel processament d'imatge, àmbit fonamental per aplicacions de visió com són la majoria de les que intervenen en robòtica.

- **PointCloudLibrary:** Llibreria inclosa a ROS per a la reconstrucció d'escenaris 3D mitjançant mesures d'escàners làsers. Molt utilitzada per aplicacions SLAM, és a dir, localització i navegació autònoma.

Capítol 3

Desenvolupament del robot Canyonero

En aquest capítol s'explicarà el desenvolupament que s'ha dut a terme per a la creació del robot Canyonero des de zero. Primerament es presentarà l'arquitectura completa del sistema, mostrant el flux de dades entre cada component. Seguidament es comentaran les parts principals del disseny mecànic, passant pel xassís, el sistema motriu i la distribució dels components. També s'analitzaran les parts crítiques de la plataforma, calculant la capacitat de càrrega màxima que poden suportar els motors seleccionats i la que pot suportar la plataforma estructuralment. Finalment s'exposarà el mètode de fabricació de les diferents parts del robot i el seu cost final.

3.1. Arquitectura del sistema

Els requeriments del sistema exigeixen que el robot disposi del seu propi ordinador d'abord amb capacitat de comunicar-se via Wi-Fi amb l'exterior i amb suficients ports USB per connectar-hi perifèrics com una càmera o un Arduino, així que l'ordinador escollit va ser la Raspberry Pi 3. Aquest nou model incorpora mòdul Wi-Fi i compleix els requeriments d'*opensource* i baix cost. Degut a la seva baixa capacitat de computació per a operacions molt complexes com poden ser SLAM (*Simultaneous Localization And Mapping*), hi ha la possibilitat de connectar-se via Wi-Fi amb un altre ordinador, el qual s'ocuparia de realitzar l'operació. D'aquesta manera, la RPi únicament s'ocuparia de l'adquisició i transferència de dades, fent les accions que corresponguin un cop rebuts els resultat de les operacions externalitzades.

Per altra banda, el control dels motors es realitza amb un Arduino Mega, seguint complint amb el requeriment d'*opensource* i baix cost i donant-li més interrupcions al sistema per tal de llegir els encoders que incorporarien els motors. Com que l'Arduino en si no disposa dels *divers* necessaris per controlars motors DC, es necessari afegir la *motor shield* V2 d'Adafruit per tal de controlar els quatre motors de la plataforma.

Finalment, tot i que s'hagués volgut innovar més en aquest aspecte i tenir una qualitat d'imatges més elevada, degut al pressupost i la falta de compatibilitat amb els altres elements del sistema, la càmera encarregada d'adquirir totes les dades referents a visió serà una Kinect de Microsoft, compatible i molt testejada dins de l'entorn ROS.

A la Figura 16 es pot visualitzar un esquema de tot el sistema amb el flux de dades corresponent per cada component. S'entén que les línies discontinues són tipus de connexió *wireless* i la

continues no.

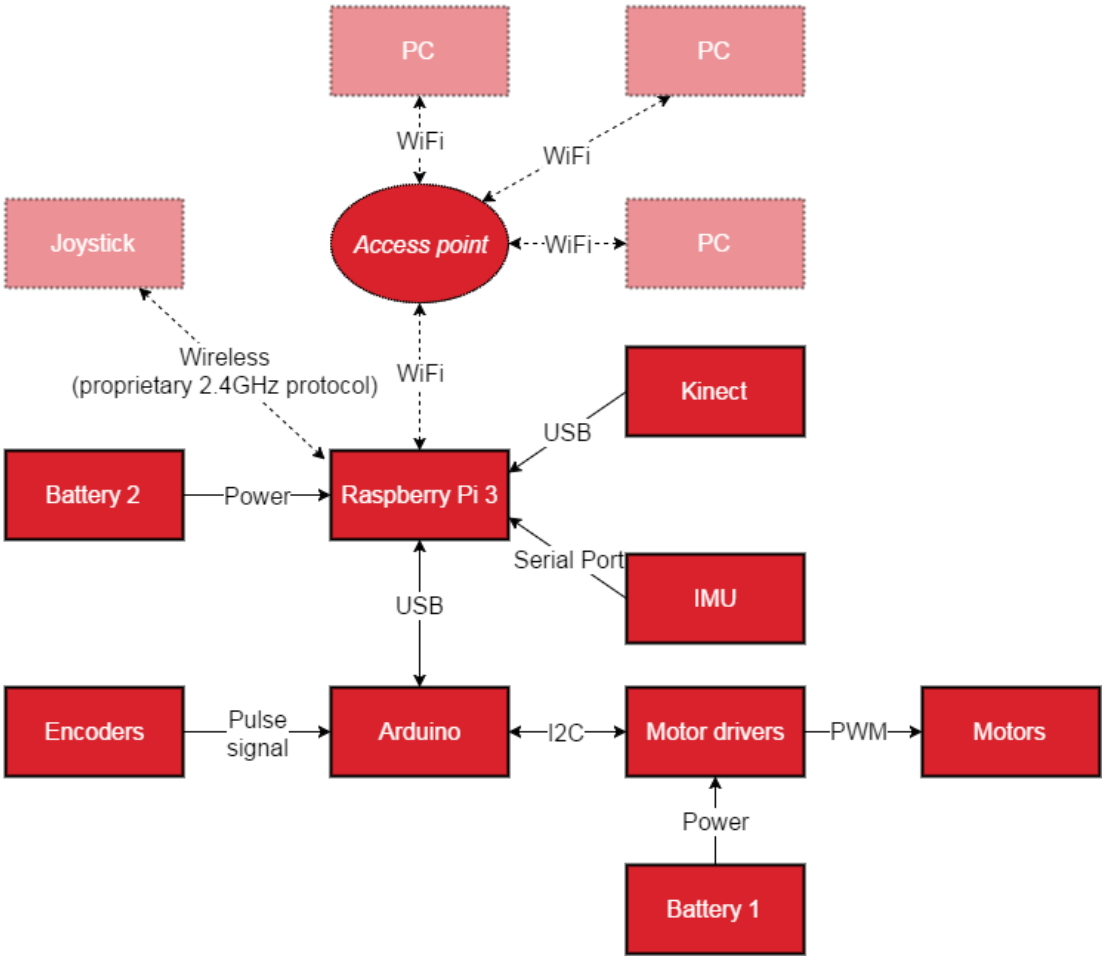

Figura 16. Diagrama de l'arquitectura del sistema del robot Canyonero

3.2. Disseny de la plataforma

El disseny de la plataforma ha estat una de les parts fonamentals del projecte que ha fet que, gràcies a un disseny acurat, acotat i ben revisat fos molt senzill i àgil la fabricació i muntatge del robot resultat.

Els requeriments del disseny, com bé es comenten en l'apartat 1.2 Objectius, van estar molt clars des del principi i era primordial que es complissin per tal d'assegurar l'èxit del projecte. Aquests eren que la plataforma en si fos feta per components de baix cost, fos modular i robusta i molt fàcil de fabricar i muntar, ja que el temps del que es disposava era limitat i es volia evitar que el disseny fos molt complexa i es dificultés assolir altres objectius tan propis d'aquest subprojecte com dels altres, apart de que el màster en robòtica, del qual aquest projecte forma part, està més enfocat a *software* que a mecànica.

Així doncs, a continuació es separa el disseny complet de la plataforma en tres parts principals: el xassís, el sistema motriu i la distribució dels components que formen tot el sistema. Els plànols de totes les peces dissenyades que conformen el disseny es poden trobar a l'Annex I: Plànols.

3.2.1. Xassís

El xassís del robot Canyonero consisteix en una estructura de 400x600mm formada per perfils d'alumini extruïts de 20x20mm de longituds diferents estandarditzats i utilitzats àmpliament a la industria, creant tres nivells o alçades diferents on s'hi podran distribuir els components del sistema al gust de l'usuari gràcies a les guies que tenen als quatre costats. Els perfils estan units entre ells amb juntes impreses en 3D. El material amb el que s'han imprès és plàstic ABS, el qual li dona més resistència a la peça que si s'hagués imprès amb PLA, el plàstic més comú en impressió 3D.

Figura 17. Render del xassís fet amb Fusion 360 d' Autodesk

Les juntes han estat dissenyades expressament per aconseguir la forma que es mostra a la Figura 17, inspirant-se lleugerament amb el xassís del robot Husky de Clearpath. La part inferior està inclinada 45º respecte la part superior per tal de que les rodes del robot topin abans que l'estructura del robot amb possibles obstacles, evitant-los. Gràcies a aquest disseny propi, s'ha aconseguit que el xassís sigui únic i totalment personalitzable i modular, permetent canviar les dimensions únicament modificant les longituds dels perfils. A la Figura 18 es pot veure com es realitzen aquestes unions mitjançant les juntes sense necessitat de fer cap forat als perfils per travessar-los amb cargols.

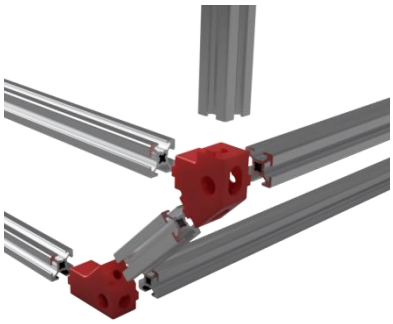

Figura 18. Unions entre perfils i juntes

Així doncs, per assegurar la robustesa de la plataforma, apart de imprimir les juntes amb una alta densitat i parets sòlides ben gruixudes, aconseguint unes juntes gairebé massisses tot i els orificis que tenen interiorment per passar-hi els cargols, els extrems dels perfils estan roscats amb, com a mínim, un recorregut de 25mm perquè siguin units amb cargols *allen* de M6. A més, la gran avantatge d'aquest disseny és que permet muntar el robot sencer únicament amb l'esquelet del xassís i les tapes que es posarien a cada nivell per tal de suportar els components de *hardware*, però tenint la capacitat de guanyar molta més rigidesa afegint cobertes tallades amb làser d'alumini de 2mm a la part inferior per aguantar possibles cops i de metacrilat translúcid de 3mm a la part superior amb el nom del robot i de la universitat gravats, cuidant així també l'estètica.

Finalment s'han afegit un parell de nanses per tal de garantir la portabilitat del robot. Han estat dissenyades expressament perquè es puguin unir amb un parell de cargols de M5 als perfils d'alumini superiors i impreses en 3D amb les mateixes característiques que les juntes perquè siguin capaces de suportar tot el pes de la plataforma.

3.2.2. Sistema motriu

Seguint amb la simplicitat mecànica de tota la plataforma, el sistema motriu s'ha fet de tal manera que produeixi les mínimes complicacions possibles, utilitzant un eix fet a mida amb el torn per donar-li la robustesa que necessita i una transmissió directa 1:1 per no complicar el sistema amb transmissions de cadenes o corretges, evitant així problemes amb destensionaments o desalineacions en la transmissió.

Figura 19. Render del sistema motriu fet amb Fusion 360 d'Autodesk

Com que un del requeriments de la plataforma era que pogués moure's tant per interiors com per exteriors, la selecció de les rodes i el neumàtic en si era important, apart de la del motor. Es volia una plataforma prou gran perquè pogués suportar el *hardware* necessari per realitzar un gran ventall d'aplicacions robòtiques, per això eren necessàries unes rodes d'un diàmetre força gran, per tal de que un motor amb un parell elevat pogués moure càrregues elevades avançant notablement per cada volta gràcies al gran perímetre de la roda. El diàmetre de les rodes són de 200mm, amb càmera d'aire i neumàtic amb dibuix i relleu per tal de poder tenir suficient adherència en diferents tipus de terrenys.

L'avantatge d'unes rodes grans també es que aixequen la plataforma per tal de que puguin passar petits obstacles per sota, però segueixi estan prou baixa per assegurar un centre de gravetat baix. L'inconvenient és que el forat central és molt gran i per tant s'ha hagut de dissenyar un acoblador personalitzat per aquestes rodes, per tal d'adaptar el forat a les dimensions d'un eix de transmissió raonable. Al ser una peça de geometria complexa i totalment feta a mida, també s'ha hagut d'imprimir en 3D amb les mateixes característiques d'impressió que les juntes per assegurar la seva resistència.

Figura 20. Adaptador-acoblador per l'eix i roda

Així doncs, els motors seleccionats pel sistema motriu van ser uns motors DC de 12V de DFRobot amb reductora i encoder incorporat de 663 polsos per volta, amb un parell d'1N·m (10Kg·cm) i capaç d'arribar a 146rpm sense càrrega. Per unir aquest motor a l'estructura es va dissenyar una platina d'alumini de 3mm pel motor que es subjecta tant al primer com segon nivell. I per realitzar la transmissió es va tornejar un eix d'alumini de 20mm per cada roda,

disminuint el seu diàmetre per acoblar-lo a la roda i perquè es restringís el seu moviment longitudinal amb un coixinet de ròtula de 16mm de diàmetre de la marca IGUS.

3.2.3. Distribució dels components

Finalment, gràcies a una estructura que composa tres nivells i dos dels quals ocupen gairebé la totalitat de plataforma (400x600mm) es fa molt senzill distribuir el *hardware* mínim per poder complir els requeriments del sistema.

Els components s'han distribuït de la següent manera: La part més pesada, que són el sistema motriu i la bateria es troben al nivell inferior, mantenint així el punt de gravetat el més avall possible per garantir l'estabilitat del sistema en terrenys complicats. L'ordinador d'abord, que en aquest cas és una Raspberry Pi 3 amb Wi-Fi incorporat i el controlador de motors, que s'ha fet servir un Arduino Mega amb la *motor shield* V2 d'Adafruit, s'han colLocat a la part frontal del nivell central, refugiats de agents externs però accessibles per l'usuari. La part de darrera del nivell central s'ha deixat lliure per tal de que es pugui col·locar un ordinador portàtil per realitzar operacions més complexes on la Raspberry no pogués arribar. I finalment, al nivell superior es troben la càmera, la qual s'ha utilitzat una Kinect de Microsoft i una IMU 9*DOF* de Sparkfun unides per un suport dissenyat expressament per aquests dos components i així saber exactament les seves coordenades respecte l'origen de coordenades.

En la següent figura es pot observar la distribució de tots els components tal i com s'ha explicat en el paràgraf anterior i veure l'aparença final del robot amb totes les cobertes i tots els elements:

Figura 21. Render de l'aparença final del robot Canyonero fet amb Fusion 360 d'Autodesk

3.3. Anàlisi de les parts crítiques de la plataforma

En aquest apartat s'analitzaran les parts més critiques de la plataforma, que venen a ser la càrrega que poden moure els motors amb certes velocitats i acceleracions del robot i la càrrega que pot suportar la plataforma estructuralment. Per calcular aquests dos tipus de càrrega, és necessari fer un càlcul precís del pes total del robot, el qual es presenta a continuació:

3.3.1. Càlcul del pes total de la plataforma

Components

Quantitat	Peça	Pes unitari (g)	Pes total (g)
1	Arduino Mega	37	37
1	Motor shield V2 Adafruit	32	32
1	Raspberry Pi 3	30	30
1	IMU Razor	2	2
1	Kinect	230	230
1	Suport Arduino	15	15
1	Suport raspberry Pi	20	20
1	Suport Kinect + IMU	20	20
2	Nansa	29	58
2	Bateria Tetrix 11,1V 3000mAh	580	1160
1	Cargoleria	1500	1500
		Pes total (g)	**3104**

Taula 2. Llistat de pesos dels components del robot

Part: Sistema motiu

Quantitat	Peça	Pes unitari (g)	Pes total (g)
4	Platina motor	105	420
4	Motor 12V de DFRobot	270	1080
4	Eix acoblador roda motor	100	400
4	Coixinet de ròtula 16mm IGUS	24	96
4	Adaptador roda eix ABS	20	80
4	Roda	500	2000
		Pes total (g)	**4076**

Taula 3. Llistat de pesos del sistema motriu del robot

Part: Xassís

Quantitat	Peça	Pes unitari (g)	Pes total (g)
2	Perfil lateral inferior	180	360
2	Perfil travesser horitzontal inferior	184	369
2	Perfil travesser vertical inferior	203	406
4	Perfil 45	28	112
2	Perfil lateral superior	249	498
3	Perfil travesser horitzontal superior	184	552
4	Perfil travesser vertical superior	263	1052
4	Perfil pilar	46	184
2	Junta 45 oberta esquerra	15	30
2	Junta 45 oberta dreta	15	30
2	Junta 45 pilar esquerra	15	30
2	Junta 45 pilar dreta	15	30
		Pes total (g)	**3653**

Taula 4. Llistat de pesos del xassís del robot

Part: Planxes

Quantitat	Peça	Pes unitari (g)	Pes total (g)
1	Tapa inferior (alumini)	1677	1677
2	Tapa lateral inferior (alumini)	377	754
1	Tapa frontal inferior (alumini)	414	414
1	Tapa darrera inferior (alumini)	426	426
1	Tapa darrera central (metacrilat)	348	348
1	Tapa electrònica central (metacrilat)	347	347
2	Tapa lateral superior (metacrilat)	230	460
1	Tapa frontal superior (metacrilat)	168	168
1	Tapa superior (metacrilat)	235	235
		Pes total (g)	**4829**

Taula 5. Llistat de pesos de les plaques del robot

En resum:

Descripció	Pes (g)
Components	3.474
Sistema motriu	4.076
Xassís	3.652
Planxes	4.829
Pes total: 16.031g	

Taula 6. Resum del càlcul de pesos del robot Canyonero

3.3.2. Càlcul de la càrrega màxima que poden moure els motors

Dades:

$Velocitat\ angular\ del\ motor\ sense\ càrrega\ (\omega) = 146\ rpm = 15{,}23\ rad/s$

$Parell\ del\ motor\ (\tau) = 10\ Kg \cdot cm = 1\ N \cdot m$

$Diàmetre\ de\ la\ roda\ (d) = 20\ cm = 0{,}2\ m$

$Pes\ de\ la\ plataforma\ (massa_{Robot}) = 16\ Kg \cdot gravetat = 157\ N$

$Acceleració\ desitjada\ de\ la\ platadorma\ carregada\ (a_d) = 0 \rightarrow 1\ m/s\ en\ 2s = 0{,}5\ m/s^2$

Càlculs:

$$Velocitat\ lineal\ (v) = \omega \cdot (\frac{d}{2}) = 15{,}23\ [rad/s] \cdot 0{,}1[m] = 1{,}523\ m/s$$

$$Velocitat\ lineal\ desitjada\ de\ la\ plataforma\ carregada\ (v_d) = 1{,}5\ m/s$$

$$Potència\ motor\ (P_{motor}) = \tau \cdot \omega = 1\ [N \cdot m] \cdot 15[rad/s] = 15{,}23\ W$$

$$Potència\ total\ (P_{TOTAL}) = \#_{motors} \cdot P_{motor} = \left(massa_{TOTAL} \cdot a_d \cdot pèrdues\ (30\%)\right) \cdot v_d$$

$$massa_{TOTAL} = \frac{\#_{motors} \cdot P_{motor}}{a_d \cdot pèrdues\ (30\%) \cdot v_d} = \frac{4 \cdot\ 15{,}23\ [W]}{0{,}5\ [m/s^2] \cdot 1{,}30 \cdot\ 1{,}523\ [m/s]} = 61{,}5\ Kg\ aprox.$$

$$Càrrega\ màxima = \frac{massa_{TOTAL} - massa_{Robot}}{Factor\ de\ seguretat} = \frac{61{,}5\ [Kg] - 16\ [Kg]}{1{,}5} = 30\ Kg$$

La càrrega màxima que poden moure els motors és de 30 Kg aproximadament.

La capacitat de càrrega màxima que tindrà la plataforma doncs pot variar molt segons l'acceleració desitjada. A continuació es presenten tres casos més per observar com varia la capacitat de càrrega màxima suportada per la potència dels motors en relació a l'acceleració desitjada de tot el sistema:

Acceleració	Càrrega màxima
Per una acceleració de 0→1m/s en 5s $a_d = 0,2\ m/s^2$	**90 Kg**
Per una acceleració de 0→1m/s en 8s $a_d = 0,125\ m/s^2$	**150 Kg**
Per una acceleració de 0→1m/s en 10s $a_d = 0,1\ m/s^2$	**195 Kg**

Taula 7. Variació de càrrega màxima suportada per la potència dels motors en funció de l'acceleració desitjada

3.3.3. Càlcul del *payload* màxim que pot suportar la plataforma estructuralment

L'estructura tindria principalment dues parts crítiques a remarcar, una seria els eixos dels motors, ja que són curs i de 6mm de diàmetre i les juntes de plàstic que uneixen els perfils del xassís. La part crítica de les juntes s'ha solucionat, apart de imprimint-les amb una alta densitat i un bon gruix de solidesa, col·locant les planxes d'alumini de 3mm com a carrosseria, ajudant a que el xassís guanyi molta resistència i rigidesa. La part dels eixos si que s'haurà d'analitzar més profundament.

L'eix acoblador de la roda al motor s'ha dissenyat de manera que sigui d'un diàmetre prou gran perquè no rebi deformacions degut a la pròpia càrrega del robot o a cop externs. L'eix del motor però, està unit per una banda a l'eix acoblador amb un cargol presoner i per l'altra banda amb el coixinet de la reductora del motor, aquí és on s'ha de tenir en compte l'esforç tallant que pot suportar. Segons les especificacions del fabricant, l'eix del motor pot suportar una càrrega radial de 35N, això equival, tenint en compte l'àrea de la seva secció a un esforç d'uns 0,6MPa.

A continuació es mostren un parell de simulacions que s'han fet amb el Fusion 360 per tal de comprovar els esforços que pateix aquest eix prenent únicament el pes del robot sense càrrega addicional com amb un *payload* dues vegades superior al que pot suportar en el cas de voler les millor condicions esmentades anteriorment, una acceleració de $0,5 m/s^2$, per tal d'afegir un factor de seguretat.

Les simulacions s'han fet afegint restriccions rígides tant al motor com al coixinet on hi rota l'eix acoblador. L'eix del motor està unit amb una unió de revolució amb l'eix acoblador, igual que aquest mateix element amb el coixinet. Així doncs, el primer assaig ha estat de la següent manera:

Dades:

$$Màxima\ càrrega\ radial\ suportada\ per\ l'eixdel\ motor: 35N$$

$$Àrea\ de\ la\ secció\ de\ l'eix: 2\pi \cdot r^2 = 2\pi \cdot (3 \cdot 10^{-3})^2 = 0{,}0000566\ mm^2$$

$$Esforç\ tallant\ que\ pot\ suportar\ cada\ motor: \tau = \frac{F}{A} = \frac{35}{0{,}0000566} = 618895{,}04\ Pa = 0{,}619\ MPa$$

$$pes\ de\ la\ plataforma = massa_{Robot} \cdot gravetat = 16{,}5\ [Kg] \cdot 9.81\ \left[\frac{m}{s^2}\right] = 162\ N$$

$$càrrega\ que\ suporta\ cada\ motor = \frac{pes_{Robot}}{4} = \frac{162}{4} = 40{,}5N$$

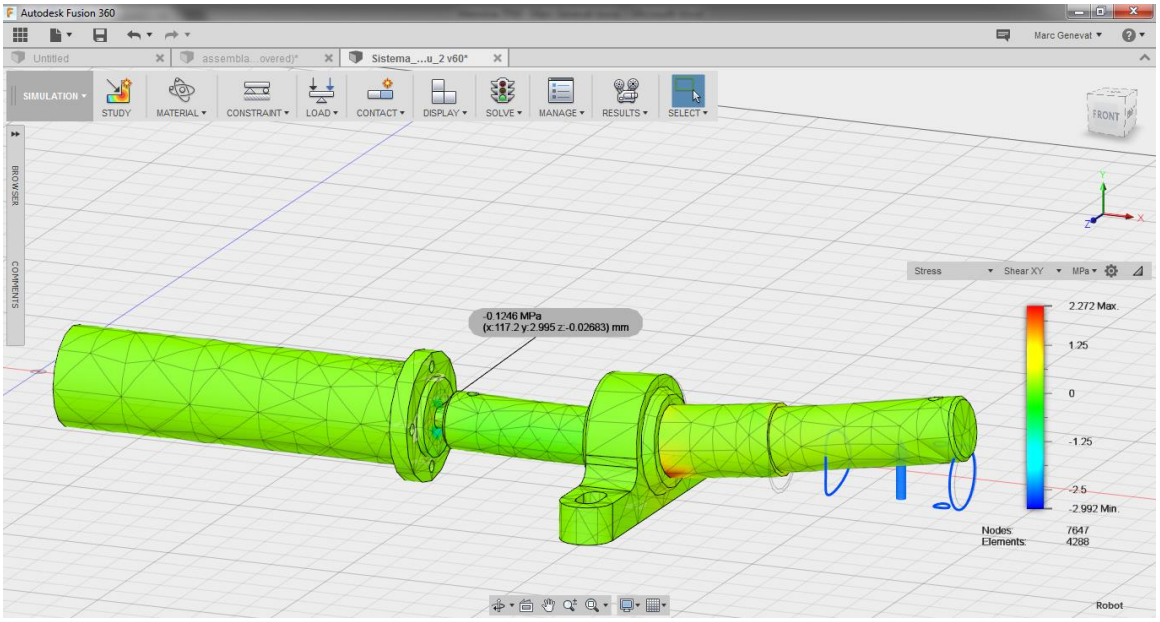

Figura 22. Simulació càrrega radial amb pes plataforma

Els resultats que es mostren a la simulació són els esperats. L'esforç tallant que està suportant l'eix del motor per una càrrega de 40,5N (pes de la plataforma entre 4) és de -0,1246MPa, valor del qual s'ha de mirar en valor absolut i és inferior als 0,619 MPa que teòricament suporta l'eix.

A continuació, es presenta el segon cas, el qual es calcularà amb un *payload* (45Kg) amb què el robot podria tenir una acceleració de 0,5m/^{2} amb un factor de seguretat de 1,5.

$$pes\ de\ la\ plataforma + payload = (massa_{Robot} + payload \cdot factor\ de\ seguretat) \cdot gravetat$$
$$= (16{,}5\ [Kg] + 45[Kg] \cdot 1{,}5) \cdot 9.81\ \left[\frac{m}{s^2}\right] = 824\ N\ aprox.$$

$$càrrega\ que\ suporta\ cada\ motor = \frac{pes_{Robot} + payload}{4} = \frac{824}{4} = 206\ N\ aprox.$$

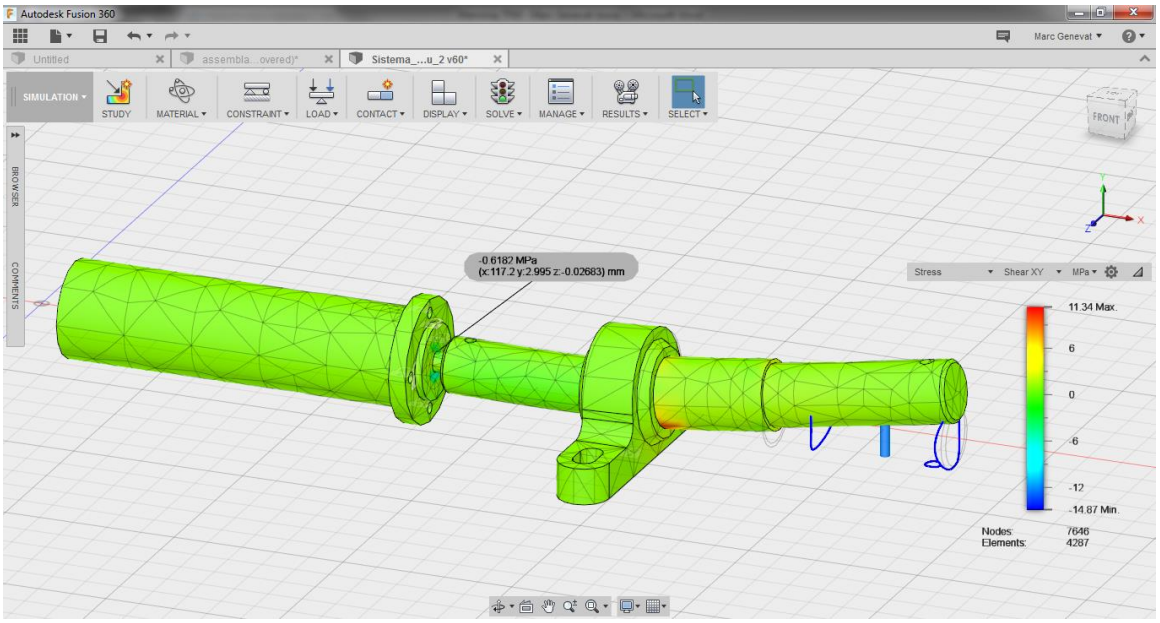

Figura 23. Simulació càrrega radial amb pes plataforma + payload

Al resultat s'observa que amb una càrrega addicional de 45Kg, l'esforç tallant que rep l'eix del motor arriba gairebé al límit del que permet físicament, però tenint en compte que s'ha aplicat un factor de seguretat del 1,5, podem concloure amb què el *payload* màxim que pot carregar estructuralment els motors, tot i sent capaços de moure càrregues molt més grans a acceleracions baixes, és de 45 Kg. Més del doble del que s'havia establert com a requeriment en el projecte, així que aquest resultat és un èxit.

3.4. Cost de fabricació

Un dels requeriments més importants del projecte era que es desenvolupés una plataforma robòtica basada en ROS de baix cost. Per tal de complir aquest aspecte, s'ha anat en compte amb el disseny del robot, simplificant el màxim l'estructura per tal de no utilitzar més elements dels necessaris i s'ha escollit amb cura quins materials caldrien. A més, tots els components de *hardware* són els més populars dins del món *maker* i *opensource* i, per tant, de baix cost.

A continuació es presenta el llistat complet de tots els elements que formen el robot Canyonero amb el seu preu aproximat:

Quantitat	Descripció	Preu unitari	Preu línia
3	Perfiles de montaje 3000mm Aluminio, 20 x 20mm	25,56 €	76,68 €
1	Barra alumini 16mm 2m	20,00 €	20,00 €
1	Planxa alumini 3mm 1000x 3000m	30,00 €	30,00 €
2	Planxa de metacrilat vermell translúcid 60 x 100 cm	32,20 €	64,40 €
1	Cargoleria	30,00 €	30,00 €
1	Bobina ABS Vermell 3mm	18,00 €	18,00 €
4	Motor 12V DFRobot amb encoder	49,27 €	197,08 €
4	Coixinets IGUS 16mm	11,56 €	46,22 €
1	Pack 4 rodes *off-road*	50,00 €	50,00 €
1	Raspberry Pi 3	40,00 €	40,00 €
1	Arduino Mega	30,00 €	30,00 €
1	*Motor shield* V2 Adafruit	20,00 €	20,00 €
1	IMU Razor	70,00 €	70,00 €
1	Kinect	60,00 €	60,00 €
2	Bateria 11,1V 3000mAh	60,00 €	120,00 €
		Preu total	**872,38 €**

Taula 8. Cost de fabricació del robot Canyonero

El preu resultant de gairebé 900€ per un robot d'aquestes dimensions capaç de moure's tant per interior i exterior, basat en ROS i comptant amb sensòrica com odometria, visió 2D i 3D i inercial, fa que sigui un robot molt competitiu pel mercat i que superi els requeriments amb escreix.

Capítol 5

Integració a ROS

Una de les contribucions que ROS ha fet al món de la robòtica que ha suposat un gran avanç ha estat la possibilitat de descriure tot un sistema robotitzat en un fitxer estandarditzat en format XML. Aquest tipus de descripció s'anomena URDF (*Unified or Universal Robot Description Format*). Els robots descrits o modelitzats d'aquesta manera tant poden ser estàtics com dinàmics i és possible introduir totes les propietats físiques del sistema, com ara pesos, inèrcies, col·lisions, etc. Un cop es disposa del model URDF del sistema, és possible visualitzar el seu comportament en funció dels *inputs* que rebi i fins i tot simular-lo en l'entorn Gazebo.

En aquest capítol es veurà com s'ha implementat tant el model URDF com la seva aplicació en Gazebo pel robot Canyonero.

4.1. Modelització del sistema en URDF

Per tal de que el robot Canyonero estigui totalment basat i integrat en ROS, era necessari convertir el seu model 3D en un model URDF, on es podria descriure el comportament de cada element respecte al sistema.

El model del robot Canyonero doncs tindrà certs elements ("*links*") que s'uniran a altres elements de manera fixe ("*fixed joint*"), com per exemple els sensors, que estan en una posició concreta de la plataforma de manera completament estàtica. També tindrà *links* que hauran de ser mòbils respecte la plataforma, les rodes. Aquestes seran unides a la base amb una unió tipus *continuous joint*, que vol diré que l'element pot girar sense restriccions respecte un eix especificat.

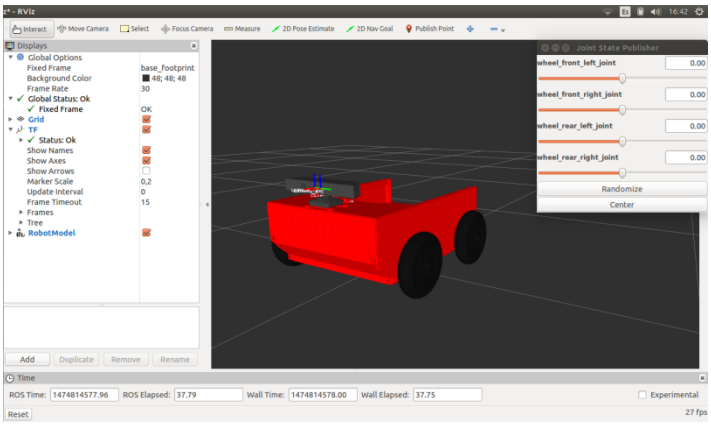

Figura 24. Visualització del model URDF del Canyonero en RViz

L'estructura del URDF és representada en l'esquema de la Figura 25.

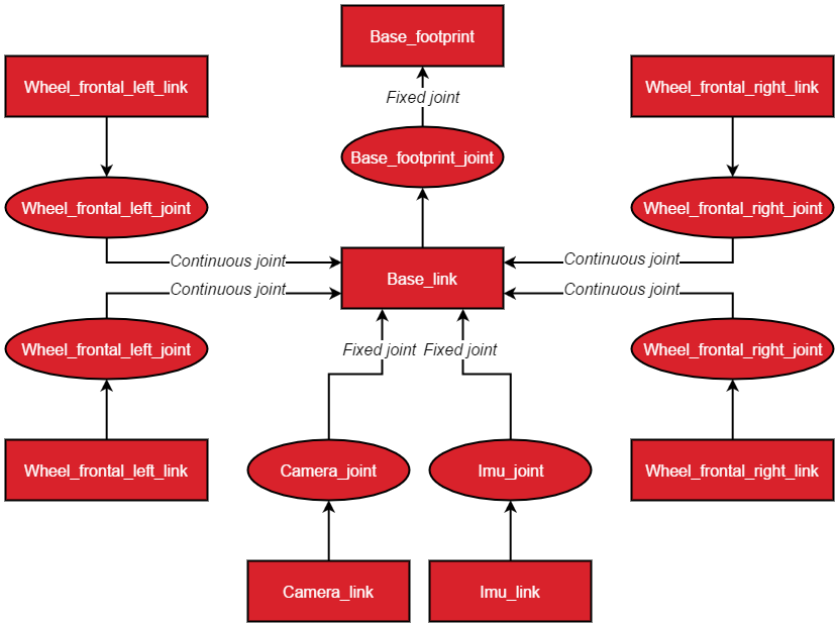

Figura 25. Diagrama del sistema modelitzat en URDF

ROS disposa d'un llenguatge per fer "macros" en XML, anomenat Xacro (XML Macros). Amb Xacro es poden escriure fitxers XML més simplificats i entenedors utilitzant macros expansibles a expressions més grans de XML.

Com que el URDF complet del model del Canyonero era força extens per un sol fitxer, s'ha utilitzat Xacro per tal de col·locar en diferents fitxers cada element i cridar-los després en fitxer "master", tal i com es mostra a l'esquema de la Figura 26. A més, les macros fetes amb Xacro poden disposar de paràmetres per tal de incloure un mateixa macro de diferents maneres en funció dels valors d'aquests paràmetres. Això s'ha fet per les rodes. S'ha fet una macro de *wheels*, anomenada *wheel.urdf.xacro*, amb diferents paràmetres perquè quan s'insereixen les quatre rodes en el fitxer *canyonero_base.urdf.xacro*, es diferenciïn cada element segons els valors dels paràmetres.

Tot el codi implementat per la modelització del sistema en URDF utilitzant Xacros es pot trobar a l'**¡Error! No se encuentra el origen de la referencia.**.

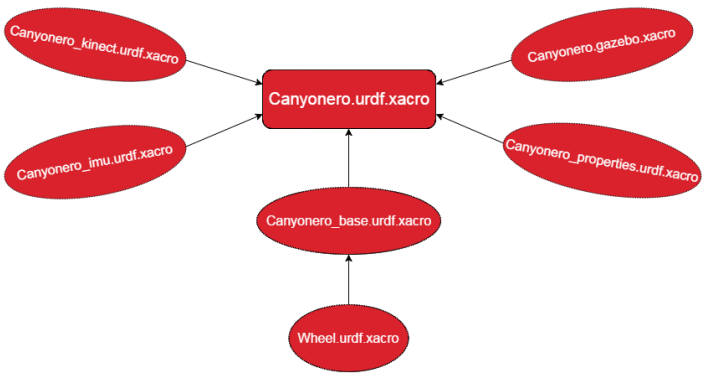

Figura 26. Esquema de l'estructura Xacro que s'ha implementat pel URDF del Canyonero

4.2. Implementació del sistema a l'entorn de simulació Gazebo

Tal i com s'explica a l'apartat 2.4.2 Eines i llibreries més utilitzades, ROS disposa del simulador de robots 3D Gazebo compatible amb els models URDF del qual s'han parlat a anteriorment. Gazebo és doncs una eina molt utilitzada en recerca i en educació a l'hora de testejar i simular el comportament del robot envers als algoritmes que es vagin desenvolupant. Gràcies a una representació ben acurada del sistema robotitzat amb el model URDF del robot, l'usuari és capaç de no posar en perill el seu robot real amb *software* experimental analitzant-lo amb el robot simulat.

Per utilitzar el model URDF en Gazebo, únicament s'ha d'afegir els *plugins* (fitxers on es descriu el comportament de l'element simulat en qüestió a partir de diversos paràmetres) corresponents per cada sensor o actuador que vagi controlat perquè els *links* del URDF actuïn el més similar possible a la realitat. Pel cas del Canyonero, han calgut afegir els *plugins* de la kinect, tant per la càmera RGB com per la de infraroig (IR), el de la IMU i el del control dels motors.

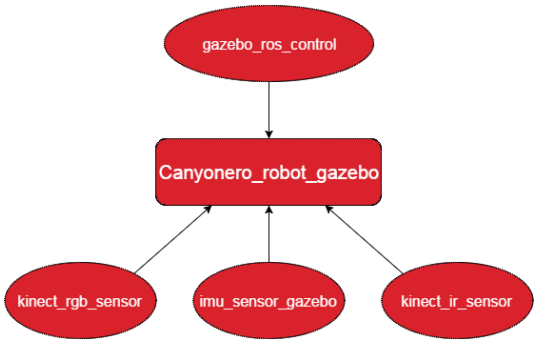

Figura 27. Diagrama de la inserció dels plugins de Gazebo pel model URDF del Canyonero

Dins de Gazebo es poden crear escenaris o *worlds*, on s'hi afegeixen els elements que es vulgui per tal de poder simular el robot en un entorn realista i analitzar com es comporta interactuant amb ells. S'han implementat un parell d'escenaris molt senzills per tal de que el robot Canyonero simulat tingui uns petits espais amb alguns objectes amb què poder testejar els seus algoritmes.

El primer escenari és simplement un sala tancada on el robot podria moure's per dins autònomament o teleoperat, podent testejar algoritmes navegació amb odometria per exemple o detecció de fronteres, etc. I el segon escenari és bàsicament el mateix que el primer però amb tres esferes de colors diferents on el robot podria testejar tant algoritmes de navegació o visió utilitzant la càmera kinect.

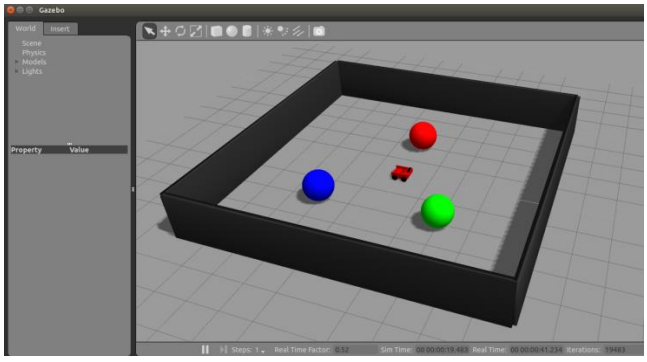

Figura 28. Escenari pràctic 2 implementat a Gazebo pel Canyonero

Conclusions

A l'acabar el projecte es poden extreure forces conclusions del tot el procediment i el resultat. Com bé s'indica a la introducció, el projecte Canyonero va ser des de l'inici un projecte molt ambiciós que requeria formar un equip de com a mínim tres persones per portar-lo a terme i assegurar uns bons resultats. Així doncs, comparant els requeriments presentats a l'apartat 1.2 Objectius amb el resultat final del projecte, es pot afirmar que aquesta divisió de tasques era necessària i a més ha estat eficaç, ja que s'han complet la major part dels objectius del projecte i dels requeriments que havia de tenir la plataforma. Tot seguit els repassem:

Figura 29. Render final del robot Canyonero fet amb Fusion 360 d'Autodesk

Pel que fa a la part mecànica, la plataforma s'ha dissenyat i construït amb menys de dos mesos per tal de que s'arribés a complir tots els demés objectius. Els materials utilitzats, tot i que la carrosseria no ha acabat sent ni hermètica ni impermeable per garantir una bona accessibilitat al *hardware*, han permès que el robot pesi menys de 20 Kg, acabant pesant 16,5 Kg per ser exactes i mesuri uns 400x600x300mm, tenint la capacitat de ser modular gràcies a les juntes que s'han dissenyat. El seu moviment és diferencial amb tracció a les quatre rodes i capaç de moure's tant per interiors com per exteriors sense cables. S'han integrat la majoria dels sensors que es requerien per tenir una plataforma preparada per la recerca i l'educació, deixant espai per ampliar encara més la sensòrica i podent suportar càrregues de fins a 45 Kg. A més, l'autonomia del robot pot arribar a 45 minuts amb la bateria pressupostada però sent perfectament ampliable.

A nivell de ROS, s'ha implementat tot el que es pretenia des del principi. S'ha aconseguit acabar el projecte disposant de dos paquets ROS on es pot trobar la modelització del Canyonero en URDF i la implementació d'aquest amb els corresponents *plugins* per simular-lo en Gazebo.

Però un dels objectius que s'ha assolit i per al qual es pot estar molt orgullós és el cost de fabricació de la plataforma. Ja que gràcies a un disseny simple però robust, a les noves tecnologies de fabricació i a components de *hardware opensoure* i de baix cost, s'ha aconseguit que el cost final d'elaboració del Canyonero sigui inferior a 900 €, un cost molt més baix que el que s'indicava com a requeriment (2.000 €) i que els robots comercials actuals basats en ROS.

Per concloure, dir que tot el procés de desenvolupament del robot Canyonero ha estat intens, amb alts i baixos, però molt gratificant i educatiu. El projecte era tot un repte pels autors, però la il·lusió i les ganes, el treball en equip, la bona organització i el pacient i instructiu seguiment que s'ha fet des dels dos tutors del projecte han fet que resulti sent tot un èxit.

Línies de futur

Una vegada exposades les conclusions, es presentaran certes millores o futures consideracions a tenir en compte si es vol continuar treballant amb el robot Canyonero.

El Canyonero ha acabat sent, com es requeria, un robot basat en ROS de baix cost i estructuralment molt robust. Tant és així, que se'n podria treure profit de tot això i millorar les seves prestacions.

En primer lloc, hi hauria dues opcions per millorar l'estructura. Una seria si es volgués fer-la més lleugera, per exemple reduint el gruix de les planxes d'alumini, ja que és la part més pesada del robot i amb planxes de 2mm potser seria suficient per protegir la part inferior del robot i donar-li rigidesa al xassís. I l'altra que fos encara més robusta, canviant les planxes de metacrilat per exemple per unes d'alumini, que són les que poden percebre ruptures quan hagin de carregar *payloads* de pes elevats, canviant els motors per uns amb una major capacitat de càrrega radial i fer les juntes completament sòlides o d'algun material que no sigui ABS, com materials híbrids que comencen a fer-se servir en impressió 3D amb partícules de fibra de carboni per exemple.

En segon lloc, com que el cost de fabricació ha estat molt baix finalment, es podria augmentar una mica el pressupost per millorar la part de *hardware*, aconseguint així una major capacitat de computació i una adquisició de dades de millor qualitat. Per exemple, la Raspberry Pi 3 es podria canviar per un dels ordinadors de l'estat de l'art actual en pc *embedded*s un NUC Skull Canyonn amb router WiFi de 5GHz i la kinect, que ja ha deixat de fabricar-se, per la nova càmera 3D de Intel, la RealSense (R200), molt més lleugera, de dimensions molt reduïdes, sense problemes amb la llum solar i amb qualitat molt més elevada.

Per acabar, gràcies a les seves dimensions considerables i la capacitat de càrrega que té, es poden afegir més elements, com ara un sensor GPS, càmeres d'altres tipus o fins i tot un braç robòtic per tal de fer aplicacions de manipulació i introduir-se també al programa MoveIt. Per exemple es podria incorporar el nou braç de BCN3D, el BCN3D Moveo. Apart, pel que fa al *software*, es podrien crear més escenaris en Gazebo per realitzar diferents tipus de pràctiques. Tot per millorar l'experiència de l'usuari tant sigui en educació com en recerca.

Referències

http://www.ros.org/about-ros/

http://wiki.ros.org/

http://www.generationrobots.com/blog/en/2016/03/ros-robot-operating-system-2/

http://robohub.org/ros-101-intro-to-the-robot-operating-system/

https://erlerobotics.com/docs/Robot_Operating_System/ROS/Basic_concepts/index.html

http://www.clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/

https://www.clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/

http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx

http://www.mobilerobots.com/ResearchRobots/P3AT.aspx

http://www.innok-robotics.de/en/products/heros

http://www.innok-robotics.de/en/products/tx

https://www.enovarobotics.eu/index.php/products/mini-lab

https://www.enovarobotics.eu/index.php/products/pearlguard

http://www.robotnik.es/robots-moviles/summit-xl/

http://www.robotnik.es/robots-moviles/turtlebot-2/

http://www.autodesk.com/products/fusion-360/learn-training-tutorials

http://ingemecanica.com/tutorialsemanal/tutorialn63.html

http://gazebosim.org/tutorials/?tut=ros_urdf

Marghitu, D. (2001). *Mechanical engineer's handbook*. San Diego, CA: Academic Press.

Annex I: Plànols

Annex II: Codi

canyonero_properties.urdf.xacro

```
<?xml                                                                    version="1.0"?>
<!--

 This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs
3.0                                 Unported                                License.
 To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-
nd/3.0/             or             send            a            letter            to
 Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

-->

<robot                                        xmlns:xacro="http://ros.org/wiki/xacro">

       <!--                    common                     properties                    -->
       <xacro:property name="PI"    value="3.1415926535897931"/>
       <material                                                              name="Red">
   <color              rgba="0.8              0.0              0.0              1.0"/>
 </material>
 <material                                                                  name="Black">
   <color              rgba="0.05             0.05             0.05             1.0"/>
 </material>
 <material                                                                name="DarkGrey">
  <color               rgba="0.2              0.2              0.2              1.0"/>
</material>

       <!--                    base                     properties                    -->
       <!--       robot's       elevation       from       the       ground       -->
       <xacro:property                name="rob_pos_Z"                  value="0.167"/>
       <!--                    wheel                    properties                    -->
       <xacro:property                name="wheel_pos_X"                value="0.162"/>
       <xacro:property                name="wheel_pos_Y"                value="0.165"/>
       <xacro:property                name="wheel_pos_Z"                value="0.068"/>
       <xacro:property                 name="wheel_torque"                value="1.0"/>
       <xacro:property               name="wheel_velocity"               value="3.0"/>
       <xacro:property                name="wheel_radius"                value="0.20"/>

       <!--                    kinect                     properties                    -->
       <!--        kinect        position        at        the        robot        -->
       <xacro:property           name="cam_pos_X"           value="0.262"           />
       <xacro:property            name="cam_pos_Y"            value="0.0"            />
       <xacro:property           name="cam_pos_Z"           value="0.141"           />
       <xacro:property      name="cam_or_R"      value="${PI      /      2}"      />
       <xacro:property            name="cam_or_P"            value="0.0"            />
       <xacro:property      name="cam_or_Y"      value="${PI      /      2}"      />
       <!--     kinect     cam     position     at     the     kinect     itself     -->
       <xacro:property            name="cam_px"           value="0.029"           />
       <xacro:property            name="cam_py"           value="-0.01"           />
       <xacro:property            name="cam_pz"           value="0.019"           />
       <xacro:property             name="cam_or"            value="0.0"            />
       <xacro:property             name="cam_op"            value="0.0"            />
       <xacro:property             name="cam_oy"            value="0.0"            />

       <!--                    imu                     properties                    -->
       <xacro:property                name="imu_pos_X"               value="0.1885"/>
       <xacro:property                 name="imu_pos_Y"                  value="0.0"/>
       <xacro:property                name="imu_pos_Z"                value="0.108"/>
       <xacro:property                 name="imu_or_R"                  value="0.0"/>
       <xacro:property                 name="imu_or_P"                  value="0.0"/>
       <xacro:property        name="imu_or_Y"        value="${-PI        /        2}"/>

</robot>
```

canyonero_base.urdf.xacro

```
<?xml version="1.0"?>
<!--

 This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.
 To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-nd/3.0/ or send a letter to
 Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

-->

<robot xmlns:xacro="http://ros.org/wiki/xacro">

       <xacro:include filename="$(find canyonero_description)/urdf/wheels/canyonero_wheel.urdf.xacro"/>

       <xacro:macro name="wheeled_base" params="name">
       <!-- BASE-FOOTPRINT -->
              <!-- base_footprint is a fictitious link(frame) that is on the ground right below base_link origin, navigation stack depends on this frame -->
              <link name="${name}_footprint"/>

              <joint name="${name}_footprint_joint" type="fixed">
                     <parent link="${name}_footprint"/>
                     <child link="${name}_link"/>
                     <origin xyz="0 0 ${rob_pos_Z}" rpy="0 0 0"/>
              </joint>

              <!-- BASE-LINK -->
          <!--Actual body/chassis of the robot-->
              <link name="${name}_link">
                     <visual>
                            <origin xyz="0 0 0" rpy="0 0 0"/>
                            <geometry>
                                   <mesh filename="package://canyonero_description/meshes/canyonero_plates.stl"/>
                            </geometry>
                            <material name="Red"/>
                     </visual>
                     <collision>
                     <origin xyz="0 0 0" rpy="0 0 0"/>
                     <geometry>
                     <mesh filename="package://canyonero_description/meshes/canyonero_plates_collision.stl"/>
                     </geometry>
              </collision>
                     <inertial>
                     <origin xyz="0.023661 -0.000069 -0.036285" rpy="0 0 0" />
                     <mass value="10"/>
                     <inertia ixx="0.27658267" ixy="0.0" ixz="-0.02663389"
                            iyy="0.38926449" iyz="0.0"
                            izz="0.57775046"/>
          </inertial>
              </link>

              <gazebo reference="${name}_link">
     <material value="Gazebo/Red"/>
   </gazebo>

              <!-- Wheels -->
           <xacro:wheel sideX = "front" sideY = "left" positionX = "1.0" positionY = "1.0" reflect = "-1.0" parent = "${name}" torque="${wheel_torque}"
```

```
velocity="${wheel_velocity}"                radius="${wheel_radius}"                />
          <xacro:wheel sideX = "front" sideY = "right" positionX = "1.0" positionY = "-
1.0"    reflect    =    "1.0"    parent    =    "${name}"    torque="${wheel_torque}"
velocity="${wheel_velocity}"                                    radius="${wheel_radius}"/>
          <xacro:wheel sideX = "rear" sideY = "left" positionX = "-1.0" positionY =
"1.0"    reflect    =    "-1.0"    parent    =    "${name}"    torque="${wheel_torque}"
velocity="${wheel_velocity}"                                    radius="${wheel_radius}"/>
          <xacro:wheel sideX = "rear" sideY = "right" positionX = "-1.0" positionY = "-
1.0"    reflect    =    "1.0"    parent    =    "${name}"    torque="${wheel_torque}"
velocity="${wheel_velocity}"                                    radius="${wheel_radius}"/>

       </xacro:macro>

</robot>
```

canyonero_wheel.urdf.xacro

```
<?xml                                                                    version="1.0"?>
<!--

 This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs
3.0                                Unported                                License.
 To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-
nd/3.0/             or             send             a             letter             to
 Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

-->

<robot              xmlns:xacro              =              "http://ros.org/wiki/xacro">

       <xacro:macro name="wheel" params="sideY sideX positionY positionX reflect parent
torque                                velocity                                radius">
          <link                                      name="wheel_${sideX}_${sideY}_link">
              <visual>
              <origin    xyz    =    "0    0    0"    rpy    =    "0    ${-PI    /    2}    0"/>
              <geometry>
                            <mesh
filename="package://canyonero_description/meshes/canyonero_wheel.stl"/>
              </geometry>
              <material                                                    name="Black"/>
              </visual>
                 <inertial>
                     <origin     xyz="0.0     0.0     0.102638"     rpy="0     0     0"/>
                     <mass                                                value="1.419"/>
                     <inertia         ixx="0.00305"         ixy="0.0"         ixz="0.0"
                                           iyy="0.00305"                        iyz="0.0"
                                      izz="0.00512"/>

                 </inertial>
              <collision>
              <origin     xyz="0     0     0"     rpy="0     ${-PI     /     2}     0"/>
              <geometry>
                            <mesh
filename="package://canyonero_description/meshes/canyonero_wheel_collision.stl"/>
              </geometry>
              </collision>
          </link>

          <joint             name="wheel_${sideX}_${sideY}_joint"
```

```
type="continuous">
              <parent                                          link="${parent}_link"/>
              <child                                  link="wheel_${sideX}_${sideY}_link"/>
              <origin  xyz="${wheel_pos_X  *  positionX}  ${wheel_pos_Y  *  positionY}  ${-
wheel_pos_Z}"       rpy="${(PI       /       2)       *       reflect}       0       0"/>
              <axis                    xyz="0                    0                    1"/>
                                      <limit   effort="${torque}"   velocity="${velocity   /
radius}"/>
          </joint>

                     <transmission                                    name="simple_trans">
                      <type>transmission_interface/SimpleTransmission</type>
                      <joint                        name="wheel_${sideX}_${sideY}_joint">

       <hardwareInterface>VelocityJointInterface</hardwareInterface>
                      </joint>
                      <actuator                                              name="DC_motor">
                                   <mechanicalReduction>1</mechanicalReduction>
                      </actuator>
                     </transmission>

                     <gazebo                  reference="wheel_${sideX}_${sideY}_link">
                            <mu1                                             value="0.75"/>
                            <mu2                                             value="0.75"/>
                            <!--kp               value="10000000.0"                  /-->
                            <!--kd                   value="1.0"                     /-->
                            <fdir1             value="0             0              0"/>
                            <material                              value="Gazebo/Black"/>
                     </gazebo>

       </xacro:macro>
</robot>
```

canyonero_imu.urdf.xacro

```
<?xml                                                                  version="1.0"?>
<!--

 This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs
3.0                                Unported                                    License.
 To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-
nd/3.0/             or             send             a             letter             to
 Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

-->
<robot                                          xmlns:xacro="http://ros.org/wiki/xacro">

 <xacro:macro                     name="imu_sensor"                     params="parent">

   <joint                         name="imu_joint"                          type="fixed">
     <origin  xyz  =  "${imu_pos_X}  ${imu_pos_Y}  ${imu_pos_Z}"  rpy  =  "${imu_or_R}
${imu_or_P}                                                                ${imu_or_Y}"/>
     <parent                                                             link="${parent}"/>
     <child                                                               link="imu_link"/>
   </joint>
   <link                                                                  name="imu_link">
     <visual>
       <origin     xyz     =     "0     0     0"     rpy     =     "0     0     0"     />
       <geometry>
         <mesh  filename  =  "package://canyonero_description/meshes/9dof_razor_imu.stl"
scale="0.1                                  0.1                                    0.1"/>
       </geometry>
```

```
      <material                                                    name="Red"/>
    </visual>
    <collision>
      <origin       xyz="0.0       0.0       0.0"       rpy="0       0       0"/>
      <geometry>
        <box                 size="0.27                0.40                0.5"/>
      </geometry>
    </collision>
    <inertial>
      <mass                                                      value="0.01"/>
      <origin                     xyz="0                     0                   0"/>
      <inertia   ixx="0.0000001"   ixy="0.0"   ixz="0.0"   iyy="0.0000001"   iyz="0.0"
izz="0.0000001"/>
    </inertial>
  </link>
 </xacro:macro>

</robot>
```

canyonero_kinect.urdf.xacro

```
<?xml                                                          version="1.0"?>
<!--

 This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs
3.0                                Unported                                License.
 To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-
nd/3.0/            or            send            a            letter            to
 Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

-->
<robot          name="sensor_kinect"          xmlns:xacro="http://ros.org/wiki/xacro">

      <!-- Parameterised in part by the values in canyonero_properties.urdf.xacro -->
 <xacro:macro                   name="sensor_kinect"                  params="parent">
      <joint                     name="camera_joint"                     type="fixed">
            <origin  xyz="${cam_pos_X}  ${cam_pos_Y}  ${cam_pos_Z}"  rpy="0  0  0"/>
            <parent                                                 link="${parent}"/>
            <child                                                 link="camera_link"/>
      </joint>
      <link                                                         name="camera_link">
    <visual>
     <origin    xyz="0    0    0"    rpy="${cam_or_R}    ${cam_or_P}    ${cam_or_Y}"/>
      <geometry>
       <mesh  filename="package://canyonero_description/meshes/kinect.stl"  scale="0.1
0.1                                                                          0.1"/>
      </geometry>
      <material                          name="DarkGrey"                          />
    </visual>
       <collision>
      <origin       xyz="0.0       0.0       0.0"       rpy="0       0       0"/>
          <geometry>
            <box             size="0.07271             0.27794             0.073"/>
          </geometry>
        </collision>
    <inertial>
      <mass                                value="0.564"                              />
      <origin               xyz="0               0               0"               />
      <inertia            ixx="0.003881243"            ixy="0.0"            ixz="0.0"
              iyy="0.000498940"                                          iyz="0.0"
              izz="0.003879257"                                                    />
    </inertial>
```

```
    <gazebo                                                    reference="camera_link">
      <material>Gazebo/Black</material>
    </gazebo>
  </link>

  <!--                      RGB                        camera                      -->
  <joint                    name="camera_rgb_joint"                  type="fixed">
    <origin xyz="${cam_px} ${cam_py} ${cam_pz}" rpy="${cam_or} ${cam_oy} ${cam_op}"/>
    <parent                                                  link="camera_link"/>
    <child                        link="camera_rgb_frame"                       />
  </joint>
  <link                                                 name="camera_rgb_frame"/>

  <joint                name="camera_rgb_optical_joint"                type="fixed">
    <origin      xyz="0      0      0"      rpy="${-PI/2}      0      ${-PI/2}"      />
    <parent                        link="camera_rgb_frame"                       />
    <child                      link="camera_rgb_optical_frame"                   />
  </joint>
  <link                                           name="camera_rgb_optical_frame"/>

  <!--                     Depth                       camera                     -->
  <joint                    name="camera_depth_joint"                type="fixed">
            <origin    xyz="0    ${2.4    *    -cam_py}    0"    rpy="0    0    0"    />
            <parent                    link="camera_rgb_frame"                    />
            <child                   link="camera_depth_frame"                    />
            </joint>
            <link                                          name="camera_depth_frame"/>

            <joint          name="camera_depth_optical_joint"          type="fixed">
            <origin     xyz="0     0     0"     rpy="${-PI/2}     0     ${-PI/2}"     />
            <parent                  link="camera_depth_frame"                    />
            <child               link="camera_depth_optical_frame"                />
            </joint>
            <link                                  name="camera_depth_optical_frame"/>

      </xacro:macro>
</robot>
```

canyonero.urdf.xacro

```
<?xml                                                              version="1.0"?>
<!--

 This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs
3.0                                Unported                                License.
 To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-
nd/3.0/             or             send             a             letter             to
 Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

-->

<robot            xmlns:xacro="http://ros.org/wiki/xacro"            name="canyonero">

            <xacro:include                                          filename="$(find
canyonero_description)/urdf/canyonero_properties.urdf.xacro"/>"
            <xacro:include                                          filename="$(find
canyonero_description)/urdf/canyonero_base.urdf.xacro"/>
            <xacro:include                                          filename="$(find
canyonero_description)/urdf/sensors/canyonero_kinect.urdf.xacro"/>
            <xacro:include                                          filename="$(find
```

```
canyonero_description)/urdf/sensors/canyonero_imu.urdf.xacro"/>
              <xacro:include                                          filename="$(find
canyonero_gazebo)/urdf/canyonero.gazebo.xacro"/>

              <!--                                Base                              -->
              <xacro:wheeled_base                                       name="base"/>

              <!--                    3D                    sensor                    -->
              <xacro:sensor_kinect                                   parent="base_link"/>

              <!--                                IMU                               -->
              <xacro:imu_sensor                                      parent="base_link"/>

              <!--                               Gazebo                             -->
              <xacro:canyonero_robot_gazebo/>

</robot>
```

show.launch

```
<launch>

 <!--                Load                your                robot                -->
 <arg                          name="robot"                          default="canyonero"/>

 <param     name="robot_description"     command="$(find     xacro)/xacro.py     '$(find
canyonero_description)/urdf/canyonero.urdf.xacro'"                                    />

 <node             name="robot_state_publisher"             pkg="robot_state_publisher"
type="robot_state_publisher"                                                          />
 <node             name="joint_state_publisher"             pkg="joint_state_publisher"
type="joint_state_publisher"                  args="_use_gui:=True"                   />

 <node       name="rviz"       pkg="rviz"       type="rviz"       args="-d       $(find
canyonero_description)/config/show.rviz"                                              />

</launch>
```

imu.gazeboxacro

```
<?xml                                                                  version="1.0"?>

<robot                                        xmlns:xacro="http://ros.org/wiki/xacro">

       <xacro:macro  name="imu_sensor_gazebo"  params="link_name  imu_topic  update_rate">
              <gazebo>
                     <plugin                                name="${link_name}_controller"
filename="libgazebo_ros_imu.so">
                           <alwaysOn>true</alwaysOn>
                           <updateRate>${update_rate}</updateRate>
                           <bodyName>${link_name}</bodyName>
                           <topicName>${imu_topic}</topicName>
                           <gaussianNoise>0.0</gaussianNoise>
                           <xyzOffset>0                0                0</xyzOffset>
                           <rpyOffset>0                0                0</rpyOffset>
                           <serviceName>/default_imu</serviceName>
```

```
                </plugin>
        </gazebo>

        <!-- Imu definition required for hardware interface of ros control -->
        <gazebo                                         reference="${link_name}">
                <sensor             name="9dof_razor_imu"             type="imu">
                        <always_on>1</always_on>
                        <update_rate>${update_rate}</update_rate>
                        <imu>
                                <noise>
                                        <type>gaussian</type>
                                        <rate>
                                <mean>0.0</mean>
                                <stddev>2e-4</stddev>
                                <bias_mean>0.0000075</bias_mean>
                                <bias_stddev>0.0000008</bias_stddev>
                            </rate>
                                        <accel>
                                <mean>0.0</mean>
                                <stddev>1.7e-2</stddev>
                                <bias_mean>0.1</bias_mean>
                                <bias_stddev>0.001</bias_stddev>
                            </accel>
                                </noise>
                        </imu>
                </sensor>
                <material                                       value="Gazebo/Red"/>
        </gazebo>
    </xacro:macro>

</robot>
```

kinect.gazebo.xacro

```
<?xml                                                           version="1.0"?>
<!--

 This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs
3.0                                 Unported                                License.
 To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-
nd/3.0/            or            send            a            letter            to
 Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

-->
<robot                                          xmlns:xacro="http://ros.org/wiki/xacro">

    <xacro:macro name="kinect_ir_sensor" params="link_name frame_name camera_name">
        <gazebo                                         reference="${link_name}">
                <sensor        type="depth"        name="${link_name}_ir_sensor">
                        <always_on>true</always_on>
                        <update_rate>1.0</update_rate>
                        <camera>
                                <horizontal_fov>${57.0*PI/180.0}</horizontal_fov>
                                <image>
                                        <format>L8</format>
                                        <width>640</width>
                                        <height>480</height>
                                </image>
                                <clip>
                                        <near>1.2</near>
                                        <far>3.5</far>
                                </clip>
```

```
                        </camera>
                        <plugin                         name="${link_name}_controller"
filename="libgazebo_ros_openni_kinect.so">
                                <baseline>0.2</baseline>
                                <alwaysOn>true</alwaysOn>
                                <updateRate>1.0</updateRate>
                                <cameraName>${camera_name}_ir</cameraName>

        <imageTopicName>/${camera_name}/depth/image_raw</imageTopicName>

        <cameraInfoTopicName>/${camera_name}/depth/camera_info</cameraInfoTopicName>

        <depthImageTopicName>/${camera_name}/depth/image_raw</depthImageTopicName>

        <depthImageCameraInfoTopicName>/${camera_name}/depth/camera_info</depthImageCame
raInfoTopicName>

        <pointCloudTopicName>/${camera_name}/depth/points</pointCloudTopicName>
                                <frameName>${frame_name}</frameName>
                                <pointCloudCutoff>0.5</pointCloudCutoff>
                                <distortionK1>0.00000001</distortionK1>
                                <distortionK2>0.00000001</distortionK2>
                                <distortionK3>0.00000001</distortionK3>
                                <distortionT1>0.00000001</distortionT1>
                                <distortionT2>0.00000001</distortionT2>
                                <CxPrime>0</CxPrime>
                                <Cx>0</Cx>
                                <Cy>0</Cy>
                                <focalLength>0</focalLength>
                                <hackBaseline>0</hackBaseline>
                        </plugin>
                </sensor>
                <material                                       value="Gazebo/Black"/>
        </gazebo>
</xacro:macro>

<xacro:macro name="kinect_rgb_sensor" params="link_name frame_name camera_name">
        <gazebo                                                 reference="${link_name}">
                <sensor         type="depth"        name="${link_name}_rgb_sensor">
                        <always_on>true</always_on>
                        <update_rate>1.0</update_rate>
                        <camera>
                                <horizontal_fov>${57.0*PI/180.0}</horizontal_fov>
                                <image>
                                        <format>R8G8B8</format>
                                        <width>640</width>
                                        <height>480</height>
                                </image>
                                <clip>
                                        <near>1.2</near>
                                        <far>3.5</far>
                                </clip>
                        </camera>
                        <plugin                         name="${link_name}_controller"
filename="libgazebo_ros_openni_kinect.so">
                                <alwaysOn>true</alwaysOn>
                                <updateRate>1.0</updateRate>
                                <cameraName>${camera_name}_rgb</cameraName>

        <imageTopicName>/${camera_name}/rgb/image_raw</imageTopicName>

        <cameraInfoTopicName>/${camera_name}/rgb/camera_info</cameraInfoTopicName>

        <depthImageTopicName>/${camera_name}/depth/image_raw</depthImageTopicName>

        <depthImageCameraInfoTopicName>/${camera_name}/depth/came
```

```
ra_info</depthImageCameraInfoTopicName>

       <pointCloudTopicName>/${camera_name}/depth_registered/points</pointCloudTopicNam
e>
                                    <frameName>${frame_name}</frameName>
                                    <pointCloudCutoff>0.5</pointCloudCutoff>
                                    <distortionK1>0.00000001</distortionK1>
                                    <distortionK2>0.00000001</distortionK2>
                                    <distortionK3>0.00000001</distortionK3>
                                    <distortionT1>0.00000001</distortionT1>
                                    <distortionT2>0.00000001</distortionT2>
                                    <CxPrime>0</CxPrime>
                                    <Cx>0</Cx>
                                    <Cy>0</Cy>
                                    <focalLength>0</focalLength>
                                    <hackBaseline>0</hackBaseline>
                                </plugin>
                        </sensor>
                        <material                                    value="Gazebo/Black"/>
                </gazebo>
       </xacro:macro>
</robot>
```

canyonero.gazebo.xacro

```
<?xml                                                                version="1.0"?>
<!--

 This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs
3.0                                 Unported                                License.
 To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-
nd/3.0/             or             send            a            letter            to
 Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

-->
<robot                                          xmlns:xacro="http://ros.org/wiki/xacro">
 <xacro:include  filename="$(find  canyonero_gazebo)/urdf/sensors/kinect.gazebo.xacro"/>
 <xacro:include   filename="$(find   canyonero_gazebo)/urdf/sensors/imu.gazebo.xacro"/>

 <xacro:macro                                             name="canyonero_robot_gazebo">
   <!--         Gazebo         plugin         for         ROS         Control         -->
   <gazebo>
     <plugin       name="gazebo_ros_control"       filename="libgazebo_ros_control.so">
       <robotNamespace>canyonero</robotNamespace>
     </plugin>
   </gazebo>

   <!--       Gazebo       plugin       for       Kinect       IR       sensor       -->
   <xacro:kinect_ir_sensor    link_name="camera_link"    frame_name="camera_depth_frame"
camera_name="kinect"/>

   <!--       Gazebo       plugin       for       Kinect       RGB       sensor       -->
   <xacro:kinect_rgb_sensor    link_name="camera_link"    frame_name="camera_rgb_frame"
camera_name="kinect"/>

   <!--         Gazebo         plugin         for         IMU         sensor         -->
   <xacro:imu_sensor_gazebo          link_name="imu_link"          imu_topic="imu_data"
update_rate="100.0"/>

 </xacro:macro>
</robot>
```

practice1.world

```
<?xml                                   version="1.0"                                   ?>
<sdf                                                                          version="1.4">
       <world                                                                 name="default">
              <include>
                     <uri>model://ground_plane</uri>
              </include>
              <include>
                     <uri>model://sun</uri>
              </include>
              <model                                                    name='closed_room'>
   <link                                                                      name='Wall_1'>
     <collision                                                    name='Wall_1_Collision'>
       <geometry>
         <box>
           <size>5                              0.2                              1.5</size>
         </box>
       </geometry>
       <pose>0            0            0.75            0            -0             0</pose>
     </collision>
     <visual                                                      name='Wall_1_Visual'>
       <pose>0            0            0.75            0            -0             0</pose>
       <geometry>
         <box>
           <size>5                              0.2                              1.5</size>
         </box>
       </geometry>
       <material>
         <script>
           <uri>file://media/materials/scripts/gazebo.material</uri>
           <name>Gazebo/Grey</name>
         </script>
       </material>
     </visual>
     <velocity_decay>
       <linear>0</linear>
       <angular>0</angular>
     </velocity_decay>
     <pose>0            -2.5             0            0            0             0</pose>
   </link>
   <link                                                                      name='Wall_2'>
     <collision                                                    name='Wall_2_Collision'>
       <geometry>
         <box>
           <size>5                              0.2                              1.5</size>
         </box>
       </geometry>
       <pose>0            0            0.75            0            -0             0</pose>
     </collision>
     <visual                                                      name='Wall_2_Visual'>
       <pose>0            0            0.75            0            -0             0</pose>
       <geometry>
         <box>
           <size>5                              0.2                              1.5</size>
         </box>
       </geometry>
       <material>
         <script>
           <uri>file://media/materials/scripts/gazebo.material</uri>
           <name>Gazebo/Grey</name>
         </script>
       </material>
     </visual>
```

```
    <velocity_decay>
      <linear>0</linear>
      <angular>0</angular>
    </velocity_decay>
    <pose>2.5           0           0           0           0           1.5708</pose>
  </link>
  <link                                                          name='Wall_3'>
    <collision                                                 name='Wall_3_Collision'>
      <geometry>
        <box>
          <size>5                           0.2                           1.5</size>
        </box>
      </geometry>
      <pose>0           0           0.75           0           -0           0</pose>
    </collision>
    <visual                                                     name='Wall_3_Visual'>
      <pose>0           0           0.75           0           -0           0</pose>
      <geometry>
        <box>
          <size>5                           0.2                           1.5</size>
        </box>
      </geometry>
      <material>
        <script>
          <uri>file://media/materials/scripts/gazebo.material</uri>
          <name>Gazebo/Grey</name>
        </script>
      </material>
    </visual>
    <velocity_decay>
      <linear>0</linear>
      <angular>0</angular>
    </velocity_decay>
    <pose>0           2.5           0           0           -0           -3.14159</pose>
  </link>
  <link                                                          name='Wall_4'>
    <collision                                                 name='Wall_4_Collision'>
      <geometry>
        <box>
          <size>5                           0.2                           1.5</size>
        </box>
      </geometry>
      <pose>0           0           0.75           0           -0           0</pose>
    </collision>
    <visual                                                     name='Wall_4_Visual'>
      <pose>0           0           0.75           0           -0           0</pose>
      <geometry>
        <box>
          <size>5                           0.2                           1.5</size>
        </box>
      </geometry>
      <material>
        <script>
          <uri>file://media/materials/scripts/gazebo.material</uri>
          <name>Gazebo/Grey</name>
        </script>
      </material>
    </visual>
    <velocity_decay>
      <linear>0</linear>
      <angular>0</angular>
    </velocity_decay>
    <pose>-2.5           0           0           0           0           -1.5708</pose>
  </link>
  <static>1</static>
</model>
```

```
</world>
</sdf>
```

practice2.world

```
<?xml                                version="1.0"                                ?>
<sdf                                                                  version="1.4">
 <world                                                               name="default">
   <include>
     <uri>model://ground_plane</uri>
   </include>
   <include>
     <uri>model://sun</uri>
   </include>

   <model                                                               name="spheres">
     <static>true</static>
     <link                                                             name='sphere_1'>
       <pose>0            2            0.5            0            0            0</pose>
       <collision                                                   name='collision'>
         <geometry>
           <sphere>
             <radius>0.5</radius>
           </sphere>
         </geometry>
       </collision>
       <visual                                                        name='visual'>
         <geometry>
           <sphere>
             <radius>0.5</radius>
           </sphere>
         </geometry>
         <material>
           <script>
             <uri>file://media/materials/scripts/gazebo.material</uri>
             <name>Gazebo/Red</name>
           </script>
         </material>
       </visual>
     </link>
     <link                                                             name='sphere_2'>
       <pose>-2           -2           0.5           0           0           0</pose>
       <collision                                                   name='collision'>
         <geometry>
           <sphere>
             <radius>0.5</radius>
           </sphere>
         </geometry>
       </collision>
       <visual                                                        name='visual'>
         <geometry>
           <sphere>
             <radius>0.5</radius>
           </sphere>
         </geometry>
         <material>
           <script>
             <uri>file://media/materials/scripts/gazebo.material</uri>
             <name>Gazebo/Blue</name>
           </script>
         </material>
       </visual>
```

```
    </link>
    <link                                                    name='sphere_3'>
      <pose>2           -2           0.5           0           0           0</pose>
      <collision                                            name='collision'>
        <geometry>
          <sphere>
            <radius>0.5</radius>
          </sphere>
        </geometry>
      </collision>
      <visual                                                 name='visual'>
        <geometry>
          <sphere>
            <radius>0.5</radius>
          </sphere>
        </geometry>
        <material>
          <script>
            <uri>file://media/materials/scripts/gazebo.material</uri>
            <name>Gazebo/Green</name>
          </script>
        </material>
      </visual>
    </link>
  </model>

  <model                                                  name="closed_room">
    <link                                                      name="Wall_1">
      <collision                                          name="Wall_1_Collision">
        <geometry>
          <box>
            <size>10                        0.2                         1.5</size>
          </box>
        </geometry>
        <pose>0           0           0.75          0           -0          0</pose>
      </collision>
      <visual                                               name="Wall_1_Visual">
        <pose>0           0           0.75          0           -0          0</pose>
        <geometry>
          <box>
            <size>10                        0.2                         1.5</size>
          </box>
        </geometry>
        <material>
          <script>
            <uri>file://media/materials/scripts/gazebo.material</uri>
            <name>Gazebo/DarkGrey</name>
          </script>
        </material>
      </visual>
      <velocity_decay>
        <linear>0</linear>
        <angular>0</angular>
      </velocity_decay>
      <pose>0           -5           0           0           0           0</pose>
    </link>
    <link                                                      name="Wall_2">
      <collision                                          name="Wall_2_Collision">
        <geometry>
          <box>
            <size>10                        0.2                         1.5</size>
          </box>
        </geometry>
        <pose>0           0           0.75          0           -0          0</pose>
      </collision>
      <visual                                               name="Wall_2_Visual">
```

```
      <pose>0 0 0.75 0 -0 0</pose>
      <geometry>
        <box>
          <size>10 0.2 1.5</size>
        </box>
      </geometry>
      <material>
        <script>
          <uri>file://media/materials/scripts/gazebo.material</uri>
          <name>Gazebo/DarkGrey</name>
        </script>
      </material>
    </visual>
    <velocity_decay>
      <linear>0</linear>
      <angular>0</angular>
    </velocity_decay>
    <pose>5 0 0 0 0 1.5708</pose>
  </link>
  <link name="Wall_3">
    <collision name="Wall_3_Collision">
      <geometry>
        <box>
          <size>10 0.2 1.5</size>
        </box>
      </geometry>
      <pose>0 0 0.75 0 -0 0</pose>
    </collision>
    <visual name="Wall_3_Visual">
      <pose>0 0 0.75 0 -0 0</pose>
      <geometry>
        <box>
          <size>10 0.2 1.5</size>
        </box>
      </geometry>
      <material>
        <script>
          <uri>file://media/materials/scripts/gazebo.material</uri>
          <name>Gazebo/DarkGrey</name>
        </script>
      </material>
    </visual>
    <velocity_decay>
      <linear>0</linear>
      <angular>0</angular>
    </velocity_decay>
    <pose>0 5 0 0 -0 -3.14159</pose>
  </link>
  <link name="Wall_4">
    <collision name="Wall_4_Collision">
      <geometry>
        <box>
          <size>10 0.2 1.5</size>
        </box>
      </geometry>
      <pose>0 0 0.75 0 -0 0</pose>
    </collision>
    <visual name="Wall_4_Visual">
      <pose>0 0 0.75 0 -0 0</pose>
      <geometry>
        <box>
          <size>10 0.2 1.5</size>
        </box>
      </geometry>
      <material>
        <script>
```

```
            <uri>file://media/materials/scripts/gazebo.material</uri>
            <name>Gazebo/DarkGrey</name>
          </script>
        </material>
      </visual>
      <velocity_decay>
        <linear>0</linear>
        <angular>0</angular>
      </velocity_decay>
      <pose>-5          0          0          0          0          -1.5708</pose>
    </link>
    <static>1</static>
  </model>
 </world>
</sdf>
```

canyonero_bringup.launch

```
<launch>

      <arg                        name="teleop"                        default="true"/>

      <!--Teleop                                                                  -->
      <group                              if="$(arg                              teleop)">
            <include    file="$(find    canyonero_control)/launch/teleop.launch"    />
      </group>

   <!--         Load         Canyonero         control         information         -->
      <include         file="$(find         canyonero_control)/launch/control.launch"/>

 </launch>
```

practice_worlds.launch

```
<launch>
      <arg                      name="practice1"                      default="false"/>
      <arg                      name="practice2"                      default="false"/>
      <arg                      name="practice3"                      default="false"/>
      <arg                      name="practice4"                      default="false"/>
      <arg                      name="practice5"                      default="false"/>

      <!--     Adding     empty_world.launch     and     practice     worlds     -->
      <group                              if="$(arg                                practice1)">
            <include       file="$(find       gazebo_ros)/launch/empty_world.launch">
                  <arg                name="world_name"                value="$(find
canyonero_gazebo)/worlds/practice1.world"/>
            </include>
      </group>

      <group                              if="$(arg                                practice2)">
            <include       file="$(find       gazebo_ros)/launch/empty_world.launch">
                  <arg                name="world_name"                value="$(find
canyonero_gazebo)/worlds/practice2.world"/>
            </include>
      </group>

      <group                              if="$(arg                                practice3)">
            <include       file="$(find       gazebo_ros)/launch/empty_world.launch">
                  <arg                name="world_name"                value="$(find
```

```
canyonero_gazebo)/worlds/practice3.world"/>
            </include>
      </group>

      <group                          if="$(arg                          practice4)">
            <include      file="$(find      gazebo_ros)/launch/empty_world.launch">
                  <arg              name="world_name"              value="$(find
canyonero_gazebo)/worlds/practice4.world"/>
            </include>
      </group>

      <group                          if="$(arg                          practice5)">
            <include      file="$(find      gazebo_ros)/launch/empty_world.launch">
                  <arg              name="world_name"              value="$(find
canyonero_gazebo)/worlds/practice5.world"/>
            </include>
      </group>
</launch>
```

canyonero_sim.launch

```
<launch>

      <!--                  Practice                  worlds                  -->
      <include     file="$(find     canyonero_gazebo)/launch/practice_worlds.launch"/>

      <!--              Bring              up              robot              -->
      <include    file="$(find    canyonero_gazebo)/launch/canyonero_bringup.launch"/>

      <!--                    Sim                    params                    -->
      <arg                      name="paused"                      default="false"/>
      <arg                     name="use_sim_time"                      default="true"/>
      <arg                         name="gui"                      default="true"/>
      <arg                      name="headless"                     default="false"/>
      <arg                        name="debug"                      default="false"/>

      <!--    Load    the    URDF    into    the    ROS    Parameter    Server    -->
      <param    name="robot_description"    command="$(find    xacro)/xacro.py    '$(find
canyonero_description)/urdf/canyonero.urdf.xacro'"                                    />

      <!-- Run a python script to the send a service call to gazebo_ros to spawn a
URDF                                robot                                 -->
      <node  name="urdf_spawner"  pkg="gazebo_ros"  type="spawn_model"  respawn="false"
output="screen"
      args="-urdf    -model    canyonero    -param    robot_description    -z    0.01"/>

</launch>
```