TecnoCampus
Escola Superior Politècnica

Centre adscrit a la
upf. Universitat Pompeu Fabra Barcelona

Grau en enginyeria electrònica industrial i automàtica.

ESTUDI I APLICACIÓ DEL ROBOT MÒBIL TURTLEBOT3 BURGER

SERGI COLOMÉ GRAU

PONENT: JOAN TRIADÓ

TARDOR 2018

TecnoCampus
Mataró-Maresme

Sumari de documents

1. Memòria
2. Estudi econòmic
3. Annexes

TecnoCampus
Escola Superior Politècnica

Centre adscrit a la
upf. Universitat Pompeu Fabra Barcelona

Grau en enginyeria electrònica industrial i automàtica.

ESTUDI I APLICACIÓ DEL ROBOT MÒBIL TURTLEBOT3 BURGER

Memòria

SERGI COLOMÉ GRAU

PONENT: JOAN TRIADÓ

TARDOR 2018

TecnoCampus
Mataró-Maresme

Agraïments

En primer lloc, agrair-li al ponent Joan Triadó pel suport i la motivació durant aquests mesos que ha durat el projecte.

A en Josep Lòpez, el qual tot i no ser el ponent del projecte ha estat disponible i s'ha ofert per resoldre qualsevol dubte, en tot moment.

I per últim, i no menys important als meus pares que han fet possible que pogués estudiar el que volia.

Resum

En aquesta memòria es pot contemplar un estudi, amb la finalitat de conèixer quin és l'entorn amb el qual treballa el robot mòbil TurtleBot3 Burger, dur a terme la posada en marxa d'aquest, és a dir, instal·lar els paquets necessaris en l'ordinador principal i al robot perquè aquest funcioni. I finalment, hi ha una aplicació en la qual es pot visualitzar les diferents parts que s'han anat parlant en el projecte.

Resumen

En esta memoria se puede contemplar un estudio, con el fin de conocer cuál es el entorno con el que trabaja el robot móvil TurtleBot3 Burger, llevar a cabo la puesta en marcha del mismo, es decir instalar los paquetes necesarios en el ordenador principal y en el robot para que este funcione. Y finalmente, hay una aplicación en la que se puede visualizar las diferentes partes que se han ido hablando en el proyecto.

Abstract

Within the memory you can conduct a study, this will provide you with detailed information on how the mobile robot TurtleBot3 Burger works and the environment in which it has explored. To carry out the implementation of this study, all necessary packages in the main computer should be installed. An application is also available where you can visualize the different topics that have been discussed throughout this project

Índex.

Índex de figures.

Índex de taules.

Glossari de termes.

IP	Protocol d'Internet
PC	Personal Computer
QFD	Quality Function Deployment. Desplegament de la Funció de Qualitat
ROS	Robot Operating System, Open Robotics
SBC	Single Board Computer
SDK	Software development kit
SRC	Source. Directori principal de Linux
SSH	Secure SHell. S'utilitza per accedir a màquines de manera remota

1. Objectius.

1.1. Propòsit.

El propòsit principal d'aquest projecte és realitzar un estudi de l'entorn amb el qual treballa Turtlebot3 Burger, ja que ha sorgit la necessitat de conèixer quines són les eines d'aquest per poder desenvolupar una aplicació. Inclourà un manual d'instruccions, el qual es veurà reflectida tant la instal·lació de l'entorn d'Ubuntu, com l'entorn de TurtleBot. A més, es farà una posada en marxa del Turtbot3 Burger i es dissenyarà una aplicació.

1.2. Finalitat.

La finalitat del projecte és realitzar un estudi de l'entorn el qual treballa TurtleBot3 Burger, amb l'objectiu de desenvolupar una aplicació i dur a terme la posada en marxa del robot amb l'aplicació desenvolupada.

1.3. Objecte.

L'objecte del present projecte és un estudi, coneixent quin és l'entorn amb el qual treballa aquest robot, per posteriorment realitzar una aplicació pel robot mòbil TurtleBot3 Burger. Aquesta aplicació tindrà un caire industrial, és a dir, la funcionalitat d'aquesta serà un prototip per una empresa industrial.

1.4. Abast.

S'inclou un manual d'instruccions dels passos que s'han seguit per instal·lar com s'ha dut a terme la instal·lació de l'entorn de treball de TurtleBot3 Burger, des de la instal·lació d'Ubuntu a l'ordinador, a quines instruccions s'han d'introduir al terminal de l'ordinador per fer la posada en marxa del robot. A més, inclourà un seguit d'aplicacions aplicades a TurtleBot3 Burger, amb l'objectiu d'incloure una aplicació on es puguin visualitzar alguns dels aspectes treballats durant el projecte.

2. Introducció.

La robòtica mòbil és una eina valuosa per explorar aquells entorns inaccessibles per l'ésser humà, sigui pel cost, perillositat o bé la seva distància. A més, és útil per realitzar tasques desagradables o laborioses. És un camp relativament nou, i fins fa poc experimental, però que ja s'està aplicant a problemes reals amb resultats satisfactoris.

TurtleBot3 és un robot mòbil de nova generació, el qual és modular, compacte i sobretot personalitzable, es podria dir que aquesta és la característica més destacable del robot. A més, aprofita una nova generació de maquinària de computació i detecció a baix cost, per tal d'introduir el màxim possible en el robot mòbil més petit i econòmic possible.

2.1. Objectiu.

El projecte sorgeix de la necessitat de realitzar un estudi del robot mòbil Turtlebot3 Burger, per posteriorment poder establir i realitzar una aplicació sobre el robot, basada en una tasca destinada en l'àmbit industrial.

El que es pretén amb aquest projecte és desenvolupar una aplicació pel robot mòbil Turtlebot3 Burger. Aquesta aplicació consistirà a fer anar el robot des d'un punt conegut de l'espai, fins a un altre punt també conegut, simulant el transport de matèria dins d'una indústria, fent que el robot detecti els obstacles i actuï quan es topi amb algun d'ells.

La principal finalitat d'aquesta aplicació és eliminar una tasca monòtona com és el transport de matèries primeres, i posar-hi un robot que el substitueixi, així doncs, es podrà reemplaçar la tasca de subministrament de matèria primera, per part d'un operari.

Per a poder dur a terme aquest projecte, és necessari un estudi previ de les especificacions tècniques del Turtlebot3 Burger, i quina és la finalitat que se li ha d'implementar. Per tant, d'aquesta manera es pretén fer una anàlisi del software i hardware del robot per poder decidir de quina forma es desenvoluparà l'aplicació.

Tanmateix, serà necessari realitzar una revisió d'antecedents de les diferents possibilitats de programació que disposa el robot, i escollir la que millor s'adapti per ~~a~~ l'aplicació, i d'un estudi dels diferents components dels quals disposa el Turtlebot3 Burger de fàbrica. Un cop es tingui la informació documentada, es dissenyarà una rúbrica per poder

determinar quin codi de programació és el més adequat, realitzant així una proposta de programació per a l'aplicació.

2.2. Revisió d'antecedents i necessitats d'informació.

En el present apartat es cerca tota aquella informació necessària per poder desenvolupar el projecte. Per poder abordar-ho de manera pautada s'ha separat en diversos apartats; en primer lloc es presentarà el robot mòbil TurtleBot3 Burger, per tenir una idea global dels aspectes generals d'aquest, en segon lloc, es parlarà de les rutes dinàmiques i finalment de les diferents maneres d'actuar davant dels obstacles.

2.2.1. TurtleBot3 Burger.

TurtleBot3 es troba destinat en l'àmbit de la investigació i desenvolupament de productes, ja que a través de l'entorn ROS, permet crear aplicacions de caràcter educatiu.

Principalment el robot Turtlebot3 Burger és un robot creat per la recerca, aprenentatge estudiantil i productes de desenvolupament, aquest fet és degut que es pugui arribar a diferents nivells de programació i interacció amb el robot, generant així un gran ventall ampli de programacions.

Es podria dir que TurtleBot3 aprofita una nova generació de maquinari de computació i detecció a un cost relativament baix, a més a més, com ja s'ha comentat amb anterioritat, consta d'una plataforma la qual conté una completa instal·lació de ROS de la manera més simplificada possible, però sense perdre la capacitat i la comoditat. Per tant, es podria dir que un dels objectius amb TurtleBot3 és reduir de manera dràstica la mida i el preu de la plataforma robòtica, sense perdre la capacitat, funcionalitat, i qualitat que Open Robotis portava oferint els darrers anys. A més, es disposa de peces addicionals, com ara poden ser xassís, ordinadors i sensors, de tal manera que es pot personalitzar el robot tant com es desitgi.

Especificacions tècniques.

- Màxima velocitat de translació: 0,22 m / s
- Màx. velocitat de rotació: 162.72 ° / s

- Màx. càrrega útil: 15 kg
- Dimensions: 138 × 178 × 192 mm
- Pes (+ SBC + bateria + sensors): 995 g
- Màx. pas travessable: 10 mm
- Autonomia: 2h 30m
- Temps de càrrega: 2h 30m
- Connexió per a l'ordinador: USB
- IMU: 3 eixos giroscopi, 3 eixos acceleròmetre, 3 eixos magnetòmetre
- Connectors d'alimentació:
    - 3.3V / 800 mA
    - 5V / 2A
    - 12V / 1A
- Pins d'expansió: 18 pins GPIO, 32 pins Arduino
- Diverses seqüències de sons programables
- Arduino LED x 1
- 2 botons
- Bateria de polímer de liti 11.1V 1800 mAh / 19.98 Wh 5C
- Adaptador de recàrrega:
    - Input: 100-240V, AC 50/60 Hz, 1.5A max
    - Output: 12 Vdc, 5A

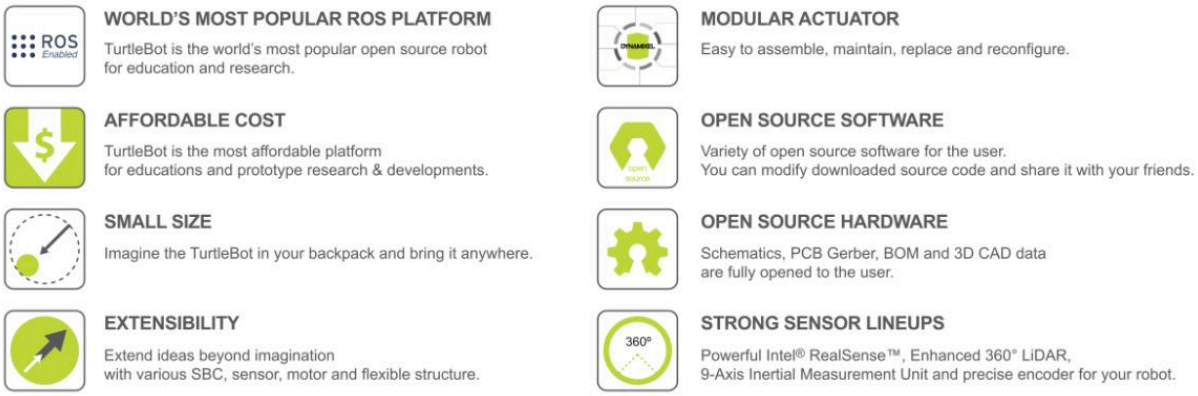


Figura 2.1. Principals característiques Turtlebot3 Burger

Font: [1]

Estructura del Turtlebot3 Burger.

El cos del robot es troba construït amb plaques de plàstic, barres de metall, rodes, caragols, femelles i reblons. La seva dimensió és només de 138 mm x 178 mm x 192 mm, reduint així en una quarta part les dimensions de l'antecessor.

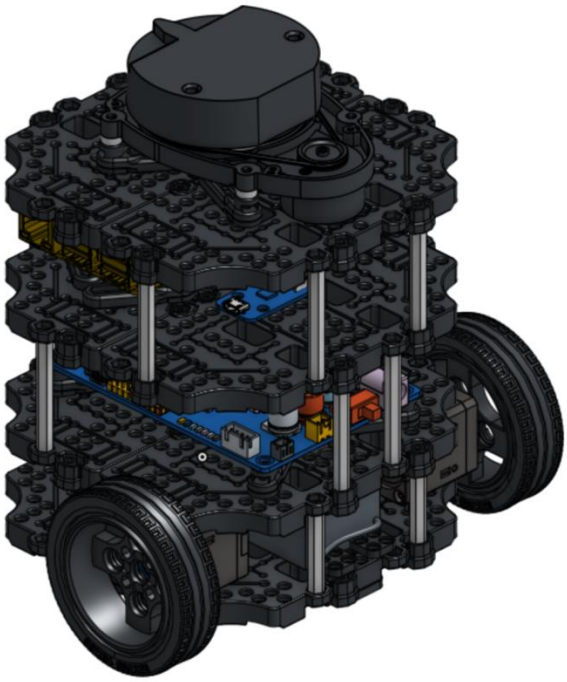


Figura 2.2. Estructura TurtleBot3

Font: [2]

Com es pot observar, compta amb diversos nivells apilats un sobre l'altre, permetent així simplificar l'estructura.

A continuació, es mostraran els diferents nivells dels quals disposa TurtleBot3 Burger.

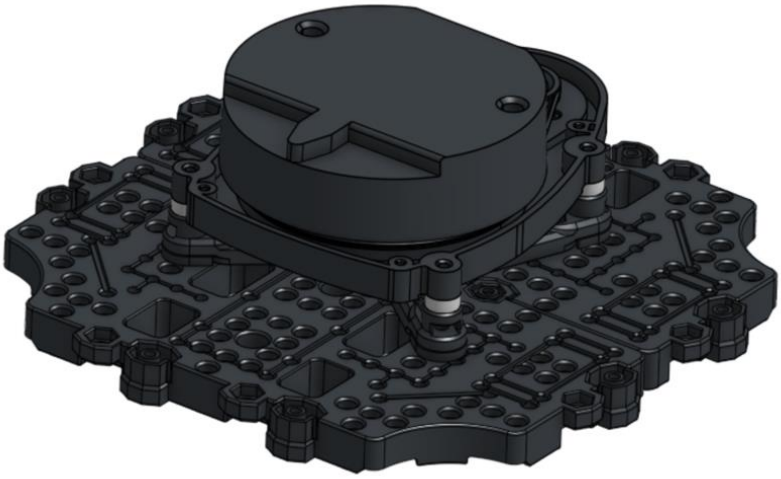


Figura 2.3. Quarta planta TurtleBot3 Burger

Font: [2]

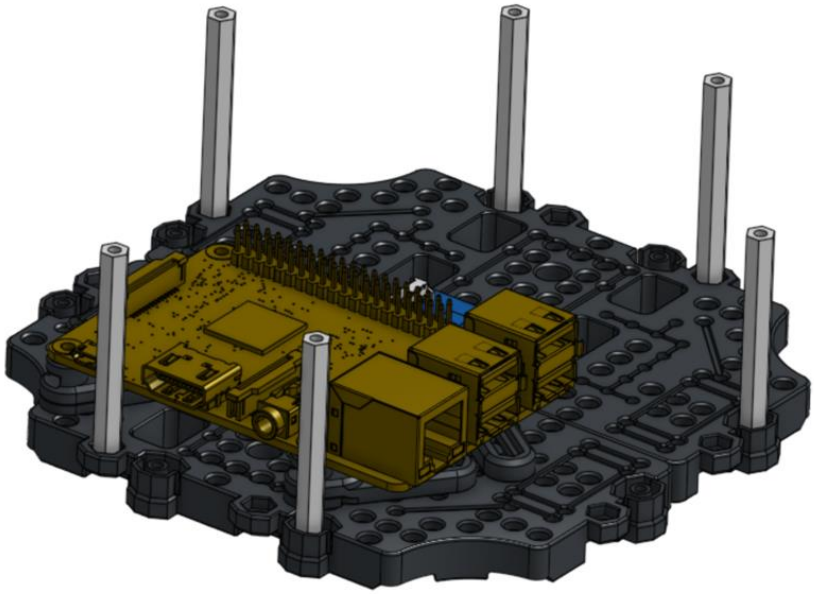


Figura 2.4. Tercera planta TurtleBot3 Burger

Font: [2]

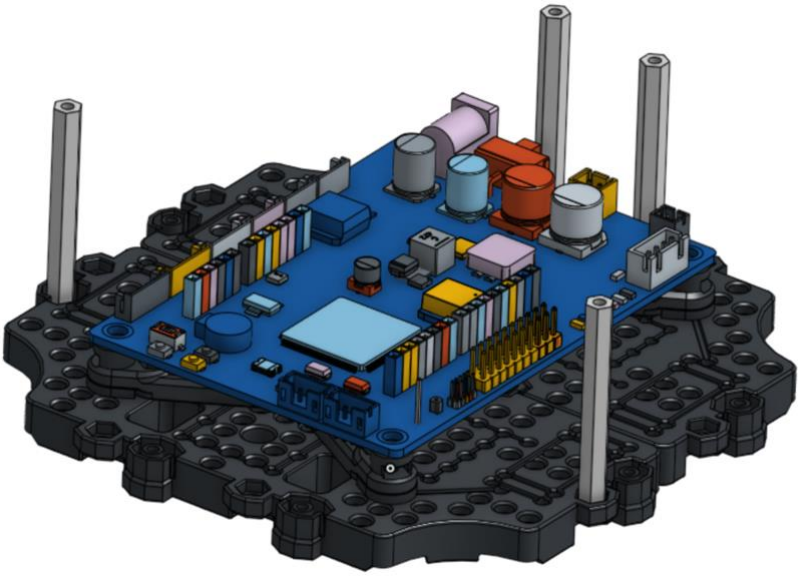


Figura 2.5. Segona planta TurtleBot3 Burger

Font: [2]

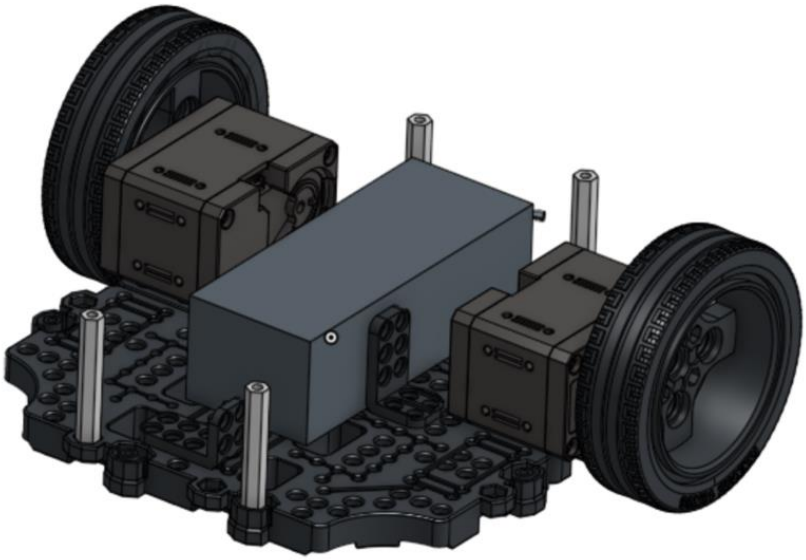


Figura 2.6. Primera planta TurtleBot3 Burger

Font: [2]

Hardware del Turtlebot3 Burger.

En aquest apartat, una vegada presentats els aspectes generals i la seva estructura, es farà incisió amb el hardware del robot.

Motors DYNAMIXEL (XL430-W250-T):

La DYNAMIXEL X-Series és una nova línia d'actuadors de xarxa d'alt rendiment.

La sèrie Dynamixel XL adopta noves funcions que permeten el mode de control de 360 graus amb el seu *encoder* magnètic sense contacte i l'estructura de muntatge del cas buit. La sèrie XL té la mateixa estructura mecànica que la XM430 i la XH430 i és compatible amb els respectius models. [3]

Figura 2.7. DYNAMIXEL (XL430-W250-T)

Font: [4]

Les principals característiques que aquests motors ofereixen són:

- Precisió: 0.088º
- Resolució 4096 steps/volta
- Control de l'algorisme: PID
- Mètodes operatius
- Control de velocitat
- Control de posició
- Control extens de posició
- Control PWM

Placa OpenCR1.0:

OpenCR1.0 es desenvolupa pels sistemes integrats de ROS per proporcionar hardware i software totalment oberts.

Tot sobre el tauler; Schemics, PCB Gerber, BOM i el codi font del firmware de TurtleBot3 i OP3 són lliures de distribuir sota llicències de codi obert per als usuaris i la comunitat ROS.

El xip de la sèrie STM32F7 dins de la placa OpenCR1.0 es basa en un ARM Cortex-M7 molt potent amb unitat de punt flotant.

L'entorn de desenvolupament de OpenCR1.0 és obert des d'Arduino IDE i Scratch per a joves estudiants fins al desenvolupament tradicional del firmware per a l'expert.

OpenCR1.0 proporciona pins digitals i analògics d'entrada / sortida que poden interactuar amb un tauler d'extensió o diversos sensors. A més, OpenCR1.0 presenta diverses interfícies de comunicació: USB per connectar-se a PC, UART, SPI, I2C, CAN per a altres dispositius incrustats.

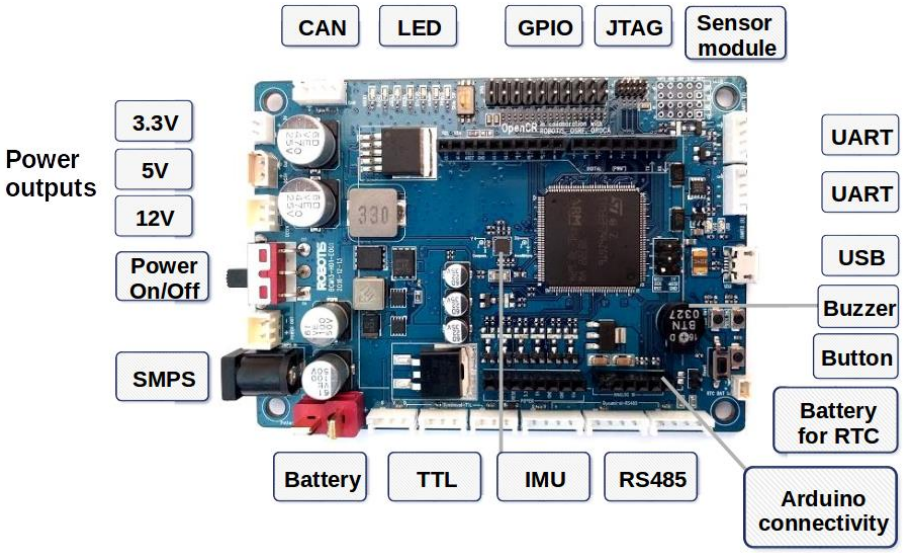


Figura 2.8. OpenCR1.0

Font: [2]

Raspberry pi 3 Model B:

La Raspberry Pi 3 té Bluetooth 4.1 (de baix consum) i WiFi 802.11n integrat, sense afegir cap accessori més, també disposa d'un processador ARM Cortex A53, un processador de quatre nuclis a 1.2GHz de 64 bits i que, segons les seves dades, té un 50% més que la Raspberry Pi 2, el model anterior.

Figura 2.9. Raspberry Pi 3 Model B

Font: [5]

Sensor LDS (HLS-LFCD2):

El LDS-01 (sensor de distància LASER) és un sensor que recopila un conjunt de dades de distància i els envia a l'amfitrió per a la tècnica de localització i cartografia simultània (SLAM).

Les principals especificacions tècniques que ofereix aquest sensor són:

- Tensió d'alimentació: 5V
- Consum de corrent: 400mA aproximadament
- Distància de detecció: 120mm – 3.500mm
- Resistència a la llum ambiental: 10.000 lux
- Gama angular: 360º

Figura 2.10. Sensor LDS (HLS-LFCD2)

Font: [6]

El LDS emet un làser infraroig modulat mentre gira completament. El sensor processa el senyal retornant i extreu l'angle i la distància. En la següent figura es pot veure un esbós del seu funcionament.

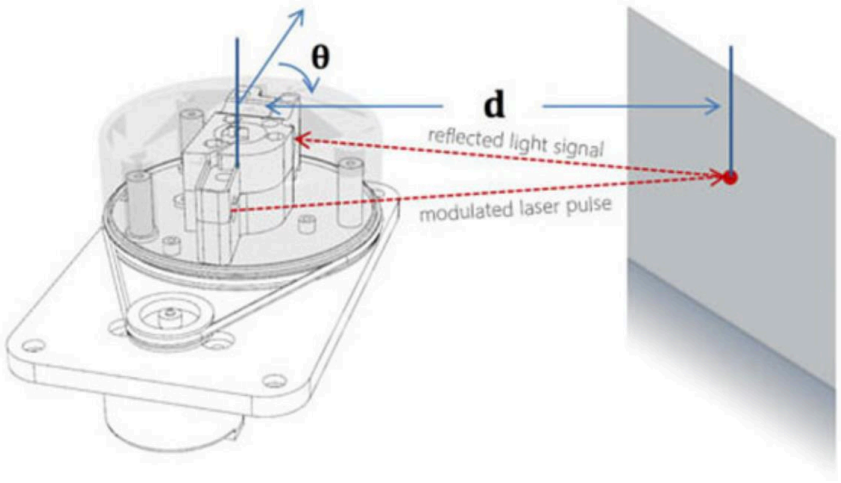


Figura 2.11. Funcionament sensor LIDAR

Font: [6]

2.2.2. Tipus de rutes segons la demanda.

En aquest apartat es tractarà els diferents tipus de planificació que s'han de tenir en compte per tal d'esquivar els obstacles que es trobi Turtlebot3 Burger.

Rutes fixes.

Les rutes fixes mantenen un recorregut estable a causa de:

- Punts de lliurament o de recollida fixos o semi fixos.
- Volums estables.
- Poques restriccions de vehicles, clients, personal.

Per poder tenir un coneixement més extens del que són les rutes dinàmiques, a continuació es durà a terme una taula amb els avantatges i inconvenients principals.

AVANTATGES	INCONVENIENTS
- Es segueixen unes rutes diàries - Vehicle adaptat a la zona - Sistema informàtic menys complex	- Els lliuraments poden superar la capacitat dels vehicles - Mala qualitat del servei per ajornament dels lliuraments

Taula 2.1. Avantatges i inconvenients de les rutes fixes

Font: Elaboració pròpia

Rutes dinàmiques.

Les rutes dinàmiques es programen en funció de la demanada a causa de:

- La gran variació dels punts de lliurament.
- Oscil·lació significativa en els volums de càrrega o descarrega.
- Restriccions importants dels clients, vehicles, personal.

Per poder comprendre millor, es durà a terme una taula amb els avantatges i inconvenients principals de les rutes dinàmiques.

AVANTATGES	INCONVENIENTS
- Optimitzen el temps emprat - Optimitzen la distància recorreguda - Optimitzen el cost integral - Milloren el nivell de servei	- Es necessita un sistema informàtic complex - L'usuari ha de tenir uns coneixements mínims d'informàtica

Taula 2.2. Avantatges i inconvenients de les rutes dinàmiques

Font: Elaboració pròpia

2.2.3. Maneres d'actuar davant d'obstacles.

Amb la principal situació que es trobarà el robot en l'àmbit industrial, seran obstacles bàsicament frontals, com poden ser persones, màquines o palets.

A continuació, es presentaran diferents alternatives per evitar aquests possibles. Cal dir, que quan es parla d'obstacles, sempre s'està referint a obstacles mòbils, i no fixes, aquests se suposa que els detecta com a parets.

Esperar fins que l'obstacle es mogui.

Aquest cas va destinat a aquell àmbit on es tenen les cotes de recorregut molt marcades i distingides pel robot, i aquest sap que qualsevol obstacle que pugui trobar-se sempre serà mòbil. Per tant, és una bona solució per aquelles aplicacions on hi ha trànsit de personal i les rutes no són canviants.

Fer un gir i buscar una nova trajectòria.

Per poder desenvolupar aquesta aplicació sobre el robot, aquest ha de disposar d'espai suficient per poder maniobrar i buscar una ruta més òptima i lliure d'obstacles. Sovint aquestes rutes solen ser més complexes, ja que s'ha d'identificar la nova ruta per la qual es mourà el robot.

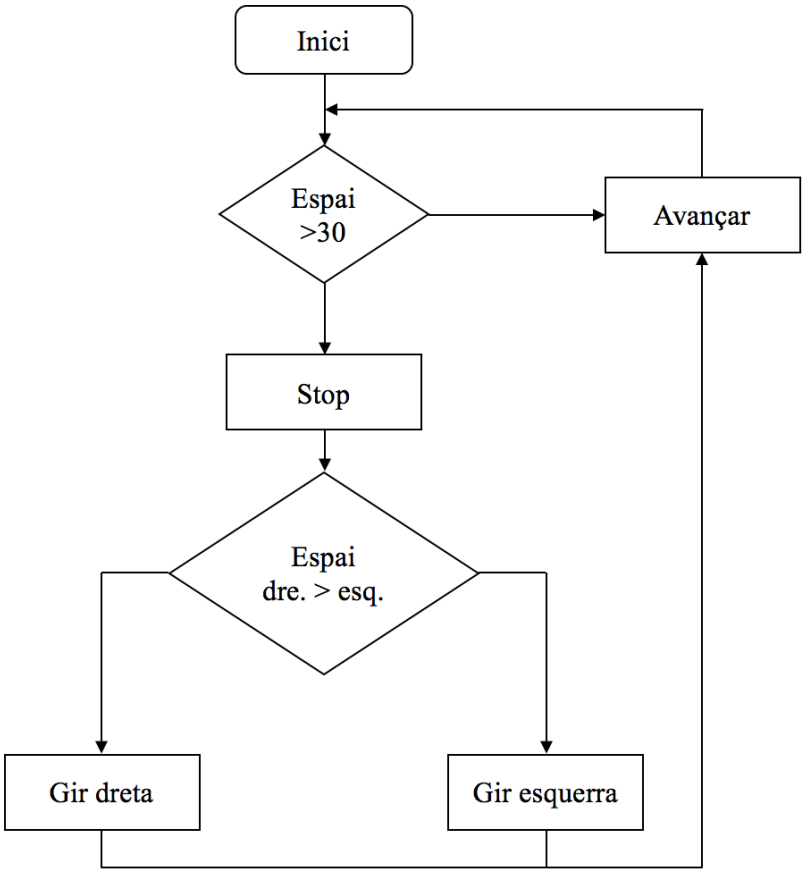


Figura 2.12. Exemple detector objecte

Font: Elaboració pròpia

2.2.4. Programació del software.

ROS (Robot Operating System)

ROS proporciona biblioteques i eines per ajudar als desenvolupadors de programari crear aplicacions robòtiques. Proporciona una abstracció del maquinari, dels controladors de dispositius, les biblioteques, visualitzadors, entre d'altres.

Per tant, aquest conjunt d'eines i llibreries proporciona:

Reutilització de codi:

Els paquets estàndard proporcionats en les distribucions de ROS implementen molts dels algoritmes comunament emprats en robòtica que ja han estat depurats i usats de forma

estable. A més dels paquets estàndard, n'hi ha molts d'altres en la comunitat que implementen interfícies per als seus sistemes (robots, sensors, llibreries ...).

Testejat ràpid:

A causa del disseny de comunicació pel pas de missatges, ROS ens permet simular molts dels dispositius amb els quals el nostre sistema treballarà, d'aquesta manera podem aïllar la funcionalitat del nostre sistema del codi de comunicació entre les diferents parts del sistema, sensors i actuadors. Un dels aspectes claus en desenvolupar una aplicació és la capacitat de repetir els experiments i poder simular els sensors i actuadors que ens permet crear conjunts de prova.

Per tant, ROS disposa d'una àmplia llibreria d'exemple d'altres projectes, les quals ja han estat provats i depurats. Aquest fet facilita la tasca de programació del robot, ja que es poden agafar algunes estructures d'altres programes i treballar sobre d'elles.

Raspberry Pi 3 Model B i Python.

Python és una sintaxi que té el seu origen en 1991, any en què va ser creat per Guido Van Rossem amb l'objectiu de fer un llenguatge de programació àgil i senzill, amb una corba d'aprenentatge molt curta.

Aquest llenguatge és una sintaxi de propòsit general molt intuïtiu: qualsevol desenvolupador pugui dedicar poc temps a aprendre-i ocupar-se més en com crear productes innovadors amb ell, aprofitant sobretot la seva gran flexibilitat. Això el converteix en un llenguatge divertit i versàtil per als programadors.

Les tres principals característiques d'aquest llenguatge són:

És una gran multiplataforma:

Python és un llenguatge de programació interpretat, de manera que funciona en qualsevol tipus de sistema que integri el seu interpretador. A part d'aquest avantatge, Python ofereix dialectes com el ja conegut Jython, que s'utilitza per escriure en Java.

És lliure i ens ofereix codi obert:

Pel que fa a la llicència que posseeix, aquesta és Python Programari Foundation License, llicència molt semblant a la de GPL, però trobant l'excepció que es poden distribuir els binaris del llenguatge sense haver d'annexar les fonts.

Python: programació orientada a objectes:

Si parlem de programació orientada a objectes, podem dir que ens trobem davant d'un paradigma que proposa modelar tot en funció a classes i a objectes, el qual ofereix un ús de conceptes de cohesió, herència, abstracció i molt més.

2.3. Límits del projecte.

El present projecte presenta un estudi del robot mòbil Turtlebot3 Burger, amb l'objectiu de poder establir i realitzar una aplicació sobre el robot, la qual, estarà basada en una tasca destinada de l'àmbit industrial.

El primer pas a realitzar serà definir l'objecte del projecte, el qual mitjançant una cerca d'informació i d'antecedents, es definiran els objectius del projecte en qüestió.

A partir d'una sessió de creativitat, i una posterior rúbrica per fer una valoració, es determinaran quin són els objectius principals a assolir en el projecte. Quan es tinguin els objectius principals definits, es definiran una sèrie d'especificacions tècniques que s'hauran de complir per tal d'assolir els objectius prèviament marcats. Els quals són:

- Dur a terme la posada en marxa del robot, configurant tots els paràmetres.
- Moure el robot per un entorn no conegut i conèixer que succeeix en el seu entorn.

Tot seguit, es plantejaran una sèrie d'alternatives de solució, per definir a partir de rúbriques quina és la millor alternativa a desenvolupar en cada cas. Posteriorment, es realitzarà una anàlisi de viabilitat tècnica, econòmica i mediambiental de l'alternativa a desenvolupar.

Per últim, es procedirà a realitzar la planificació del projecte i poder determinar quin serà el pressupost d'aquest.

En el projecte s'inclou un prototip, també s'inclou la posada en marxa de la proposta a desenvolupar. Per tant, el que es pot trobar en aquest present projecte és una proposta d'actuació del robot mòbil Turtlebot 3 Burger, el qual consistirà en moure el robot per una superfície desconeguda, i a través del sensor i de la càmera moure's a través d'ella.

3. Objectius de detall i especificacions tècniques.

La finalitat del present apartat és definir i exposar els diversos objectius que es pretenen assolir amb la realització del projecte. Abans però, ha sigut necessària una cerca curada de les especificacions tècniques del robot, siguin del hardware o del software, a més d'una anàlisi de les diferents maneres de programació dels quals es disposa.

Després de realitzar una anàlisi detallat, s'han obtingut uns resultats que han permès descartar aquelles idees menys favorables per a la implementació de l'aplicació del robot. A continuació, es citen els objectius definitius.

- Dur a terme la posada en marxa del robot.
- Poder moure el robot per un entorn desconegut.
- Saber reconèixer els obstacles que es troben en el entorn del robot, amb l'objectiu de poder moure's sense topar-se.

Un cop els objectius principals s'han definit, s'hauran de detallar les especificacions tècniques corresponents.

- Dur a terme la posada en marxa del robot.
    - Configurar tots els paràmetres tant del robot com de l'ordinador, per poder treballar amb ell.
- Poder moure el robot per un entorn desconegut.
    - Moure el robot a través d'un teclat extern al robot.
- Saber reconèixer els obstacles que es troben en el entorn del robot, amb l'objectiu de poder moure's sense topar-se.
    - Detectar els obstacles a partir del sensor LIDAR.

4. Marc conceptual.

En aquest capítol està centrat per entendre els conceptes necessaris per poder comprendre i desenvolupar en l'entorn de TurtleBot3 Burger.

4.1. Ubuntu.

Ubuntu és el sistema operatiu lliure més popular, amb més de 20 milions d'usuaris arreu del món. Està basat en el nucli Linux, l'escriptori GNOME i l'entorn Unity, completament traduït al català. Disposa de més de 25.000 programes gratuïts fàcilment instal·lables, els principals dels quals es troben disponibles en català.

Entre els programes que s'instal·len per defecte es troba el navegador Firefox, el gestor de correu Thunderbird, el paquet ofimàtic LibreOffice, el programa de gestió de fotos Shotwell, el programa de missatgeria Empathy, el reproductor de música Rhythmbox, el reproductor de vídeo Totem, l'enregistrador de CD/DVD Brasero, un client de BitTorrent i moltes altres eines que permeten treballar i navegar per Internet des del primer moment. I tot completament de franc.

Ubuntu disposa d'una versió estàndard d'escriptori per a 32 i 64 bits. [7]

4.2. ROS.

4.2.1. Què és ROS.

ROS, també conegut com a Sistema Operatiu Robòtic, és una estructura de programari que permet el desenvolupament de programes per a robots, proveint funcionalitats semblants a un sistema operatiu. ROS implementa els serveis típics d'un sistema operatiu tals com desenvolupament de hardware, és a dir, controladors de dispositius a baix nivell; implementació de les funcionalitats més comunes, com el pas de missatges entre processos i manegament de paquets.

ROS té dues parts bàsiques: la part del sistema operatiu, ROS, com s'ha descrit anteriorment i ros-pkg, una suite de paquets aportada per la contribució d'usuaris

(organitzats en conjunts anomenats pilars) que implementen les funcionalitats de comptes com localització i mapa simultani, planificació, percepció, simulació, etc. [8]

4.2.2. Per què ROS.

ROS és el SDK definitiu per al desenvolupament d'aplicacions de robots. Proporciona una arquitectura distribuïda i conté algoritmes implementats de l'estat de l'art, mantinguts per experts en el camp. Tot amb una llicència permissiva que en uns pocs anys canviarà el panorama de la robòtica i és àmpliament adoptat per la investigació, centres educatius i la indústria.

Els nodes de ROS no han d'estar en el mateix sistema ni tan sols ser de la mateixa arquitectura. Es pot tenir un TurtleBot3 publicant missatges i un ordinador portàtil subscrivint-se a ells. Al ser un sistema operatiu de codi obert mantingut, utilitzat i desenvolupat per nombrosos usuaris i entitats, es tracta d'una infraestructura molt flexible i adaptable a diverses necessitats.

Un dels principals avantatges de treballar amb ROS és que disposa d'un ventall molt gran de codis de programació per a diferents àmbits, fet que simplifica la tasca de recerca, ja que es pot partir d'una base més o menys construïda, i després desenvolupar-la fins al punt desitjat.

A més, la comunicació de dades ROS és compatible no només per un sistema operatiu, sinó també per múltiples sistemes operatius, maquinari i programes, aspecte que el fa molt adequat per al desenvolupament de robots, on es combinen diversos maquinaris.

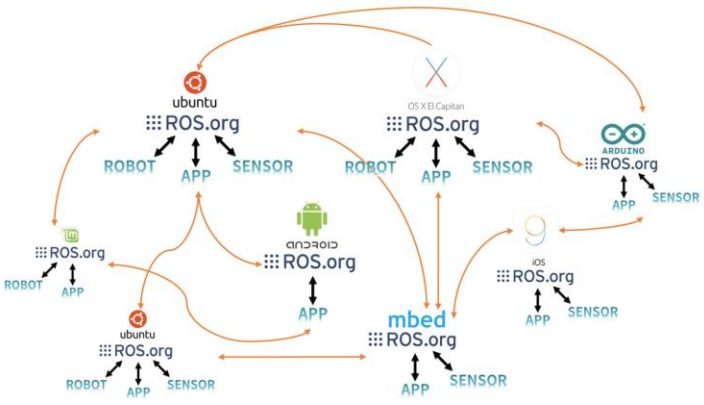


Figura 4.1. ROS Multi-comunication

Font: Elaboració pròpia

4.2.3. Història de ROS.

ROS va ser originalment desenvolupat el 2007 pel Laboratori d'Intel·ligència Artificial de Stanford (SAIL) amb el suport del projecte Standfor AI Robot. El 2008 es va continuar el desenvolupament en Willow Garage, incubadora d'empreses i un laboratori d'investigació robòtica dedicada a la creació de programari de codi obert per a les aplicacions per a robots personals amb més de 20 institucions col·laborant en el desenvolupament. El febrer de 2013, ROS es va transferir a la Fundació Robòtica de codi obert.

ROS està alliberada sota els termes de la llicència BSD (Berkeley Software Distribution) i un programari de codi obert, gratuït per a ús comercial i d'investigació.

ROS promou la reutilització de codi, així els desenvolupadors i científics no han de reinventar la ronda tot el temps, podent agafar el codi dels repositoris, millorar-lo i compartir-lo de nou.

4.2.4. Conceptes principals de ROS.

- **Màster:** El Mestre ROS proporciona el registre de nom i consulta el rest del gràfic de computació. Sense el mestre, els nodes no serien capaços de trobar el resta de nodes, intercanviar missatges o invocar serveis.

- **Nodes:** Els nodes són els processos que realitzen la computació. Un sistema de control de robots està comprès per molts nodes. Per exemple, un node controla un làser, un node controla els motors de les rodes, un node porta a terme la localització, un node realitza una trajectòria de planificació, un node proporciona una vista gràfica del sistema, i així successivament. Un node de ROS s'escriu amb l'ús de les llibreries client de ROS, roscpp i rospy.

  Un node té un nom únic en el sistema. El nom és utilitzat per permetre al node connectar-se amb el resta de nodes utilitzant noms sense ambigüitat. Un node pot ser escrit usant diferents llibreries com: roscpp i rospy; roscpp és per C ++ i rospy és per Python.

ROS té eines per gestionar els nodes i donar-los informació sobre ells com a rosnode. L'eina *rosnode* és una eina de línia de comanda per mostrar informació sobre els nodes:

- **rosnode info node:** mostra informació sobre el node.
- **rosnode kill node:** Mata el node o envia un senyal per matar-lo.
- **rosnode lis:** mostra una llista amb els nodes actius.
- **rosnode machine hostname:** Llista els nodes executant en una màquina en concret.
- **rosnode ping node:** Mostra la connectivitat amb el node.
- **rosnode cleanup:** neteja la informació de registre per a nodes inaccessibles.

- **Servidor de paràmetres:** El servidor de paràmetres permet emmagatzemar dades mitjançant una localització central, sovint és part del Master.

- **Missatges:** Els nodes es comuniquen mitjançant el pas de missatges. Un missatge és una estructura de dades compostes. Un missatge comprèn una combinació de tipus primitius i missatges. Els tipus primitius de dades (enter, punt flotant, booleà, etc.) estan suportats, com els *arrays* de tipus primitiu. Els missatges poden incloure estructures i matrius (com les estructures en C).

- **Tòpics:** Els tòpics són canals de comunicació per transmetre dades. Els tòpics es poden transmetre sense una comunicació directa entre nodes, significa que la producció i consum de dades està desacoblat. Un tema pot tenir diversos subscriptors.

  Cada tòpic és fortament tipat per un tipus de missatge de ROS que es publica, els nodes poden rebre missatges d'un determinat tipus. Un node pot subscriure's a un sol tema si té el mateix tipus de missatge.

  El tema en ROS es pot transmetre usant TCP / IP o UDP. El transport basat en TCP / IP és conegut com a TCPROS i utilitza TCP / IP per a la connexió.

- **Serveis:** El model de publicació / subscripció és un paradigma de comunicació molt flexible, però en molts casos el transport en un únic sentit no és suficient per a les interaccions de petició i resposta que sovint es requereixen en un sistema distribuït. La petició i resposta es realitza a través dels serveis, que es defineixen a partir d'una estructura de missatges: una per a la petició i una altra per a la resposta. Un node proporciona un servei amb un nom i un client utilitza aquest servei mitjançant l'enviament del missatge de petició i espera a la resposta. La llibreria client de ROS generalment presenta aquesta interacció com si fos una trucada a un procediment remot.

- **Bosses:** Les bosses és un format per guardar i reproduir de nous missatges de ROS. Les bosses són un mecanisme important per a l'emmagatzematge de dades, com a lectures de sensors, que poden ser difícilment adquirits però són necessàries per al

5. Generació i plantejament de possibles alternatives de solució.

Plantejada la manera com s'implementarien els diversos objectius proposats, el següent pas consisteix a proposar un seguit d'alternatives de solució, les quals, a partir d'un anàlisi profund d'aquestes, permeti escollir la més idònia. Tot això és el que es podrà trobar dintre d'aquest apartat amb les corresponents especificacions tècniques més rellevants per a poder valorar i analitzar.

TurtleBot 3 Burger bàsicament està pensat per ser programat amb l'entorn ROS.

La distribució ROS és un conjunt de versions de paquets ROS. Semblants a les distribucions de Linux (p. Ex. Ubuntu). El propòsit de les distribucions de ROS és deixar que els desenvolupadors funcionin contra una base de codi relativament estable fins que estiguin preparats per fer tot el possible.

És per aquest motiu que no es planteja cap alternativa més enllà de programar amb l'entorn ROS. Però dins de ROS es troben diverses distribucions d'instal·lació, i fins i tot diferents maneres de programar els nodes que enllacen els diferents components del TurtleBot3 Burger. Per tant, es plantegen dos camins per poder realitzar la instal·lació de l'entorn pel robot Turtlebot3 Burger, és per això que s'estudiarà cada un d'ells per poder decidir amb quins dels dos es treballa.

Es troben moltes distribucions diferents dins de l'entorn de ROS, però principalment se'n destaquen dos per les seves altes capacitats de desenvolupament dins aquest entorn. El primer d'ells és ROS lunar Loggerhead, el qual és l'última versió que ha tret al mercat ROS, i ROS Kinetic Kame, la qual és una distribució que ROS va llançar el maig del 2016 i es troba força dispersa i unificada en el món de ROS.

6. Selecció de l'alternativa més adequada.

Un cop plantejades les alternatives en el punt anterior, s'ha d'escollir quina de les solucions proposades serà la que es durà a terme. Per poder decidir quina de les dues solucions proposades serà la que es desenvolupi, principalment s'ha reflexionat amb quina de les dues possibles es disposa més recursos per posar-la en marxa.

Per tant, després de fer l'anàlisi de les diferents opcions, finalment s'ha decidit dur a terme el projecte de Turtlebot 3 Burger a través de ROS Kinetic Kame. S'ha cregut què és la manera més eficient de treballar amb el temps disponible del qual es disposa, i amb els recursos dels quals es disposa.

ROS Kinetic Kame, tot i no ser l'última versió de ROS, és la que hi ha més llibreries relacionades amb Turtlebot procedents de ROBOTIS, amb la qual cosa facilita bastant la posada en marxa del projecte. A més disposen d'una pauta per iniciar-se en el món de Turtlebot, sigui el 2 o el 3 com és en aquest cas.

ROS Kinetic Kame, en aquest cas encarat a Turtlebot, disposa d'una llibreria per poder fer mapejats per l'entorn on es troba el robot, permetent així conèixer la zona de treball que aquest s'ha de desplaçar. Un altre aspecte a tenir en compte és que fent el mapejat de la superfície, permet saber quines són les limitacions de l'entorn, parets, objectes, i com es pot observar a través de l'ordinador es pot conèixer la ubicació precisa del robot en cada moment.

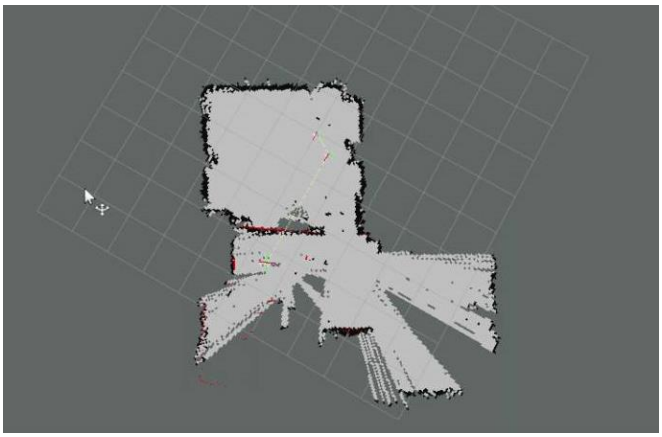


Figura 6.1. Mapejat Turtlebot3

Font: Elaboració pròpia

Per tant, per poder aconseguir que el robot es mogui d'una posició a una coneguda, en conèixer l'entorn pel qual s'ha de moure, només serà necessari marcar-li en quin punt ha d'anar, i aquest anirà esquivant els obstacles fins al punt desitjat. Això és perquè treballa a un entorn conegut.

7. Anàlisi de viabilitat.

La viabilitat tècnica és la condició que determina si és possible el funcionament i desenvolupament del projecte atenent a les seves característiques tecnològiques. Un cop estudiats els objectius finals del projecte i cadascun dels elements necessaris que els compondran, es determinarà el model exacte de cada element necessari per poder dur a terme el desenvolupament del projecte.

7.1. Viabilitat tècnica.

Com ja s'ha esmentat en l'apartat 6, per a dur a terme el projecte, finalment s'ha decidit programar amb l'entorn ROS Kinetic, per tant, primerament s'ha de dur a terme la instal·lació de Linux Ubuntu a la PC remota, que serà on aniran tots els paquets instal·lats per poder moure el robot.

Una vegada s'ha dut a terme la instal·lació d'Ubuntu, s'ha d'instal·lar i preparar l'entorn ROS i els paquets disponibles per poder fer la posada en marxa de TurtleBot3 Burger. Per poder fer els primers passos amb el robot una vegada es té tot instal·lat, serà inicialitzant uns exemples, on a partir d'ells es durà a terme una aplicació.

7.2. Viabilitat mediambiental.

Aquest projecte no consta d'un gran impacte mediambiental, ja que en tractar-se d'un projecte de caràcter educatiu, és a dir que no anirà destinat a la planta de producció d'una empresa, no implica uns impactes mediambientals ni per maquinària, ni per fase d'explotació d'un hipotètic producte.

Tot i que el projecte no té un impacte mediambiental elevat, es podria dir que hi ha un punt a destacar, la bateria durant la fase de reciclatge. La bateria està composta de polímer de liti, la qual cosa significa que en la seva fase de reciclatge necessitarà un tractament especial, el qual consistirà a portar la bateria a una planta de reciclatge, ja que és necessari que sigui tractada, perquè es tracta d'un material contaminant pel medi ambient.

Durant la fase d'ús del TurtleBot3 Burger, no implica un impacte mediambiental més enllà del que pugui generar una persona en el seu dia quotidià.

7.3. Viabilitat econòmica.

Per poder desenvolupar la viabilitat econòmica del projecte del TurtleBot3 Burger, com que no es tracta d'un projecte orientat a un producte, s'ha hagut d'abordar des d'un punt de vista totalment diferent. Aquest projecte va destinat a la universitat Tecnocampus de Mataró, ja que és un projecte de recerca i desenvolupament del robot mòbil TurtleBot3 Burger, per tant, es calcularan els costos de material necessaris per abordar el projecte i les hores que s'han necessitat per elaborar el projecte.

Material	**Cost**
TurtleBot3 Burger	496,11 €
Lenovo ThinkPad P51s - 20HB000VSP INTEL I7-7500U/8GB./256GB SSD/15,6" FULL HD./QUADRO M520M 2GB./W10PRO	1.323,92 €
Tarjeta SD Raspberry Pi MicroSD 32 GB	20,49 €
Cámara Raspberry, Módulo de vídeo V2 Daylight, 1 canal, CSI-2	22,07 €
COST TOTAL	2.357,73 €

Taula 7.1. Cost d'inversió

Font: Elaboració pròpia

Com es pot observar en la taula 7.1. tots els costos d'inversió són de materials, això és degut al fet que no es tracta d'un projecte el qual es necessiti maquinària per fabricar algun producte, sinó que simplement ha estat necessari el material.

8. Planificació.

S'ha realitzat una planificació del projecte on es divideix en les tasques esmentades en el següent punt, tenint en compte que la data d'inici és el dia 30 d'octubre de 2017 i finalitzarà el dia 21 de maig de 2018.

8.1. Tasques principals

La totalitat del projecte està dividida en 15 activitats principals. No obstant, degut a la grandària d'alguna d'aquestes activitats s'han dividit en subactivitats. En quant al camí crític, a partir de les pre-relacions [veure annex I], es marcaran les activitats en vermell en el diagrama de Gantt. El projecte per tant, consta de les següents activitats principals:

Nom tasca	Dedicació	Inici	Finalització
1. Identificar el problema	68 hores	dis 25/11/17	diu 10/12/17
1.1. Objecte del projecte	5 hores	dis 25/11/17	dis 25/11/17
1.2. Revisió d'antecedents	28 hores	dis 25/11/17	dim 05/12/17
1.3. Necessitats d'informació	20 hores	dij 30/11/17	dim 05/12/17
1.4. Abast del projecte	5 hores	dim 06/12/17	div 08/12/17
1.5. Objectius i especificacions tècniques	10 hores	dij 07/12/17	diu 10/12/17
2. Selecció de l'alternativa més adequada	28 hores	diu 10/12/17	dim 20/12/17
2.1. Cerca d'alternatives	16 hores	diu 10/12/17	div 15/12/17
2.2. Estudi d'alternatives	10 hores	div 15/12/17	diu 17/12/17
2.3. Selecció de l'alternativa més adequada	2 hores	diu 17/12/17	dim 20/12/17
3. Anàlisi de la viabilitat	20 hores	dim 20/12/17	dij 28/12/17
3.1. Anàlisi de viabilitat tècnica	8 hores	dim 20/12/17	dill 25/12/17
3.2. Anàlisi de viabilitat mediambiental	6 hores	div 22/12/17	dim 26/12/17
3.3. Anàlisi de viabilitat económica	6 hores	dis 23/12/17	dij 28/12/17
4. Desenvolupament de la solució	10 hores	dill 25/12/17	dim 02/01/18
5. Planificació	10 hores	dim 02/01/18	div 12/01/18
5.1. Ennumeració de tasques generals	2 hores	dim 02/01/18	div 05/01/18
5.2. Establir duracións tasques	1 hora	div 05/01/18	dis 06/01/18
5.3. Establir tasques predecesores	2 hores	dis 06/01/18	dim 09/01/18
5.4. Elaboració de la planificació en MS-Project	5 hores	dim 09/01/18	div 11/01/18
LLIURAMENT AVANTPROJECTE DILL 12/01/18			
6. Objectius	8 hores	dis 20/01/18	diu 21/01/18
6.1. Propòsit	2 hores	dis 20/01/18	diu 21/01/18
6.2. Finalitat	2 hores	dis 20/01/18	diu 21/01/18

6.3. Objecte	2 hores	diu 21/01/18	diu 21/01/18
6.4. Abast	2 hores	diu 21/01/18	diu 21/01/18
7. Introducció	3 hores	dis 27/01/18	diu 28/01/18
7.1. Objectius	1 hora	dis 27/01/18	diu 28/01/18
7.2. Revisió antecedents i necessitats informació	1 hora	dis 27/01/18	diu 28/01/18
7.3. Límits del projecte	1 hora	dis 27/01/18	diu 28/01/18
8. Objectius de detall i especificacions tècniques	1 hora	diu 28/01/18	dim 30/01/18
9. Marc conceptual	4 hores	dim 30/01/18	dis 03/02/18
9.1. Ubuntu	2 hores	dim 30/01/18	dis 03/02/18
9.2. ROS	2 hores	div 02/02/18	dis 03/02/18
10. Selecció de l'alternativa més adequada	2 hores	dis 03/02/18	diu 04/02/18
10.1. Estudi d'alternatives	1 hora	dis 03/02/18	diu 04/02/18
10.2. Selecció de l'alternativa més adequada	1 hora	diu 04/02/18	diu 04/02/18
11. Anàlisi de la viabilitat	3 hores	dis 17/02/18	div 23/02/18
11.1. Anàlisi de viabilitat tècnica	1 hora	dis 17/02/18	diu 18/02/18
11.2. Anàlisi de viabilitat mediambiental	1 hora	diu 18/02/18	dill 19/02/18
11.3. Anàlisi de viabilitat econòmica	1 hora	dim 21/02/18	div 23/02/18
12. Planificació	8 hores	div 23/02/18	div 02/03/18
13. Instal·lació entorn TurtleBot3 Burger	20 hores	dis 10/03/18	dill 18/03/18
LLIURAMENT MEMÒRIA INTERMÈDIA DILL 19/03/18			
14. Aplicacions TurtleBot3 Burger	110 hores	dim 21/03/18	dis 21/05/18
14.1. Cerca aplicacions	30 hores	dim 21/03/18	div 30/03/18
14.2. Comprensió arxius aplicacions	35 hores	div 30/03/18	dij 05/04/18
14.3. Execució aplicacions	45 hores	dij 05/04/18	dis 21/04/18
15. Aplicació TurtleBot3 Burger	105 hores	dis 21/04/18	diu 20/05/18
15.1. Recopilació de aplicacions	35 hores	dis 21/04/18	diu 29/04/18
15.2. Elaboració aplicació	70 hores	diu 29/04/18	diu 20/05/18
LLIURAMENT MEMÒRIA FINAL DILL 21/05/18			
Duració total del projecte	**400 hores**	**dis 25/11/17**	**diu 20/05/18**

Taula 8.1. Activitats del projecte

Font: Elaboració pròpia

8.2. Cost tasques

És important indicar el cost de cada tasca, per tal d'afegir-se posteriorment en el pressupost del projecte, és per això que el cost/hora és de 30 €/hora. Un cop especificat el cost del recurs es pot imputar a cadascuna de les tasques un preu, segons la durada i el recurs:

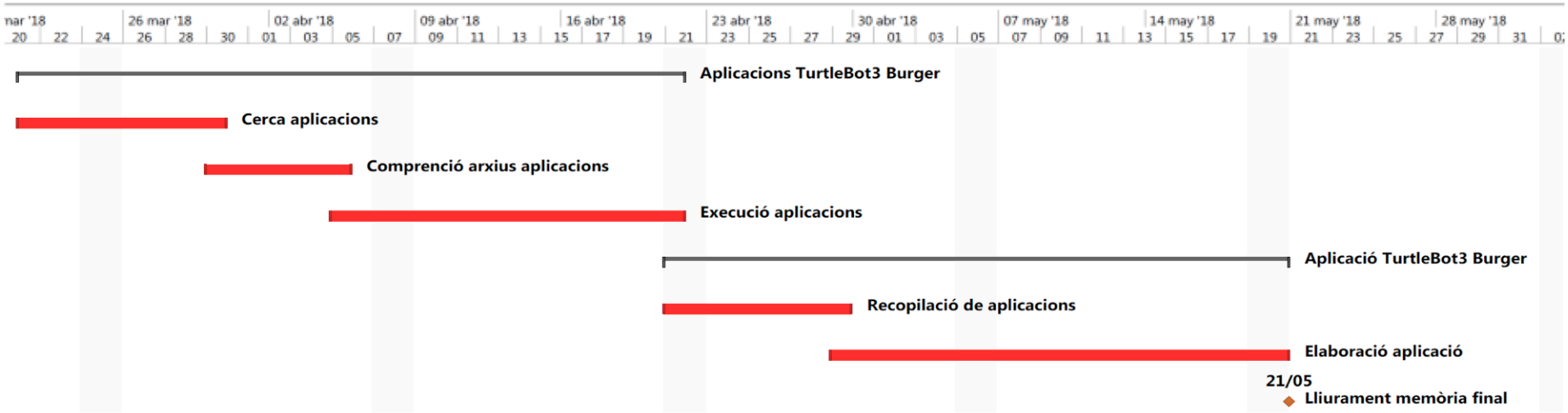
Nom tasca	**Cost activitat**
1. Identificar el problema	2.040,00 €
2. Selecció de l'alternativa més adequada	840,00 €
3. Anàlisi de la viabilitat	600,00 €
4. Desenvolupament de la solució	300,00 €
5. Planificació	300,00 €
6. Objectius	240,00 €
7. Introducció	90,00 €
8. Objectius de detall i especificacions tècniques	30,00 €
9. Marc conceptual	120,00 €
10. Selecció de l'alternativa més adequada	60,00 €
11. Anàlisi de la viabilitat	90,00 €
12. Planificació	240,00 €
13. Instal·lació entorn TurtleBot3 Burger	600,00 €
14. Aplicacions TurtleBot3 Burger	3.300,00 €
15. Aplicació TurtleBot3 Burger	3.150,00 €
Cost total del projecte	**12.000,00 €**

Taula 8.2. Cost activitats

Font: Elaboració pròpia

7.4. Diagrama de Gantt


22 ene '18
29 ene '18
05 feb '18
12 feb '18
19 feb '18
26 feb '18
05 mar '18
12 mar '18
19 mar '18
26 mar '18
02 abr '
19 21 23 25 27 29 31 02 04 06 08 10 12 14 16 18 20 22 24 26 28 02 04 06 08 10 12 14 16 18 20 22 24 26 28 30 01 03
Objectius
Propòsit
Finalitat
Objecte
Abast
Introducció
Objectius
Revisió antecedents i necessitats informació
Límits del projecte
Objectius de detall i especificacions tècniques
Marc conceptual
Ubuntu
ROS
Selecció de l'alternativa més adequada
Estudi d'alternatives
Selecció de l'alternativa més adequada
Anàlisi de la viabilitat
Anàlisi de viabilitat tècnica
Anàlisi de viabilitat mediambiental
Anàlisi de viabilitat econòmica
Planificació
Instal·lació entorn TurtleBot3 Burger
19/03
Lliurament memòria intermèdia


nar '18
26 mar '18
02 abr '18
09 abr '18
16 abr '18
23 abr '18
30 abr '18
07 may '18
14 may '18
21 may '18
28 may '18
Aplicacions TurtleBot3 Burger
Cerca aplicacions
Comprenció arxius aplicacions
Execució aplicacions
Aplicació TurtleBot3 Burger
Recopilació de aplicacions
Elaboració aplicació
21/05
Lliurament memòria final

9. Instal·lació entorn Turtlebot3 Burger.

Per poder realitzar el projecte s'ha necessitat instal·lar l'entorn de treball necessari per treballar amb el TurtleBot3 Burger. Per tant, s'han seguit una sèrie de passos per poder instal·lar l'entorn per treballar amb el robot mòbil TurtleBot3 Burger.

9.1. Instal·lar Ubuntu al PC remota.

L'escriptori d'Ubuntu és fàcil d'usar, fàcil d'instal·lar, a més és de codi obert, segur, accessible i gratuït per descarregar. Per poder instal·lar Ubuntu 16.04 és necessari disposar com a mínim de 5GB.

Una vegada descarregat l'arxiu d'Ubuntu[1] s'ha d'instal·lar en l'ordinador a través del "boot menu".

Figura 9.1. Boot menu

Font: Elaboració pròpia

A partir d'aquest pas la instal·lació és trivial, només s'han de seguir els passos fins a arribar a tenir Ubuntu instal·lat a l'ordinador.

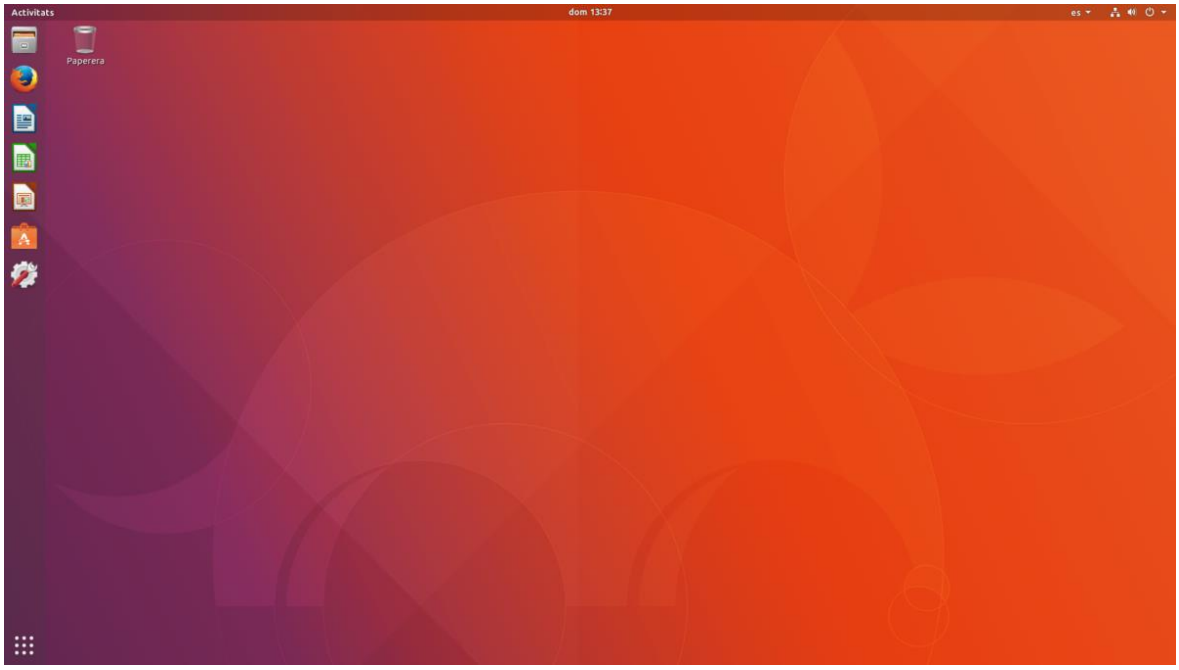


Figura 9.2. Escriptori d'Ubuntu

Font: Elaboració pròpia

9.2. Configuració del *router* sense fils.

Una vegada instal·lat Ubuntu, s'ha de configurar el router sense fils per poder enllaçar el Turtlebot3 Burger amb el Master, en aquest cas és Ubuntu.

S'ha de disposar d'una xarxa local proporcionada per un *router*, que no cal que tingui connexió a internet, ja que la finalitat d'aquesta xarxa és crear un llaç entre el robot i l'ordinador remot.

Una vegada obtingut el *router*, s'ha procedit a la seva configuració per fixar una IP a l'ordinador, i una IP diferent en el Turtlebot3 Burger, amb l'objectiu que el router sempre assigni la mateixa IP cada vegada que es connecta aquesta xarxa local, s'ha seguit el següent procediment:

1. S'ha de fixar l'adreça IP d' Ubuntu, amb l'objectiu que cada vegada que es connecti a la xarxa PRUEBA, aquest li assigni sempre la mateixa direcció IP. En aquest cas per poder fixar l'adreça IP s'ha de fer des del router, ja que és aquest qui li assigna una IP o una altra.

2. Per poder entrar en les configuracions del router, s'ha de fer a través de la adreça IP específica que aquest disposa. Per poder-la obtenir s'ha d'escriure al terminal: "*ifconfig*".

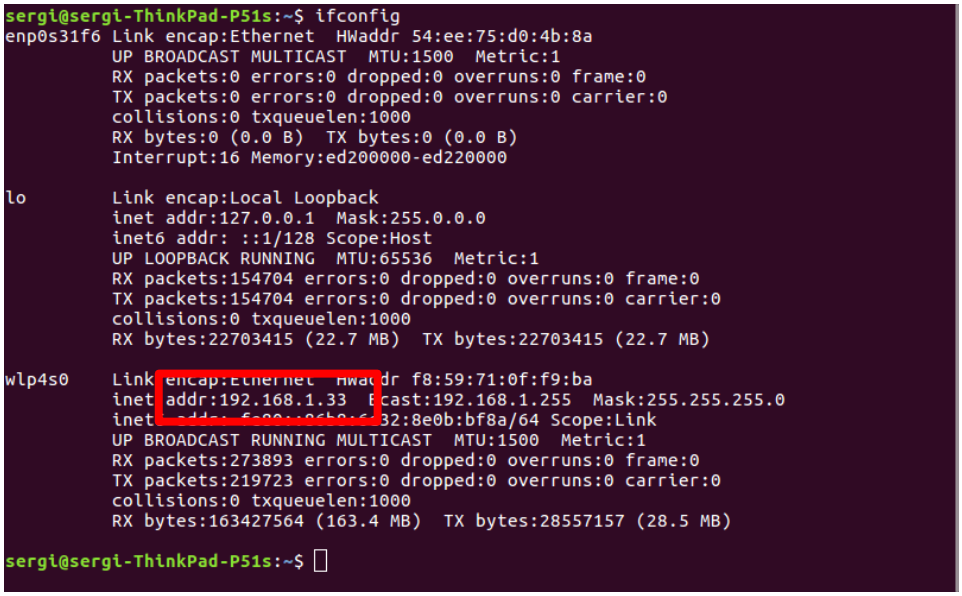


Figura 9.3. IP Terminal d'Ubuntu

Font: Elaboració pròpia

3. Per canviar la configuració del *router*, s'ha d'introduir l'adreça IP que marca el *router*, en aquest cas 192.168.1.1, i a continuació s'entrarà en el menú de configuració.

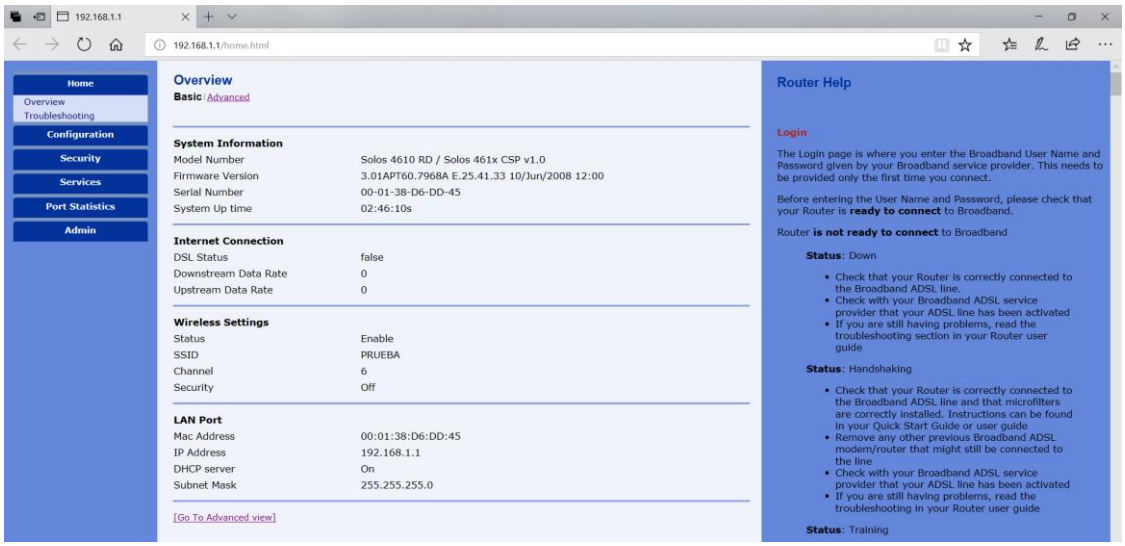


Figura 9.4. Configuració router

Font: Elaboració pròpia

4. Per fixar l'adreça IP de l'ordinador, s'ha de conèixer quin és la direcció MAC de l'ordinador. La direcció MAC és única per a cada ordinador, per tant quan el *router* detecti aquesta direcció Mac, li assignarà la IP que s'hagi establert.

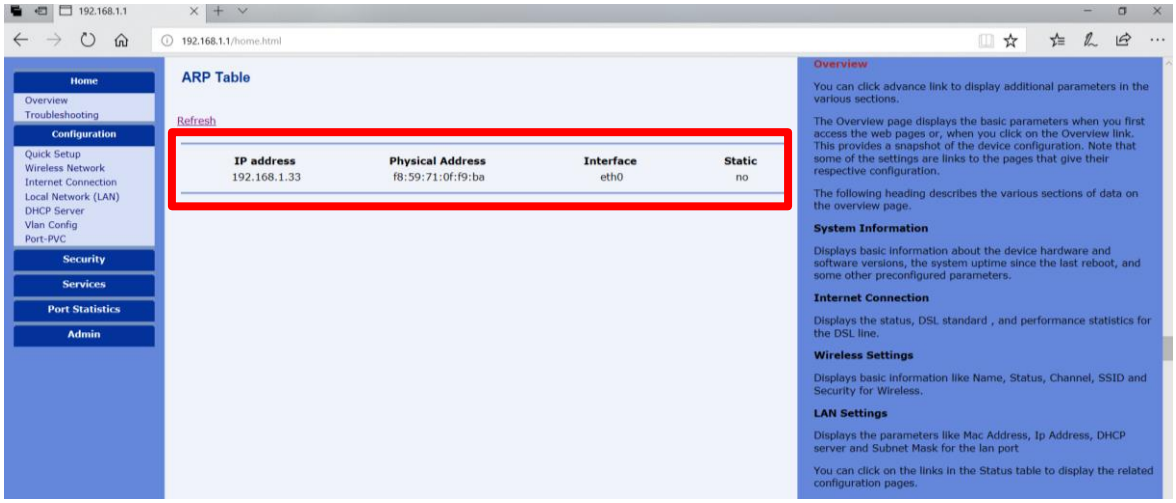


Figura 9.5. Obtenir l'adreça MAC de l'ordinador.

Font: Elaboració pròpia

5. Com es pot observar l'adreça MAC és la "f8:59:71:0f:f9:ba", i quan el router la detecti es vol que li assigni l'adreça IP "192.168.1.33". Per tant, fixant l'adreça IP s'assegura que sempre serà aquella mateixa IP, per aquest ordinador.

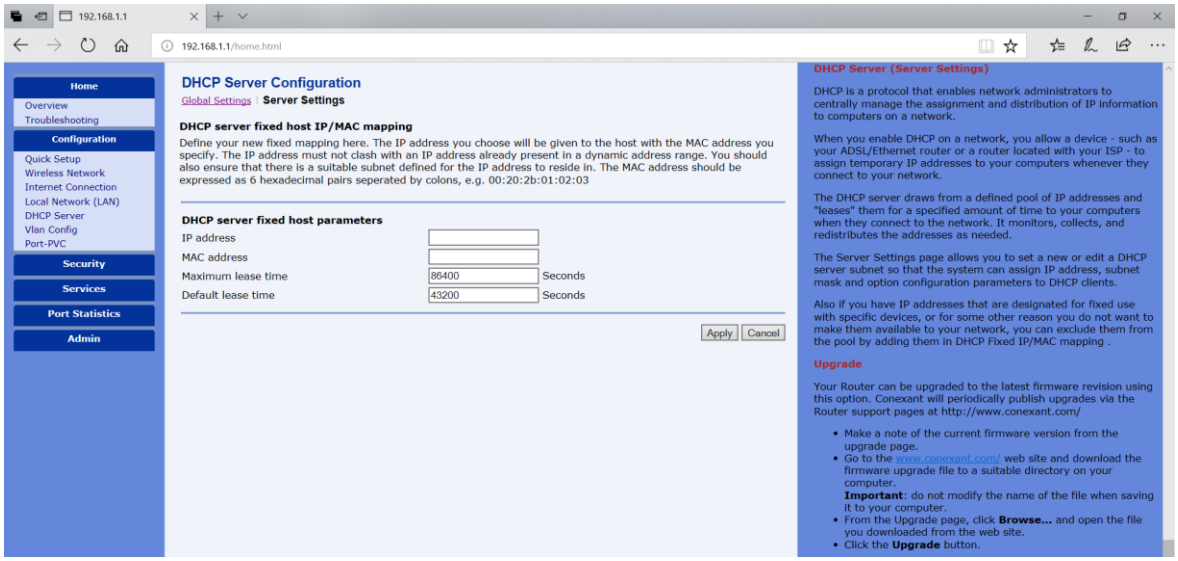


Figura 9.6. Fixar l'adreça IP amb l'adreça MAC

Font: Elaboració pròpia

6. Finalment com es pot observar l'adreça IP s'ha assignat a l'adreça MAC de l'ordinador.

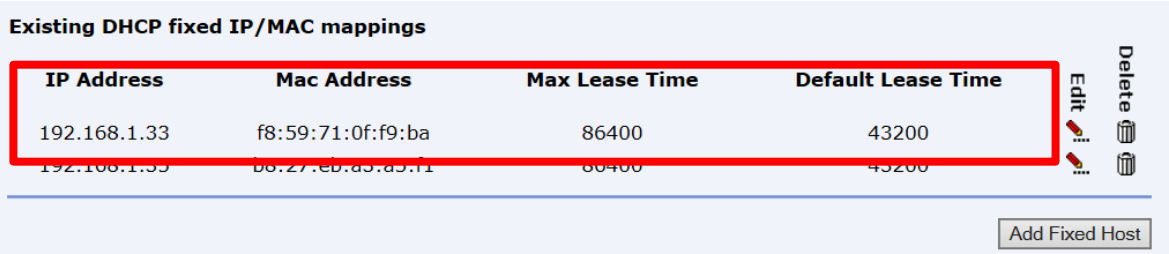


Figura 9.7. Adreça IP Ubuntu

Font: Elaboració pròpia

9.3. Instal·lació de programari per a PC.

Els continguts d'aquest capítol corresponen al PC remota que controlarà TurtleBot3, sobretot no s'ha d'aplicar aquestes instruccions al TurtleBot3.

9.3.1. Instal·lar ROS al PC remota.

Instal·lar ROS mitjançant l'ús d'un arxiu de script d'instal·lació senzilla.

- Primer mètode, instal·lar ROS mitjançant l'ús d'un arxiu de script d'instal·lació senzilla:

Tot i que no és obligatori, es recomana actualitzar tots els paquets Ubuntu instal·lats abans d'instal·lar ROS.

$ sudo apt-get Update

$ sudo apt-get upgrade

$ sudo apt-get install ros-kinetic-desktop-full

- Configurar l'entorn ROS, el següent pas és instal·lar els paquets dependents per al control TurtleBot3.

Una bona manera de comprovar és assegurar que s'estableixin variables d'entorn com **ROS_ROOT** i **ROS_PACKAGE_PATH**:

$ printenv | grep ROS

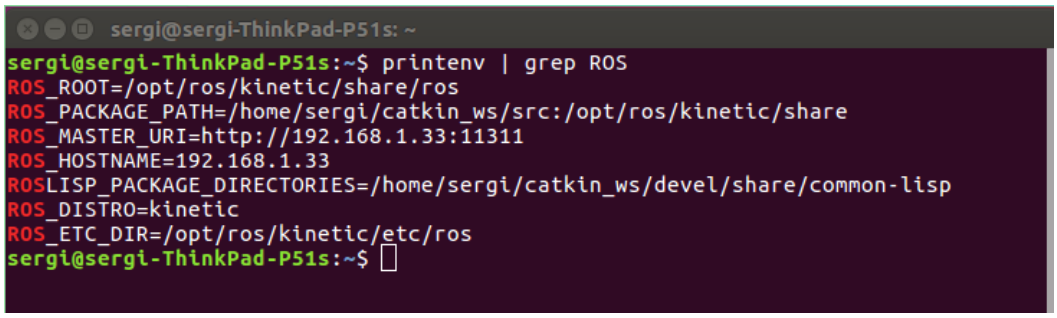


Figura 9.8. Variables d'entorn ROS

Font: Elaboració pròpia

$ source /opt/ros/kinetic/setup.bash

Anem a crear i construir un espai de treball de catkin:

$ mkdir -p ~/catkin_ws/src

$ cd ~/catkin_ws/

$ catkin_make

$ source devel/setup.bash

Per assegurar-se que l'script de configuració de l'espai de treball s'ha instal·lat correctament, s'ha d'assegurar que la variable d'entorn **ROS_PACKAGE_PATH** inclogui el directori on es troba actualment.

$ echo $ROS_PACKAGE_PATH

Si s'ha instal·lat correctament, hauria de sortir:

/home/youruser/catkin_ws/src:/opt/ros/kinetic/share

9.3.2. Instal·lar paquets dependents.

$ sudo apt-get install ros-kinetic-joy ros-kinetic-teleop-twist-joy ros-kinetic-teleop-twist-keyboard ros-kinetic-laser-proc ros-kinetic-rgbd-launch ros-kinetic-depthimage-to-laserscan ros-kinetic-rosserial-arduino ros-kinetic-rosserial-python ros-kinetic-rosserial-

server ros-kinetic-rosserial-client ros-kinetic-rosserial-msgs ros-kinetic-amcl ros-kinetic-map-server ros-kinetic-move-base ros-kinetic-urdf ros-kinetic-xacro ros-kinetic-compressed-image-transport ros-kinetic-rqt-image-view ros-kinetic-gmapping ros-kinetic-navigation ros-kinetic-interactive-markers

Per tal de instal·lar el entorn catkin_make, s'ha d'introduir la següent instrucció:

$ source /opt/ros/kinetic/setup.bash

$ cd ~/catkin_ws/src/

$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git

$ git clone https://github.com/ROBOTIS-GIT/turtlebot3.git

$ cd ~/catkin_ws && catkin_make

Si la instrucció catkin_make es completa sense errors, la preparació per TurtleBot3 és completada.

9.3.3. Configuració de la xarxa

Figura 9.9. Configuració IP de la PC remota

Font: [2]

$ ifconfig

$ gedit ~/.bashrc

S'ha de modificar l'adreça localhost amb l'adreça IP adquirit a la finestra del terminal superior, és a dir la IP del ordinador, en aquest cas 192.168.1.33.

```
alias eb='gedit ~/.bashrc'
alias sb='source ~/.bashrc'
alias gs='git status'
alias gp='git pull'
alias cw='cd ~/catkin_ws'
alias cs='cd ~/catkin_ws/src'
alias cm='cd ~/catkin_ws && catkin_make'
source /opt/ros/kinetic/setup.bash
source ~/catkin_ws/devel/setup.bash
export ROS_MASTER_URI=http://localhost:11311
export ROS_HOSTNAME=localhost
alias eb='gedit ~/.bashrc'
alias sb='source ~/.bashrc'
alias gs='git status'
alias gp='git pull'
alias cw='cd ~/catkin_ws'
alias cs='cd ~/catkin_ws/src'
alias cm='cd ~/catkin_ws && catkin_make'
source /opt/ros/kinetic/setup.bash
source ~/catkin_ws/devel/setup.bash
export ROS_MASTER_URI=http://192.168.1.33:11311
export ROS_HOSTNAME=192.168.1.33
export TURTLEBOT3_MODEL=burger
export PATH=$PATH:$HOME/tools/arduino-1.8.5
                    sh ▾  Amplada de la tabulació: 8 ▾    Ln 136, Col. 35 ▾    INSER
```

Figura 9.10 Arxiu Bashrc del TurtleBot3 Burger

Font: Elaboració pròpia

$ source ~/.bashrc

9.4. Configuració del programari SBC.

Per poder instal·lar l'entorn de treball del Turtlebot3 Burger, es durà a terme instal·lant Linux en memòria SD, la qual ha de ser com a mínim de 8GB d'espai buit, i a partir d'aleshores s'instal·laran tots els paquets necessaris per treballar amb el Turtlebot3 Burger.

9.4.1. Instal·lar Linux a TurtleBot3 Burger (Raspberry Pi 3).

Primerament s'ha de descarregar Ubuntu MATE 16.04 per a Raspberry Pi 3 des del següent enllaç [3].

Per instal·lar Ubuntu MATE des d'un fitxer d'imatge, s'ha fet l'ús de discs de GNOME amb l'opció Restaurar imatge de disc.

$ sudo apt-get install gnome-disk-utility

S'introdueix la memòria SD a l'ordinador, es formateja i s'instal·la la imatge de Linux per Raspberry Pi 3 descarregat amb anterioritat.

9.4.2. Instal·lar ROS i Paquets del TurtleBot 3 Burger.

Els continguts d'aquest capítol corresponen al SBC de TurtleBot3 (Raspberry Pi 3) que serà l'ordinador principal de TurtleBot3. Sobretot no s'ha d'aplicar aquestes instruccions al PC remota.

- Primer mètode, instal·lar ROS mitjançant l'ús d'un arxiu de script d'instal·lació senzilla.

$ sudo apt-get update

$ sudo apt-get upgrade

$ wget https://raw.githubusercontent.com/ROBOTIS-GIT/robotis_tools/master/install_ros_kinetic_rp3.sh && chmod 755 ./install_ros_kinetic_rp3.sh && bash ./install_ros_kinetic_rp3.sh

- Instal·lar paquets dependents, el següent pas és instal·lar els paquets dependents per al control TurtleBot3.

$ cd ~/catkin_ws/src

$ git clone https://github.com/ROBOTIS-GIT/hls_lfcd_lds_driver.git

$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git

$ git clone https://github.com/ROBOTIS-GIT/turtlebot3.git

$ cd ~/catkin_ws && catkin_make

$ sudo apt-get install ros-kinetic-joy ros-kinetic-teleop-twist-joy ros-kinetic-teleop-twist-keyboard ros-kinetic-laser-proc ros-kinetic-rgbd-launch ros-kinetic-depthimage-to-laserscan ros-kinetic-rosserial-arduino ros-kinetic-rosserial-python ros-kinetic-rosserial-server ros-kinetic-rosserial-client ros-kinetic-rosserial-msgs ros-kinetic-amcl ros-kinetic-map-server ros-kinetic-move-base ros-kinetic-urdf ros-kinetic-xacro ros-kinetic-

compressed-image-transport ros-kinetic-rqt-image-view ros-kinetic-gmapping ros-kinetic-navigation ros-kinetic-interactive-markers

Si la instrucció catkin_make es completa sense errors, la preparació per TurtleBot3 és completada.

9.4.3. Configuració USB en el TurtleBot 3 Burger.

Les comandes següents permeten utilitzar el port USB per OpenCR1.0 sense adquirir permisos externs.

$ rosrun turtlebot3_bringup create_udev_rules

9.4.4. Configuració de la xarxa en el TurtleBot3 Burger.

Com ja s'ha comentat anteriorment, ROS requereix adreces IP per comunicar-se entre TurtleBot3 Burger i la PC remota.

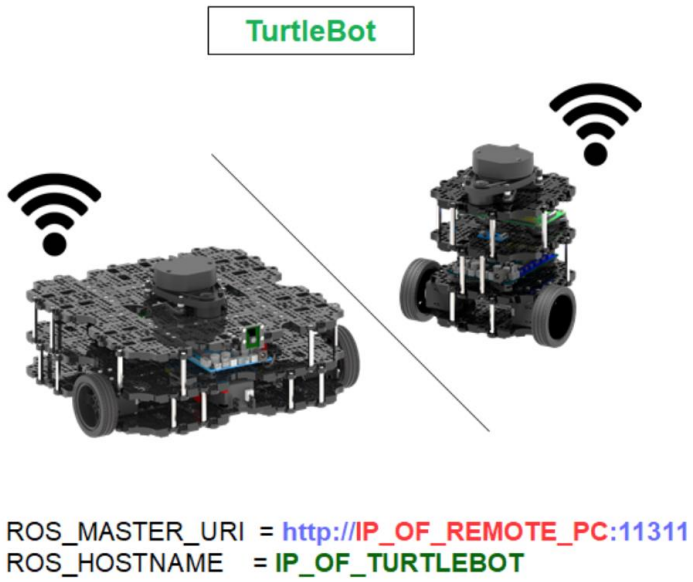


Figura 9.11. Configuració IP del Turtlebot 3 Burger

Font: [2]

Primerament s'han de seguir les instruccions de l'apartat 8.2, per fixar l'adreça IP de Turtlebot a partir de la màscara d'aquest. Per poder saber quina és l'adreça IP d'aquest s'ha d'introduir la següent instrucció en el terminal d'Ubuntu Mate.

$ ifconfig

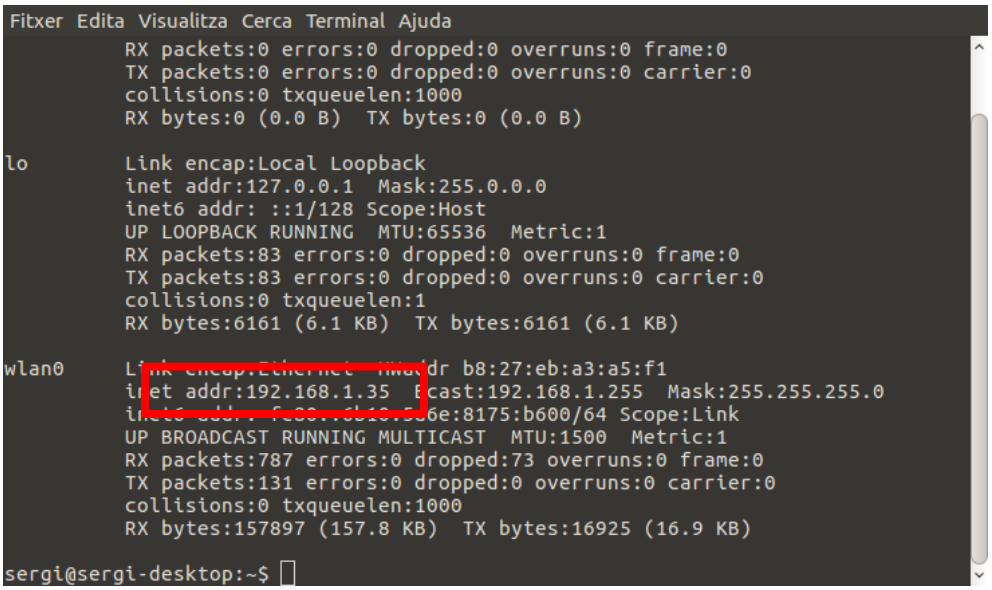


Figura 9.12. IP Terminal d'Ubuntu Mate (TurtleBot3 Burger)

Font: Elaboració pròpia

Com es pot comprovar en la figura 9.12, l'adreça IP del TurtleBot3 Burger és 192.168.1.35.

$ gedit ~/.bashrc

S'ha de substituir el "localhost" a l'adreça ROS_MASTER_URI amb l'adreça IP adquirida a la configuració de la xarxa de PC remota, i també s'ha de reemplaçar el "localhost" a l'adreça ROS_HOSTNAME amb l'adreça IP adquirida a la finestra de terminal anterior, que és l'adreça IP de TurtleBot3.

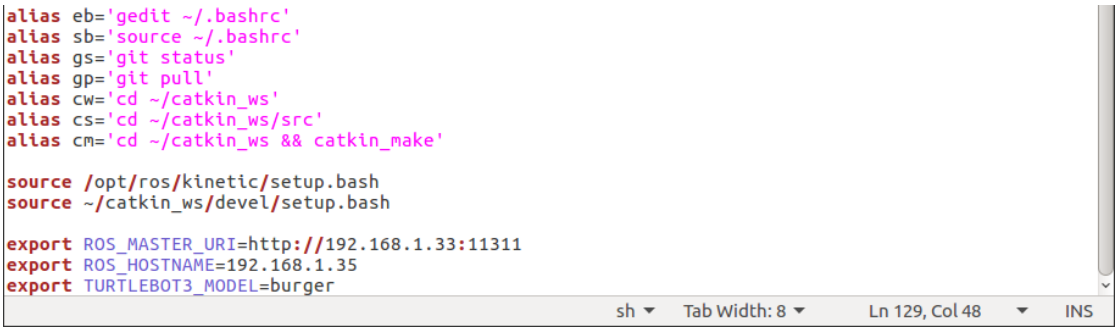


Figura 9.13 Arxiu Bashrc del TurtleBot3 Burger.

Font: Elaboració pròpia

$ source ~/.bashrc

9.5. Configuració del programari OpenCR1.0.

Els continguts d'aquest capítol corresponen a la PC remota que controlarà TurtleBot3. No s'ha d'aplicar aquesta instrucció al TurtleBot3 Burger.

OpenCR1.0 està pre-carregat amb el programari necessari per executar TurtleBot3 Burger. Si es vol modificar el programari existent o escriure un nou programari per OpenCR1.0, consultar la informació.

OpenCR1.0 controla DYNAMIXEL amb instruccions del SBC. Per poder controlar DYNAMIXEL, cal instal·lar un firmware específic al tauler. Veure les descripcions i configurar la configuració.

9.5.1. Configuració del port USB.

Les comandes següents permeten utilitzar el port USB OpenCR1.0 per carregar el programa Arduino IDE sense adquirir el permís de root.

$ wget https://raw.githubusercontent.com/ROBOTIS-GIT/OpenCR/master/99-opencr-cdc.rules

$ sudo cp ./99-opencr-cdc.rules /etc/udev/rules.d/

$ sudo udevadm control --reload-rules

$ sudo udevadm trigger

9.5.2. Configuració del compilador.

Les biblioteques OpenCR1.0 estan dissenyades per a la plataforma de 32 bits, per tant, la PC de 64 bits requereix un compilador de 32 bits per a Arduino IDE.

$ sudo apt-get install libncurses5-dev:i386

9.5.3. Instal·lar Arduino IDE.

S'ha de descarregar la versió més recent d'Arduino IDE des de la pàgina web oficial d'Arduino [4] i instal·lar el programa. En aquest cas s'ha instal·lat Arduino IDE 1.8.5, ja que era la versió més recent.

S'ha d'extreure el fitxer descarregat a la carpeta desitjada i executar el fitxer d'instal·lació a la carpeta des del terminal.

A continuació a partir del terminal s'ha d'entrar a la carpeta on s'han extret els fitxers d'Arduino IDE.

$ cd ~/tools/arduino-1.8.5

$./install.sh

Seguidament, s'ha d'obrir el fitxer de seqüència amb l'ordre següent i copiar a la part final del fitxer *"export PATH=$PATH:$HOME/tools/arduino-1.8.1"*

$ gedit ~/.bashrc

```
alias eb='gedit ~/.bashrc'
alias sb='source ~/.bashrc'
alias gs='git status'
alias gp='git pull'
alias cw='cd ~/catkin_ws'
alias cs='cd ~/catkin_ws/src'
alias cm='cd ~/catkin_ws && catkin_make'
source /opt/ros/kinetic/setup.bash
source ~/catkin_ws/devel/setup.bash
export ROS_MASTER_URI=http://localhost:11311
export ROS_HOSTNAME=localhost
alias eb='gedit ~/.bashrc'
alias sb='source ~/.bashrc'
alias gs='git status'
alias gp='git pull'
alias cw='cd ~/catkin_ws'
alias cs='cd ~/catkin_ws/src'
alias cm='cd ~/catkin_ws && catkin_make'
source /opt/ros/kinetic/setup.bash
source ~/catkin_ws/devel/setup.bash
export ROS_MASTER_URI=http://192.168.1.33:11311
export ROS_HOSTNAME=192.168.1.33
export TURTLEBOT3_MODEL=burger
export PATH=$PATH:$HOME/tools/arduino-1.8.5
```

sh ▾ Amplada de la tabulació: 8 ▾ Ln 136, Col. 35 ▾ INSER

Figura 9.14. Bashrc Arduino

Font: Elaboració pròpia

Finalment s'ha executat la següent instrucció per aplicar els canvis.

$ source ~/.bashrc

9.5.4. Arduino IDE.

Per executar Arduino IDE a la plataforma Linux, s'ha d'introduir la següent comanda en el terminal.

$ arduino

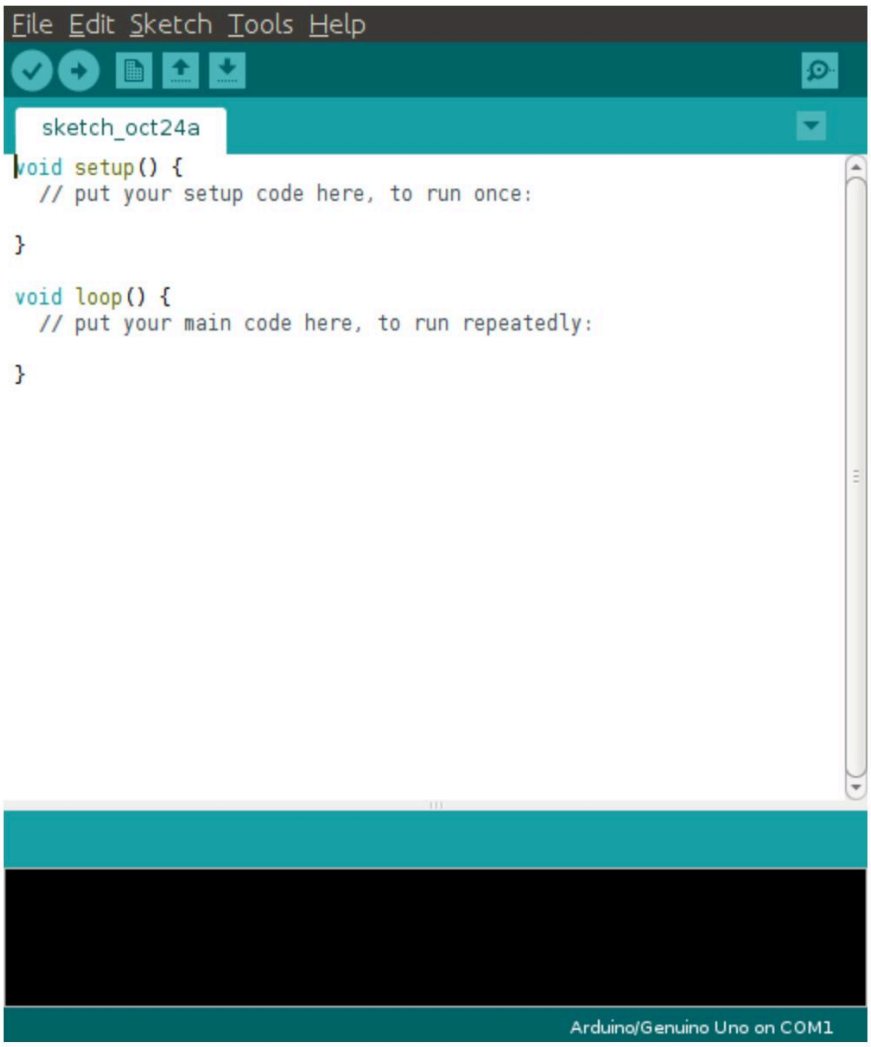


Figura 9.15. Arduino IDE

Font: Elaboració pròpia

Quan Arduino IDE s'estigui executant, s'ha anat a Fitxer → Preferències del menú del programa. Quan aparegui la finestra Preferències, s'ha de copiar i enganxar el següent text URL "Additional Boards Manager URLs".

$ https://raw.githubusercontent.com/ROBOTIS-GIT/OpenCR/master/arduino/opencr_release/package_opencr_index.json

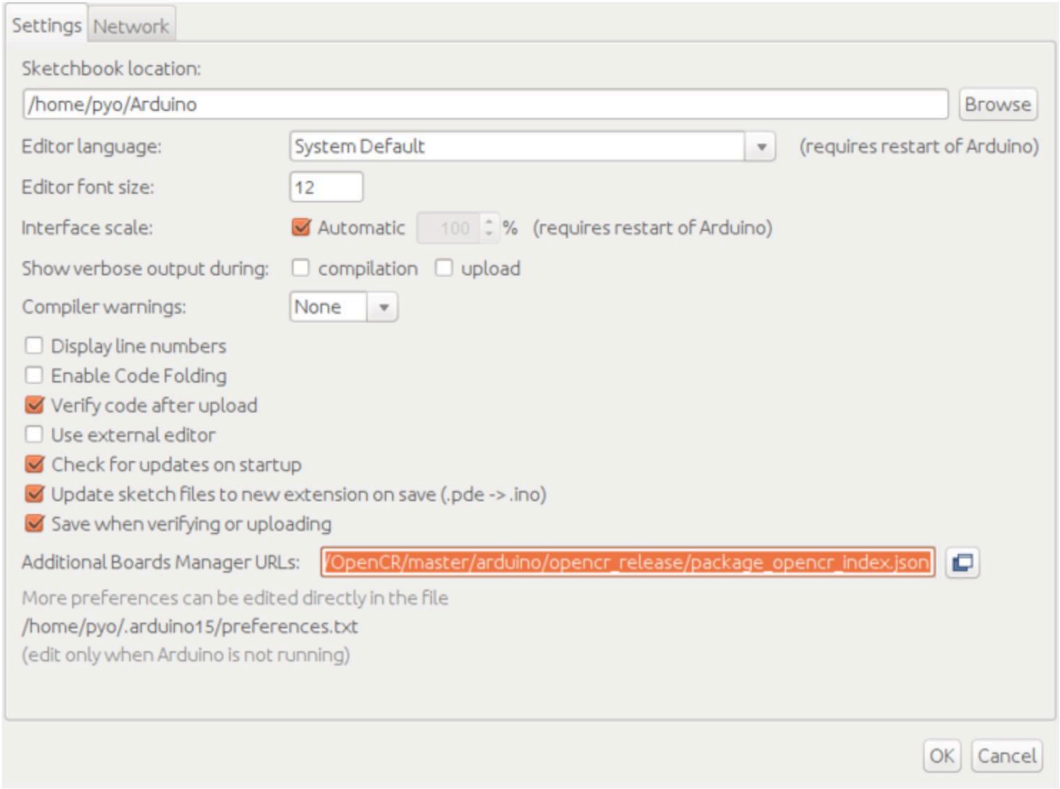


Figura 9.16. Preferències Arduino IDE

Font: Elaboració pròpia

A continuació s'ha d'instal·lar l'entorn per OpenCR per Arduino

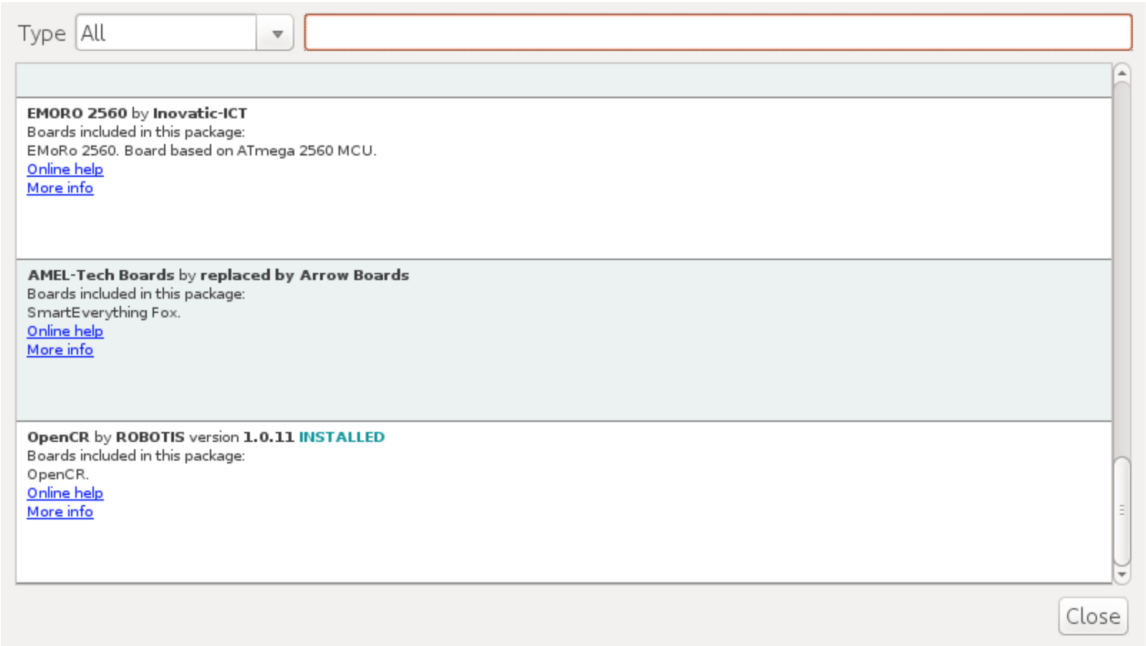


Figura 9.17. Paquet OpenCR per Arduino

Font: Elaboració pròpia

Per dur a terme les següents instruccions, s'ha de connectar la placa OpenCR a l'ordinador.

Primerament s'ha de seleccionar el port que es farà servir per a les següents instruccions.

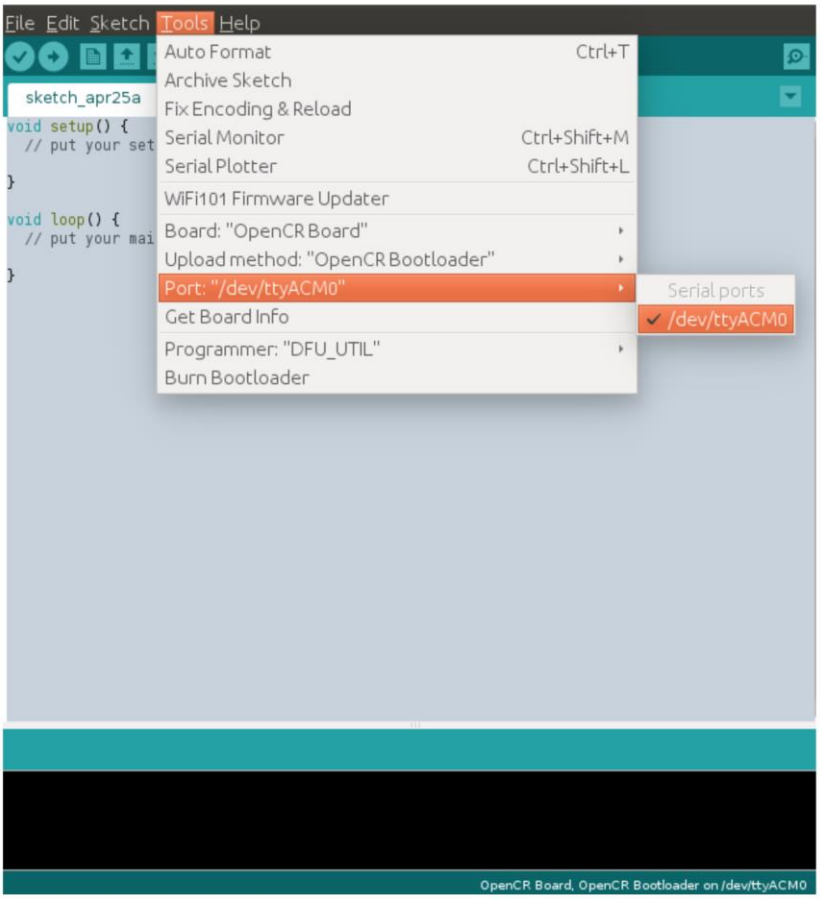


Figura 9.18. Selecció del port d'Arduino IDE

Font: Elaboració pròpia

9.5.5. Configurar el carregador d'arrencada del programa.

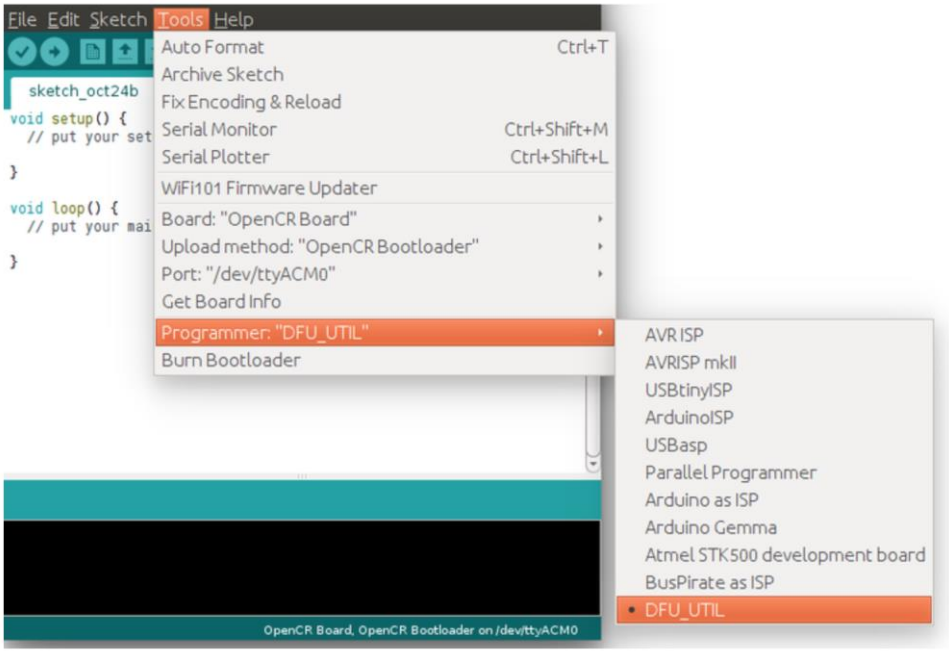


Figura 9.19. Configuració del programador

Font: Elaboració pròpia

S'ha de prémer el botó "Boot", i a continuació, premeu el botó "Reset "després de pocs segons mentre es prem el botó "Boot". Mantenir premut el botó "Boot", deixar anar el botó "Reset", i finalment, deixeu anar el botó "Boot". Això permet activar el mode DFU.

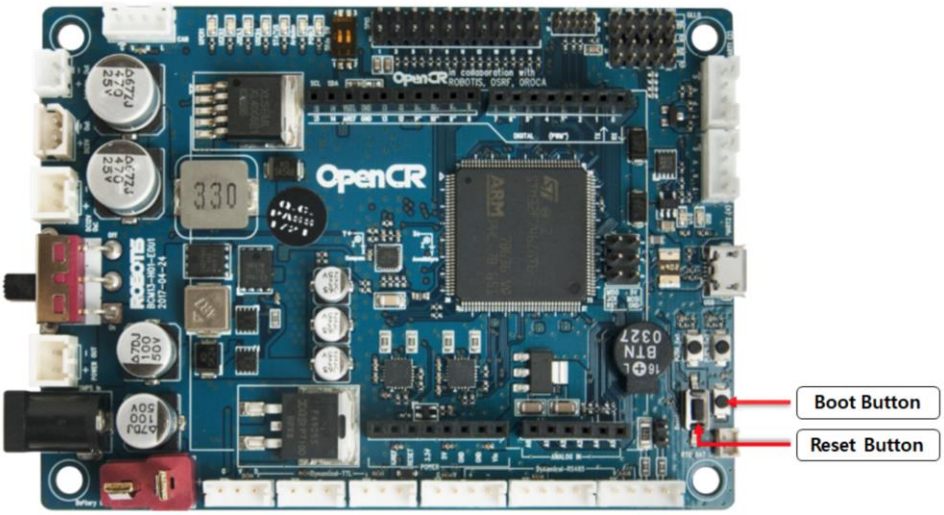


Figura 9.20. Mode DFU Arduino

Font: Elaboració pròpia

Per verificar si s'ha introduït correctament el mode DFU, introduint lsusb en el terminal, ha de sortir el mateix que en la figura 8.16.

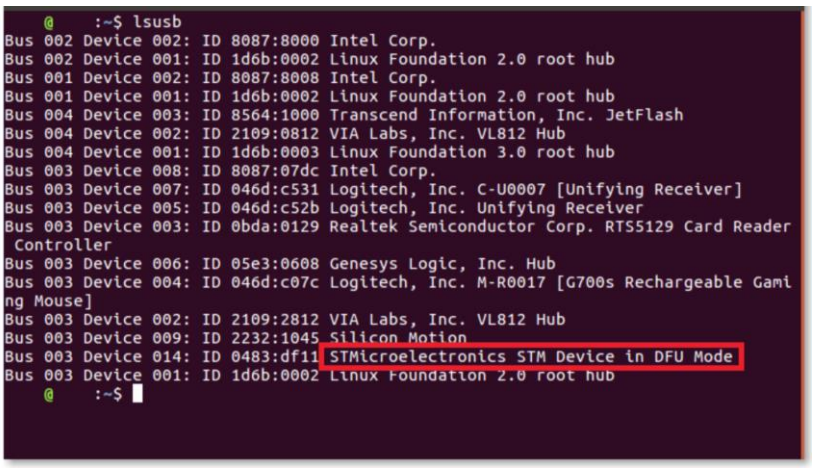


Figura 9.21. Terminal DFU mode

Font: Elaboració pròpia

Per poder descarregar Bootlander s'ha de seguir la següent instrucció.

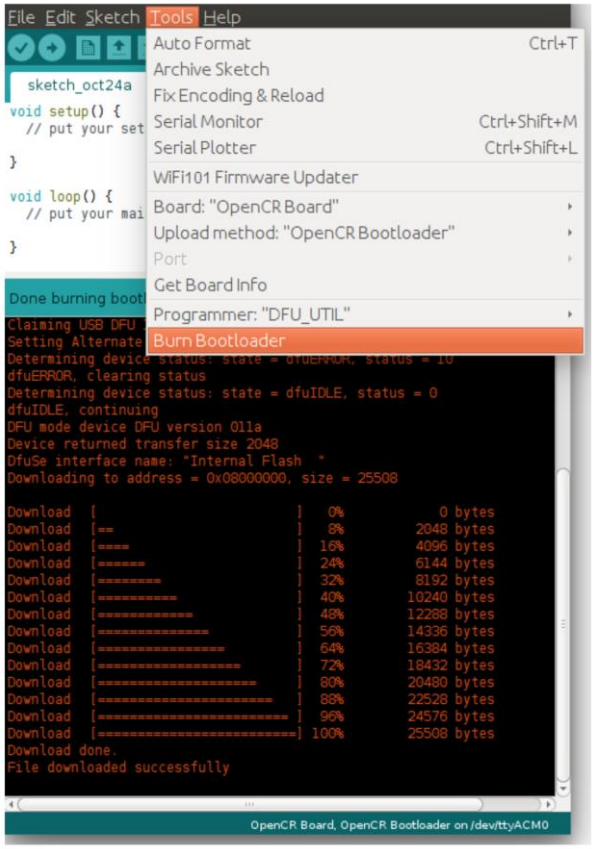


Figura 9.22. Descarregar Bootlander

Font: Elaboració pròpia

El firmware OpenCR1.0 per ROS és controlar DYNAMIXEL i sensors al ROS. El firmware es troba en un exemple d'OpenCR. Per a TurtleBot3 Burger, s'han de seguir les següents instruccions `File` → `Examples` → `turtlebot3` → `turtlebot3_burger` → `turtlebot3_core`.

I finalment "carregar-lo"

Figura 9.23. Firmware per OpenCR1.0

Font: Elaboració pròpia

9.6. Activa el servidor SSH a Raspberry Pi.

Es pot accedir a la línia d'ordres d'un Raspberry Pi remotament des d'una altra computadora o dispositiu a la mateixa xarxa mitjançant SSH en aquest cas es farà ús per dur a terme la posada en marxa del robot sense haver de connectar sempre un monitor al TurtleBot3 Burger.

Raspberry Pi funciona com un dispositiu remot, és a dir permet connectar-se des d'un altre ordinador, en aquest cas, al terminal de TurtleBot3 Burger a través de la PC remota.

Tot i que la IP del TurtleBot3 Burger ja és coneguda, ja que s'ha configurat en el apartat 9.4.4, l'ús de la comanda ifconfig mostrarà informació sobre l'estat actual de la xarxa, inclosa l'adreça IP, en aquest cas és 192.168.1.35.

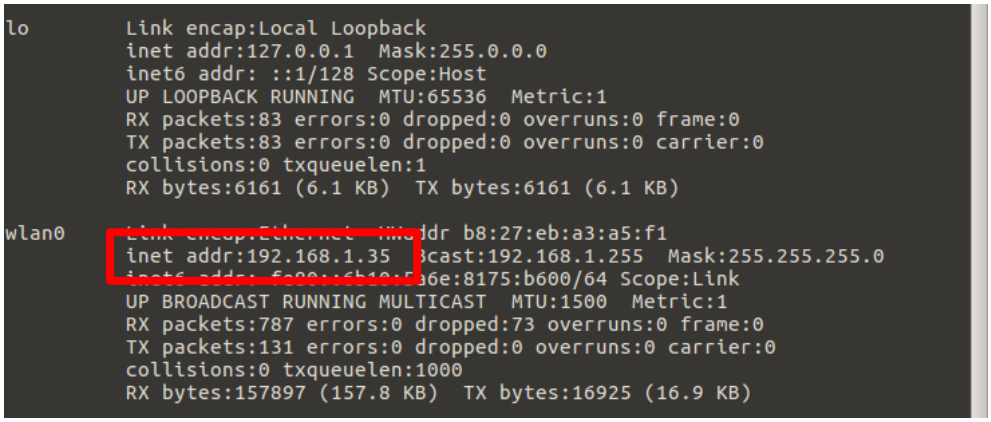


Figura 9.24. IP Terminal TurtleBot3 Burger

Font: Elaboració pròpia

A partir de la versió de novembre de 2016, Raspbian té el servidor SSH desactivat de manera predeterminada. Es pot activar manualment des de l'escriptori:

- Iniciar la configuració de Raspberry Pi des del menú Preferències
- Anar a la pestanya Interfícies
- Seleccionar *Habilitar* al costat de SSH
- Fer clic a Acceptar

Una vegada activat aquest servidor, només faltarà conèixer quin és nom d'usuari del TurtleBot 3Burger, per poder-se connectar. Com es pot visualitzar en la figura 9.25, el nom d'usuari es pot visualitzar en el terminal de la Raspberry Pi 3.

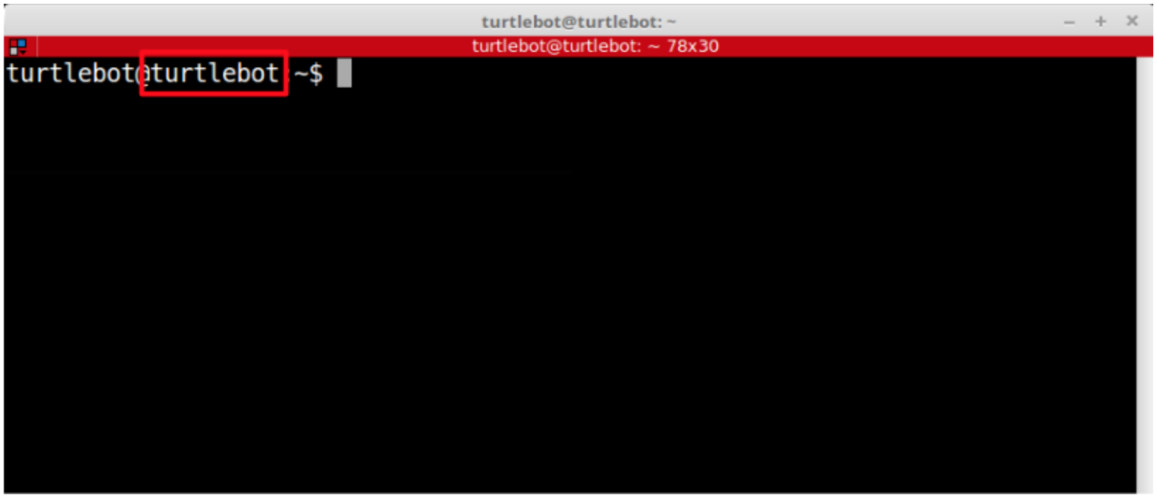


Figura. 9.25. Nom del TurtleBot3 Burger a la Raspberry Pi 3

Font: Elaboració pròpia

Per tant una vegada conegut quin és el *host name*, és a dir el nom d'usuari del TurtleBot3 Burger i la seva IP, ja es pot fer la connexió des de la PC remota al TurtlbeBot3 Burger fen ús de la següent instrucció:

$ ssh turtlebot@192.168.1.35

10. Posada en marxa del robot.

Aquesta instrucció està dissenyada per executar-se a la PC remota, sobretot no s'ha d'executar aquestes instruccions sobre TurtleBot, NO s'ha d'executar l'ordre roscore.

Assegureu-vos que l'adreça IP de cada dispositiu estigui configurada correctament.

Quan el voltatge de la bateria és inferior a 11 V, l'alarma del timbre sonarà de forma contínua i els actuadors estaran desactivats. La bateria s'ha de recarregar quan sona l'alarma del brunzidor.

Abans d'iniciar la captura de TurtleBot3, s'ha d'afegir a bashrc un comandament el qual indica quin és el model del robot.

$ gedit ~/.bashrc

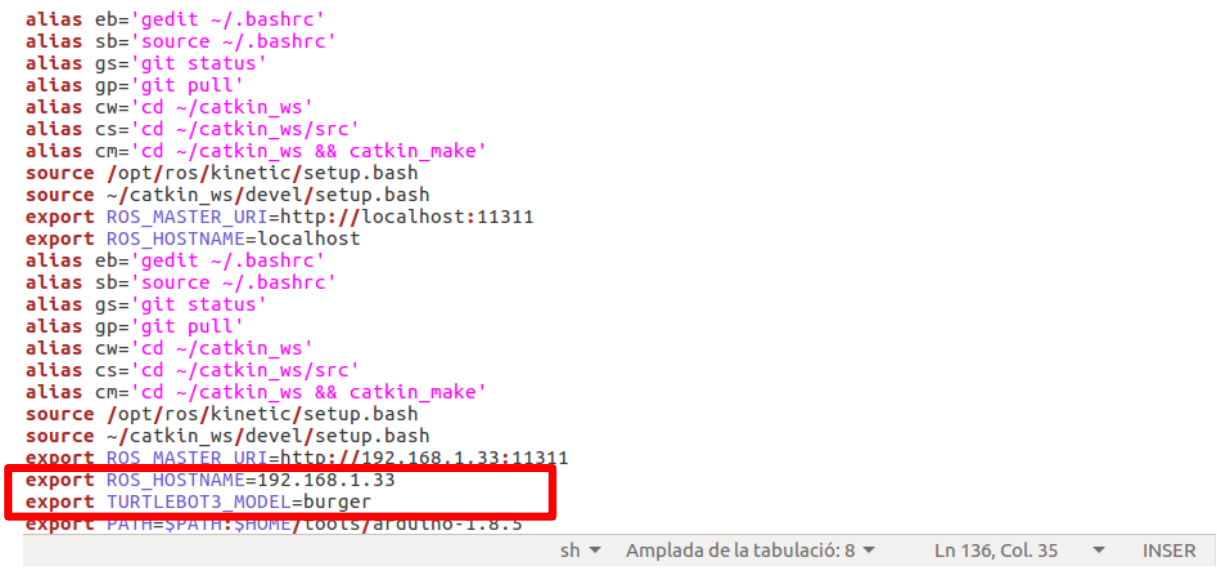


Figura 10.1. Bashrc "Model Robot"

Font: Elaboració pròpia

Per poder guardar els canvis en el fitxer bashrc:

$ source ~/.bashrc

S'ha de tenir un roscore en execució per tal que els nodes ROS es comuniquin, aquest comando s'ha d'escriure en el terminal de la PC remota, sobretot no s'ha d'executar sobre TurtleBot3 Burger.[5]

$ roscore

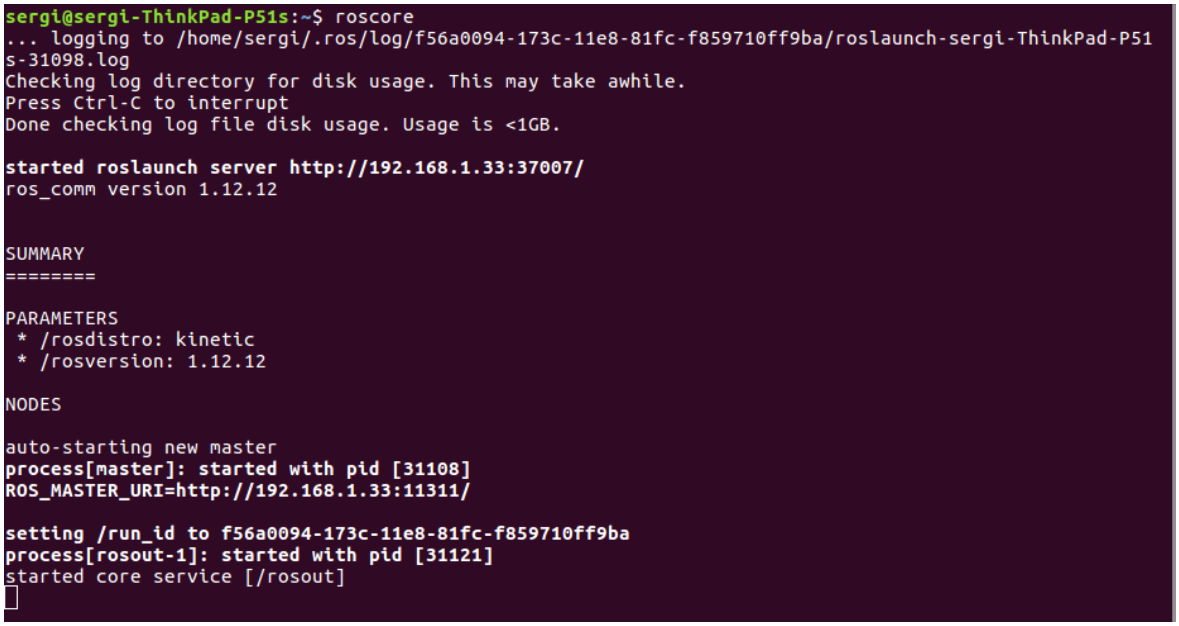


Figura 10.2. Roscore (PC remota)

Font: Elaboració pròpia

Per poder inicialitzar els paquets bàsics per iniciar aplicacions TurtleBot3, aquests comandos s'han d'escriure en el terminal de TurtleBot3 Burger.

$ turtlebot3_bringup turtlebot3_robot.launch

Si es vol fer servir el sensor i el nucli per separat, s'han d'introduir els següents comandaments:

$ roslaunch turtlebot3_bringup turtlebot3_lidar.launch

$ roslaunch turtlebot3_bringup turtlebot3_core.launch

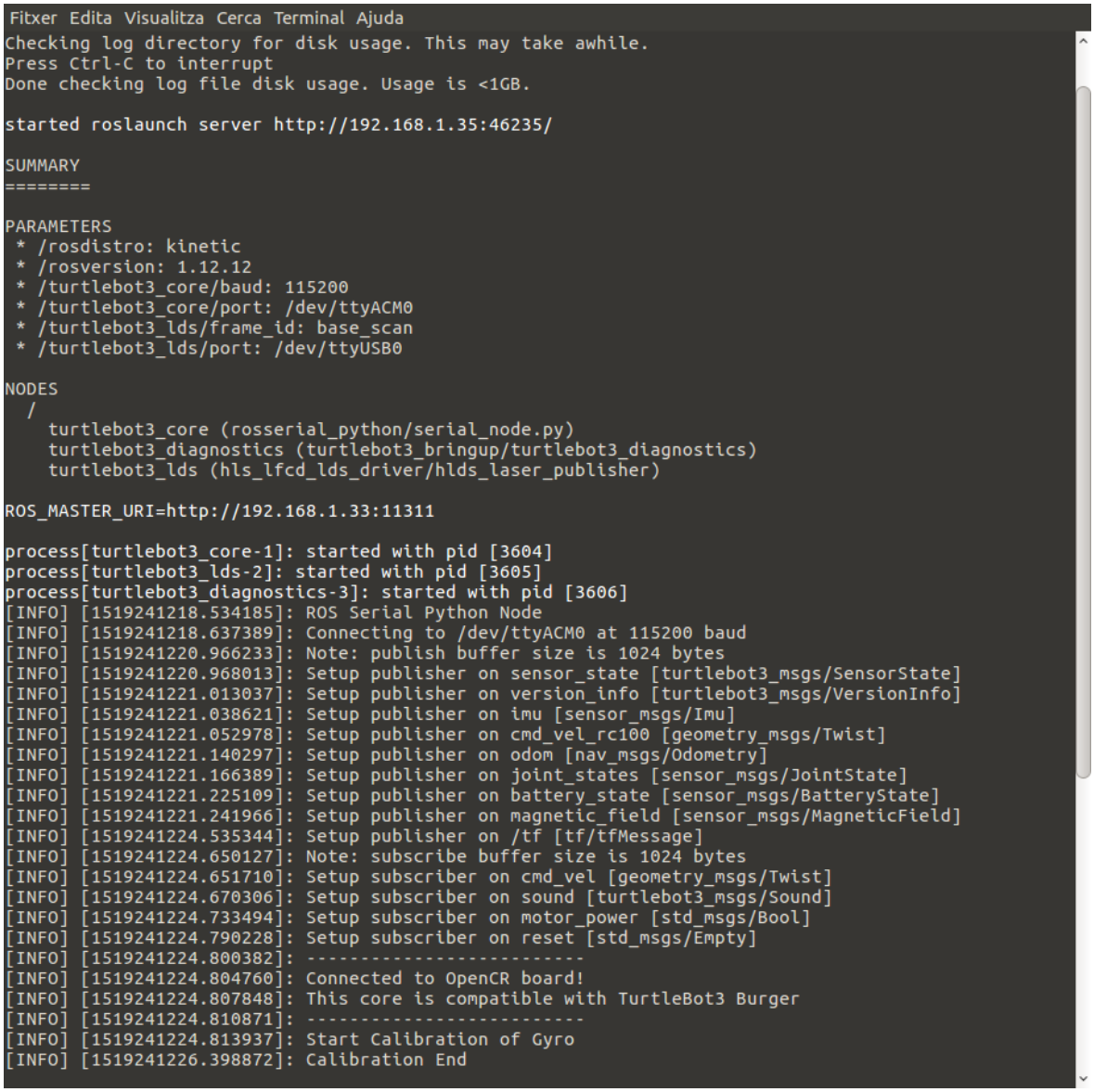


Figura 10.3. Bring up TurtleBot3 Burger

Font: Elaboració pròpia

Per treballar amb el robot s'ha de fer el "bring up", que és la inicialització de l'entorn de treball del TurtleBot3 Burger. Per tant, per poder desenvolupar les aplicacions del robot, s'han d'executar les instruccions anteriors.

Finalment per visualitzar l'entorn de TurtleBot3 Burger en la PC remota, s'ha de fer a través de RViz:

$ export TURTLEBOT3_MODEL=burger

$ roslaunch turtlebot3_bringup turtlebot3_remote.launch

```
sergi@sergi-ThinkPad-P51s:~$ export TURTLEBOT3_MODEL=burger
sergi@sergi-ThinkPad-P51s:~$ roslaunch turtlebot3_bringup turtlebot3_remote.launch
... logging to /home/sergi/.ros/log/f56a0094-173c-11e8-81fc-f859710ff9ba/roslaunch-sergi-ThinkPad-P51
s-31462.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

xacro: Traditional processing is deprecated. Switch to --inorder processing!
To check for compatibility of your document, use option --check-order.
For more infos, see http://wiki.ros.org/xacro#Processing_Order
started roslaunch server http://192.168.1.33:34785/

SUMMARY
========

PARAMETERS
 * /robot_description: <?xml version="1....
 * /robot_state_publisher/publish_frequency: 50.0
 * /rosdistro: kinetic
 * /rosversion: 1.12.12

NODES
  /
    robot_state_publisher (robot_state_publisher/robot_state_publisher)

ROS_MASTER_URI=http://192.168.1.33:11311

process[robot_state_publisher-1]: started with pid [31482]
```

Figura 10.4. Bringup TurtleBot3 Burger (PC remota)

Font: Elaboració pròpia

$ rosrun rviz rviz -d `rospack find turtlebot3_description`/rviz/ model.rviz

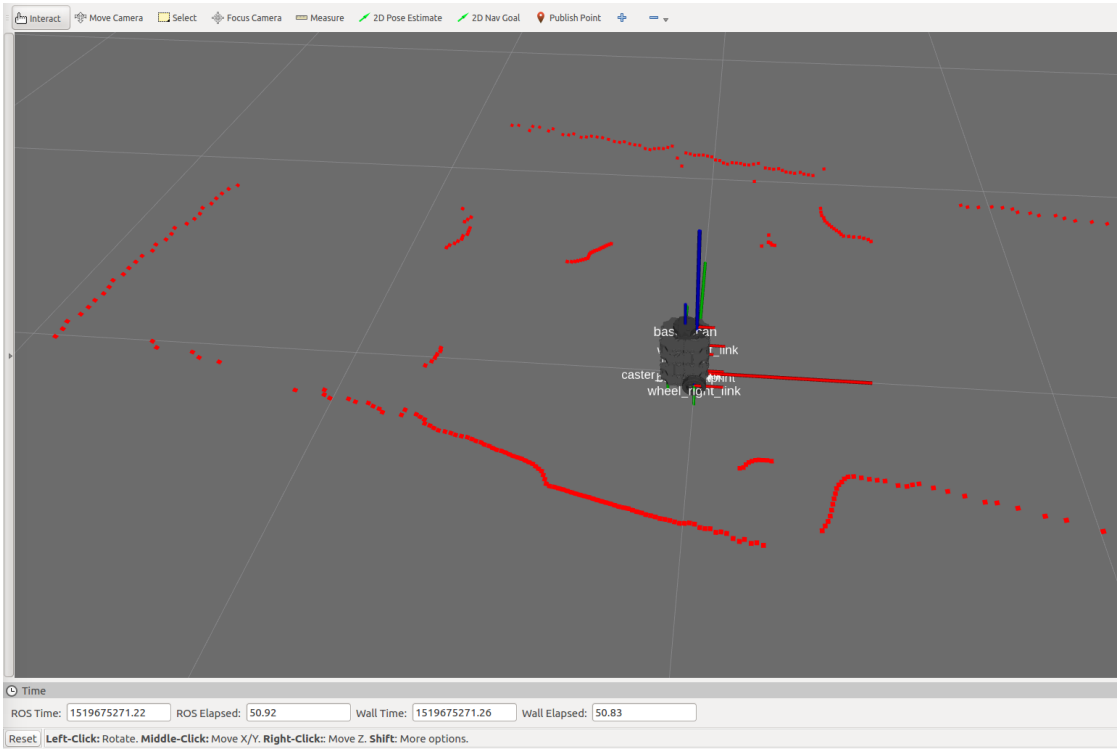


Figura 10.5. Rviz

Font: Elaboració pròpia

11. Aplicacions TurtleBot3 Burger.

En aquest apartat es mostraran diferents exemples que s'han trobat en diferents xarxes socials, i com s'han implementat en el Turtlebot3 Burger.

11.1. Interacció a través del teclat de la PC remota.

Per poder dur a terme una interacció entre TurtleBot3 Burger i la PC remota, una possible aplicació és fer-ho a través del teclat d'aquest últim. Per tant, s'ha d'iniciar el fitxer per fer una prova simple de teleoperació.

$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

Si el programa s'executa amb èxit, apareixerà la següent instrucció a la finestra del terminal.

```
sergi@sergi-ThinkPad-P51s:~$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
... logging to /home/sergi/.ros/log/f56a0094-173c-11e8-81fc-f859710ff9ba/roslaunch-sergi-ThinkPad-P51
s-32200.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.1.33:46789/

SUMMARY
========

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.12

NODES
  /
    turtlebot3_teleop_keyboard (turtlebot3_teleop/turtlebot3_teleop_key)

ROS_MASTER_URI=http://192.168.1.33:11311

process[turtlebot3_teleop_keyboard-1]: started with pid [32217]

Control Your Turtlebot3!
---------------------------
Moving around:
        w
   a    s    d
        x

w/x : increase/decrease linear velocity
a/d : increase/decrease angular velocity
space key, s : force stop

CTRL-C to quit

currently:      linear vel 0.01  angular vel 0
currently:      linear vel 0.02  angular vel 0
currently:      linear vel 0.03  angular vel 0
currently:      linear vel 0.03  angular vel -0.1
currently:      linear vel 0.03  angular vel -0.2
```

Figura 11.1. Teminal aplicació teleoperation key (PC remota)

Font: Elaboració pròpia

Aquesta aplicació és una manera molt simple de verificar que la connectivitat amb el TurtleBot3 Burger és correcte, i així tenir una primera toma de contacte, la qual cosa sempre és d'agrair després d'una instal·lació no del tot simple.

Per tant, com molt ben explicat es troba en el terminal de l'aplicació, per poder moure el TurtleBot3, es fa a través del teclat del PC remota. Les tecles que s'utilitzen per moure el robot són les següents:

W	Accelerar endavant
X	Accelerar enrere
D	Accelerar cap a la dreta
A	Accelerar cap a l'esquerra
S	Aturar el TurtleBot3 Burger
ESPAI	Forçar una parada

Per tant, aquesta aplicació permetrà moure el robot per una superfície coneguda, ja sigui perquè l'usuari pot veure el robot, o perquè a través del sensor pot visualitzar, com s'ha fet a l'apartat anterior, les distàncies que l'envolten.

11.2. Mapejat "SLAM".

La localització i assignació simultània, o SLAM, és una tècnica per dibuixar un mapa calculant la ubicació actual en un espai arbitrari, i movent-se a través d'ell va dibuixant un mapa de la localització on es troba.

El SLAM és una característica coneguda de TurtleBot des dels seus predecessors, a més TurtleBot3 pot dibuixar un mapa amb la seva plataforma compacta i assequible.

Per poder executar el SLAM en el TurtleBot3 Burger, s'ha d'executar les següents instruccions en el terminal de la PC remota.

$ export TURTLEBOT3_MODEL=Burger

$ roslaunch turtlebot3_slam turtlebot3_slam.launch

```
sergi@sergi-ThinkPad-P51s:~$ roslaunch turtlebot3_slam turtlebot3_slam.launch
... logging to /home/sergi/.ros/log/f56a0094-173c-11e8-81fc-f859710ff9ba/roslaunch-sergi-ThinkPad-P51
s-32414.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

xacro: Traditional processing is deprecated. Switch to --inorder processing!
To check for compatibility of your document, use option --check-order.
For more infos, see http://wiki.ros.org/xacro#Processing_Order
started roslaunch server http://192.168.1.33:34135/

SUMMARY
========

PARAMETERS
 * /robot_description: <?xml version="1....
 * /robot_state_publisher/publish_frequency: 50.0
 * /rosdistro: kinetic
 * /rosversion: 1.12.12
 * /turtlebot3_slam_gmapping/angularUpdate: 0.2
 * /turtlebot3_slam_gmapping/astep: 0.05
 * /turtlebot3_slam_gmapping/base_frame: base_footprint
 * /turtlebot3_slam_gmapping/delta: 0.05
 * /turtlebot3_slam_gmapping/iterations: 5
 * /turtlebot3_slam_gmapping/kernelSize: 1
 * /turtlebot3_slam_gmapping/lasamplerange: 0.005
 * /turtlebot3_slam_gmapping/lasamplestep: 0.005
 * /turtlebot3_slam_gmapping/linearUpdate: 0.2
 * /turtlebot3_slam_gmapping/llsamplerange: 0.01
 * /turtlebot3_slam_gmapping/llsamplestep: 0.01
 * /turtlebot3_slam_gmapping/lsigma: 0.075
 * /turtlebot3_slam_gmapping/lskip: 0
 * /turtlebot3_slam_gmapping/lstep: 0.05
 * /turtlebot3_slam_gmapping/map_update_interval: 2.0
 * /turtlebot3_slam_gmapping/maxUrange: 4.0
 * /turtlebot3_slam_gmapping/minimumScore: 100
 * /turtlebot3_slam_gmapping/odom_frame: odom
 * /turtlebot3_slam_gmapping/ogain: 3.0
 * /turtlebot3_slam_gmapping/particles: 120
 * /turtlebot3_slam_gmapping/resampleThreshold: 0.5
 * /turtlebot3_slam_gmapping/sigma: 0.05
 * /turtlebot3_slam_gmapping/srr: 0.01
 * /turtlebot3_slam_gmapping/srt: 0.02
 * /turtlebot3_slam_gmapping/str: 0.01
 * /turtlebot3_slam_gmapping/stt: 0.02
 * /turtlebot3_slam_gmapping/temporalUpdate: 0.5
 * /turtlebot3_slam_gmapping/xmax: 10.0
 * /turtlebot3_slam_gmapping/xmin: -10.0
 * /turtlebot3_slam_gmapping/ymax: 10.0
 * /turtlebot3_slam_gmapping/ymin: -10.0

NODES
  /
    robot_state_publisher (robot_state_publisher/robot_state_publisher)
    turtlebot3_slam_gmapping (gmapping/slam_gmapping)

ROS_MASTER_URI=http://192.168.1.33:11311

process[robot_state_publisher-1]: started with pid [32434]
```

Figura 11.2. Teminal mapejat TurtleBot3 Burger

Font: Elaboració pròpia

La instrucció SLAM consisteix en anar captant punts, i a partir d'ells va fent el mapejat de la zona en què es troba el robot.

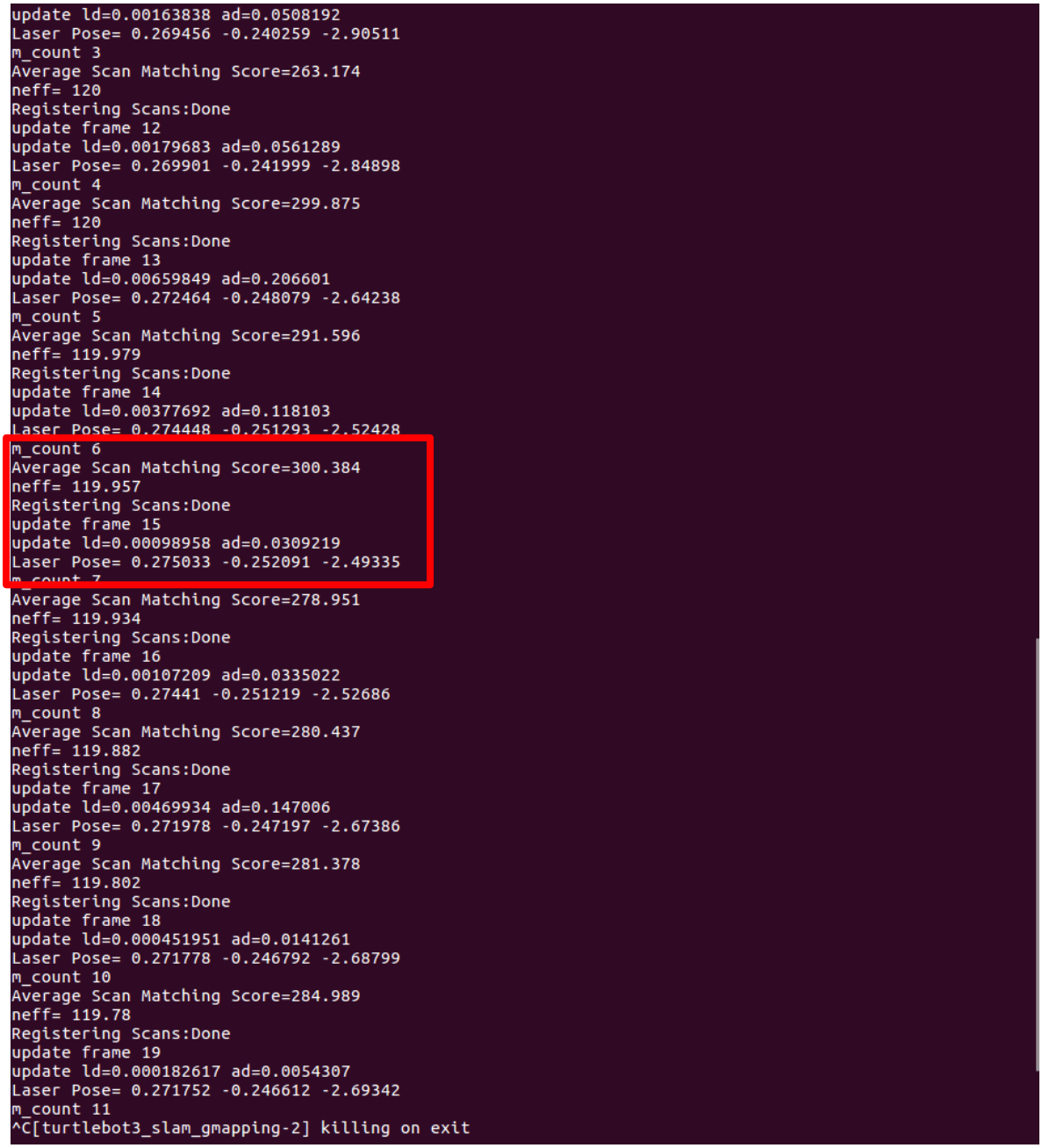


Figura 11.3. Teminal aplicació SLAM TurtleBot3 Burger

Font: Elaboració pròpia

A la figura anterior es pot visualitzar com el programa de SLAM capta aquests punts que s'han comentat anteriorment, amb l'objectiu de després veure'ls reflectits en un mapa de forma visual. Per poder veure el mapejat que està duent a terme TurtleBot3 Burger, s'ha d'executar la següent instrucció en una finestra del terminal nova.

$ rosrun rviz rviz -d `rospack find turtlebot3_slam`/rviz/turtlebot3_slam.rviz

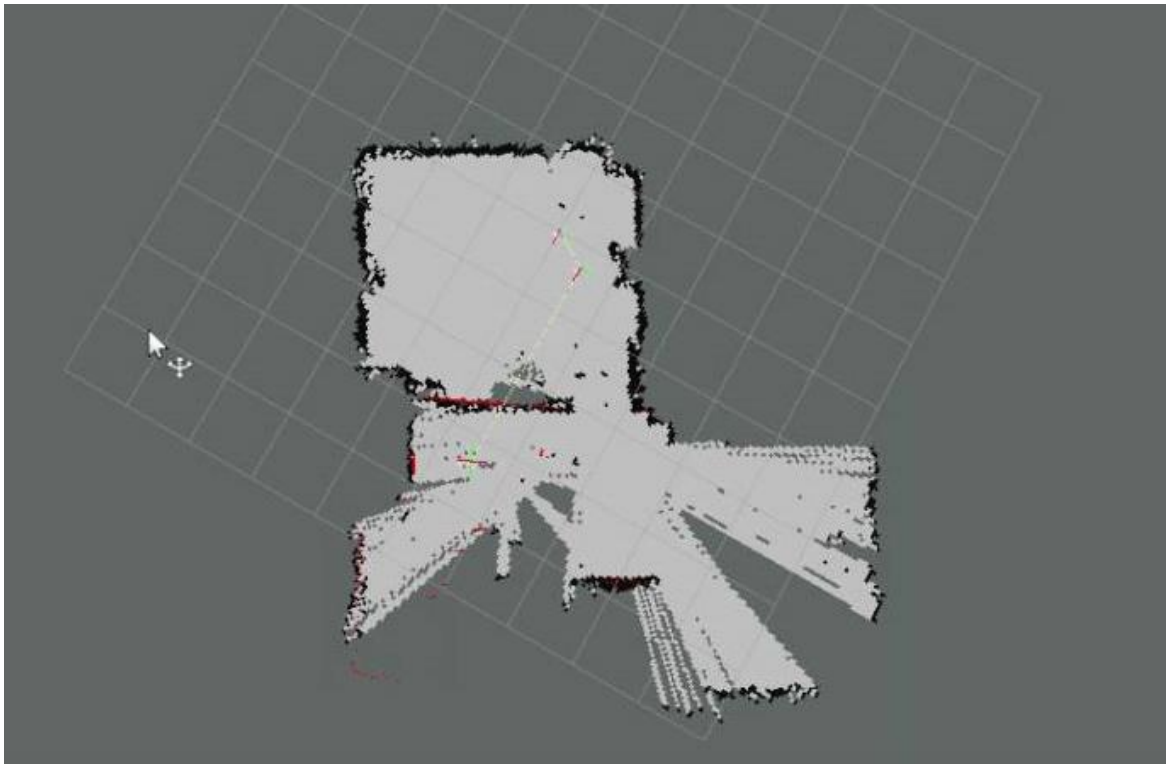


Figura 11.4. Mapejat aula laboratori de robòtica

Font: Elaboració pròpia

Per poder guardar el mapejat que ha fet TurtleBot3 Burger i continuar treballant amb ell posteriorment, s'ha de d'executar la següent instrucció:

$ rosrun map_server map_saver -f ~/map

11.3. Interacció amb el robot amb fletxes.

Turtlebot3 es pot moure amb marcadors interactius a RViz, és a dir, es pot moure a partir de moviments lineals o de gir utilitzant marcadors interactius, en aquest cas, com es pot veure a la figura 11.5, fletxes.

Per dur a terme aquesta aplicació, primerament s'ha de inicialitzar la posada en marxa del robot, com s'ha explicat en el capítol 10, és a dir executant les següent comandes en el terminal:

$ export TURTLEBOT3_MODEL=Burger

$ roslaunch turtlebot3_bringup turtlebot3_remote.launch

Una vegada inicialitzat el robot, s'ha de executar la aplicació que permetrà interaccionar amb el robot a través de les fletxes. Per poder executar-la es fa de la següent manera:

$ roslaunch turtlebot3_example interactive_markers.launch

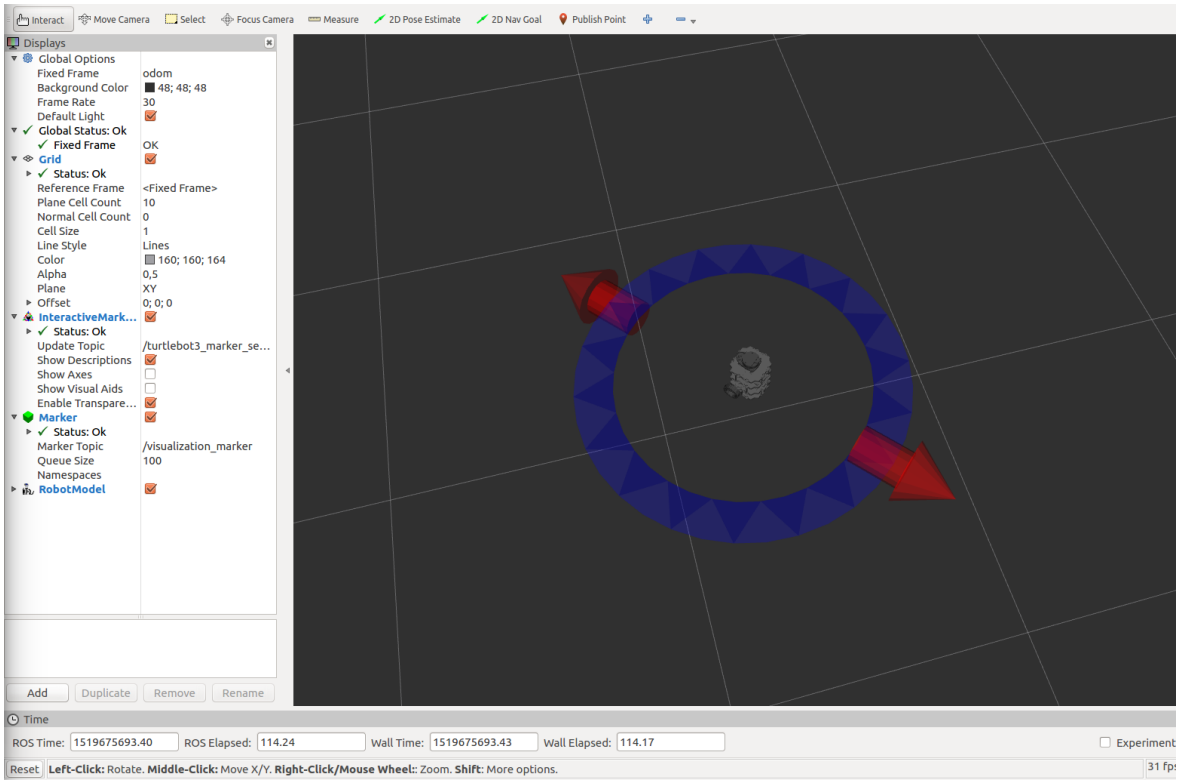


Figura 11.5. Teminal aplicació interacció amb el robot amb fletxes

Font: Elaboració pròpia

Una vegada ja s'ha inicialitzat la aplicació, s'ha d'executar el rviz, el qual és el programa que permetrà interactuar de manera visual amb el robot. Per poder executar rviz es fa de la següent manera:

$ rosrun rviz rviz -d `rospack find turtlebot3_example`/rviz/turtlebot3_interactive.rviz

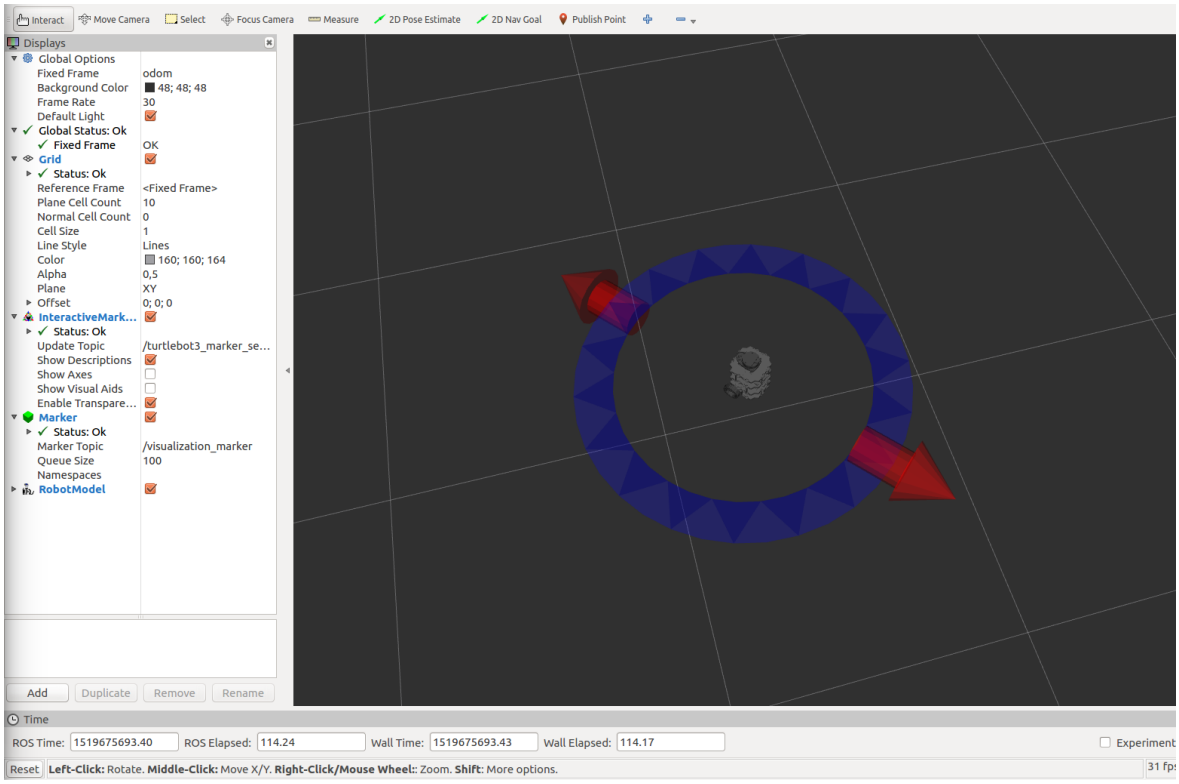


Figura 11.6. Interacció amb el robot amb fletxes

Font: Elaboració pròpia

11.4. Detecció d'obstacles.

Aquesta aplicació permet detectar obstacles a partir de la distància que capta el sensor LIDAR, i plasma aquesta distància en el terminal. A més el programa es troba configurat de tal manera, que quan el robot detecta una distància inferior a 0.3 metres, el robot s'aturi. Aquest programa es pot veure en el annex II.

Per poder dur a terme aquesta aplicació, primerament s'ha de inicialitzar la posada en marxa del robot, com s'ha explicat en el capítol 10.

$ rosrun turtlebot3_example turtlebot3_obstacle.py

```
sergi@master:~$ rosrun turtlebot3_example turtlebot3_obstacle.py
[INFO] [1527969136.422314]: distance of the obstacle : 0.412000
[INFO] [1527969136.818935]: distance of the obstacle : 0.411000
[INFO] [1527969137.018203]: distance of the obstacle : 0.409000
[INFO] [1527969137.424639]: distance of the obstacle : 0.416000
[INFO] [1527969137.634422]: distance of the obstacle : 0.414000
[INFO] [1527969138.029493]: distance of the obstacle : 0.418000
[INFO] [1527969138.431539]: distance of the obstacle : 0.408000
[INFO] [1527969138.827775]: distance of the obstacle : 0.400000
[INFO] [1527969139.049409]: distance of the obstacle : 0.396000
[INFO] [1527969139.251937]: distance of the obstacle : 0.392000
[INFO] [1527969139.627605]: distance of the obstacle : 0.385000
[INFO] [1527969139.848499]: distance of the obstacle : 0.380000
[INFO] [1527969140.235655]: distance of the obstacle : 0.372000
[INFO] [1527969140.646210]: distance of the obstacle : 0.363000
[INFO] [1527969141.040652]: distance of the obstacle : 0.355000
[INFO] [1527969141.271749]: distance of the obstacle : 0.351000
[INFO] [1527969141.648717]: distance of the obstacle : 0.342000
[INFO] [1527969142.039784]: distance of the obstacle : 0.334000
[INFO] [1527969142.269349]: distance of the obstacle : 0.329000
[INFO] [1527969142.646504]: distance of the obstacle : 0.321000
[INFO] [1527969143.049855]: distance of the obstacle : 0.313000
[INFO] [1527969143.453370]: distance of the obstacle : 0.304000
[INFO] [1527969143.854997]: Stop!
[INFO] [1527969144.247702]: Stop!
[INFO] [1527969144.650796]: Stop!
[INFO] [1527969145.052848]: Stop!
[INFO] [1527969145.284518]: Stop!
[INFO] [1527969145.660598]: Stop!
[INFO] [1527969146.064285]: Stop!
[INFO] [1527969146.458056]: Stop!
[INFO] [1527969146.702685]: Stop!
[INFO] [1527969147.077187]: Stop!
```

Figura 11.7. Teminal aplicació detector obstacles

Font: Elaboració pròpia

En la figura anterior es pot visualitzar les distàncies que va captant TurtleBot3 Burger a mesura que aquest es va movent cap a la paret. Quan aquest detecta que hi ha un obstacle a menys de 0,3 metres, aquest es para.

Per tant, aquesta aplicació es podria utilitzar per evitar que topés amb algun obstacle, reaccionant abans de que arribés, o esperant-se, fins que aquell obstacles es mogués.

11.5. Operació moviment per coordenades.

El TurtleBot3 es pot moure amb 2D punt (x, y) i una z-angular. Per exemple, si s'insereix en el terminal (0.5, 0.3, 60), TurtleBot3 es mou al punt (x = 0.5m, y = 0.3m) i gira 60º respecte el seu punt inicial.

Per poder dur a terme aquesta aplicació, primerament s'ha de inicialitzar la posada en marxa del robot, com s'ha explicat en el capítol 10.

Una vegada inicialitzada la posada en marxa del Turtlebot, per executar el programa de moure el robot a través de coordenades, s'ha d'introduir la següent instrucció en el terminal.

$ roslaunch turtlebot3_example turtlebot3_pointop_key.launch

```
sergi@sergi-ThinkPad-P51s:~$ roslaunch turtlebot3_pointop turtlebot3_pointop_key.launch
... logging to /home/sergi/.ros/log/f56a0094-173c-11e8-81fc-f859710ff9ba/roslaunch-sergi-ThinkPad-P51
s-8177.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.1.33:39089/

SUMMARY
========

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.12

NODES
  /
    turtlebot3_pointop_keyboard (turtlebot3_pointop/turtlebot3_pointop_key.py)

ROS_MASTER_URI=http://192.168.1.33:11311

process[turtlebot3_pointop_keyboard-1]: started with pid [8194]

control your Turtlebot3!
-----------------------
Insert xyz - coordinate.
x : position x (m)
y : position y (m)
z : orientation z (degree: -180 ~ 180)
If you want to close, insert 's'
-----------------------

| x | y | z |
0.4 0.2 90
```

Figura 11.8. Teminal aplicació moviment per coordenades

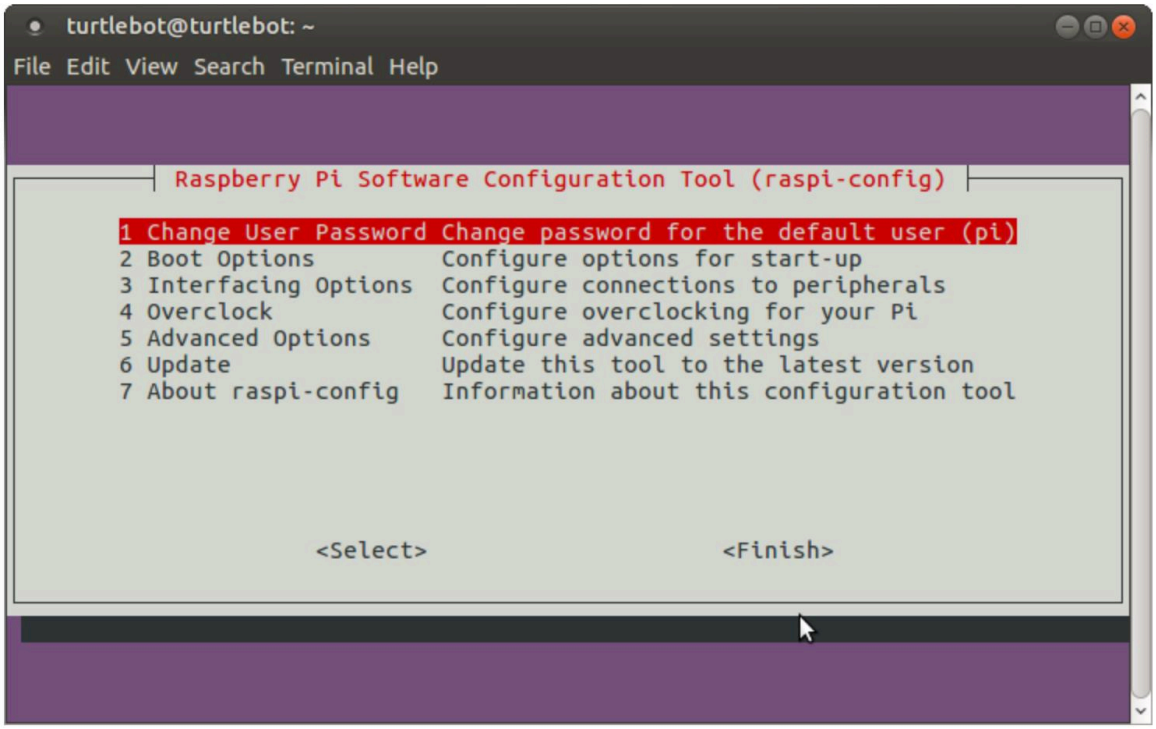
Font: Elaboració pròpia

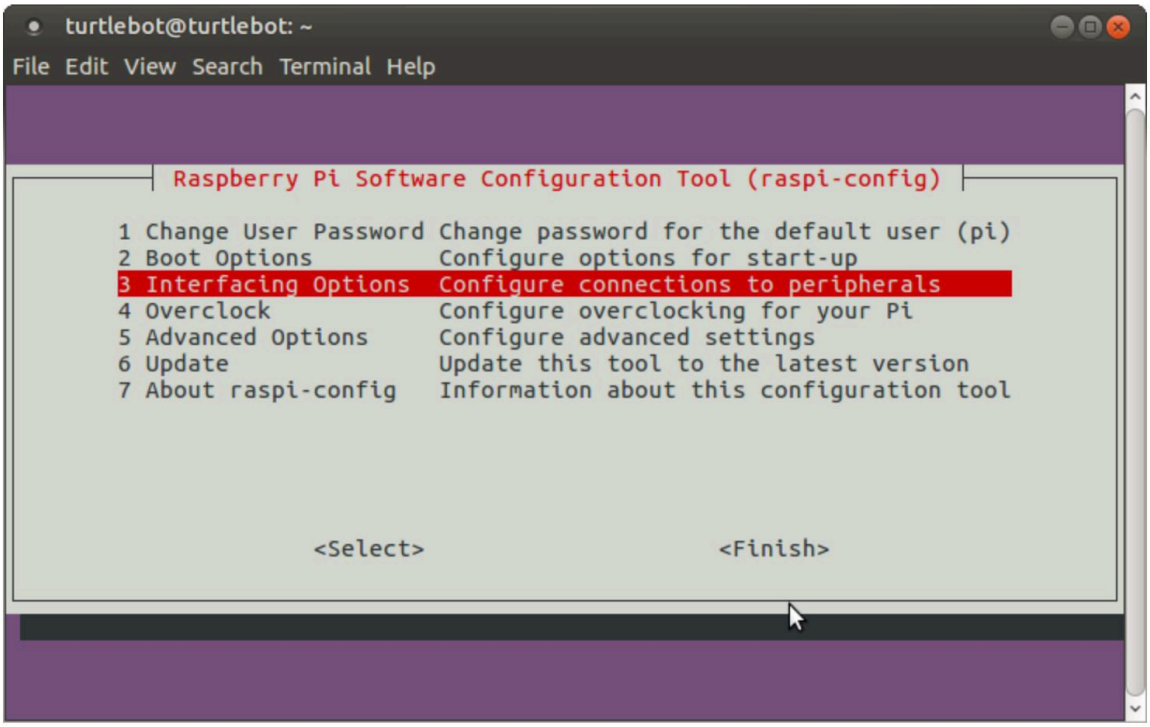
11.6. Aplicació càmera module V2.

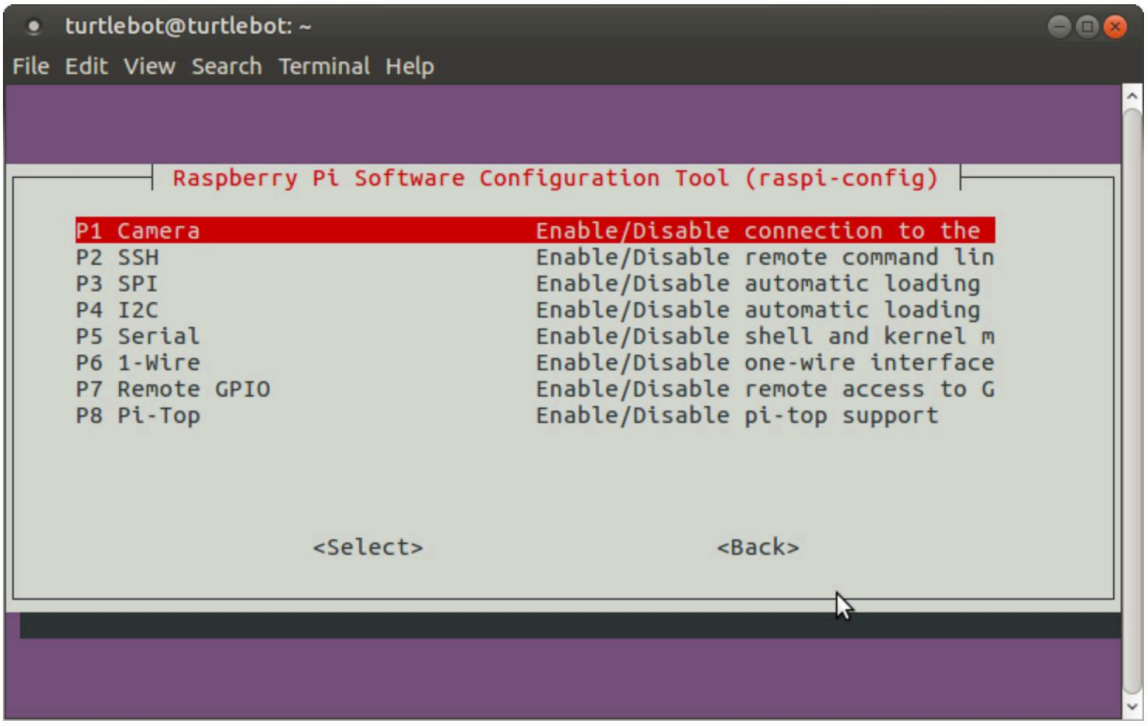
El mòdul de la càmera v2 té un sensor Sony IMX219 de 8 megapíxels. El mòdul de la càmera es pot utilitzar per gravar vídeos d'alta definició, així com fotografies de fotogrames Es poden fer ús de biblioteques que empaquetem amb la càmera per crear efectes.

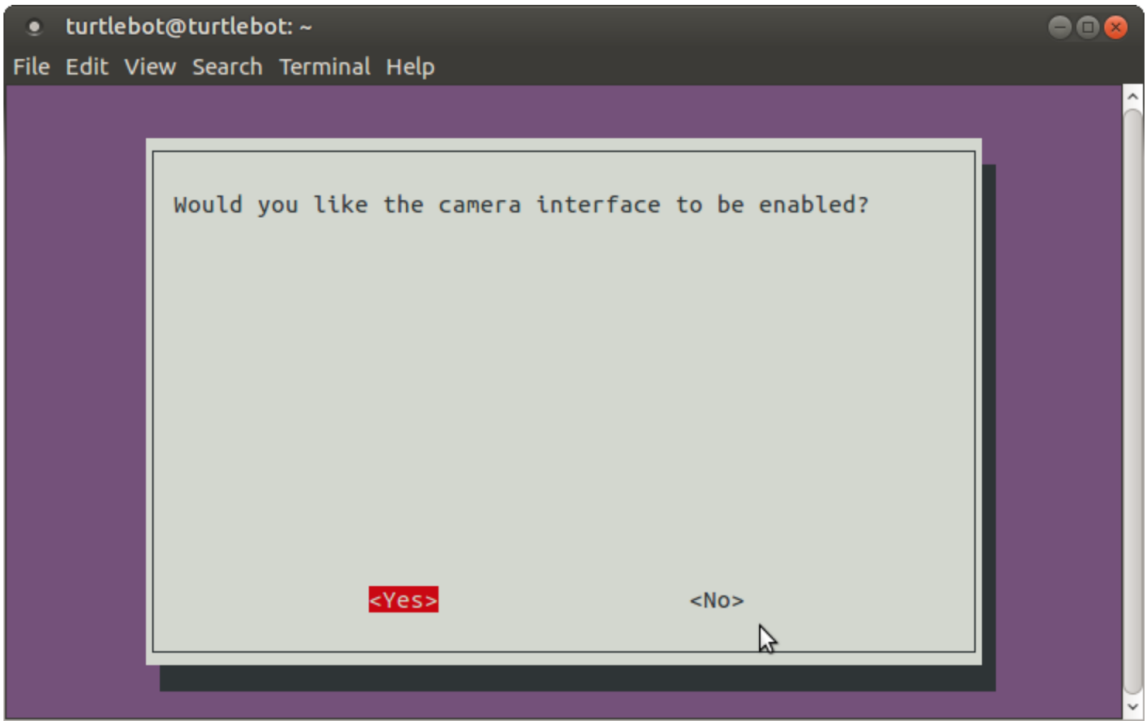
A continuació es procedirà a la instal·lació de la càmera tant en el TurtleBot 3 Burger, com en la PC remota, per tal de poder interactuar amb ella.

Primerament s'ha de connectar la càmera module V2 en la Raspberry Pi, posteriorment s'haurà d'activar el paràmetre de la càmera en la RaspberryPi, és a dir en el TurtleBot 3 Burger. Per poder activar-los s'han de seguir els següents passos.






turtlebot@turtlebot: ~
File Edit View Search Terminal Help
Raspberry Pi Software Configuration Tool (raspi-config)
P1 Camera Enable/Disable connection to the
P2 SSH Enable/Disable remote command lin
P3 SPI Enable/Disable automatic loading
P4 I2C Enable/Disable automatic loading
P5 Serial Enable/Disable shell and kernel m
P6 1-Wire Enable/Disable one-wire interface
P7 Remote GPIO Enable/Disable remote access to G
P8 Pi-Top Enable/Disable pi-top support
<Select>
<Back>


turtlebot@turtlebot: ~
File Edit View Search Terminal Help
Would you like the camera interface to be enabled?
<Yes>
<No>

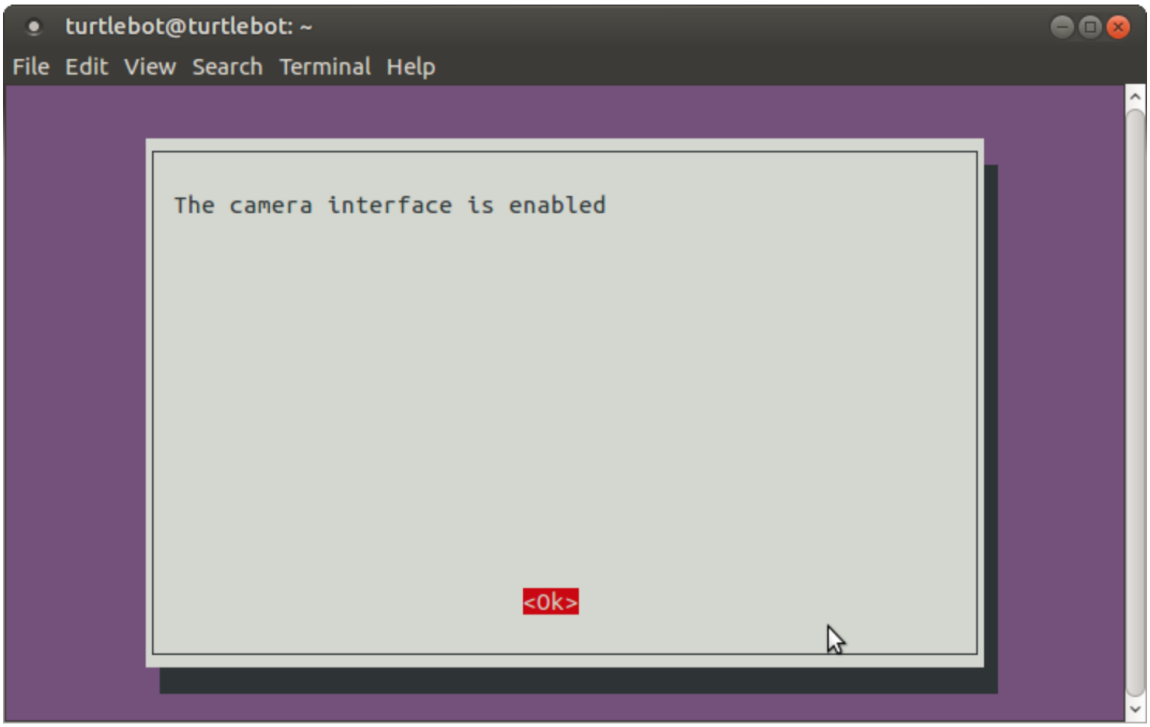


Figura 11.9. Activa la interfície de la càmera

Font: Elaboració pròpia

Una vegada s'ha activat la càmera en el TurtleBot3 Burger, és important fer un reboot, que consisteix en reinicia el robot.

Per comprovar si aquesta interfície de la càmera s'ha instal·lat i funciona, es recomana provar en el terminal la següent instrucció, la qual permetrà veure la càmera durant cinc segons i a continuació farà una fotografia i la guardarà com a *test,jpeg*

$ raspistill -v -o test.jpg

Una altre manera de fer una comprovació és compilar un petit programa des de TurtleBot3 Burger, amb l'objectiu de comprovar si el procediment anterior de configuració de la càmera s'ha instal·lat i funciona correctament.

L'aplicació que ha de desenvolupar aquest programa és molt simple, primerament s'inicialitza la càmera i funcionarà durant 20 segons, de tal manera que es podrà veure en el TurteBot3 Burger, i una vegada passat aquest temps la càmera es desconnectarà. Per poder fer el aquest programa, s'ha de crear un arxiu amb extensió *.py* i introduir aquest codi que es troba a continuació:

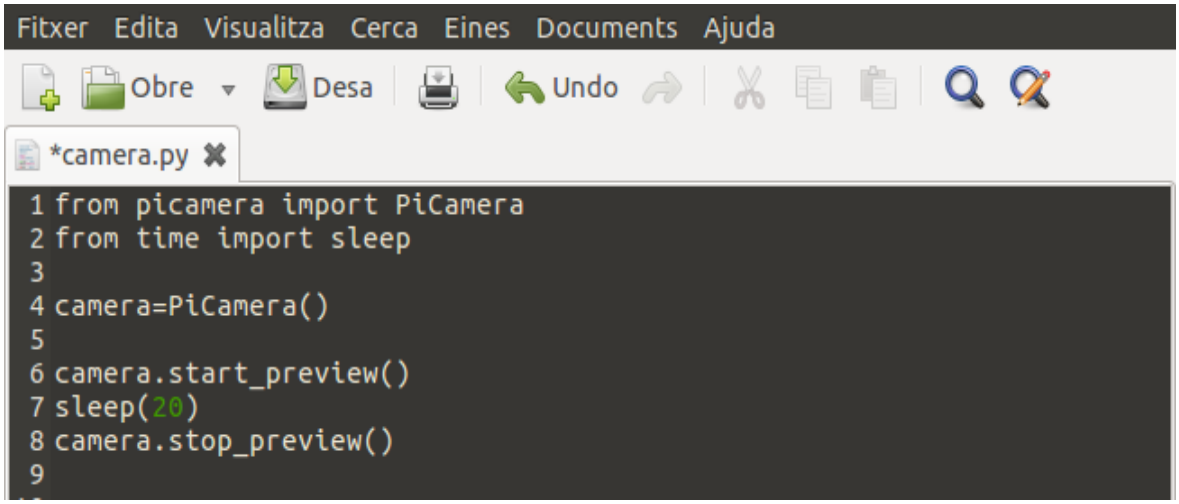


Figura 11.10. Codi visualització càmera

Font: Elaboració pròpia

A continuació, comprovat que la càmera s'ha instal·lat sense cap problema, s'ha de procedir a la instal·lació dels paquets de la càmera dins del sistema de ROS. Les comandes següents instal·laran els paquets rellevants de Raspberry Pi Camera en el sistema ja instal·lat de ROS de TurtleBot3 Burger.

$ cd ~/catkin_ws/src

$ git clone https://github.com/UbiquityRobotics/raspicam_node.git

$ sudo apt-get install ros-kinetic-compressed-image-transport ros-kinetic-camera-info-manager

$ cd ~/catkin_ws && catkin_make

I finalment una vegada ja instal·lats tots els paquets necessaris en el entorn de ROS, ja es pot activar el node raspicam en el TurtleBot3 Burger, el qual és el que permet fer la visualització des de la PC remota.

$ roslaunch raspicam_node camerav2_410x308_30fps.launch

Arribat aquest punt, ja s'ha creat el node necessari per visualitzar la càmera des de la PC remota i és per això que s'ha de introduir la següent instrucció en el terminal de la PC remota.

$ rqt_image_view

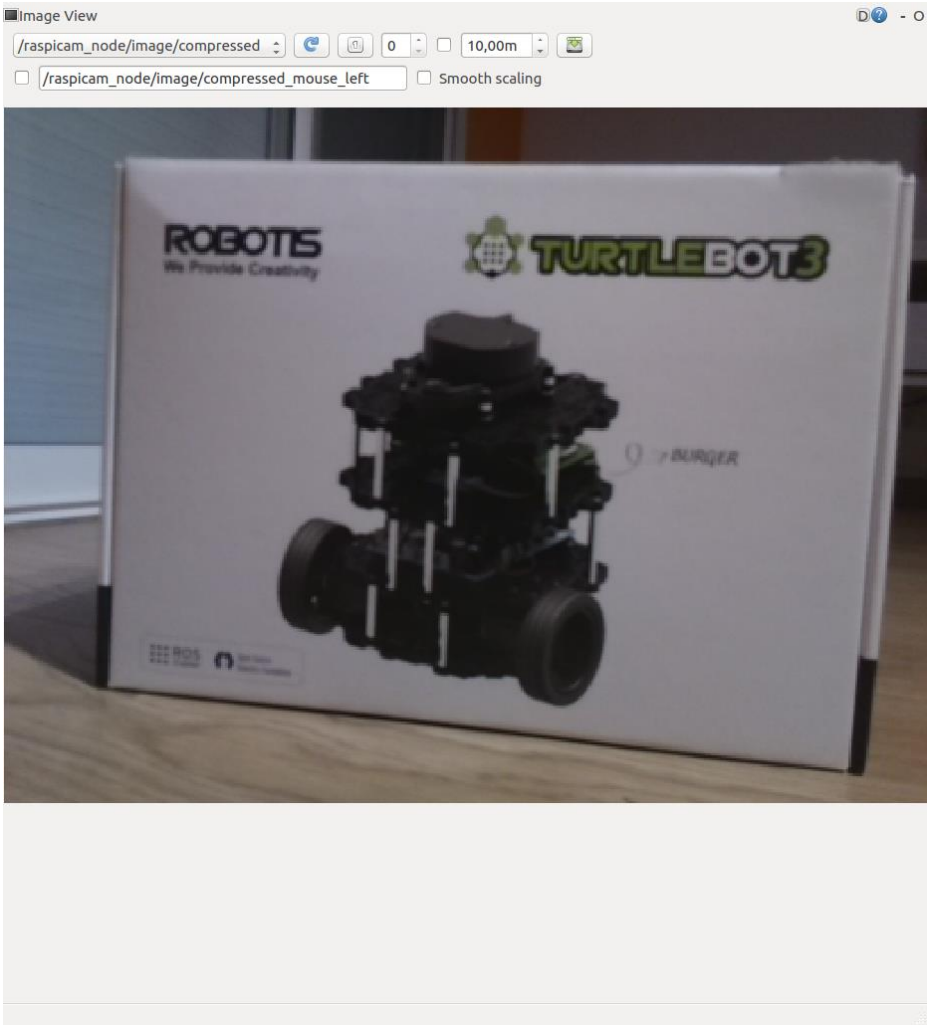


Figura 11.11. Rqt_image_view

Font: Elaboració pròpia

12. Aplicació TurtleBot3 Burger.

Finalment l'aplicació que s'ha decidit abordar, consisteix a partir d'algunes aplicacions esmentades en el capítol 11, aconseguir fer-ne una única aplicació.

És per això que s'ha decidit moure el robot a partir del teclat de la PC remota veien en el rviz la lectura del sensor LIDAR, i a més visualitzar la càmera, d'aquesta manera es podrà moure el robot per una zona desconeguda, ja que mitjançant la càmera frontal i la lectura del sensor LIDAR, es podrà moure el robot per conèixer la zona i sense topar amb cap obstacle.

Per poder desenvolupar aquesta aplicació s'han de seguir els següents passos:

Primerament s'ha d'inicialitzar el roscore en execució per tal que els nodes ROS es comuniquin, aquest comando s'ha d'escriure en el terminal de la PC remota, sobretot no s'ha d'executar sobre TurtleBot3 Burger.

```
sergi@sergi-ThinkPad-P51s:~$ roscore
... logging to /home/sergi/.ros/log/f56a0094-173c-11e8-81fc-f859710ff9ba/roslaunch-sergi-ThinkPad-P51
s-31098.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.1.33:37007/
ros_comm version 1.12.12


SUMMARY
========

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.12

NODES

auto-starting new master
process[master]: started with pid [31108]
ROS_MASTER_URI=http://192.168.1.33:11311/

setting /run_id to f56a0094-173c-11e8-81fc-f859710ff9ba
process[rosout-1]: started with pid [31121]
started core service [/rosout]
```

Figura 12.1. Roscore (PC remota)

Font: Elaboració pròpia

Una vegada els nodes de ROS es troben activats, a partir de el comando SSH s'inicialitzaran els paquets bàsics per iniciar aplicacions TurtleBot3, aquests comandos s'han d'escriure en el terminal de TurtleBot3 Burger.

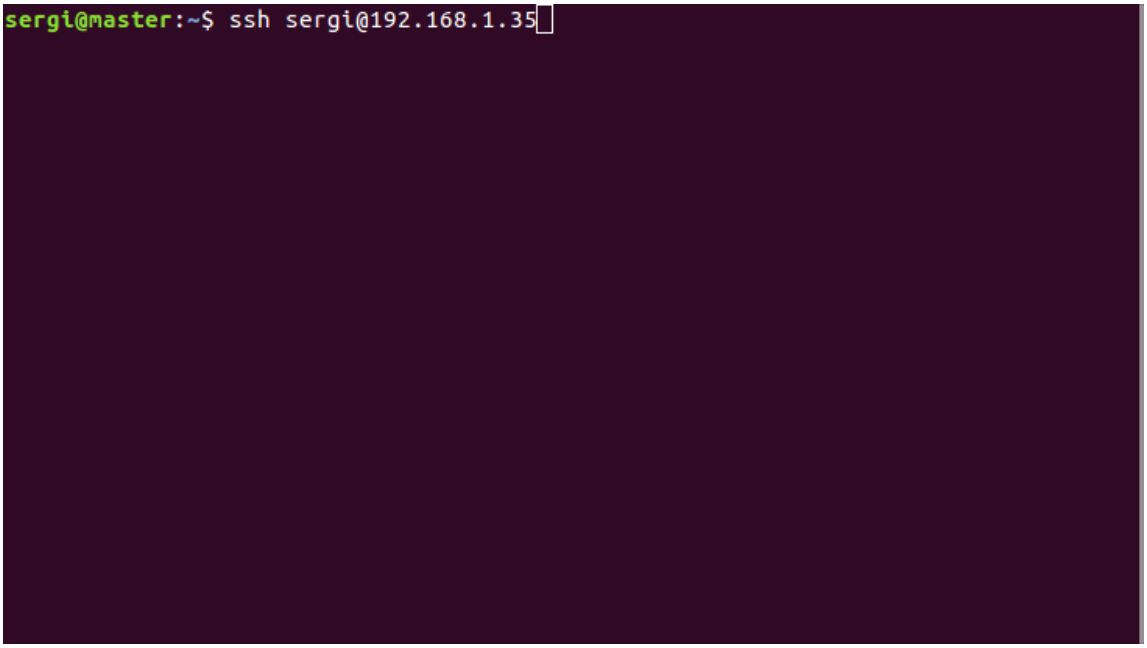


Figura 12.2. Terminal SSH a la PC remota

Font: Elaboració pròpia

Una vegada s'ha connectat amb el TurtleBot3 Burger ha partir del SSH, s'ha d'introduir la següent instrucció per inicialitzar els paquets bàsics del TurtleBot3 Burger.

$ turtlebot3_bringup turtlebot3_robot.launch

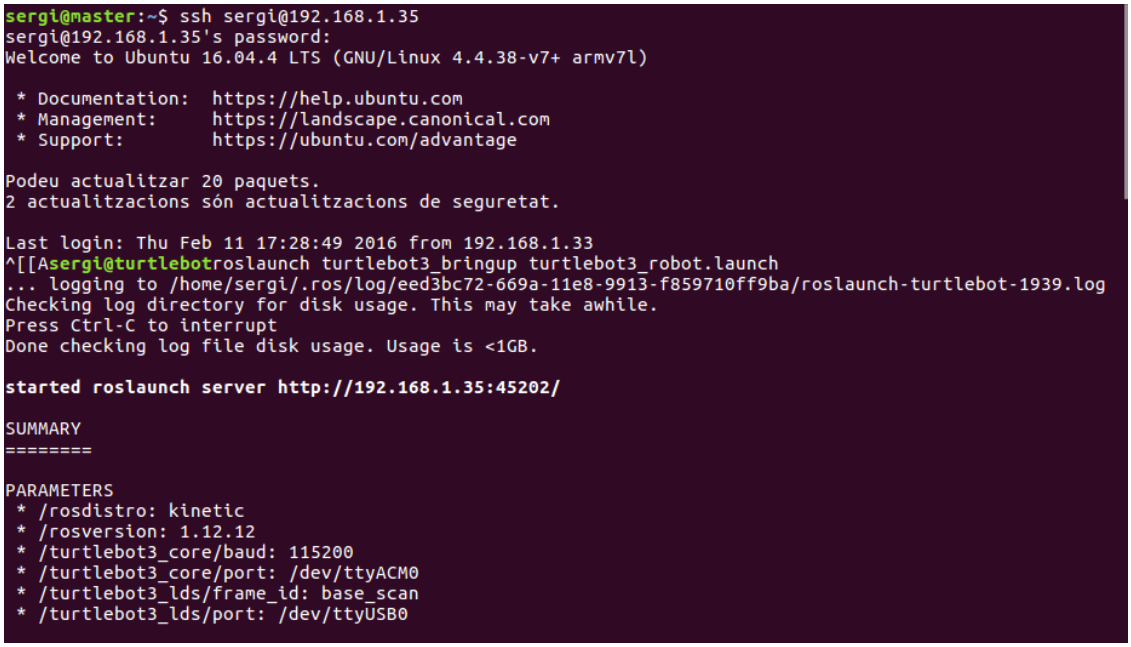


Figura 12.3. Bring up SSH TurtleBot3 Burger

Font: Elaboració pròpia

Una vegada inicialitzat el entorn del robot, en un terminal diferent s'ha de inicialitzar el node de la càmera. Per inicialitzar aquest node s'ha de fer a través del TurtleBot3 Burger, és per això que es tornarà a fer ús del SSH per tal d'introduir la següent instrucció:

$ roslaunch raspicam_node camerav2_1280x960.launch

```
sergi@master:~$ ssh sergi@192.168.1.35
sergi@192.168.1.35's password:
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.38-v7+ armv7l)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Podeu actualitzar 20 paquets.
2 actualitzacions són actualitzacions de seguretat.

Last login: Thu Feb 11 17:28:27 2016 from 192.168.1.33
sergi@turtlebot:~$ roslaunch raspicam_node camerav2_410x308_30fps.launch
... logging to /home/sergi/.ros/log/eed3bc72-669a-11e8-9913-f859710ff9ba/roslaun
ch-turtlebot-1966.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.1.35:38493/

SUMMARY
========

PARAMETERS
 * /raspicam_node/camera_frame_id: raspicam
 * /raspicam_node/camera_info_url: package://raspica...
 * /raspicam_node/exposure_mode: auto
 * /raspicam_node/framerate: 20
 * /raspicam_node/height: 308
 * /raspicam_node/shutter_speed: 0
 * /raspicam_node/width: 410
 * /rosdistro: kinetic
 * /rosversion: 1.12.12

NODES
  /
    raspicam_node (raspicam_node/raspicam_node)

ROS_MASTER_URI=http://192.168.1.33:11311

process[raspicam_node-1]: started with pid [1982]
[ INFO] [1455208224.669439667]: Loading CameraInfo from package://raspicam_node/
camera_info/camerav2_410x308.yaml
[ INFO] [1455208225.193152520]: Camera component done

[ INFO] [1455208225.200391565]: Encoder component done

[ INFO] [1455208225.221772243]: camera calibration URL: package://raspicam_node/
camera_info/camerav2_410x308.yaml
[ INFO] [1455208225.307406724]: Camera successfully calibrated
[ INFO] [1455208226.180424570]: Reconfigure Request: contrast 0, sharpness 0, br
ightness 50, saturation 0, ISO 400, exposureCompensation 0, videoStabilisation 0
, vFlip 0, hFlip 0, zoom 1.00, exposure_mode auto, awb_mode auto
[ INFO] [1455208226.183030293]: Reconfigure done
[ INFO] [1455208226.431394070]: Starting video capture (410, 308, 80, 20)

[ INFO] [1455208226.432791670]: Video capture started
```

Figura 12.4. Raspicam node SSH TurtleBot3 Burger

Font: Elaboració pròpia

Una vegada inicialitzat els nodes principals de connectivitat entre el robot i la PC remota, es procedeix a la visualització de l'entorn de TurtleBot3 Burger en la PC remota.

$ export TURTLEBOT3_MODEL=Burger

$ roslaunch turtlebot3_bringup turtlebot3_remote.launch

```
sergi@sergi-ThinkPad-P51s:~$ export TURTLEBOT3_MODEL=burger
sergi@sergi-ThinkPad-P51s:~$ roslaunch turtlebot3_bringup turtlebot3_remote.launch
... logging to /home/sergi/.ros/log/f56a0094-173c-11e8-81fc-f859710ff9ba/roslaunch-sergi-ThinkPad-P51
s-31462.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

xacro: Traditional processing is deprecated. Switch to --inorder processing!
To check for compatibility of your document, use option --check-order.
For more infos, see http://wiki.ros.org/xacro#Processing_Order
started roslaunch server http://192.168.1.33:34785/

SUMMARY
========

PARAMETERS
 * /robot_description: <?xml version="1....
 * /robot_state_publisher/publish_frequency: 50.0
 * /rosdistro: kinetic
 * /rosversion: 1.12.12

NODES
  /
    robot_state_publisher (robot_state_publisher/robot_state_publisher)

ROS_MASTER_URI=http://192.168.1.33:11311

process[robot_state_publisher-1]: started with pid [31482]
```

Figura 12.5. Bringup TurtleBot3 Burger (Pc remota)

Font: Elaboració pròpia

Una vegada està inicialitzat l'entorn per treballar amb el robot, s'ha d'executar l'aplicació, veure codi a l'annex III . Per poder-la executar, s'ha d'anar a la carpeta en la qual es troba l'arxiu App.py a través del terminal a partir de comandos [annex IV] i executar l'arxiu.

$ python App.py

Aquesta aplicació el que permetrà és moure el robot a través del teclat de la PC remota, de tal manera que aquest es podrà moure a través de les següents tecles:

W	Accelerar endavant
S	Accelerar enrere
D	Accelerar cap a la dreta
A	Accelerar cap a l'esquerra
ESPAI	Aturar el TurtleBot3 Burger

Una vegada executada la aplicació, faltaria poder visualitzar per quin entorn es mou el TurtleBot3 Burger, de tal manera que aquest es podria moure per una superfície desconeguda, però a partir del sensor LIDAR i la visualització de la càmera aquesta superfície podria arribar a ser coneguda.

Per tant per executar la visualització del sensor LIDAR, s'ha de fer a través de RViz, com s'ha comentat en capítols anteriors.

$ rosrun rviz rviz -d `rospack find turtlebot3_description`/rviz/ model.rviz

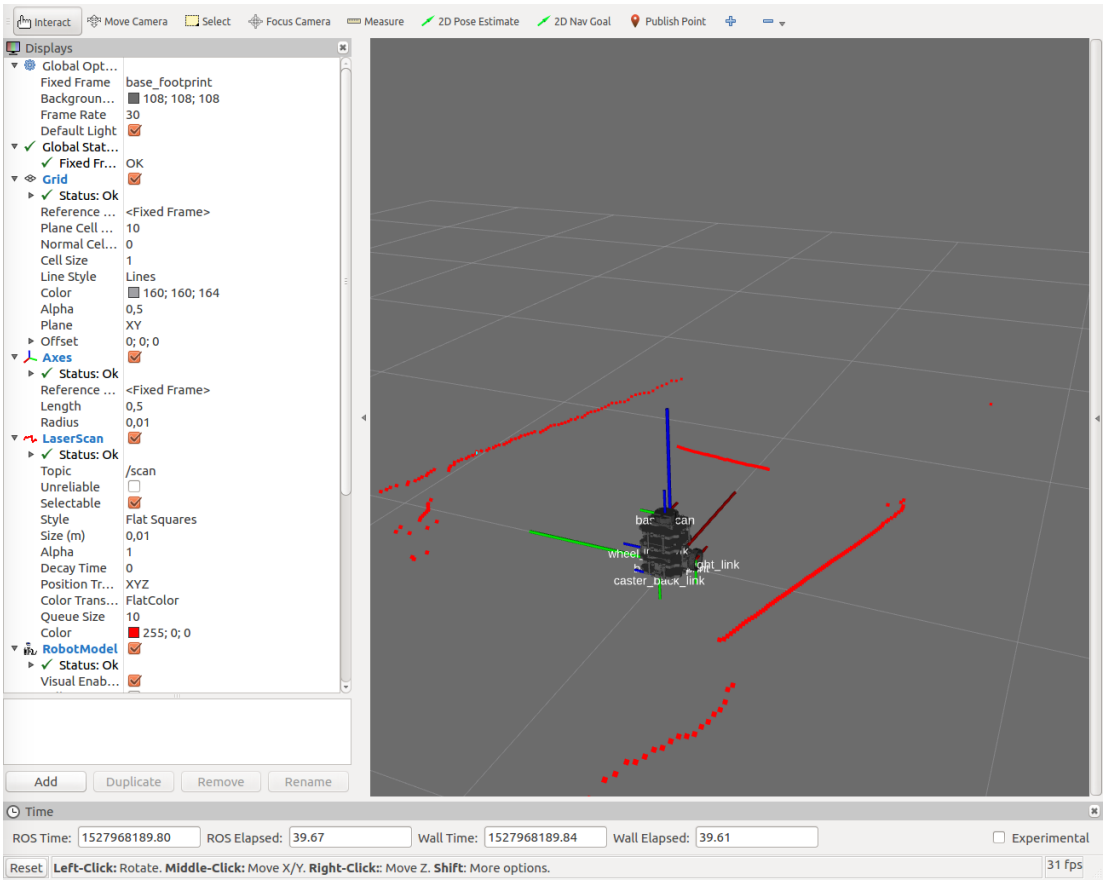


Figura 12.6. Visualització sensor LIDAR a traves de RViz a la PC remota

Font: Elaboració pròpia

Una vegada es visualitza la captació de dades del sensor LIDAR, el que s'ha fet és visualitzar el node de la càmera, que s'havia activat en el TurtleBot3 Burger, en la mateixa interfície gràfica del RViz.

Per poder activar el node de la càmera, com es pot observa a la figura 12.6 a la part esquerra de la imatge hi ha una finestra on es mostren els paràmetres del robot que s'han

inicialitzat, s'haurà de afegir a través de botó *Add* i allà seleccionar, com es pot visualitzar a la imatge 12.7, el node de la càmera que s'ha activar anteriorment.

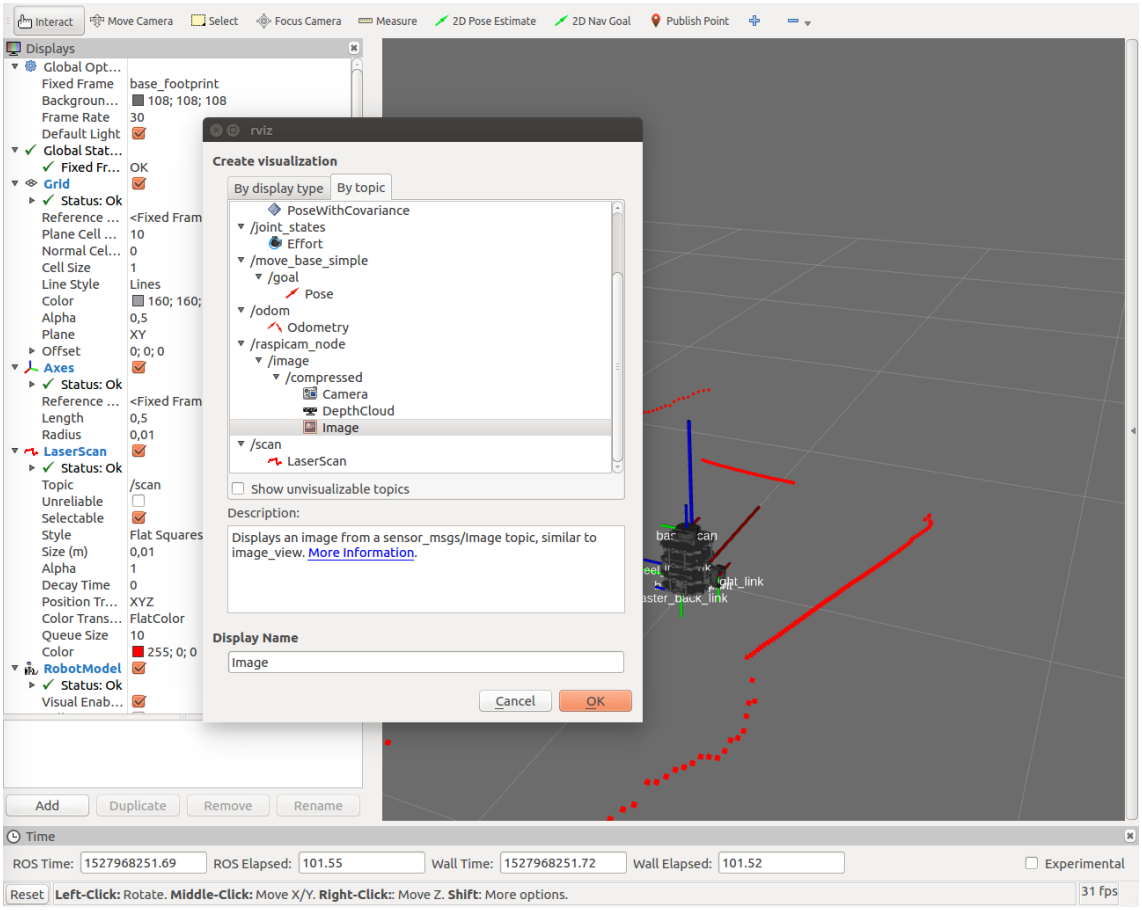


Figura 12.7. Afegir node raspicam RViz

Font: Elaboració pròpia

I finalment es pot observar la càmera i la lectura dels sensor LIDAR.

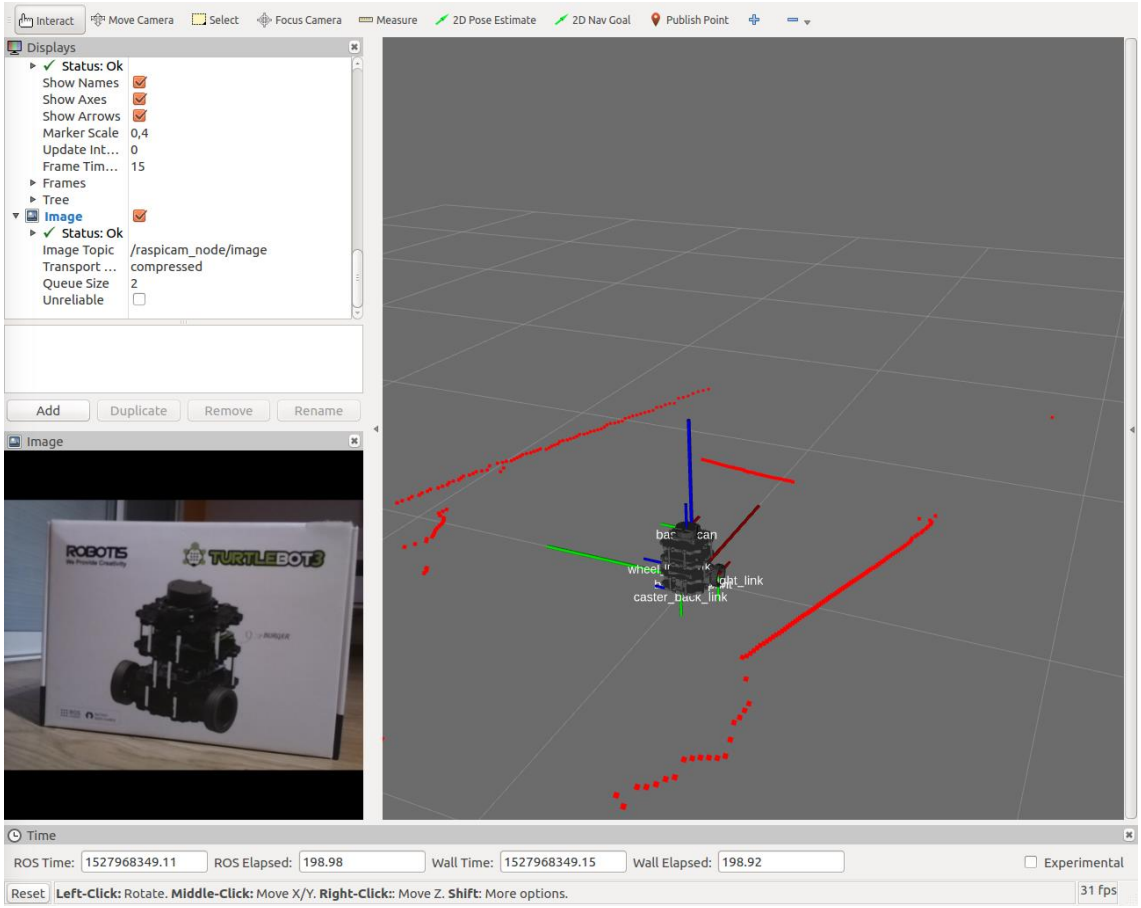


Figura 12.8. Visualització sensor LIDAR i visualització raspicam a través de RViz

Font: Elaboració pròpia

En conclusió, a través de l'aplicació executada anteriorment, el TurtleBot3 Burger es pot moure per una superfície desconeguda, però a través de la càmera es pot saber per on es mou, i a través del sensor LIDAR quins són els obstacles que es troben al seu voltant.

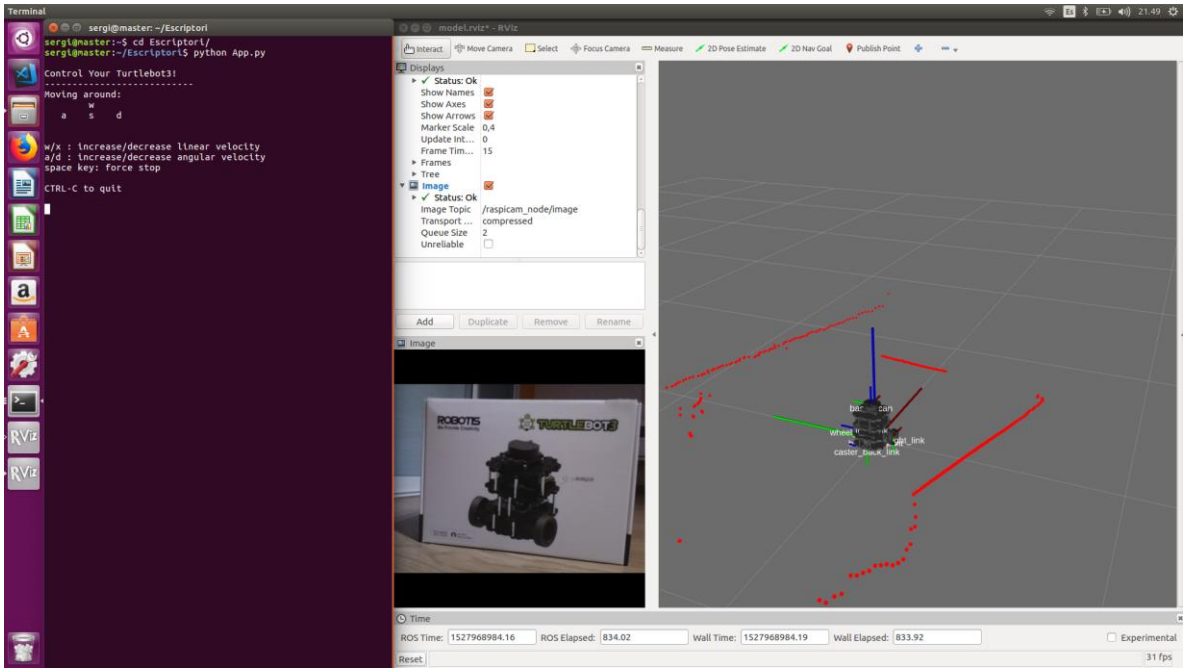


Figura 12.9. Captura de pantalla de l'espai de treball

Font: Elaboració pròpia

13. Conclusions.

13.1. Resum del projecte

Per tal d'obtenir una visió global del robot mòbil TurtleBot3 Burger, s'ha realitzat una recerca d'antecedents i d'informació necessària, amb dos objectius totalment diferents, el primer de tots per conèixer quin robot era amb el que s'anava a treballar i conèixer algunes de les seves especificacions tècniques, i l'altre per profunditzar més en el robot i saber quin és el seu entorn de programació, en aquest cas ROS, i conèixer com funcionava aquest sistema operatiu de Linux. Després de definir els objectius del projecte, i aprovats pel client, es van desenvolupar una sèrie d'especificacions tècniques per tal de complir amb els objectius.

En aquest cas no s'ha treballat amb el mètode del QFD, ja que les especificacions tècniques anaven molt lligades als objectius marcats pel client. Aquestes especificacions tècniques van ser determinades en la fase de l'avantprojecte, i quan es va procedir a la revisió d'aquest, abans de començar a treballar en la memòria del projecte, juntament amb el client es va establir que en l'avantprojecte es partia d'uns objectius molt extensos, ja que es desconeixia la complexitat de configurar l'entorn de treball pel TurtleBot3 Burger, i va ser per això que es van haver d'acotar.

Amb la solució obtinguda i determinades les necessitats del projecte, s'ha realitzat la viabilitat tècnica, la viabilitat mediambiental i per últim la viabilitat econòmica.

Una vegada es va comprovar la viabilitat del projecte, es va procedir a la instal·lació de l'entorn del robot TurtleBot3 Burger, el qual es troba detallat en el capítol 9. A continuació es va fer la posada en marxa del robot, i quins són els passos per dur-la a terme.

Una vegada ja estava tot l'entorn de programació tant del robot, com de la PC remota, s'han comentat algunes possibles aplicacions que es podrien executar, ja sigui per tal de comprovar el funcionament del robot, com de combinar diferents aplicacions per fer-ne una de més complexa.

Pel desenvolupament de l'aplicació final es va decidir partir de la base d'una aplicació, com és la de moure el robot a través del teclat de la PC remota, i modificar alguns

paràmetres, i a través del sensor i de la càmera instal·lada en el robot poder veure per on es mou el robot, és a dir, convertir una superfície possiblement desconeguda, en una totalment coneguda per l'usuari que es mogui amb el robot a través de la PC remota.

Arribat aquest punt, el següent pas que s'ha realitzat un pressupost, ja que aquest projecte va destinat a la Universitat Tecnocampus de Mataró. No s'ha realitzat un estudi de rendibilitat, ja que es tracta d'un projecte destinat a l'àmbit de recerca i innovació pel màster d'Indústria 4.0.

Per últim s'ha dut a terme el tancament, on es poden observar les desviacions de la planificació i del pressupost del projecte; incloent-hi també un resum de tot el coneixement après al llarg d'aquest per tal que en un futur pugui ser d'utilitat per a altres projectes similars, i on també s'ha afegit un apartat de possibles línies futures per aquest tipus de projecte.

13.2. Justificació de les desviacions

13.2.1. Desviacions de la planificació

Inicialment, es va fer una planificació del projecte, amb l'ajuda del software Microsoft Project, per tal d'obtenir uns costos i un termini d'entrega. Es varen enumerar una sèrie de tasques i sub-tasques per tal de programar el projecte de detall.

Un cop acabada la planificació es va obtenir el següent llistat de tasques:

Nom tasca	Inici	Finalització
1. Identificar el problema	dis 25/11/17	diu 10/12/17
2. Selecció de l'alternativa més adequada	diu 10/12/17	dim 20/12/17
3. Anàlisi de la viabilitat	dim 20/12/17	dij 28/12/17
4. Desenvolupament de la solució	dill 25/12/17	dim 02/01/18
5. Planificació	dim 02/01/18	div 12/01/18
LLIURAMENT AVANTPROJECTE DILL 12/01/18		
6. Objectius	dis 20/01/18	diu 21/01/18
7. Introducció	dis 27/01/18	diu 28/01/18
8. Objectius de detall i especificacions tècniques	diu 28/01/18	dim 30/01/18

9. Marc conceptual	dim 30/01/18	dis 03/02/18
10. Selecció de l'alternativa més adequada	dis 03/02/18	diu 04/02/18
11. Anàlisi de la viabilitat	dis 17/02/18	div 23/02/18
12. Planificació	div 23/02/18	div 02/03/18
13. Instal·lació entorn TurtleBot3 Burger	dis 10/03/18	dill 18/03/18
LLIURAMENT MEMÒRIA INTERMÈDIA DILL 19/03/18		
14. Aplicacions TurtleBot3 Burger	dim 21/03/18	dis 21/05/18
15. Aplicació TurtleBot3 Burger	dis 21/04/18	diu 20/05/18
LLIURAMENT MEMÒRIA FINAL DILL 21/05/18		
Duració total del projecte	**dis 25/11/17**	**diu 20/05/18**

Taula 13.1. Tasques principals planificades

Font: Elaboració pròpia

Malgrat aquest fet, inicialment tot anava sobre la planificació prevista, fins que es va començar a fer la instal·lació de l'entorn TurtleBot3 Burger, ja que es tenien establertes 20 hores perquè es suposava que seria relativament simple, però a l'hora de posar-ho en pràctica la realitat no va ser aquesta.

Com s'ha fet esment amb anterioritat, es tracta d'un robot relativament nou en el mercat, la qual cosa li implicava un plus per trobar determinada informació, i de vegades molta de la que es trobava no estava del tot ben contrastada, moltes vegades amb errades, és per això que a l'hora de fer la instal·lació algunes instruccions no estaven ben definides i per això donava error, si això se li suma la inexperiència de treballar amb comandos, es va promulgar més temps del que estava previst.

Així doncs, primer de tot es va haver de parlar amb el ponent, ja que el projecte s'havia convertit en fer una aplicació més complexa, Es suposava que la instal·lació es podria dur a terme en poques hores i aquesta no seria cap complicació del projecte, no obstant, no va ser així i la posada en marxa ha estat realitzada amb una aplicació més complexa. És per això que finalment es va decidir modificar les dates d'inici i finalització de les activitats, quedant doncs de la següent manera:

Nom tasca	Inici	Finalització
1. Identificar el problema	dis 25/11/17	diu 10/12/17
2. Selecció de l'alternativa més adequada	diu 10/12/17	dim 20/12/17
3. Anàlisi de la viabilitat	dim 20/12/17	dij 28/12/17
4. Desenvolupament de la solució	dill 25/12/17	dim 02/01/18
5. Planificació	dim 02/01/18	div 12/01/18
LLIURAMENT AVANTPROJECTE DILL 12/01/18		
6. Objectius	dis 20/01/18	diu 21/01/18
7. Introducció	dis 27/01/18	diu 28/01/18
8. Objectius de detall i especificacions tècniques	diu 28/01/18	dim 30/01/18
9. Marc conceptual	dim 30/01/18	dis 03/02/18
10. Selecció de l'alternativa més adequada	dis 03/02/18	diu 04/02/18
11. Anàlisi de la viabilitat	dis 17/02/18	div 23/02/18
12. Planificació	div 23/02/18	div 02/03/18
13. Instal·lació entorn TurtleBot3 Burger	dis 10/03/18	diu 08/04/18
LLIURAMENT MEMÒRIA INTERMÈDIA DILL 19/03/18		
14. Aplicacions TurtleBot3 Burger	diu 08/04/18	div 04/05/18
15. Aplicació TurtleBot3 Burger	div 04/05/18	diu 20/05/18
LLIURAMENT MEMÒRIA FINAL DILL 21/05/18		
Duració total del projecte	**dis 25/11/17**	**diu 20/05/18**

Taula 13.2. Reprogramació de la planificació

Font: Elaboració pròpia

Aquestes modificacions no han tingut un impacte directe en el pressupost, ja que es canvia la planificació del projecte, però el total d'hores continua essent el mateix.

13.3. Línies futures de treball

Com a línies futures hi ha un ventall extens dins del món del TurtleBot3 Burger, ja que avui en dia es pot trobar molta més informació que uns mesos enrere, amb l'avantatge que

en tractar-se d'un llenguatge de codi obert és molt més senzill que s'evolucioni de manera més ràpida.

El concepte de ROS permet crear nodes amb diferents varietats d'aplicacions, les quals poden estar escrites amb diferents tipus de llenguatges i a través de missatges comunicar-se entre ells, fent així de diferents aplicacions fer-ne una de més complexa sense la necessitat d'un gran codi que uneixi totes les diferents parts del codi.

Una solució interessant, la qual era la que es plantejava de manera inicial sense tenir coneixement de la dificultat de la instal·lació, era aconseguir moure el robot per una superfície ja coneguda per ell, possiblement fent un mapejat (SLAM), com s'ha comentat en aquest projecte, i marca dues estacions amb obstacles, siguin mòbils o fixes i que el robot fos capaç d'anar d'una estaciona a un altre sense topar amb cap obstacle.

Un altre possible solució seria fer ús de diferents aplicacions o exemple explicats en aquest projecte, com podrien ser fer un mapejat d'una sala i que aquest el faci de manera autònoma, és a dir sense la necessitat d'un usuari que el mogui per la sala i aquest anés captant els diferents punts de la sala.

Fins i tot, es podria fer ús de la càmera per identificar alguns objectes, ja que aquesta té la capacitat a través d'un programa de trobar similituds entre una imatge qualsevol i la real, poden aconseguir d'aquesta manera que el robot reconegui si es troba en la posició indicada per l'usuari.

14. Bibliografia i referències.

[1] J. &. M. Mistakes, «ROBOTIS e-manual,» 2018. [En línia]. Available: http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/.

[2] «Robotis TurtleBot3 Burger,» [En línia]. Available: https://cad.onshape.com/documents/2586c4659ef3e7078e91168b/w/14abf4cb615429a14a2732cc/e/9ae9841864e78c02c4966c5e.

[3] «Trossen Robotics,» [En línia]. Available: http://www.trossenrobotics.com/dynamixel-xl430-w250-t.aspx. [Últim accés: 2018].

[4] «ROBOTIS e-manual (XL430-W250),» [En línia]. Available: http://support.robotis.com/en/product/actuator/dynamixel_x/xl_series/xl430-w250.htm.

[5] «Raspberry Pi,» [En línia]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[6] T. Deneuville, «Thomas Deneuville,» 3 Octubre 2017. [En línia]. Available: https://www.thomasdeneuville.com/turtlebot3-first-explorations/. [Últim accés: 11 Febrer 2018].

[7] «Ubuntu,» [En línia]. Available: https://www.softcatala.org/programes/ubuntu/.

[8] W. Newman, A Systematic Approach to Learning Robot Programming with ROS, CRC Press, 2018.

[9] C. Lalancette, «ROS,» 23 Octubre 2017. [En línia]. Available: http://wiki.ros.org/Distributions. [Últim accés: 24 Febrer 2018].

TecnoCampus
Escola Superior Politècnica

Centre adscrit a la
upf. Universitat Pompeu Fabra Barcelona

Grau en enginyeria electrònica industrial i automàtica.

ESTUDI I APLICACIÓ DEL ROBOT MÒBIL TURTLEBOT3 BURGER

ESTUDI ECONÒMIC

SERGI COLOMÉ GRAU

PONENT: JOAN TRIADÓ

TARDOR 2018

TecnoCampus
Mataró-Maresme

Índex

Índex de taules.

1. Pressupost

1.1. Amidaments

En aquesta secció del pressupost es recullen els amidaments corresponents al treball que es realitzarà d'enginyeria (disseny i desenvolupament de la solució).

1.1.1. Capítol I: Elaboració del projecte

Nom tasca	Dedicació
1. Identificar el problema	68 hores
2. Selecció de l'alternativa més adequada	28 hores
3. Anàlisi de la viabilitat	20 hores
4. Desenvolupament de la solució	10 hores
5. Planificació	10 hores
6. Objectius	8 hores
7. Introducció	3 hores
8. Objectius de detall i especificacions tècniques	1 hora
9. Marc conceptual	4 hores
10. Selecció de l'alternativa més adequada	2 hores
11. Anàlisi de la viabilitat	3 hores
12. Planificació	8 hores
13. Instal·lació entorn TurtleBot3 Burger	135 hores
14. Aplicacions TurtleBot3 Burger	65 hores
15. Aplicació TurtleBot3 Burger	35 hores

Taula 1. Capítol I: Elaboració del projecte

Font: Elaboració pròpia

1.2. Quadre de preus

1.2.1. Capítol I: Elaboració del projecte

Codi	Concepte	Preu unitari
1	Hores enginyer	30,00 €/hora
2	Hores enginyer	30,00 €/hora
3	Hores enginyer	30,00 €/hora
4	Hores enginyer	30,00 €/hora
5	Hores enginyer	30,00 €/hora
6	Hores enginyer	30,00 €/hora
7	Hores enginyer	30,00 €/hora
8	Hores enginyer	30,00 €/hora
9	Hores enginyer	30,00 €/hora
10	Hores enginyer	30,00 €/hora
11	Hores enginyer	30,00 €/hora
12	Hores enginyer	30,00 €/hora
13	Hores enginyer	30,00 €/hora
14	Hores enginyer	30,00 €/hora
15	Hores enginyer	30,00 €/hora

Taula 2. Quadre de preus Capítol I

Font: Elaboració pròpia

1.3. Pressupost parcial

1.3.1. Capítol I: Elaboració del projecte

Nom tasca	Dedicació	Preu unitari (€)	Cost activitat
1. Identificar el problema	68 hores	30	2.040,00 €
2. Selecció de l'alternativa més adequada	28 hores	30	840,00 €
3. Anàlisi de la viabilitat	20 hores	30	600,00 €
4. Desenvolupament de la solució	10 hores	30	300,00 €
5. Planificació	10 hores	30	300,00 €
6. Objectius	8 hores	30	240,00 €
7. Introducció	3 hores	30	90,00 €
8. Objectius de detall i especificacions tècniques	1 hora	30	30,00 €
9. Marc conceptual	4 hores	30	120,00 €
10. Selecció de l'alternativa més adequada	2 hores	30	60,00 €
11. Anàlisi de la viabilitat	3 hores	30	90,00 €
12. Planificació	8 hores	30	240,00 €
13. Instal·lació entorn TurtleBot3 Burger	135 hores	30	600,00 €
14. Aplicacions TurtleBot3 Burger	65 hores	30	3.300,00 €
15. Aplicació TurtleBot3 Burger	35 hores	30	3.150,00 €

Taula 3. Costos directes Capítol I

Font: Elaboració pròpia

Concepte	Unitats	Import(€)
Costos indirectes mà d'obra	10% Costos directes	1.200 €

Taula 4. Costos indirecte Capítol I
Font: Elaboració pròpia

Concepte	Import(€)
Costos Directes Capítol I	12.000 €
Costos Indirectes Capítol I	1.200 €
Marge (30%)	3.960 €
TOTAL CAPÍTOL I	**17.160,00 €**

Taula 5. Pressupost parcial Capítol I

Font: Elaboració pròpia

1.3.2. Capítol II: Material

Codi	Concepte	Unitats	Preu unitari (€)	Import (€)
2.1	Material d'oficina	5	15	75,00
2.2	Impressió dels documents del projecte	180	0,09	16,20
2.3	Enquadernació del projecte	1	25	25,00

Taula 6. Costos directes Capítol II

Font: Elaboració pròpia

Codi	Concepte	Unitats	Import(€)
2.4	Costos indirectes de material	10% Costos directes	11,62

Taula 7. Costos indirecte Capítol II

Font: Elaboració pròpia

Concepte	Import(€)
Costos Directes Capítol II	116,20
Costos Indirectes Capítol II	11,62
Possibles imprevistos (15%)	19,17
TOTAL CAPÍTOL II	**146,99 €**

Taula 8. Pressupost parcial Capítol II

Font: Elaboració pròpia

1.3.3. Capítol III: Amortitzacions

Codi	Concepte	Unitats	Preu unitari (€)	Import (€)
3.1	TurtleBot3 Burger	1	496,11 €	496,11 €
3.1	Ordinador (Lenovo ThinkPad P51s)	1	1.323,92 €	1.323,92 €
3.2	Tarjeta SD Raspberry Pi MicroSD 32 GB	1	20,49 €	20,49 €
3.3	Cámara Raspberry, Módulo de vídeo V2	1	22,07 €	22,07 €

Taula 9. Amortitzacions Capítol III

Font: Elaboració pròpia

Concepte	Import(€)
Amortitzacions Capítol III	1.862,60
TOTAL CAPÍTOL III	**1.862,60 €**

Taula 10. Pressupost parcial Capítol III

Font: Elaboració pròpia

1.4. Pressupost Global

El pressupost del projecte inclou el cost del disseny del projecte. A continuació es mostra el cost del pressupost:

TOTAL CAPÍTOL I	17.160,00 €
TOTAL CAPÍTOL II	146,99 €
TOTAL CAPÍTOL III	1.862,60 €
BASE IMPONIBLE	19.169,59 €
IVA (21%)	4.025,61 €
TOTAL PRESSUPOST PROJECTE	**23.195,20 €**

Taula 11. Pressupost global

Font: Elaboració pròpia

o

TecnoCampus
Escola Superior Politècnica

Centre adscrit a la
upf. Universitat Pompeu Fabra Barcelona

Grau en enginyeria electrònica industrial i automàtica.

ESTUDI I APLICACIÓ DEL ROBOT MÒBIL TURTLEBOT3 BURGER

ANNEXES

SERGI COLOMÉ GRAU

PONENT: JOAN TRIADÓ

TARDOR 2018

TecnoCampus
Mataró-Maresme

Índex

Índex de figures.

Annex I. Planificació

Id.	Llista de tasques de disseny	Duració (h)	Prelacions	ASSIG. RECURSOS
A	**Identificar el problema**			
A1	Objecte del projecte	5 hores	INICI	Sergi Colomé
A2	Revisió d'antecedents	28 hores	A1	Sergi Colomé
A3	Necessitats d'informació	20 hores	A1	Sergi Colomé
A4	Abast del projecte	5 hores	A2,A3	Sergi Colomé
A5	Objectius i especificacións tecniques	10 hores	A2,A3	Sergi Colomé
B	**Selecció de l'alternativa més adequada**			
B1	Cerca d'alternatives	16 hores	A4,A5	Sergi Colomé
B2	Estudi d'alternatives	10 hores	B1	Sergi Colomé
B3	Selecció de l'alternativa més adequada	2 hores	B2	Sergi Colomé
C	**Anàlisi de la viabilitat**			
C1	Anàlisi de viabilitat tècnica	8 hores	B3	Sergi Colomé
C2	Anàlisi de viabilitat mediambiental	6 hores	B3	Sergi Colomé
C3	Anàlisi de viabilitat económica	6 hores	B3	Sergi Colomé
D	**Desenvolupament de la solució**			
D1	TurtleBot3 Burger	10 hores	C1	Sergi Colomé
E	**Planificació**			
E1	Ennumeració de tasques generals	2 hores	D1,D2	Sergi Colomé
E3	Establir duracións tasques	1 hora	E2	Sergi Colomé
E4	Establir tasques predecesores	2 hores	E3	Sergi Colomé
E5	Elaboració de la planificació en MS-Project	5 hores	E4	Sergi Colomé
F	**Entrega Avantprojecte**		E5	Sergi Colomé
G	**Objectius**			
G1	Propòsit	2 hores	F	Sergi Colomé
G2	Finalitat	2 hores	F	Sergi Colomé
G3	Objecte	2 hores	F	Sergi Colomé
G4	Abast	2 hores	F	Sergi Colomé
H	**Introducció**			
H1	Objectius	1 hora	G4	Sergi Colomé
H2	Revisió antecedents i necessitats informació	1 hora	G4	Sergi Colomé
H3	Límits del projecte	1 hora	G4	Sergi Colomé
I	**Objectius de detall i especificacions tècniques**			
I1	Objectius de detall i especificacions tècniques	1 hora	H3	Sergi Colomé
J	**Marc conceptual**			
J1	Ubuntu	2 hores	I1	Sergi Colomé
J2	ROS	2 hores	I1	Sergi Colomé
K	**Selecció de l'alternativa més adequada**			
L1	Estudi d'alternatives	1 hora	J2	Sergi Colomé
L2	Selecció de l'alternativa més adequada	1 hora	L1	Sergi Colomé
L	**Anàlisi de la viabilitat**			
L1	Anàlisi de viabilitat tècnica	1 hora	L2	Sergi Colomé
L2	Anàlisi de viabilitat mediambiental	1 hora	L2	Sergi Colomé
L3	Anàlisi de viabilitat económica	1 hora	L2	Sergi Colomé
M	**Planificació**			
M1	Desviacions de planificació	8 hores	L3	Sergi Colomé

N	**Instal·lació entorn TurtleBot3 Burger**			
N1	Instal·lació programari TurtleBot3 Burger	20 hores	M1	Sergi Colomé
O	**Entrega Memòria intermèdia**		N1	Sergi Colomé
P	**Aplicacions TurtleBot3 Burger**			
P1	Cerca aplicacions	30 hores	O	Sergi Colomé
P2	Comprensió arxius aplicacions	35 hores	P1	Sergi Colomé
P3	Execució aplicacions	45 hores	P2	Sergi Colomé
Q	**Aplicació TurtleBot3 Burger**			
Q1	Recopilació de aplicacions	35 hores	P3	Sergi Colomé
Q2	Elaboració aplicació	70 hores	Q1	Sergi Colomé
R	**Lliurament documentació final**		Q2	Sergi Colomé

Figura 1. Llista detallada de tasques

Font: Elaboració pròpia

Annex II. Codi detecció d'obstacles

Obre Desa

```
#!/usr/bin/env python
################################################################################
# Copyright 2018 ROBOTIS CO., LTD.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
################################################################################

# Authors: Gilbert #

import rospy
from sensor_msgs.msg import LaserScan
from geometry_msgs.msg import Twist

if __name__ == '__main__':
    rospy.init_node('turtlebot3_LDS')
    cmd_pub = rospy.Publisher('/cmd_vel', Twist, queue_size=1)
    scan = LaserScan()
    r = rospy.Rate(10)
    LIDAR_ERR = 0.05

    while not rospy.is_shutdown():
        scan = rospy.wait_for_message("/scan", LaserScan)
        twist = Twist()
        scan_filter = []
        twist.linear.x = 0.5
        twist.angular.z = 0.0

        for i in range(360):
            if i <= 15 or i > 335:
                if scan.ranges[i] >= LIDAR_ERR:
                    scan_filter.append(scan.ranges[i])

        if min(scan_filter) < 0.3:
            twist.linear.x = 0.0
            twist.angular.z = 0.5
        else:
            twist.linear.x = 0.05
            twist.angular.z = 0.0

        print(min(scan_filter))
        cmd_pub.publish(twist)
        r.sleep()
```

Python Amplada de la tabulació: 8 Ln 53, Col. 1 INSER

Figura 2. Codi detecció obstacles

Font: Elaboració pròpia

Annex III. Codi detecció d'obstacles

Obre Desa

```
#!/usr/bin/env python

# Copyright (c) 2011, Willow Garage, Inc.
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions are met:
#
#    * Redistributions of source code must retain the above copyright
#      notice, this list of conditions and the following disclaimer.
#    * Redistributions in binary form must reproduce the above copyright
#      notice, this list of conditions and the following disclaimer in the
#      documentation and/or other materials provided with the distribution.
#    * Neither the name of the Willow Garage, Inc. nor the names of its
#      contributors may be used to endorse or promote products derived from
#       this software without specific prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
# AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
# IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
# ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
# LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
# CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
# SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
# INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
# CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
# ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
# POSSIBILITY OF SUCH DAMAGE.

import rospy

from geometry_msgs.msg import Twist

import sys, select, termios, tty

msg = """
Mou el Turtlebot3 Burger!
---------------------------
        w
   a    s    d


w: Accelerar endavant
s: Accelerar enrere / Disminuir velocitat
a: Girar a la esquerra
d: Girar a la dreta
Tecla espai: Parar

CTRL-C per sortir de la App
"""

def getKey():
    tty.setraw(sys.stdin.fileno())
    rlist, _, _ = select.select([sys.stdin], [], [], 0.1)
    if rlist:
```

Python Amplada de la tabulació: 8 Ln 64, Col. 98 INSER

Figura 3. Primera part codi detecció obstacles

Font: Elaboració pròpia

Obre Desa

```
def getKey():
    tty.setraw(sys.stdin.fileno())
    rlist, _, _ = select.select([sys.stdin], [], [], 0.1)
    if rlist:
        key = sys.stdin.read(1)
    else:
        key = ''

    termios.tcsetattr(sys.stdin, termios.TCSADRAIN, settings)
    return key

def vels(target_linear_vel, target_angular_vel):
    return "currently:\tlinear vel %s\t angular vel %s " % (target_linear_vel,target_angular_vel)

if __name__=="__main__":
    settings = termios.tcgetattr(sys.stdin)

    rospy.init_node('turtlebot3_app')
    pub = rospy.Publisher('/cmd_vel', Twist, queue_size=5)

    status = 0
    target_linear_vel = 0
    target_angular_vel = 0
    control_linear_vel = 0
    control_angular_vel = 0
    try:
        print msg
        while(1):
            key = getKey()
            if key == 'w' :
                target_linear_vel = target_linear_vel + 0.1
                status = status + 1
                print vels(target_linear_vel,target_angular_vel)
            elif key == 's' :
                target_linear_vel = target_linear_vel - 0.1
                status = status + 1
                print vels(target_linear_vel,target_angular_vel)
            elif key == 'a' :
                target_angular_vel = target_angular_vel + 0.1
                status = status + 1
                print vels(target_linear_vel,target_angular_vel)
            elif key == 'd' :
                target_angular_vel = target_angular_vel - 0.1
                status = status + 1
                print vels(target_linear_vel,target_angular_vel)
            elif key == ' ' :
                target_linear_vel   = 0
                control_linear_vel  = 0
                target_angular_vel  = 0
                control_angular_vel = 0
                print vels(0, 0)
            elif status == 14 :
                print msg
                status = 0
```

Python Amplada de la tabulació: 8 Ln 64, Col. 98 INSER

Figura 4. Segona part codi detecció obstacles

Font: Elaboració pròpia

Obre Desa

```
            key = getKey()
            if key == 'w' :
                target_linear_vel = target_linear_vel + 0.1
                status = status + 1
                print vels(target_linear_vel,target_angular_vel)
            elif key == 's' :
                target_linear_vel = target_linear_vel - 0.1
                status = status + 1
                print vels(target_linear_vel,target_angular_vel)
            elif key == 'a' :
                target_angular_vel = target_angular_vel + 0.1
                status = status + 1
                print vels(target_linear_vel,target_angular_vel)
            elif key == 'd' :
                target_angular_vel = target_angular_vel - 0.1
                status = status + 1
                print vels(target_linear_vel,target_angular_vel)
            elif key == ' ' :
                target_linear_vel   = 0
                control_linear_vel  = 0
                target_angular_vel  = 0
                control_angular_vel = 0
                print vels(0, 0)
            elif status == 14 :
                print msg
                status = 0
            else:
                if (key == '\x03'):
                    break

            if target_linear_vel > control_linear_vel:
                control_linear_vel = min( target_linear_vel, control_linear_vel + (0.01/4.0) )
            else:
                control_linear_vel = target_linear_vel

            if target_angular_vel > control_angular_vel:
                control_angular_vel = min( target_angular_vel, control_angular_vel + (0.1/4.0) )
            else:
                control_angular_vel = target_angular_vel

            twist = Twist()
            twist.linear.x = control_linear_vel; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = control_angular_vel
            pub.publish(twist)

    except:
        print e

    finally:
        twist = Twist()
        twist.linear.x = 0; twist.linear.y = 0; twist.linear.z = 0
        twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 0
        pub.publish(twist)

    termios.tcsetattr(sys.stdin, termios.TCSADRAIN, settings)
```

Python Amplada de la tabulació: 8 Ln 64, Col. 98 INSER

Figura 5. Tercera part codi detecció obstacles

Font: Elaboració pròpia

Annex IV. Instruccions bàsiques terminal Linux

- **Cat:** és una meravellosa utilitat que permet visualitzar el contingut d'un arxiu de text sense la necessitat d'un editor. Per utilitzar-lo només s'ha de mencionar el fitxer que es desitja visualitzar:

  $ cat Prova.txt

- **Ls:** permet llistar el contingut d'un directori o fitxer. La sintaxi és:

  $ ls /home/directorio

- **Cd (change directory o canviar directori):** és com el seu nom indica la comanda que es necessitarà per accedir a una ruta diferent de la que es troba inicialment. Per exemple, si es parteix del directori /home i es vol accedir a /home/exercicis, seria:

  $ cd /home/exercicis

  Si es parteix de /home/exercicis i es vol pujar un nivell (és a dir anar al directori /home), executes:

  $ cd ..

- **Mkdir (make directory o crear directori):** crea un directori nou tenint en compte la ubicació actual, és a dir, si es troba a /home i es desitja crear el directori exercicis, seria:

  $ mkdir /home/ejercicios

- Mv (*move* o moure), mou un fitxer a una ruta específica, i a diferència de cp, ja que aquest l'elimina de l'origen. Per exemple:

  $ mv /home/prueba.txt /home/respaldos/prueba2.txt

- Clear (netejar), és un senzill comandament que neteja el terminal completament deixant-la com acabada d'obrir. Per a això s'ha d'executar:

  $ clear