INDOOR REAL-TIME NAVIGATION FOR ROBOT VEHICLES

Projet GSE

LIU BOHUA – SUN HAO
POLYTECH NICE SOPHIA
930 Route des Colles, 06410 Biot

Table de contents

Objectifs :

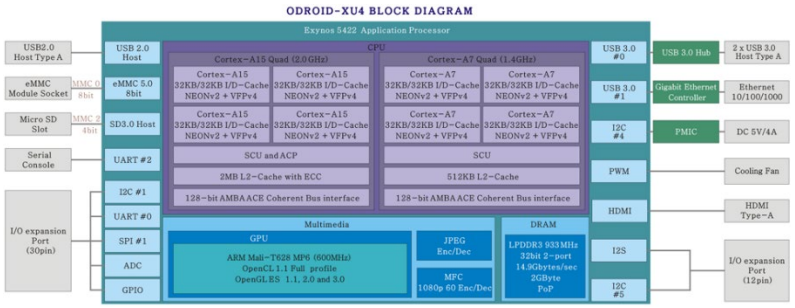
L'objectif du projet est de concevoir un système de navigation en temps réel pour un véhicule robot.

I. Vue globale système

Pour intégrer un système de navigation en temps réel, d'abord, nous utilisons un scanner laser autonome portable (Lidar) contrôlé par la carte Exynos comme le dispositif d'acquisition et une webcam pour transmettre les images. Et après, les données collectées par Lidar sont transmises à PC exécutant SLAM et des logiciels de visualisation. L'appareil d'acquisition et la station de base communiquent à l'aide du Robot Operating System (ROS) et d'une connexion Wifi. Ensuite, les images d'objets transmis par la webcam sont analysées par YOLO. Enfin, nous utilisons le Turtlebot pour réaliser la navigation.

II. Composants matériels

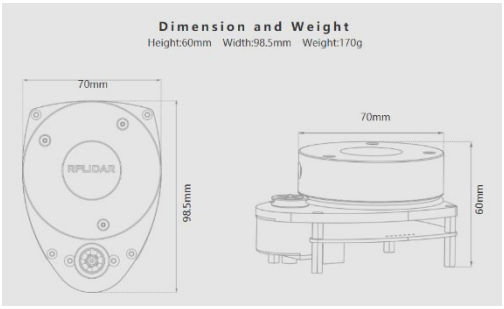
A. Carte Exynos



La carte Exynos ODROID-XU4 que nous utilisons est une nouvelle génération de dispositif informatique avec un matériel plus puissant et économe en énergie et un facteur de forme plus petit. La carte Exynos peut exécuter différentes versions de Linux en offrant un support open source. Et voici les spécifications ci-dessous de la carte Exynos :

Processor	Samsung Exynos5 Octa ARM Cortex™-A15 Quad 2Ghz and Cortex™-A7 Quad 1.3GHz CPUs
Memory	2Gbyte LPDDR3 RAM at 933MHz (14.9GB/s memory bandwidth) PoP stacked
3D Accelerator	Mali-T628 MP6(OpenGL ES 3.0/2.0/1.1 and OpenCL 1.1 Full profile)
Video	supports 1080p via HDMI cable(H.264+AAC based MP4 container format)
Video Out	Standard Type A HDMI connector
Audio	Instead of On-board Audio codec, There is I2S Expansion Port(CON11)
USB3.0 Host	SuperSpeed USB standard A type connector x 2 port, Max Load: total 2Amp for two USB 3.0 host ports
USB2.0 Host	High Speed standard A type connector x 1 ports, Max Load: 500mA/port
Display	HDMI monitor
Storage (Option)	MicroSD Card Slot, eMMC module socket : eMMC 5.0 HS4000 Flash Storage
Gigabit Ethernet LAN	10/100/1000Mbps Ethernet with RJ-45 Jack (Auto-MDIX support)
Serial console port	Connecting to a PC gives access to the Linux console. You can see the log of the boot, or to log in to the C1 to change the video or network settings. Note that this serial UART uses a 1.8 volt interface. We recommend the USB-UART module kit from Hardkernel. Molex 5268-04a(2.5mm pitch) is mounted on the PCB. Its mate is Molex 50-37-5043 Wire-to-Board Crimp Housing.
RTC (Real Time Clock) backup battery connector	If you want to add a RTC functions for logging or keeping time when offline, just connect a Lithium coin backup battery (CR2032 or equivalent). All of the RTC circuits are included on the ODROID-XU4 by default. Molex 53398-0271 1.25mm pitch Header, Surface Mount, Vertical type (Mate with Molex 51021-0200)
WiFi(Option)	USB IEEE 802.11b/g/n 1T1R WLAN with Antenna (USB module)
HDD/SSD SATA interface (Optional)	SuperSpeed USB (USB 3.0) to Serial ATA3 adapter for 2.5"/3.5" HDD and SSD storage
Power (included)	5V 4A Power
Case(Option)	Mechanical case & cooler (90 x 59 x 28 mm approx.)
PCB Size	83 x 58 x 20 mm approx.

B. RP Lidar



Le lidar que nous utilisons s'appelle RPLIDAR A1. En tant que le capteur principal, il peut rapidement obtenir les informations générales de l'environnement. Comparé à la technologie traditionnelle, avec un appareil laser infrarouge, le RPLIDAR A2 peut réaliser le toucher multipoint sur un écran très grand, qui a une résolution plus élevée, réagit rapidement et résiste à la lumière ambiante. Voici la performance de mesure ci-dessous :

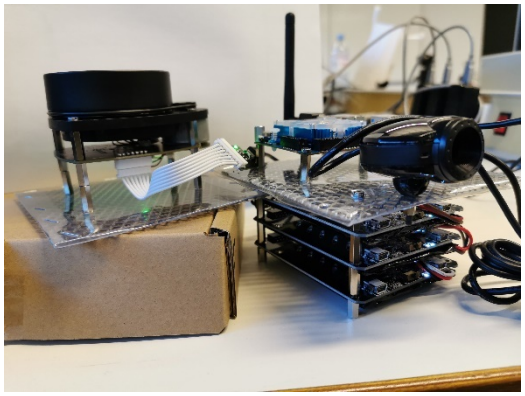
Measurement Performance

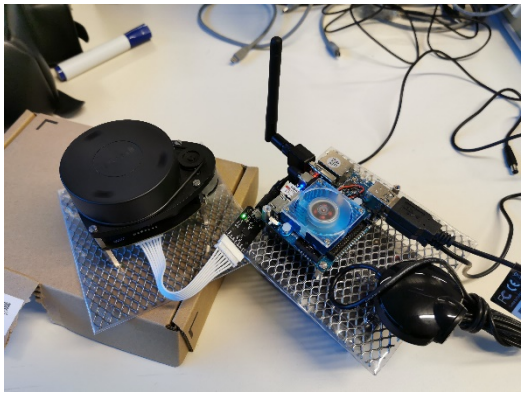
Item	Unit	Min	Typical	Max	Comments
Measurement Range	Meter(s)	TBD	0.15 - 12	TBD	Test on while reflective objects
Angular Range	Degrees	Not Applicable	0 - 360	Not Applicable	
Measurement Res.	mm	Not Applicable	< 0.5 --- <1% of actual distance *	Not Applicable	Within 1.5 meters --- For whole measurement range *
Angular Res.	Degrees	Not Applicable	≤1	Not Applicable	Scan at 5.5hz
Time for single measurement	ms	Not Applicable	0.5	Not Applicable	
Measurement Freq.	Hz	2000	≥4000	8000	
Scan Freq.	Hz	5	5.5	10	Typical value based on 360 measurements per round

C. Webcam

La webcam que nous utilisons c'est « Hercules Dualpix Exchange ». Une fois que l'image a été collectée par la webcam, l'image peut être reconnu par l'ordinateur par le circuit du composant photosensible et le composant de contrôle à l'intérieur de la caméra, et ensuite elle est traitée et convertie en un signal numérique avant d'être transmis à PC par Wi-Fi.

D. Intégration des composants matériels





Nous avons intégré les composants matériels comme la photo ci-dessus. Pour alimenter la carte Exynos en puissance 5V et 4A, nous utilisons trois batteries (chacun de 1~1.5 A et 5v) en parallèle. Ensuite nous communiquons entre la carte Exynos et le lidar ou la webcam via une connexion USB. Pour le support de montage, nous utilisons les plats métalliques avec une couche de plastique qui le recouvre et après nous avons fixé tous les composants dessus.

III. Composants logiciels

Voici les introductions et les fonctions des composants logiciels, et puis nous avons mis la méthode d'installation dans l'annexe.

A. Système d'opération

Nous espérons utiliser un système embarqué sur la carte Exynos pour collecter les données et de même temps un système sur le PC pour recevoir les données et après il doit analyser et traiter les données. Nous pouvons réaliser une carte à intégrer les composants matériels et obtenir un robot portable. Evidement, nous obtenons le PC qui peut traiter les données rapidement et réaliser la visualisation.

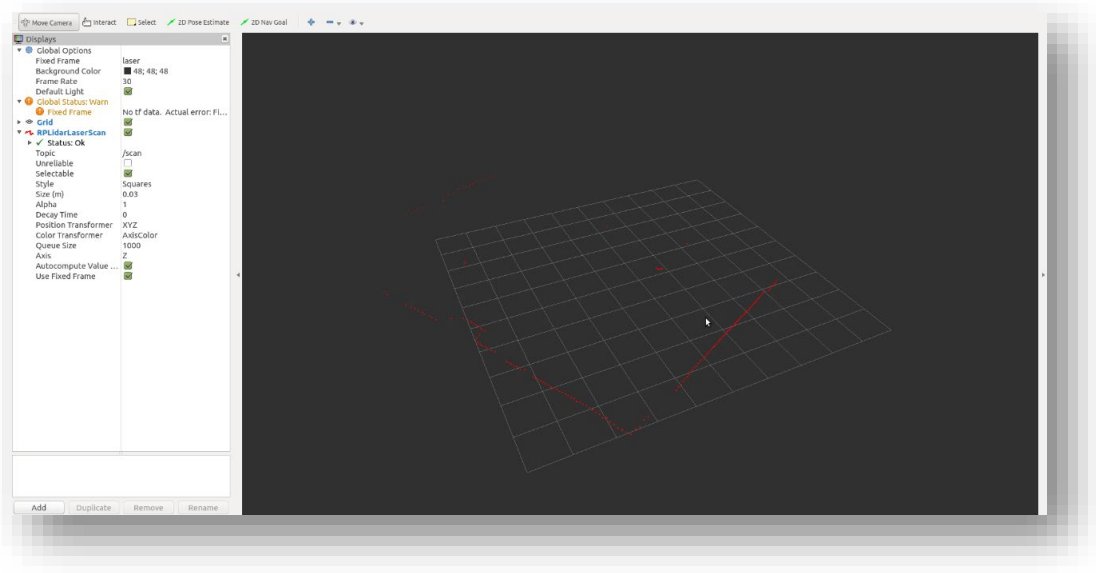
La version de Ubuntu de carte Exynos est ubutu16.4. D'abord, nous avons téléchargé l'image d'Ubuntu en version de ubuntu-16.04.3-4.14-mate pour l'odroid-xu4. Il faut faire attention que la version d'Ubuntu doit exactement pareil que la carte Exynos, ici c'est xu4. Et ensuite nous avons chargé le système d'opération sur une carte SD en utilisant une application téléchargée (Win32 Disk Imager). Dernière étape, nous avons fait l'installation et le système d'opération a fonctionné bien.

B. SLAM algorithme

La localisation et cartographie simultanées, connue en anglais sous le nom de SLAM (Simultaneous Localization And Mapping) ou CML (Concurrent Mapping and Localization), consiste, pour un robot ou véhicule autonome, à simultanément construire ou améliorer une carte de son environnement et de s'y localiser. [Wikipédia] Dans ce projet, nous pouvons réaliser un SLAM algorithme intégré dans l'environnement ROS. Le SLAM a deux parties : RVIZ, l'outils de visualisation 3D et Google Cartographer.

i. RVIZ

RVIZ est un outil de visualisation 3D pour les applications ROS. Il fournit une vue du modèle de robot avec l'obtention des informations des capteurs du robot et la lecture des données. Ensuite il peut afficher les données des caméras, des Lidar ou des appareils 3D et 2D. Voici la photo ci-dessous :

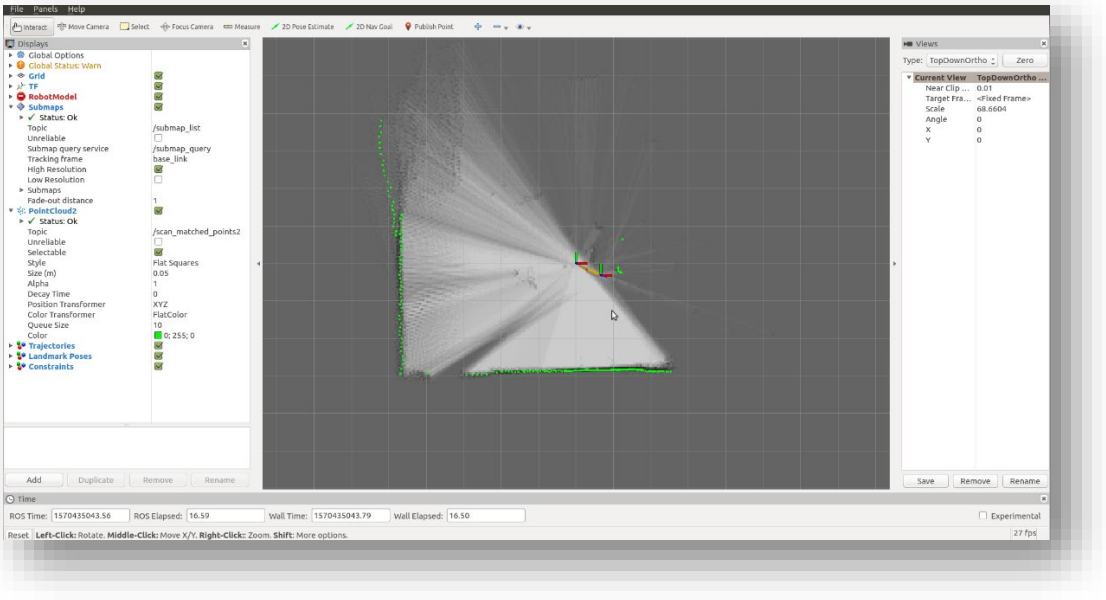


C'est la surface de RVIZ. On peut voir qu'il peut afficher les points qui est reçus par le lidar. Nous ne pouvons pas tracer les choses maintenant parce ce sont justement les informations collectées sans les analyser.

ii. Google Cartographer

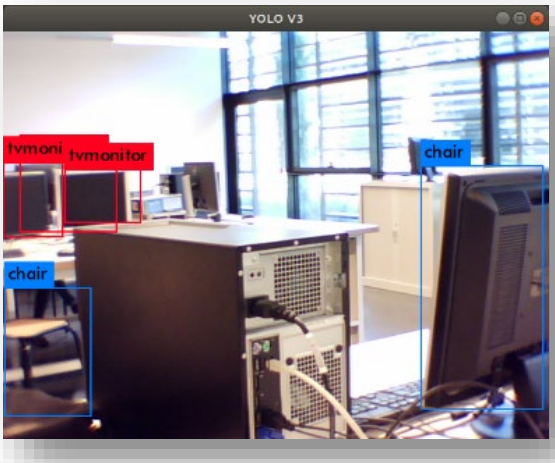
Pour analyser et traiter les informations collectées, il faut utiliser le Google Cartographer. Il est un système qui fournit une localisation et une cartographie simultanées en temps réel en 2D et 3D sur

plusieurs plates-formes et configurations de capteurs. Grâce à Google Cartographer, nous pouvons distinguer le mur, la porte ou le fenêtre au lieu des points dispersé. Après l'installation de Cartographer, nous obtenir la photo d'un coin de notre laboratoire comme ci-dessous :



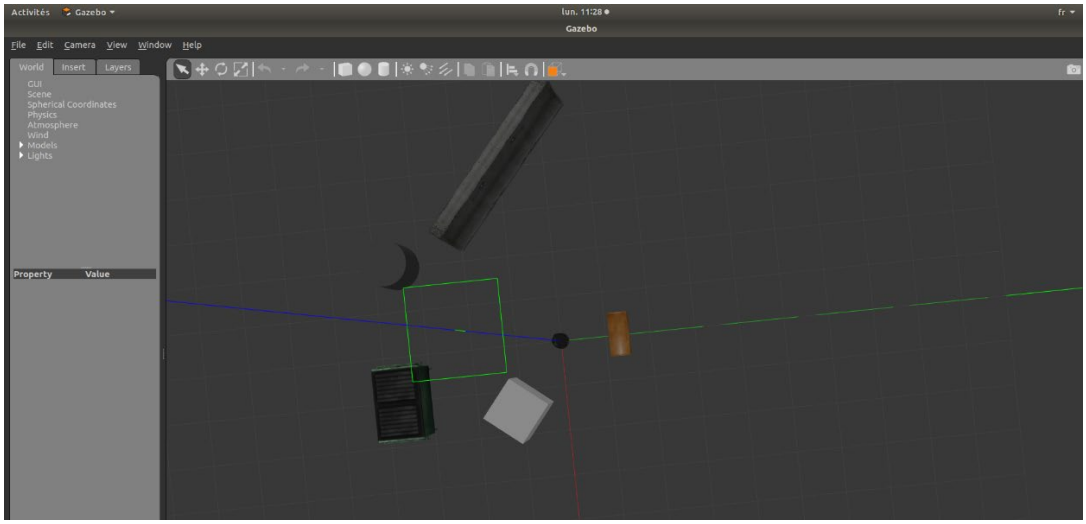
C. Détection d'objets

YOLO (You Only Look Once) est un système de détection d'objet en temps réel, qui est utilisé avec le support d'$OpenCV_{3.4.3}$($OpenCV_4$ ne fonctionne pas bien). Nous utilisons le framework darknet de YOLO pour détecter les objets capturés par la Webcam en temps réel comme ci-dessous.



D. Navigation par Turtlebot

Nous utilisons le package Turtlebot pour la navigation de ROS. Parmi eux, le move_base est pour effectuer la planification du trajet en fonction du message référence, afin que le robot mobile atteigne la position désignée. Le gmapping est pour créer une carte sur des données de Lidar. L'amcl est pour la localisation sur des cartes existantes. D'abord nous lançons le monde du Gazebo comme ci-dessous, puis on veut enregistrer une carte à l'aide de map_server, mais il y a quelques problème pour lancer le fichier amcl_demo.launch, donc finalement nous n'avons pas réussi à faire la navigation autonome. Mais nous avons enregistré un rosbag.

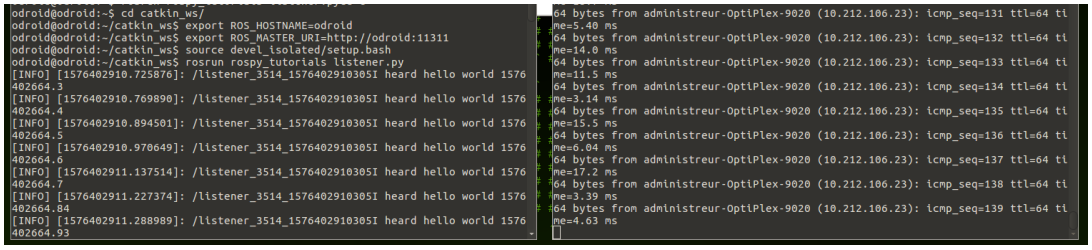


E. Communication

i. Configuration des adresses

Nous espérons réaliser la communication entre un PC et la carte Exynos sur un robot mobile dans un environnement de laboratoire. Le PC est utilisé comme le Slave et la carte Exynos est utilisé comme le Master. Il y a un WiFi sur le PC et aussi sur la carte Exynos. Deux WiFi sont sur le même réseau.

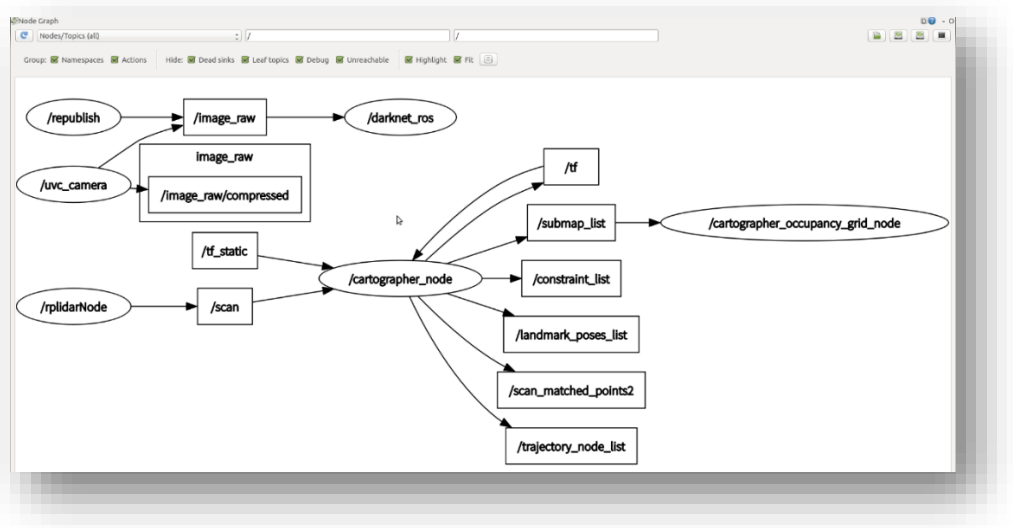
Voici la photo de la configuration de IP. Il faut écrire les IPs de deux dispositifs dans la fichier */etc/hosts*.



Après faire la configuration de IP, nous pouvons réaliser la communication entre les deux comme la photo ci-dessus. Si après nous écrivons ping, il affiche la vitesse et la latence, c'est-à-dire les deux dispositifs est bien connectée. Pour réaliser la communication dans l'environnement ROS, nous utilisons un *listener* et un *talker*. Le *talker* transmet des messages et ensuite le *listener* va le recevoir. Si nous pouvons voir les messages reçus dans le *listener*, il prouve qu'il réussit à communiquer.

ii. Communication entre les composants

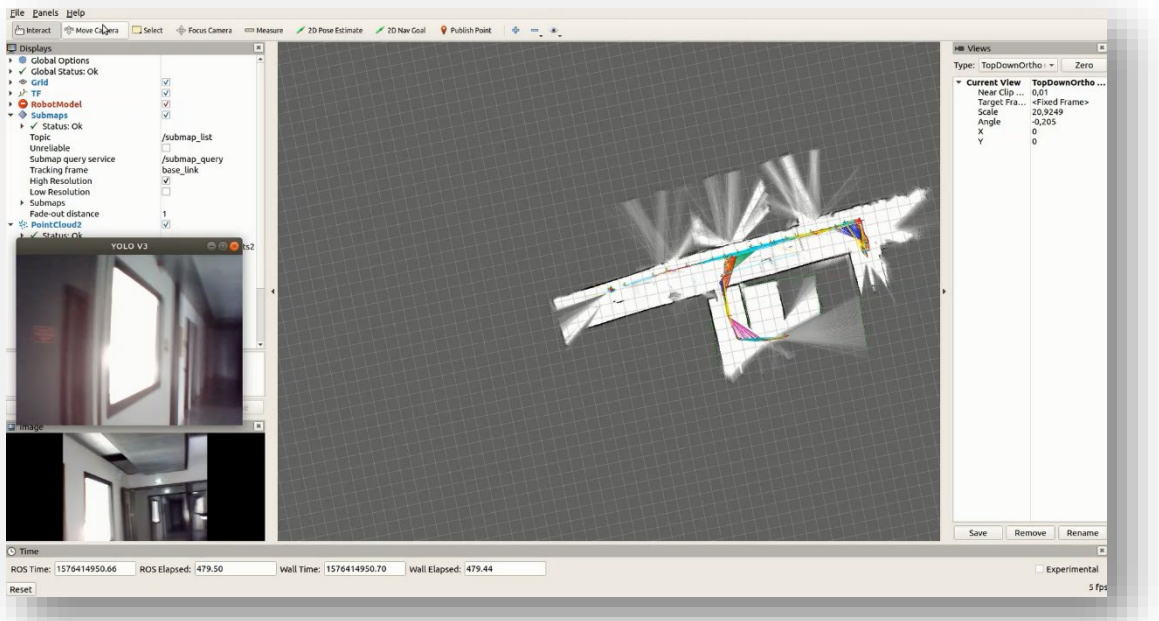
Pour réaliser la communication entre les composants, il faut avoir besoin d'aide avec *Node*, *Topic* et *Message* de ROS. Un *Node* est un fichier exécutable, et les *Node* communiquent via le système ROS. Ensuite les *Node* peuvent publier des messages sur les *Topic* ou s'abonner à certains *Topic* pour recevoir des *Message* sur ces *Topic*. Enfin le *Message* est un type de données de ROS utilisé pour s'abonner ou publier un *Topic*. Voici le schéma ci-dessous affiche tous les éléments (*Node*, *Topic* et *Message*) de notre ROS pendant la communication :



Nous pouvons voir qu'il y a le *Node* de lidar et de webcam qui sont les périphériques à transmettre les messages. Le *Topic* de webcam est transmis à darknet et le Topic de lidar est transmis à Google Cartographer qui est à PC. Comme ça, nous avons fini la communication entre les composants.

IV.Résultat

Nous avons tracé la carte de deuxième étage négatif du bâtiment S, le couloir et une chambre. Ci-dessous est une capture d'écran de notre mesure réelle :



V. Conclusion :

Dans ce projet, nous avons intégré un système de navigation en temps réel. Nous avons intégré tous les composants matériels à la carte Exynos et ensuite nous avons installé les composants logiciels comme RVIZ, Google Cartographer en base de ROS, YOLO en base de OpenCV et Turtlebot. Donc nous avons réalisé les communications entre différents composants, la détection et le lever de la carte d'un bâtiment. Mais enfin nous n'avons pas bien réussi à faire la partie de navigation.

VI.Annexe :

A. Les références d'installation et de configuration :

- L'installation de ubutu16.4 en Odroid-xu4 :
  - https://www.youtube.com/watch?v=VEIS1bzxN2E
- L'installation de ROS en Ubuntu 16.04 (version Kinetic)
  - https://blog.csdn.net/geerniya/article/details/83381597
  - https://blog.csdn.net/huacode/article/details/83182384 (catkin_make_isolated)
- ROS Cartographer
  - https://google-cartographer-ros.readthedocs.io/en/latest/compilation.html#
  - https://blog.csdn.net/qq_29230261/article/details/84101758
- Linux swap :
  - https://www.tok9.com/archives/466/
- Télécharger de 2D Demo :
  - $ wget -P ~/Downloads https://storage.googleapis.com/cartographer-public-data/bags/backpack_2d/cartographer_paper_deutsches_museum.bag
  - $ roslaunch cartographer_ros demo_backpack_2d.launch
- Communication distribuée multi-machine ROS
  - https://blog.csdn.net/hehedadaq/article/details/82898307#ros_0
  - https://blog.csdn.net/heyijia0327/article/details/42080641
- Le solution de problème « Could not connect to display »:
  - https://blog.csdn.net/u011331731/article/details/95009911
- La configuration de uvc camera en ROS :
  - https://blog.csdn.net/qq_43433255/article/details/89332667
- Les types de webcam que Linux supporte :
  - https://blog.csdn.net/htjx99/article/details/13613571
- La bibliothèque de OpenCV :
  - https://www.w3cschool.cn/opencv/opencv-2gnx28u3.html
- Tutoriel de construction de simulation de TurtleBot3 :
  - https://www.jianshu.com/p/014552bcc04c
  - https://www.ncnynl.com/archives/201609/809.html
- La navigation autonome de Turtlebot :
  - https://learn.turtlebot.com/2015/02/03/9/

B. Les étapes d'opération :

i. Sur le système de carte Exynos (Master) :

- Terminal 1 :
  - Lancer le ROS master
  - $sudo su
  - $swapon /root/swapfile
  - $echo "/var/swapfile swap swap defaults 0 0" >>/etc/fstab
  - $free –h
  - $cd catkin_ws
  - $source /home/odroid/catkin_ws/install_isolated/setup.bash
  - $catkin_make_isolated
  - $ssh odroid
  - $roscore
- Terminal 2 :
  - Connecter le lidar
  - $sudo chmod 666 /dev/ttyUSB0
  - $export ROS_HOSTNAME=odroid
  - $export ROS_MASTER_URI=http://odroid:11311
  - $source /home/odroid/catkin_ws/devel_isolated/setup.bash
  - $roslaunch rplidar_ros rplidar.launch
- Terminal 3 :
  - $export ROS_HOSTNAME=odroid
  - $export ROS_MASTER_URI=http://odroid:11311
  - $source /home/odroid/catkin_ws/devel_isolated/setup.bash
  - $rosrun rospy_tutorials listener.py
- Terminal 4 :
  - Connecter la webcam
  - $export ROS_HOSTNAME=odroid
  - $export ROS_MASTER_URI=http://odroid:11311
  - $source /home/odroid/catkin_ws/devel_isolated/setup.bash
  - $rosrun uvc_camera uvc_camera_node

ii. Sur le système de PC (Slave) :

- Terminal 1 :
  - Lancer le Lidar
  - $cd catkin_ws
  - $ssh administreur-OptiPlex-9020
  - $export ROS_HOSTNAME=administreur-OptiPlex-9020
  - $export ROS_MASTER_URI=http://odroid:11311
  - $export DISPLAY=':0.0'
  - $source /home/odroid/catkin_ws/devel_isolated/setup.bash
  - $roslaunch cartographer_ros demo_revo_lds.launch
- Terminal 2 :
  - Lancer le darknet_ros
  - $cd catkin_ws
  - $ssh administreur-OptiPlex-9020
  - $export ROS_HOSTNAME=administreur-OptiPlex-9020
  - $export ROS_MASTER_URI=http://odroid:11311

- $export DISPLAY=':0.0'
- $source /home/odroid/catkin_ws/devel_isolated/setup.bash
- roslaunch darknet_ros darknet_ros.launch

- Terminal 3 :
  - Lancer le monde de Gazebo
  - $cd catkin_ws
  - $ssh administreur-OptiPlex-9020
  - $export ROS_HOSTNAME=administreur-OptiPlex-9020
  - $export ROS_MASTER_URI=http://odroid:11311
  - $export DISPLAY=':0.0'
  - $ roslaunch turtlebot_gazebo turtlebot_world.launch world_file:=<full path to the world file>
- Terminal 4 :
  - Exécuter la navigation
  - $cd catkin_ws
  - $ssh administreur-OptiPlex-9020
  - $export ROS_HOSTNAME=administreur-OptiPlex-9020
  - $export ROS_MASTER_URI=http://odroid:11311
  - $export DISPLAY=':0.0'
  - roslaunch turtlebot_gazebo amcl_demo.launch map_file:=<full path to map yaml file>
- Terminal 5 :
  - Lancer le Rviz
  - $cd catkin_ws
  - $ssh administreur-OptiPlex-9020
  - $export ROS_HOSTNAME=administreur-OptiPlex-9020
  - $export ROS_MASTER_URI=http://odroid:11311
  - $export DISPLAY=':0.0'
  - roslaunch turtlebot_rviz_launchers view_navigation.launch