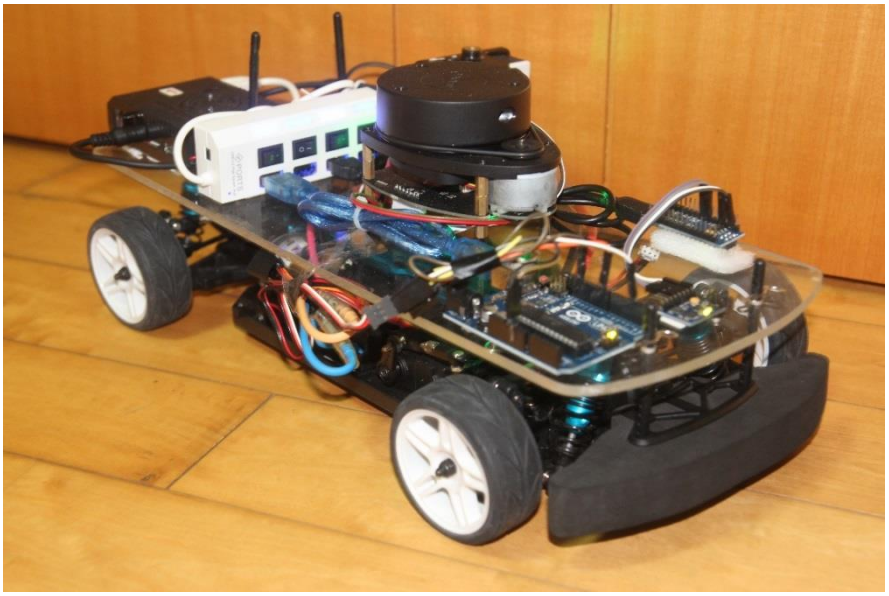
HyphaROS RaceCar Tutorial

HyphaROS Workshop

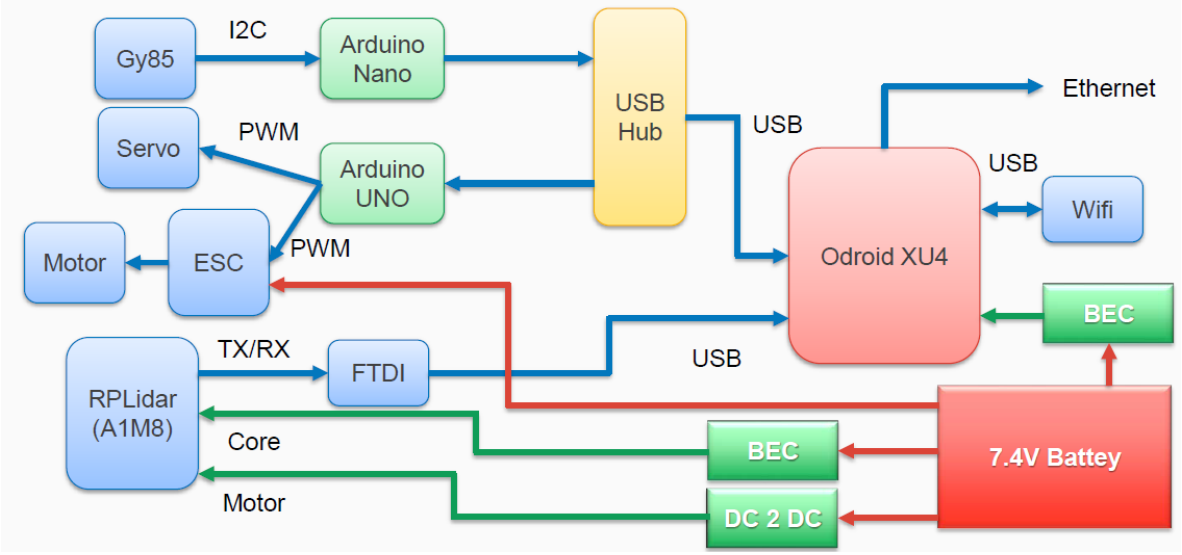


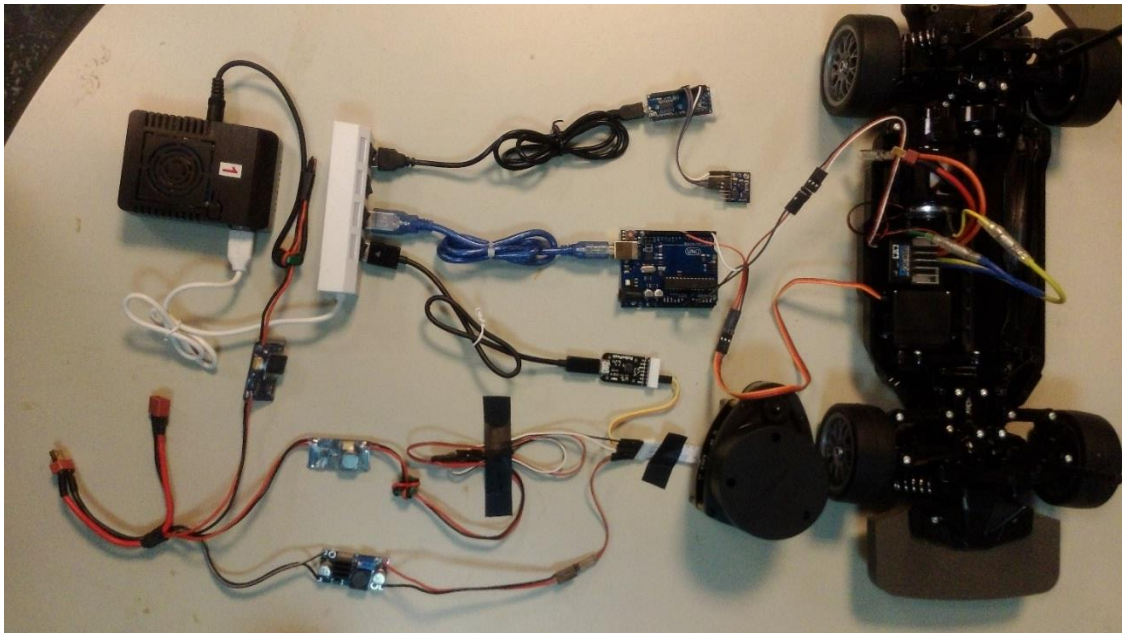
Contents

Hardware Integration

Hardware Configuration

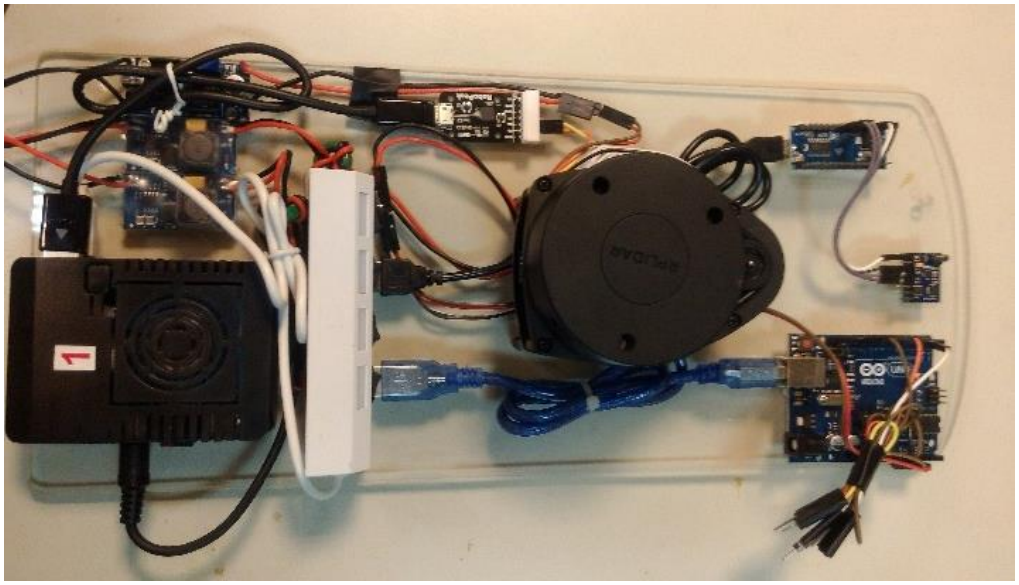
Item	Quantity
1/10 ready to go car set: **HSP 94123 PRO 1/10 RC Car kit (or Tamiya TT02)**	1
FUTABA S9151 digital servo (or Tower pro MG996R Servo)	1
7.4v 5000 Mah battery: **TANK POWER 7.4V/5200mah, 30C LiPo battery**	1
TAMIYA ESC TBLE-02S	1
B6 IMAX Charger (or any kind of charger for LiPo battery)	1
T-plug Connector (1 pair)	2
Odroid XU4 (US plug)	1
Odroid official wifi module 3	1
Odroid Cases	1
16G Micro SD	1
low voltage protection buzzer (~2 USD)	1
RPLidar A1 (A1m1 or A1m8)	1
USB 2.0 Hub	1
CRIUS BEC 5V/5A	2
DC to DC (5~10V output)	1
Arduino UNO	1
Arduino Nano	1
GY-85 IMU	1
Nylon Pillars M3 12mm	8
Copper Pillars M3 10mm	10
40P Color Dupont line	1
Acrylic board	1





The hardware can divided into two part: chassis and payload. Chassis contains the car's structure; Drive Line System; Tires; motor; ESC （Electronic speed control ）; Steering system servo and Battery.

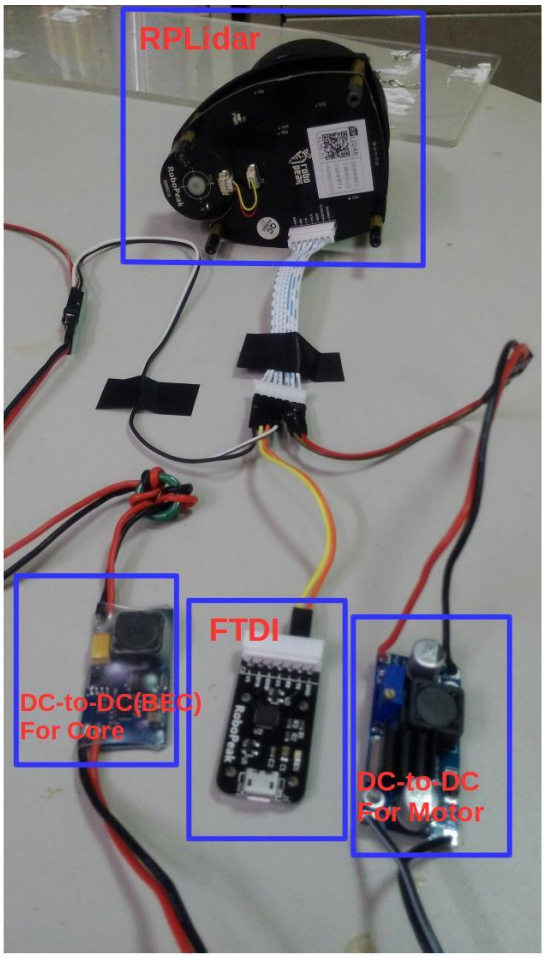
Upper stage carry the sensor; computer and all the Electronic needed.

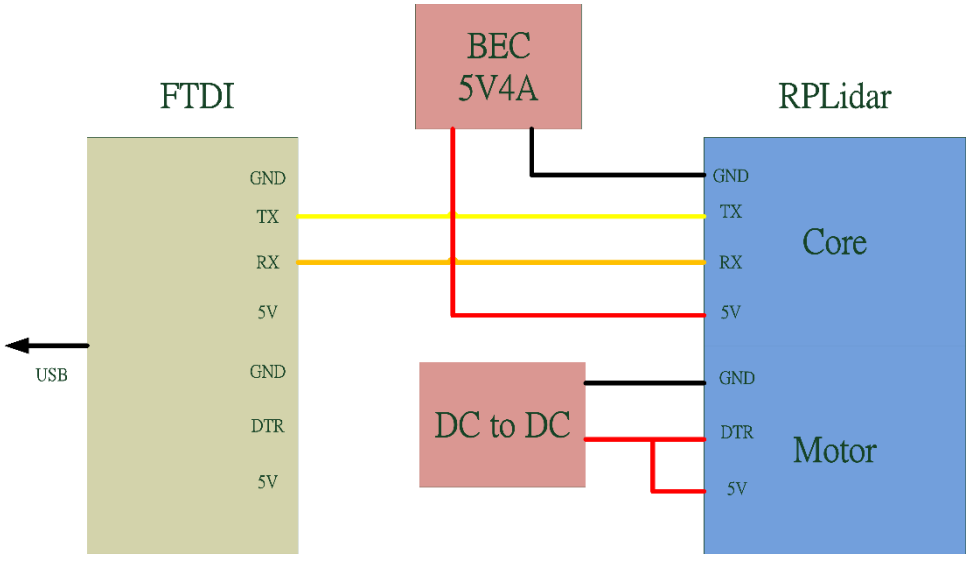


Component connection

- RPLidar

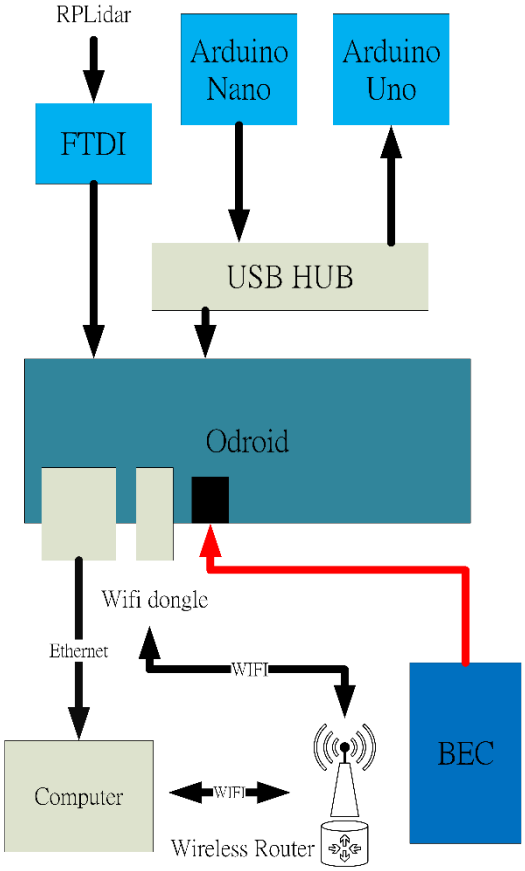
RPLidar requires high quality power supply, therefor we connect the core's power to one standalone BEC to provide power. And the Motor's power provide a variable DC to DC converter so we can adjust the rotation speed of motor.





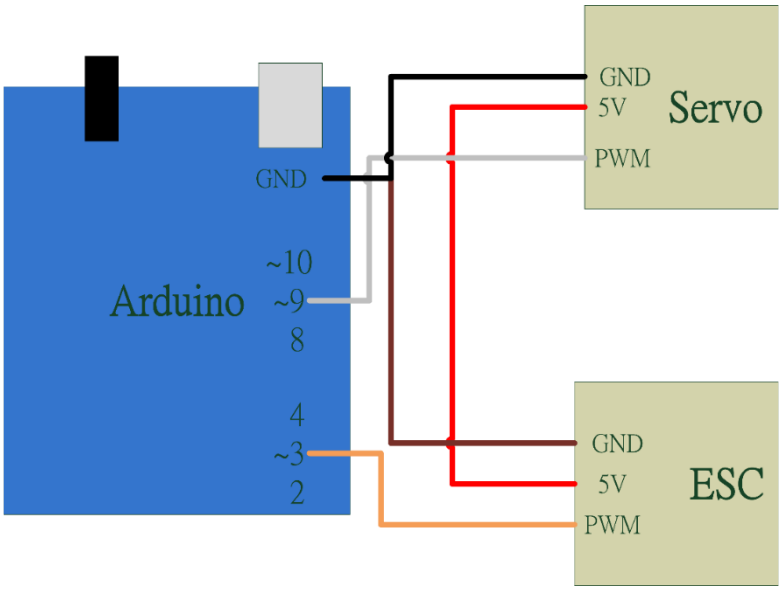
- Odroid

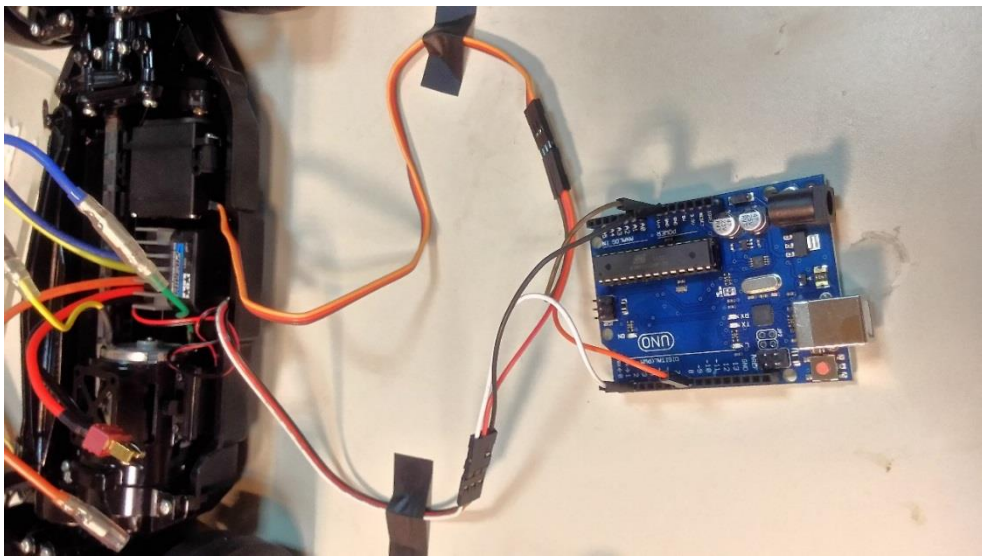
The Odroid connection can refer to the following figure. One thing important is don't connect the RPLidar FTDI though USB Hub, it should directly connect to Odroid USB to make sure it have stable communication.



- Arduino Uno

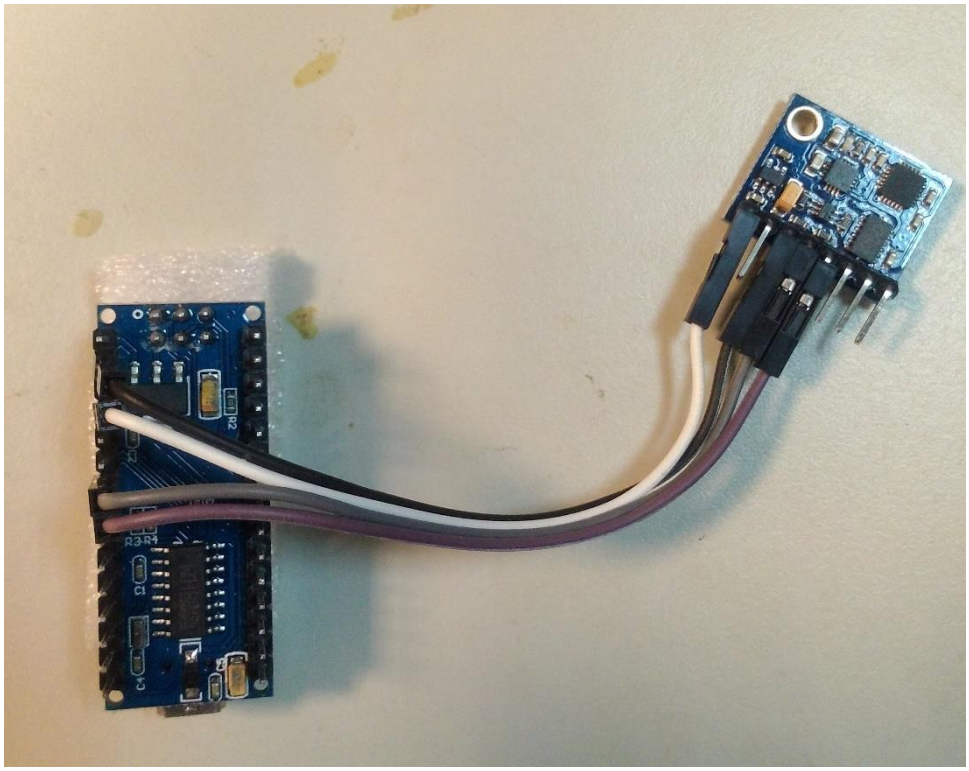
Arduino Uno in this project is responsible for control the car, means control the motor though ESC and Servo. The ESC we use here is contain the BEC, it can provide the 5V power to SERVO.

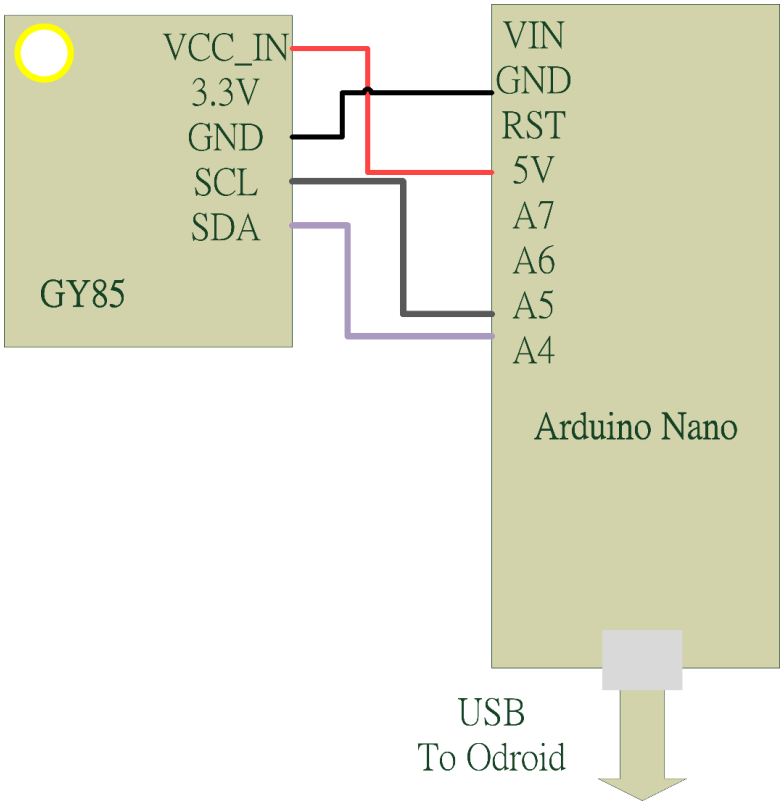




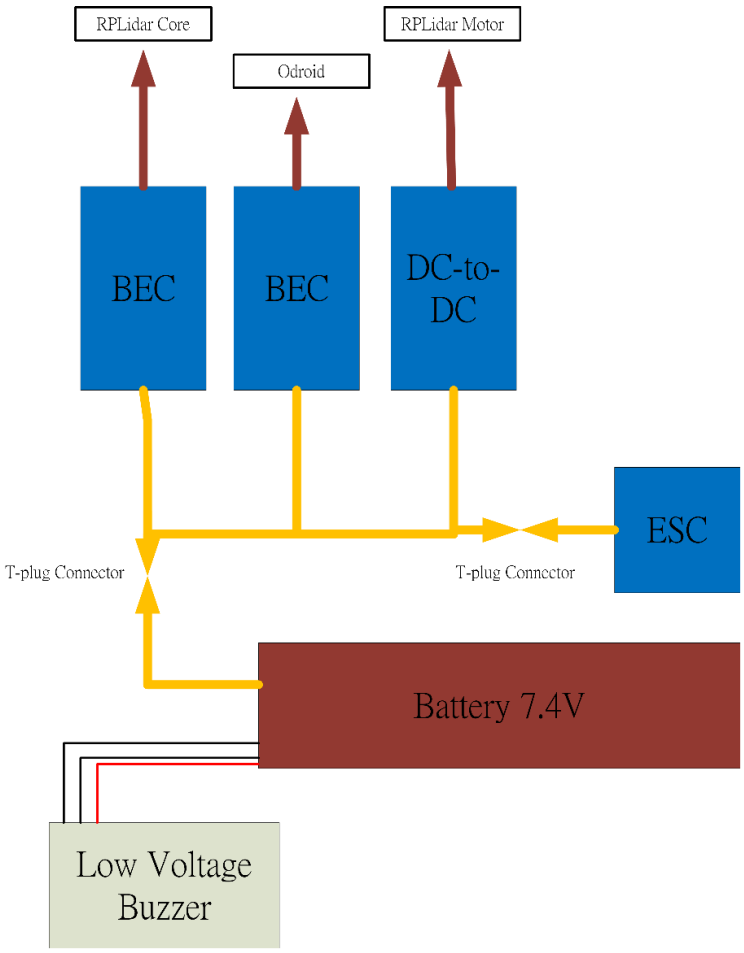
- Arduino Nano

Arduino is responsible for reading the GY85 sensor value. The communication between Nano and GY85 is though I2C.





- Power



- Motor

The ESC (Electronic speed controller) we choose here can use for both brushed motor and Brushless motor, and we using brushed motor in this project, so the connection between motor and ESC only two wire.

If you want use other ESC please notice that the ESC have:

1. Over current protection.
2. Middle point memory.

Initial setting

Computer setting (PC)

Windows

64bit RAM >= 8G
VMware Workstation Player 12 64bit (http://www.vmware.com/products/player/playerpro-evaluation.html)

VMWare Image：

https://drive.google.com/drive/folders/0B-yq9HXd5lDkRGNJUm9KR0h0VEU

Ubuntu (16.04)

Requirement :64bit RAM > 4G.
Please install Ubuntu 16.04 and ROS Kinetic - (Desktop-Full Install) (Follow the link: http://wiki.ros.org/kinetic/Installation/Ubuntu)

1. sudo apt-get install remmina synaptic gimp -y
2. sudo apt-get install ros-kinetic-navigation -y
3. sudo apt-get install ros-kinetic-hector-slam ros-kinetic-hector-mapping -y
4. sudo apt-get install arduino ros-kinetic-geographic-msgs -y
5. sudo apt-get install ros-kinetic-rosserial-arduino -y
6. sudo apt-get install ros-kinetic-rosserial -y
7. sudo apt-get install ros-kinetic-slam-gmapping -y
8. sudo apt-get install ros-kinetic-mrpt-slam ros-kinetic-mrpt-icp-slam-2d -y
9. Create your own catkin_ws (http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment#Create_a_ROS_Workspace)
10. cd catkin_ws/src
11. git clone https://github.com/cra-ros-pkg/robot_localization.git
12. git clone https://github.com/ros-planning/navigation_tutorials.git
13. git clone https://github.com/MAPIRlab/mapir-ros-pkgs.git
14. git clone https://github.com/Hypha-ROS/hypha-racecar
15. cd ..
16. catkin_make

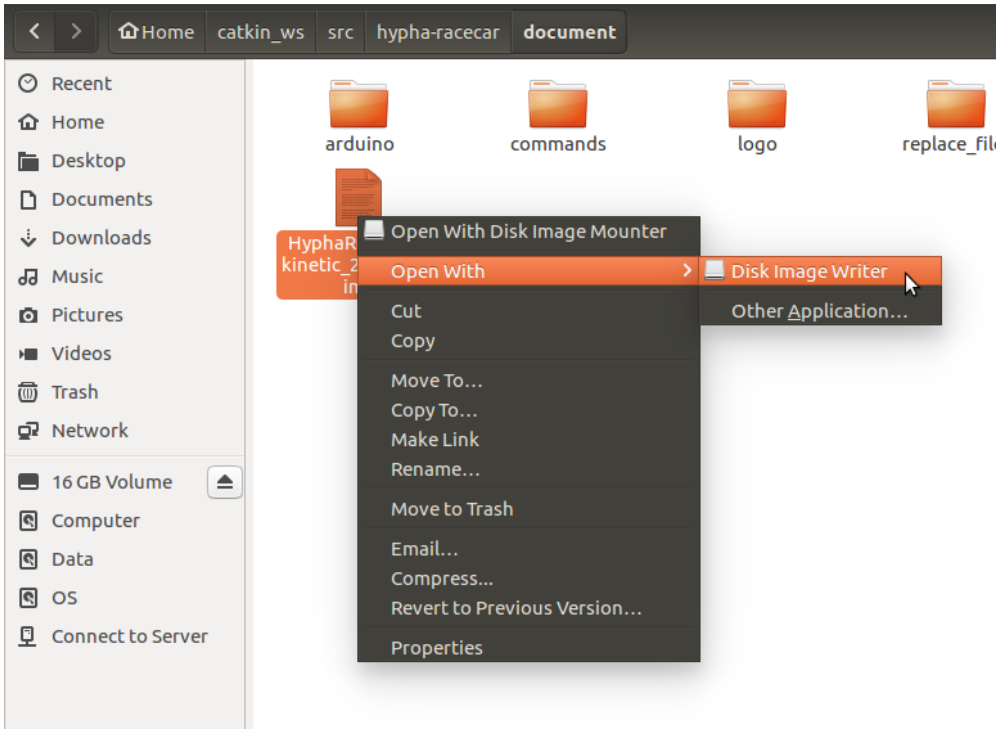
SD card

The Image file can download from here:

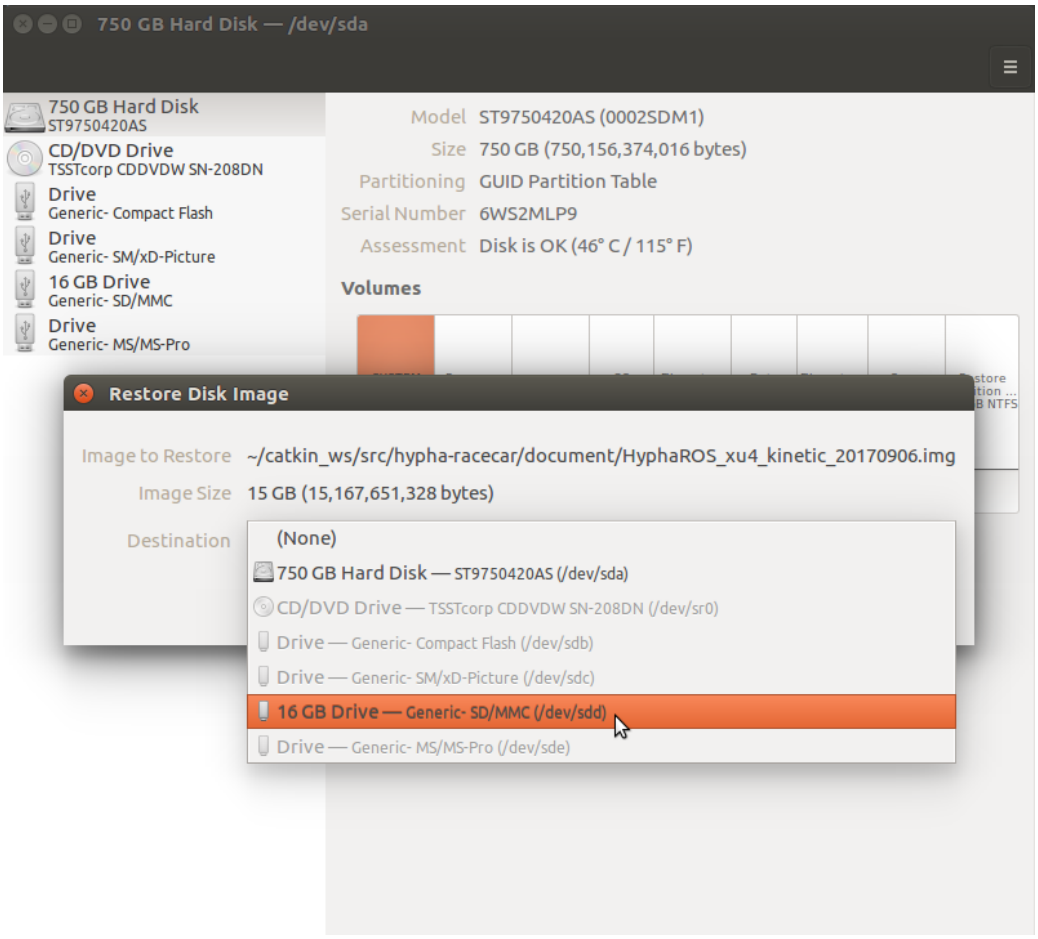
[Google drive] https://drive.google.com/open?id=0B7f-a3PiitSec3RyQWRTQVZ1NzA

1. Prepare a card reader which can read MicroSD card directly (Use adapter may damage the SD card).

2. Insert the Micro SD card to card reader and plug the card reader to computer.
3. Find the image file, right click and choose "Open with">"Disk Image Writer"



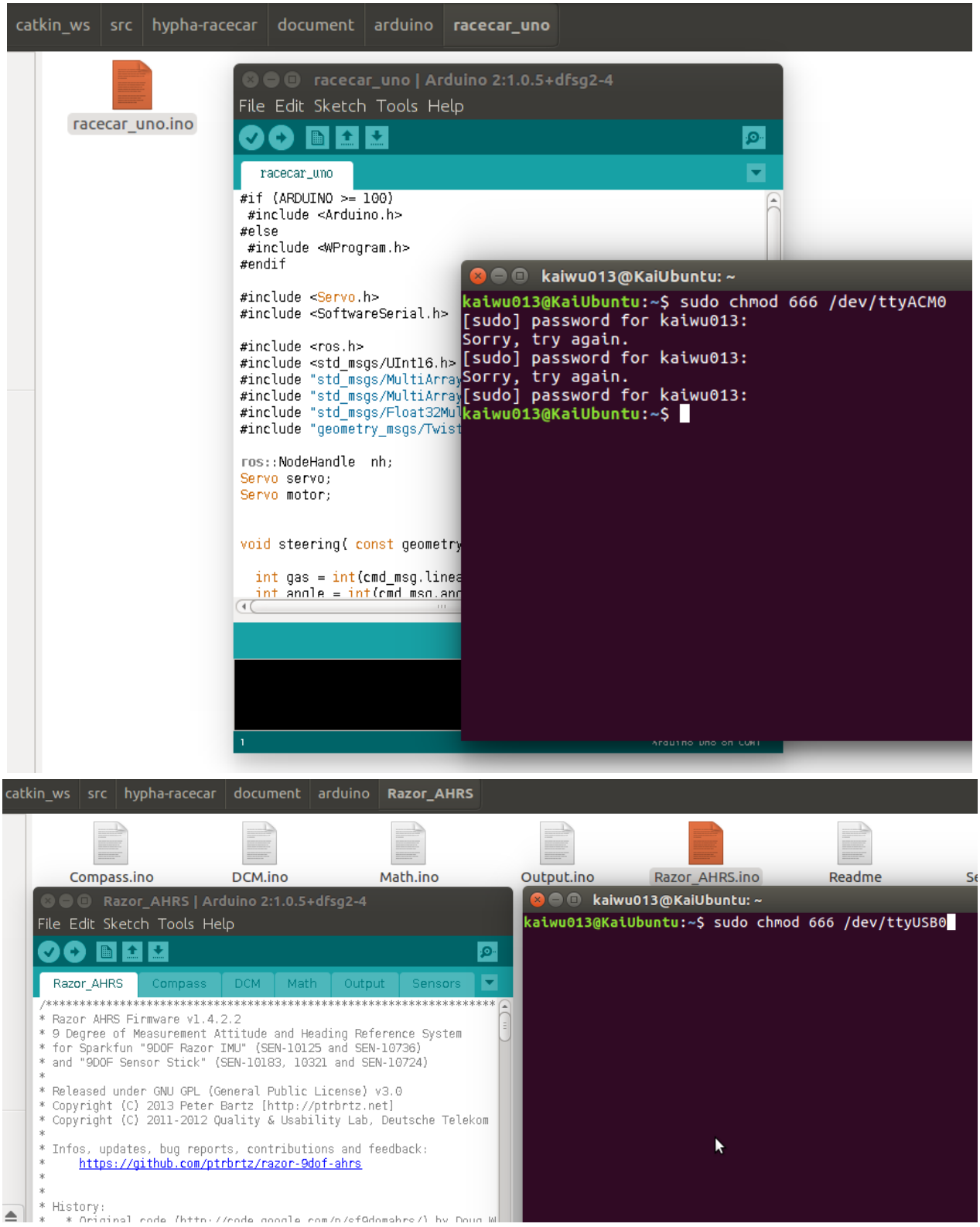
4. Choose the target SD card at "Destination"



5. "Start Restoring" and wait the process finished.
6. The Micro SD card is ready.

Arduino upload

1. Connect the Arduino to your computer.
2. Check your device name. (For this example is ttyACM0)
3. Give the authority to access the Arduino:
   $ Sudo chmod 666 /dev/ttyACM0
4. After the Initial computer setting the Arduino IDE should be ready.
   Start the Arduino IDE.
5. For Arduino Uno choose the racecar_uno.ino(in hypha-racecar/document/arduino/racecar_uno)
6. For Arduino Nano Choose the Razor_AHRS.ino(in hypha-racecar/document/arduino/Razor_AHRS)
7. Choose the right board (UNO) and port.
8. Upload.



Document replacement

The rf2o laser odometry need to do some modification to prevent the conflict topic be publish.

We have provide the code for correct this issue, which can find the file in “hypha-racecar/document/replace_files/rf2o” to replace the file in “mapir-ros-pkgs/src/rf2o_laser_odometry/src”

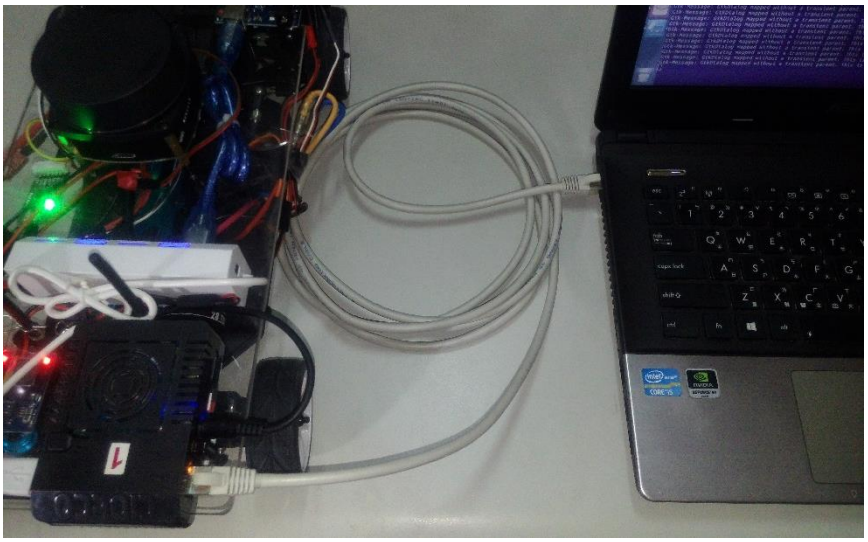
Power Connection

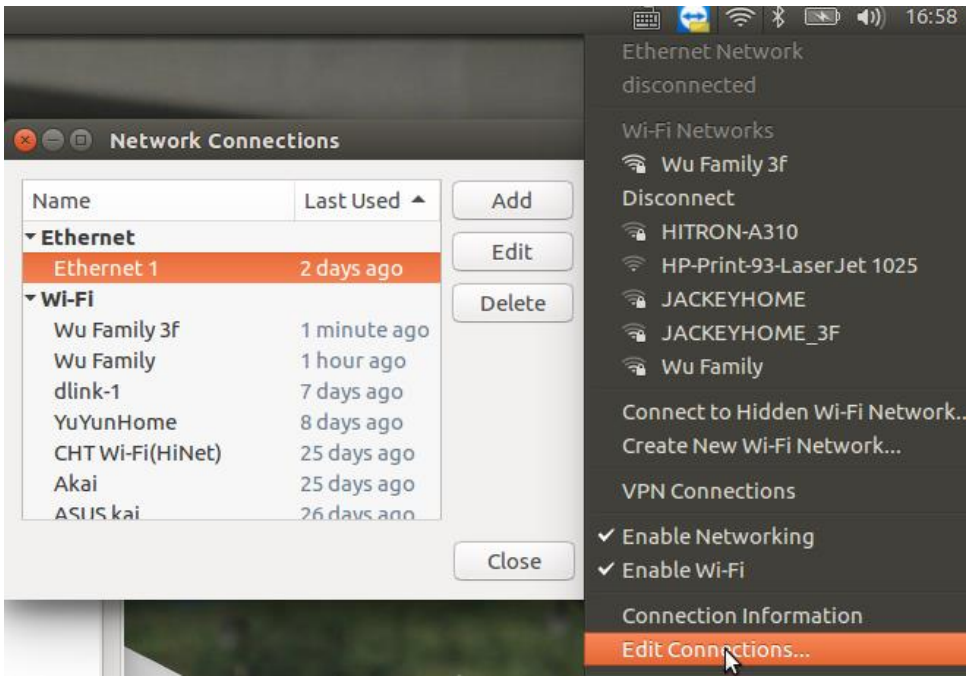
1. Unplug the Odroid power cable.(Important!!)
2. Check the ESC power is off
3. Connect the battery.
4. Plugging the Odroid power cable.

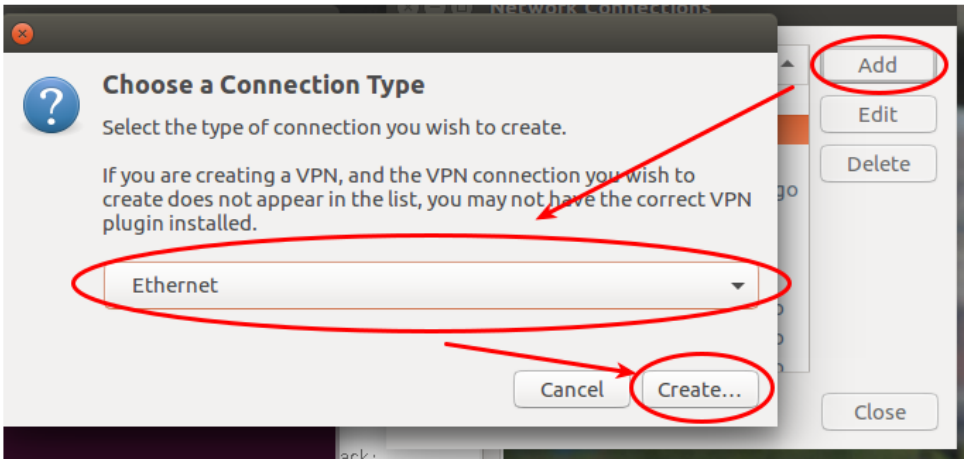
(Since the process of connect two T-plug connector may power up many time, this can damage the Odroid, therefor we need to unplug the Odroid until the power connection is stable.)

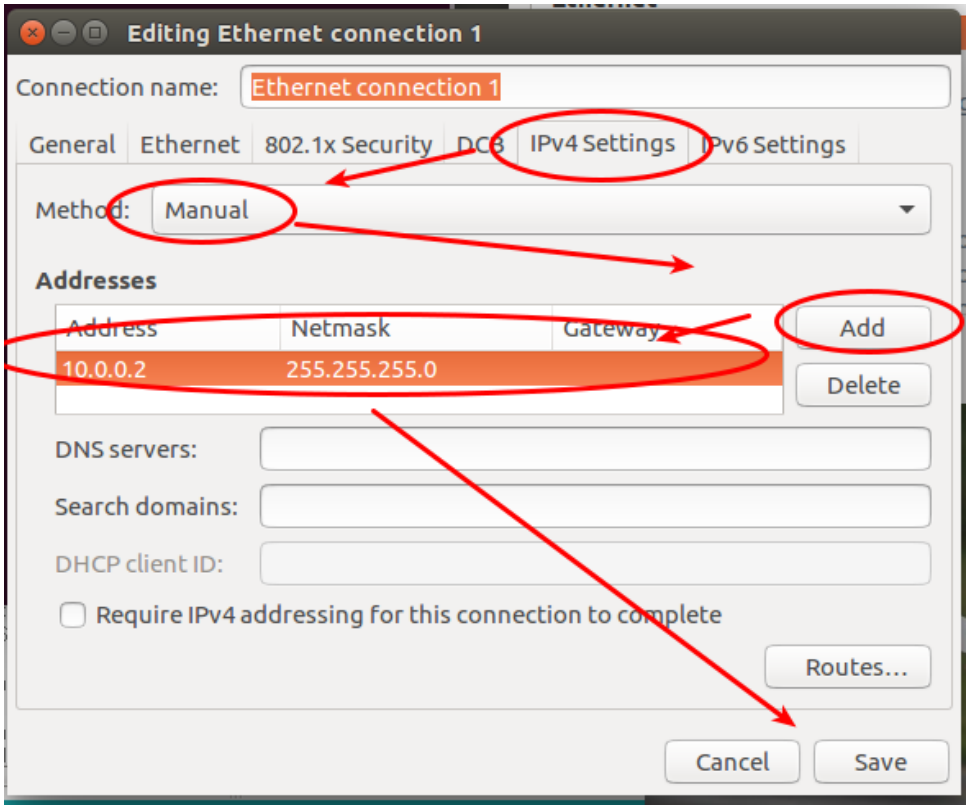
Internet connection

At beginning we need to use Ethernet connection









VNC Connection

Odroid already finished the setting, if you are using the image file provide previously.

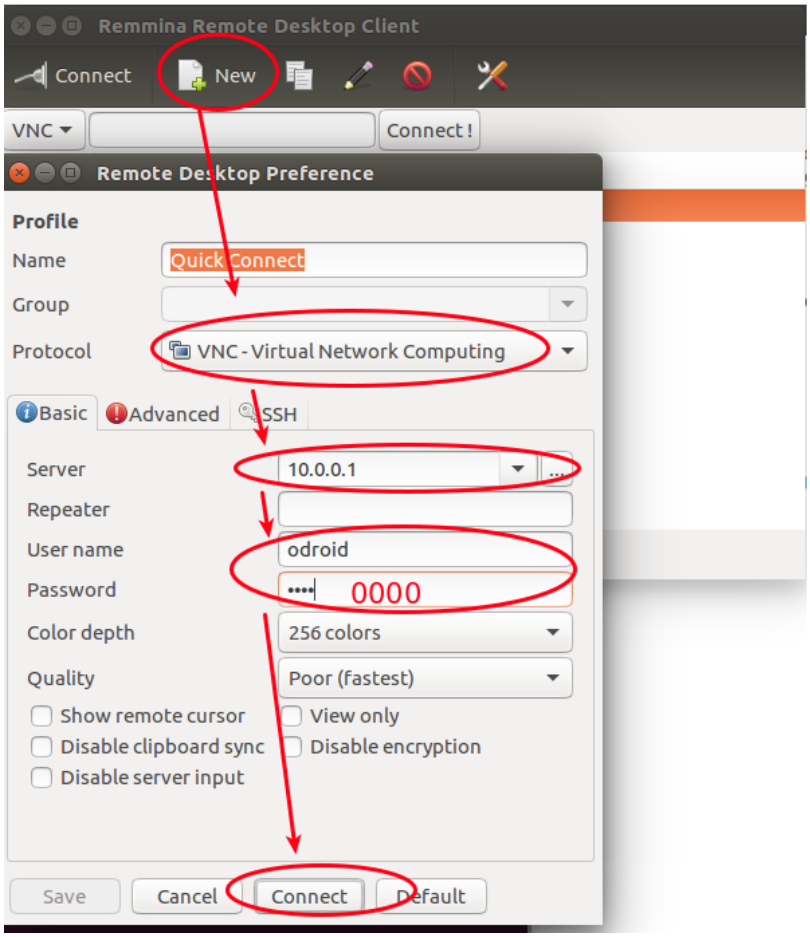
Start the Remmina for VNC connection (or you can use any other software can use VNC)

$ Remmina

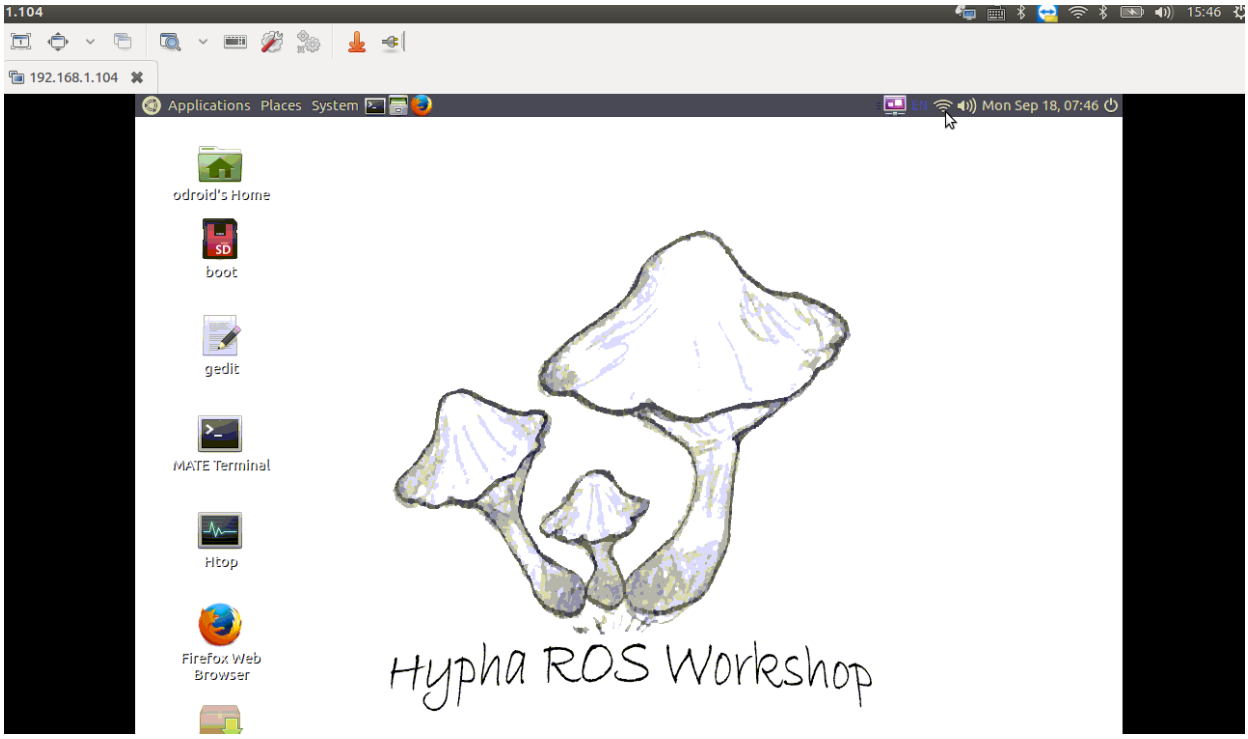
IP:10.0.0.1(if using Ethernet connection)(wireless connection use Odroid IP)

Password:0000

First connection is success, we can setup the wifi connection to WIFI router.



First connection is success, we can setup the wifi connection to WIFI router.



SSH connection

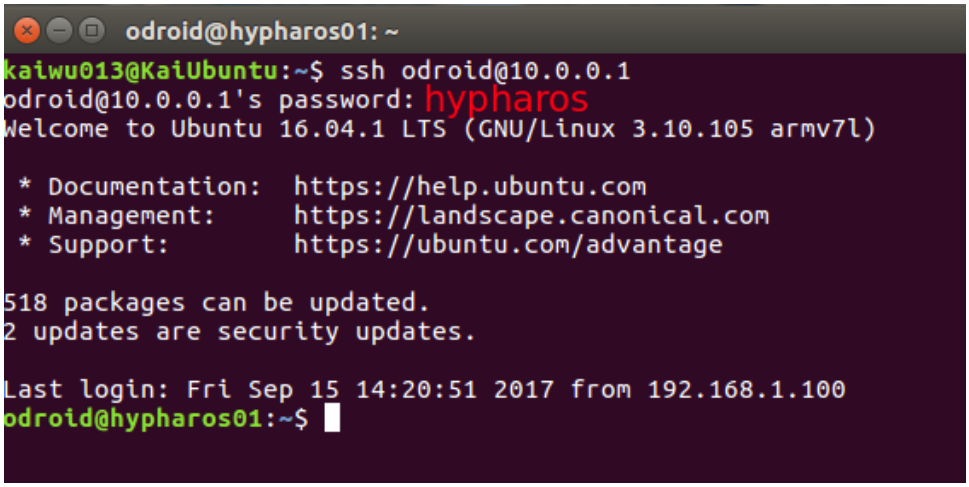
$ ssh odroid@10.0.0.1

(This is for Ethernet connection)

Wireless connection use Odroid IP

$ssh odroid@192.168.xxx.xxx

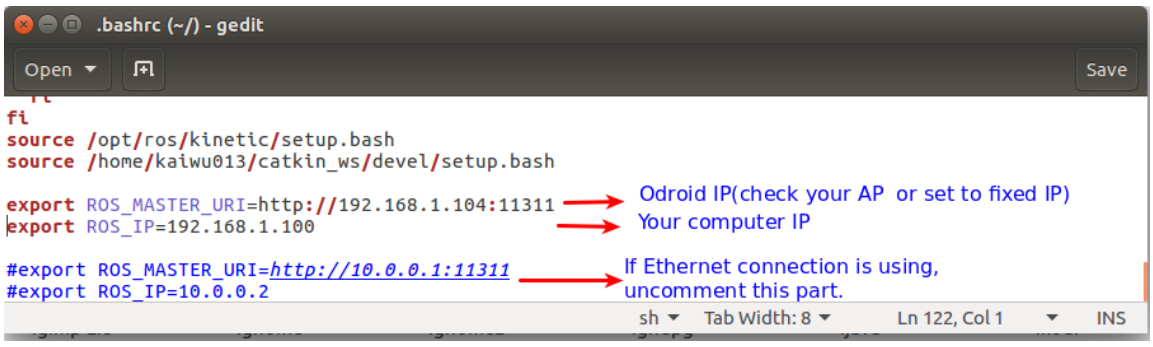
Password: hypharos



ROS multi-machine setting

Connection IP setting

At the Home folder press “ctrl+H” and find .bashrc file. Add the connection setting at the end of file. The setting should be depend on what type of connection you use.



Hostname

If you have more than one device, change hostname can help you not to confuse. Also if you want setting the router DHCP server to keep IP for specific device (Odroid) you need to change the hostnbame.

$sudo odroid-utility.sh

Change your hostname and then reboot.

Operation

Power Connection

1. Unplug the Odroid power cable. (Important!!)
2. Check the ESC power is off
3. Connect the battery.
4. Plugging the Odroid power cable.

(Since the process of connect two T-plug connector may power up many time, this can damage the Odroid, therefor we need to unplug the Odroid until the power connection is stable.)

Test Run

At this stage, first test the car and sensor; try to remote control the car to move also observer the real time Lidar and mapping result on the local computer Rviz interface.

RACECAR:

Check the multi-machine setting at RACECAR is correct. (feet the internet connection you ar using)

Open a new Terminal at local computer and use SSH connect to RACECAR (Odroid), and choose the mapping method you want to test.

$ roslaunch hypha_racecar Test_gmapping.launch

or

$ roslaunch hypha_racecar Test_icp_mapping.launch

After the launch file active, it will start the IMU calibration at beginning initial process, so do not touch or move the car.

Local computer:

Check local computer multi-machine setting is correct.

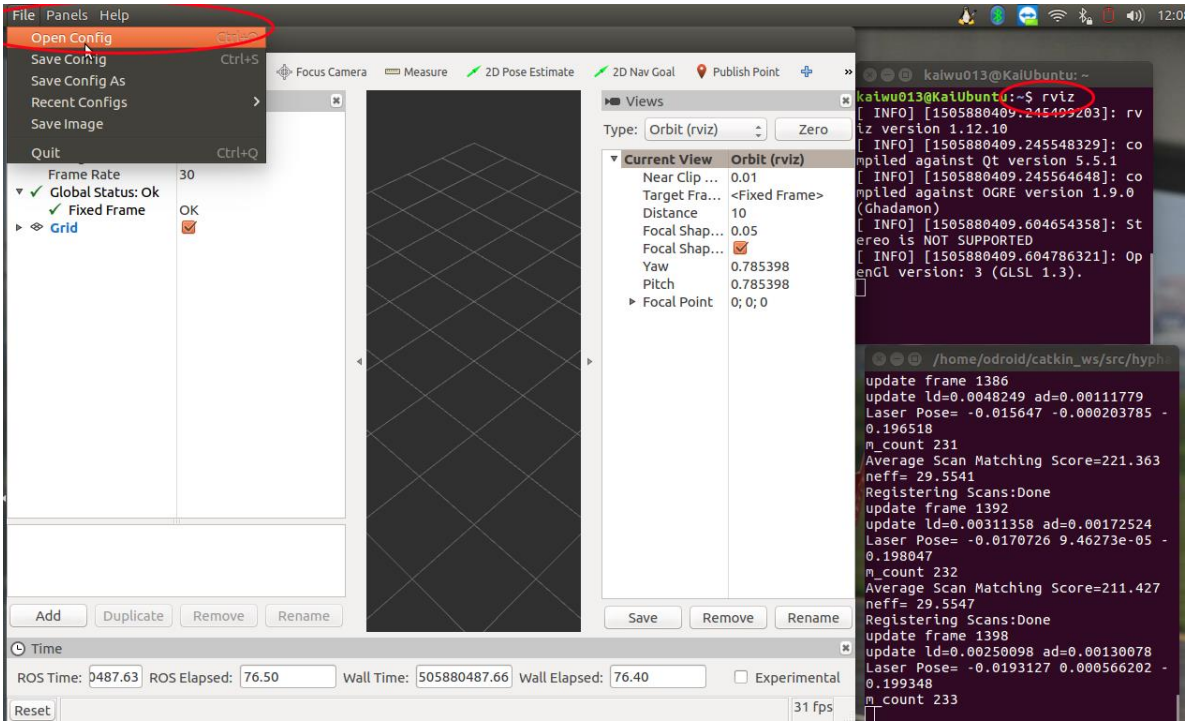
[**Start the rviz**]

Open a new Terminal

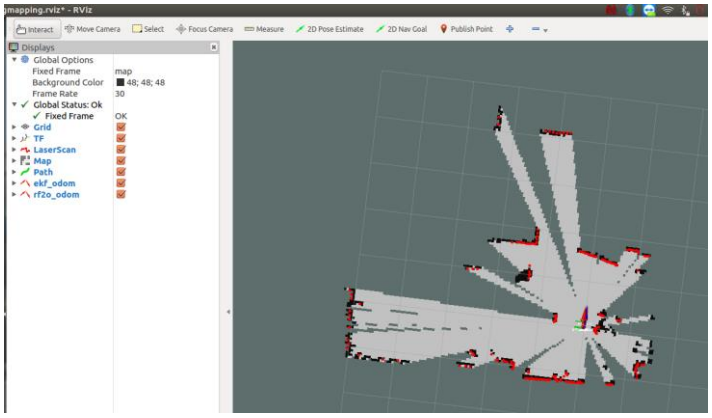
$ rviz

Choose File>Open config File

Depending on the Gmapping or Icp you use to choose the config file at "catkin_ws/src/hypha-racecar/rviz_cfg"



This process may need a few time, just wait until it show the map.

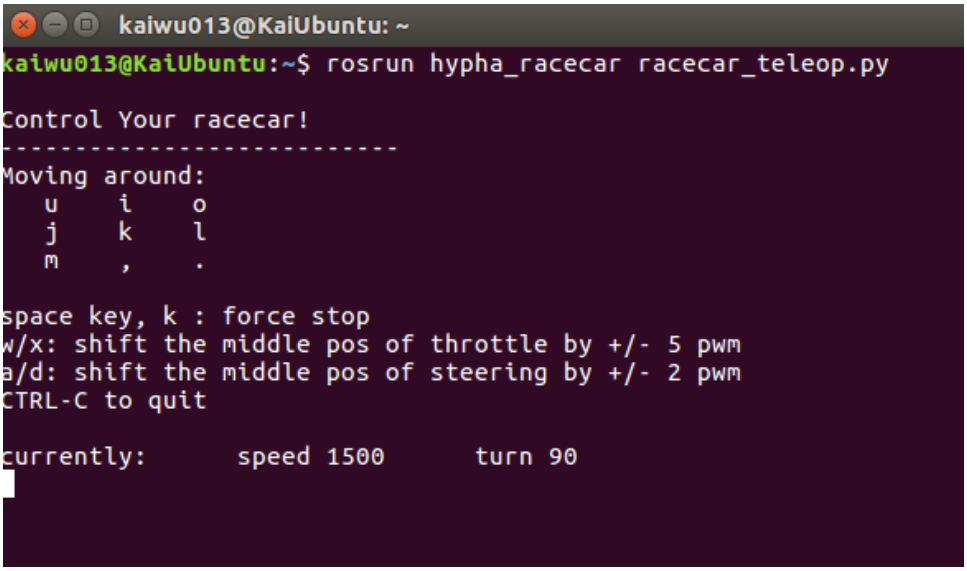


[Control car by keyboard]

Open a new Terminal.

$ rosrun hypha_racecar racecar_teleop.py

Then open the power of ESC.



Now we can use keyboard to control the RACECAR, and important thing is to adjust the middle point of servo and ESC.

Use W/X to change the middle point of ESC(throttle), the middle point should let car stop(press key “K”) and can move forward (key “i”) and move back word (key” , ”).

For Servo (steering) use a/d to adjust the middle point so that the car can perfectly straight forward.

Build map

Currently we have two way to build up the map:

- Odroid on board real-time mapping.(Test_gmapping.launch OR Test_icp_mapping.launch)
- Recall all the data on the car, and do the mapping at computer.

The first method operation is just like pervious section “test Run” , the mapping algorithm is running at Odroid. The result is not good then running at computer, so we will suggestion using second method, which is running the algorithm on your computer. The following step focus on second method.

mapping :offline, Local computer

odroid: RACECAR_bag_record.launch

Local computer: RACECAR_bag_icp.launch OR RACECAR_bag_gmapping.launch

Record

1. At **RACECAR** launch *RACECAR_bag_record.launch*

$ roslaunch hypha_racecar RACECAR_bag_record.launch

2. At **local computer** start the racecar_teleop

$ rosrun hypha_racecar racecar_teleop.py

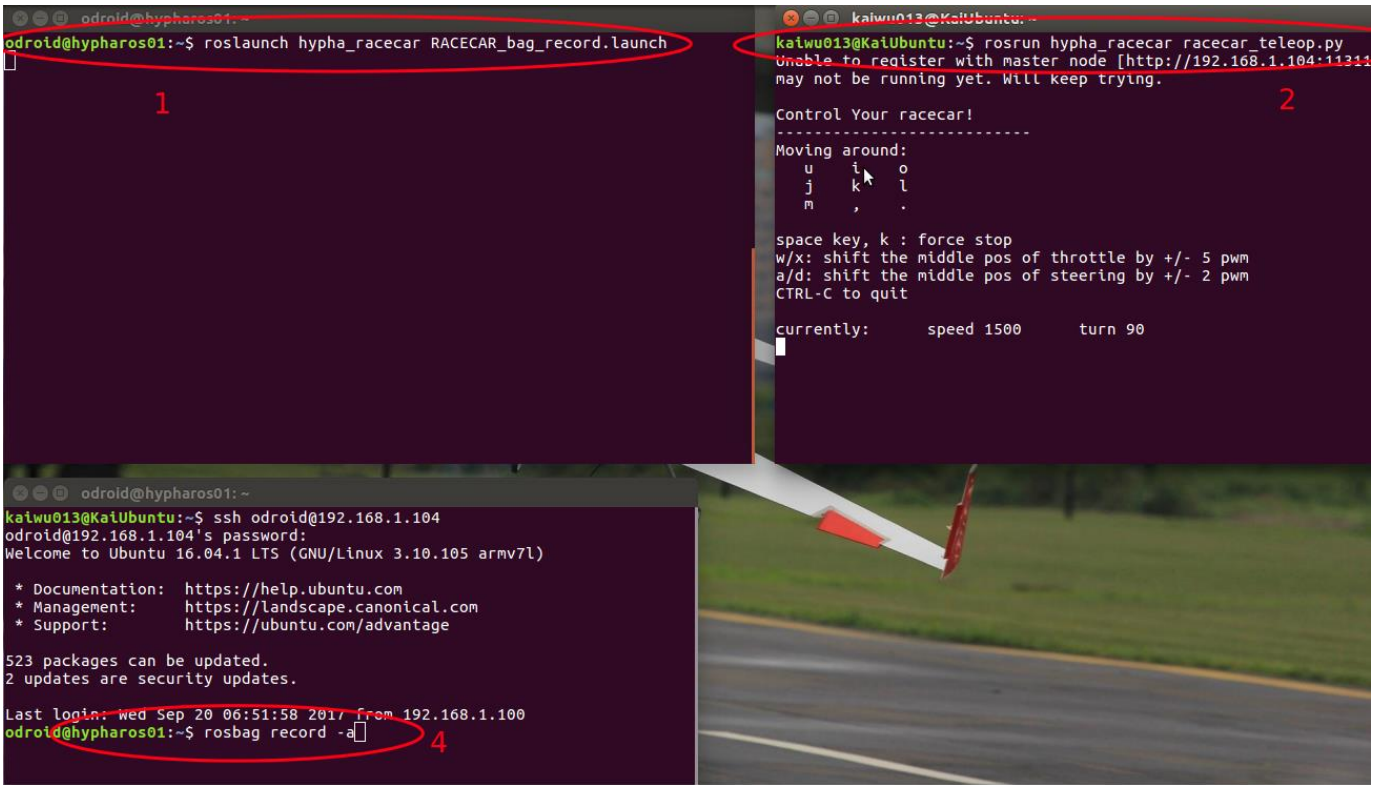
3. Turn On the ESC power.

4. At **RACECAR** start record the data.

$rosbag record –a

This will start save all the publish topic into a .bag file.

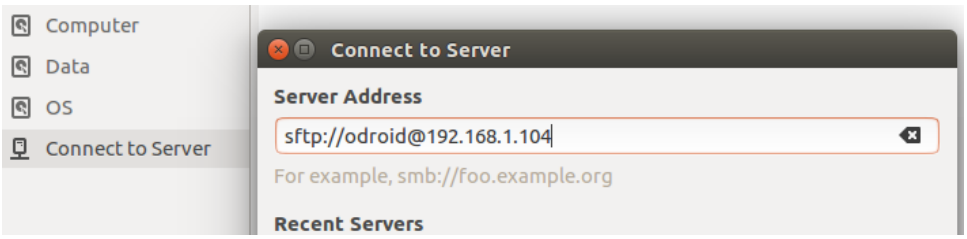
5. Then you can control your RACECAR to run on the road you want it to run.
6. When all finished stop the record. If you start the “rosbag record” at “HOME” folder the .bag file will be there.



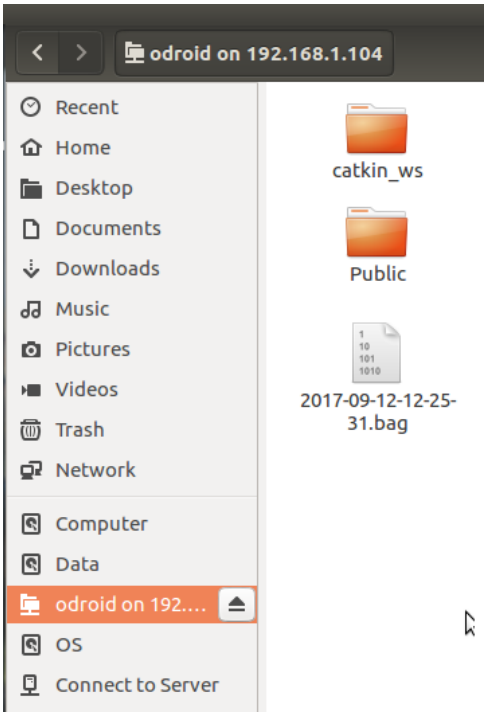
Build map

1. Copy the bag file from Odroid to your computer.

(We can use server to take file from Odroid. The IP should change to your Odroid IP)



Troubleshooting:
When start the launch file if see the error: cannot find the device…
If is relate to the GY85 or Arduino, Just simply close the arduino and nano power on the hub, and turn on it again.



2. Modify the launch file "RACECAR_bag_gmapping.launch" or "RACECAR_bag_icp.launch".
Let the bag file pointing to your bag file.

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

<launch>
    <arg name="use_rviz" default="true" />
    <arg name="use_rf2o" default="true" />
    <arg name="use_ekf"  default="true" />

    <!--PLAY BAG FILE-->
    <param name="use_sim_time" value="true" />
    <node pkg="rosbag" type="play" name="player" output="screen" args="$(find hypha_racecar)/bags/imu_laser_tf_001.bag --clock"/>

```

3. Modify .bashrc
Comment all the multi-machine setting.
Open a new terminal.

4. Launch the launch file "RACECAR_bag_gmapping.launch" or "RACECAR_bag_icp.launch".

$roslaunch hypha_racecar RACECAR_bag_gmapping.launch

Or

$roslaunch hypha_racecar RACECAR_bag_icp.launch

5. Wait the bag file is finished the play, the map should be show on the RVIZ.

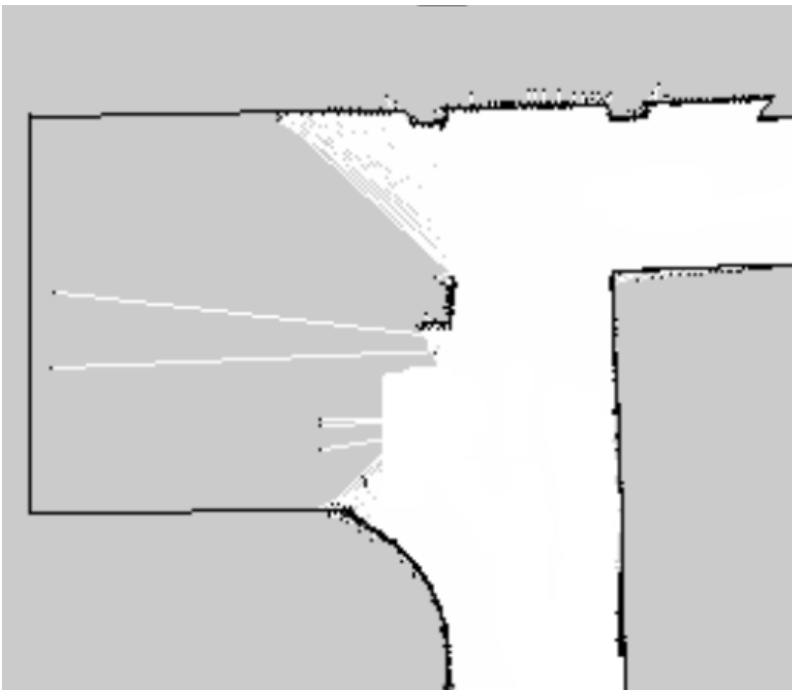
At this stage we can save the map.

$ rosrun map_server map_saver

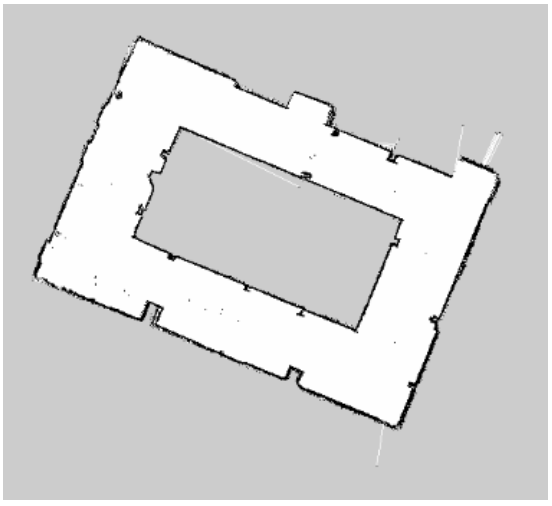
Edit map

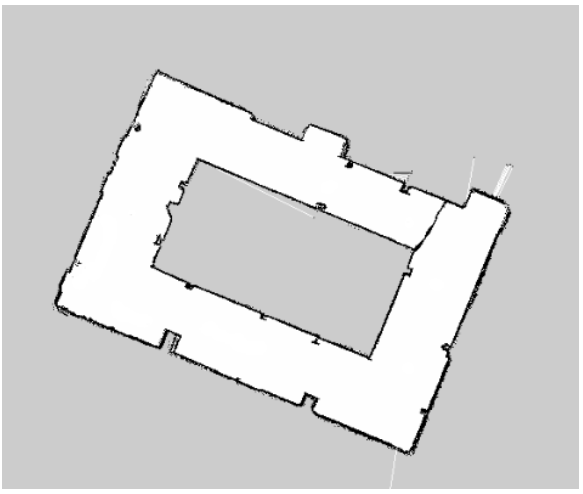
The map for navigation should be do some modify, we need to close the edge and add an end line.

Close the edge: The map may not be complete. We need to manually do some modify to let the edge become continues line.



End line: Add a line at where you want to start and end. This can avoid the path planning from planning a backsword path.





GIMP is a good software to doing this job, it can support png and pgm file.

Run

[Setting Launch File]

Map

Setting the launch file, let the map path point to the new map.

Edit “hypha-racecar/launch/RACECAR_amcl_nav.launch”

```
<!-- Map server -->
<node name="map_server" pkg="map_server" type="map_server" args="$(find hypha_racecar)/map/test/map.yaml"/>
```

Speed and servo middle point

The running speed can adjust by modify “baseSpeed” in same launch file, you can try different speed to run. And the "baseAngle" is to adjust the middle point of servo, the values is depend on what you turning at Test Run section.

```
<!-- ESC -->
<param name="baseSpeed" value="1425"/> <!-- pwm for motor constant speed, 1480: stop, 1440: ~0.5m/s, 1430: ~1.5m/s
-->
  <!-- Servo -->
<param name="baseAngle" value="90.0"/> <!-- the middle pos of servo, for tt02: 87, for hsp: ? -->
```

Those modify should change on your RACECAR (Odroid).

[Multi-machine setting]

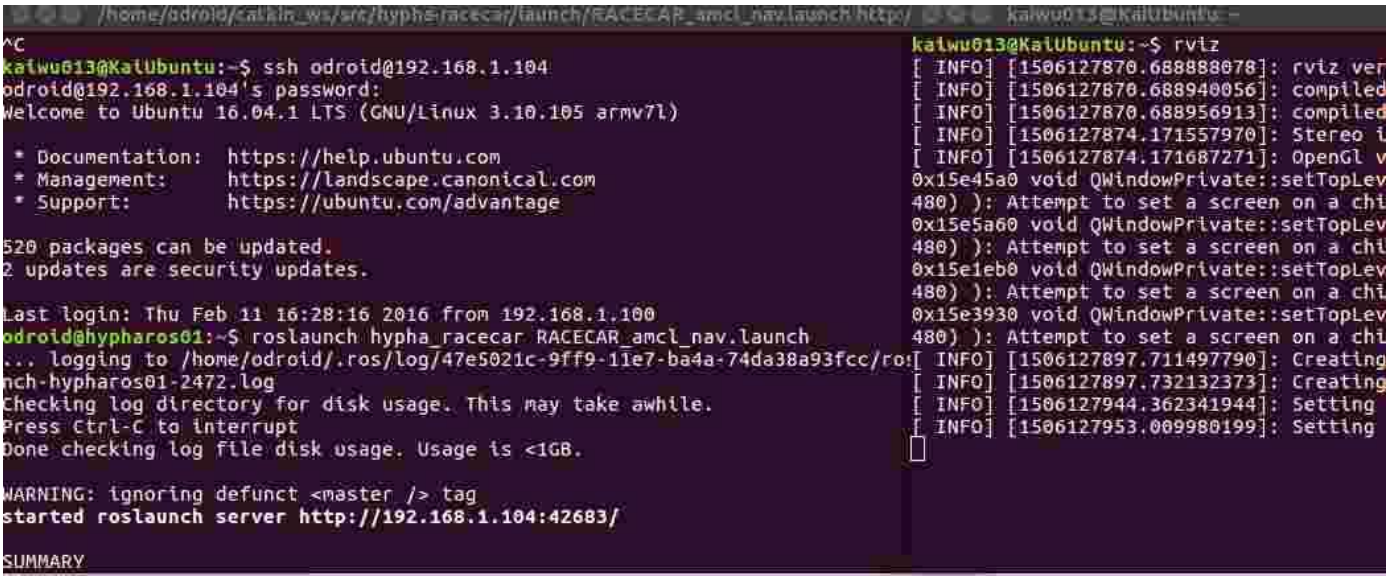
Renumber to modify your multi-machine setting. (when build map on local computer we comment those setting.)

[Launch]

Launch “RACECAR_amcl_nav.launch” on RACECAR.

Use SSH connect to the RACECAR (Odroid)

$ roslaunch hypha_racecar RACECAR_amcl_nav.launch



[rviz]

Start a new Terminal. Run Rviz.

$ rviz

At Rviz interface open configure file. “hypha-racecar/rviz_cfg/ hypha_amcl.rviz”

Give **initial position** by press “2D Pose Estimate”, and then click on map where the RACECAR locaton is.

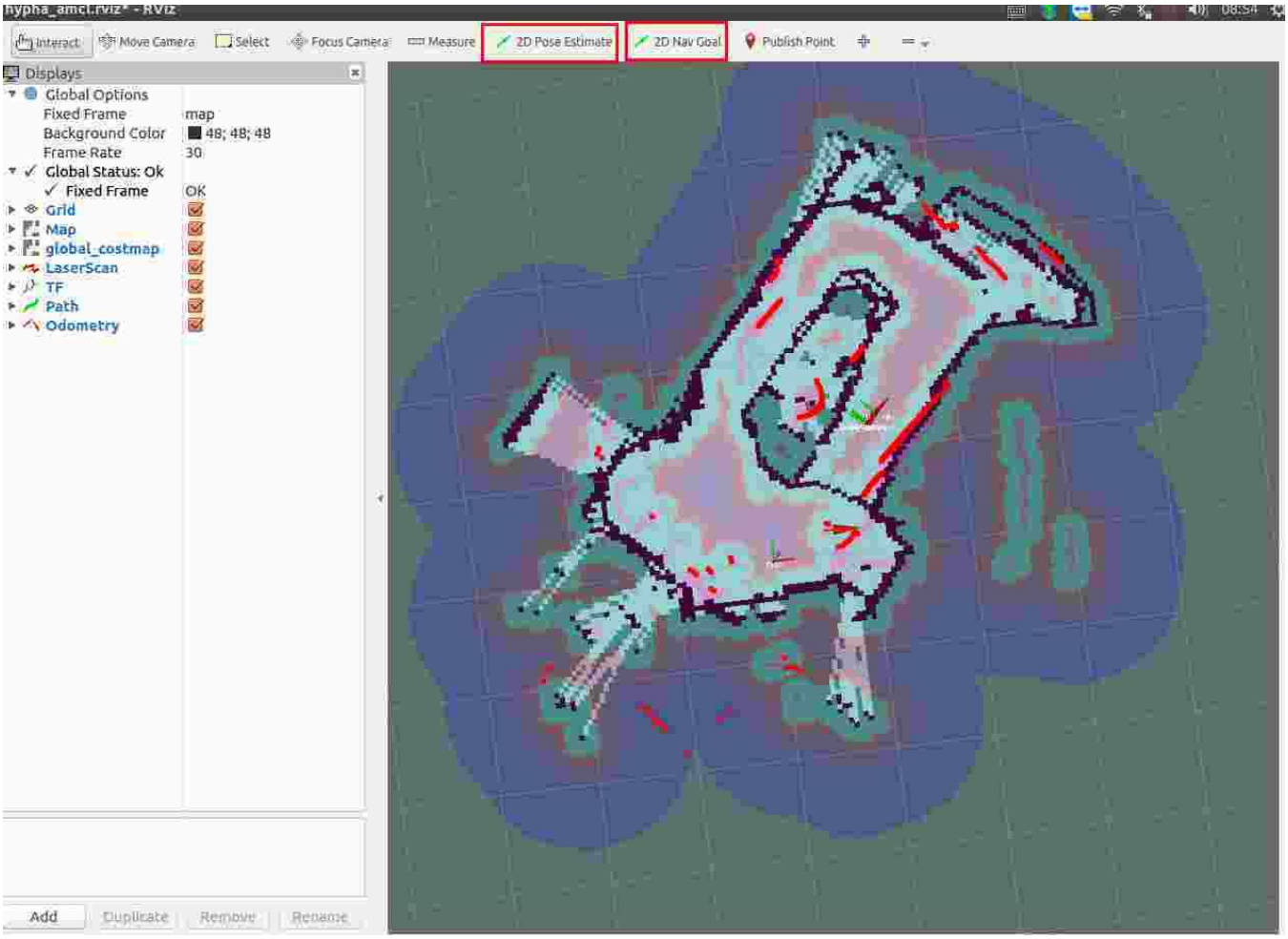
Then **give it a goal**.

Press “2D Pose Estimate”, and click on map to tell the RACECAR where the final goal is.

If the path planning is success, you can see a green path should on map, so the RACECAR is ready.

[Open the power of ESC]

Now you can turn on the power of ESC so the car will start running.



Note:

When choosing the place or yard layout, there has some limit.

The Runaway cannot too monotonous, you can put some object randomly, otherwise the RACECAR cannot identify where its location on the map is.

If the Runaway is too width, your RACECAR can not run success try to slowdown the speed of car.

Contact

GitHub: https://github.com/Hypha-ROS/hypha-racecar

FB Page: https://www.facebook.com/HyphaROS/

Website: https://hypharosworkshop.wordpress.com/

Youtube: https://www.youtube.com/watch?v=jgS6gpmWPoY (3 m/s)

Youku: http://i.youku.com/hypharos

E-mail: hypha.ros@gmail.com

Author: KaiChun, Wu (kai.wu013@gmail.com)

HaoChih, LIN

September 25, 2017