ROSYZ-01B Robot Platform

USER MANUAL

Rev: 3.0(Dec 2020)

SHENZHEN YZ ROBOT CO. LTD

§1 Brief Introduction

§1.1 ROSYZ-01B Basic Information

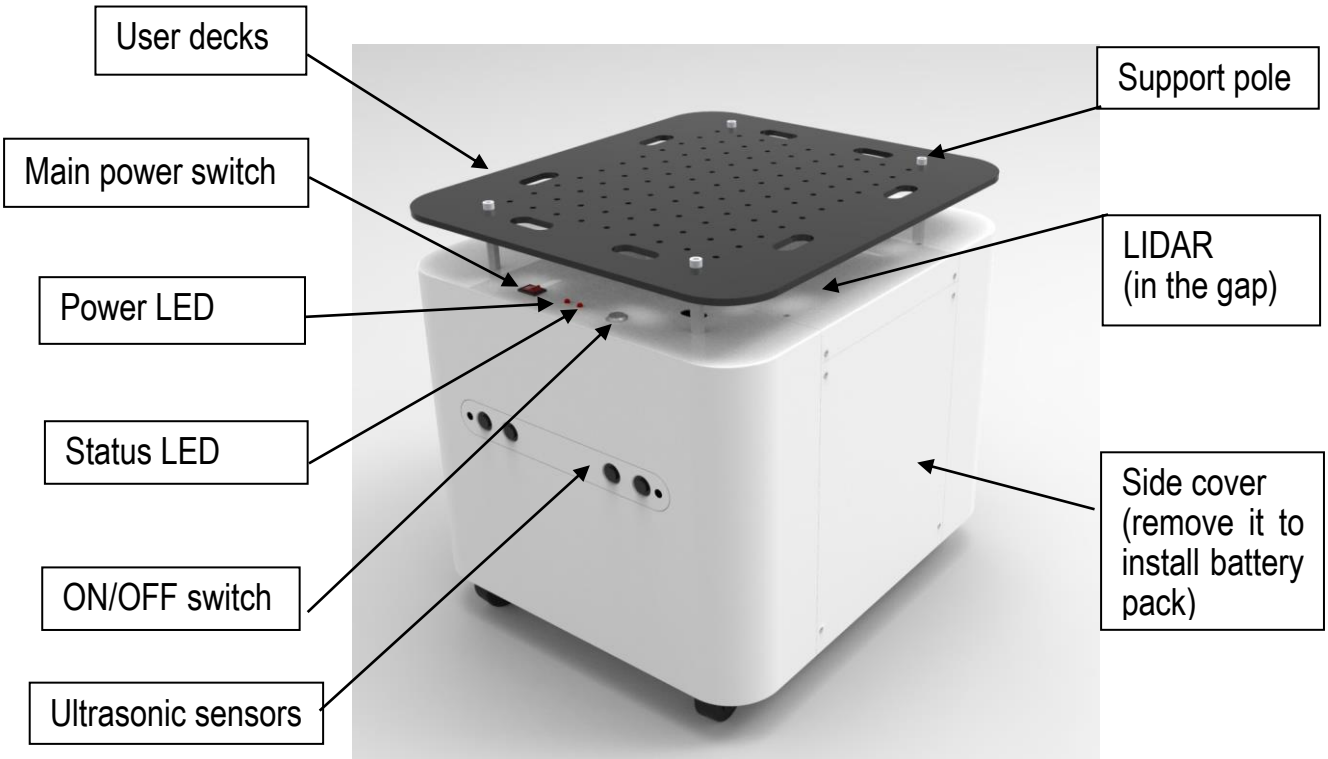
ROSYZ-01B robot platform is a two wheel differential and large load robot motion chassis platform based on ROS architecture. It is very suitable for college students some enterprise R & D engineers.

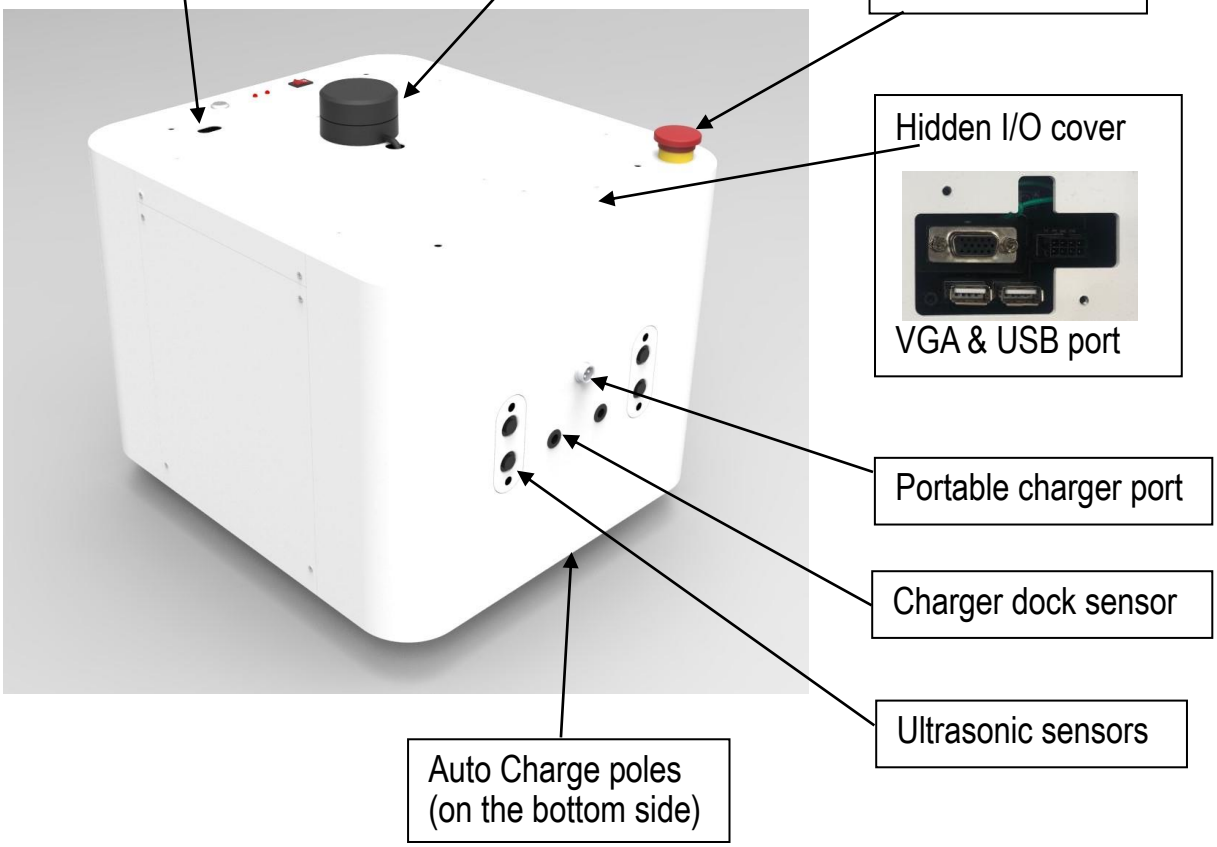
The chassis platform is an integrated hub motor with high efficiency and high load. The platform can load up to 50KG, and the maximum walking speed can reach 1.0 meters per second. ROSYZ-01B's built-in DCDC power conversion module can provide four different voltages of 5V, 12V and 16V, which can basically solve the power supply problems of various sensors that the robot needs to carry. In order to facilitate developers to do in-depth research, the ROS platform motion control board and DCDC power board circuit diagram and detailed interface drawings we also provide.

This YZ-01B ROS platform has built-in Intel i5 CPU industrial personal computer, installed Ubuntu 16.04 O/S and ROS Kinetic packages. YZ-01B also has ultrasonic wave sensors which can provide basic obstacle avoidance. As optional parts, we provide auto charging dock and LIDAR for real applications.

At the same time, like other ROS platform robots, ROSYZ-01B provides open source basic ROS application examples and basic motion driven node programs that allow developers to use it very easily.

§1.2 Main parts of ROSYZ-01B





Status LED indictors:

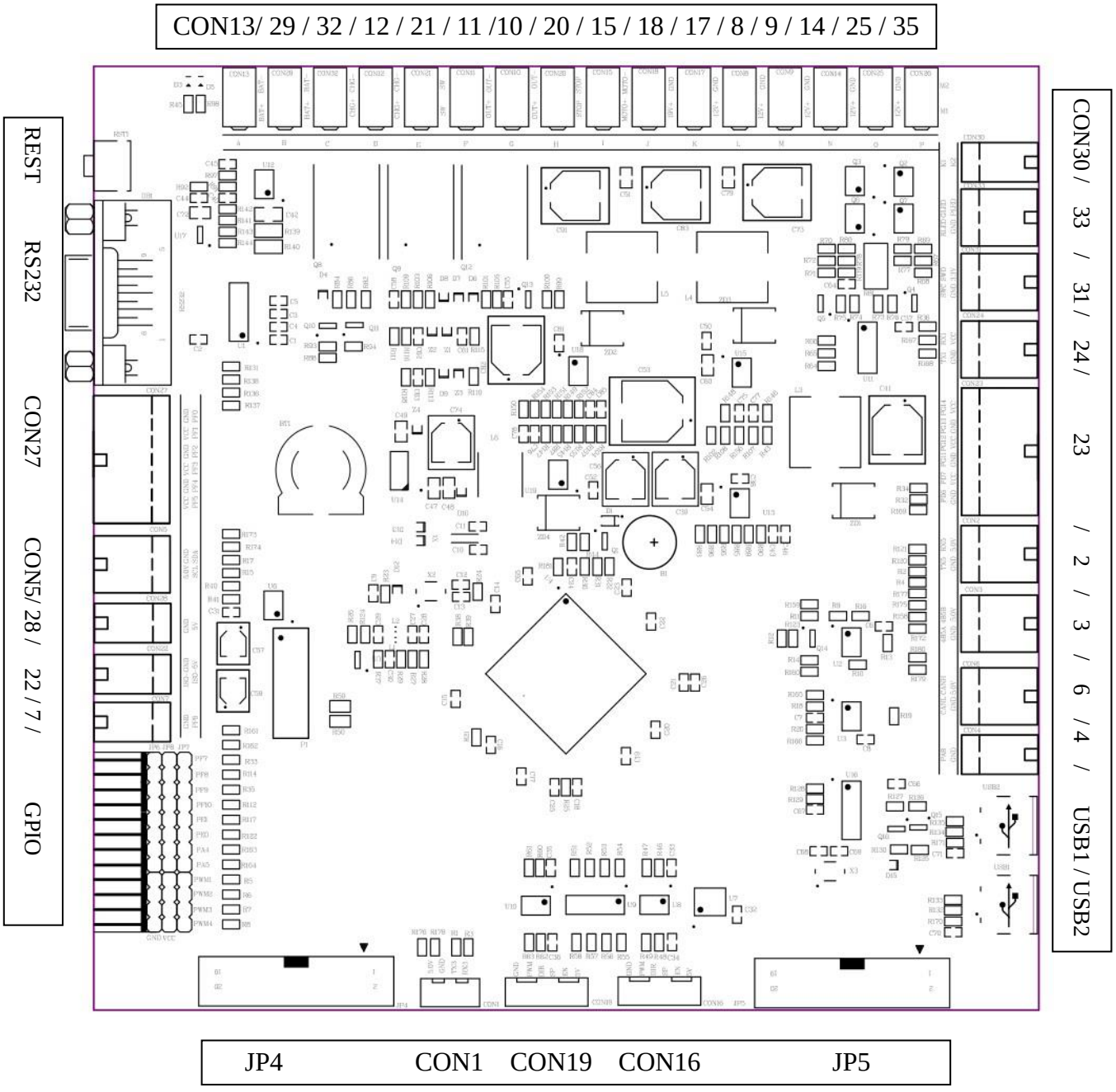
Power status: RED for POWER ON

Operating status: RED solid: Charging now

GREEN flashing: normal working

RED flashing: low battery

§1.3 ROSYZ-01B Control Board connectors definition



Connectors Definition:

Power ports

CON13	Battery input(16V)
CON29	Battery input (16V)
CON32	Charger input
CON12	Charger input
CON21	Power switch
CON10	Battery output (16V)
CON11	battery output (16V)
CON20	Emergency stop switch
CON15	Motor power output
CON18	+16V output
CON17	+12V output
CON8	+12V output
CON9	+12V output
CON14	+12V output
CON25	+12V output
CON26	+12V output (isolated)
CON28	+5V output
CON22	+5V output (isolated)

Communication and others

RS232 UART port

CON27: Front ultrasonic sensors

CON5: NA

CON7:NA

CON30: ON/OFF button

CON33: LEDs display board

CON31: SWD

CON24 : NA

CON23: Back ultrasonic sensors

CON2: TTL RS232

CON3 :NA

CON6: NA

CON4: NA

USB1: USB debug

USB2:NA

JP4: NA

JP5: NA

CON1: NA

CON19: Left wheel driver

CON16: Right wheel driver

JP6, JP7,JP8: GPIO

JP6: GND

JP8: VCC

JP7:

PF7: power on signal

PF8:NA

PF9,PF10: Auto charging guide signal

PE0,PE1: NA

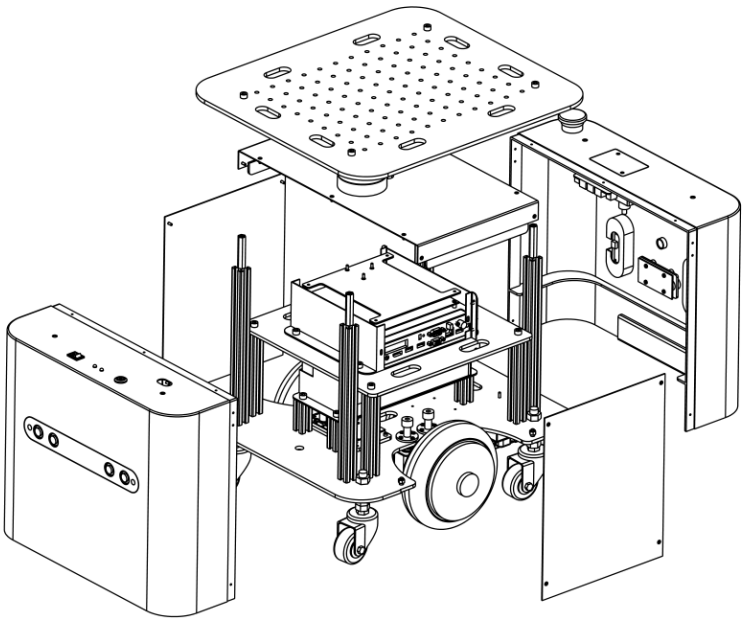
PA4,PA5:NA

PWM1,PWM2: NA

PWM3:NA

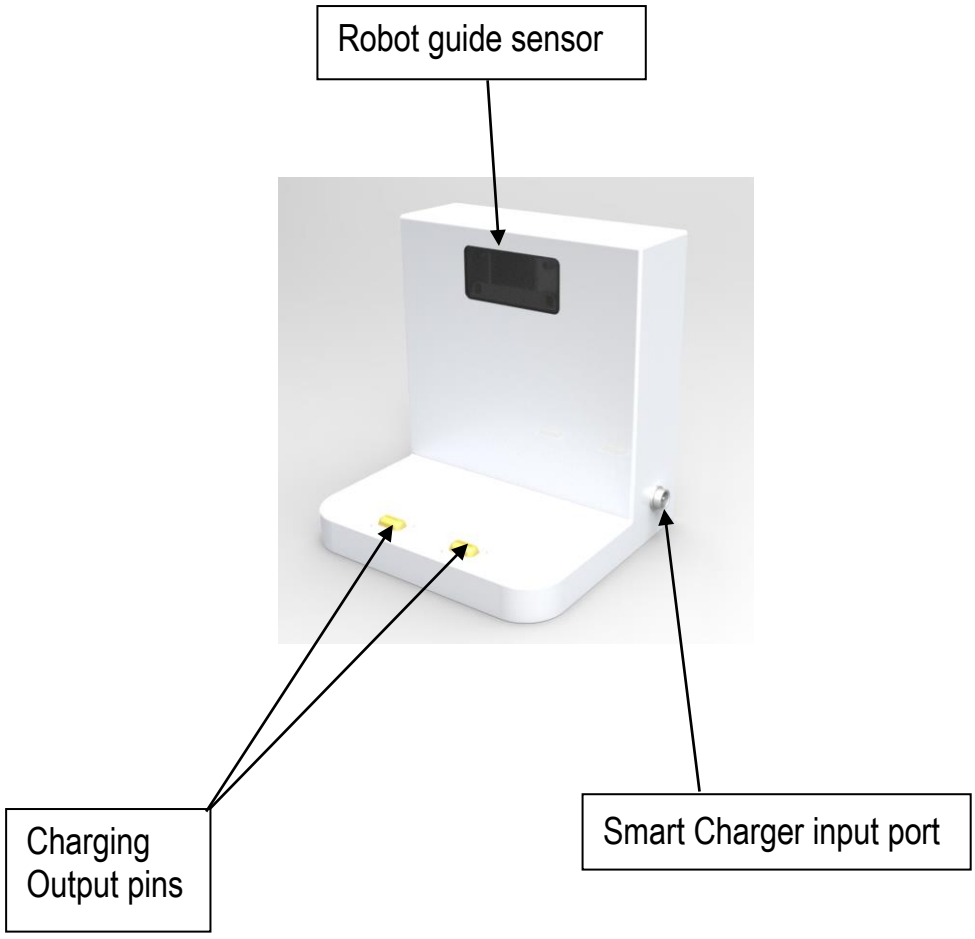
PWM4:NA

Explosion structure diagram



§1.4 Auto Charging dock (Optional part)

ROSYZ-01 Auto charging dock profile



Installation guide： Please put this charging dock against a flat wall, make sure the front of charging dock is empty enough (> 2.0 meters @ 120°), without any obstacle in this area.

§1.5 Software installed on the robot computer

ROSYZ-01 has Intel i5 CPU personal computer and pre-installed following items：

1. LINUX ： UBUNTU 16.04, **user name: robot; password: 1**
2. ROS basic ： KINETIC FULL-DESK
3. ROS working space folder: ~/ws/, including as folloing

3.1 build : created when catkin_make

3.2 devel : created when catkin_make

3.3 install : private localization and navigation packs from YZ robot Tech

3.4 src : robot platform driving, all sensors driving and YZ demo codes

3.4.1 YZBOT_STM32CONNECT ： platform driving code

3.4.2 ROBOT_MSGS ： platform message code

3.4.3 YZBOT_DESCRIPTION ： platform model URDF

3.4.4 ROS-NAVIGATION : ROS standard navigation packages

3.4.5 YZBOT_NAV ： YZ demo codes

3.4.5.1 launch ：

rs_amcl_startup.launch ： run standard ROS amcl node

rs_movebase_startup.launch ： run standard ROS navigation node

yzbot_joystick_startup.launch ： run joystick robot drive(YZ provided) node

yzbot_keyboard_teleop.launch ： run keyboard robot drive node

yzbot_loadmodel.launch ： load robot description

Laser/lidar_laser.launch : run LIDAR (from EAI) node

Laser/lslidar_laser : run LIDAR (from LEISHEN) node

yzbot_dmcl_startuo.launch ： run YZ private localization node

yzbot_gmapping_startuo.launch ： run gmapping node

yzbot_movebase_startup.launch ： run YZ private navigation node

yzbot_stm32control.launch ： run robot platform node

yzbot_createmap.launch ： launch all necessary nodes to create map

yzbot_navigation_byAndroid.launch ： for remote Android application usage

yzbot_navigation_startup.launch ： launch all YZ private navigation nodes

rs_navigation_startup.launch ： launch all standard ROS navigation nodes

3.4.5.2 maps ： maps created

3.4.5.3 nodes ： navigation demo script (PYTHON)

3.4.5.4 config ： config file for navigations

3.5 YZBOT_SENSORS ： all sensors drive code

3.5.1 YDLIDAR ： LIDAR from EAI

3.5.2 MIIBOO_IMU: IMU form MIIBOO

3.5.3 ASTRA_CAMERA : RGBD camera from ASTRA

3.5.4 JOY : joystick from BT

3.5.5 LS01BV2: LIDAR from LEISHEN

3.6 YZBOT_MSATER_CONTROL : for remote Android application usage

§1.6 ROSYZ-01 sensors and optional part listed:

ROSYZ-01 pre-installed sensors list (include optional parts)

1. Ultrasonic wave module
2. Wheel encoder
3. LIDAR (Optional part, required by user)
4. Auto charging dock (Optional part, required by user)
5. IMU module (Optional part, required by user)
6. Wireless keyboard and mouse (Optional part, required by user)
7. Wireless joystick (Optional part, required by user)
8. Mobile VGA monitor (Optional part, required by user)
9. RGBD camera (Optional part, required by user)

§1.7 ROSYZ-01B default packing list

Item	qty	unit	Remark
Main platform	1	set	No battery and decks
User decks	1	pc	
Support pole	4	pc	
Battery Charger	1	pc	P/N: G168180080
AC power cord	1	pc	Europe type plug
M4 Screw	5	pc	
Screwdriver	1	pc	For M4 M6 screw
ROS SDK and Demo	1	set	Free download
User Manual	1	pc	PDF format
LiFePO4 BATTERY	1	pc	16V21Ah

Optional parts list

LIDAR	1	pc	EAI G4
Auto-charge dock	1	Set	Green-Digial
9DOF IMU	1	pc	miiboo
RGBD camera	1	pc	Astra pro
Keyboard & mouse	1	set	rapoo
Mobile 7’ VGA monitor		pc	DC12V
Joystick	1	pc	BT

§2 How to use

§2.1 First of first

The ROSYZ-01B robot platform is designed for ROS robot developers. Operators must have the basic knowledge of ROS robots. Please read through this manual before using it, especially read the “Cautions” on the next page carefully.

§2.2 Basic preparation

2.2.1 **Check the completeness of the accessories：**

Open the packing cartons of the robot, take out the ROSYZ-01B robot movement chassis and all parts, and check the packing list, check whether there are leakage and wrong loading.

2.2.2 **Install battery-pack and decks**

For transportation safety, the battery pack and the body of the ROSYZ-01B robot are individually packaged. Before using, please install the battery firstly.

Remove the side cover and put the battery-pack onto the chassis. Then plug the battery-pack plug into “BAT” connector in the cabin.

2.2.3 Switch power on

Main switch is on the front side of top cover, see Page 2. After turn on it, the power LED will be RED solid; then push the ON/OFF button, the operating status LED will be flashing GREEN.

If the power LED is not lighting after main switch closed, please check the battery voltage is good or not. For some reason, when you found the battery voltage is <= 15V, please plug in the portable battery charger and charge the battery at least 3 hours, then try again. The portable charging port and auto charging dock terminals are all on the rear cover of robot

2.2.4 **Install a joystick driver when necessary. (Optional part).**

If you want to drive your robot from joystick (for example, when you run gmapping function), please insert the joystick receiver USB part into the robot computer USB port.

Note: If it the joystick is not purchased from us, you should install the your joystick ROS driver package firstly.

2.2.5 Connect the robot to local WiFi network

First, remove the VGA&USB cabin cover, locate on the rare-top side of robot, then insert the mouse, keyboard, and VGA display. Connect the robot to the WiFi network in your local LAN, and record the IP address of the robot.

Note: The user name of robot Ubuntu system is robot, and the password is 1

§2.3 Run standard ROS demo samples -- from robot PC:

There are two ways to control the YZ01 robot, one is "**from robot PC"**, is directly control robot from robot inner PC. Connect the mouse, keyboard, joystick and VGA display on the IPC (need to open the rear of the robot VGA&USB cabin cover plate), enter the ROS command line to operate

Another way is **"from local network**", need firstly connect robot PC to your WIFI LAN, see 2.2.5 above， and then use the other computers in the LAN, such as your laptop, to drive the ROS commands. If you wish to use LAN mode to remotely control the robot, please skip to Section 2.4.

2.3.0 Connect VGA monitor, keyboard and mouse with Robot PC:

Use screw driver to open the robot VGA & USB cabin cover(see page 3),

Insert your keyboard/mouse and joystick USB header into these USB port, connect your portable VGA monitor with above VGA port.

2.3.1 Map building example

1) Open 0ne or tow terminals and start the following nodes respectively

Terminal1 $ roslaunch yzbot_nav yzbot_createmap.launch

Terminal2 $ roslaunch yzbot_nav keyboard_teleop.launch (if no joystick)

2) After the above four nodes are up and running, open the 3rd terminal

Terminal3 $ rosrun rviz rviz -d `rospack find yzbot_nav`/gmapping.rviz

now you should see a prototype map of the origin attachment. Then you can remote control robot (with wireless joystick or keyboard) slowly walking in the laboratory, straight go to the starting point of the complete paths and return. Please refer to Appendix B for the use of wireless joystick.

3) Open the 4th terminal

Terminal6 $roscd yzbot_nav/maps

$rosrun map_server map_saver -f map

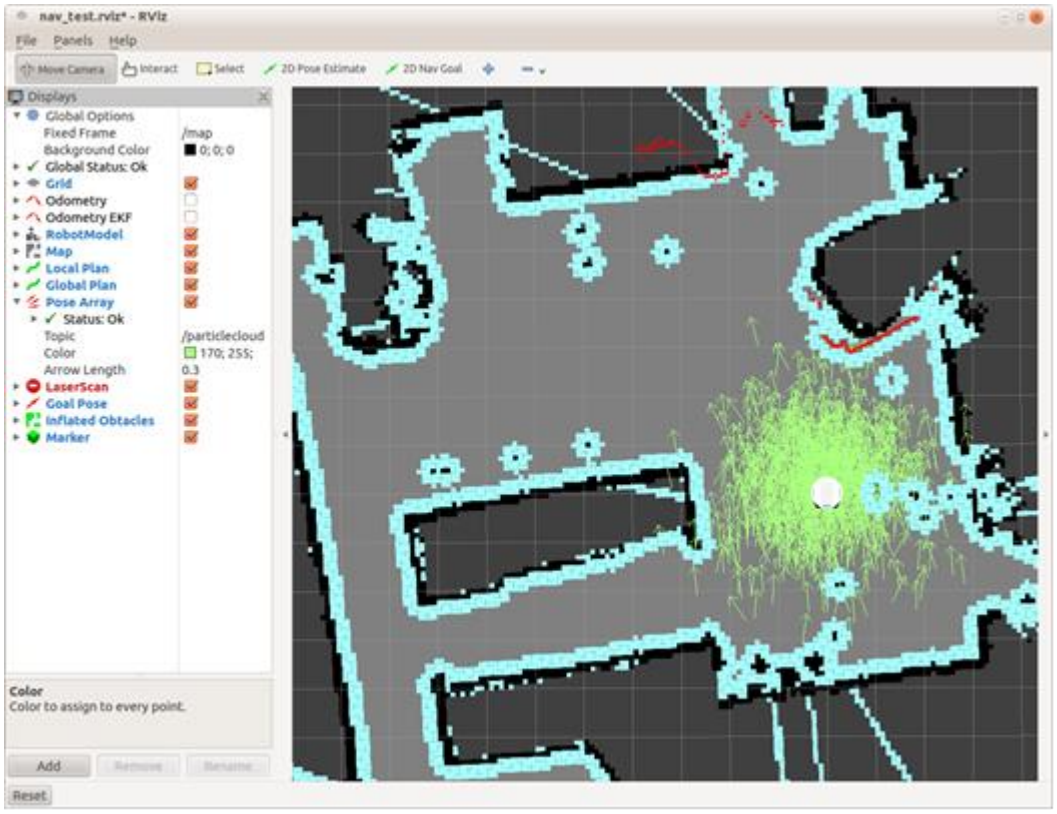
The name of your map is "map" above and can be seen in the yzbot_nav/maps folder. At this point, the drawing is over. If you are not satisfied with the map, you can try it again.

2.3.2 Let robot walk to the designated position of the mouse in the built map.

1) Assume that you have completed the SLAM mapping experiment in 2.3.1 above. The generated map is located in the yzbot_nav/maps directory and the map name is map.

2) Open 2 terminals and run the following nodes:
Terminal1 $ roslaunch yzbot_nav rs_navigation_startup.launch
Terminal2 $ $ rosrun rviz rviz -d `rospack find yzbot_nav`/nav_test.rviz

"2D Pose Estimate" button is used to set the current position of the robot
"2D Nav Goal" button is used to set a target position for the robot to walk to.
"Publish Point" button is used to view the map coordinates at the mouse location



First, use "2D Pose Estimate" button to set the initial position of the robot (laser spot match the map line), then use the "2D Nav Goal" button to select a location on map, your robot will automatically walk to this selected location.

2.3.3 Let robot walks according to the task plan in the map.

When you have completed the above two tasks, let the robot do a slightly more complicated task: select five different places, let the robot walk back and forth between the five different places, and print out the success rate of task execution, cumulative walking distance and cumulative running time on the screen terminal. This program is written in Python script, you can modify it slightly to achieve your own definition tasks (for example: from point a to point B and then to point C, stay for 5 minutes at each point, and then cycle).

1) Modify yzbot_nav/nodes/nav_test.py file with a text editor, such as GEDIT, replace the coordinates in locations [XX] with the expected locations in your office map. These positions coordinate can be viewed with the “publish point” button in rviz.

2) Open 3 terminals and run the following nodes:
Terminal1 $ roslaunch yzbot_nav rs_navigation_startup.launch
Terminal2 $ roscd yzbot_nav/nodes
$ ./nav_test.py
Terminal3 $ rviz rviz -d `rospack find yzbot_nav` /nav_test.rviz

3) If everything works properly, terminal three will be displayed on the screen
*** Click the 2D Pose Estimate button in RViz to set the robot's initial…
At this time, you need to use the “2D pose estimate” button on the rviz window to set the robot on correct initial pose. Then the robot will execute automatically nav_ test.py.
The success rate of task execution, cumulative walking distance, cumulative running time and other information will be displayed on the terminal 3.

You can read the script content of the launch batch command in the above example carefully, and learn how to use the official code of ROS for positioning and navigation.

§2.4 Run standard ROS demo samples -- from local network:

Section 2.3 above uses the local mode of the robot computer to control the robot. You can also access a remote computer to control the robot via local network. Here, there is no need to install a VGA monitor, keyboard and mouse on the robot the robot. This in many cases is very practical value.

2.4.1 Before using the local area network to control the robot, please connect the computer of the robot to the local area network Wi-Fi, this work requires temporary access to the VGA monitor and mouse keyboard, so that the robot can access the network WiFi, use ifconfig command to check the IP address of the robot computer.

2.4.2 Please log in to the Wi-Fi router management interface and binding the IP address of the robot you just saw with the robot network card MAC , so that in the future, the robot will automatically connect to the Wi-Fi, and the IP of the robot's address will be fixed (for example, 192.168.1.101). After the completion of the above work, you can remove the monitor, keyboard and mouse from robot.

2.4.3 Create a new map with LIDAR

1) Suppose that your laptop has installed ROS kinetic, please use the LAN file copy command to copy robot's "~/ws/src/yzbot_nav" files package to your /ws/src directory of your laptop's ROS working folder and compiled it. Next, we will use this laptop to control and operate your robot through SSH to complete the task:
   $ ssh robot@192.168.1.101 (robot IP), login password is 1
2) use SSH to login robot PC, open 2 terminals. Each terminal should firstly run
   $ export ROS_HOSTNAME= robot's IP (such as 192.168.1.101)
   $ export ROS_IP= robot's IP
   Then run:
   Terminal1 $ roslaunch yzbot_nav yzbot_createmap.launch
   Terminal2 $ roslaunch yzbot_nav keyboard_teleop.launch (if no joystick)
3) Run rviz on your own PC terminal to view the screen (terminal 3)
   Note: this is not SSH, firstly you need to run ROS master in your PC terminal
   $ export ROS_ Hostname = laptop IP
   $ export ROS_ MASTER_ Uri = http: / / robot IP: 11311
   $ export ROS_ IP = laptop IP
   Then:
   $ rosrun rviz rviz -d `rospack find yzbot_ nav`/ gmapping.rviz
   now, you should be able to see the map prototype attached to the origin
4) Use SSH to login robot's PC, open the 4th terminal
   Terminal4 $roscd yzbot_nav/maps
   $rosrun map_server map_saver -f map

The name of your map is "map" above and can be seen in the yzbot_nav/maps folder. At this point, the drawing is over. If you are not satisfied with the map, you can try it again.

2.4.4 Let robot walk to the designated position of the mouse in the built map.

1) Assume that you have completed the SLAM mapping experiment in 2.3.1 above. The generated map is located in the yzbot_nav/maps directory and the map name is map.

2) Use SSH to login robot's PC, open 1 terminal, firstly run

$ export ROS_HOSTNAME= robot's IP (such as 192.168.1.101)

$ export ROS_IP= robot's IP

Then run:

Terminal1 $ roslaunch yzbot_nav rs_navigation_startup.launch

3) Run rviz on your own PC terminal to view the screen (terminal 2)

Note: this is not SSH, firstly you need to run ROS master in your PC terminal

$ export ROS_ Hostname = laptop IP

$ export ROS_ MASTER_ Uri = http: / / robot IP: 11311

$ export ROS_ IP = laptop IP

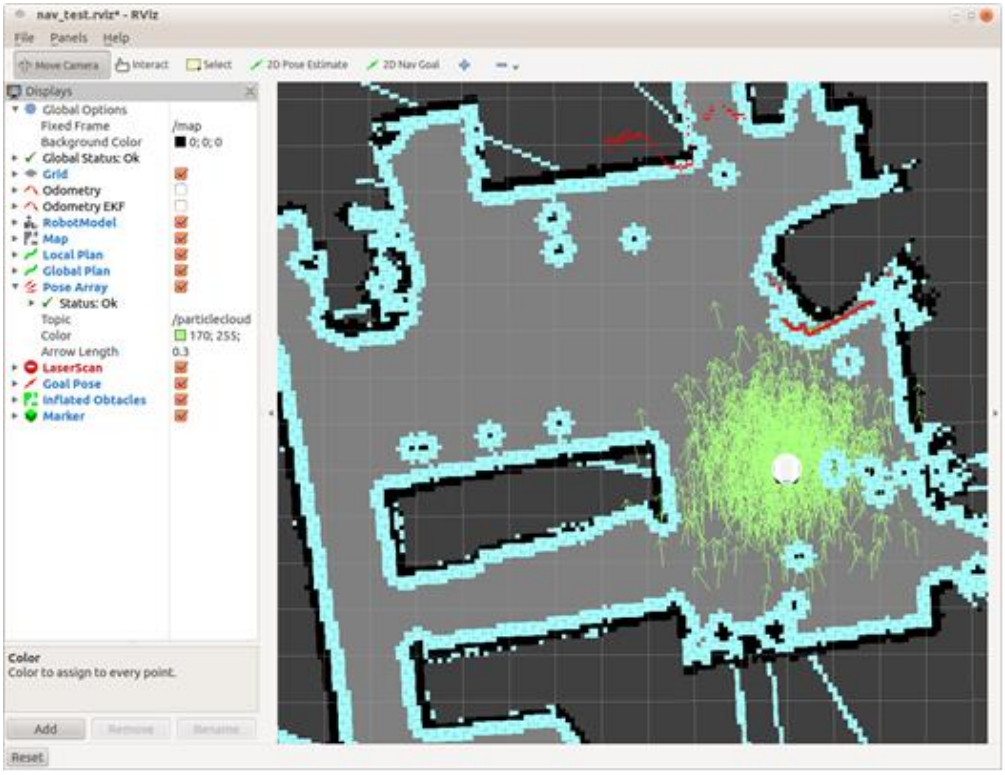
Then:

$ rosrun rviz rviz -d `rospack find yzbot_nav`/nav_test.rviz

"2D Pose Estimate" button is used to set the current position of the robot

"2D Nav Goal" button is used to set a target position for the robot to walk to.

"Publish Point" button is used to view the map coordinates at the mouse location



First, use "2D Pose Estimate" button to set the initial position of the robot (laser spot match the map line), then use the "2D Nav Goal" button to select a location on map, your robot will automatically walk to this selected location.

2.4.5 Let robot walks according to the task plan in the map.

When you have completed the above two tasks, let the robot do a slightly more complicated task: select five different places, let the robot walk back and forth between the five different places, and print out the success rate of task execution, cumulative walking distance and cumulative running time on the screen terminal. This program is written in Python script, you can modify it slightly to achieve your own definition tasks (for example: from point a to point B and then to point C, stay for 5 minutes at each point, and then cycle).

1) Use SSH to login robot's PC, and modify yzbot_nav/nodes/nav_test.py file with a text editor, such as vim or nano. Replace the coordinates in locations [XX] with the expected locations in your office map. These positions coordinate can be viewed with the "publish point" button in rviz.
2) Use SSH to login robot's PC, open 2 terminals, Each terminal firstly run
   $ export ROS_HOSTNAME= robot's IP (such as 192.168.1.101)
   $ export ROS_IP= robot's IP
   Then run:
   Terminal1 $roslaunch yzbot_nav rs_navigation_startup.launch
   Terminal2 $roscd yzbot_nav/nodes
   $ ./nav_test.py
3) Run rviz on your own PC terminal to view the screen (terminal 3)
   Note: this is not SSH, firstly you need to run ROS master in your PC terminal
   $ export ROS_ Hostname = laptop IP
   $ export ROS_ MASTER_ Uri = http: / / robot IP: 11311
   $ export ROS_ IP = laptop IP
   Then:
   $ rosrun rviz rviz -d `rospack find yzbot_nav`/nav_test.rviz
4) If everything works properly, terminal three will be displayed on the screen
   *** Click the 2D Pose Estimate button in RViz to set the robot's initial…
   At this time, you need to use the "2D pose estimate" button on the rviz window to set the robot on correct initial pose. Then the robot will execute automatically nav_ test.py.
   The success rate of task execution, cumulative walking distance, cumulative running time and other information will be displayed on the terminal 3.

§2.5 Use YZ private navigation package to run robot

The open source navigation algorithm of ROS official website can provide good learning guidance for beginners, but these open source localization & navigation algorithms have some limitations. There are still many limitations in practical application. It's easy to deviate or even get lost, and the effect of motion navigation is not very good. For real application scenarios, Shenzhen YZ Robot Co. have developed a special robot localization & navigation algorithm packs with to overcome above limitations. We have also integrated the executable packages of these algorithms into the rosyz01 /02 robot system. You only need to replace the relevant launch commands in above 2.3 section and 2.4 section sample:

Note: these algorithms are only for the research and learning of rosyz platform. Do not use these software for commercial purposes or copy them to a third party without the authorization and license of Shenzhen Pushu Technology Co., Ltd.

2.5.1 Map related knowledge

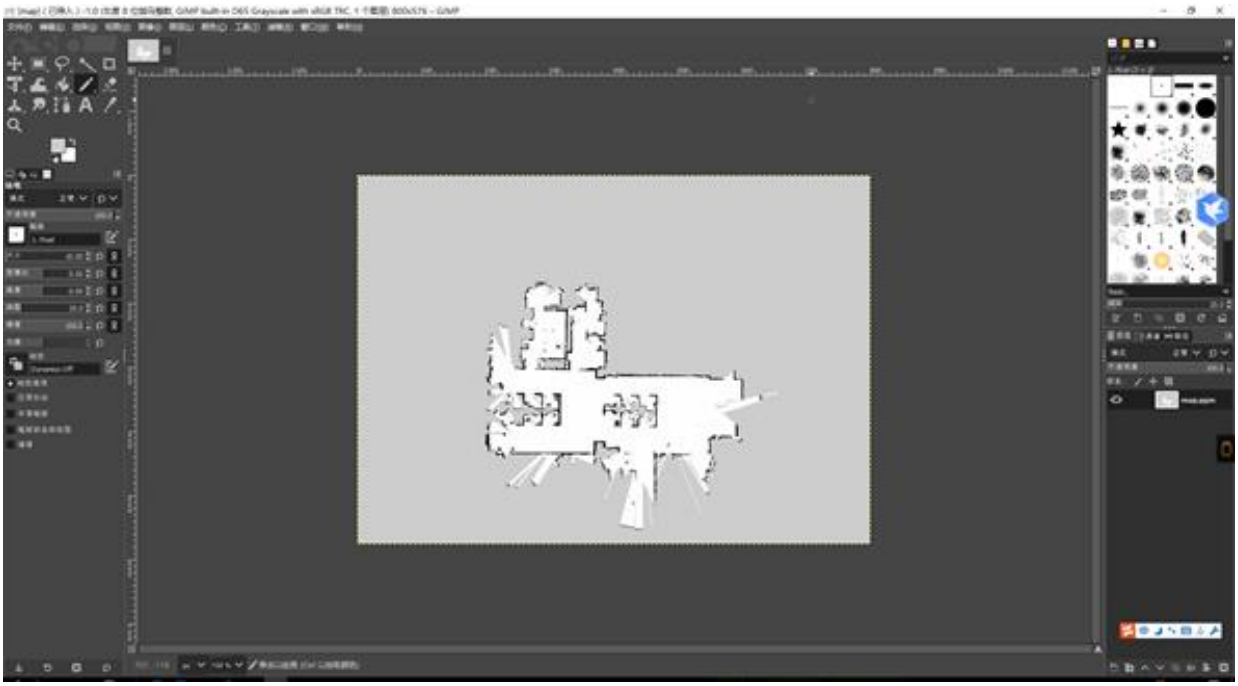
Please refer to Section 2.3.1 or 2.4.3 to build a new map for robot working site. For the actual needs of customers, for example, to set up some areas where robots are not allowed to enter, we should do some modification on this raw map firstly.

If you are using the robot local computer, please use the GMIP software in the robot computer to open “map.pgm” image in yzbot_nav/maps directory. If you use local LAN wi-fi network computer, please copy this “map.pgm” image to your desktop computer via LAN, then open the image using GMIP or Photoshop on your desktop PC. Here's an example:

Set the map size: Click Image --> Canvas Size --> to enter the appropriate width and height and cut out the map.

Note: Skew map does not hinder the robot's positioning and navigation, but it may be uncomfortable for users. You can use the map rotation command to rotate the image to your satisfaction, and then crop the map.

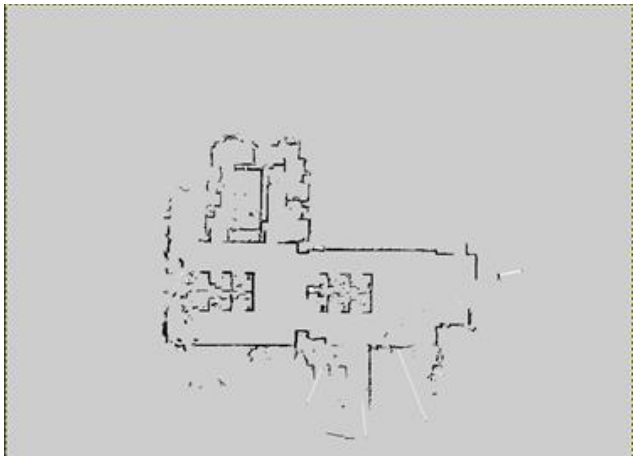
Principle: the gray color area can be cut off, only leave a bit, as below:



Click Change Size to confirm

Fill the white area with gray color:

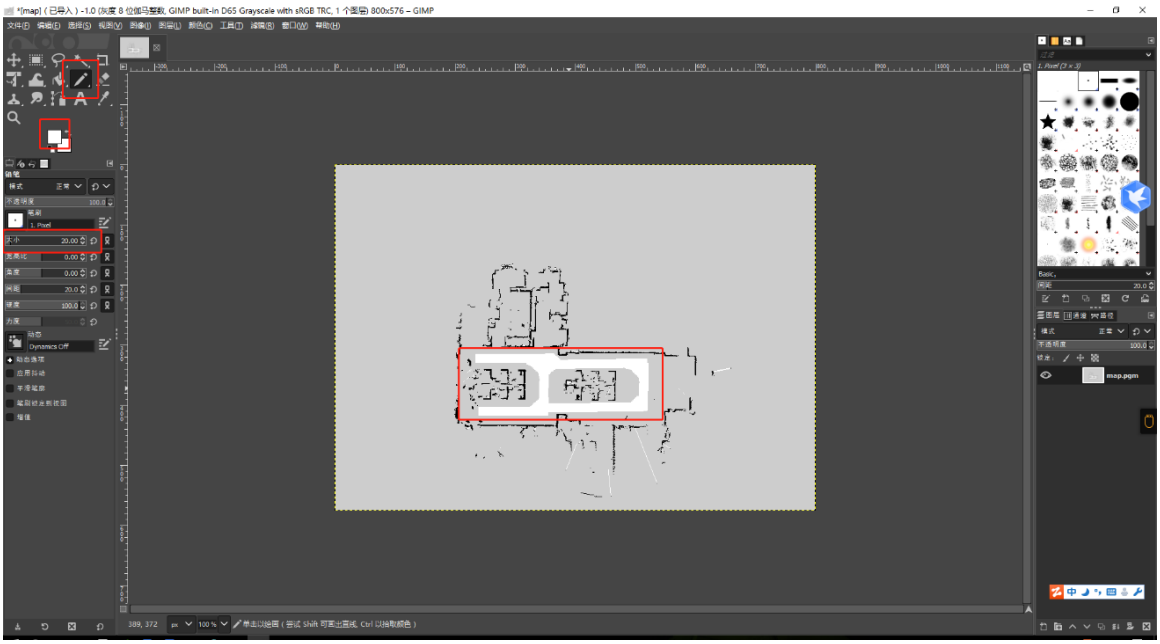
Click the "Bucket Fill Tool", select a color (hold Ctrl and click on the gray area), select the gray, and click the oil bucket on the white area to get the effect as shown in following figure:



Fill white area in the picture:

This operation will directly affect the walking space area of the robot. White color is the valid area for robot walking.

Select the Pencil Tool (U), set color white, and the brush size as appropriate with white lines painted in, robot will only permited to walk in the white area. It is recommended to use SHIFT shortcut key to draw the line to paint the line, so that the effect is relatively flat. As shown in figure



Save the map:

Click File ---> "Export as" ----> Click Export (note the export path)

When it finished, put the new map in the /maps directory and name map.pgm

Rule of Map Editing:

Greying all and then draw white.

Gray: Set up a no-go area, the robot will not permitted walk in these gray areas.

Paint white: Set the robot valid walking area with white color.

Map pixels and real world coordinate relationship:

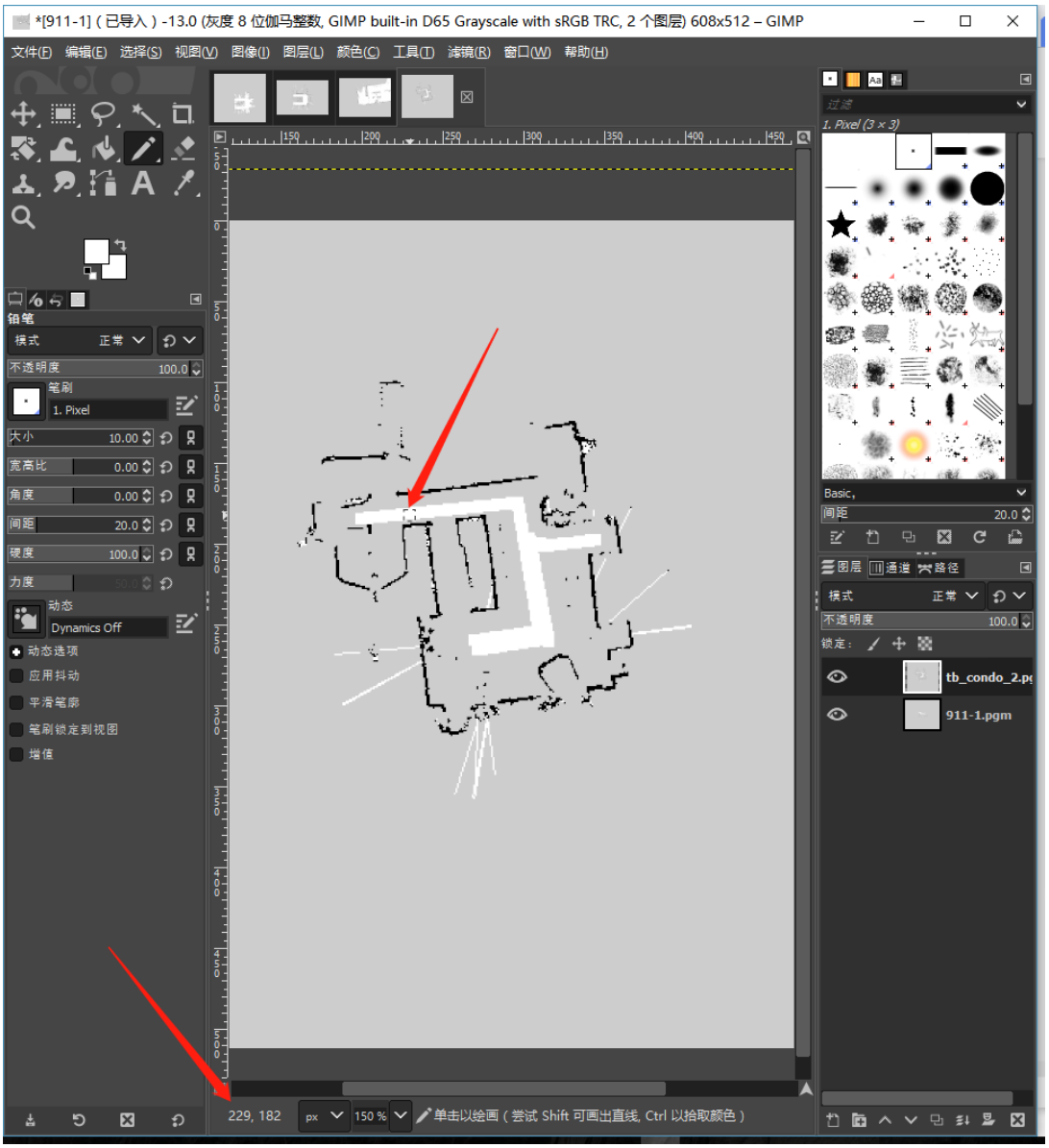
1. According to the usage habits of the map, the lower left corner of the map image is set as the origin of the real world (0,0).
2. By default, the ratio of map pixels to the real world is 0.05m/pixel. This parameter is selected at map creating job (such as gmapping in earlier).

The 0.05 m/pixel is suitable for indoor construction under normal circumstances, and 0.1 m/pixel or even bigger can be selected for outdoor construction. Indoor drawings of small areas can also be 0.025m/pixel.

Assuming a map image is 600 X 500 pixels and a scale of 0.05 meters per pixel,

this map corresponds to a real-world area will be 30 meters long and 25 meters wide.

For example, the width of this map W is 608, the height H is 512:



The pixel number at the box position in the figure above is 229,182, and the corresponding world X and Y coordinates of this pixel are:

X = 229 * 0.05 = 11.45

Y = (512-182) * 0.05 = 16.5 (512 is the total height of this map)

A(Angle) = 0.0 (Orientation to the right is 0.0,
Orientation to the left is 3.14,
Orientation to the top is 1.57,
Orientation to the bottom is -1.57)

Conclusion: The coordinate parameters of the above pixels are (11.45, 16.50, 0.0) (the robot faces to the right)

2.5.2 Samples for using YZ private navigation package to drive robot.

If you have completed the map building and map editing in 2.5.1 above, follow these steps to drive robot under YZ private navigation package to control the robot navigation tasks:

1. Let the robot walk to assigned target location in your map

The basic operation instructions are almost the same as those in Section 2.3.2 or 2.4.4 above, except that "rs_navigation_startup" should be replaced by "yzbot_navigation_startup", the map should be replaced with the new map.

As shown in Section 2.3.2, simply change the command line for Terminal 1 to:

Terminal 1: $roslaunch yzbot_nav yzbot_navigation_startup.launch

2. Let robot walks according to the task plan in the map

Again, the instructions are almost the same as in Section 2.3.3 or 2.4.5, and you need to replace "rs_navigation_startup" with ""yzbot_navigation_startup, also replace the old map with the new map.

As shown in Section 2.3.3, simply change the command line for Terminal 1 to:

Terminal 1: $roslaunch yzbot_nav yzbot_navigation_startup.launch

3. A brief introduction to AMCL and DMCL of robot localization algorithm

The recommended localization algorithm package on ROS official website is AMCL. http://wiki.ros.org/amcl is ROS official website, there are detailed introduction to AMCL package.

In order to make the AMCL package accurately realize the real-time robot localization function, the prerequisite is to set initial position x, y, a(the robot orientation Angle) before the robot moving.

There are two ways to do this obtained the initial position of the robot x, y, a:

The first method is the RVIZ diagram described in the previous example, which starts with the "2D Pose Estimate" button, set the initial position of the robot so that the laser spots basically coincide with the contour of the obstacles on the map. Check at this time the printing content of the RVIZ terminal window, the initial coordinate value of the robot will be printed on the screen, please record these values.

The second method is to use an image editing software (such as GIMP or Photoshop) to open the map image and place the mouse over the current position of the robot located, and view the pixel number (X,Y) of the mouse position, as shown in the previous section method to calculate x, y coordinate values, the value of a also refer to the content of this section.

§3 CAUTION

3.1 **Pre charging**: before running the robot for the first time, please charge battery at least 3 hours. For the transportation reasons, the battery before shipping only has very little electricity.

3.2 **Charge temperature:** pay special attention to charging at room temperature of 0~35 degrees Celsius. High or too low ambient temperature can damage the battery!

3.3 **External power interface:** please connect the external battery and output DCDC power strictly according to the pin position and polarity of the power supply interface. The wrong wiring will damage the interface board or other devices. Make sure that the maximum current used by the device does not exceed the limited current value of the DCDC board.

3.4 **Troubleshooting:** if the robot is in the use of abnormal function, please try to turn off the power, and then restart, in general, the robot will be restored to normal, if restarted, the robot can not be used normally, please notify the technical service personnel for remote guidance.

3.4 **Emergency issues:**

When the robot walks abnormal, please press the red emergency switch!

When the robot or charger has smoke, please turn power immediately!

When the robot is in serious collision or fall accident, turn off the power!

§4 ROSYZ-01B Datasheet

Move base size	45X40X37 cm
Move base weight	22KG(not including battery)
Move drive mode	two-wheel differential
Battery required	16V21AH LiFePO4 battery-pack
Motor type	12~24V wheel-hub motors
Inside Computer features	CPU: Intel® Core™ i5-3470T processor RAM: 4G Storage Disk: 500G I/O : USB2.0*4, VGA, Microphone/headset port, DP, RS232 Ethernet: Integrated 100M/1000M Ethernet Wireless: 802.11b/g/n Combo (WLAN+BT)
DCDC provided	5V2A、12V2A、16V4A
Move speed	0.1-1.0m/s
Maximum load weight	50KG
Externa charger	18V8A smart charger
Auto charging guide	Optional
Control Board	STM32F10X ARM chip
Emergency stop	Push RED button
Upper layer height	5CM
Hardware data provided	All electronic schematic drawings are provided
ROS diver provided	Provide ROS node binary file which can output each wheel’s ticks and accept setting speed value
ROS demo	Provide a basic keyboard remote control moving demo application. C++ source code of this demo is used.

Appendix A: ROS Topics of YZ robot diver node

1. Firstly, launch robot diver node:

   roslaunch stm32_connect start_basecontrol.launch

2. Topics of robot diver node

   2.1. **Topic name: "Auto_Charging**", function: to enable or disable auto-charging flag

   Topic value: auto_charging_flag = 0: Disable = 1:Enable

   【Example】: in a terminal , input following text:

   rostopic pub /Auto_Charging robot_msgs/Charging_Control "auto_charging_flag:1"

   Above instruction will let robot go to charging dock automatic

   2.2. **Topic name："PMS_get_status**", check battery power level and status

   Topic value: pms_charging_flag = 0: now not charging = 1: now charging

   Pms_battary_level = n : means battery power level is n% remain

   【Example】: in a terminal , input following text:

   rostopic echo /PMS_get_status

   Above instruction will display battery status and remain capacity level

   2.3. **Topic name："Ultrasound_result"** , check sonar sensor trigger status

   Topic value: cs_obs = 0：none obstacle =1：front obstacle

   =16：rear obstacle =17：front and rear obstacles

   【Example】: in a terminal , input following text:

   rostopic echo /Ultrasound_result

   Above instruction will display ultrasonic obstacle sensor status

   2.4. **Topic name："Wheel_Switch" ,** check emergency switch status

   Topic value: Switch = 0：Pushed(Enable) =1：Release(Disable)

   【Example】: in a terminal , input following text:

   rostopic echo /Wheel_Switch

   Above instruction will display emergency switch status

   2.5. **Topic name："cmd_vel"** , check or set the linear/angular velocity

   【Example】: in a terminal , input following text:

   rostopic pub /cmd_vel geometry_msgs/Twist "linear：

   x:0.3 y:0.0 z:0.0 angular: x:0.0 y:0.0 z:0.1"

   This will let robot run at 0.3m/s linear and 0.1r/s angular velocity

   2.6. **Topic name："odom"** ， check odometry data

   【Example】: in a terminal , input following text:

   rostopic echo /odom

   Above instruction will display odometer data

   2.7. **Topic name："sonar1[2,3,4]"** , check sonar sensors data, 1~4 means different sensors

   Topic value: field_of_view：cone angle of this sonar

   min_range：minimum active range of this sonar

   max_range：maxmum active range of this sonar

   range: current range measured

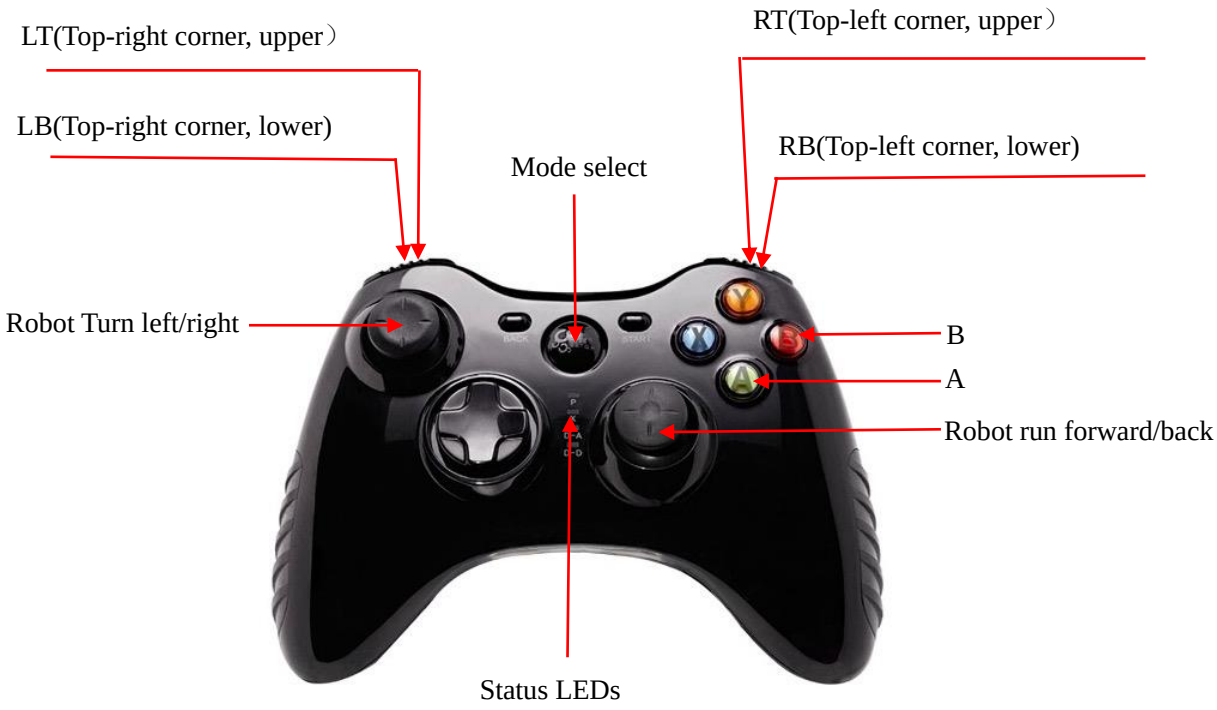
   【Example】: in a terminal , input following text:

   rostopic echo /sonar1

Appendix B: introduction of remote joystick and map creating

1、 Understanding joystick

When creating the site map and manual remote control robot for the first time, we must rely on the remote control handle. The following figure shows the front view of the remote control handle, and the distribution of the keys is as follows:



Note: X, Y, BACK, START, these four keys are not actived。

2. How to use

Firstly, make sure MR9 robot is power on. Then press the circular "mode select" key at the middle position above the handle. If you press it once, the remote control function of the handle will be turned on normally. At this time, the first and the third status LED will appear green constants. (Note: if all four LEDs flash at the same time, it means that the robot is faulty or the handle is too far away from the robot, so the handle can't receive the robot signal; if two indicator lights flash, it means the battery power in the handle is too low, please charge the handle with standard USB cable for 1 hour; if the 2nd and 4th LED are on, it means that the mode and robot do not match Press "mode selection key" again.)

2.1 Control robot move

Robot turn left/right：Hold on“LB”and push“Robot turn”left/right

Robot run forward/back：Hold on“LB”and push“Robot run”forward/back

2.4 Remote control robot goes to the charging dock to charge

When the robot is about 1 m in front of the charging dock, if the <LB> + <A> key is pressed and hold at the same time, the robot will automatically return to the charging point to charge.

When the robot is not about 1 meter in front of the charging pile, please use the remote control handle to move the robot firstly about 1 meter in front of the charging pile, and then press "LB key" + "a key" at the same time.

When the map has been built and the robot is positioning and navigating according to the map, if press the "LB key" + "RB key" + "a" key at the same time, and the robot will be terminated all current tasks, go back and recharge by yourself.

Appendix C: Size Drawing of YZ01B