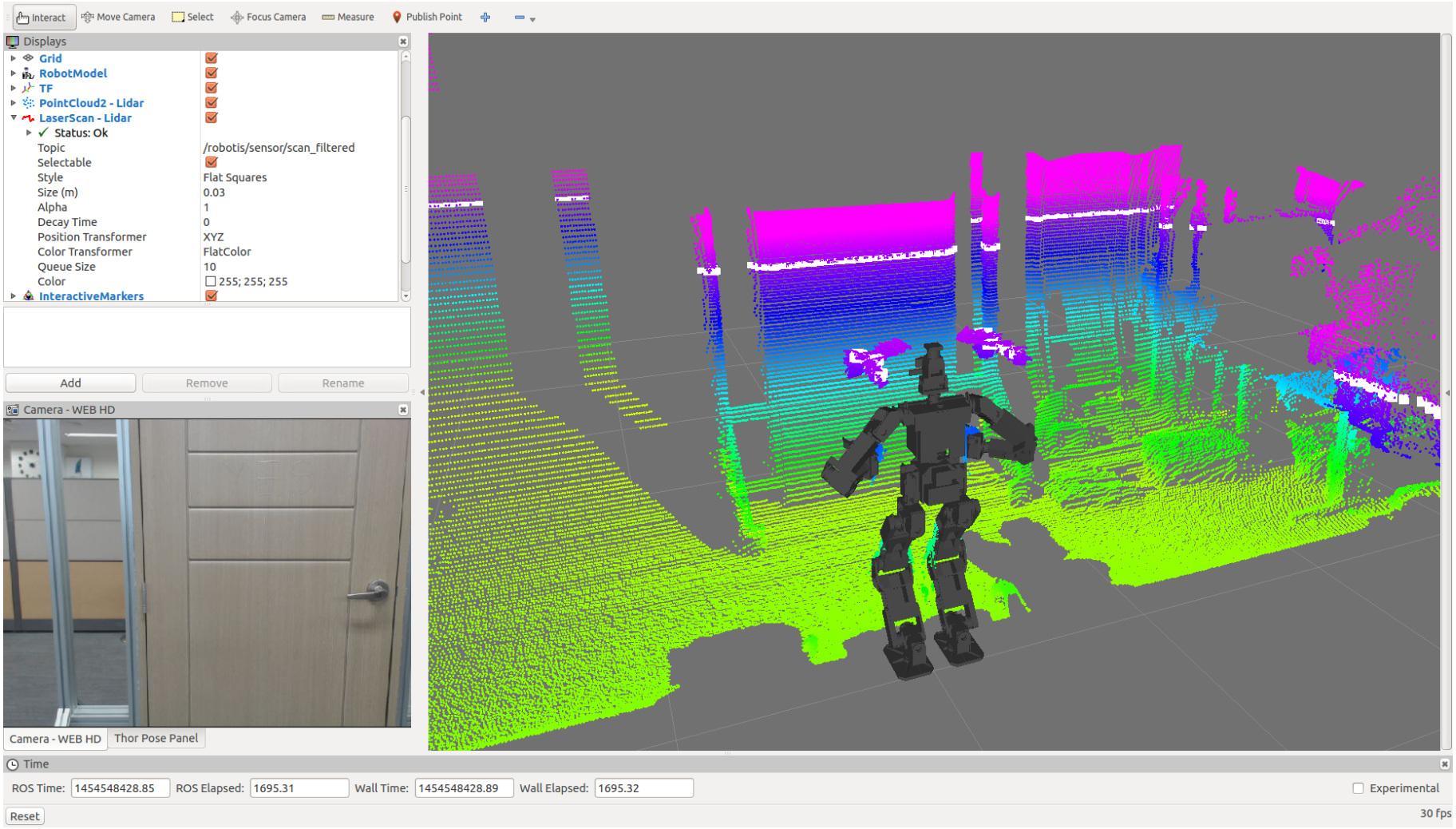
ROS Framework Tutorial

Operation, Sensors and Vision

Tutorial Overview



Tutorial Contents

CONTENTS

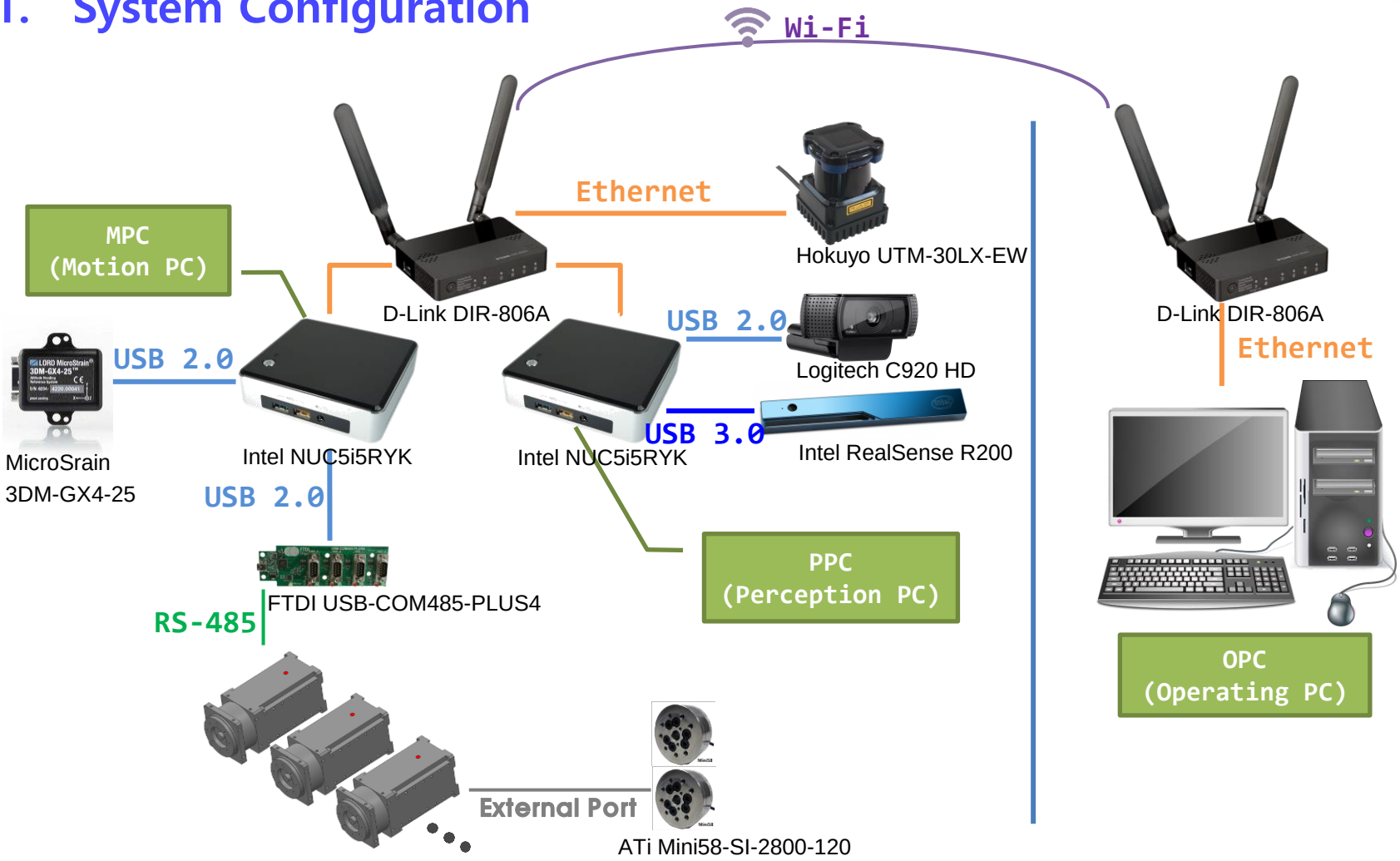
Remote Operation

Remote Operation Contents

CONTENTS

Remote Operation

1. System Configuration



2. OPC (Operating PC)

- Overview
  OPC is the PC to be used to control the robot outside of it. OPC is also used to check the robot state and the sensor.

1. OS
   - Ubuntu 14.04 LTS

2. ROS(Robot Operating System)
   - Version : Indigo
   - Installation : http://wiki.ros.org/indigo/Installation/Ubuntu
   - Environment Setup : http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment

2. ROS(Robot Operating System)

- Example of ROS Environment Setup

1. Check IP Address
   - Core (Perception PC) IP : 10.17.3.35
   - OPC (Operation PC) IP : 10.17.3.100
   - OPC IP address is an example. After OPC connect the AP server(DIR-806A), user should check IP address of OPC. (ex. ifconfig)
2. Edit the bashrc file with Your *Favorite Editor*. (~/.bashrc)
   ```
   $ gedit ~/.bashrc
   ```
3. Add the below contents.

```
# set ROS Indigo
source /opt/ros/indigo/setup.bash
source ~/catkin_ws/devel/setup.bash

# set ROS Network
export ROS_MASTER_URI=http://10.17.3.35:11311
export ROS_HOSTNAME=10.17.3.100

# set ROS alias command
alias cw='cd ~/robotis/catkin_ws'
alias cs='cd ~/robotis/catkin_ws/src'
alias cm='cd ~/robotis/catkin_ws && catkin_make'
```

3. ROBOTIS ROS Package (for OPC)
   - GIT : https://github.com/ROBOTIS-GIT
   - Installation

1. Getting the source code.

```
$ cd ~/catkin_ws/src
$ git clone https://github.com/ROBOTIS-GIT/ROBOTIS-THORMANG-OPC.git
$ git clone https://github.com/ROBOTIS-GIT/ROBOTIS-THORMANG-Common.git
```

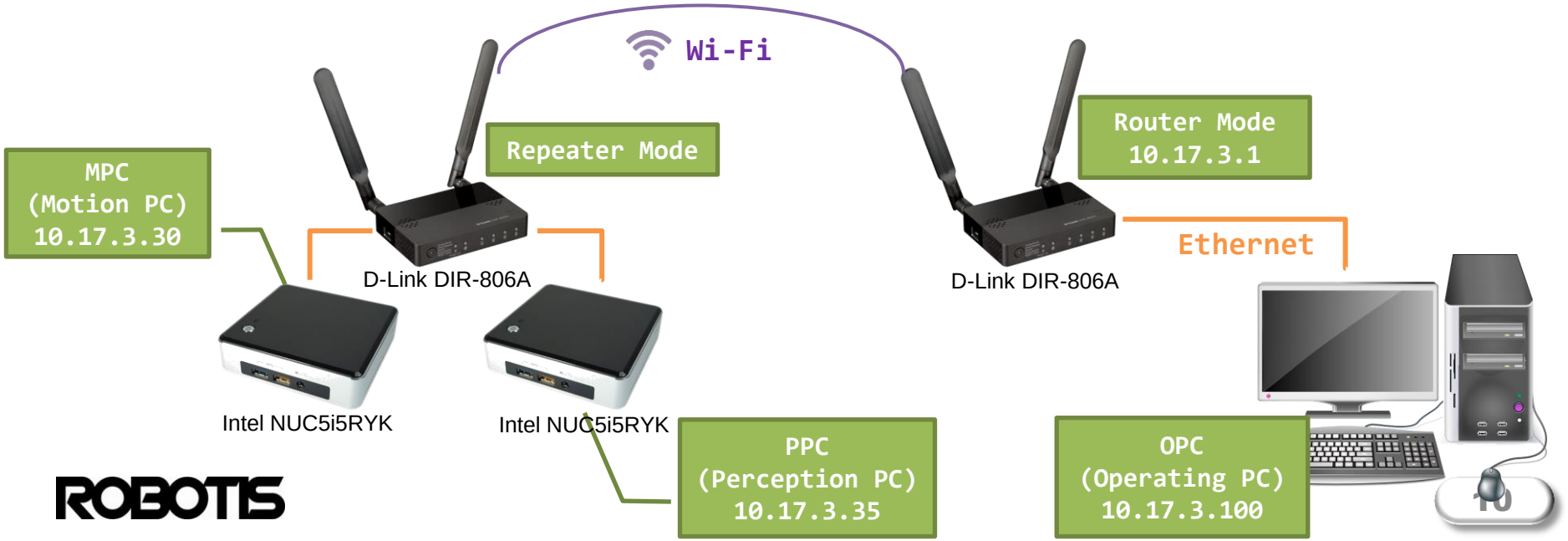
2. Building packages.

```
$ cd ~/catkin_ws
$ catkin_make
```

3. How to run THORMANG3

1. Connecting THORMANG3
   From your computer go to your LAN settings and set static IP to the same network (Example: 10.17.3.xxx)

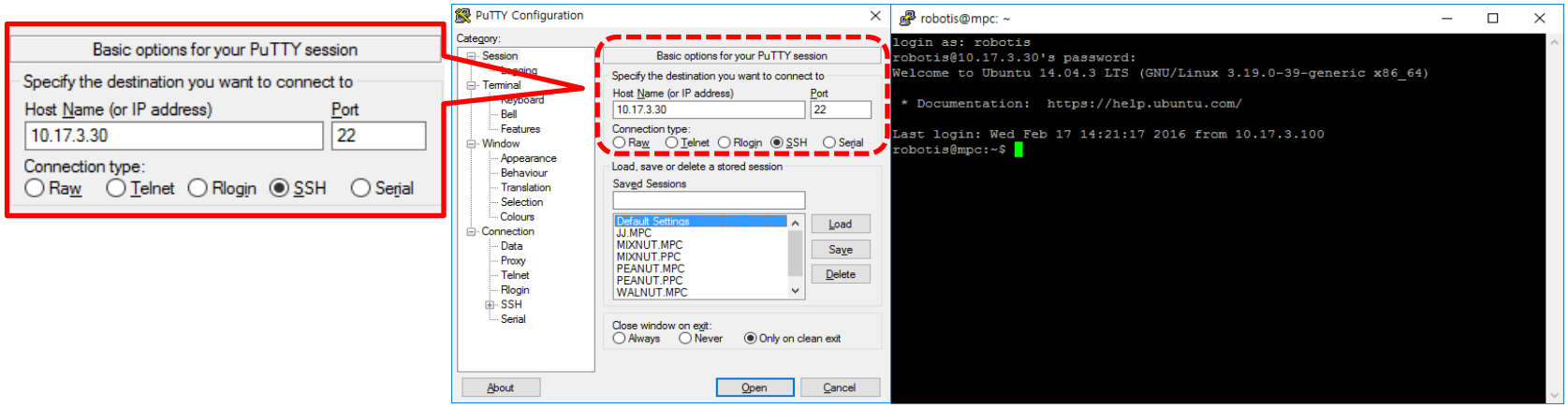
- Connection Infomation
  1) MPC (Motion PC) IP : **10.17.3.30**
  2) PPC (Perception PC) IP : **10.17.3.35**
  3) MPC & PPC user name : **robotis**
  4) MPC & PPC password : **111111**



Remote Operation

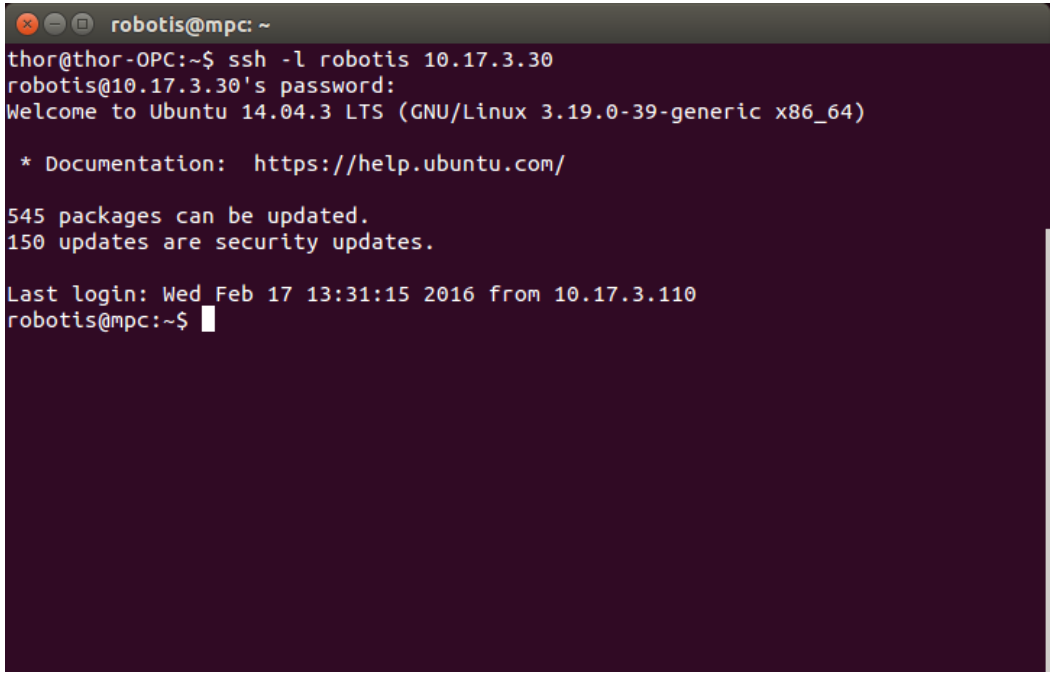
- Example with SSH Client (Windows)
  1) Execute SSH client program (ex: PuTTY)
  2) Input THOR-MANG 3 MPC's IP address : **10.17.3.30**
  3) Select **SSH** as a connection type and then open it.
  4) Input THOR-MANG 3 MPC's user name : **robotis**
  5) Input THOR-MANG 3 MPC's password : **111111**

  ROBOTIS recommends that users connect via SSH client.



Remote Operation

- Example with SSH Client (Ubuntu)
  1) Open terminal window
  2) Input SSH command with user name and MPC's IP address :
     `$ ssh -l robotis 10.17.3.30`
  3) Input THOR-MANG 3 MPC's password : **111111**

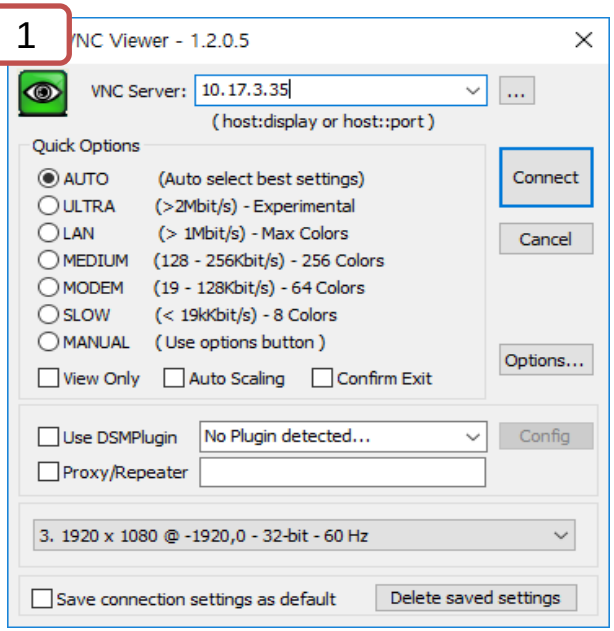


Remote Operation

- Example with VNC client (Windows)
  1) Execute VNC client program (ex: Ultra VNC Viewer)
  2) Input THOR-MANG 3 MPC's IP address : **10.17.3.30**
  3) Input THOR-MANG 3 MPC's password : **111111**

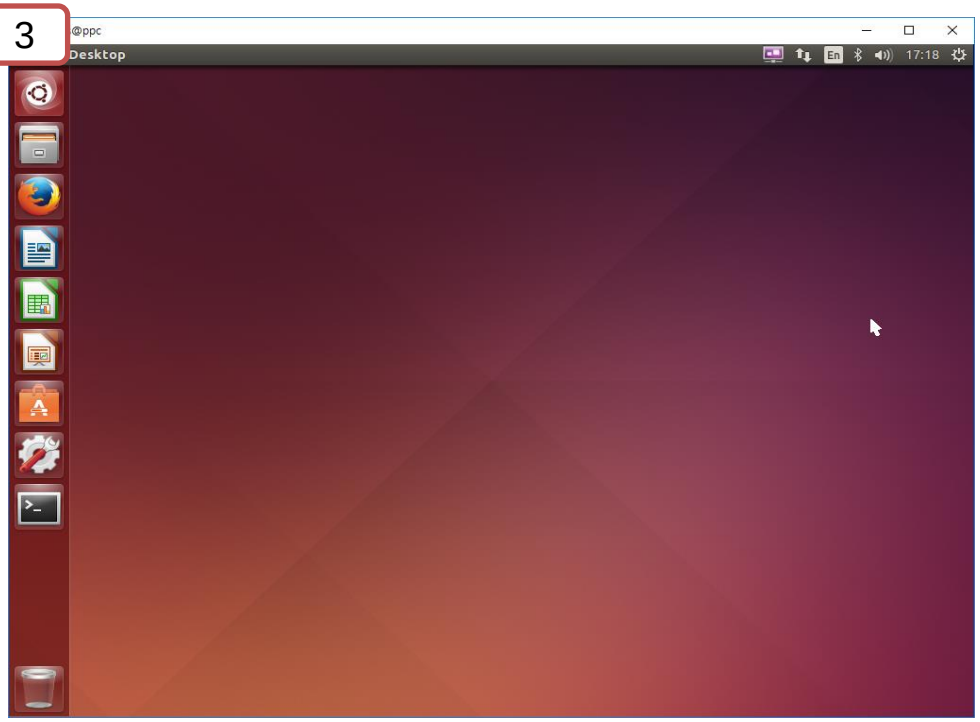
  Accessing THOR-MANG3 MPC via remote desktop may result in slower performance.

1



2

3



AP Server Information

- **Account**
  - User : admin
  - PW : admin

- **Network**
  1. IP : 10.17.3.1
  2. Mode : Router
  3. Wireless
     - 5G
       SSID : THORMANG-K30-5G
       PW : 11111111
     - 2.4G
       SSID : THORMANG-K30
       PW : 11111111

- **Mode Description**
  - Orange : Router
  - Green : Repeater
  - Red : AP
  - RESET : 10 sec

ROBOTIS

3. How to run THORMANG3

- Run Packages in PPC (Perception PC)

1. Connect to the PPC with SSH client program. (IP : 10.17.3.35)

2. roscore
   roscore is collection of nodes and programs that are pre-requisites of a ROS-based system. You **must** have a roscore running in order for ROS nodes to communicate. It is launched using the `roscore` command.
   NOTE : If you use roslaunch, it will automatically start `roscore` if it detects that it is not alread running.
   - The `roscore` can be launched using the `roscore` executable :
     ```
     $ roscore
     ```

3. How to run THORMANG3

- Run Packages in PPC (Perception PC)

3. THORMANG3's Sensors in PPC (Web HD camera, Depth camera)
   In PPC, There are 2 packages to use sensors(web camera, depth camera) and there is a package to utilize sensor data(laserscan).
   - Type below command :
     ```
     $ roslaunch thormang3_sensors thormang3_sensors.launch
     ```
     Note : If device order is not correct, type below command.
     ```
     $ roslaunch thormang3_sensors thormang3_sensors.launch other_order:=true
     ```
   - Break down the launch file.

```
<!-- filtering laserscans and assembling laserscans into pointcloud -->
<include file="$(find thormang3_sensors)/launch/thormang3_pointcloud.launch" if="$(arg use_lidar)"/>

<group if="$(arg other_order)" >
        <!-- launch RealSense (depth-camera) package -->
        <include file="$(find thormang3_sensors)/launch/thormang3_realsense.launch" if="$(arg use_realsense)"/>
        <!-- launch Web HD camera package -->
        <include file="$(find thormang3_sensors)/launch/thormang3_web_cam.launch" if="$(arg use_web_camera)"/>
</group>
```

3. How to run THORMANG3

- Run Packages in PPC (Perception PC)

3. THORMANG3's Sensors in PPC (Web HD camera, Depth camera)
   - Break down the "thormang3_pointcloud.launch" file.

```
<launch>
  <arg name="filter_input_scan" default="/robotis/sensor/scan" />
  <arg name="filter_output_scan" default="/robotis/sensor/scan_filtered" />
  <arg name="assembler_frame" default="pelvis_link" />

  <!-- filtering laserscan -->
  <node pkg="laser_filters" type="scan_to_scan_filter_chain" output="screen" name="laser_filter">
    <rosparam command="load" file="$(find thormang3_sensors)/filters/shadow_filter.yaml" />
    <remap from="scan" to="$(arg filter_input_scan)" />
    <remap from="scan_filtered" to="$(arg filter_output_scan)" />
  </node>
  <!-- service to assemble laserscan to pointcloud -->
  <node type="laser_scan_assembler" pkg="laser_assembler" name="my_assembler_chest">
    <remap from="scan" to="$(arg filter_output_scan)"/>
    <param name="max_scans" type="int" value="600" />
    <param name="fixed_frame" type="string" value="$(arg assembler_frame)" />
    <param name="ignore_laser_skew" type="bool" value="true" />
    <remap from="/assemble_scans2" to="/robotis/sensor/service/assemble_scans2" />
  </node>
  <!-- handle assemble laserscan -->
  <node pkg="thormang3_sensors" type="assemble_laser_node" output="screen" name="laser_assem" />
</launch>
```

3. How to run THORMANG3

- Run Packages in PPC (Perception PC)

3. THORMANG3's Sensors in PPC (Web HD camera, Depth camera)
   - Break down the "thormang3_web_cam.launch" file.

```
<launch>
  <!-- center camera node -->
  <node pkg="uvc_camera" type="uvc_camera_node" name="uvc_camera_center_node"  >
    <param name="frame_id" type="string" value="head_p_link" />
    <param name="device" type="string" value="/dev/video3" />
    <remap from="/image_raw" to="/robotis/sensor/camera/image_raw"/>
    <remap from="/camera_info" to="/robotis/sensor/camera/camera_info"/>
  </node>
</launch>
```

3. How to run THORMANG3

- Run Packages in PPC (Perception PC)

3. THORMANG3's Sensors in PPC (Web HD camera, Depth camera)
   - Break down the "thormang3_realsense.launch" file.

```
<launch>
    <arg name="cHeight" default="480" />
    <arg name="cWidth" default="640" />
    <arg name="dHeight" default="360" />
    <arg name="dWidth" default="480" />
    <arg name="colorCamera" default="/dev/video0" />
    <arg name="depthCamera" default="/dev/video1" />
    <arg name="pcScale" default="1000" />
    <arg name="enableDepth" default="1" />
    <arg name="enableColor" default="1" />
    <arg name="directory" default="0" />
    <arg name="playFromFile" default="0" />
    <arg name="recordToFile" default="0" />
    <arg name="serialNumber" default="-1" />

    <node pkg="nodelet" type="nodelet" name="standalone_nodelet"  args="manager" output="screen"/>
    <node pkg="nodelet" type="nodelet" name="CR200Nodelet" args="load r200_ros/CR200Nodelet
standalone_nodelet
        args... ">
    </node>
</launch>
```

3. How to run THORMANG3

- Run Packages in MPC (Motion PC)

1. Connect to the MPC with SSH client program. (IP : 10.17.3.30)

2. Synchronization time to PPC
   In the ROS system, the time synchronization between the PC is important.
   - Make the shell script as follows :

```
#! /bin/sh
sudo date --set='-2 secs‘
sudo ntpdate 10.17.3.35
sudo hwclock -w
```

   - Run the shell script

3. How to run THORMANG3

- Run Packages in MPC (Motion PC)

3. THORMANG3 Manager and IMU

`thormang3_manager` is a base node using ROBOTIS framework. thormang3_manager must be running before you run the Simple Demo nodes and before you check the sensors because these are using thormang3_manager.

- To start THORMANG3 Manager, simply run :

```
$ sudo bash
# roslaunch thormang3_manager thormang3_manager.launch
```

- Break down the launch file.

```
<launch>
  <arg name="use_imu" default="true" />
  <param name="offset_table" value="$(find thormang3_manager)/config/offset.yaml"/>
  <param name="robot_file_path" value="$(find thormang3_manager)/config/THORMANG3.robot"/>
  <param name="init_file_path" value="$(find thormang3_manager)/config/dxl_init.yaml"/>
  <param name="ft_data_path" value="$(find thormang3_manager)/config/ft_data.yaml"/>
  <param name="ft_calibration_data_path" value="$(find thormang3_manager)/config/ft_calibration_data.yaml"/>

  <!-- imu sensor package -->
  <include file="$(find imu_3dm_gx4)/launch/imu.launch" if="$(arg use_imu)"/>

  <!-- THORMANG3 Manager -->
  <node name="thormang3_manager" pkg="thormang3_manager" type="thormang3_manager" output="screen"/>

</launch>
```

3. How to run THORMANG3

- Run Packages in MPC (Motion PC)

3. THORMANG3 Manager and IMU

`imu_3dm_gx4` is a imu sensor node.

- Break down the "imu.launch" file.

```
<launch>
  <arg name="output" default="screen"/>
  ......

  <node pkg="imu_3dm_gx4" name="$(arg imu)" type="imu_3dm_gx4" output="$(arg output)">
    <param name="device" type="string" value="$(arg device)" />
    <param name="verbose" type="bool" value="$(arg verbose)"/>
    <param name="baudrate" type="int" value="$(arg baudrate)" />
    <param name="frameId" type="string" value="$(arg frame_id)"/>
    <param name="imu_rate" type="int" value="$(arg imu_rate)" />
    <param name="filter_rate" type="int" value="$(arg filter_rate)"/>
    <param name="enable_filter" type="bool" value="$(arg enable_filter)"/>
    <param name="enable_accel_update" type="bool" value="$(arg enable_accel_update)"/>
    <param name="enable_mag_update" type="bool" value="$(arg enable_mag_update)"/>
    <remap from="/imu/imu"    to="/robotis/sensor/imu/imu"/>
    <remap from="/imu/filter" to="/robotis/sensor/imu/filter"/>
  </node>
</launch>
```

Remote Operation

3. How to run THORMANG3

- Run Packages in MPC (Motion PC)

4. TF and urg_node(Lidar)
   TF is a package that lets the user keep track of multiple coordinate frames over time. TF maintains the relationship between coordinate frames in a tree structure buffered in time, and lets the user transform points, vectors, etc between any two coordinate frames at any desired point in time.

   - To make TF, type below command :
     `$ roslaunch robotis_humanoid_description robotis_humanoid.launch`

```
<launch>
  ......
A <param name="robot_description"
  command="$(find xacro)/xacro.py '$(find robotis_humanoid_description)/urdf/robotis_humanoid.xacro'" />

  <!-- send joint values -->
B <node name="robotis_joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher">
    <param name="use_gui" value="FALSE"/>
    <param name="rate" value="30"/>
    <remap from="/joint_states" to="/robotis/thor_3_0/joint_states" />
    <rosparam param="source_list" if="$(arg use_present)">["robotis/present_joint_states"]</rosparam>
    <rosparam param="source_list" unless="$(arg use_present)">["robotis/goal_joint_states"]</rosparam>
  </node>
  <!-- Combine joint values to TF-->
C <node name="robot_state_publisher" pkg="robot_state_publisher" type="state_publisher">
    <remap from="/joint_states" to="/robotis/thor_3_0/joint_states"/>
  </node>

  <!-- lidar (urg_node) -->
  <include file="$(find robotis_humanoid_description)/launch/thor_laserscan.launch" if="$(arg use_lidar)"/>
</launch>
```

3. How to run THORMANG3

- Run Packages in MPC (Motion PC)

4. TF and urg_node
   - Procedure of making TF

   A. Set Robot Model in the parameter server

   B. Set source joints angle

   C. Make TF in robot_state_publish node

   D. Check the robot model in Rviz

   - Urg_node : Lidar

```
<launch>
    <!-- lidar -->
    <node pkg="urg_node" type="urg_node" name="lidar_chest_urg_node" output="screen">
        <param name="frame_id" value="lidar_link" />
        <param name="ip_address" value="10.17.3.31" />
        <param name="publish_multiecho" value="false" />
        <!-- <param name="angle_min" value="-1.5" /> -->
        <!-- <param name="angle_max" value="1.5" /> -->
        <remap from="/scan" to="/robotis/sensor/scan" />
    </node>
</launch>
```

Remote Operation

4. How to control THORMANG3

- Run control Packages in OPC (Operation PC)

1. Synchronization time to PPC
   - Make the shell script as follows :

```
#! /bin/sh
sudo date --set='-2 secs‘
sudo ntpdate 10.17.3.35
sudo hwclock -w
```

   - Run the shell script

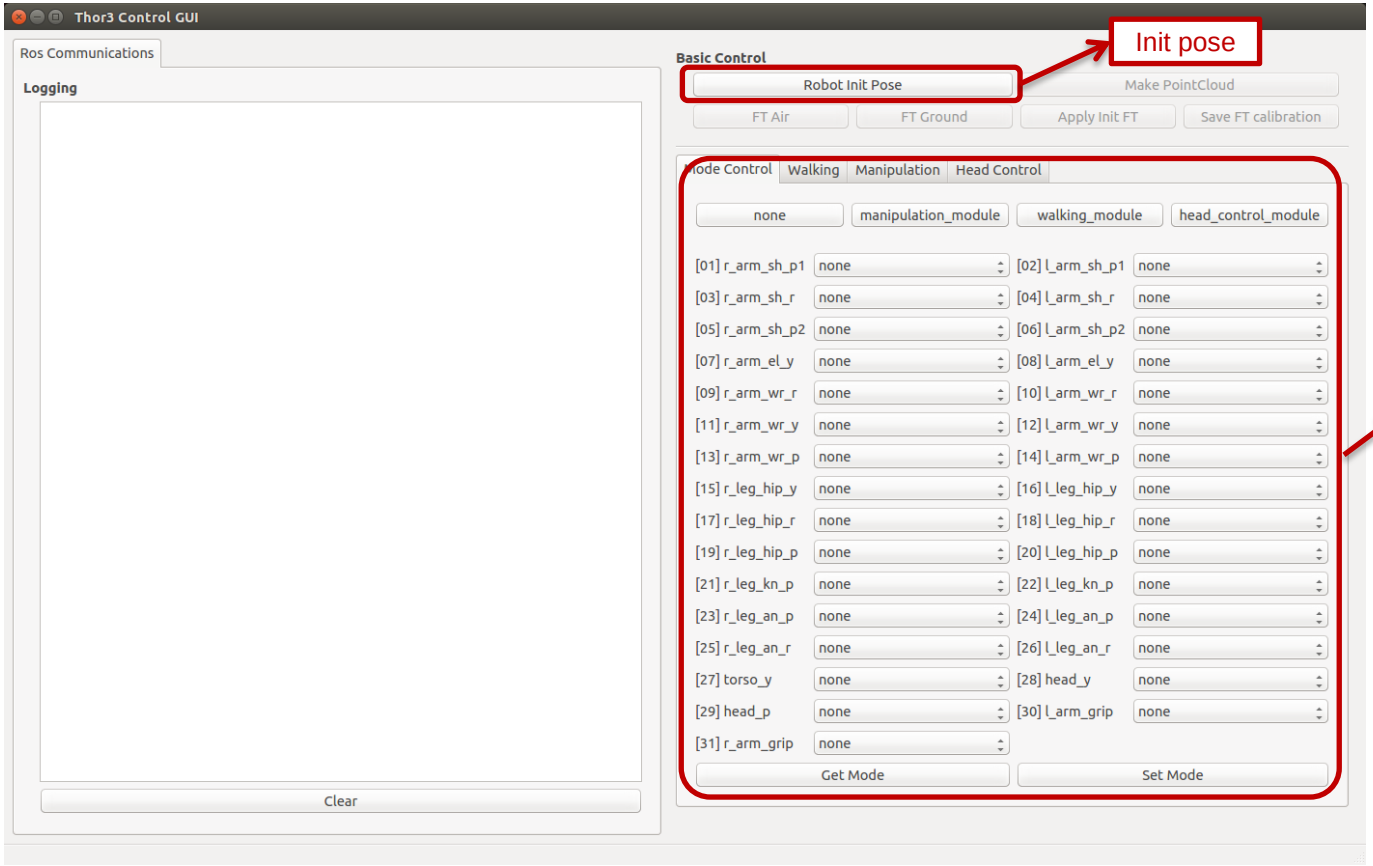
2. thormang3_demo
   thormang3_demo is THORMANG3’s demo kit using GUI interface.
   - To start thormang3 demo, simply run :
     `$ roslaunch thormang3_demo thormang3_demo.launch`
   - Break down launch file.

```
<launch>
    <param name="id_joint_table" value="$(find thormang3_demo)/data/id_joint_table.yaml"/>

    <!-- thormang3 GUI demo kit -->
    <node name="thormang3_demo_opc" pkg="thormang3_demo" type="thormang3_demo" output="screen">
        <remap from="/thormang3_demo/ik_target_pose" to="/pose_panel/pose" />
    </node>

    <!-- step generator for walking demo -->
    <node name="thormang3_foot_step_generator" pkg="thormang3_foot_step_generator"
     type="thormang3_foot_step_generator_node" output="screen"/>

</launch>
```

4. How to control THORMANG3

- Run control Packages in OPC (Operation PC)

1. thormang3_demo
   - Screenshot



4. How to control THORMANG3

- Run visualization Packages in OPC

2. Rviz

- Rivz is the 3D visualization tool for ROS. The user can check the state of robot and the value of sensors using Rviz.
- Wiki : http://wiki.ros.org/rviz
- How to run the Rviz in THORMANG3 :
  `$ roslaunch robotis_humanoid_description robotis_humanoid_control.launch`
- Break down the launch file.

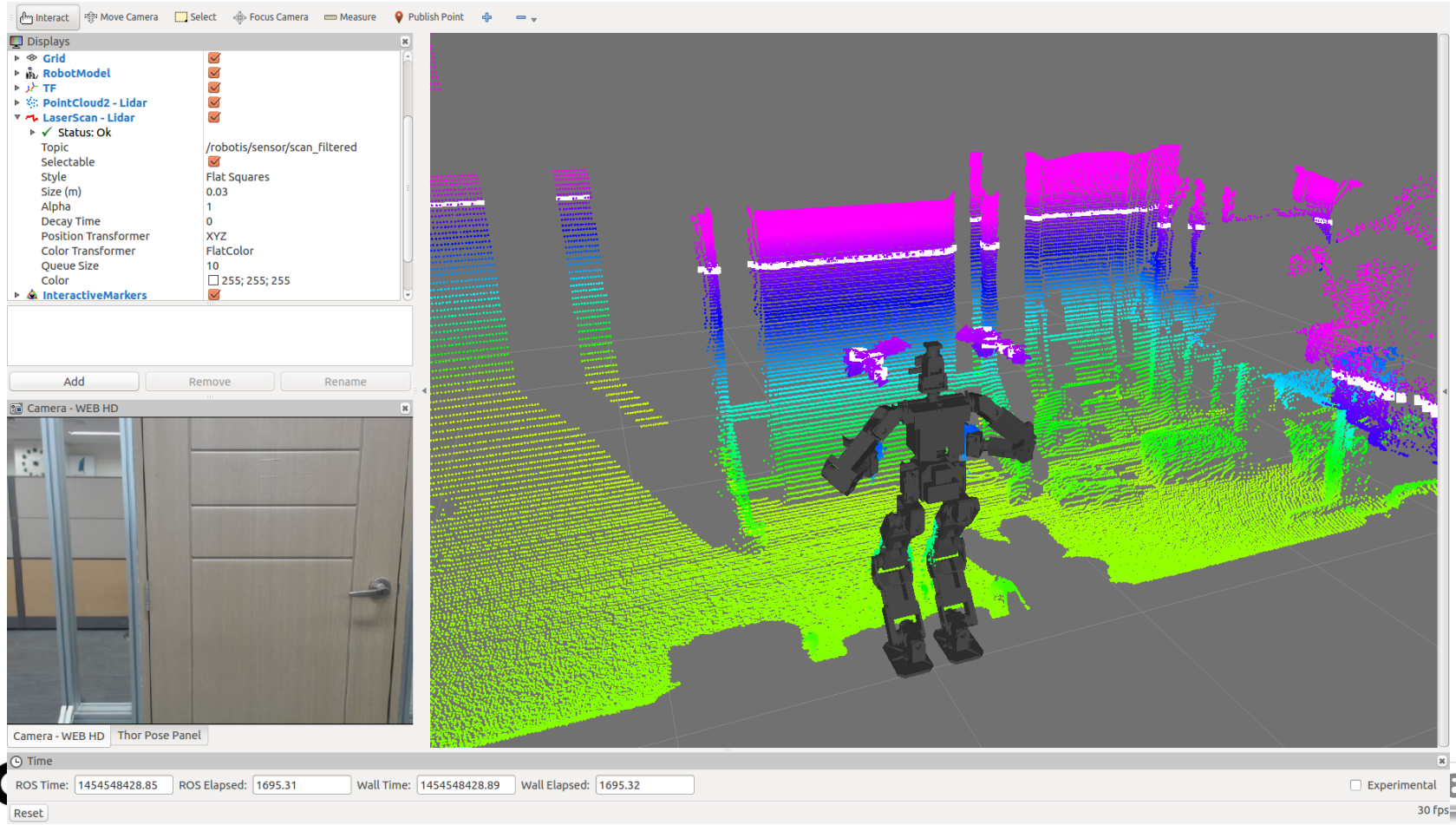
```
<launch>
  <!-- Robot Model -->
  <param name="robot_description"
  command="$(find xacro)/xacro.py '$(find robotis_humanoid_description)/urdf/robotis_humanoid.xacro'" />

  <!-- Show joint values -->
  <node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher">
    <param name="use_gui" value="TRUE"/>
    <remap from="/joint_states" to="/opc/thor_3_0/joint_states" />
    <rosparam param="source_list">["/robotis/thor_3_0/joint_states"]</rosparam>
  </node>

  <!-- Show in Rviz -->
  <node name="rviz" pkg="rviz" type="rviz"
  args="-d $(find robotis_humanoid_description)/launch/robotis_humanoid_display.rviz" output="screen"/>
</launch>
```

Remote Operation

4. How to control THORMANG3

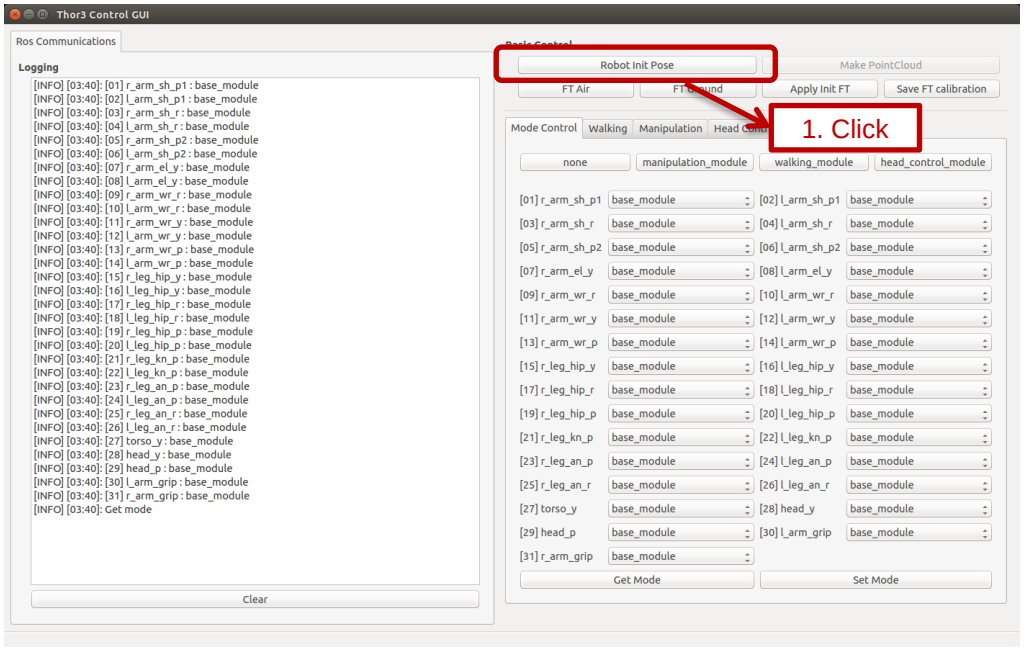
- Run visualization Packages in OPC

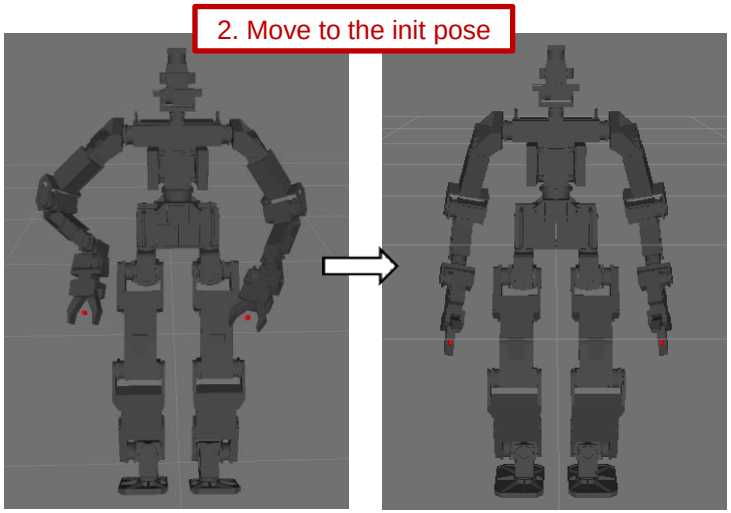
2. Rviz
   - Screenshot



5. GUI Demo : Basic demo

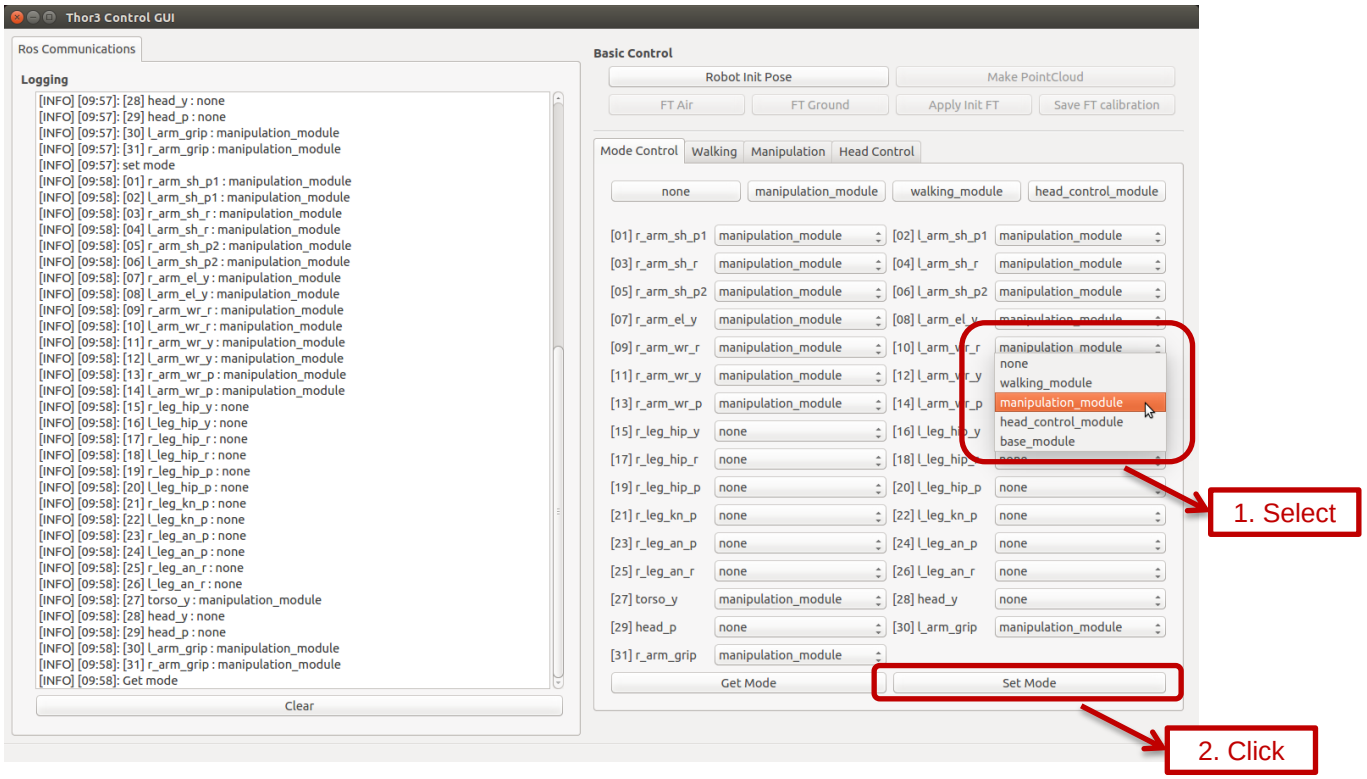
- Initial pose
  - If user click the button(Init Pose), the robot will move to initial pose.





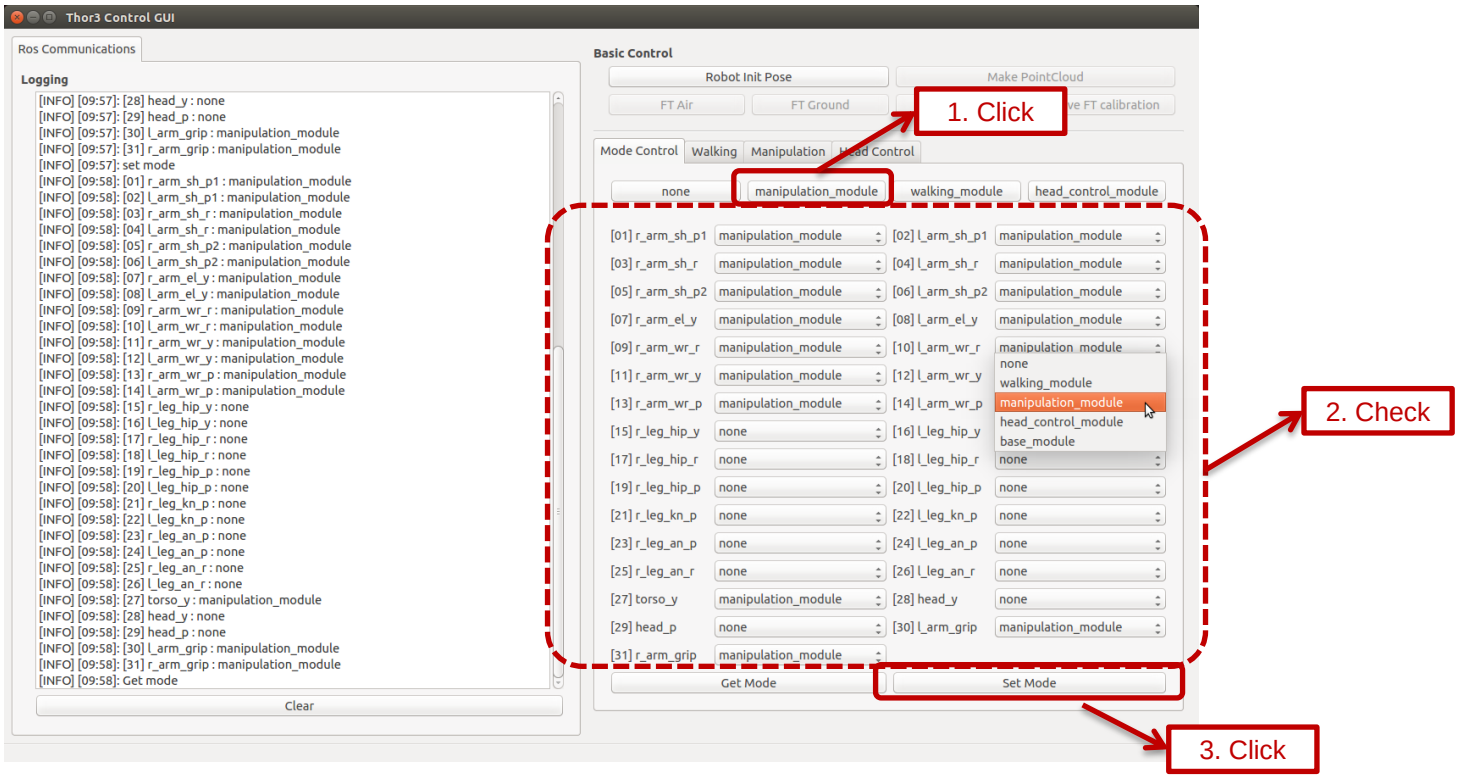
5. GUI Demo : Basic demo

- Setting the Control Module of Each Joint
  - Select the desired Control Module from the dropdown menu then click "**Set**"
  - Control Modules : Walking Module, Manipulation Module, HeadCon



5. GUI Demo : Basic demo

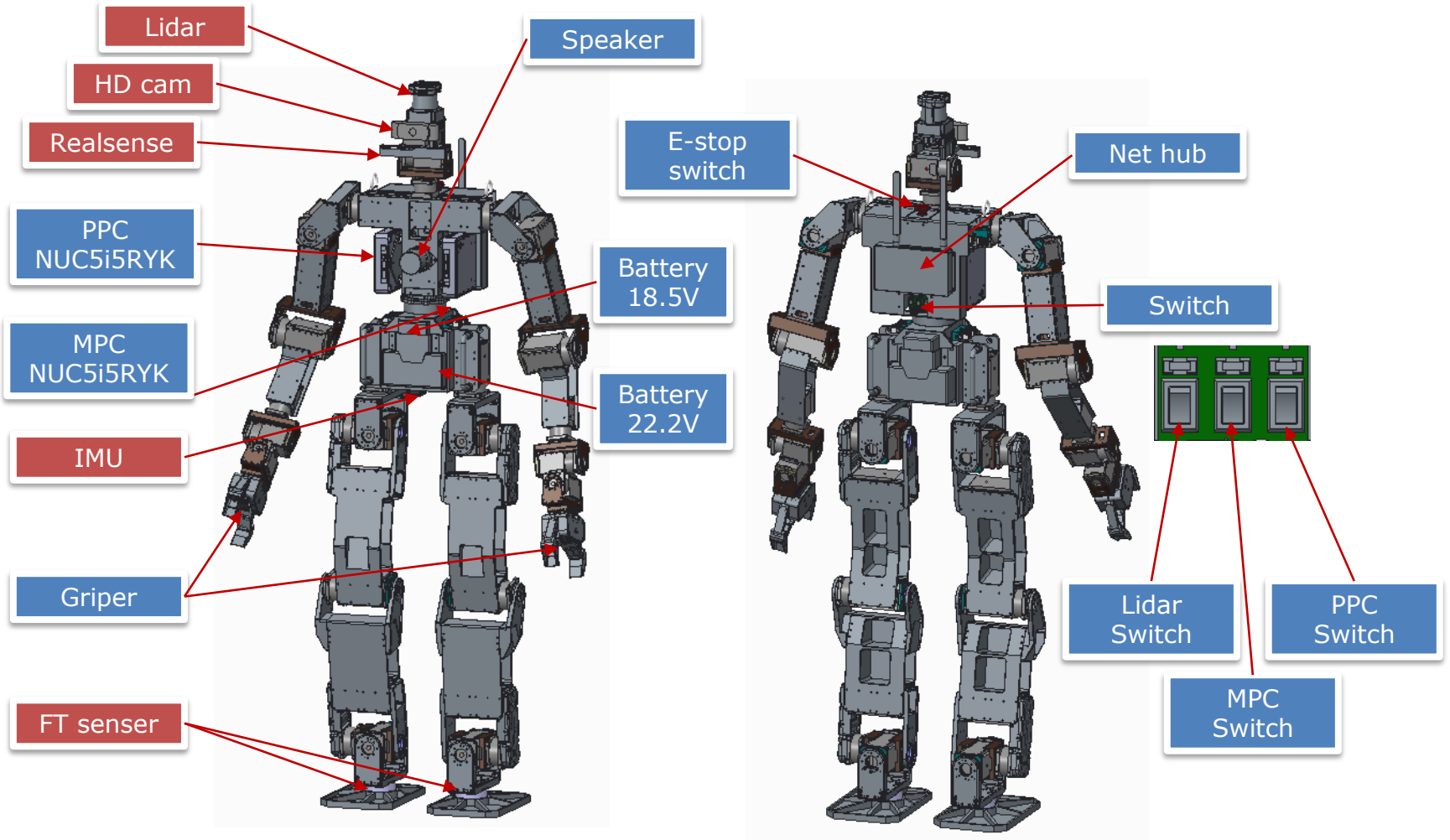
- Setting the Control Module using the preset button
  - Setting the control module using preset buttons
    Click the preset button on upper side. then click the button(set mode).



Sensors

CONTENTS

1. Description of the sensors



Sensor – HD Web Cam

1. Overview

1. HW : Logitech C920 HD
2. SW : uvc_camera (ROS package)
   - Wiki site : http://wiki.ros.org/uvc_camera
   - Installation (If user use ros-indigo version)

```
$ sudo apt-get install ros-indigo-uvc-camera
```

   - Note
     - **We assume that you're using ros indigo.**
     - **HD Web Camera is connected to the PPC(Perception PC). All commands should be typed in PPC.**

2. Topic List

	Name	Description
Topic	/robotis/sensor/camera/image_raw	A stream of images from the camera
	/robotis/sensor/camera/camrea_info	Camera intrinsics for images

3. How to run individual

```
$ roslaunch thor_sensors thor_web_cam.launch
```

Sensor – Depth camera (1)

1. Overview

1. HW : Intel Realsense R200
2. SW : RealSense_R200 (ROS package)
   - Wiki site : http://wiki.ros.org/RealSense_R200
   - Git : https://github.com/PercATI/RealSense_ROS
   - Installation : Go to the Wiki or the Git
   - Note
     - **We assume that you're using ros indigo.**
     - **Depth Camera is connected to the PPC(Perception PC). All commands should be typed in PPC.**

2. Topic List

	Name	Description
Topic	`/camera/color/image_raw`	A stream of color images from the camera
	`/camera/color/camrea_info`	Camera intrinsics for images

2. Topic List (cont.)

	Name	Description
Topic	/camera/depth/image_raw	A stream of depth images from the camera
	/camera/depth/points	Registered XYZRGB point cloud
	/camera/depth/uv	UV mapping for depth image
	/camera/depth/camera_info	Camera intrinsics for images

3. How to run individual

```
$ roslaunch thor_sensors thor_realsense.launch
```

Sensor – Lidar(option)

1. Overview

1. HW : Hokuyo UTM-30LX-EW
2. SW : urg_node (ROS package)
   - Wiki site : http://wiki.ros.org/urg_node
   - Installation (If user use ros-indigo version)

```
$ sudo apt-get install ros-indigo-urg-node
```

   - Note
     - **We assume that you're using ros indigo.**
     - **Lidar is connected to the MPC(Motion PC). All commands should be typed in MPC.**
     - **Lidar's IP address is 10.17.3.31**

2. Topic List

	Name	Description
Topic	/robotis/sensor/scan	single return output

3. How to run individual

```
$ roslaunch robotis_humanoid_description thor_laserscan.launch
```

Sensor – FT sensor

1. Overview

1. HW : ATI Mini58
2. SW : Thormang3 Manager
   - thormang3 manager publishes the ft sensors output on the feet

2. Topic List

	Name	Description
Topic	`/robotis/sensor/ft_right_foot/raw`	raw out put from ft sensor on right foot
	`/robotis/sensor/ft_right_foot/scaled`	scaled out put from ft sensor on right foot
	`/robotis/sensor/ft_left_foot/raw`	raw out put from ft sensor on left foot
	`/robotis/sensor/ft_left_foot/scaled`	scaled out put from ft sensor on left foot

Sensor – IMU

1. Overview

1. HW : MicroStrain 3DM-GX4-25
2. SW : imu_3dm_gx4
   - Wiki site : http://wiki.ros.org/imu_3dm_gx4
   - Git : https://github.com/KumarRobotics/imu_3dm_gx4
   - Installation : Please refer to above links.
   - Note
     - This pacakge should be located on MPC

2. Topic List

	Name	Description
Topic	/robotis/sensor/imu/imu	Gyro and Acceleration
	/robotis/sensor/imu/filter	Orientation and Gyro Bias

3. How to run individual

```
$  roslaunch imu_3dm_gx4 imu.launch
```

2. How to run the sensors in THORMANG3

- Refer to the previous section (Remote Operation / How to run THORMANG3).

3. Check the sensors

1. The sensors of MPC (IMU, FT, Lidar)
   - How to check the MPC's sensors
     - **IMU** : type the below command and check the messages.
       `$ rostopic echo /robotis/sensor/imu/imu`

```
robotis@mpc:~$ rostopic echo /robotis/sensor/imu/imu
header:
  seq: 218798
  stamp:
    secs: 1456227058
    nsecs: 657393447
  frame_id: imu
orientation:
  x: 0.551555931568
  y: 0.831686019897
  z: -0.0350182652473
  w: 0.0534624755383
orientation_covariance: [-1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
angular_velocity:
  x: 0.00529906712472
  y: 0.0256397109479
  z: 0.108881101012
angular_velocity_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
linear_acceleration:
  x: 1.40463167375
  y: 0.0995190255708
  z: 9.7412729802
linear_acceleration_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

Sensors

- How to check the MPC's sensors
  - **FT sensor** : type the below commands and check the messages.

```
$ rostopic echo /robotis/sensor/ft_right_foot/raw
$ rostopic echo /robotis/sensor/ft_left_foot/raw
```

```
robotis@mpc:~$ rostopic echo /robotis/sensor/ft_right_foot/raw
header:
  seq: 110484
  stamp:
    secs: 1456227469
    nsecs: 577939051
  frame_id: r_leg_foot_link
wrench:
  force:
    x: 5175.13464253
    y: 12796.9383106
    z: 9230.67538064
  torque:
    x: -134.298186423
    y: -234.033554692
    z: -267.399062596
```

```
robotis@mpc:~$ rostopic echo /robotis/sensor/ft_left_foot/raw
header:
  seq: 113669
  stamp:
    secs: 1456227506
    nsecs: 383800602
  frame_id: l_leg_foot_link
wrench:
  force:
    x: 9699.47838375
    y: 5305.99705173
    z: 10538.1527615
  torque:
    x: -243.133327243
    y: -325.658292323
    z: -202.514837491
```

Sensors

- How to check the MPC's sensors
  - **Lidar** : type the below command and check the messages.
    `$ rostopic echo /robotis/sensor/scan --noarr`

```
robotis@mpc:~$ rostopic echo /robotis/sensor/scan --noarr
header:
  seq: 403902
  stamp:
    secs: 1456227605
    nsecs: 320116247
  frame_id: lidar_link
angle_min: -2.35619449615
angle_max: 2.35619449615
angle_increment: 0.00436332309619
time_increment: 1.73611151695e-05
scan_time: 0.0250000003725
range_min: 0.0230000000447
range_max: 60.0
---
header:
  seq: 403903
  stamp:
    secs: 1456227605
    nsecs: 350245326
  frame_id: lidar_link
angle_min: -2.35619449615
angle_max: 2.35619449615
angle_increment: 0.00436332309619
time_increment: 1.73611151695e-05
scan_time: 0.0250000003725
range_min: 0.0230000000447
range_max: 60.0
```

Sensors

2. The sensors of PPC (Web-camera, Depth-camera)
   - How to check the PPC's sensors
     - **Web-camera** : type the below command and check the messages.
       `$ rostopic echo /robotis/sensor/camera/image_raw --noarr`

```
robotis@ppc:~$ rostopic echo /robotis/sensor/camera/image_raw --noarr
header:
  seq: 23
  stamp:
    secs: 1456228631
    nsecs: 803098057
  frame_id: head_p_link
height: 480
width: 640
encoding: rgb8
is_bigendian: 0
step: 1920
---
header:
  seq: 24
  stamp:
    secs: 1456228631
    nsecs: 903091604
  frame_id: head_p_link
height: 480
width: 640
encoding: rgb8
is_bigendian: 0
step: 1920
```

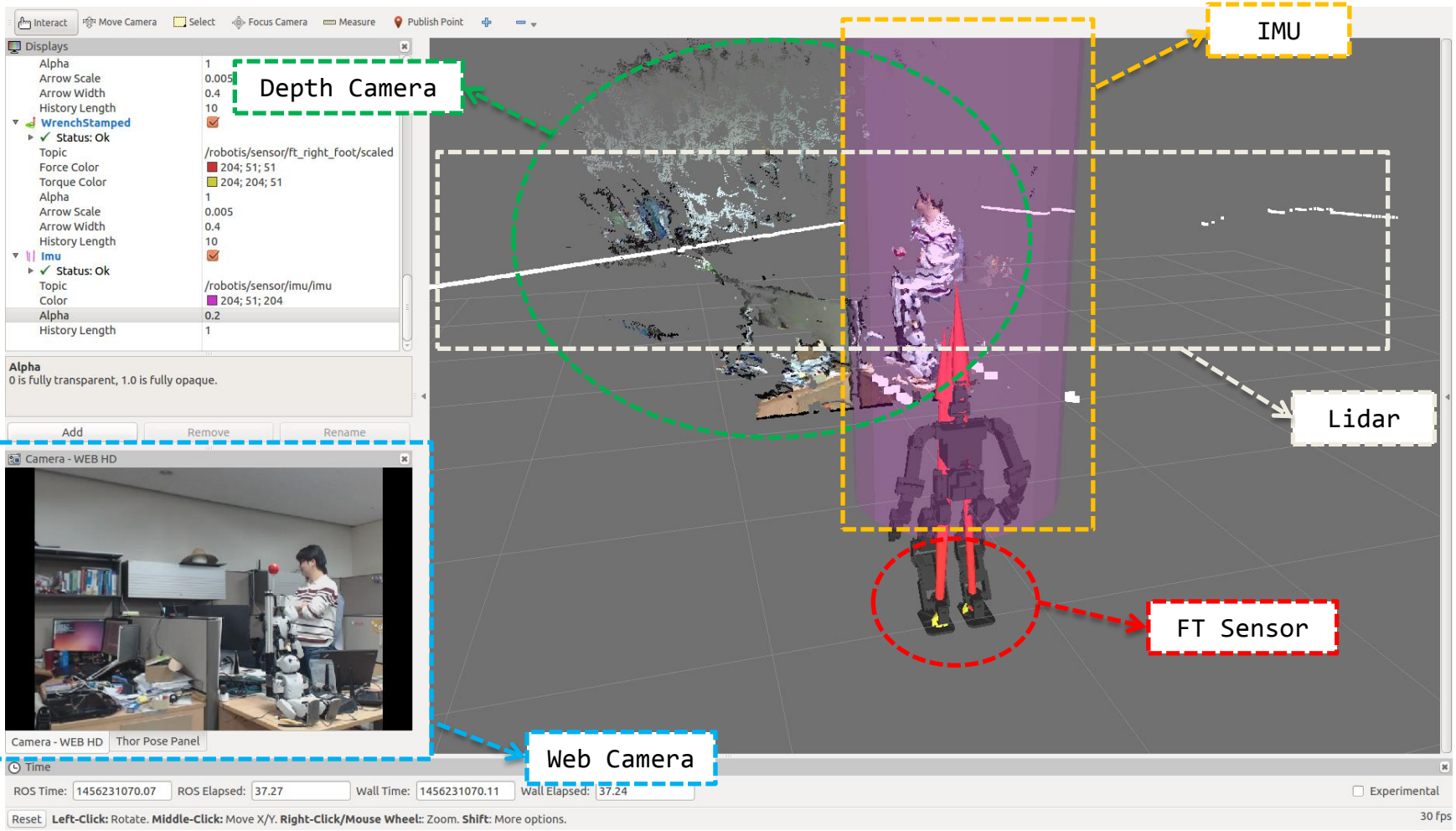
How to check the sensors (10)

- How to check the PPC's sensors
  - **Depth-camera** : type the below command and check the messages.
    `$ rostopic echo /camera/depth/points --noarr`

```
robotis@ppc:~$ rostopic echo /camera/depth/points --noarr
header:
  seq: 10430
  stamp:
    secs: 1456228679
    nsecs: 180915493
  frame_id: camera_depth_optical_frame
height: 360
width: 480
is_bigendian: False
point_step: 32
row_step: 15360
is_dense: True
---
header:
  seq: 10431
  stamp:
    secs: 1456228679
    nsecs: 199269937
  frame_id: camera_depth_optical_frame
height: 360
width: 480
is_bigendian: False
point_step: 32
row_step: 15360
is_dense: True
```

Visualization

- How to check the THOR-MANG3's sensors in Rviz

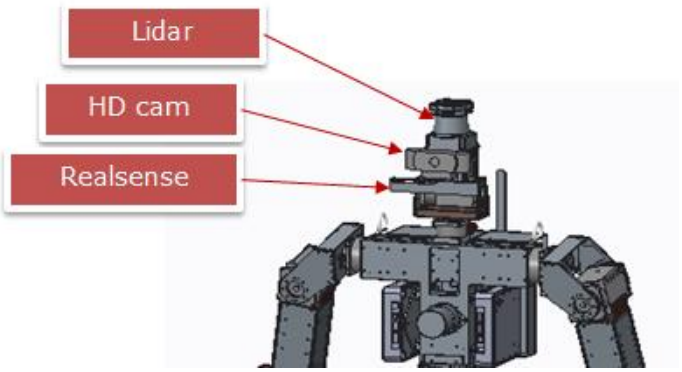


Vision

CONTENTS

1. Overview

THORMANG3 is mounted 3 vision sensors(HD cam, Realsense, Lidar) on head.



User can control the direction of the vision sensors using Head Control Module of THORMANG3 Manager.

CONTENTS

2. Head Control Module

▪ Overview

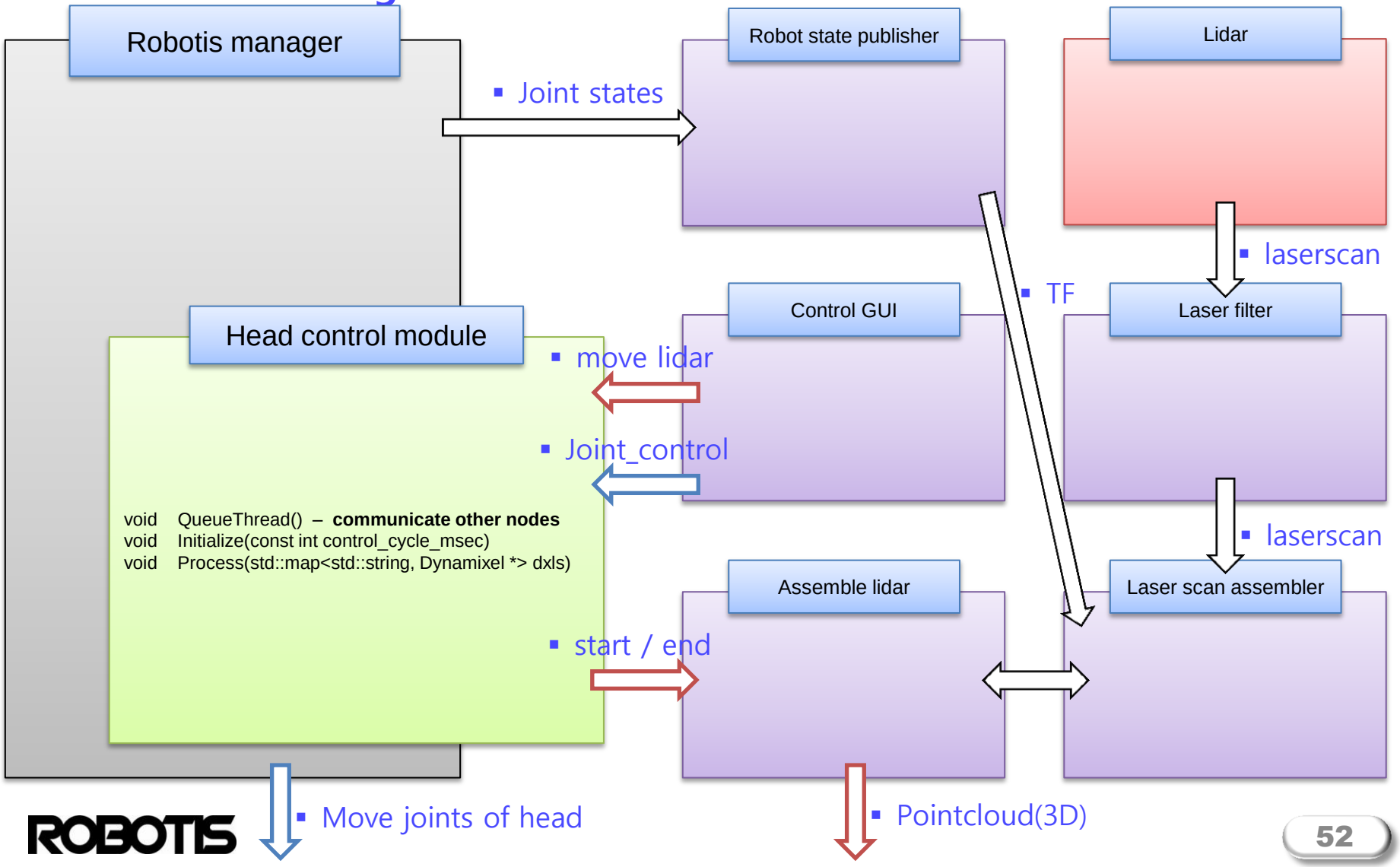
1. Move head(head yaw joint) with lidar to assemble pointcloud (laserscan to pointcloud)

2. Control individual Head joints(head yaw joint, head pitch joint)

3. Move head to the requested direction of user
   - Scheduled

Head control Module (3)

Structure Diagram



Head control Module (4)

- **Topic and Service List**

	Name	Description
Topic (Publish)	`/robotis/sensor/move_lidar`	Send a topic(start/end time of movement) to the assemble_lidar_node
Topic (Subscribe)	`/robotis/head_control/move_lidar`	Command to move head pitch joint from user
	`/robotis/head_control/joint_control`	Command to move joints of head

Head control Module (5)

Topic and Service List

1. Topic(Publisher)

- `/robotis/sensor/move_lidar (std_msgs/String)`

Variable Type	Variable Name	Description
`String`	`data`	"start" or "end" message

2. Topic(Subscriber)

- `/robotis/head_control/move_lidar (std_msgs/String)`

Variable Type	Variable Name	Description
`String`	`data`	Move head to assemble 3d point cloud

Head control Module (6)

- **Topic and Service List**

2. Topic(Subscriber)

- `/robotis/head_control/move_lidar (sensor_msgs/JointState)`

Variable Type	Variable Name	Description
`std_msgs/Header`	`header`	The header specifies the time
`string[]`	`name`	Joint name
`float64[]`	`position`	The position of the joint(rad or m)
`float64[]`	`velocity`	The velocity of the joint(rad/s or m/s)
`float64[]`	`effort`	The effort that is applied in the joint(Nm or N)

Control Head Direction

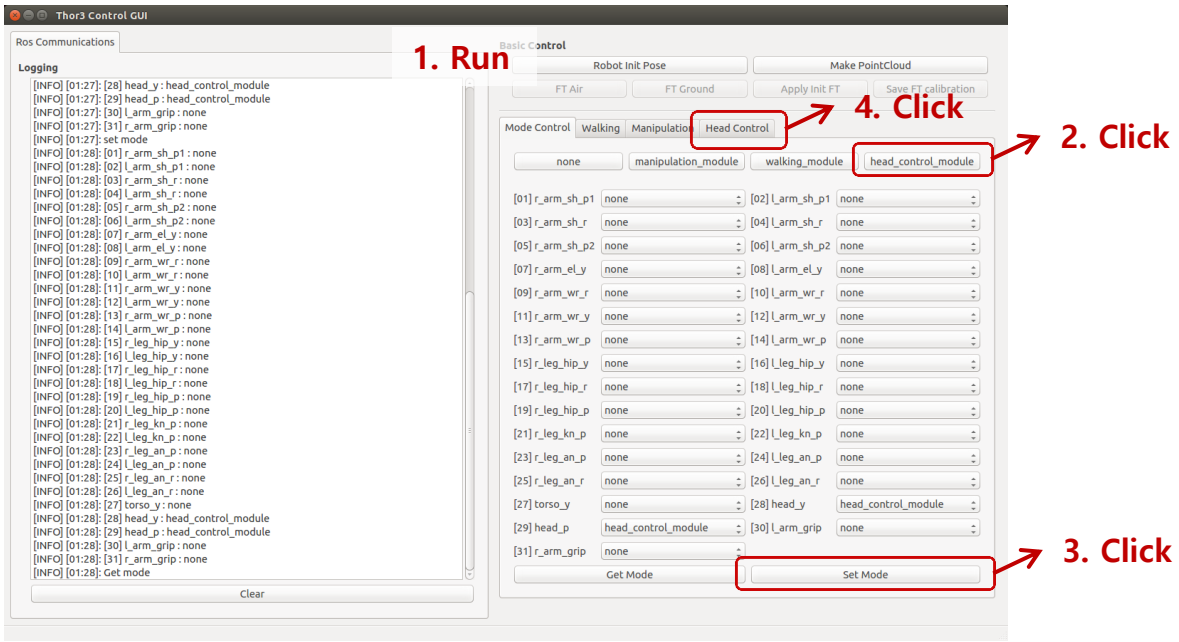
3. 2D Vision

- Control Head Direction to move the sensors of head(Web-cam, Depth-cam)

1. Run the user gui program in OPC.

```
$ roslaunch thormang3_demo thormang3_demo.launch
```

2. Set mode : head_control_module
   1. Click '**head_control_module**' preset button
   2. Click '**Set Mode**' button
3. Change control tab to "Head Control"

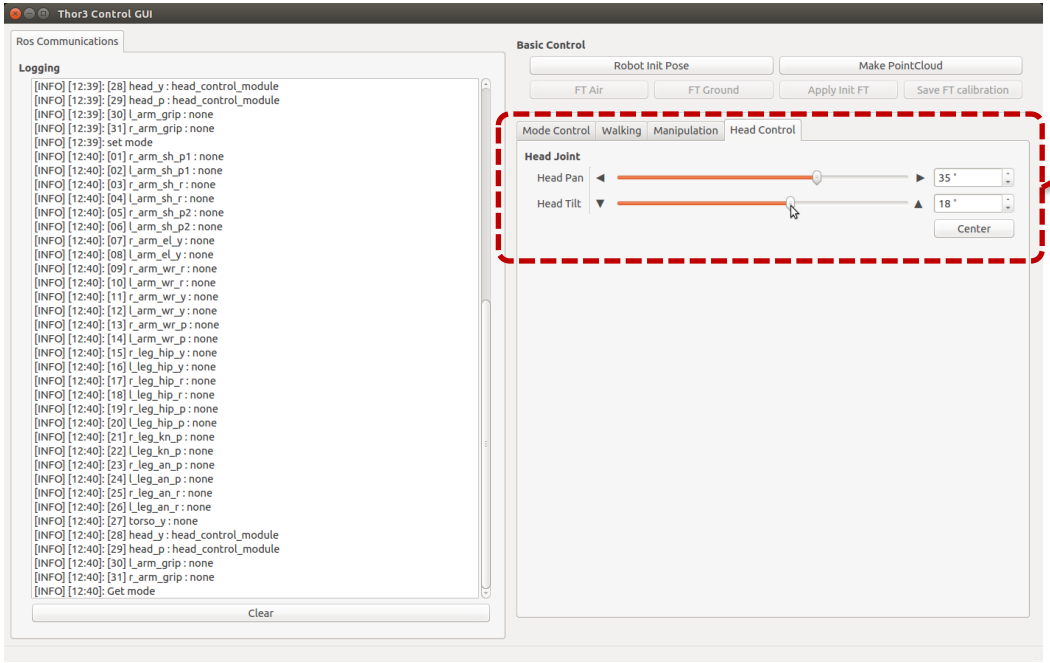


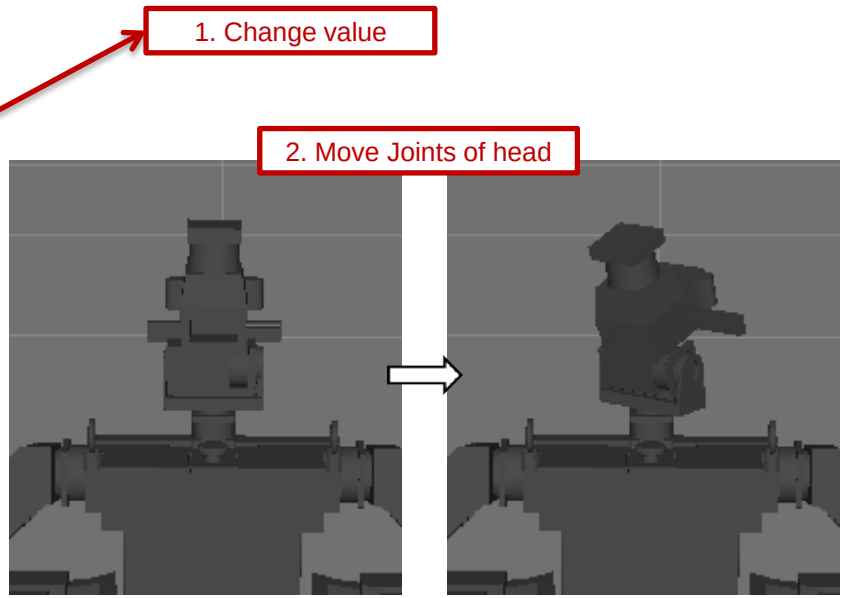
Control Head Direction

3. 2D Vision

- Control Head Direction to move the sensors of head(Web-cam, Depth-cam)

4. Move slide bar to the direction or Change angle value of joints

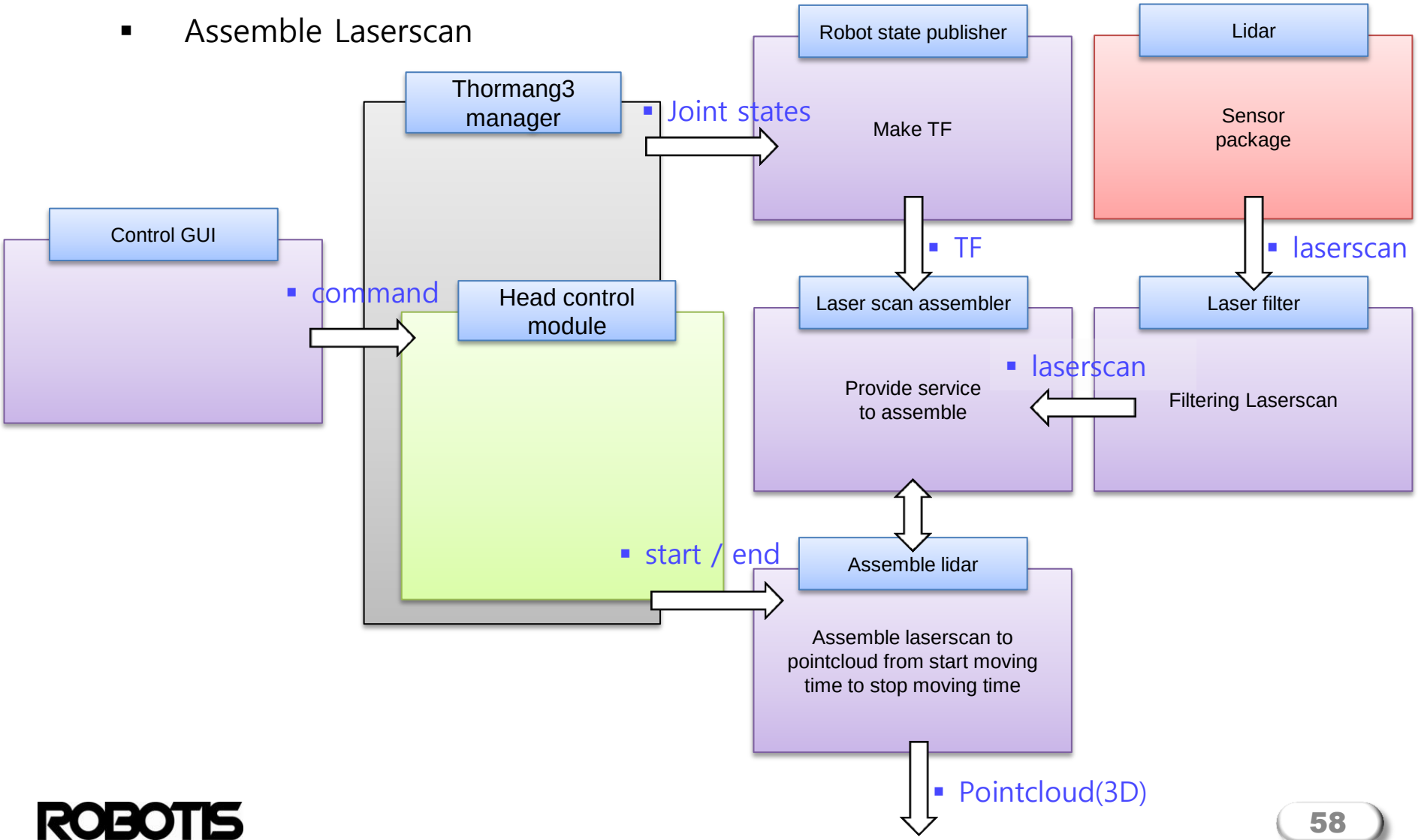




Assemble Laserscan

4. 3D Vision

- Assemble Laserscan



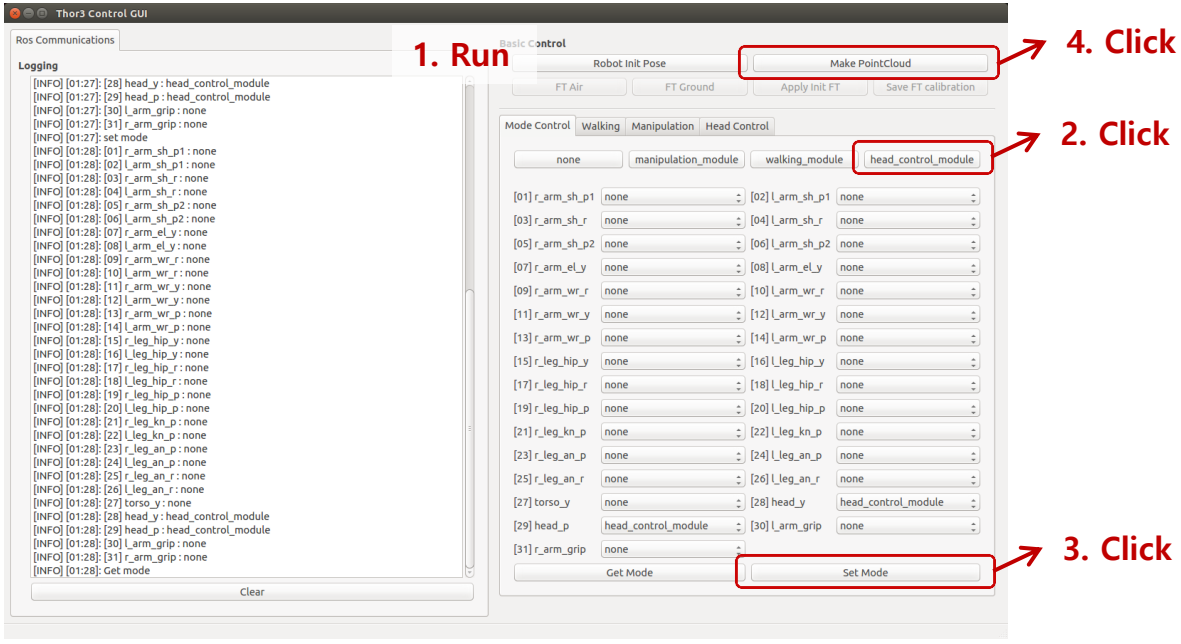
4. 3D Vision

- Assemble Laserscan

1. Head Control Demo : How to Move head pitch joint to assemble pointcloud (2d laserscan to 3d pointcloud)
   1. Run the user gui program.

```
$ roslaunch thormang3_demo thormang3_demo.launch
```

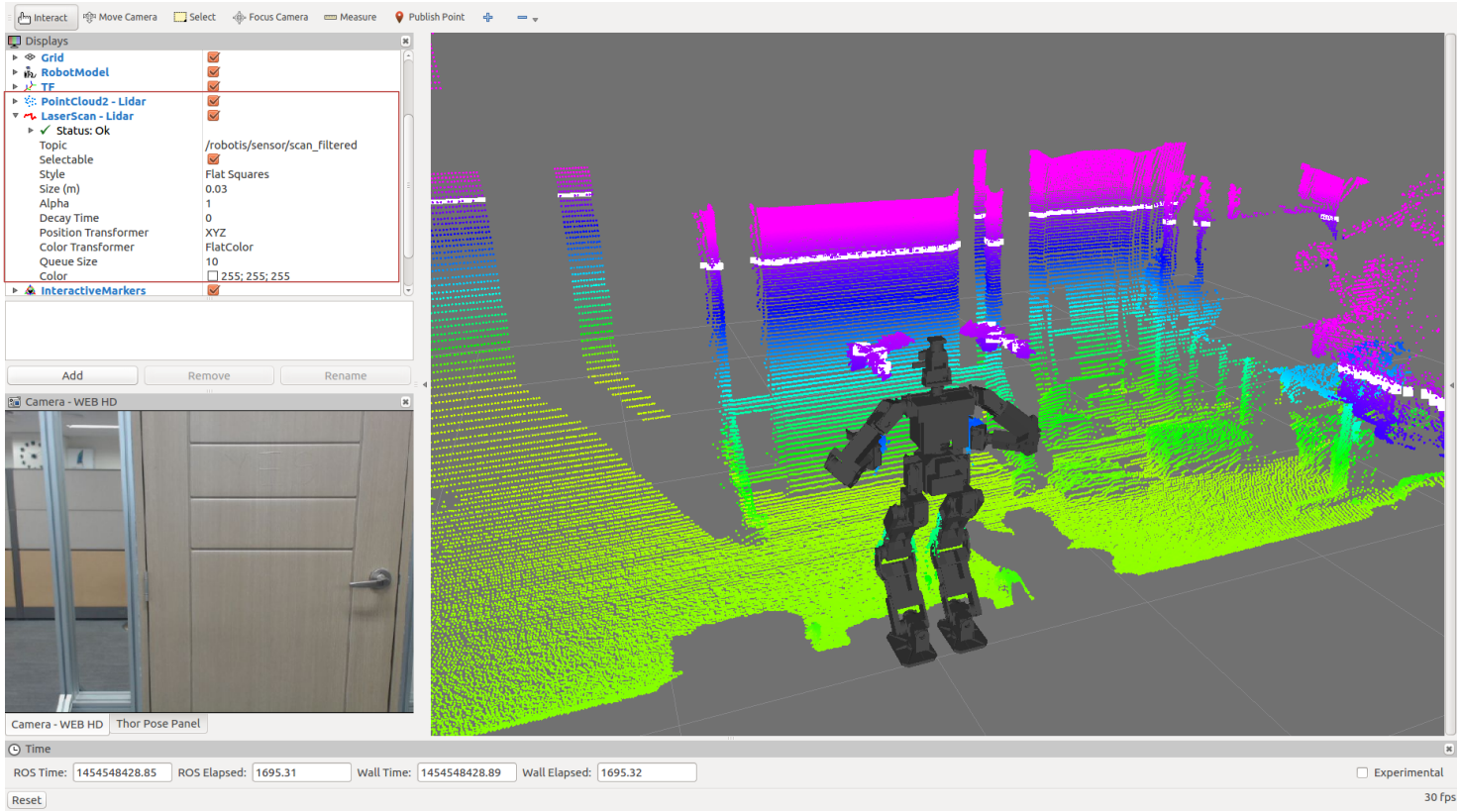
   2. Set mode : head_control_module
      1. Click '**head_control_module**' preset button
      2. Click '**Set Mode**' button
   3. Click '**Make PointCloud**' button



4. 3D Vision

1. Head Control Demo : How to Move head pitch joint to assemble pointcloud (2d laserscan to 3d pointcloud)

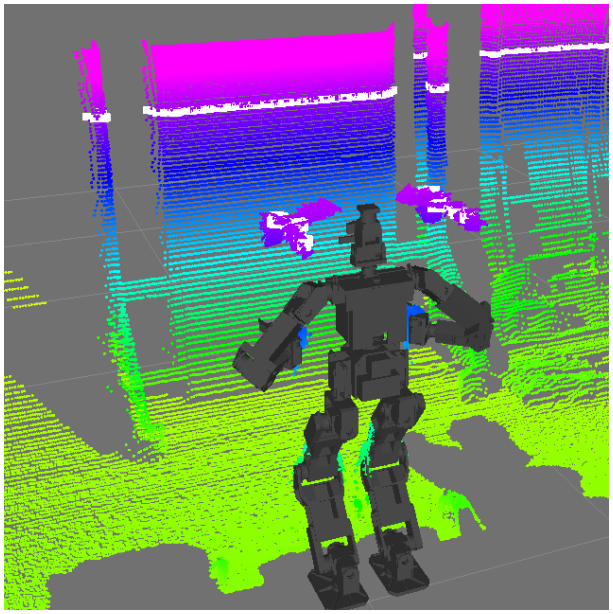
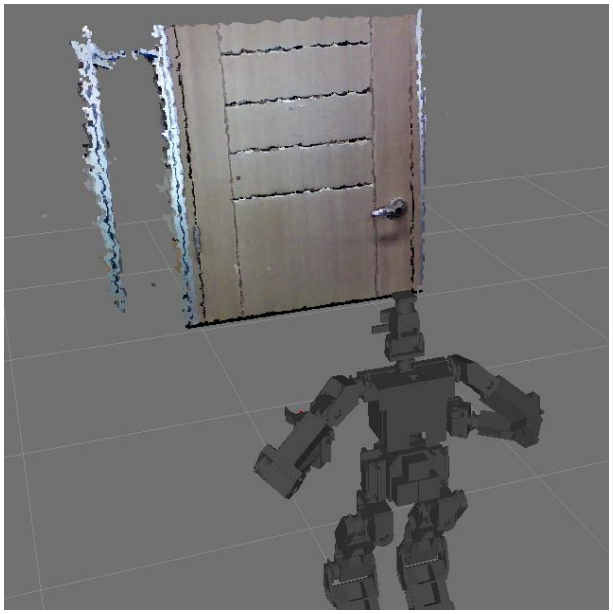
4. PointCloud(Rainbow lines) and LaserScan(the white line) Image



Assembled Lidar data vs Depth-Camera

4. 3D Vision

- Assembled Lidar data vs Depth-Camera

Assembled Laser data		Depth Camera	
Pros	**Cons**	**Pros**	**Cons**
An elaborate data	Need to scan in a stopped state	Real-time	An inaccurate data
It's possible to change the accuracy and time	Need command from user	It is automatically generated	A narrow range
It's possible to change the range	Need time to assemble		
			

Get 3D Pose in Rviz

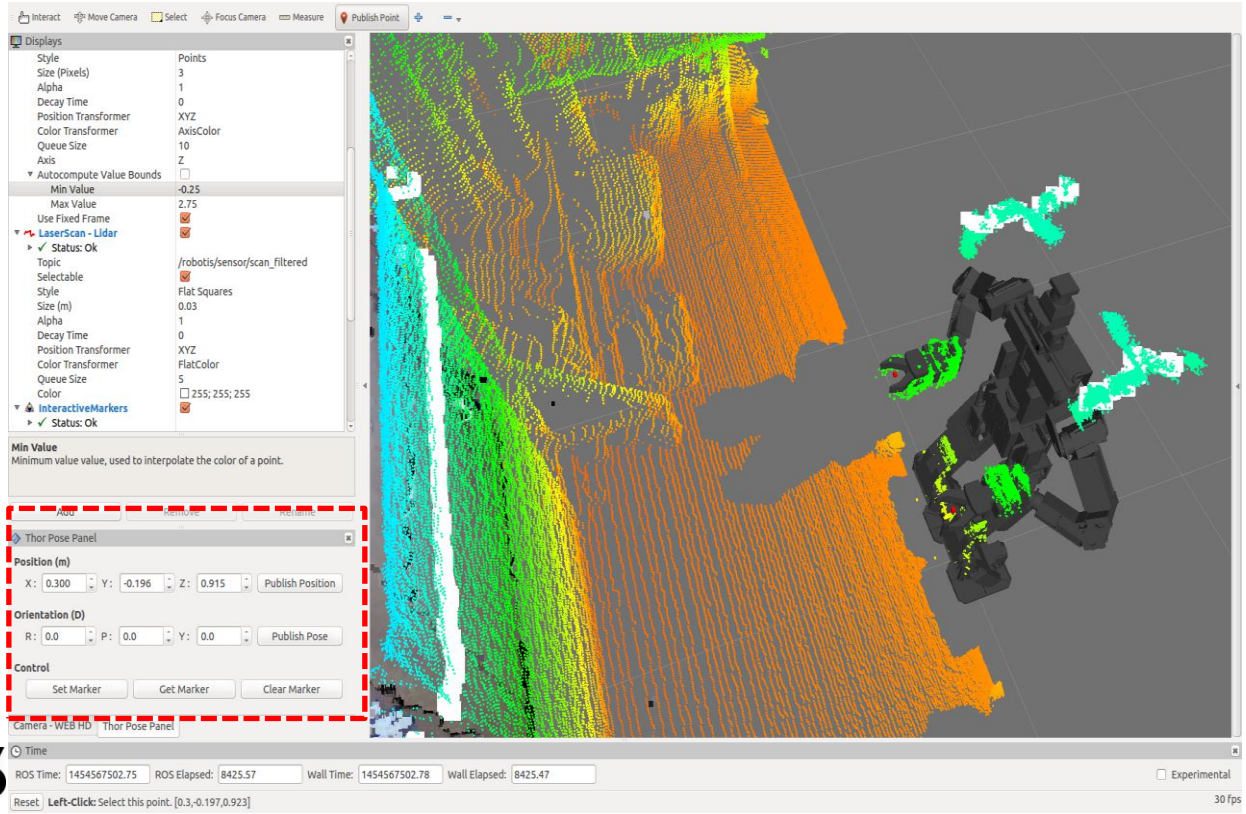
4. 3D Vision

- Get 3D Pose in Rviz

1. Run Rviz with robot model on OPC(Operating PC)

```
$ roslaunch robotis_humanoid_description robotis_humanoid_display.launch
```

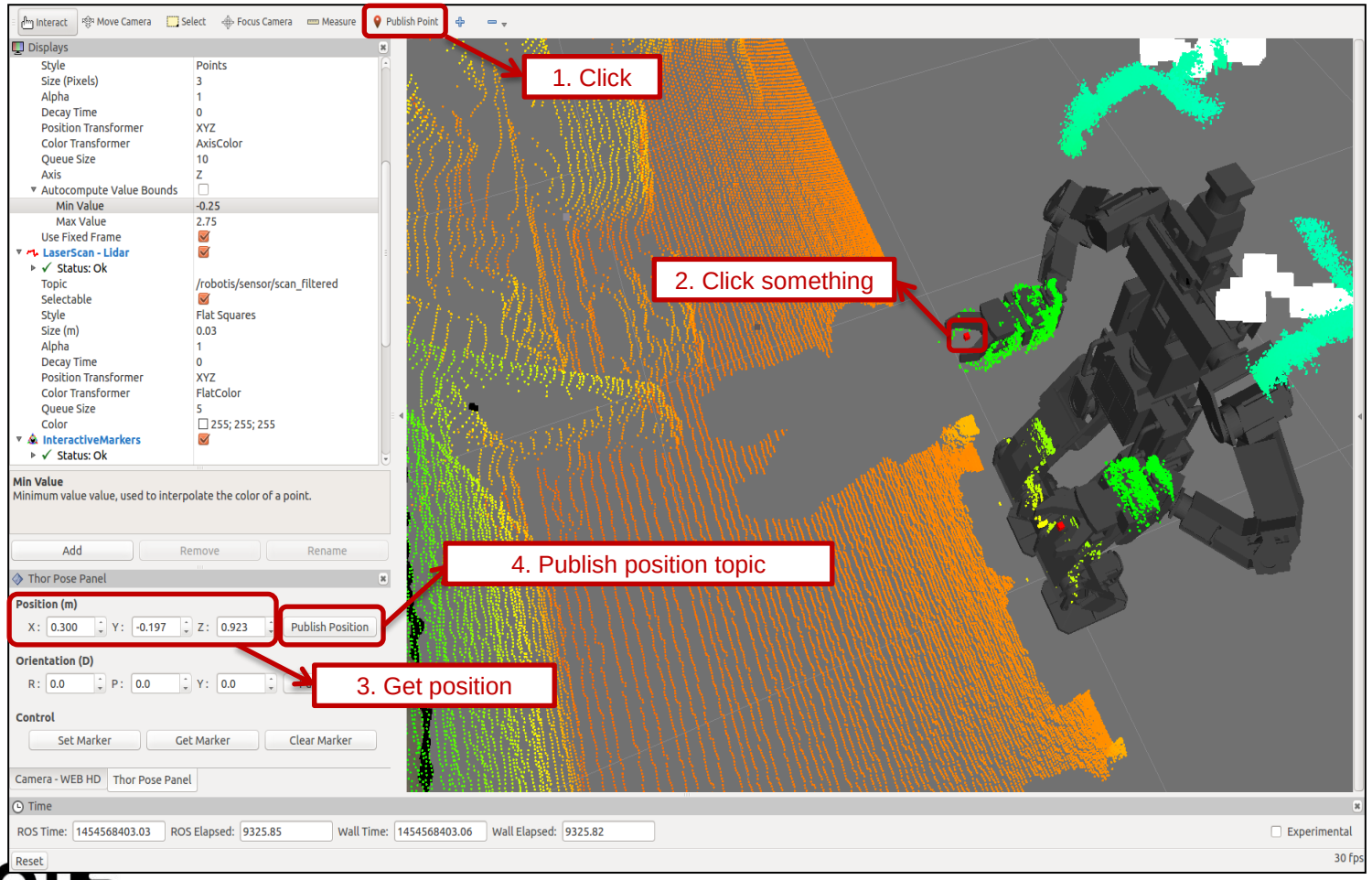
2. Go to the "Thor Pose Panel". If you don't find "Thor Pose Panel", Add it(menu -> Panels -> Add New Panel : rviz_thor3/Thor Pose Panel).



Get 3D Pose in Rviz

3. How to get a 3D point(position) in Rviz

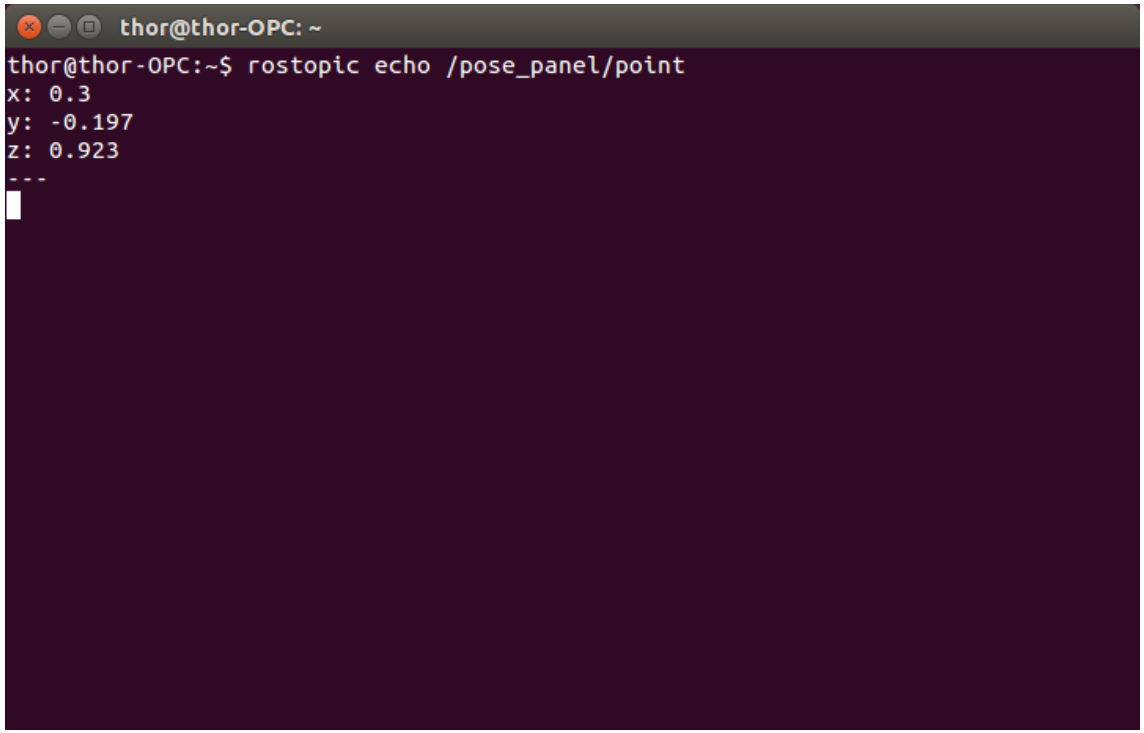
- If user click the button(Publish Point) in the upper panel and click the point in the 3D screen, a clicked point's position will be shown in the "Thor Pose Panel".



Get 3D Pose in Rviz

3. How to get a 3D point in Rviz

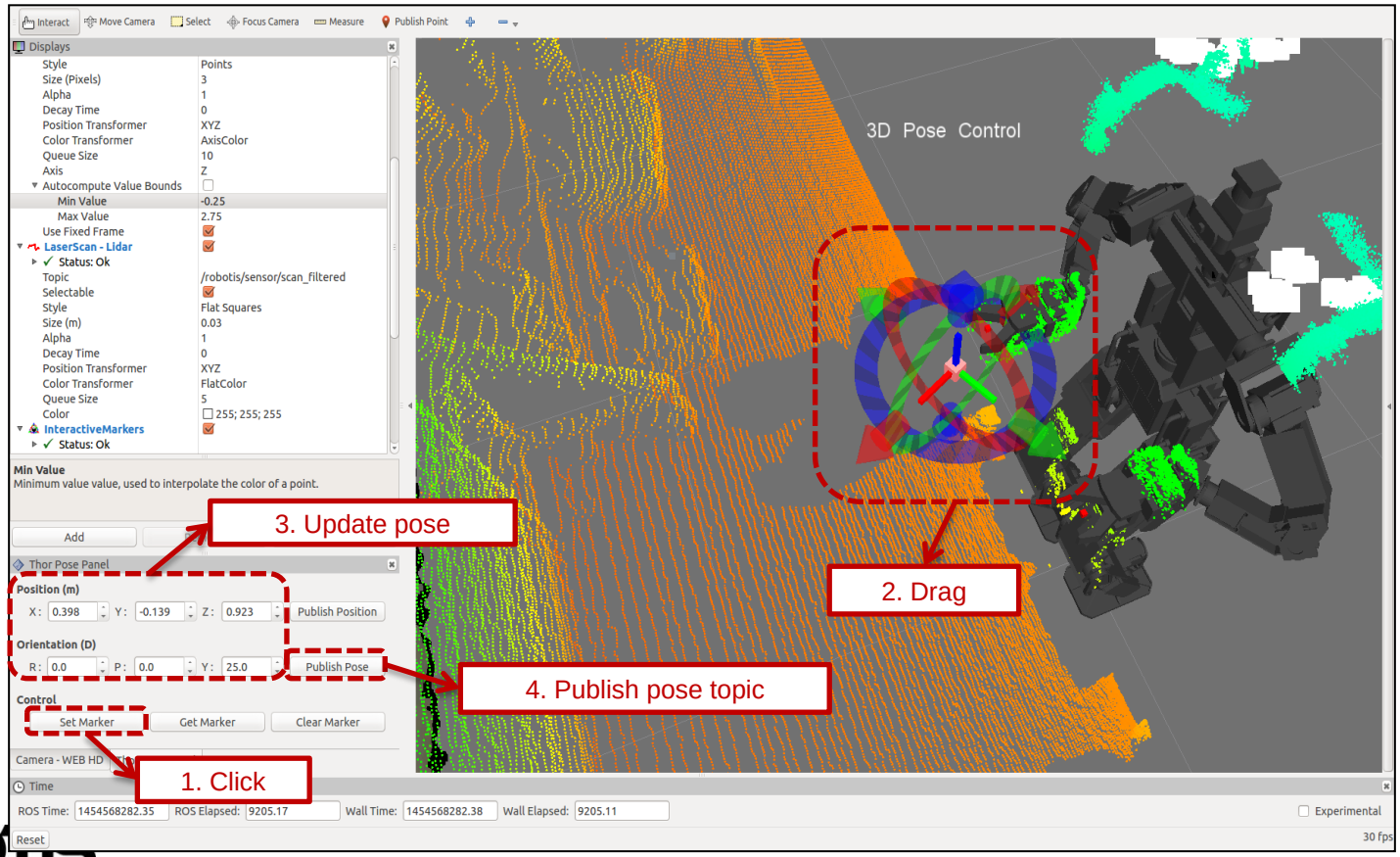
- If user click the “publish position” button, the position topic will be published.



Get 3D Pose in Rviz

4. How to get a 3D pose(position, orientation) in Rviz

- If user click the button(Set Marker) in the "Thor Pose Panel" after getting a position, the interactive marker will be shown in the 3d screen.
- It is able to get the pose(position and orientation) of the marker while dragging the marker.



Get 3D Pose in Rviz

4. How to get a 3D pose(position, orientation) in Rviz

- If user click the “publish position” button, the pose topic will be published.