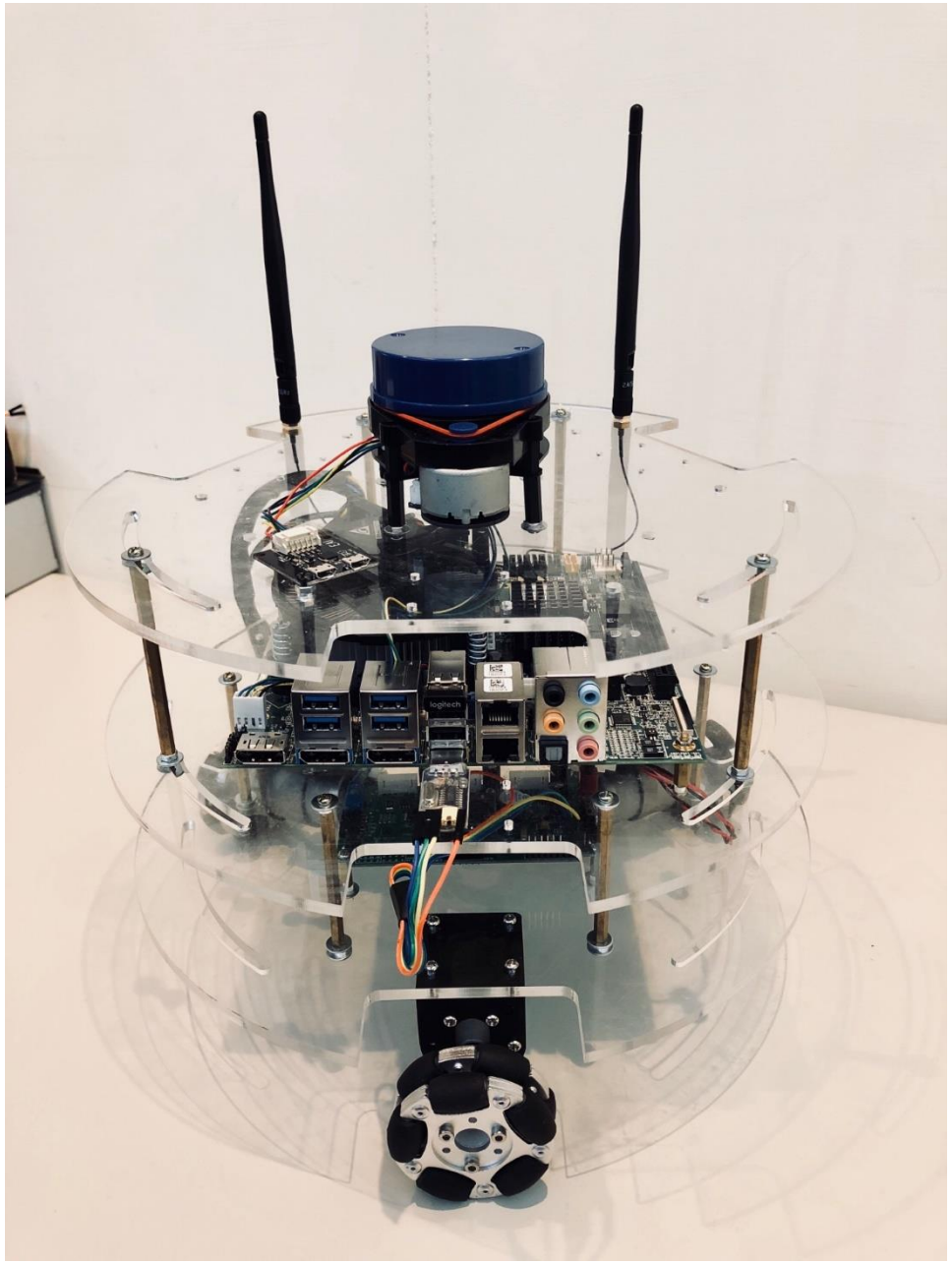
Introduction and implementation of ADLINK Neuron OmniBot

Advanced Robotic Platform Group,
HIM Segment, ADLINK Technology Inc.
2018/08/21
康心奕 Hsin-Yi Kang

ADLINK
Building Forward Together

Outline

- Introduction to OmniBot
  - Vehicle hardware
  - Fundamental package (software) environment
- Multi-Machine setup
- ROS implementation
  - (0) Prerequisite
  - (A) Hardware IO and baseline controller
  - (B) Laser SLAM
  - (C) Vehicle localization against known map
  - (D) Navigation



Project page

https://github.com/Adlink-ROS/Neuron-OmniBot

Introduction to OmniBot

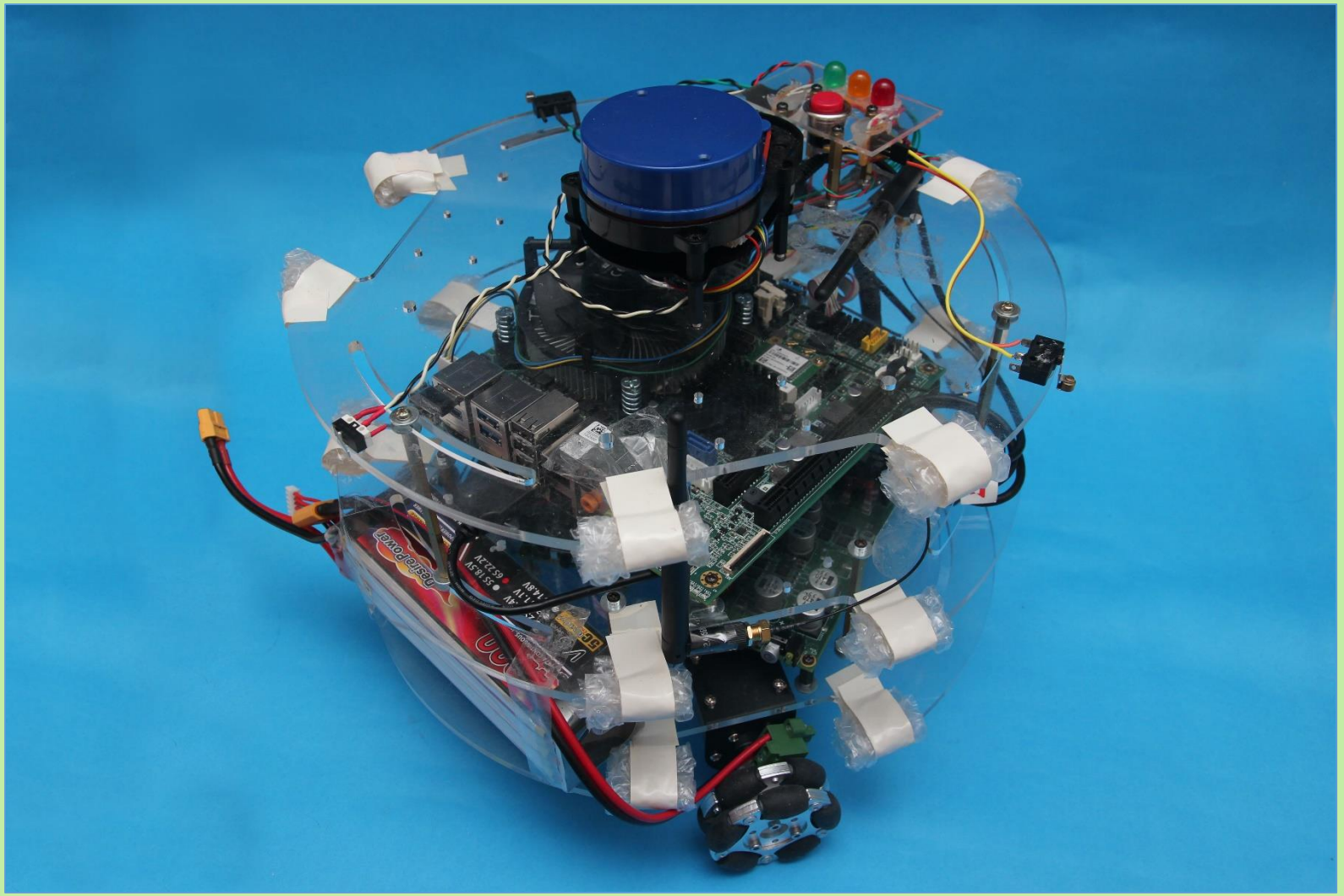
Introduction to OmniBot

Vehicle hardware

- Actuators
- VDMC
- Battery
- Lidar
- Wiring

Introduction to OmniBot

Vehicle hardware



Introduction to OmniBot

Vehicle hardware: Actuators

- Wheel
  - Three omni-directional poly wheels
  - 12V 360mA 4.32 watt DC motor
  - Blocking torque 10kg·cm @ 2.8A
  - 30x reduction gearbox
- Quadrature hall effect encoder
  - 13 poles providing 390 pulses (1560 count) per wheel revolution

Introduction to OmniBot

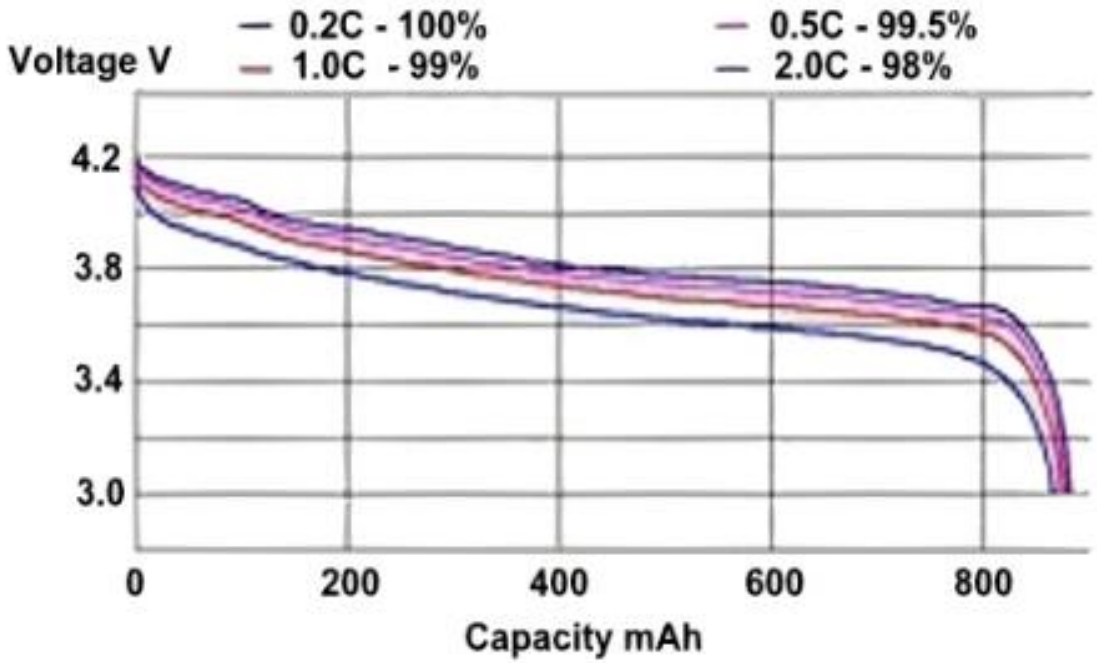
Vehicle hardware: VDMC

- Vehicle Dynamics and Motor Controller (VDMC)
- MCU: STM32F103C8T6
  - w/ fully customized VDMC firmware
- MPU6050 6-axis IMU: Gyro / Accelerometer
- A4950 DMOS Full-Bridge motor driver IC
  - PWM / 40V / +- 3.5A
- AMS1117 voltage regulator: 5V/ 3V3
- 128x64 px 0.96” OLED screen
- IO: CAN / UART*2 / SWD

Introduction to OmniBot

Vehicle hardware: Battery

- Lithium-Polymer (LiPo)
  - Lithium polymer battery
  - 3.7V - 4.2V
  - No memory effect
  - High power/weight ratio: ~0.2 kWh/kg
  - Highly explosive
  - Sensitive to voltage
- 6S
- Capacity:4200 mAh
- Discharge: 35C



CAUSION:

https://www.youtube.com/watch?v=IlhEgQ1Uyo8

http://www.ibt-power.com/Battery_packs/Li_Polymer/Lithium_polymer_tech.html

Introduction to OmniBot

Vehicle hardware: Battery

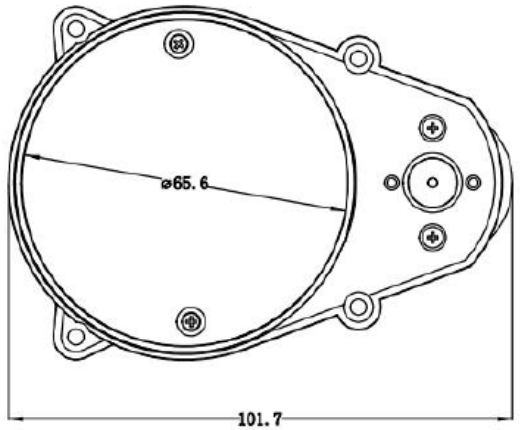
- Lithium-Polymer (LiPo)
  - Lithium polymer battery
  - 3.7V - 4.2V
  - No memory effect
  - High power/weight ratio: ~0.2 kWh/kg
  - Highly explosive
  - Sensitive to voltage
- 6S
- Capacity:4200 mAh
- Discharge: 35C
- **YOU WILL NEED**: Balanced charger(平衡充電器) & cell monitor (分壓測電表)

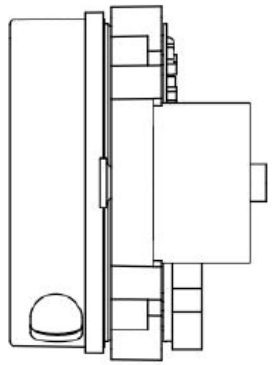
source: https://goods.ruten.com.tw/item/show?21309093087800

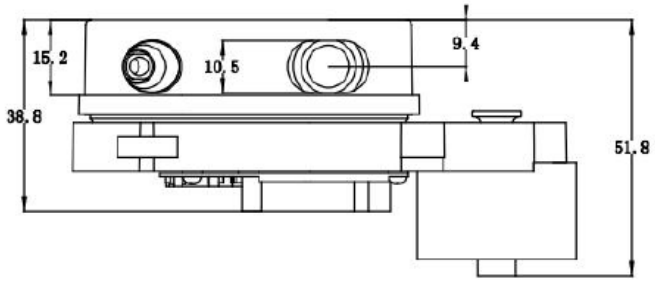
Introduction to OmniBot

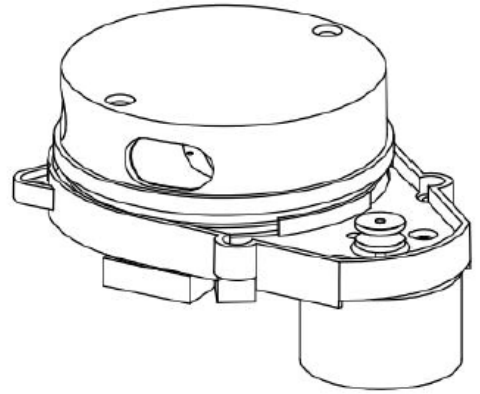
Vehicle hardware: Lidar

- Scanning Range:
  - Angle: 2π
  - Distance: 0.12 to 10 m
- Sampling rate: 5000 Hz
- Revolution rate: 6 to 12 RPS (no
- Resolution
  - <2m: 0.5mm
  - >2m: 1%
- Method: triangulation
- Laser: 785mm, 3mW





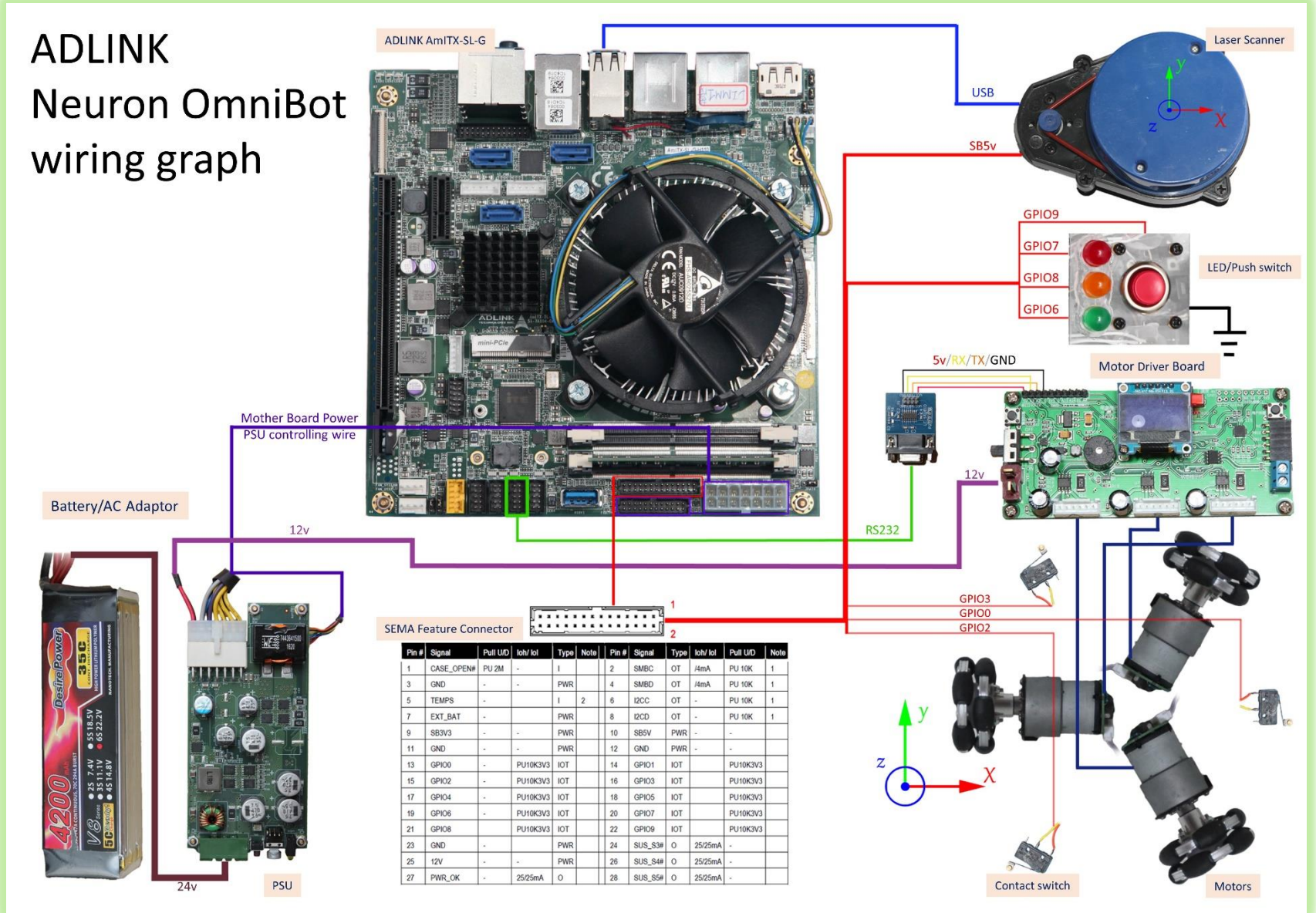




source: YDLidar X4 datasheet

Introduction to OmniBot

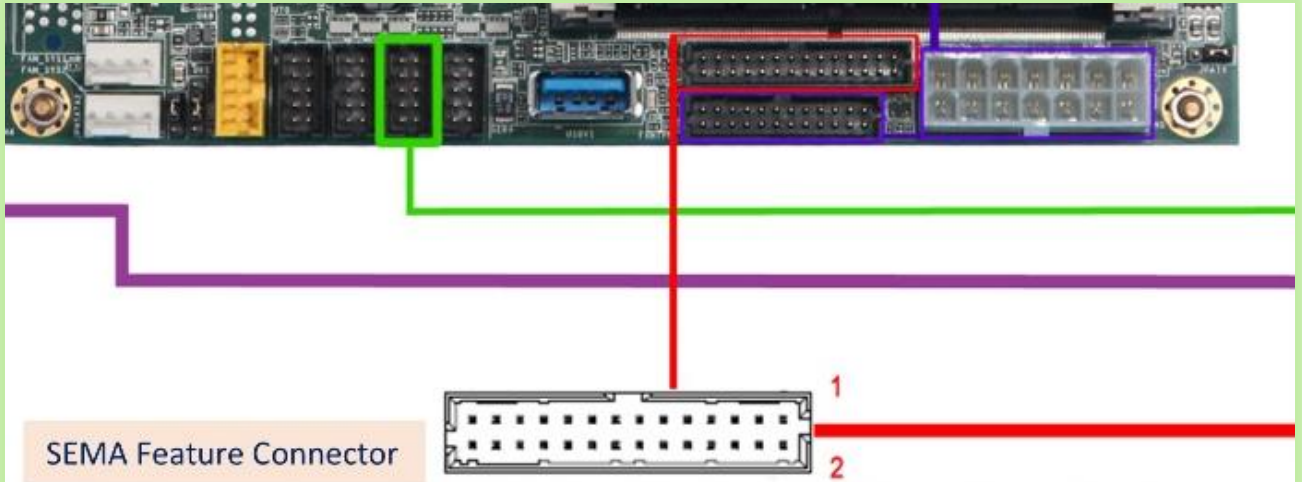
Vehicle hardware



Pin #	Signal	Pull U/D	Ioh/ Iol	Type	Note	Pin #	Signal	Type	Ioh/ Iol	Pull U/D	Note
1	CASE_OPEN#	PU 2M	-	I		2	SMBC	OT	/4mA	PU 10K	1
3	GND	-	-	PWR		4	SMBD	OT	/4mA	PU 10K	1
5	TEMPS	-		I	2	6	I2CC	OT	-	PU 10K	1
7	EXT_BAT	-		PWR		8	I2CD	OT	-	PU 10K	1
9	SB3V3	-	-	PWR		10	SB5V	PWR	-	-	
11	GND	-	-	PWR		12	GND	PWR	-	-	
13	GPIO0	-	PU10K3V3	IOT		14	GPIO1	IOT		PU10K3V3	
15	GPIO2	-	PU10K3V3	IOT		16	GPIO3	IOT		PU10K3V3	
17	GPIO4	-	PU10K3V3	IOT		18	GPIO5	IOT		PU10K3V3	
19	GPIO6	-	PU10K3V3	IOT		20	GPIO7	IOT		PU10K3V3	
21	GPIO8		PU10K3V3	IOT		22	GPIO9	IOT		PU10K3V3	
23	GND	-		PWR		24	SUS_S3#	O	25/25mA	-	
25	12V	-	-	PWR		26	SUS_S4#	O	25/25mA	-	
27	PWR_OK	-	25/25mA	O		28	SUS_S5#	O	25/25mA	-	

Introduction to OmniBot

Vehicle hardware: SEMA



SEMA Feature Connector

Pin #	Signal	Pull U/D	Ioh/ Iol	Type	Note	Pin #	Signal	Type	Ioh/ Iol	Pull U/D	Note
1	CASE_OPEN#	PU 2M	-	I		2	SMBC	OT	/4mA	PU 10K	1
3	GND	-	-	PWR		4	SMBD	OT	/4mA	PU 10K	1
5	TEMPS	-		I	2	6	I2CC	OT	-	PU 10K	1
7	EXT_BAT	-		PWR		8	I2CD	OT	-	PU 10K	1
9	SB3V3	-	-	PWR		10	SB5V	PWR	-	-	
11	GND	-	-	PWR		12	GND	PWR	-	-	
13	GPIO0	-	PU10K3V3	IOT		14	GPIO1	IOT		PU10K3V3	
15	GPIO2	-	PU10K3V3	IOT		16	GPIO3	IOT		PU10K3V3	
17	GPIO4	-	PU10K3V3	IOT		18	GPIO5	IOT		PU10K3V3	
19	GPIO6	-	PU10K3V3	IOT		20	GPIO7	IOT		PU10K3V3	
21	GPIO8		PU10K3V3	IOT		22	GPIO9	IOT		PU10K3V3	
23	GND	-		PWR		24	SUS_S3#	O	25/25mA	-	
25	12V	-	-	PWR		26	SUS_S4#	O	25/25mA	-	
27	PWR_OK	-	25/25mA	O		28	SUS_S5#	O	25/25mA	-	

Introduction to OmniBot

Fundamental package (software) environment

- Fully customized ROS motor driver node that utilize standard ROS convention
- ROS driver node of MPU6050 6-axis IMU
  - with standard ROS state estimator
- Fully integration between SEMA library and ROS2
- Easy to use ROS launch file with maximum customization capabilities
- Finely tuned ROS navigation, guidance, and control algorithms with full list of parameters provided
- Multiple user interfacing methods using the SEMA library

Multi-Machine setup

Multi-Machine setup

ADLINK

http://wiki.ros.org/ROS/Tutorials/MultipleMachines

Multi-Machine setup

Change Host name

- Modify file
  - /etc/hostname
  - /etc/hosts
- Reboot
- Ping your IP from somewhere else
- Ping your hostname.local from somewhere else

```
$sudo gedit /etc/hostname
$sudo gedit /etc/hosts
```

```
$ping ($MASTER_IP)
$ping ($MASTER_NAME).local
```

Multi-Machine setup

Configure ROS MASTER

- Add $ROS_MASTER_URI and $ROS_IP to your master bash environment
- Add ROS_MASTER_URI to slave

```
$ echo "export ROS_MASTER_URI=http://$MASTER_IP:11311"
>> ~/.bashrc
$ echo "export ROS_IP=http://$LOCAL_IP:11311" >>
~/.bashrc
```

Multi-Machine setup

Test ROS master-slave connection

ssh to your master

- With byobu
- start roscore
- Start talker demo

```
$ ssh ($MASTER_NAME).local
$ byobu
$A roscore
$B rosrun roscpp_tutorials
talker
```

On local machine

- Start rviz on the slave machine
- Echo topic on the slave machine

```
$rviz
$rostopic list
$rostopic echo /topic
```

Multi-Machine setup

If you don’t have Ubuntu with ROS set

~~Try TxxxViexxx~~

Multi-Machine setup

If you don’t have Ubuntu with ROS set

On Ubuntu

- Desktop-sharing UI setting
- Allow view and control
- Add password
- Install dconf

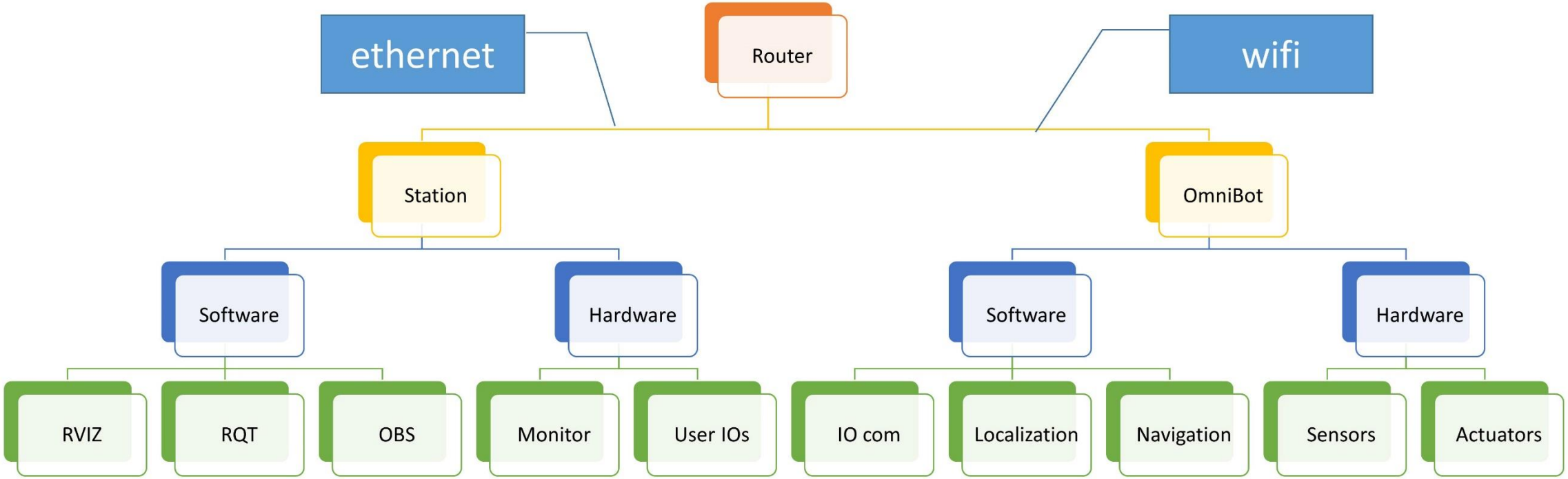
```
$ sudo apt install dconf-editor
```

- Disable “require-encryption“ in ***org -> gnome -> desktop -> remote-access***

On windows

- Install a VNC viewer
- e.g. realvnc
- Login by IP

Multi-Machine setup



ROS Implementation

Outline

- Introduction to OmniBot
  - Vehicle hardware
  - Fundamental package (software) environment
- Multi-Machine setup
- ROS implementation
  - (0) Prerequisite
  - (A) Hardware IO and baseline controller
  - (B) Laser SLAM
  - (C) Vehicle localization against known map
  - (D) Navigation
- Open questions

ROS implementation

(0) Prerequisite: connect the wires

- VDMC manual
  - connect 12V power from Neuron PSU
  - Connect UART port to MAX232 or any equivalent UART-RS232 bridge chip (**NOTE: some manufacture may have their RX/TX label reversed**)
  - connect to Neuron serial port 2
- Laser scanner (ex. YDLidar)
  - USB wire
  - 5v power (from SEMA feature connector SB5V is recommended)
- SEMA peripherals (collision detection, state indication LEDs)
  - LEDs: positive to GPIO, Negative to GND
  - switches: across GPIO and GND
- Other recommendations
  - It is recommended to have your robot's sharp edges wrapped
  - DO NOT obstruct the view of laser scanner
  - Your two wifi antennas should be pointing perpendicular(i.e. 90 degrees) to each other
  - ALWAYS put on your balance lead monitor if your using Li-Po batteries

ROS implementation

(0) Prerequisite: SEMA & environment

- Install ADLINK SEMA Your OmniBot
  - cd ${project_root}/lib
- Source Compiling

```
$cd
catkin_ws/src/neuron_omnib
ot/neuron_demo_gpio/lib
$./setlink.sh
$(chmod +x setlink.sh)
```

```
$cd ~/catkin_ws
$sudo -sE
$catkin_make
$exit
```

ROS implementation

(0) Prerequisite: ROS environment

- Install ROS kinetic and setup workspace
- Install packages (ubuntu software)
  - KATE
    - text editor (very similar to Notepad++)
  - htop
    - a low-cost system monitor
  - GKTerm
    - serial port terminal GUI monitor
  - SSH server
  - OBS

```
$sudo apt-get install kate htop
$sudo apt-get install gtkterm
$sudo apt-get install openssh-server byobu
```

ROS implementation

(0) Prerequisite

- Clone the project

```
$cd catkin_ws/src
$git clone https://github.com/Adlink-ROS/Neuron-
 OmniBot.git
```

- Build the project

```
$cd ..
$catkin_make
```

Outline

- Introduction to OmniBot
  - Vehicle hardware
  - Fundamental package (software) environment
- Multi-Machine setup
- ROS implementation
  - (0) Prerequisite
  - (A) Hardware IO and baseline controller
  - (B) Laser SLAM
  - (C) Vehicle localization against known map
  - (D) Navigation
- Open questions

ROS implementation

(A) Hardware IO

- Start the core
- Launch the launch

```
$roscore
$roslaunch omni_base_driver
omni_base_driver.launch
```

ROS implementation

(A) error?

- Let’s check the message

hint: let’s install YDLidar node

ROS implementation

(A) Hardware IO and keyboard controller

- Start the core
- Launch the launch
- Start the Tele-op keyboard

```
$roscore
$roslaunch omni_base_driver
 omni_base_driver.launch
$rosrun teleop_twist_keyboard
 teleop_twist_keyboard.py
```

ROS implementation

(A) Hardware IO and keyboard controller: error?

- Let’s check the message again

hint: install teleop

ROS implementation

(A) Hardware IO and keyboard controller error?

- Now finally we can drive the OmniBot

```
$roscore
$roslaunch omni_base_driver
 omni_base_driver.launch
$rosrun teleop_twist_keyboard
 teleop_twist_keyboard.py
```

ROS implementation

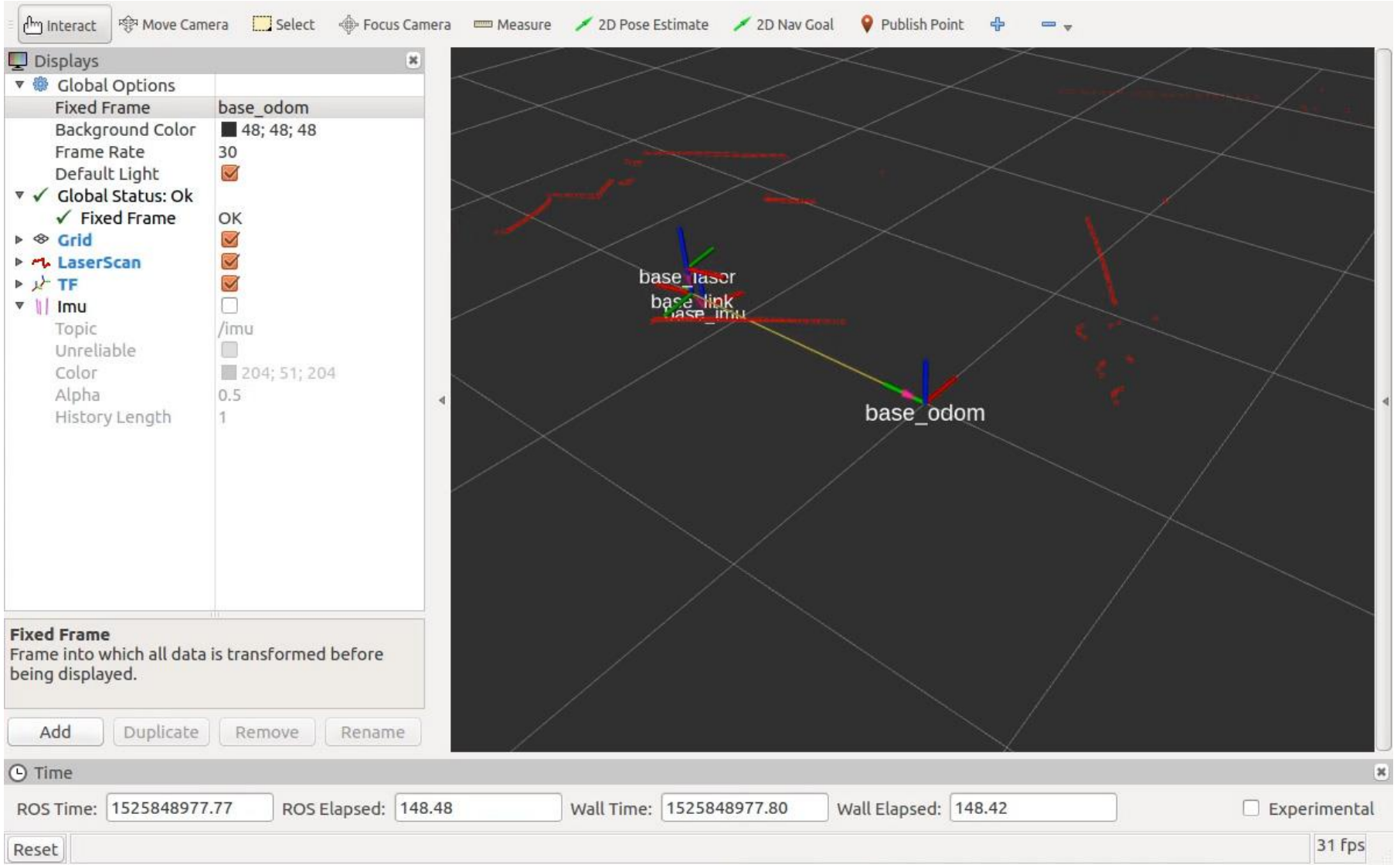
(A) Hardware IO and keyboard controller: visualization

- RVIZ
- RQT

```
$rviz
$rqt
```

ROS implementation

(A) Hardware IO and keyboard controller : visualization



ROS implementation

(A) ADVANCED: Hardware IO with state filter:

- Start the core
- Launch the launch

```
$roscore
$roslaunch omni_base_driver
omni_localization.launch
```

ROS implementation

(A) error again?

- Let’s check the message again

hint: let’s install robot-localization

ROS implementation

(A) ADVANCED: Hardware IO with state filter

- Start the core
- Launch the launch

```
$roscore
$roslaunch omni_base_driver
omni_localization.launch
```

ROS implementation

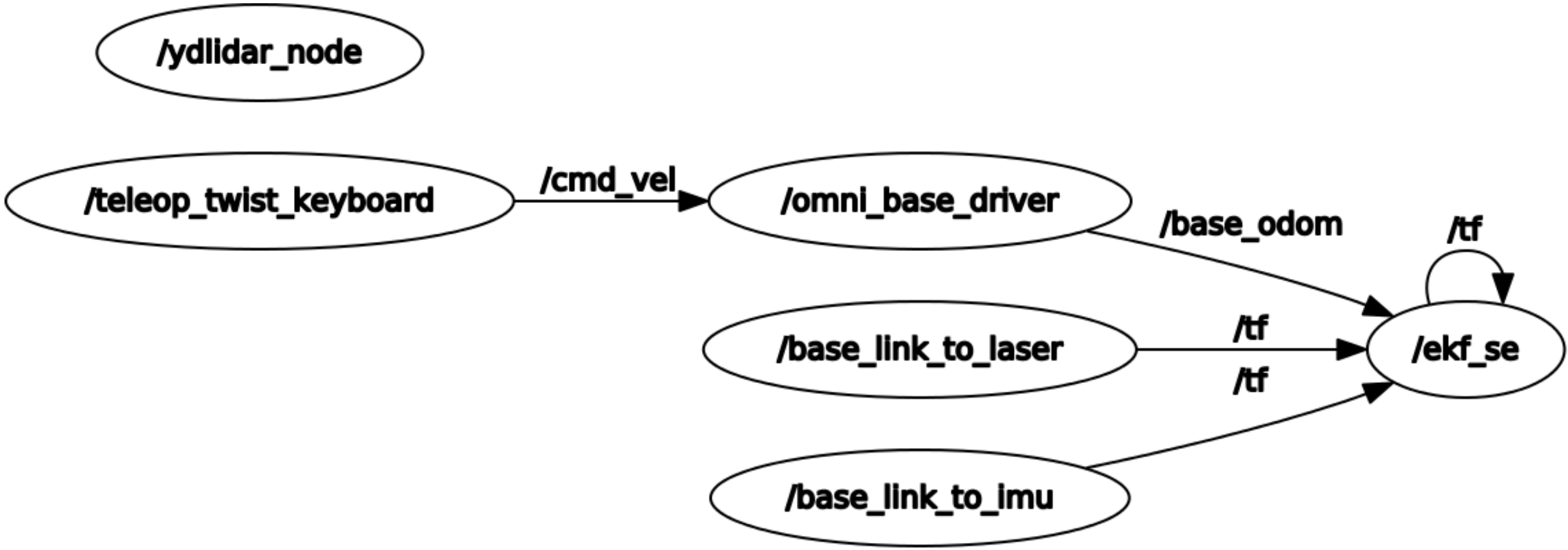
(A) ADVANCED: Hardware IO, state filter, and keyboard controller

- Start the core
- Launch the launch
- Start the Tele-op

```
$roscore
$roslaunch omni_base_driver
 omni_localization.launch
$rosrun teleop_twist_keyboard
 teleop_twist_keyboard.py
```

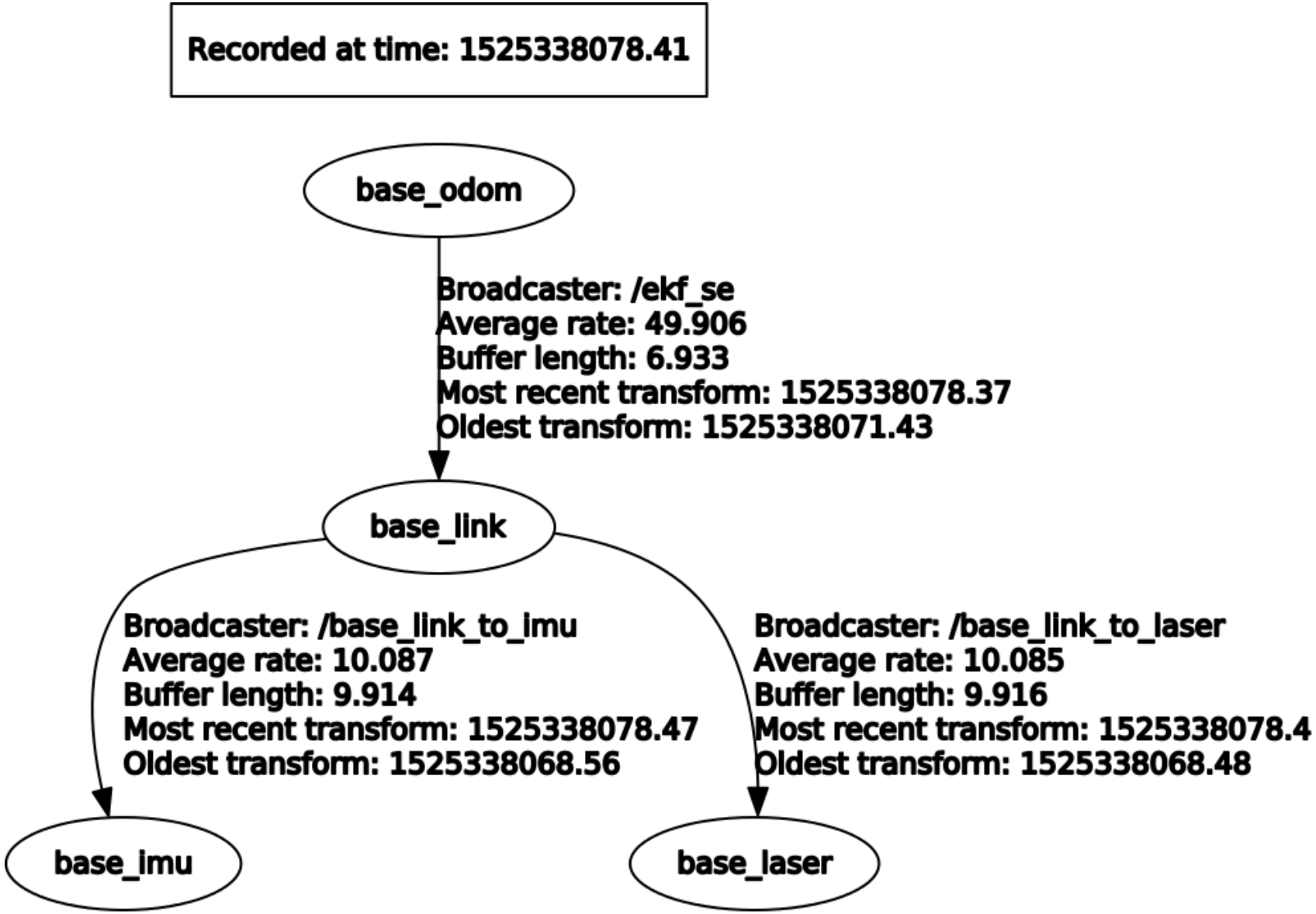
ROS implementation

(A) Hardware IO and baseline controller: visualization



ROS implementation

(A) Hardware IO and baseline controller: visualization



ROS implementation

(A) MORE ADVANCED: Remote controller

- Remote the remote
  - SSH to your destination
  - (recommended)Use byobu
  - Launch omni-driver
- Local
  - Start the Tele-op
  - Rviz, rqt, ….

```
$ssh $YOUR_ROBOT_NAME.local
$byobu
$roscore
$roslaunch omni_base_driver
  omni_localization.launch
```

```
$rosrun teleop_twist_keyboard
teleop_twist_keyboard.py
$rviz
```

Outline

- Introduction to OmniBot
  - Vehicle hardware
  - Fundamental package (software) environment
- Multi-Machine setup
- ROS implementation
  - (0) Prerequisite
  - (A) Hardware IO and baseline controller
  - (B) Laser SLAM
  - (C) Vehicle localization against known map
  - (D) Navigation
- Open questions

ROS implementation

(B) Laser SLAM

- Start the core
- Launch the driver
- Start the Tele-op
- Change RVIZ preset
- Start SLAM-ing

```
$roscore
$roslaunch omni_base_driver
  omni_localization.launch
$rosrun teleop_twist_keyboard
  teleop_twist_keyboard.py
```

```
$roslaunch omni_base_slam
omni_gmapping.launch
```

ROS implementation

(B) Laser SLAM: install packages

- Install stuff

```
$sudo apt-get install ros-
 kinetic-gmapping ros-
 kinetic-scan-tools
```

ROS implementation

(B) Laser SLAM: success!

- Start the core
- Launch the driver
- Start the Tele-op
- Start SLAM-ing

```
$roscore
$roslaunch omni_base_driver
  omni_localization.launch
$rosrun teleop_twist_keyboard
  teleop_twist_keyboard.py
```

```
$roslaunch omni_base_slam
 omni_gmapping.launch
```

ROS implementation

(B) Laser SLAM: SAVE MAP!!

- Start the core
- Launch the driver
- Start the Tele-op
- Start SLAM-ing
- Save the map

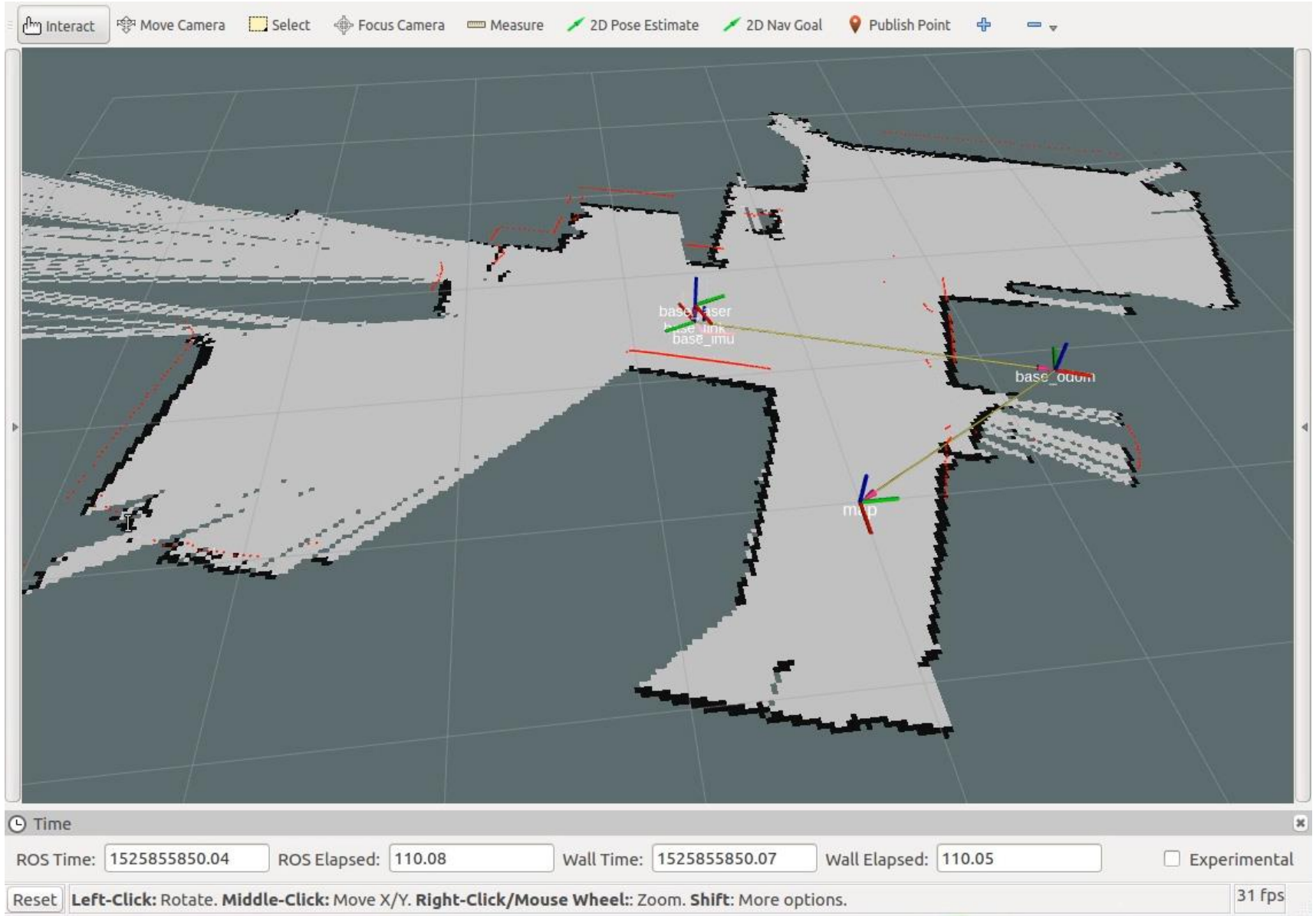
```
$roscore
$roslaunch omni_base_driver
  omni_localization.launch
$rosrun teleop_twist_keyboard
  teleop_twist_keyboard.py
```

```
$roslaunch omni_base_slam
 omni_gmapping.launch
$rosrun map_server map_saver -f
 my_map_file_name
```

```
$sudo apt-get install ros-kinetic-map-server
```

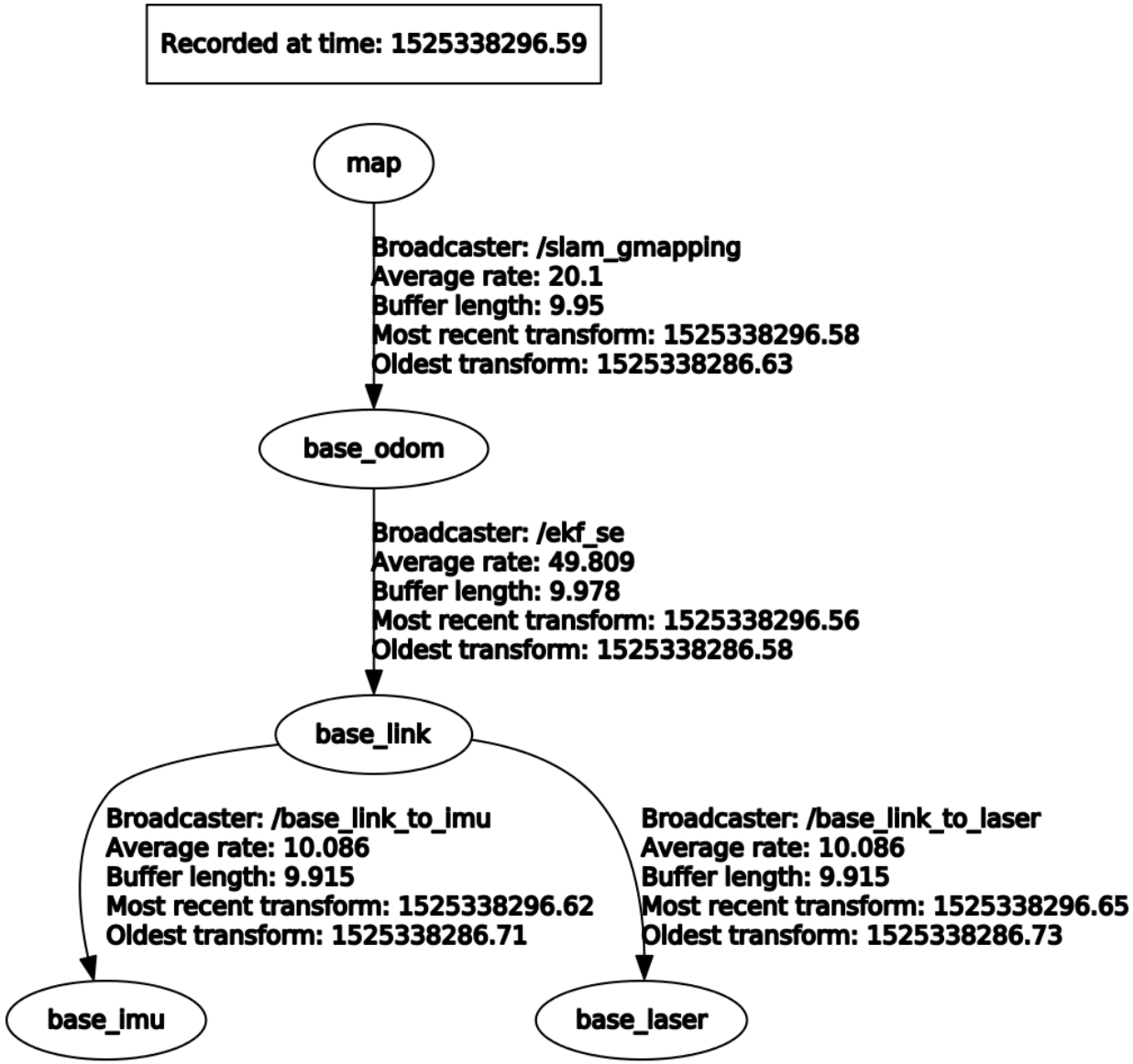
ROS implementation

(B) Laser SLAM: rviz



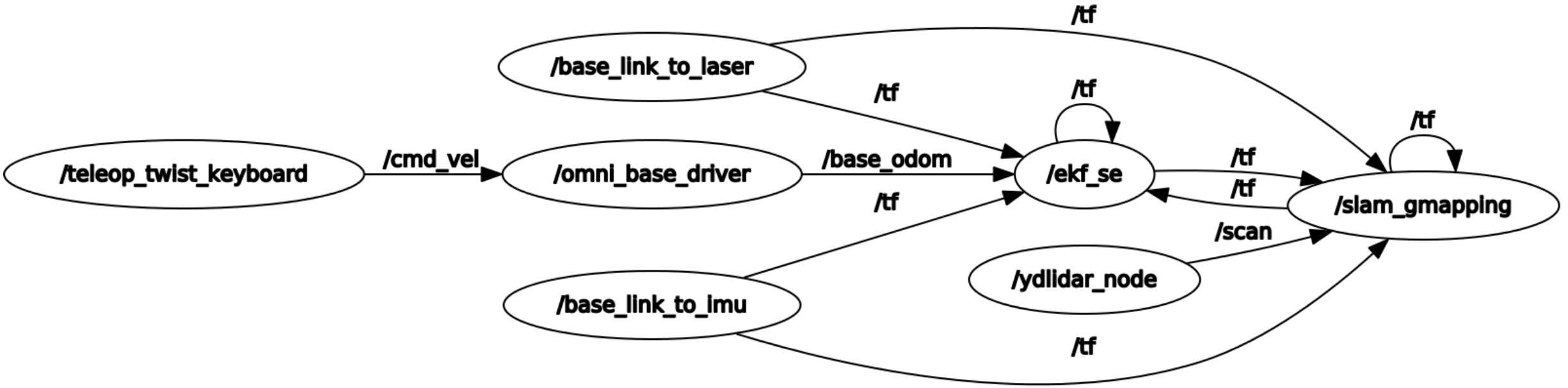
ROS implementation

(B) Laser SLAM: tf tree



ROS implementation

(B) Laser SLAM: nodes



Outline

- Introduction to OmniBot
  - Vehicle hardware
  - Fundamental package (software) environment
- Multi-Machine setup
- ROS implementation
  - (0) Prerequisite
  - (A) Hardware IO and baseline controller
  - (B) Laser SLAM
  - (C) Vehicle localization against known map
  - (D) Navigation
- Open questions

ROS implementation

(C) Vehicle localization: install AMCL

- Install package robot-localization
- AMCL: Adaptive Monte Carlo Localization

```
$sudo apt-get install ros-kinetic-robot-localization
```

ROS implementation

(C) Vehicle localization: localization

- Start the core
- Launch the driver
- Start the Tele-op
- Start robot localization

```
$roscore
$roslaunch omni_base_driver
  omni_localization.launch
$rosrun teleop_twist_keyboard
  teleop_twist_keyboard.py
```

```
$roslaunch omni_base_nav
 omni_localize.launch
```

ROS implementation

(C) Vehicle localization: initialization and update

- Start the core
- Launch the driver
- Start the Tele-op
- Start robot localization
- Location initialization and update

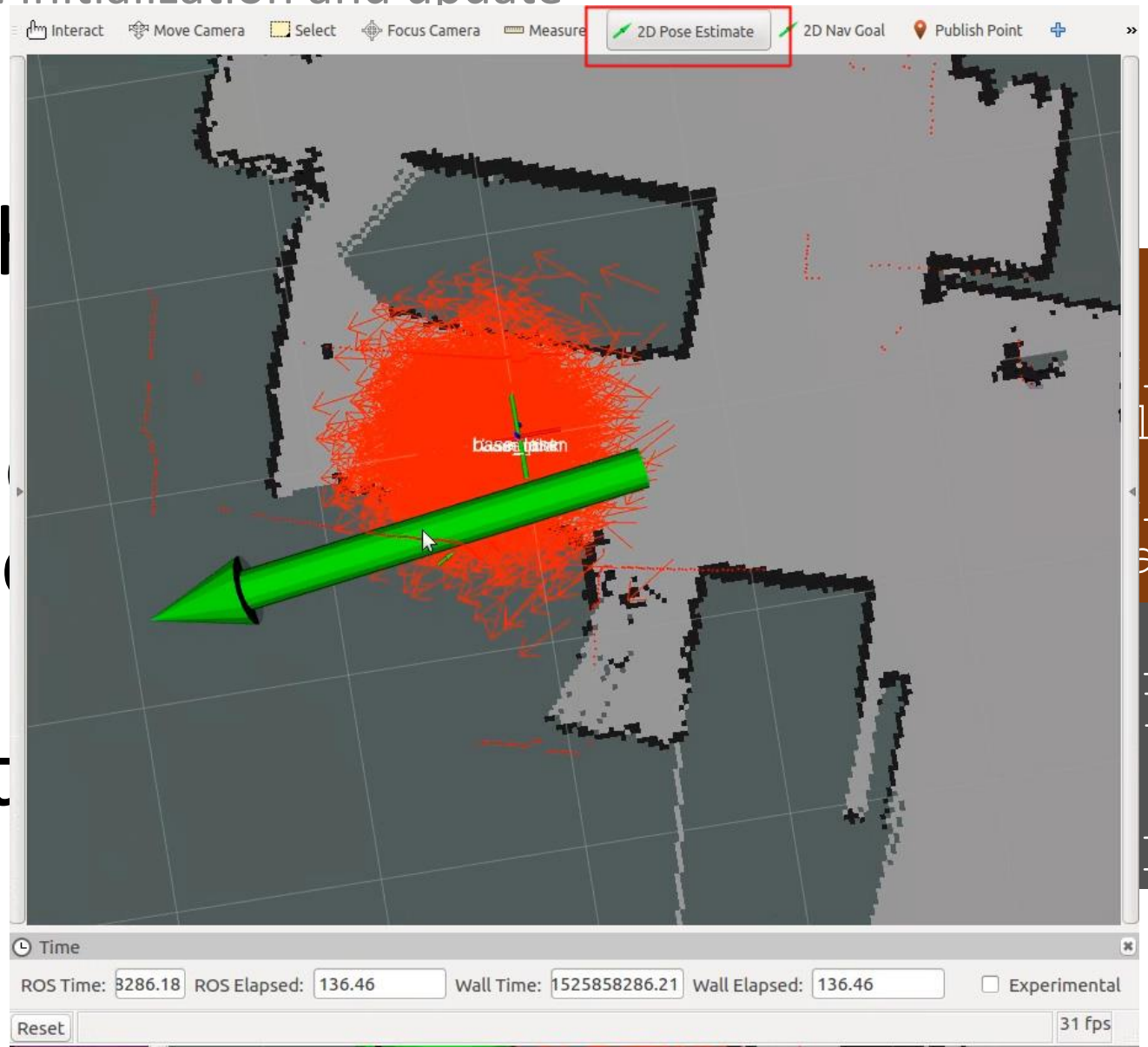
```
$roscore
$roslaunch omni_base_driver
  omni_localization.launch
$rosrun teleop_twist_keyboard
  teleop_twist_keyboard.py
```

```
$roslaunch omni_base_nav
 omni_localize.launch
$rosservice call
 /request_nomotion_update
$rosservice call
 /global_localization
```

ROS implementation

(C) Vehicle localization: initialization and update

- Start the
- Launch tl
- Start the
- Start rob
  localizati
- Location
  initializat
  update



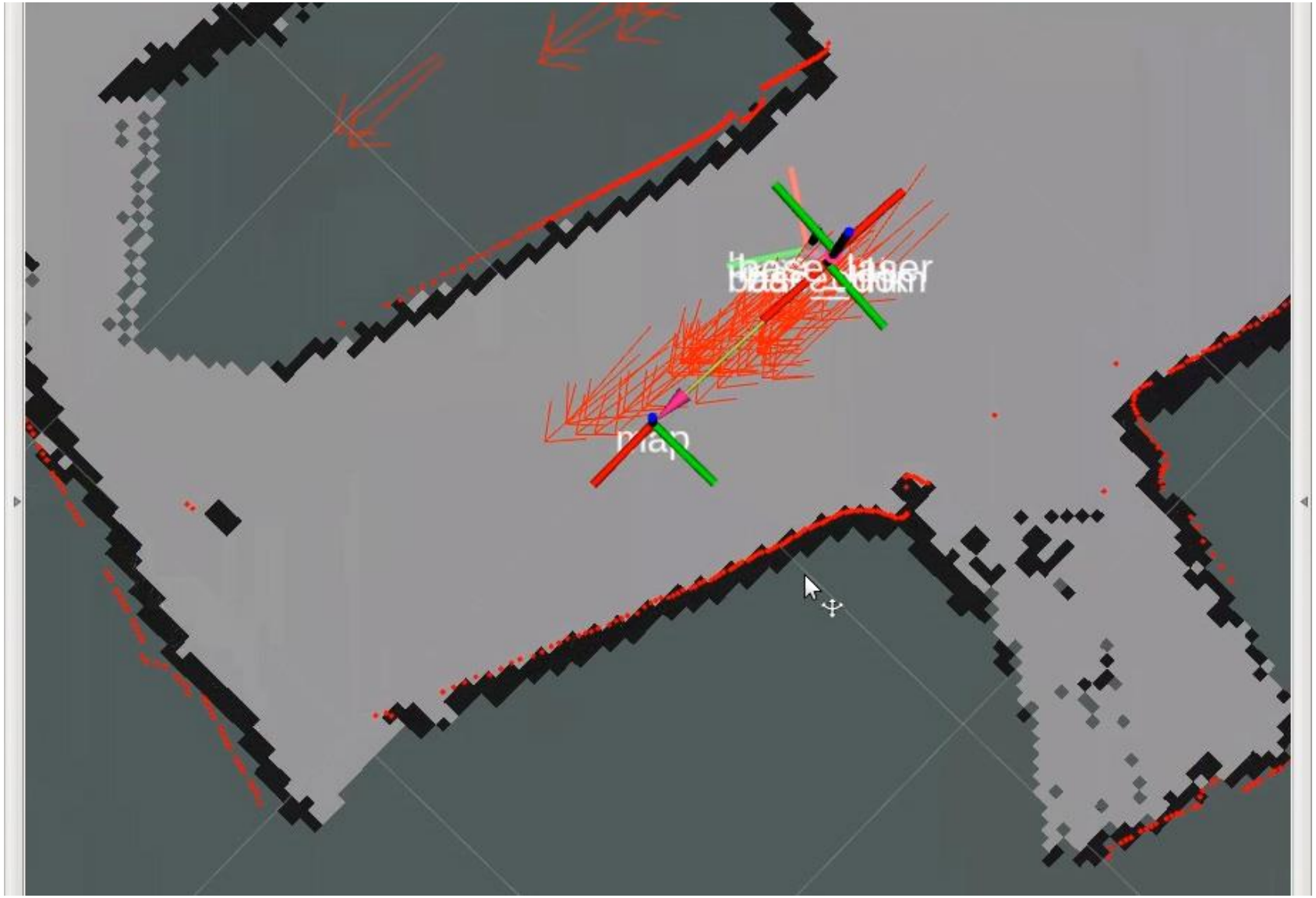

```
_driver
launch
_keyboard
ard.py

_nav
h


pdate
```

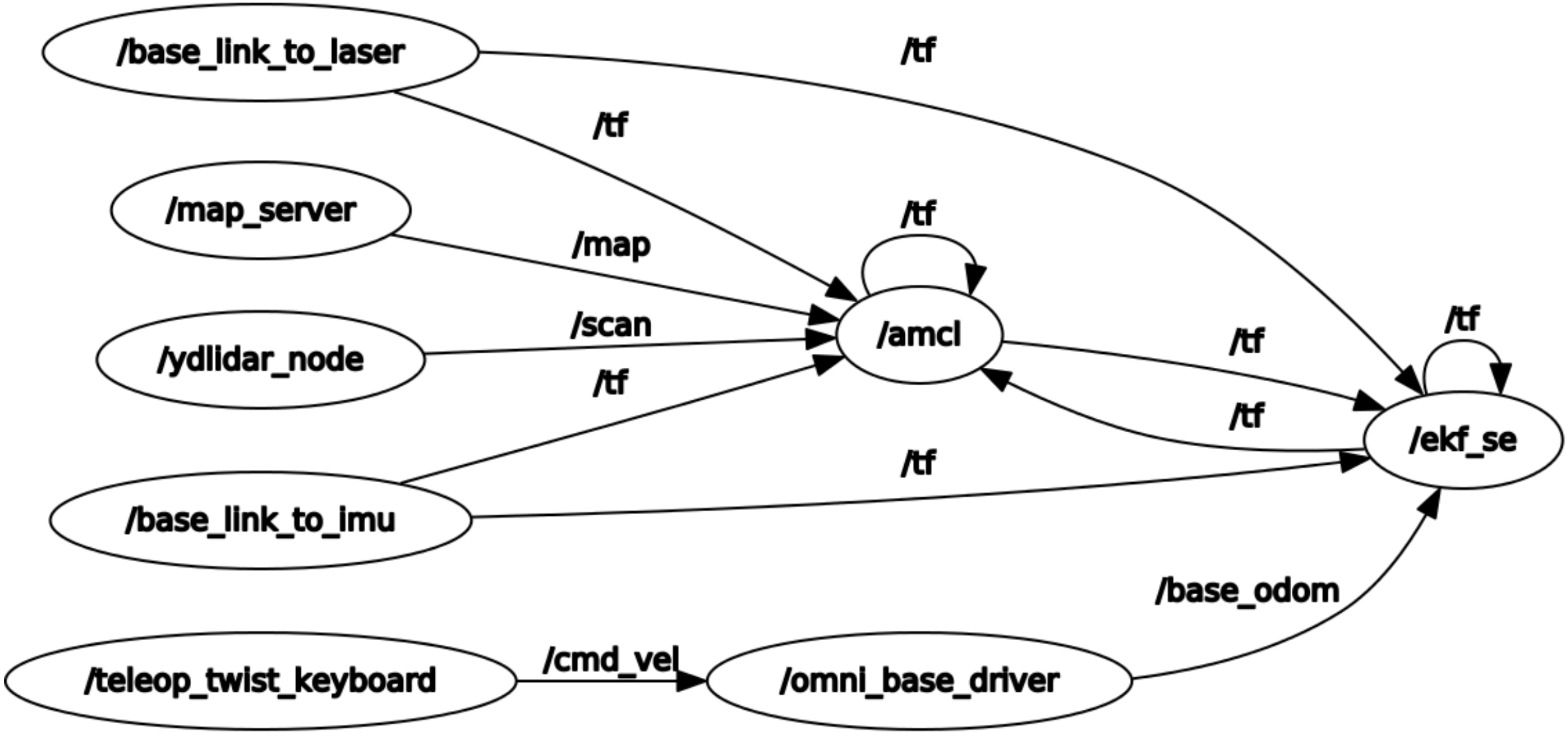
ROS implementation

(C) Vehicle localization: rviz



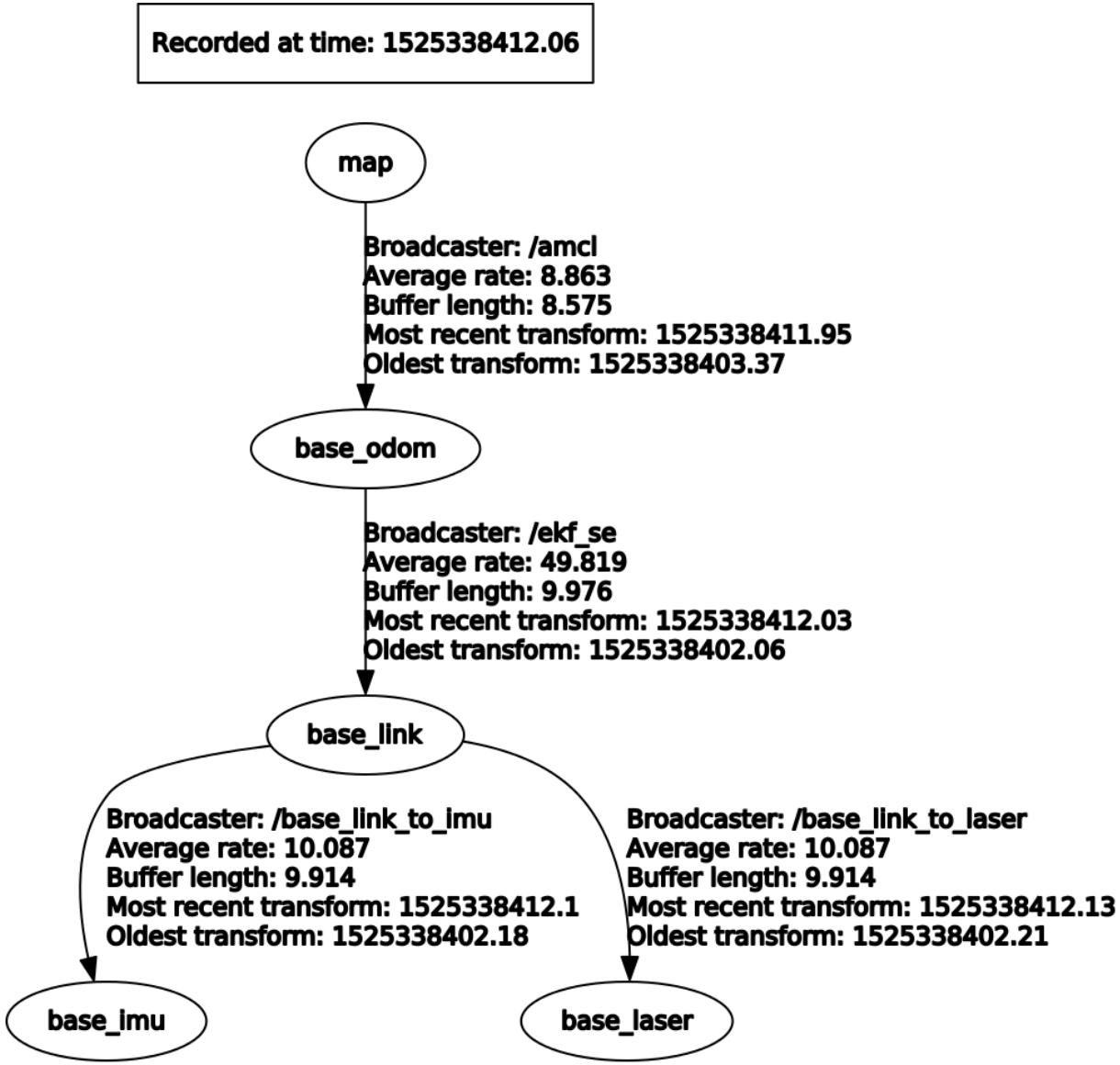
ROS implementation

(C) Vehicle localization: node graph



ROS implementation

(C) Vehicle localization: tf tree



Outline

- Introduction to OmniBot
  - Vehicle hardware
  - Fundamental package (software) environment
- Multi-Machine setup
- ROS implementation
  - (0) Prerequisite
  - (A) Hardware IO and baseline controller
  - (B) Laser SLAM
  - (C) Vehicle localization against known map
  - (D) Navigation
- Open questions

ROS implementation

(D) Navigation: package

- Global planner
- Local planner
  - dwa
  - teb
  - eband

```
$sudo apt-get install ros-kinetic-global-planner \
      ros-kinetic-dwa-local-planner \
      ros-kinetic-teb-local-planner \
      ros-kinetic-teb-local-planner-tutorials \
      ros-kinetic-eband-local-planner
```

ROS implementation

(D) Navigation

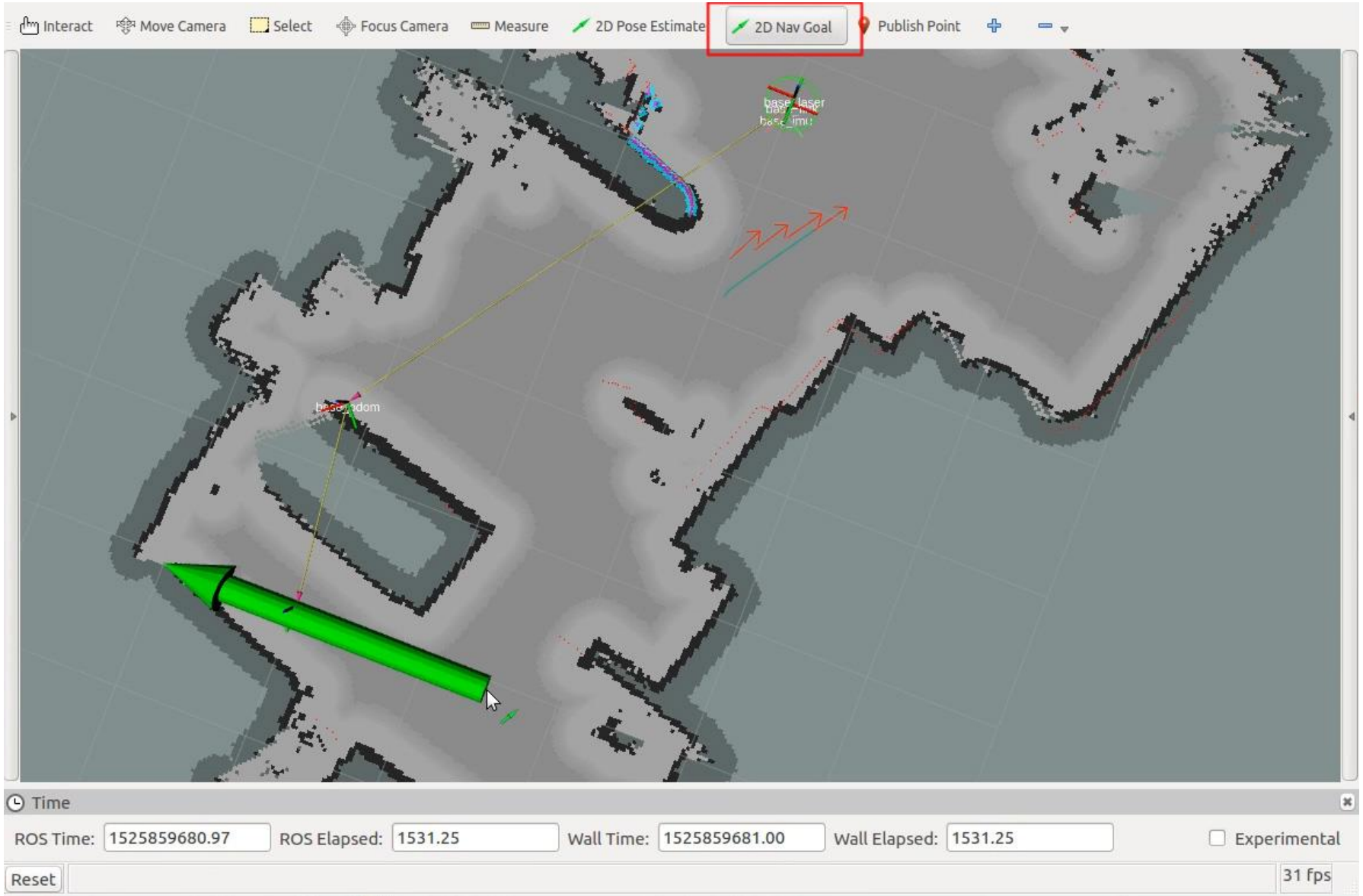
- Start the core
- Launch the driver
- Start the Tele-op
- Robot localized
- Start moving!

```
$roscore
$roslaunch omni_base_driver
  omni_localization.launch
$rosrun teleop_twist_keyboard
  teleop_twist_keyboard.py
```

```
$roslaunch omni_base_nav omni_localize.launch
$roslaunch omni_base_nav omni_nav_dwa.launch
$roslaunch omni_base_nav omni_nav_teb.launch
$roslaunch omni_base_nav omni_nav_eband.launch
```

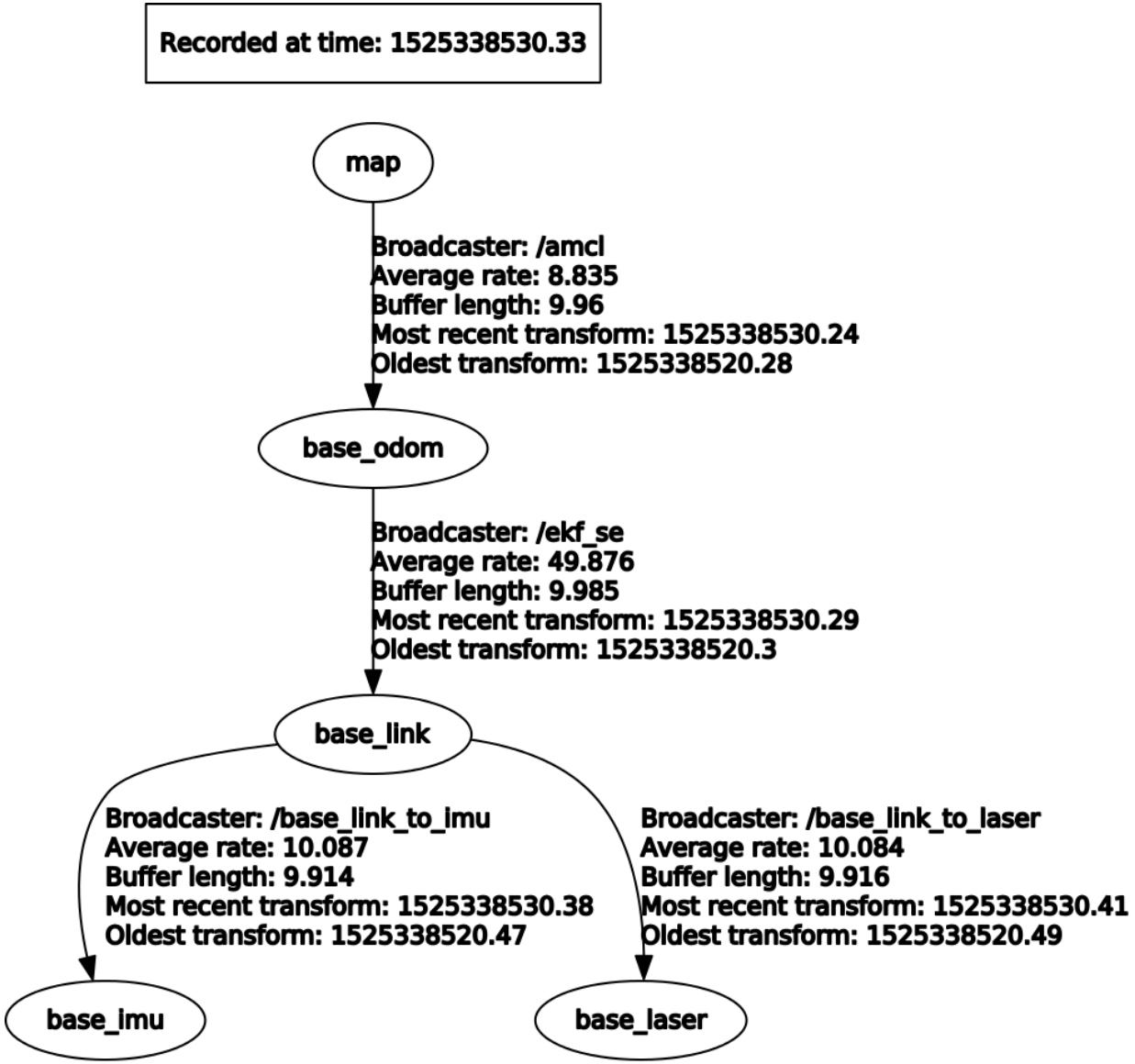
ROS implementation

(D) Navigation: Goal



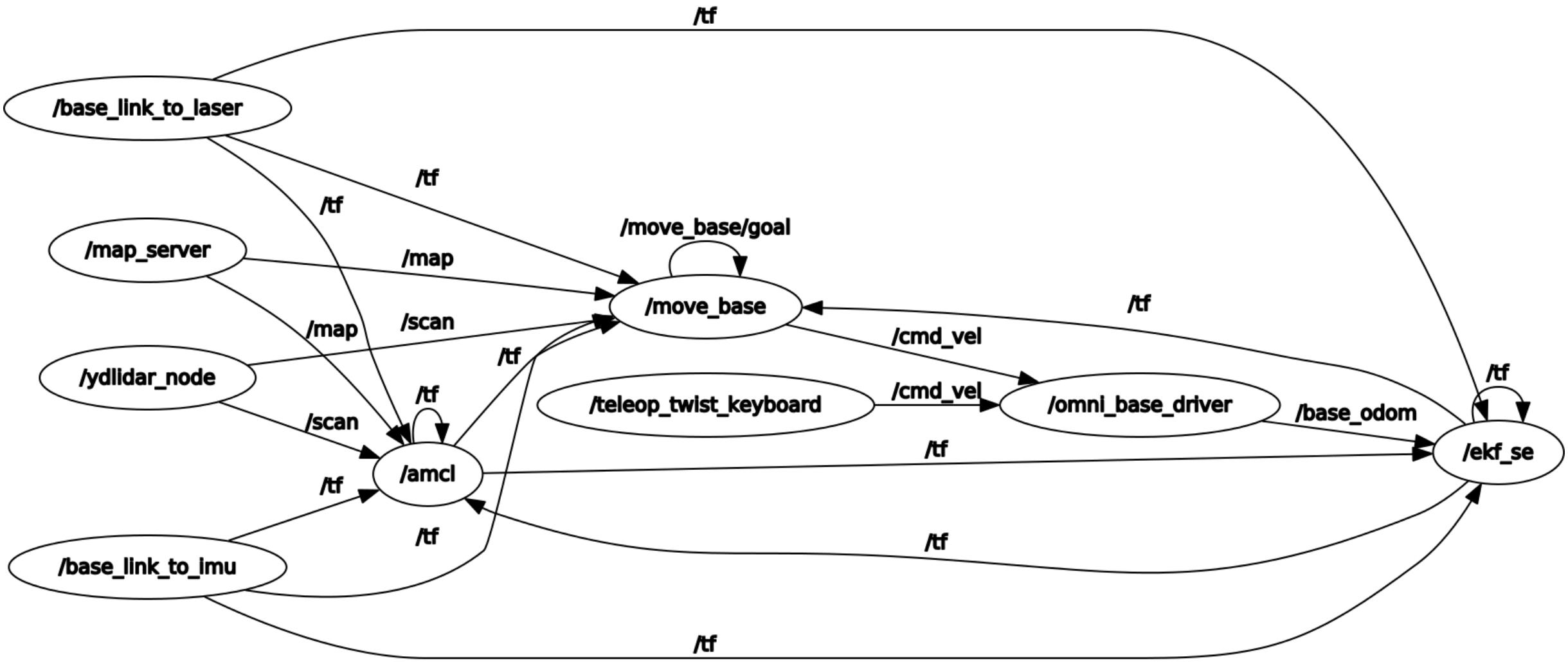
ROS implementation

(D) Navigation: TF tree



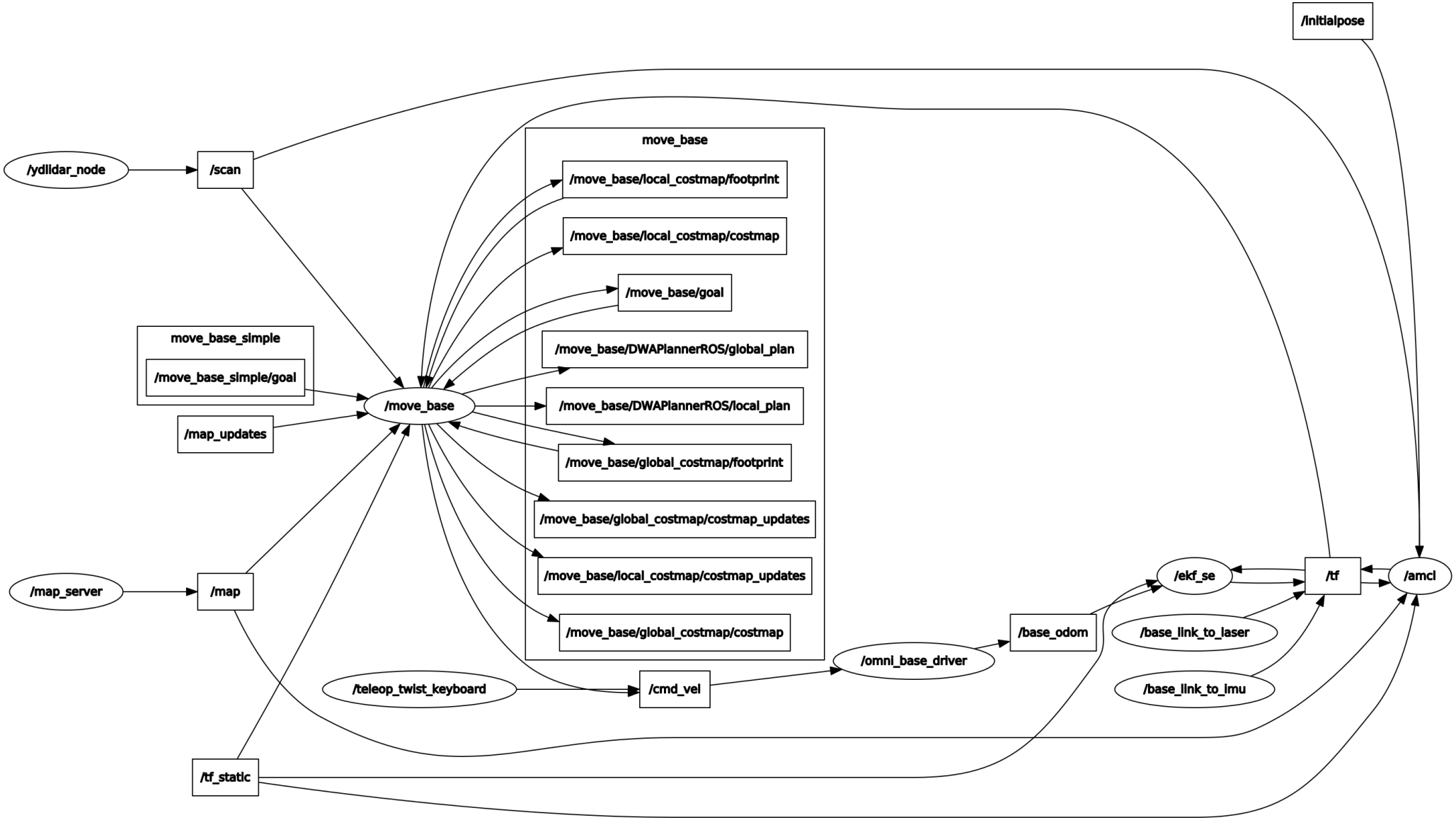
ROS implementation

(D) Navigation



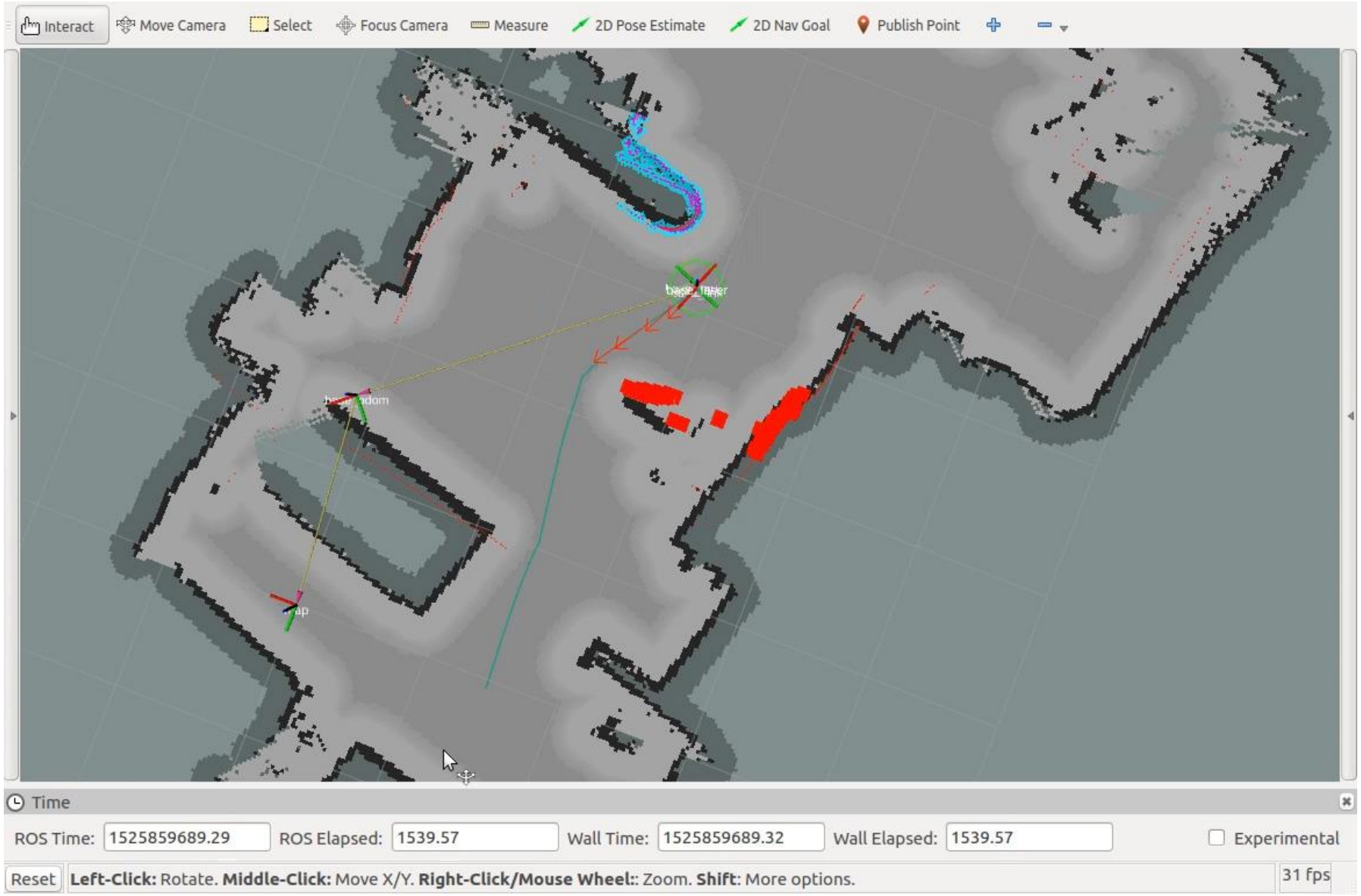
ROS implementation

(D) Navigation



ROS implementation

(D) Navigation



Open Problem

Open Questions

And future works

- Peripherals
  - Contact sensors
  - Indicators
- Algorithm
  - VDMC KF
  - NDI control
- State management
- HM interaction