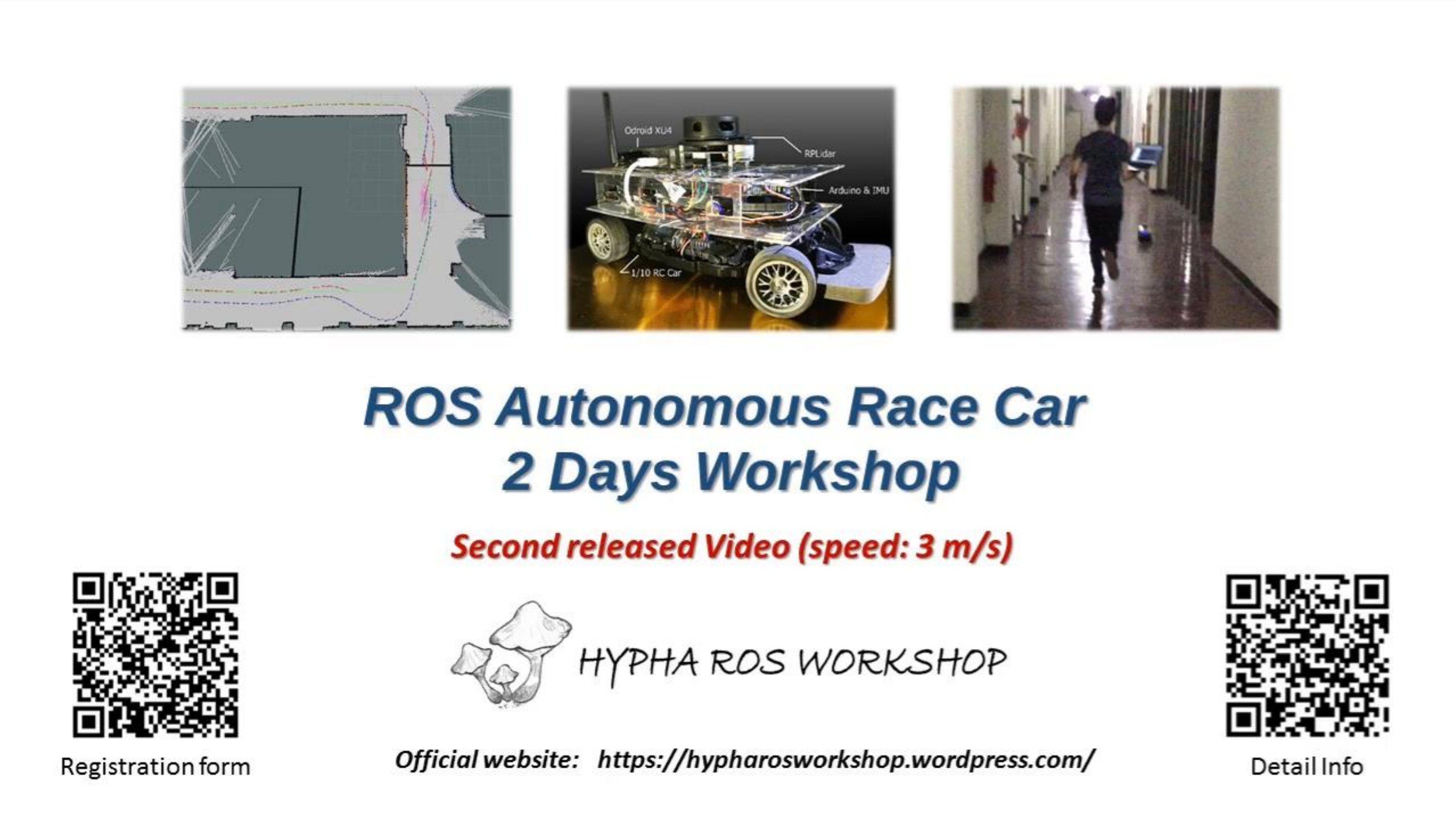
HyphaROS RaceCar Project

Presenter : HaoChih, LIN

林浩鋕

Website: https://hypharosworkshop.wordpress.com/contact/
FB Page: https://www.facebook.com/HyphaROS/
Youku: http://i.youku.com/hypha
Email: hypha.ros@gmail.com
WeChat (ID): HyphaROS


Odroid XU4
RPLidar
Arduino & IMU
1/10 RC Car

ROS Autonomous Race Car
2 Days Workshop

Second released Video (speed: 3 m/s)

HYPHA ROS WORKSHOP

Registration form

Official website: https://hypharosworkshop.wordpress.com/

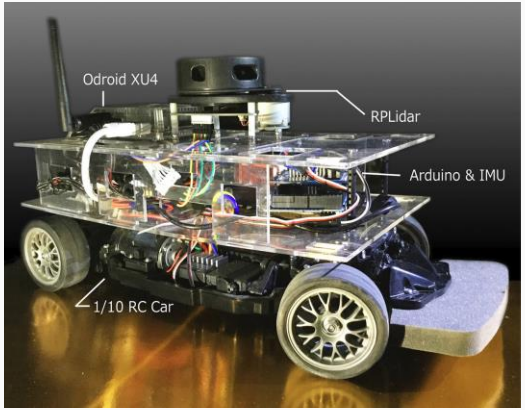
Detail Info

http://i.youku.com/hypharos

- **About us**
- **Why RaceCar**
- **Hardware Configuration**
  (odroid xu4, RPLidar, arduino, gy85, etc)
- **Software Implementation (ROS)**
  (laser based odom, IMU, EKF, L1 controller)
- **How to build the track**
- **Roadmap**
- **Q & A**
- **Basic Operation [Appendix]**

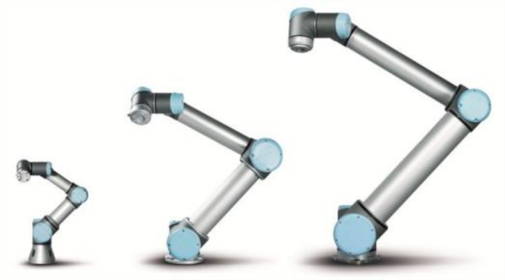
About us

- **Introduction**
  - Self-organized Workshop: 2
  - Speech & Training: 5
  - Technical supports: > 10 universities/labs & > 5 companies
  - ROS Summer School in China 2017
  - Product & Service [ROS]: Workshop, Technical Consultant, AGVs, Home robots, etc.



Why RaceCar

- **The requirements of the platform**
  - ROS fully support
  - Able to implement 2D Navigation stack (laser based)
  - **Low cost (ARM SBC, low-cost lidar, mems imu, etc)**
  - **High speed/performance**
  - Robust & Safety
  - Algorithms evaluation/comparison
  - Modularization & Extensibility



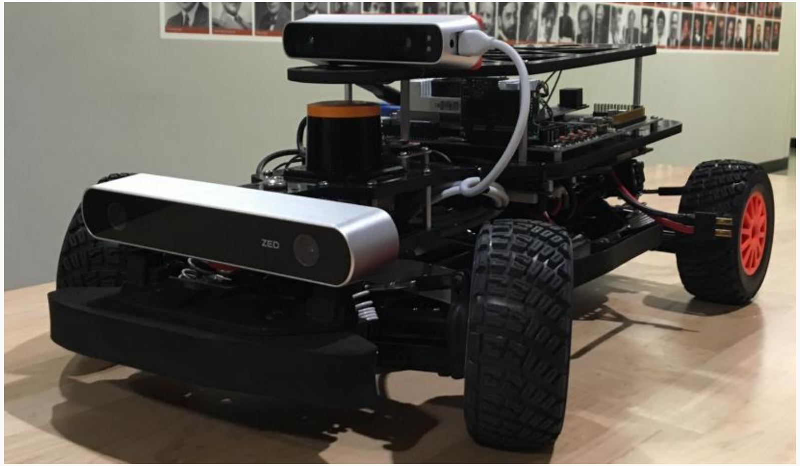
Why RaceCar

- **The origin of RACECAR**
  MIT: https://mit-racecar.github.io/
  Youtube: https://www.youtube.com/watch?v=9fzzp6oxid4

1/10-scale Traxxis Rally Car
Nvidia Jetson TX 1
Hokuyo UST-10LX laser range finder
Stereolabs ZED stereo camera
Structure.io depth camera
Sparkfun IMU
Encoder + optical flow

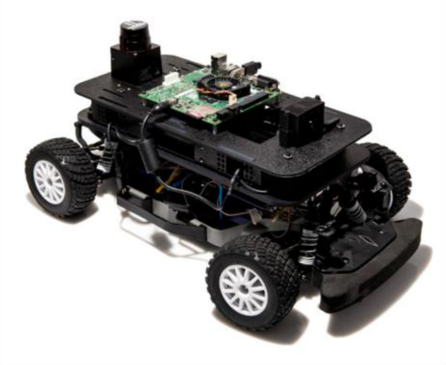
realtime onboard 4 m/s

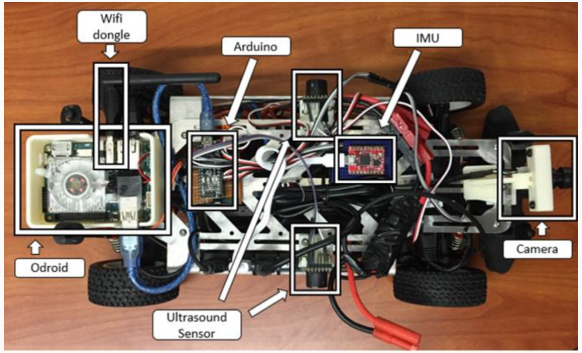
Total cost: around 3,600 USD



Why RaceCar

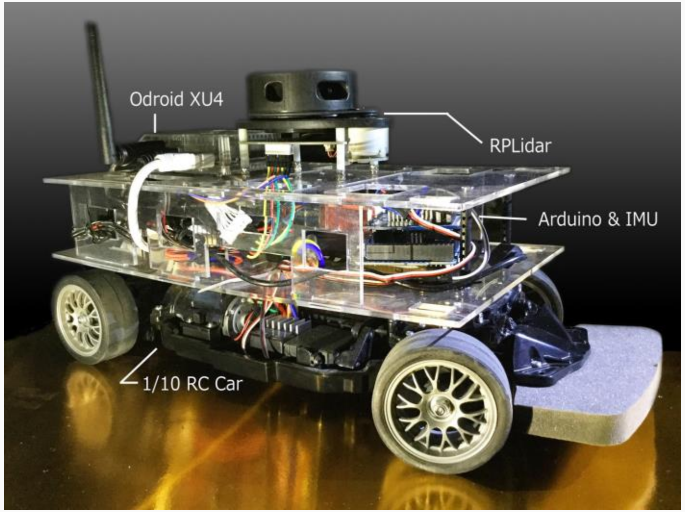
- **Similar projects**
  Penn: http://f1tenth.org/ [without slam, NAV]
  UCB: http://www.barc-project.com/projects/ [without laser]
  Georgia Tech: https://github.com/AutoRally [for outdoor]



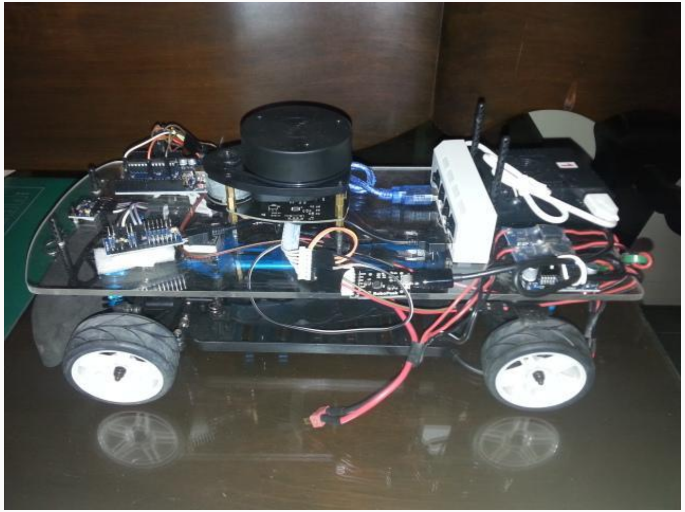


Why RaceCar

- **Our Solution**
  - Open-Source, Open-Hardware
  - **Low-cost (~ 600 USD), High-speed (~ 3 m/s)**
  - Full tutorial (ongoing)

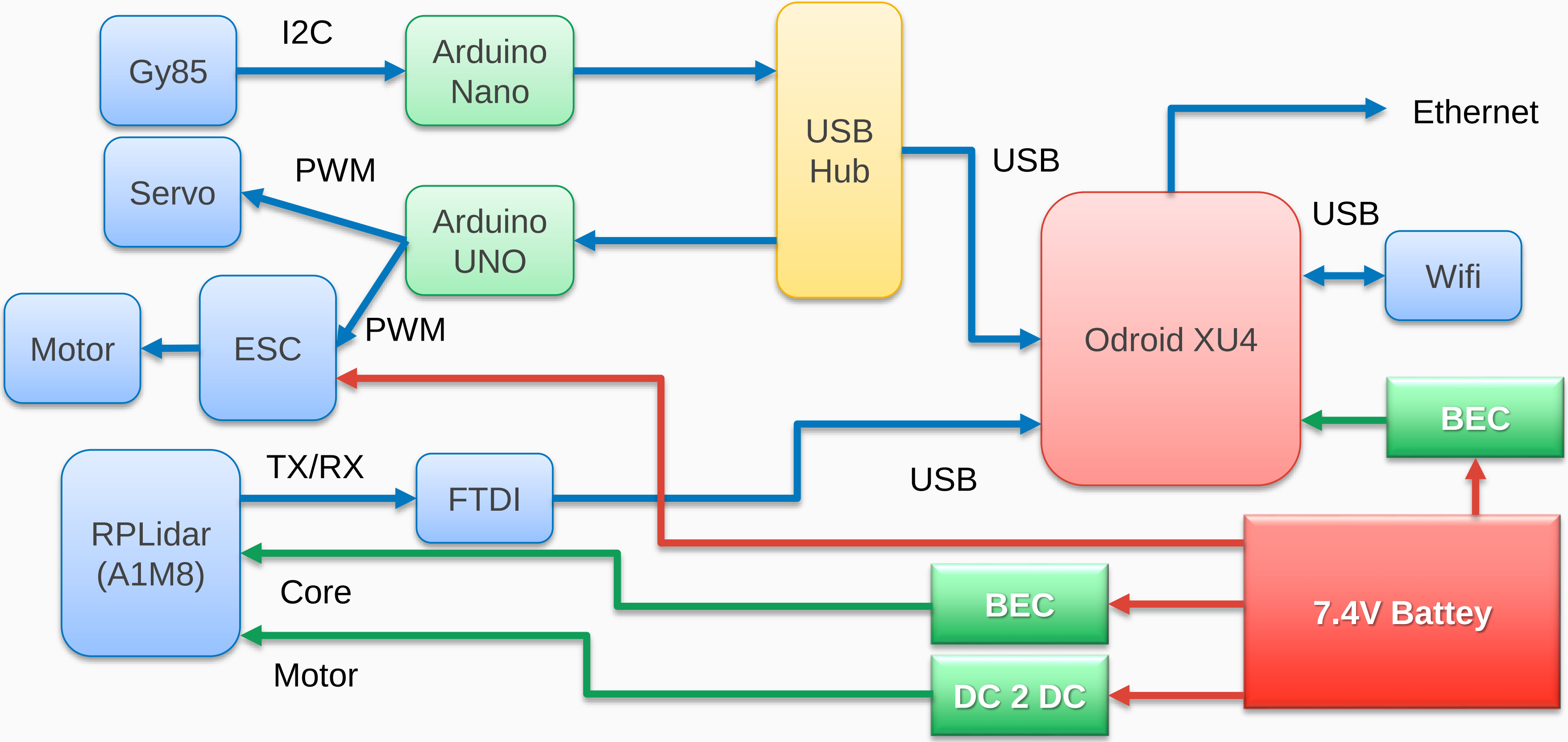


Version 1

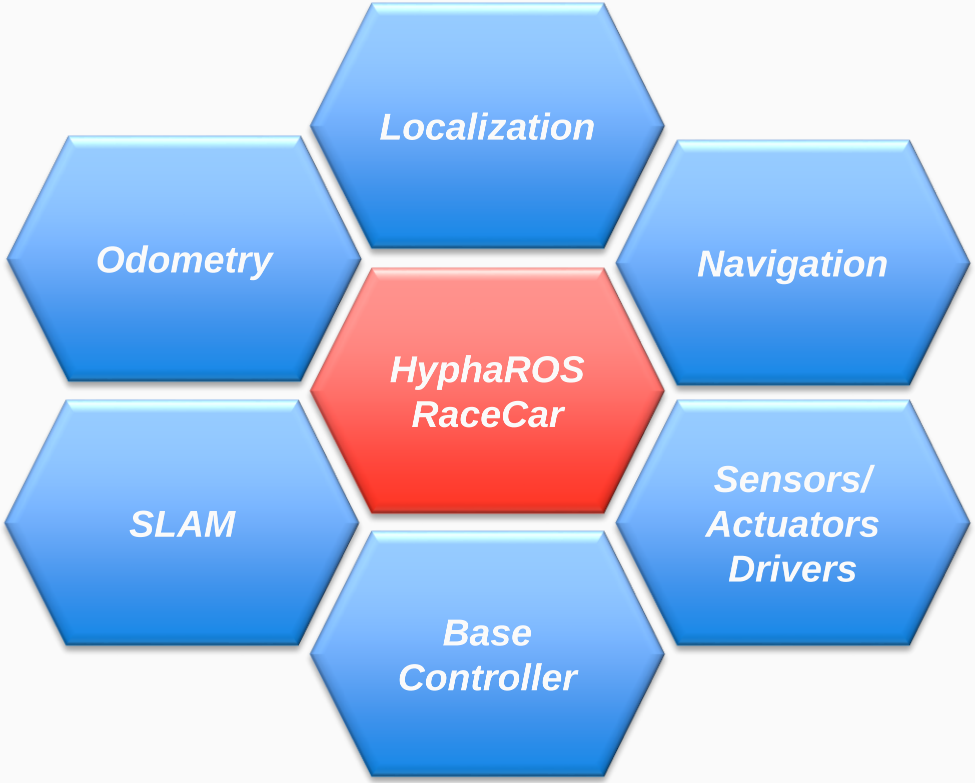


Version 2

Hardware Architecture



Software Overview

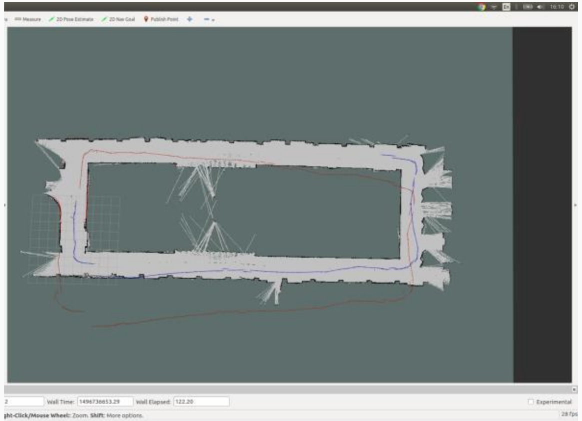


2D Laser SLAM

- **Gmapping**
  - needs external odom (in our case: laser odom fused with imu0)
  - Rao-Blackwellized Particle Filter (RBPF)
  - Loop closure
- **Hector SLAM**
  - Gauss-Newton approach
  - External odom is not necessary
  - No loop closure
  - Requires high quality sensor (e.g. UTM-30LX)
  - Supports EKF fusion (imu)
- **MRPT-ICP SLAM**
  - ICP algorithm
  - For small to mid-sized maps
  - External odom is not necessary

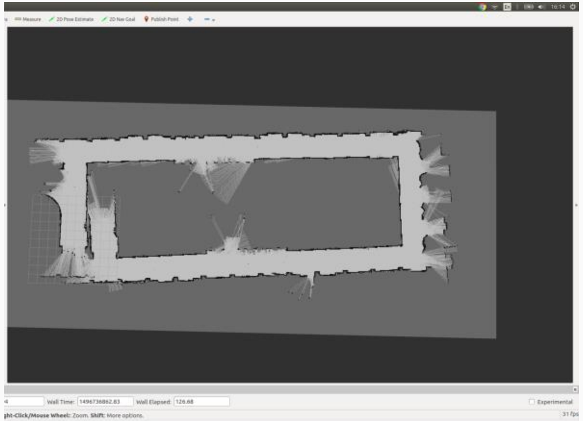
2D Laser SLAM

- **Gmapping**
  - `$ roslaunch hypha_racecar desktop_gmapping.launch`
- **MRPT-ICP SLAM**
  - `$ roslaunch hypha_racecar desktop_icp_mapping.launch`
- **Hector SLAM**
  - `$ roslaunch hypha_racecar Test_hector_rplidar.launch`



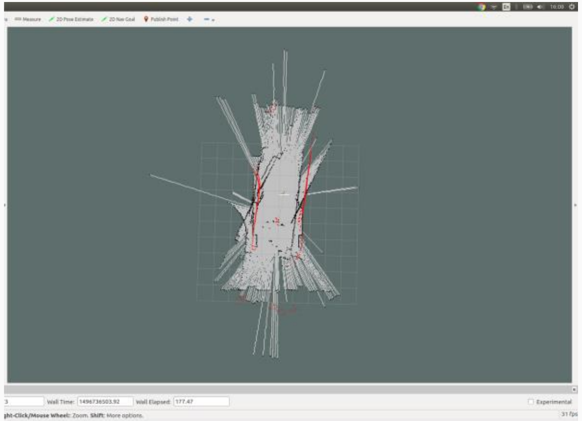
2 Wall Time: 1496736653.29 Wall Elapsed: 122.20 □ Experimental

ght-Click/Mouse Wheel: Zoom. Shift: More options. 28 fps



4 Wall Time: 1496736862.83 Wall Elapsed: 126.68 □ Experimental

ght-Click/Mouse Wheel: Zoom. Shift: More options. 31 fps



3 Wall Time: 1496736503.92 Wall Elapsed: 177.47 □ Experimental

ght-Click/Mouse Wheel: Zoom. Shift: More options. 31 fps

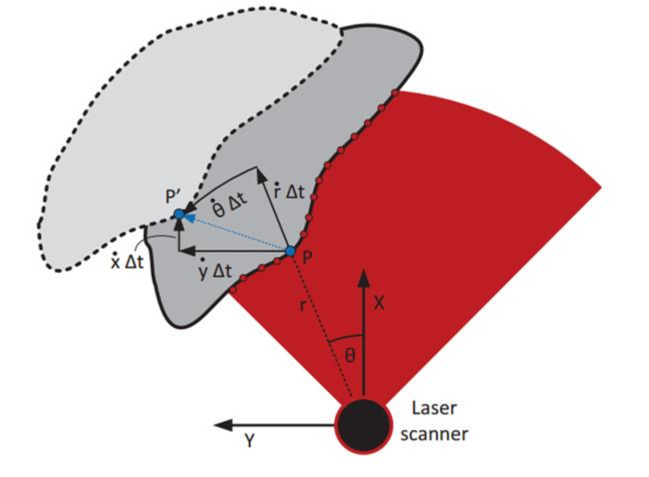
Laser Odom (EKF IMU)

Odometry resource selection

- **Wheel encoder**
  - Most reliable (in low speed)
  - Slip/Drifting Issue
  - Custom codes
- **Laser odometry**
  - Performance depends on laser resolution/environment features
  - Good estimates in translation, awful in rotation
  - Better system portability
- **Visual Odometry (or Visual Inertial Odometry)**
  - Low robust
  - High computational loading
  - Case by case tuning
  - Highly depends on sensor type/quality

Laser Odom (EKF IMU)

- **rf2o laser odom package**
  - http://mapir.isa.uma.es/mapirwebsite/index.php/mapir-downloads/papers/217
  - range flow constraint equation $\dot{r} \simeq R_t + R_\alpha \dot{\alpha} = R_t + R_\alpha k_\alpha \, \dot{\theta}$
  - Iteratively Reweighted Least Squares (IRLS)



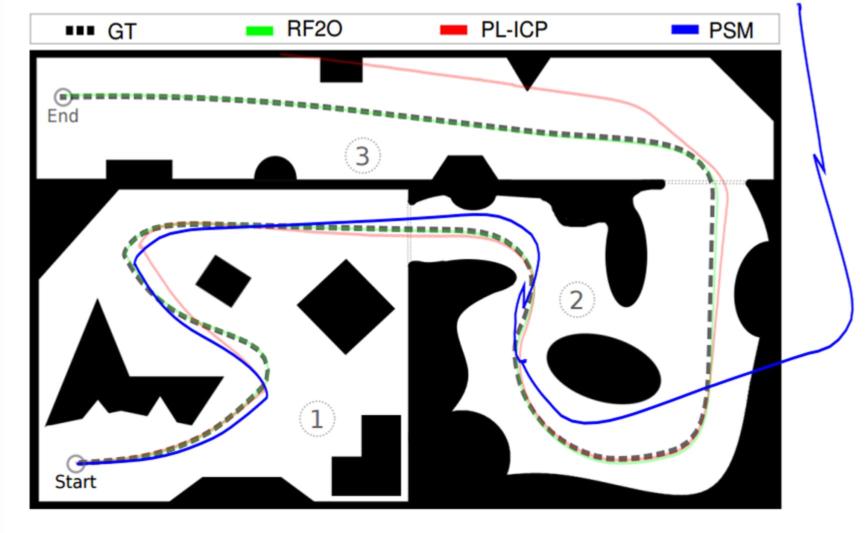


figure source:
Planar Odometry from a Radial Laser Scanner. A Range Flow-basedApproach

Laser Odom (EKF IMU)

- **Problem of rf2o**
  - Performance of rf2o is bad in rotation behavior
  - Needs external measurement to estimate yaw motion (gy85)
- **ROS EKF Package**
  - robot_pose_ekf
    - move_base built-in pkg
    - for 2D, low flexibility
  - robot_localization
    - all params are adjustable, well documented
    - 3D model based
    - supports multi-sensors (including GPS, imu, vo, etc)
  - ethzasl_msf
    - ESKF architecture (imu derived model)
    - good for VO measurement
    - best choice for drone

Laser Odom (EKF IMU)

- **robot_localization pkg**
  - http://wiki.ros.org/robot_localization
  - https://github.com/cra-ros-pkg/robot_localization
  - [document] http://docs.ros.org/indigo/api/robot_localization/html/index.html
  - [video] https://vimeo.com/142624091
  - All state estimation nodes track the 15-dimensional state of the vehicle:

$$(X, Y, Z, roll, pitch, yaw, \dot{X}, \dot{Y}, \dot{Z}, r\dot{o}ll, pi\dot{t}ch, y\dot{a}w, \ddot{X}, \ddot{Y}, \ddot{Z})$$

  - Implementation (See example file: "hypha_ekf_params.yaml")
    Vx, Vy from rf2o, yaw from imu, custom covariance settings

Laser Odom (EKF IMU)

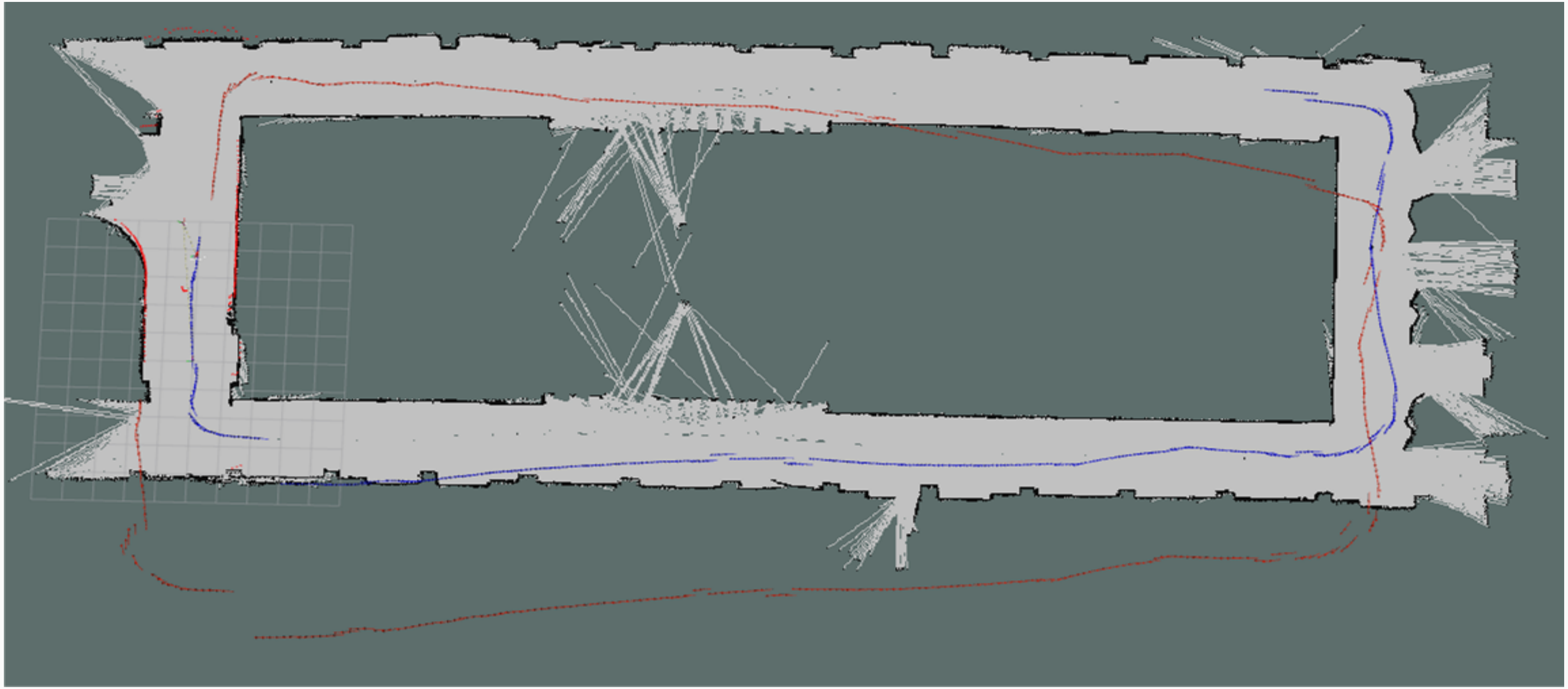
- **IMU yaw reading**
  - [arduino] http://wiki.ros.org/razor_imu_9dof (file: hypha-racecar/document/arduino)
  - [odroid] gy85 with modified pySerial code
  - Using gyro_z raw data for integration directly instead of output data from ekf
  - See example (file: imu_auto.py)

- **ROS bag Testing**
  - Try to modify ekf param file (desktop_gmapping.launch)

- **Realtime onboard testing**
  - Test_laser_odom.launch
  - Test_gmapping.launch

- **rf2o bug**
  - replace file: CLaserOdometry2D.cpp (hypha-racecar/document/replace_files/rf2o/)

Laser Odom (EKF IMU)



Communication & Control

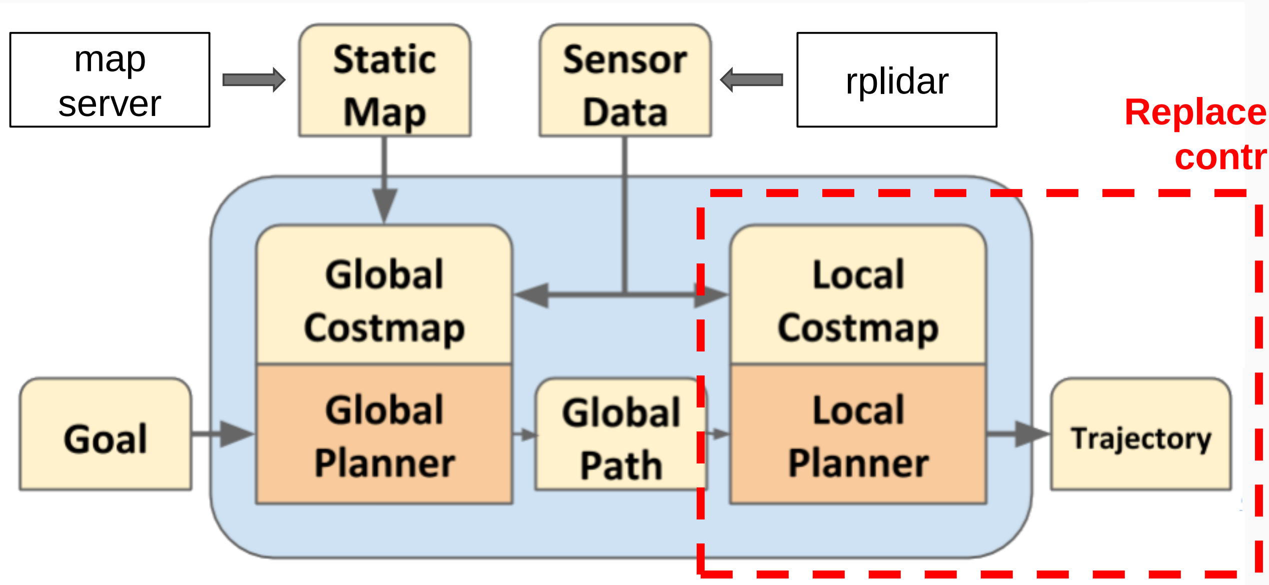
- **rosserial arduino**
  - [setup] http://wiki.ros.org/rosserial_arduino/Tutorials/Arduino%20IDE%20Setup
  - racecar_uno.ino [hypha-racecar/document/arduino/racecar_uno/]

- **Teleop testing**
  - [rosrun] $ rosrun rosserial_python serial_node.py   /dev/uno
  - [rosrun] $ rosrun rosserial_python serial_node.py   _port:=/dev/uno   _baud:=57600
  - [roslaunch]

    ```
    <!-- Arduino -->
     <node pkg="rosserial_python" type="serial_node.py" name="serial_node">
            <param name="port" value="/dev/uno"/>
            <param name="baud" value="57600"/>
     </node>
    ```
  - $ rosrun hypha_racecar racecar_teleop.py

Communication & Control

- **Overview of ROS Navigation**

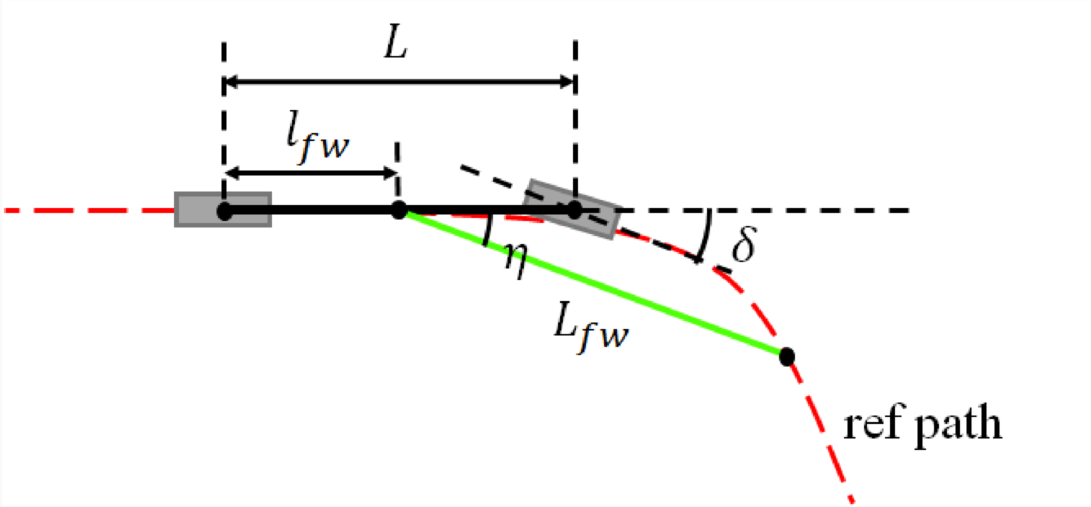


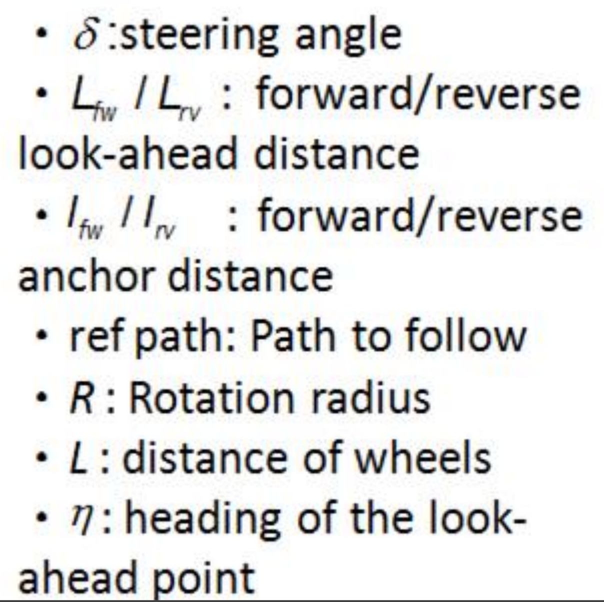
Replaced by L1 controller

figure source: http://roscon.ros.org/2014/wp-content/uploads/2014/07/ROSCON2014_DLu.pdf

Communication & Control

- **L1 controller**
  - paper: http://acl.mit.edu/papers/KuwataTCST09.pdf
  - paper: http://acl.mit.edu/papers/KuwataGNC08.pdf

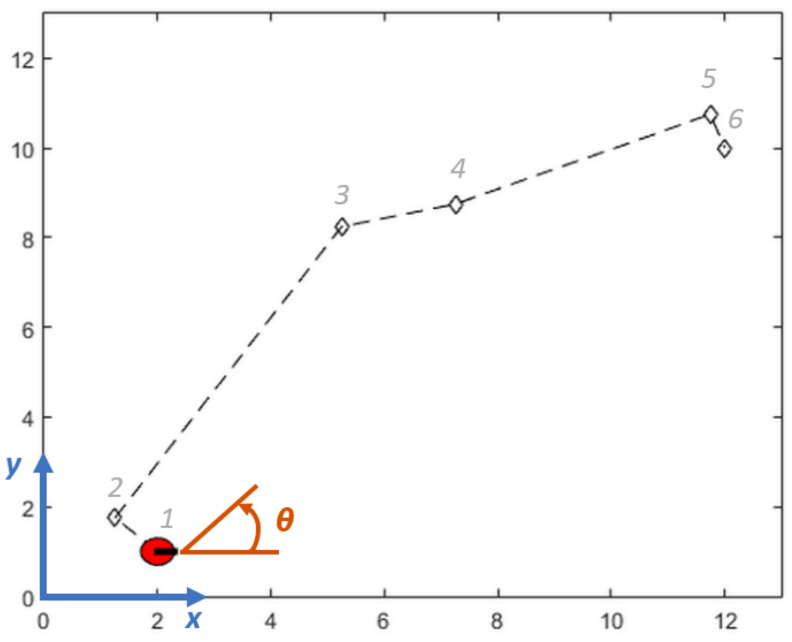


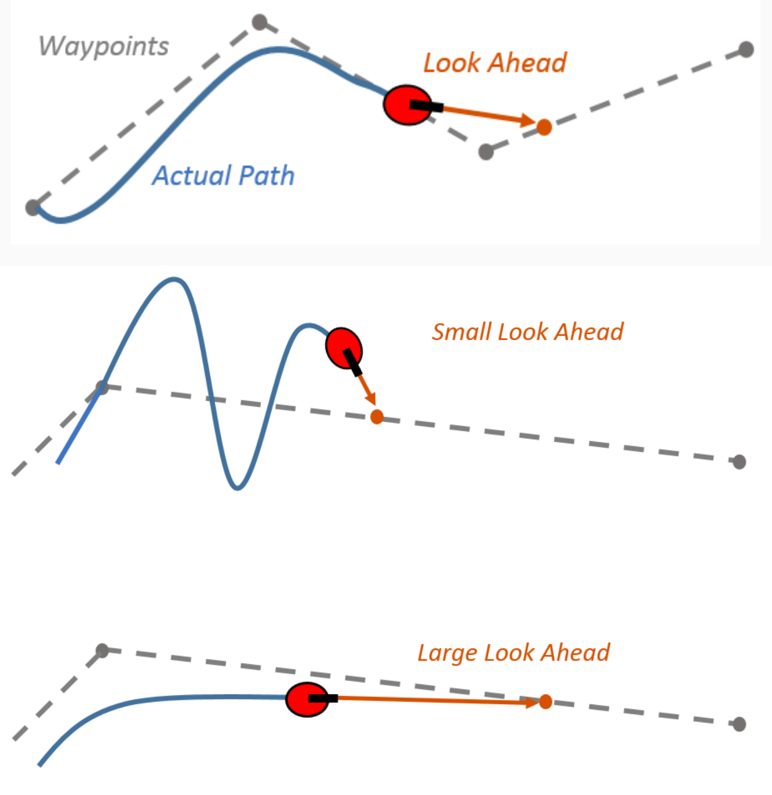


- δ :steering angle
- L_{fw} / L_{rv} : forward/reverse look-ahead distance
- l_{fw} / l_{rv} : forward/reverse anchor distance
- ref path: Path to follow
- R : Rotation radius
- L : distance of wheels
- η : heading of the look-ahead point

Communication & Control

- **L1 controller (Pure Pursuit Controller)**





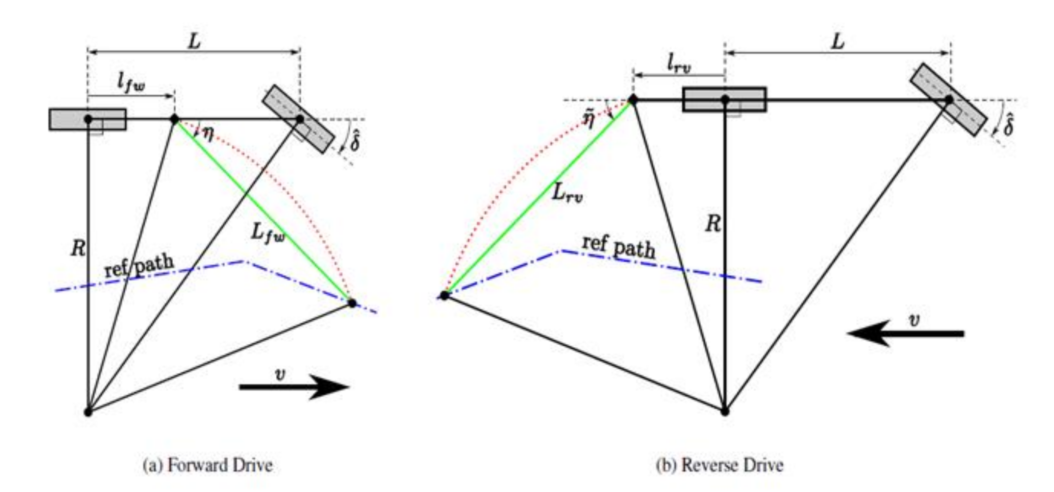
source: https://www.mathworks.com/help/robotics/ug/pure-pursuit-controller.html?requestedDomain=www.mathworks.com

Communication & Control

- **L1 controller Implemenation**
  - [source code] L1_controller_v2.cpp
  - Speed control: $u = K_p(v_{cmd} - v) + K_i \int_0^t (v_{cmd} - v)\,d\tau$
  - current version => constant speed

- u: motor command
- V_{cmd}: desired velocity
- V: current velocity

- **Path planning test**
  - RACECAR_amcl_nav.launch



(a) Forward Drive

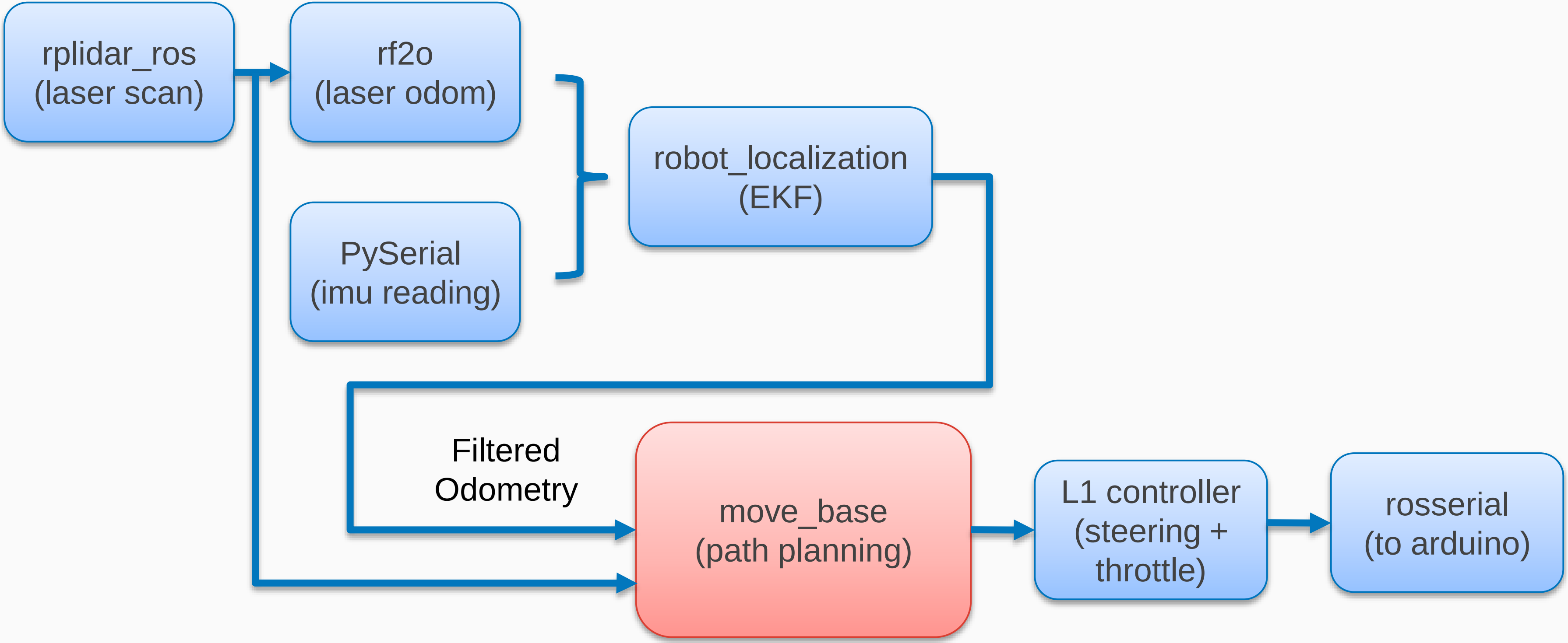
(b) Reverse Drive

Communication & Control

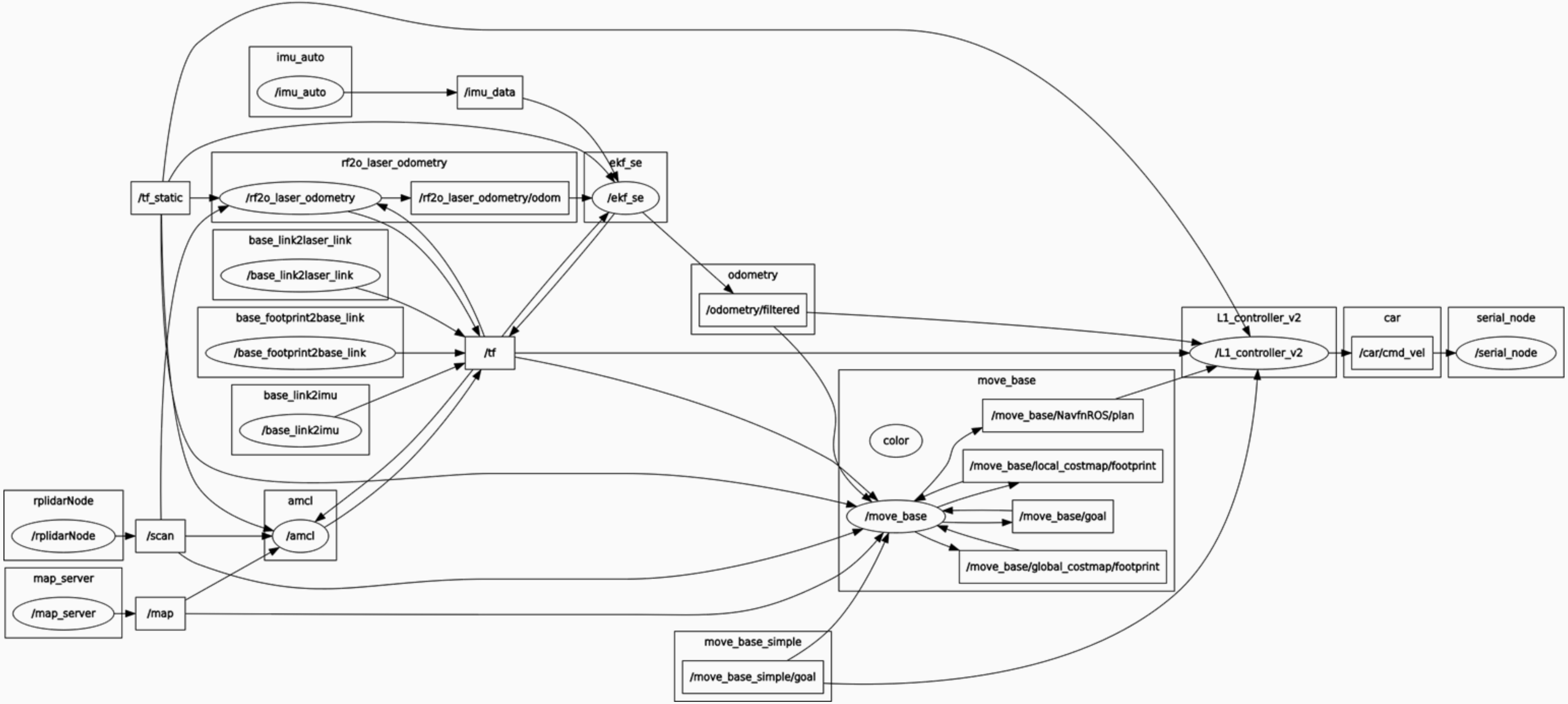
- **L1 controller launch file**

```
 <!-- L1 controller -->
  <node pkg="hypha_racecar" type="L1_controller_v2" respawn="false" name="L1_controller_v2" output="screen">
      <!-- L1 -->
      <param name="Vcmd" value="1.0" />

      <!-- ESC -->
      <param name="baseSpeed" value="1440"/>

      <!-- Servo -->
      <param name="baseAngle" value="90.0"/>
      <param name="AngleGain" value="-3.5"/>

      <remap from="/move_base_node/NavfnROS/plan" to="/move_base/NavfnROS/plan" />
  </node>
```

Software Architecture



ROS Nodes Graph



HYPHA ROS WORKSHOP

https://hypharosworkshop.wordpress.com/

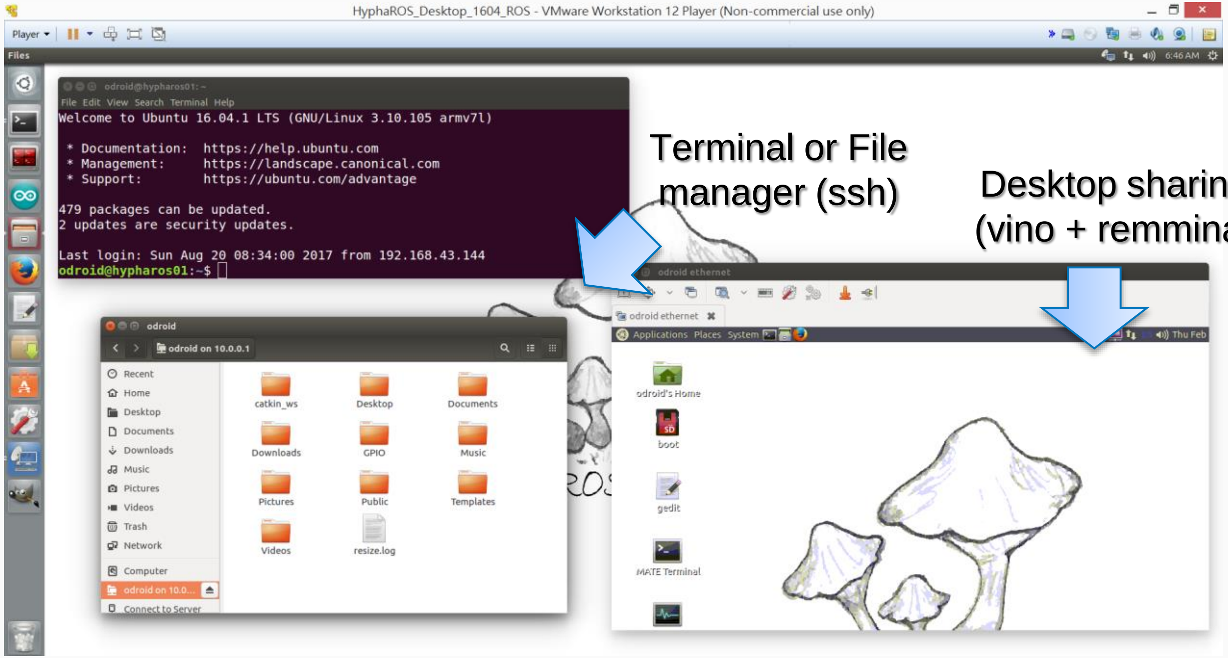
ROS

HyphaROS Github

All source code can be found on Hypha-ROS github:
https://github.com/Hypha-ROS/hypha-racecar

(including udev setting, Arduino files, necessary documents)

HyphaROS VM image

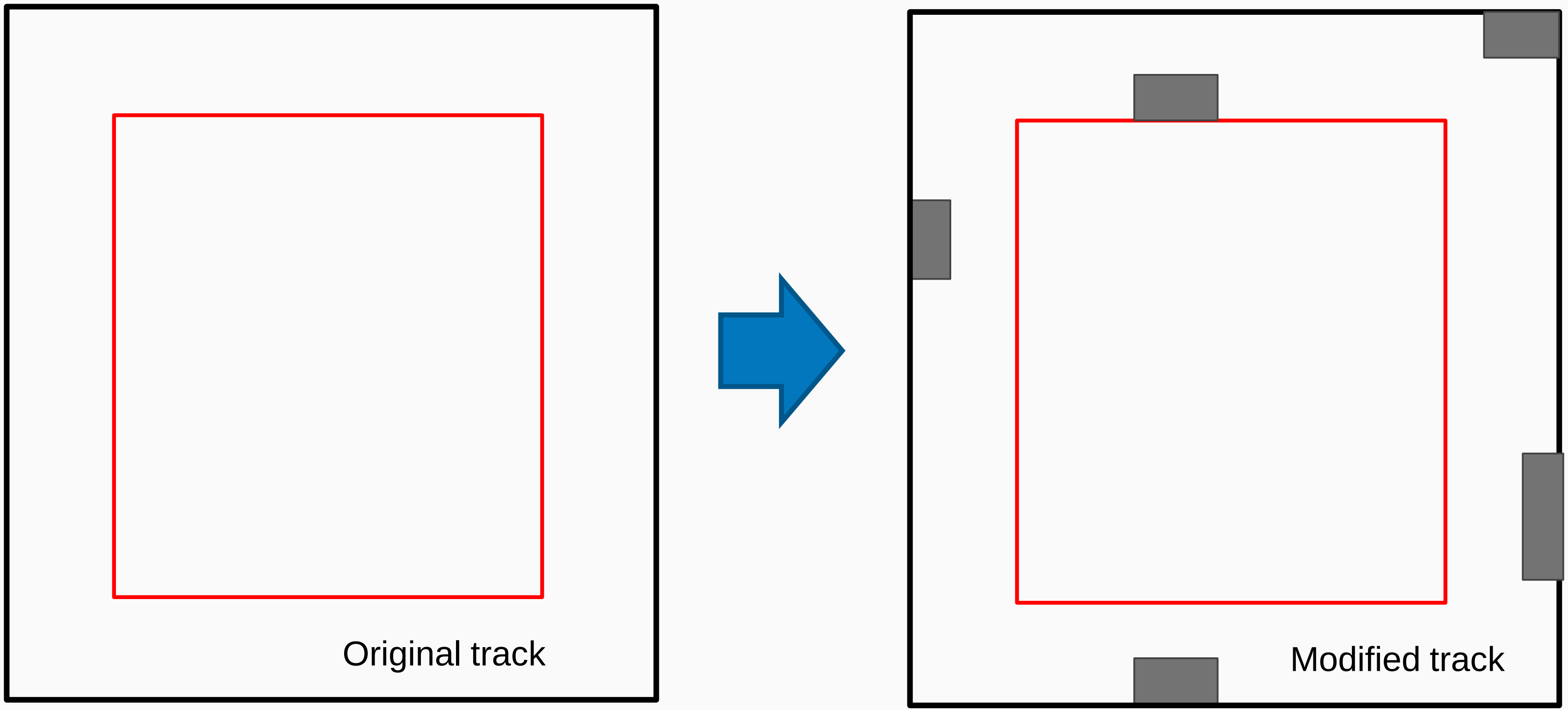


```
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 3.10.105 armv7l)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

479 packages can be updated.
2 updates are security updates.

Last login: Sun Aug 20 08:34:00 2017 from 192.168.43.144
odroid@hypharos01:~$
```

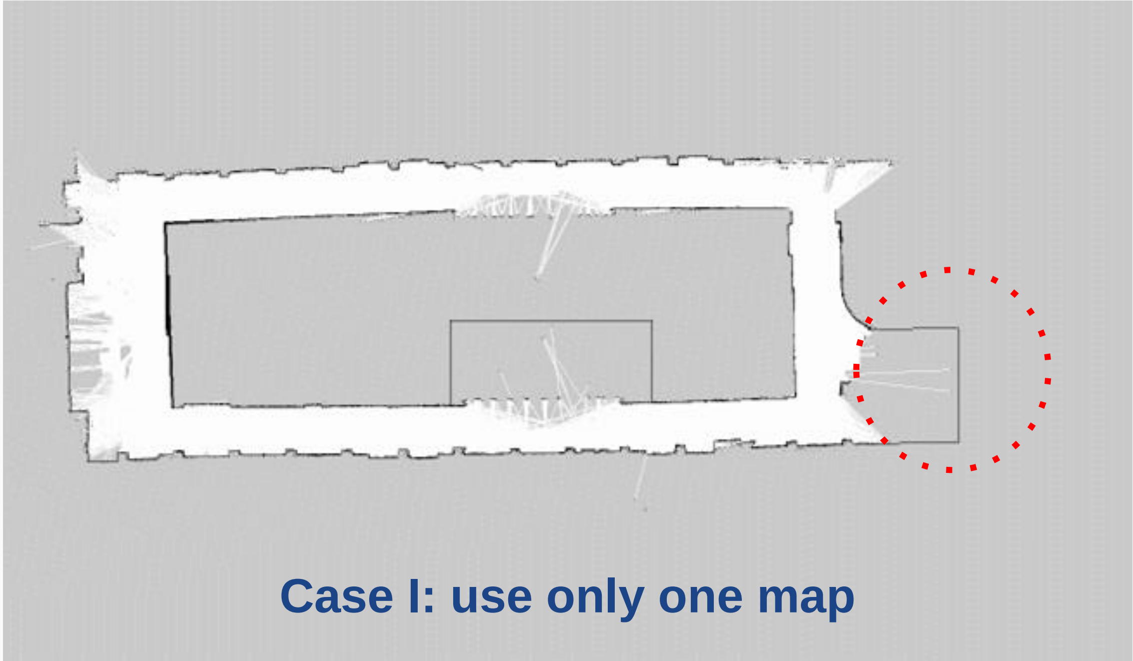
Terminal or File manager (ssh)

Desktop sharing (vino + remmina)

How to build the track

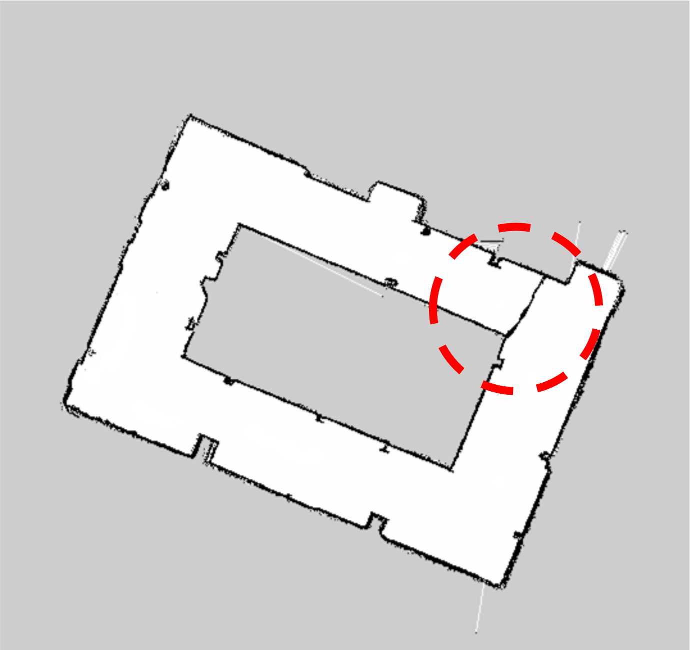


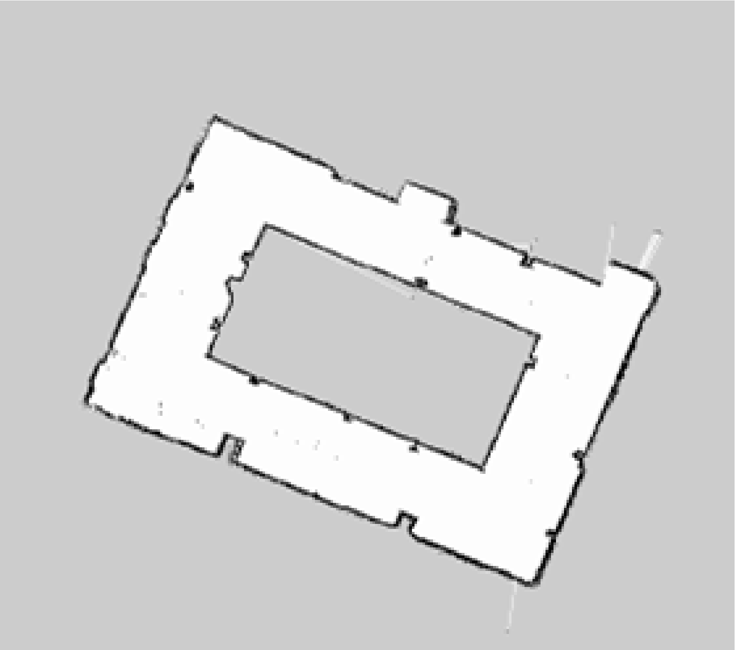
How to build the track



Case I: use only one map

How to build the track





Case II: use two maps, one for amcl, one for nav

Competition

Procedures

- [mapping] realtime, onboard
  - Test_gmapping.launch OR Test_icp_mapping.launch
- [mapping] realtime, desktop
  - Needs to modify launch file
- [mapping] offline, desktop
  - odroid: RACECAR_bag_record.launch
  - desktop: RACECAR_bag_icp.launch OR RACECAR_bag_gmapping.launch
- [edit map]
  - $ rosrun map_server map_saver
  - GIMP to edit map [support png and pgm]
- [navigation]
  - RACECAR_amcl_nav.launch

Roadmap

- Visual Odometry (SVO)
  - 120 fps mono-camera, ekf with laser-odom, imu
- English/Chinese Tutorial
  - May hold collaboration with students from worldwide
  - Step by step tutorial for fully beginners.
- 3D Obstacle Avoidance (rgbd or realsense)
  - Already have related experience on odroid SBC
- Official Released on ROS Community
  - Well documented readme, tutorial, introduction, etc.
- First open race for racecar in Taipei
  - May be held on the end of this year
- Advanced Tracking Controller
  - PI, MPCC(ethz)
- Migrate to ROS 2.0

Q & A

Hypha ROS Workshop

Website: https://hypharosworkshop.wordpress.com/
Github: https://github.com/Hypha-ROS/hypha-racecar
FB Page: https://www.facebook.com/HyphaROS/
Youku: http://i.youku.com/hypha
Gmail: hypha.ros@gmail.com
WeChat (ID): HyphaROS

Basic Operation [Appendix]

Basic Operation (Odroid x ROS)

- **[desktop] Download pkg from Hypha-ROS github**
  - $ cd catkin_ws/src
  - $ git clone https://github.com/Hypha-ROS/hypha-racecar

- **[desktop] SSH/Remmina to Odroid**
  - Wifi configuration (Hostname: hypharos0X, SSID: HyphaROS, pw: hypharos)
  - Ethernet configuration (default IP: 10.0.0.1)
  - ROS multi-machine env setting (in ~/.bashrc)
  - ssh: $ ssh odroid@192.168.X.1   (pw: hypharos) [Adhoc]
  - File manager: Connect to server -> ssh://odroid@192.168.X.1
  - remmina (vino pw: 0000)

- **[odroid] Modify wifi setting**
  - change hostame: $ sudo odroid-utility.sh (pw: hypharos)
  - ROS multi-machine env setting (in ~/.bashrc)

Basic Operation (Odroid x ROS)

- **[desktop] image backup**
  - Read: In hypha-racecar/document/commands/
  - $ sudo fdisk -l /dev/sdb
  - $ sudo dd if=/dev/sdb bs=512 count=29624319 of=~/HyphaROS_xu4_kinetic_20170610.img
  - Write: Ubuntu GUI

- **[odroid] Change adhoc setting**
  - $ sudo gedit /etc/network/interfaces (pw: hypharos)

- **[odroid] Change git config**
  - $ cd catkin_ws/src
  - $ git config --global user.name "YOUR NAME"
  - $ git config --global user.email YOUR EMAIL
  - $ git config --list   (to ckeck setting)

Basic Operation (Odroid x ROS)

- **[odroid] Create your own ROS pkg**
    - $ cd catkin_ws/src
    - $ catkin_create_pkg PACKAGE_NAME geometry_msgs move_base tf roscpp rospy std_msgs visualization_msgs

- **[odroid] Udev Setting**
    - $ lsusb (check the idV and idP of each component)
    - Select one for finding the ISB port
    - $ ls dev/ttyACM or ttyUSB
    - $ udevadm info -a /dev/ttyUSB0
    - $ sudo gedit /etc/udev/rules.d/99-NAME.rule
    - ⇒ KERNEL=="ttyUSB*", ATTRS{idProduct}=="ea60", ATTRS{idVendor}=="10c4", MODE="666", ATTRS{devpath}=="1.2.1.2", SYMLINK+="rplidar