Creating scalable customized robotic platforms

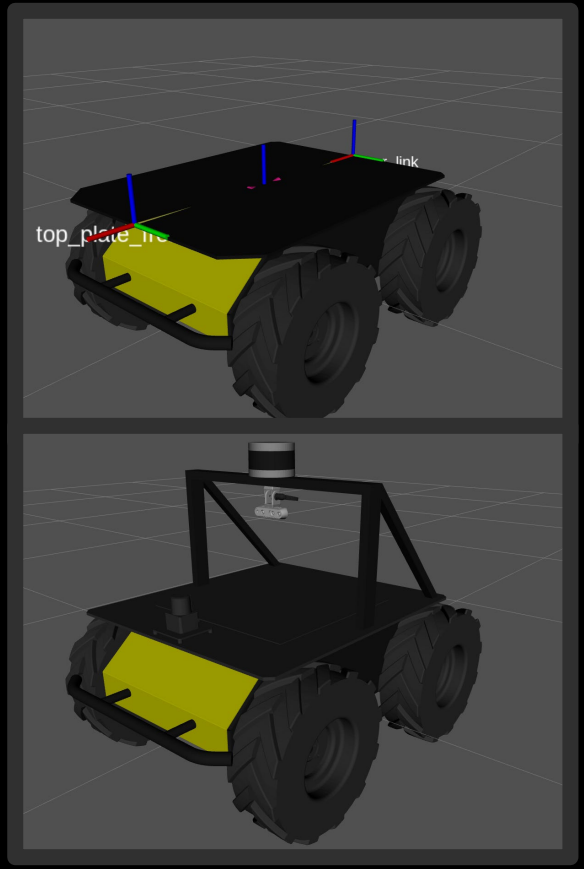
Rockwell Automation

CLEARPATH ROBOTICS

Luis Camero
Tony Baltovski
Roni Kreinin

Agenda

- Background
- Problem Statement
- Motivation
  - Limitations
  - Unifying the Ecosystem
- URDF-based Hardware
- Clearpath Configuration System
- Lessons Learned and Next Steps



Clearpath Husky model in RViz, with and without payloads

Background

- Six robotics platforms with distinct size and form factors.
- Over **20 different sensors** and **four different manipulator brands** for customers to customize the robots.
- Each robot platform's code base contained in a **separate GitHub organization**.
- Environment variables were introduced to add payloads to **xacro** and **launch** files.
- Each robot order is relatively unique, with custom hardware and software stored in an internal repository.

Customized clearpath robots.

Problem Statement

- Clearpath Robotics has a wide range of robot platforms and accessories with which our customers can tailor to their applications.

- As we scaled the number of robotics platforms, sensors, manipulators, and other robot accessories, our technical debt rapidly scaled and has hampered the development of tutorials for beginners, demos for users to extend, and developer tools for industry partners to leverage.

- How can we restructure and reimagine our robot customization system to minimize technical debt, reduce integration time, and improve user experience?

Motivation: Limitations

- Our divided code base duplicated common elements and lead to asymmetric development that ultimately limited progress.
- Environment variables do not scale well.
  - Instantiating payloads from environment variables requires unique variables for each payload.
  - The more variables the longer their names and more confusing they become
  - Large **xacro** files and **launch** files that become difficult to read and extend.
  - These variables varied among platforms.

```
export HUSKY_LASER_3D_SECONDARY_ENABLED=1
export HUSKY_LASER_3D_SECONDARY_HOST='192.168.131.21'
export HUSKY_LASER_3D_SECONDARY_TOPIC='secondary_points'
```

Example of environment variables to setup a 3D laser on Clearpath Husky.

Motivation: Unifying the Ecosystem

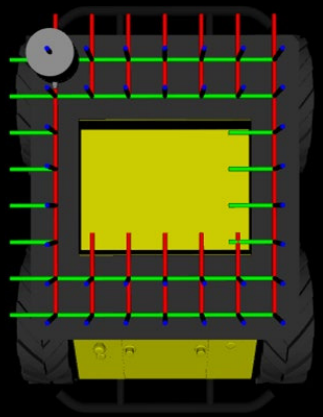
- Common plug and play software
  - Simulations that are easy to extend by students and researchers.
  - Navigation demos that work regardless of robot platform.
- Common API
  - Standardized topics and message types for all robots.
  - Facilitates software development across all our platforms.
- Common tutorials
  - Apply to all robot platforms.
  - Easy to maintain and reliable for users.

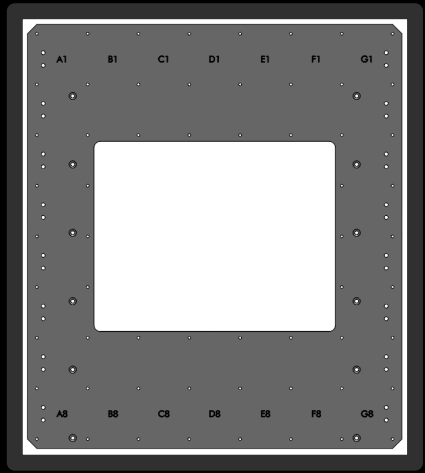
Topic	Message Type
sensors/lidar2d_#/scan	sensor_msgs/LaserScan
sensors/lidar3d_#/points	sensor_msgs/PointCloud2
sensors/lidar3d_#/scan	sensor_msgs/LaserScan
sensors/camera_#/color/image	sensor_msgs/Image
sensors/imu_#/data	sensor_msgs/Imu
sensors/gps_#/fix	sensor_msgs/NavSatFix

Clearpath API; sensor topics and their message types.

URDF-based Hardware

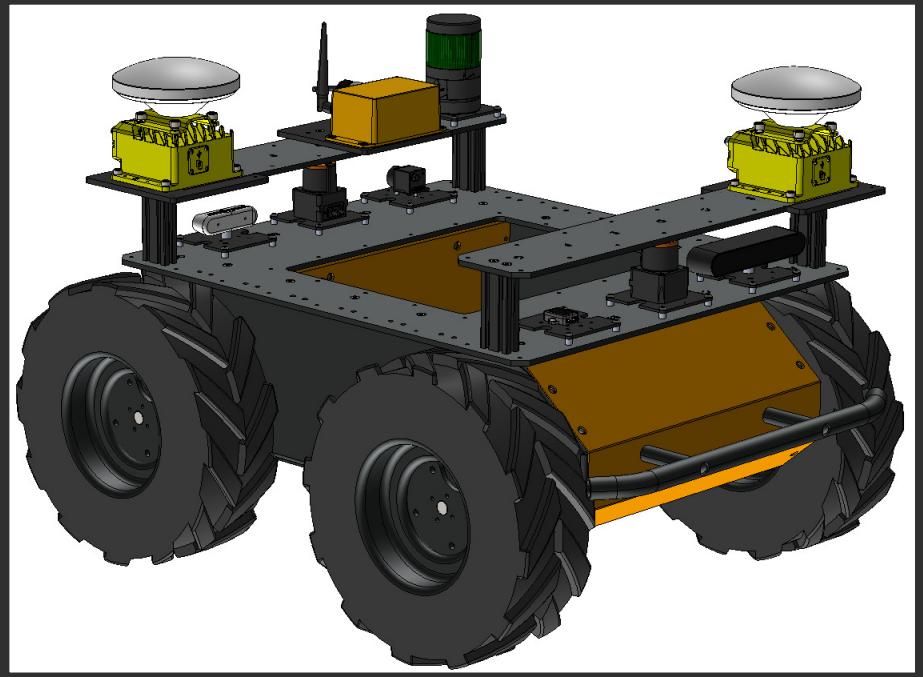
- Platform Attachment Configuration System (PACS™).
- 80x80mm grid of M5 x 0.8 threaded holes and URDF links that match the grid.
- Brackets that serve as an interface to attach any of our common sensors to the PACS™ grid.
- Repeatable robot building process.
- Enables users to upgrade and swap sensors.
- Change locations of sensors as use-case changes.
- Drawings available on the Clearpath Docs.





PACS™ husky top plate.

URDF-based Hardware

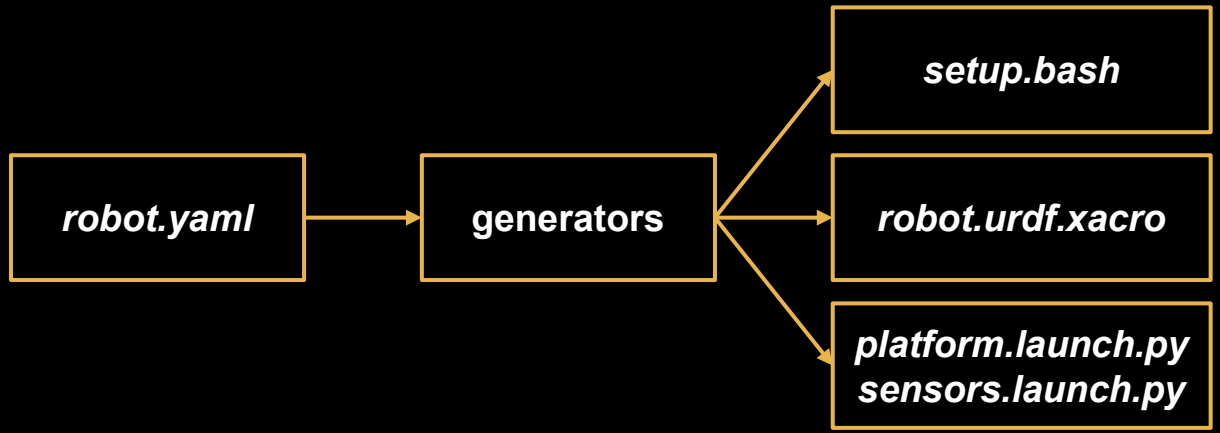


CAD of Clearpath Husky with PACS™ system.

Clearpath Dingo with PACS™.

Clearpath Configuration System

- Unify all Clearpath platforms under a single code base.
- Yet another YAML; we contain the entire robot system within one ***robot.yaml*** file.
- Generator scripts read contents of ***robot.yaml*** file and produce all files required to launch all nodes.



Clearpath Configuration YAML

- Contains all information about the system, divided in the following sections:
  - **system:** ROS 2 system information. Used to generate ***setup.bash***
  - **platform:** robot platform information, i.e. customizing controller parameters
  - **links:** exposes URDF primitive links to quickly add to the URDF.
  - **mounts:** mounting structures for sensors.
  - **sensors:** sensor description and launch parameters.

```
serial_number: a200-0000
version: 0
system:
  hosts:
    self: cpr-a200-0000
    platform:
      cpr-a200-0000: 192.168.131.1
    onboard: {}
    remote: {}
  ros2:
    username: administrator
    namespace: a200_0000
    domain_id: 0
    rmw_implementation: rmw_fastrtps_cpp
    workspaces:
      - /home/administrator/micro_ros_ws/install/setup.bash
```

Clearpath Configuration YAML sample **system** entries.

Clearpath Configuration YAML

Intel Realsense Example:

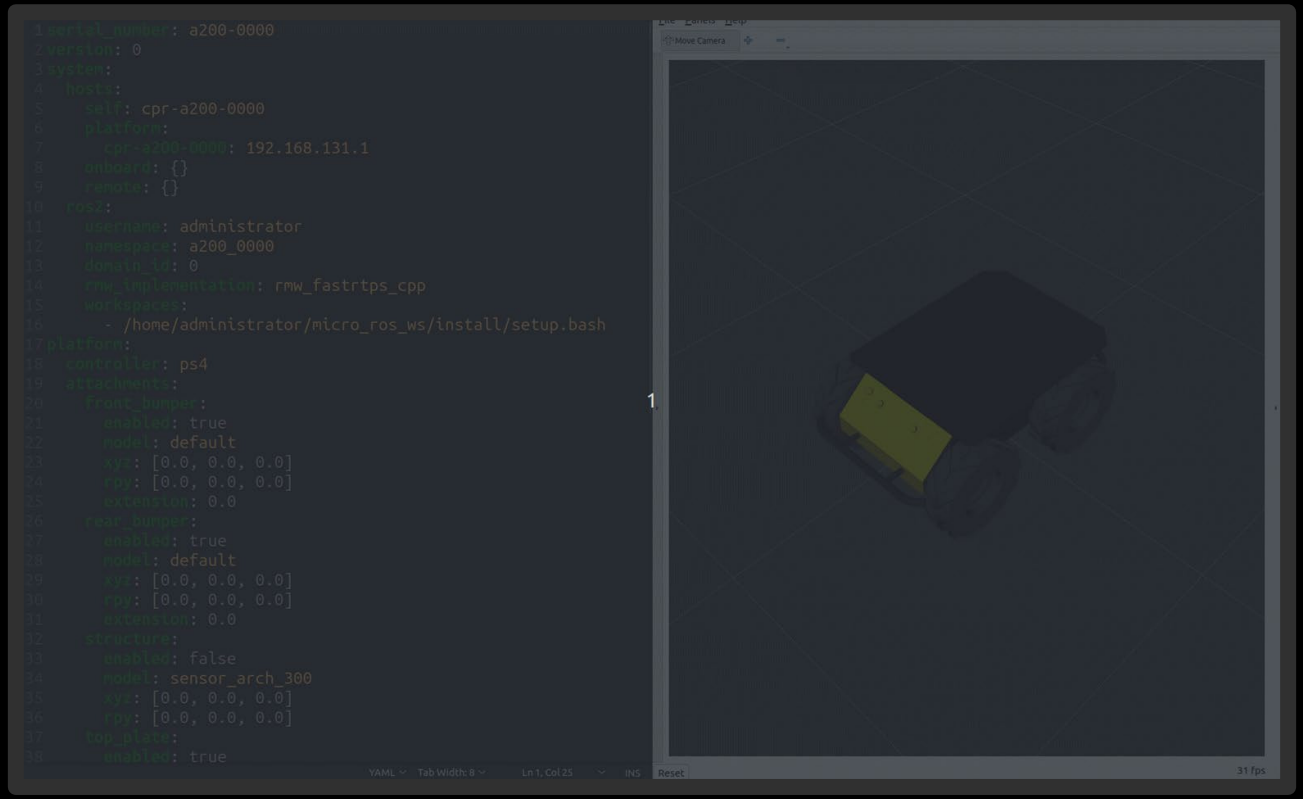
```
export HUSKY_REALSENSE_ENABLED=1
export HUSKY_REALSENSE_SERIAL='0'
export HUSKY_REALSENSE_TOPIC='realsense'
export HUSKY_REALSENSE_POINTCLOUD_ENABLED=1
export HUSKY_REALSENSE_DEPTH_ENABLED=1
export HUSKY_REALSENSE_DEPTH_FRAMERATE='30'
export HUSKY_REALSENSE_DEPTH_HEIGHT='480'
export HUSKY_REALSENSE_DEPTH_WIDTH='640'
export HUSKY_REALSENSE_COLOR_ENABLED=1
export HUSKY_REALSENSE_COLOR_FRAMERATE='30'
export HUSKY_REALSENSE_COLOR_HEIGHT='480'
export HUSKY_REALSENSE_COLOR_WIDTH='640'
export HUSKY_REALSENSE_PREFIX='camera'
export HUSKY_REALSENSE_PARENT='top_plate_link'
export HUSKY_REALSENSE_XYZ='0 0 0'
export HUSKY_REALSENSE_RPY='0 0 0'
```

Environment variables to setup a Realsense camera on a Clearpath Husky.

```
camera:
 - model: intel_realsense
   urdf_enabled: true
   launch_enabled: true
   parent: base_link
   xyz: [0.0, 0.0, 0.0]
   rpy: [0.0, 0.0, 0.0]
   ros_parameters:
     camera:
       camera_name: camera_0
       device_type: d435
       serial_no: "0"
       enable_color: true
       rgb_camera.profile: 640,480,30
       enable_depth: true
       depth_module.profile: 640,480,30
       pointcloud.enable: true
```

Clearpath Configuration YAML sample to add an Intel Realsense on *any* robot.

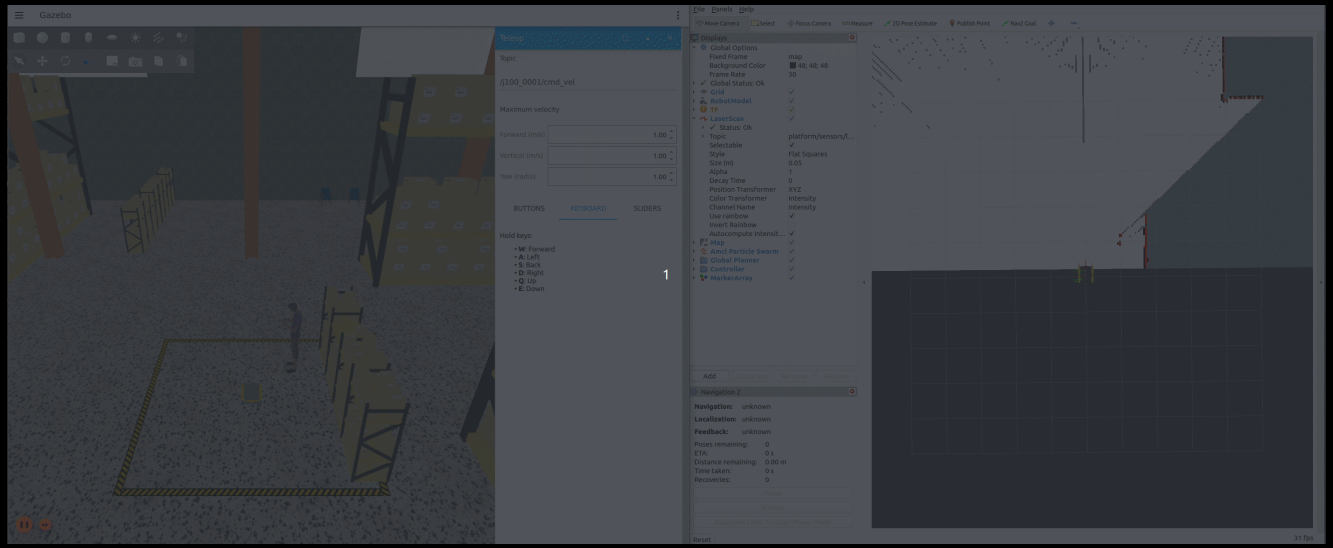
Customize Robots Live



Clearpath Configuration Live; re-generating URDF model

Simulate in Gazebo

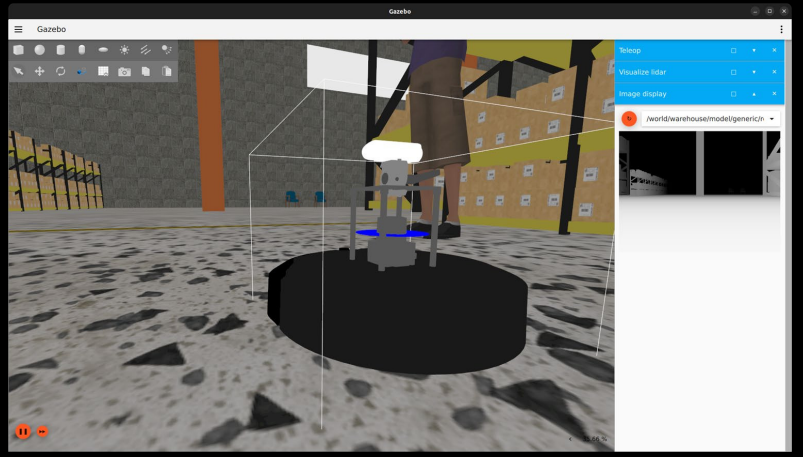
When it comes to simulations, the ***robot.yaml*** provides all the information the generators require to launch a Gazebo simulation and a bridge for every sensor added.

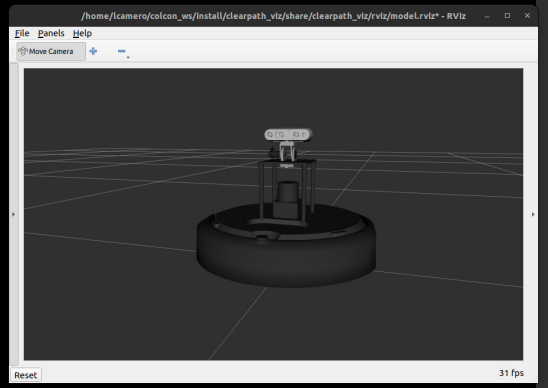


Gazebo simulation generated using the Clearpath Configuration System.

Lessons Learned and Next Steps

- Re-imagine building robots with ROS tools in mind to facilitate ROS integration.
- Common code base of all platforms and common configuration systems to build better tutorials, demos, and API tools.
- Extend configuration tools for the generic case to integrate all payloads with non-Clearpath robots in the ecosystem.
- Support more platforms, support every sensor, and add manipulators to the system.





Create 3 Robot configured using the Clearpath Configuration tool.

Thank you

Any questions?

https://docs.clearpathrobotics.com/docs/robots/