ROS2:
Supercharging the Jaguar 4x4

Toffee Albina (TRI) and Chris Lalancette (Open Robotics)

Outline

- Introduce the project
- ROS2 base node
- ROS2 manipulator node
- ROS2 camera node
- ROS2 navigation
- Wrapup

open robotics

TOYOTA RESEARCH INSTITUTE

Dr Robot Jaguar 4x4

Jaguar 4x4

- Outdoor wheeled mobile base by Dr Robot
- Diff drive (4 wheels - replaced rear wheels with casters)
- Axis cameras
- GPS
- IMU
- Laser scanner
- Optional 3DOF + gripper manipulator
- Communication over ethernet/wireless; different IPs/ports for base, camera, manipulator

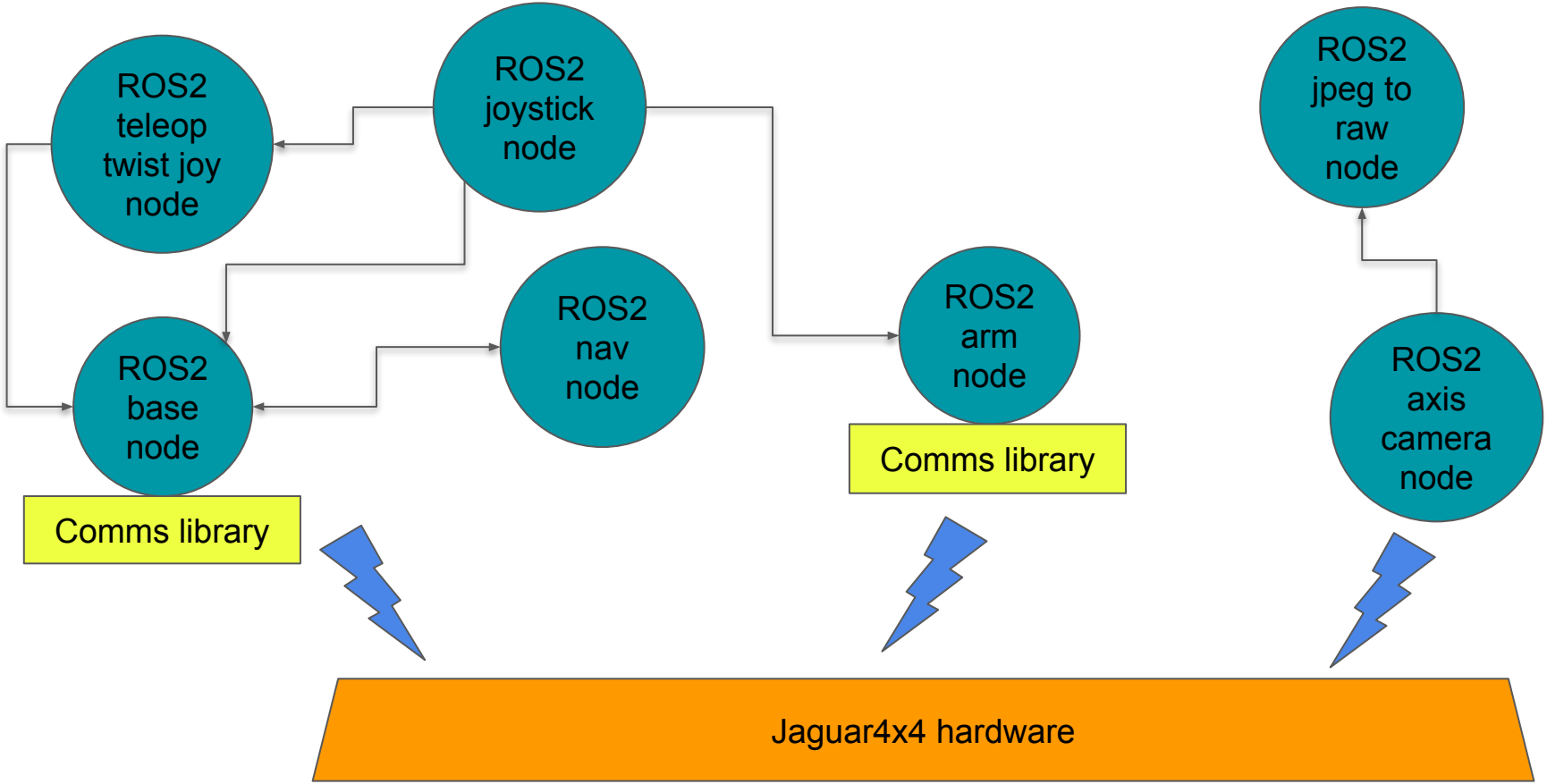
open robotics

TOYOTA RESEARCH INSTITUTE

Project

- Goals
  - Partnership
    - TRI funding OSRC core ROS2 development
    - Additionally, TRI partners with OSRC to develop ROS2 projects for TRI
  - Get experience building a ROS2 robot
  - Is ROS2 at the point where a robot can be controlled via joystick?
  - Is ROS2 at the point where a robot can complete a navigation task?
  - Jaguar 4x4 is existing platform that allowed us to get started answering these questions until custom hardware was available
  - Open source all of the code for the Jaguar 4x4: https://github.com/TRI-jaguar4x4

ROS2 Software block diagram



open robotics

TOYOTA RESEARCH INSTITUTE

ROS2 concepts

- Rest of talk will further explain pieces of block diagram
- Also discuss problems encountered:
  - How to get multiple ROS2 callbacks executing in parallel?
  - How to use ROS2 parameters for tuning?
  - How to write new-style ROS2 launch files?

Jaguar 4x4 Base

ROS2 Jaguar 4x4 base node

- Goals
  - Get feedback from the robot (temperature, encoders, IMU, etc)
  - Drive the robot via joystick
  - Autonomously drive a set distance
- Comms: https://github.com/TRI-jaguar4x4/jaguar4x4_comms
- Base: https://github.com/TRI-jaguar4x4/jaguar4x4_base

ROS2 Jaguar 4x4 base node robotics

- From scratch C++ ROS2 node
  - Uses shared comms library with the manipulator
  - Threading (future/promise)
  - Ping thread
  - Twist callback
  - Joystick callback (EStop)
  - Opportunistic gyro bias
  - Position updates from odometry
  - Service to drive set distance

ROS2 Jaguar 4x4 base node EStop

- Need software-defined EStop
- Subscribe to joystick channel directly; use button on joystick to EStop
- By default, ROS2 runs all service/topic callbacks on one thread
- Long-running service blocks joystick callback from running
- ROS2 Solution is Multi-threaded Executors and callback groups
  - Multi-threaded executors: multiple threads operate on ‘queue’ of work
  - Callback groups: All callbacks in a group handled by single thread
- Jaguar 4x4 base has all callbacks on single thread except for joystick

Multithreaded Executor/Callback Groups

- In main:

```
rclcpp::executors::MultiThreadedExecutor executor;
auto base = std::make_shared<Jaguar4x4Base>("192.168.0.60", 10001);
executor.add_node(base);
executor.spin();
```

- In constructor for Jaguar4x4Base node:

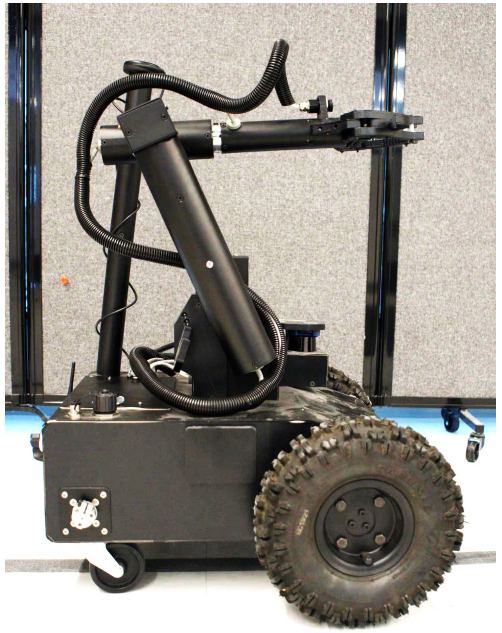
```
// separate callback group for joystick callback
joy_cb_grp_ = this->create_callback_group(
  rclcpp::callback_group::CallbackGroupType::MutuallyExclusive);
joy_sub_ = this->create_subscription<sensor_msgs::msg::Joy>("joy",
  std::bind(&Jaguar4x4Base::joyCallback, this, std::placeholders::_1),
  Cmd_vel_qos_profile, joy_cb_grp_);
```

Jaguar 4x4 Manipulator

ROS2 Jaguar 4x4 manipulator node

- Goals
  - Get feedback from the manipulator (encoders, etc)
  - Drive the manipulator via joystick
  - Autonomously move to a particular position in space
- Manipulator:

  https://github.com/TRI-jaguar4x4/jaguar4x4_arm

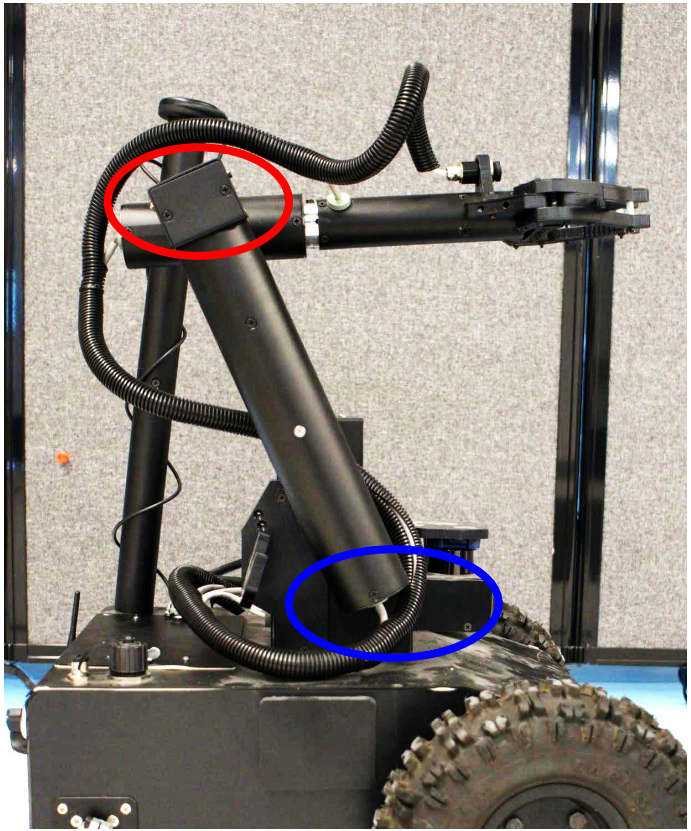


ROS2 Jaguar 4x4 manipulator

- From scratch C++ ROS2 node
  - Uses shared comms library with the base
  - Joystick callback (EStop, move “shoulder” and “elbow” joints)
  - Same threading concept as base
  - Same ping thread concept as base
  - Service to home manipulator joints

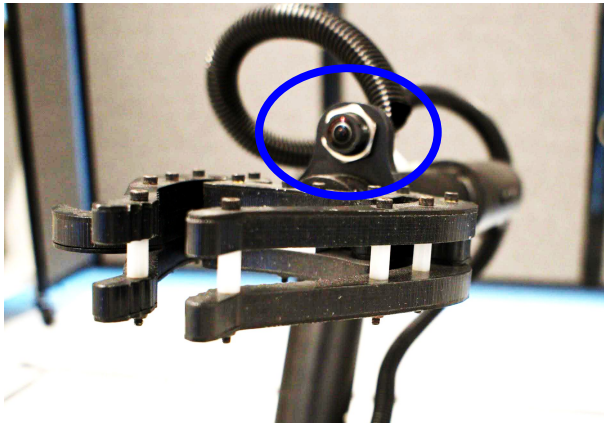
ROS2 Jaguar 4x4 manipulator robotics

- Try to calibrate manipulator position (no home position)
- Able to calibrate “shoulder” joint based on encoder feedback
- Could not do same for “elbow” joint (not robust)
- Lesson learned: can’t really calibrate this arm automatically; need to joystick to home position before controlling autonomously



Jaguar 4x4 Camera

Jaguar 4x4 manipulator camera



- Goal: Comms with Axis camera (P-1224-E)
- Started with ROS1 Axis camera driver:

  http://wiki.ros.org/axis_camera
- Ported to ROS2 for this project:

  https://github.com/TRI-jaguar4x4/axis_camera
- Camera output is JPEG; needed jpeg_to_raw node:

  https://github.com/TRI-jaguar4x4/jpeg_to_raw

Jaguar 4x4 Navigation

ROS2 Jaguar 4x4 Navigation

- Goal
  - Go to goal pose
  - Navigate over a short distance
    - Closed-loop navigation not required
    - Use odometry/IMU only
- Implemented as a service
- Used parameters for easily tuning constants
- Nav: https://github.com/TRI-jaguar4x4/jaguar4x4_nav

ROS2 Parameters

```
class Jaguar4x4Nav final : public rclcpp::Node
{
  Jaguar4x4Nav() : Node("jaguar4x4nav")
  {
    this->set_parameter_if_not_set(
      "velocity_constant",VELOCITY_CONSTANT_DEFAULT);
    this->set_parameter_if_not_set(
      "heading_constant",HEADING_CONSTANT_DEFAULT);
    this->set_parameter_if_not_set(
      "distance_epsilon", DISTANCE_EPSILON_DEFAULT);
  }

  std::string goToGoalXY(double goal_x_m, double goal_y_m)
  {
    this->get_parameter("velocity_constant", vel_const);
    this->get_parameter("heading_constant", h_const);
    this->get_parameter("distance_epsilon", dist_epsilon);
  }
```

ros2 param:

```
$ ros2 param list jaguar4x4nav
  distance_epsilon
  heading_constant
  Velocity_constant

$ ros2 param set jaguar4x4nav velocity_constant 0.4

$ ros2 param get jaguar4x4nav velocity_constant
  Double value is: 0.4
```

yaml file:

```
jaguar4x4_nav:
    ros__parameters:
        velocity_constant: 0.4
        heading_constant: 0.1
        distance_epsilon: 0.05
```


open robotics

TOYOTA RESEARCH INSTITUTE

ROS2 Nav Launch File

```
import os
import ament_index_python.packages
import launch
import launch_ros.actions

def generate_launch_description():
      jaguar_base = launch_ros.actions.Node(package='jaguar4x4_base', node_executable='jaguar4x4_base_node',
                                            output='screen')

      ttj_prm_file = os.path.join(ament_index_python.packages.get_package_share_directory('jaguar4x4'),
                                  'teleop_twist_joy_params.yaml')
      teleop_twist_joy = launch_ros.actions.Node(package='teleop_twist_joy',
                                                 node_executable='teleop_node',
                                                 output='screen',
                                                 arguments=['__params:=' + ttj_prm_file])

      return launch.LaunchDescription([jaguar_base, teleop_twist_joy, joy, jaguar_arm, jaguar_nav,
                              launch.actions.RegisterEventHandler(event_handler=launch.event_handlers.OnProcessExit(
                                          target_action=jaguar_base,
                                          on_exit=[launch.actions.EmitEvent(event=launch.events.Shutdown())],
                                          )),
      ])
```

ROS2 Jaguar 4x4 Nav in Action

- Go to goal pose (straight):

```
ros2 service call /go_to_goal_pose
jaguar4x4_nav_msgs/GoToGoalPose
'{goal_x_m:3.0, goal_y_m:0.0,
goal_theta_rad:0.0, speed_m_per_s:1.0}'
```

- Go to goal pose (diagonal):

```
ros2 service call /go_to_goal_pose
jaguar4x4_nav_msgs/GoToGoalPose
'{goal_x_m:2.0, goal_y_m:2.0,
goal_theta_rad:0.8, speed_m_per_s:1.0}'
```

Future Work

Future Work

- Work on the gyro
- Integrate gyro/odometry (EKF/robot_pose)
- Navigation PID loop (move_base?)
- Rewrite jpeg_to_raw in C++
- Try with cartographer

Wrapup

- Created a bare-bones Jaguar 4x4 system in ROS2
  - https://github.com/TRI-jaguar4x4
- Satisfactorily answered our questions about ROS2 usability
- Continuing to develop a more full-featured system on custom hardware
- Will continue to release software to the ROS2 community
- Continued TRI - OSRC partnership and and ROS2 collaboration

Thank You

Questions?