

Automated ROS code and ros_control generation



YJ Lim
Technical Product
Manager of robotics



Abhijeet Gadkari
Embedded Software
Engineer



Murat Belge
Consulting engineer

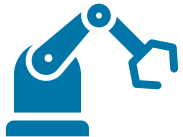
What MathWorks customers are telling us about ROS...



voyage

“Simulink + ROS allowed us to **deploy a Level 3 autonomous vehicle in less than three months.**”

— Alan Mond, Voyage



“By combining various toolboxes of MATLAB, it is easy to develop **advanced technology-based applications that are difficult to build with ROS alone**”

— Masaru Ken Morita, Yaskawa Electric Corp.



“.. ROS looks popular for industrial automation. How can I get started.”

— Lead Mechanical Engineering Manager, Industry Equipment

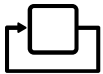
Agenda



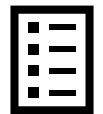
Introduction



Automated ROS code generation

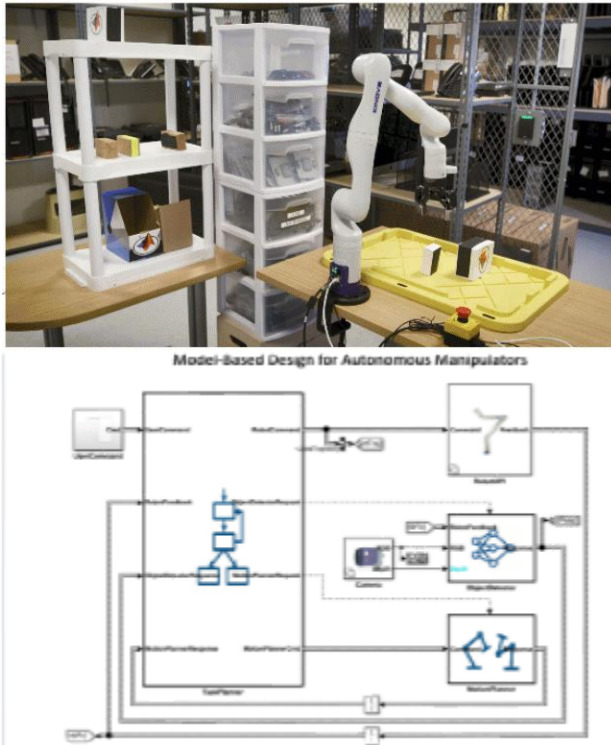


Automated ros_control generation



Summary

Introduction



Code
Generator

```
#include "AutomatedParkingValetAlgorithm.h"
#include "AutomatedParkingValetAlgorithm_private.h"

int32_T div_s32_floor(int32_T numerator, int32_T denominator)
{
    int32_T quotient;
    uint32_T absNumerator;
    uint32_T absDenominator;
    uint32_T tempAbsQuotient;
    boolean_T quotientNeedsNegation;
    if (denominator == 0) {
        quotient = numerator >= 0 ? MAX_int32_T : MIN_int32_T;
    }
    // Divide by zero handler
} else {
    absNumerator = numerator < 0 ? ~static_cast<uint32_T>(numerator) + 1U :
        static_cast<uint32_T>(numerator);
    absDenominator = denominator < 0 ? ~static_cast<uint32_T>(denominator) + 1U :
        static_cast<uint32_T>(denominator);
    quotientNeedsNegation = ((numerator < 0) != (denominator < 0));
    tempAbsQuotient = absNumerator / absDenominator;
    if (quotientNeedsNegation) {
        absNumerator %= absDenominator;
        if (absNumerator > 0U) {
            tempAbsQuotient++;
        }
    }

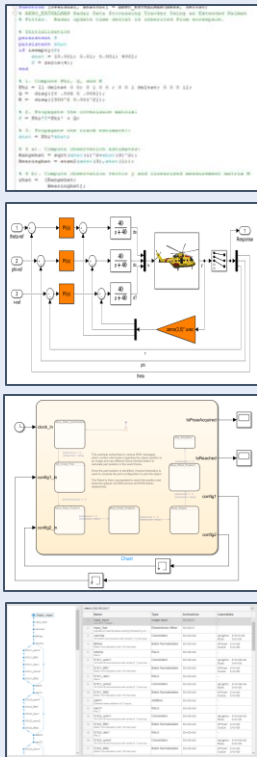
    quotient = quotientNeedsNegation ? -static_cast<int32_T>(tempAbsQuotient) :
        static_cast<int32_T>(tempAbsQuotient);
}

return quotient;
}

void AutomatedParkingValetModelClass::APV_emxInit_real_T(emxArray_real_T_T
    **pEmxArray, int32_T numDimensions)
```

Roles and Goals of Code Generation

User design in
high level /
domain specific
languages



**Code
Generator**

General purpose
programming
languages

C code

C++ code

HDL code

PLC code

GPU code



Roles

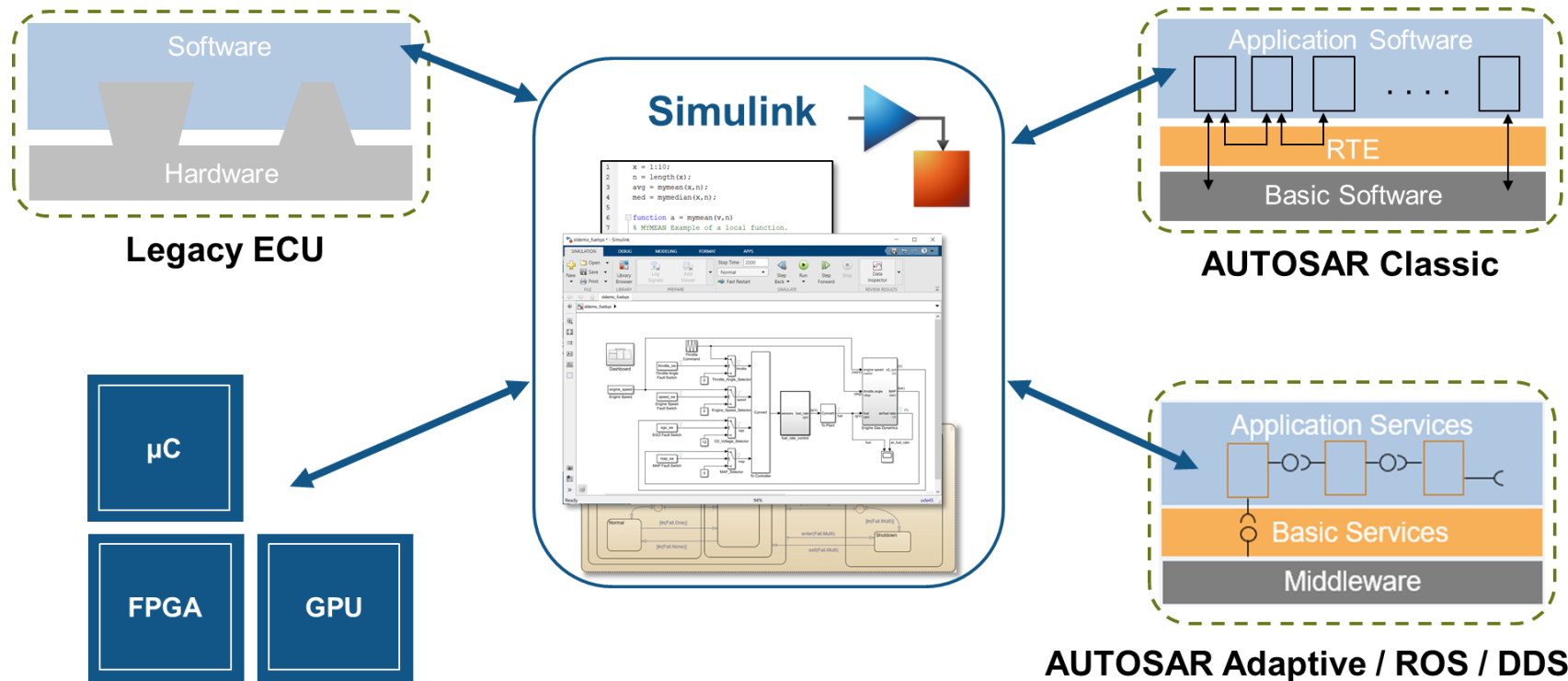
- Acceleration
- Production
- Verification

Goals

- Correct
- Efficient
- Customizable
- Certifiable
- Readable
- Scalable
- ...

Design software once, deploy to many targets

Migrate your existing tested components to new architectures while reusing existing workflows



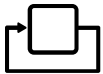
Agenda



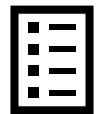
Introduction



Automated ROS code generation

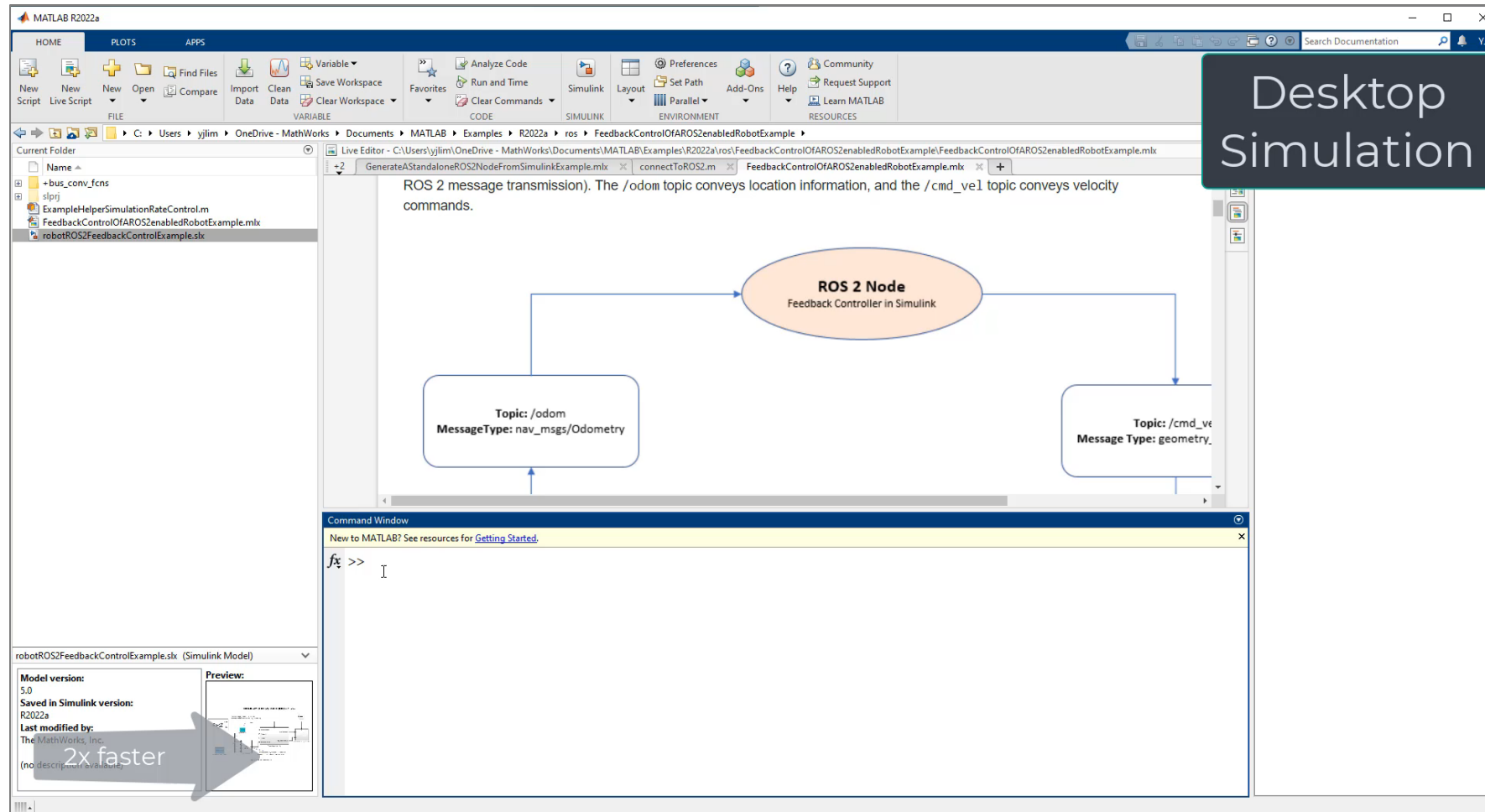


Automated ros_control generation

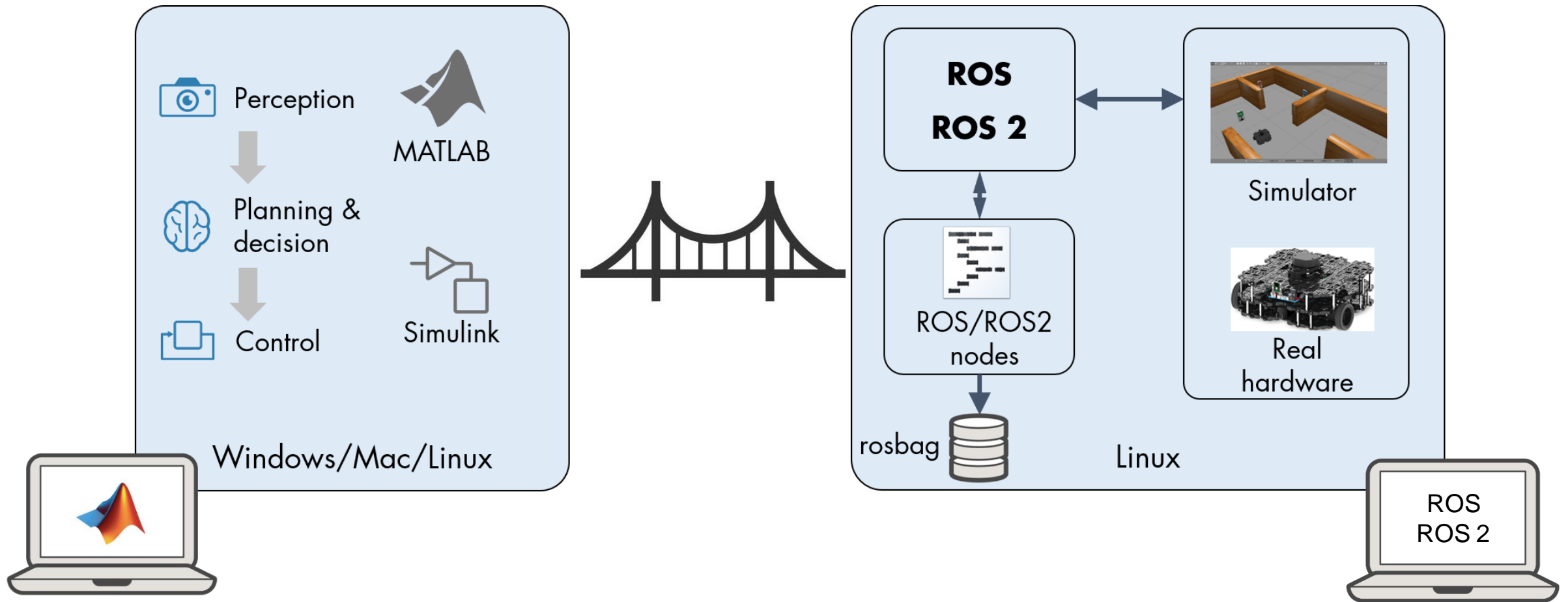


Summary

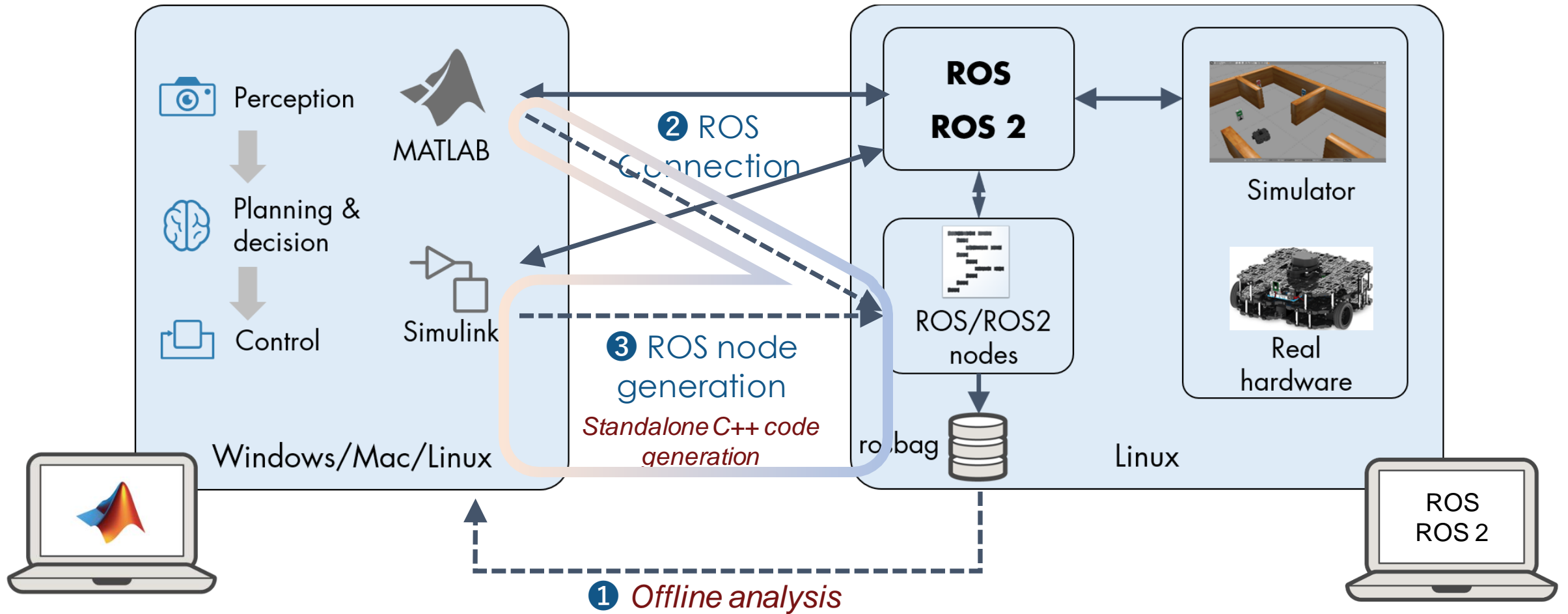
Automated ROS Code Generation



Bridging ROS / ROS 2 with MATLAB and Simulink



Bridging ROS / ROS 2 with MATLAB and Simulink



C++ ROS Code Generation from Simulink

The image shows a Simulink workspace with a model named 'signFollowingROS_deploy'. The 'Hardware Settings' block is highlighted, and a callout box shows the 'Robot Operating System (ROS)' and 'Robot Operating System 2 (ROS 2)' options. The 'Hardware Settings' block is also highlighted, and a callout box shows the 'Select Another Hardware Board' and 'Get Hardware Support Packages' options.

Robot Operating System (ROS)
Robot Operating System 2 (ROS 2)

Select Another Hardware Board
Get Hardware Support Packages

Configuration Parameters: signFollowingROS_deploy/Configuration (Active)

Search

Solver
Data Import/Export
Math and Data Types
Diagnostics
Hardware Implementation
Model Referencing

Hardware board: Robot Operating System (ROS)
Code Generation system target file: [ert.tlc](#)
Device vendor: Generic Device type: Unspecified (assume 32-bit Generic)
Device details

Configuration Parameters: mynode/Configuration (Active)

Search

Solver
Data Import/Export
Math and Data Types
Diagnostics
Hardware Implementation
Model Referencing
Simulation Target
Code Generation
Coverage
HDL Code Generation

Hardware board: Robot Operating System (ROS)
Code Generation system target file: [ert.tlc](#)
Device vendor: Generic Device type: Unspecified (assume 32-bit Generic)
Device details

Hardware board: AMD, ARM Compatible, Altera, Analog Devices, Atmel, Freescale, Infineon, Intel, Microchip, NXP, RISC-V, Renesas, STMicroelectronics, Texas Instruments, ASIC/FPGA, Custom Processor, **Generic**

OK Cancel Help Apply

C++ ROS Code Generation from Simulink

The image shows the Simulink interface for a project named 'signFollowingROS_deploy'. The 'Deploy to' dropdown menu is set to 'Remote Device (192.168.141.222)'. The 'Hardware Settings' button is highlighted. The 'Model Browser' on the left shows the project structure. The main workspace displays the title 'Sign Detection and Tracking with TurtleBot 3 (ROS)' and the copyright notice 'Copyright 2021 The MathWorks, Inc.'.

Yellow callout boxes highlight the following elements:

- 'Localhost' and 'Remote Device (192.168.141.222)' in the 'Deploy to' dropdown.
- 'Manage Remote Device' button.
- 'Build Model' button in the 'DEPLOYMENT' section.
- 'Build & Run' button in the 'DEPLOYMENT' section.

A 'Connect to ROS device' dialog box is open, showing the following fields:

- Device address: 192.168.141.222
- Username: user
- Password: (masked with dots)
- ☒ Remember my password
- ROS folder: /opt/ros/melodic
- Catkin workspace: ~/catkin_ws

Buttons at the bottom of the dialog: OK, Cancel, Test, Help.

C++ ROS Code Generation from Simulink

The screenshot shows the Simulink environment with the ROS tab selected. The Hardware Settings icon is highlighted with a blue box. A yellow arrow points from this icon to the Configuration Parameters dialog. The dialog is titled "Configuration Parameters: robotROS2FeedbackControlExample/Configuration (Active)".

Target selection

- System target file: ert.tlc
- Description: Embedded Coder
- Language: C++
- Language standard: C89/C90 (ANSI)

Build process

- ☐ Generate code only
- ☐ Package code and artifacts
- Zip file name: <empty>

Toolchain settings

Specify external ROS packages as dependencies

Build configuration: Specify

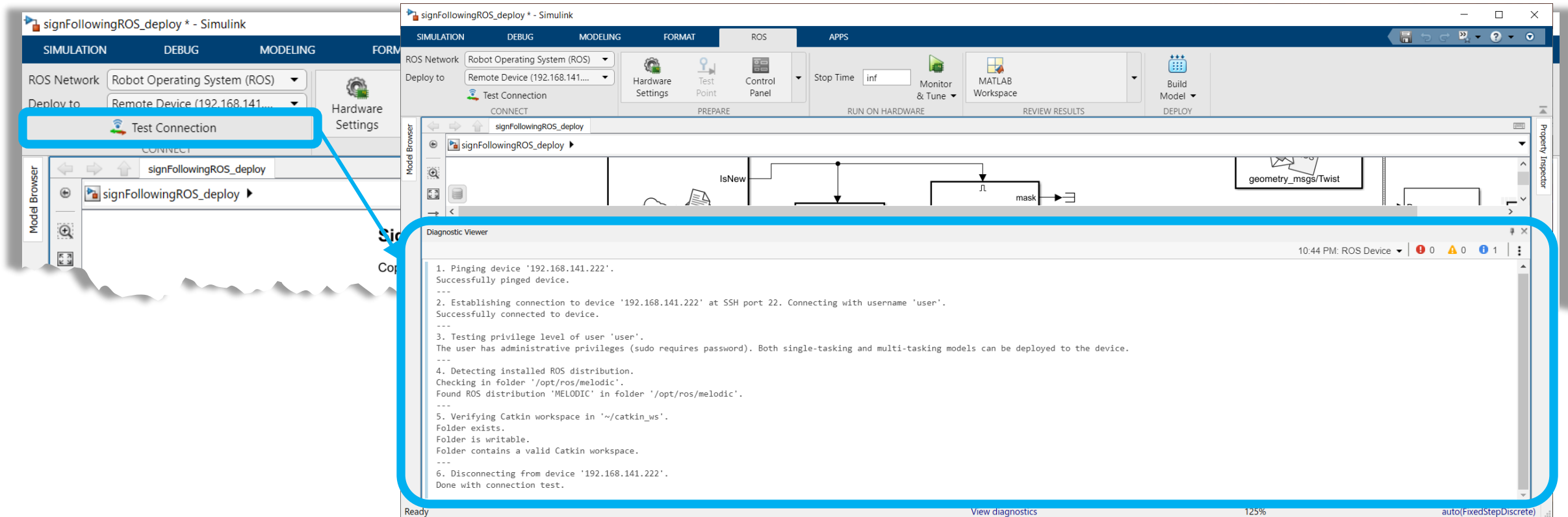
Tool	Options
Build Type	Release
Required Packages	<empty>
Include Directories	<empty>
Link Libraries	<empty>
Library Paths	<empty>
Defines	<empty>
CMake Target Type	Executable

Code generation objectives

Prioritized objectives: Unspecified

Buttons: OK, Cancel, Help, Apply

C++ ROS Code Generation from Simulink



C++ ROS Code Generation from MATLAB

Deploy MATLAB function as a C++ ROS code using MATLAB Coder

```
function myNode
% Copyright 2021 MathWorks Inc.

sub = rossubscriber('/point','geometry_msgs/Point',...
    @callback,...
    'DataFormat','struct');

fprintf('Created %s subscriber\n', sub.TopicName);
while (1)
    fprintf("Node is alive..\n");
    pause(3);
end

end % myNode

% Subscriber callback function
function callback(~,msg)
fprintf("(X,Y,Z): (%f,%f,%f)\n",msg.X, msg.Y,msg.Z);
end
```

```
cfg = coder.config('exe');
cfg.Hardware = coder.hardware(...
    'Robot Operating System (ROS)');
```



C++ ROS Code Generation from MATLAB

MATLAB Code

```
function myNode
% Copyright 2021 MathWorks Inc.

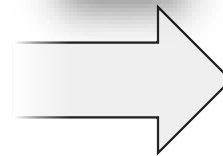
sub = rossubscriber('/point','geometry_msgs/Point',...
    @callback,...
    'DataFormat','struct');

fprintf('Created %s subscriber\n', sub.TopicName);
while (1)
    fprintf("Node is alive..\n");
    pause(3);
end

end % myNode

% Subscriber callback function
function callback(~,msg)
fprintf("(X,Y,Z): (%f,%f,%f)\n",msg.X, msg.Y,msg.Z);
end
```

codegen



C++ Code

```
void myNode()
{
    coder::ros::Subscriber sub;
    coder::timespec b_timespec;
    char varargin_1[7];
    if (!isInitialized_myNode) {
        myNode_initialize();
    }
    sub.matlabCodegenIsDeleted = true;
    sub.init();
    for (int i{0}; i < 6; i++) {
        varargin_1[i] = sub.TopicName[i];
    }
    varargin_1[6] = '\x00';
    printf("Created %s subscriber\n", &varargin_1[0]);
    fflush(stdout);
    while (1) {
        printf("Node is alive..\n");
        fflush(stdout);
        if (pauseState == 0) {
            b_timespec.tv_sec = 3.0;
            b_timespec.tv_nsec = 0.0;
            coder::TimeSleep(&b_timespec);
        }
    }
}
```


③ Automated ROS Node Generation to

Local Host



MATLAB/Simulink

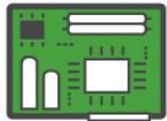
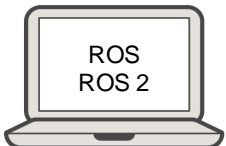


Build
Model



Build &
Run

Remote ROS Device



Build
Model

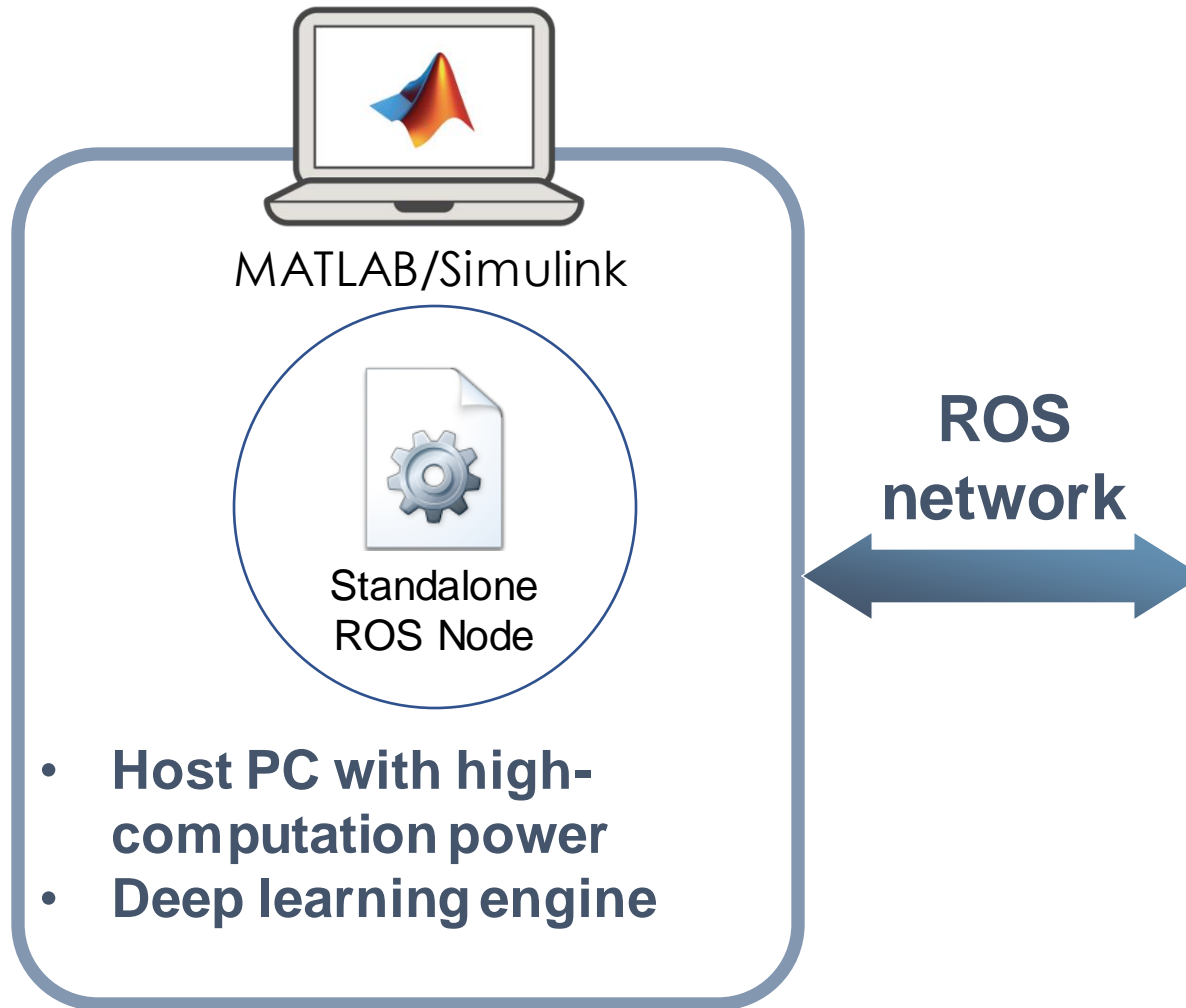
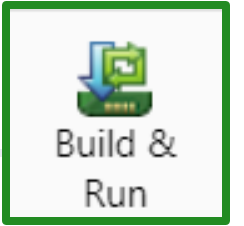


Build &
Load




Build &
Run

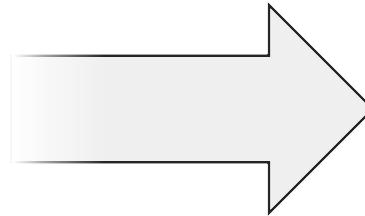
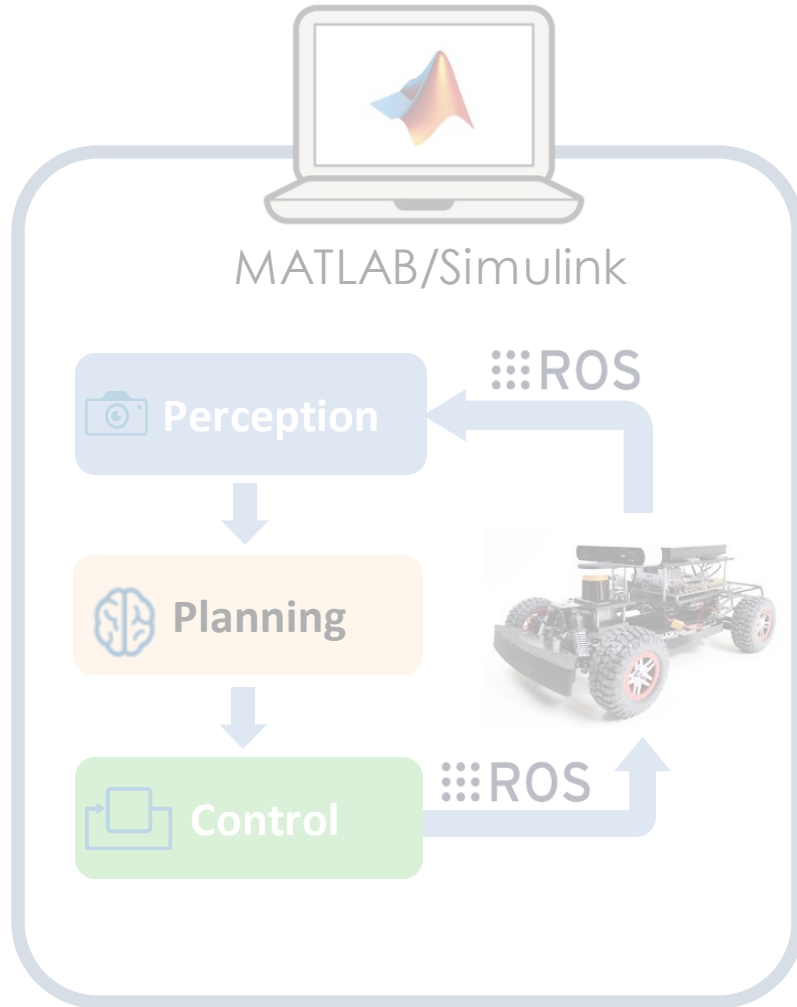
Use Case: Deploy ROS Code of Deep Learning Algorithm onto Development Machine



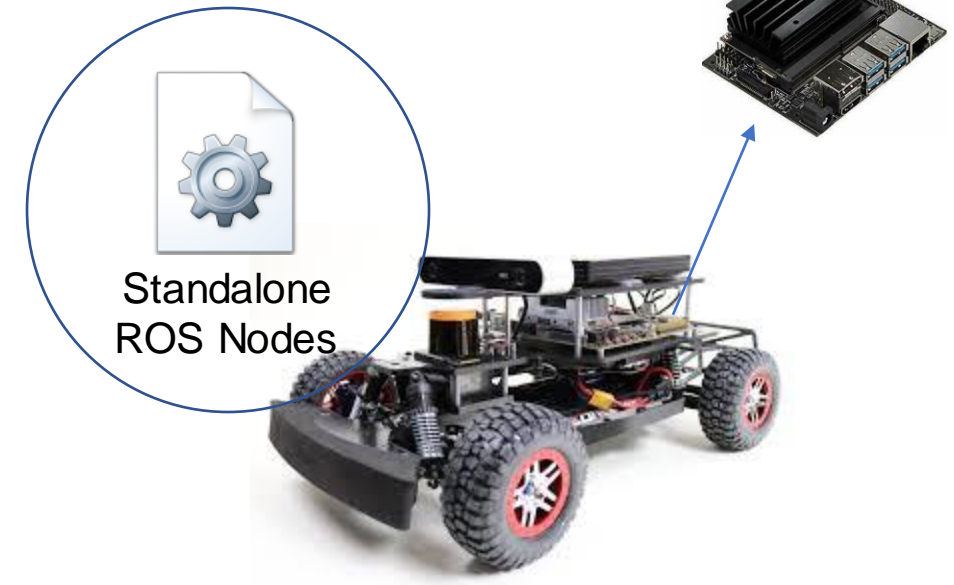
Use Case: Deploy a Standalone ROS Node to Remote Ground Robot


Build &
Load


Build &
Run



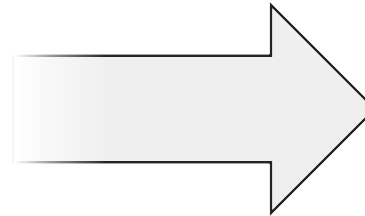
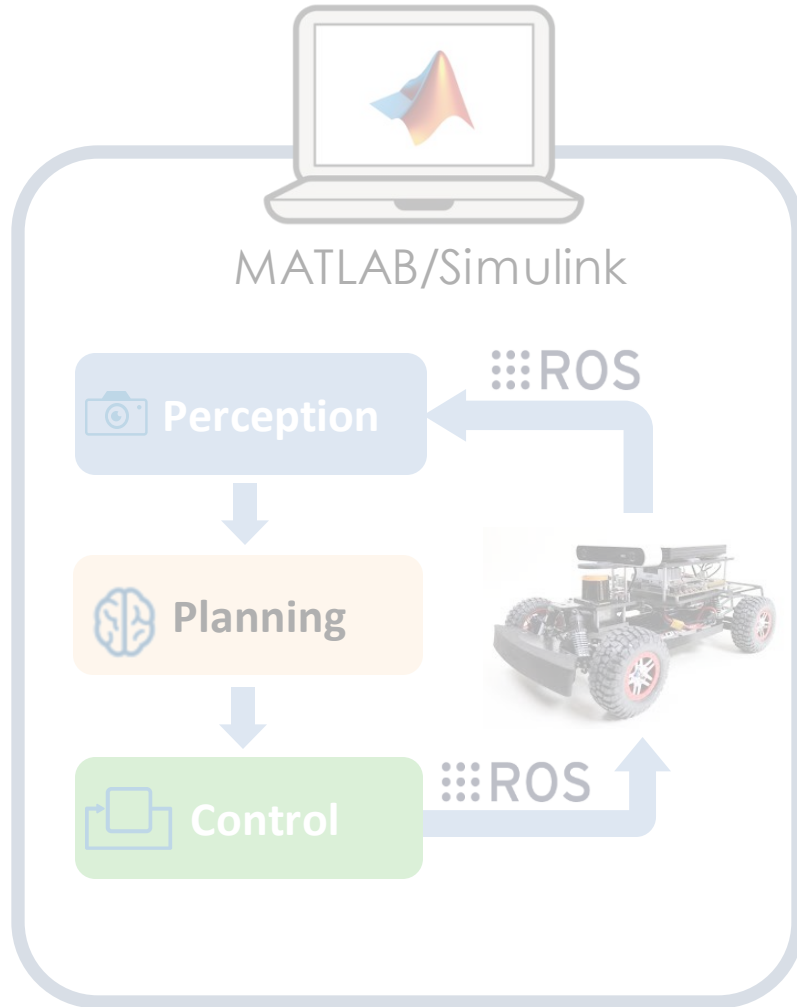
ROS-enabled Robot
with Nvidia Jetson



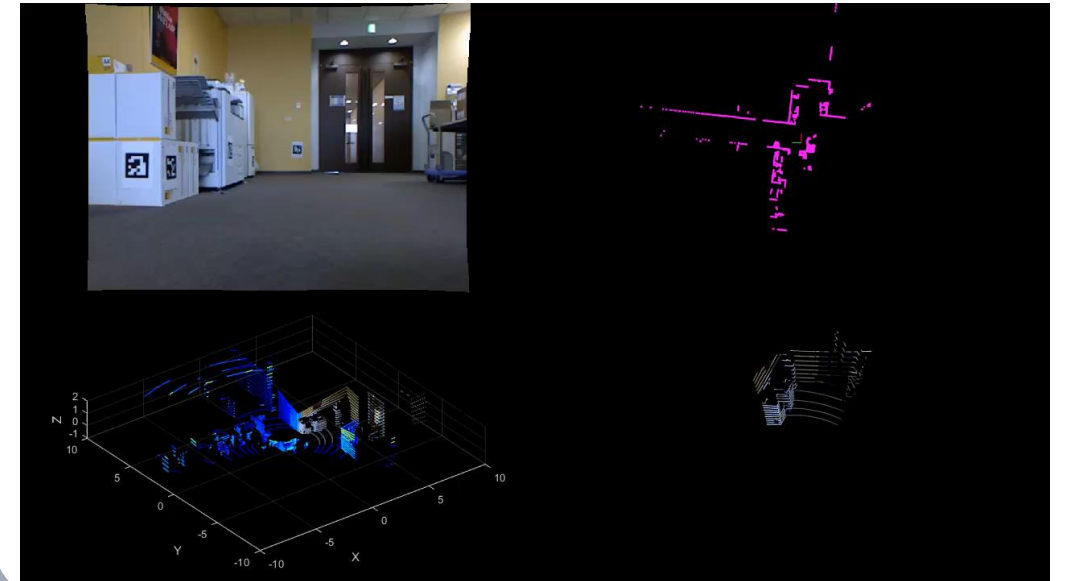
Use Case: Deploy a Standalone ROS Node to Remote Ground Robot

Build &
Load

Build &
Run



ROS-enabled Robot
with Nvidia Jetson



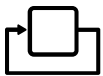
Agenda



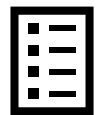
Introduction



Automated ROS code generation

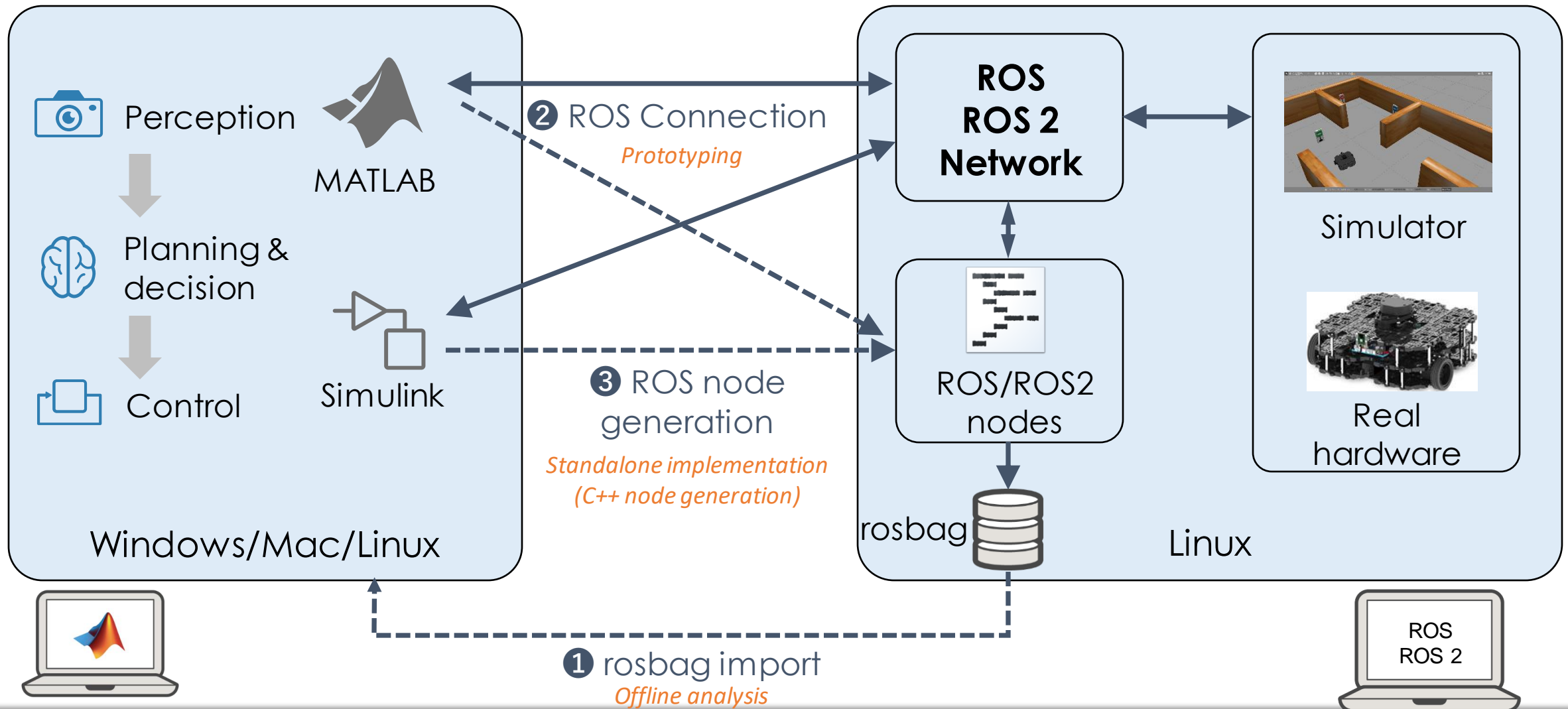


Automated ros_control generation

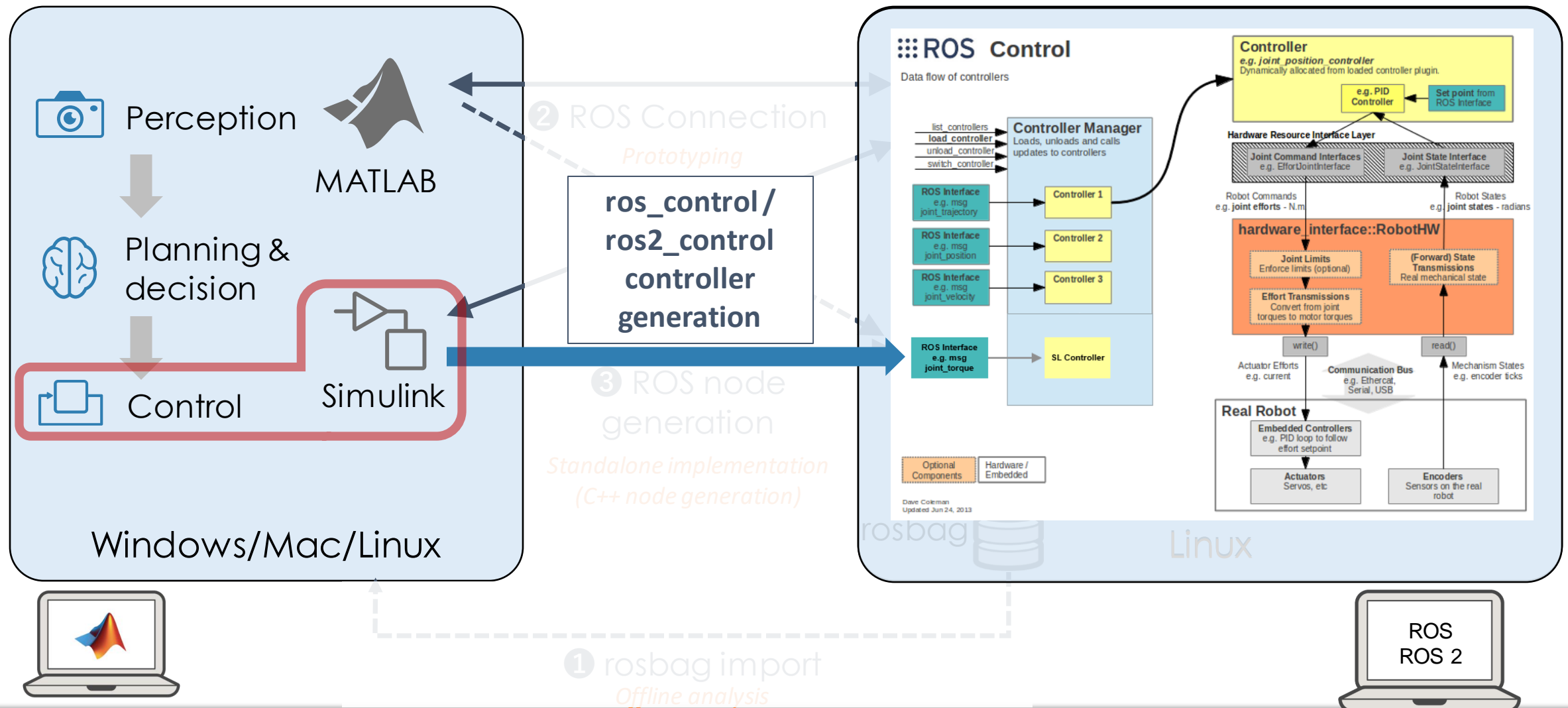


Summary

Connect MATLAB and Simulink with ROS

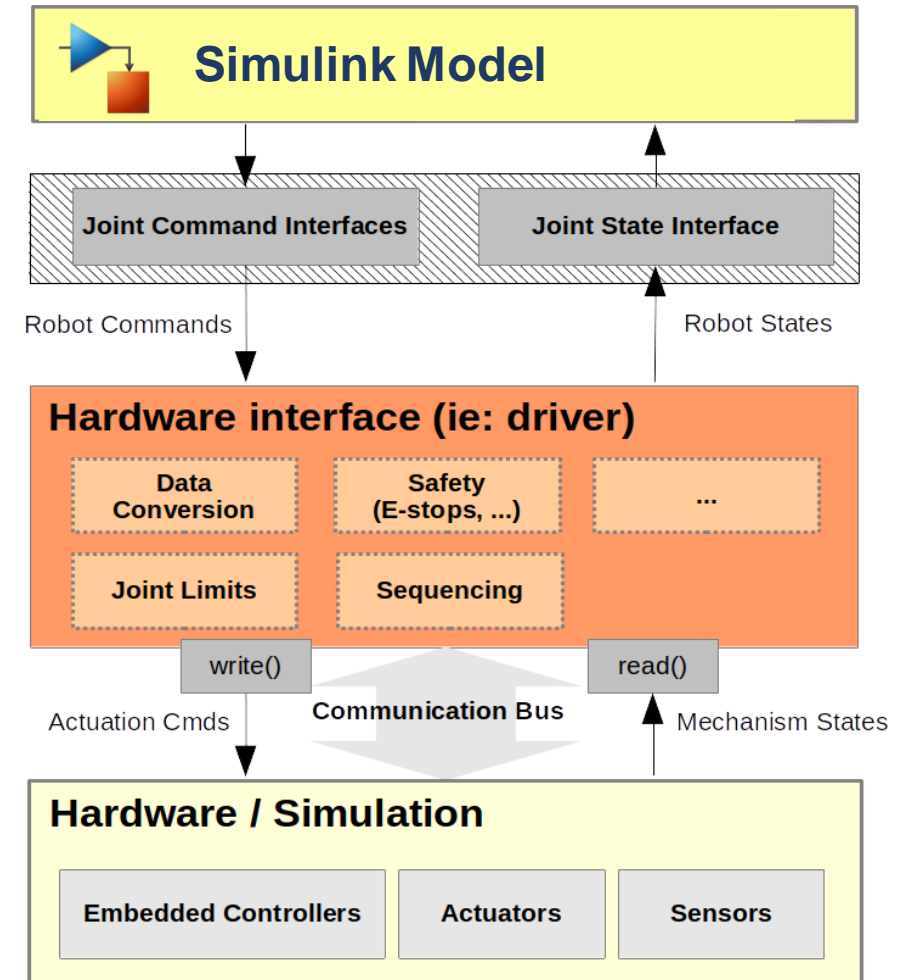


Connect MATLAB and Simulink with ROS

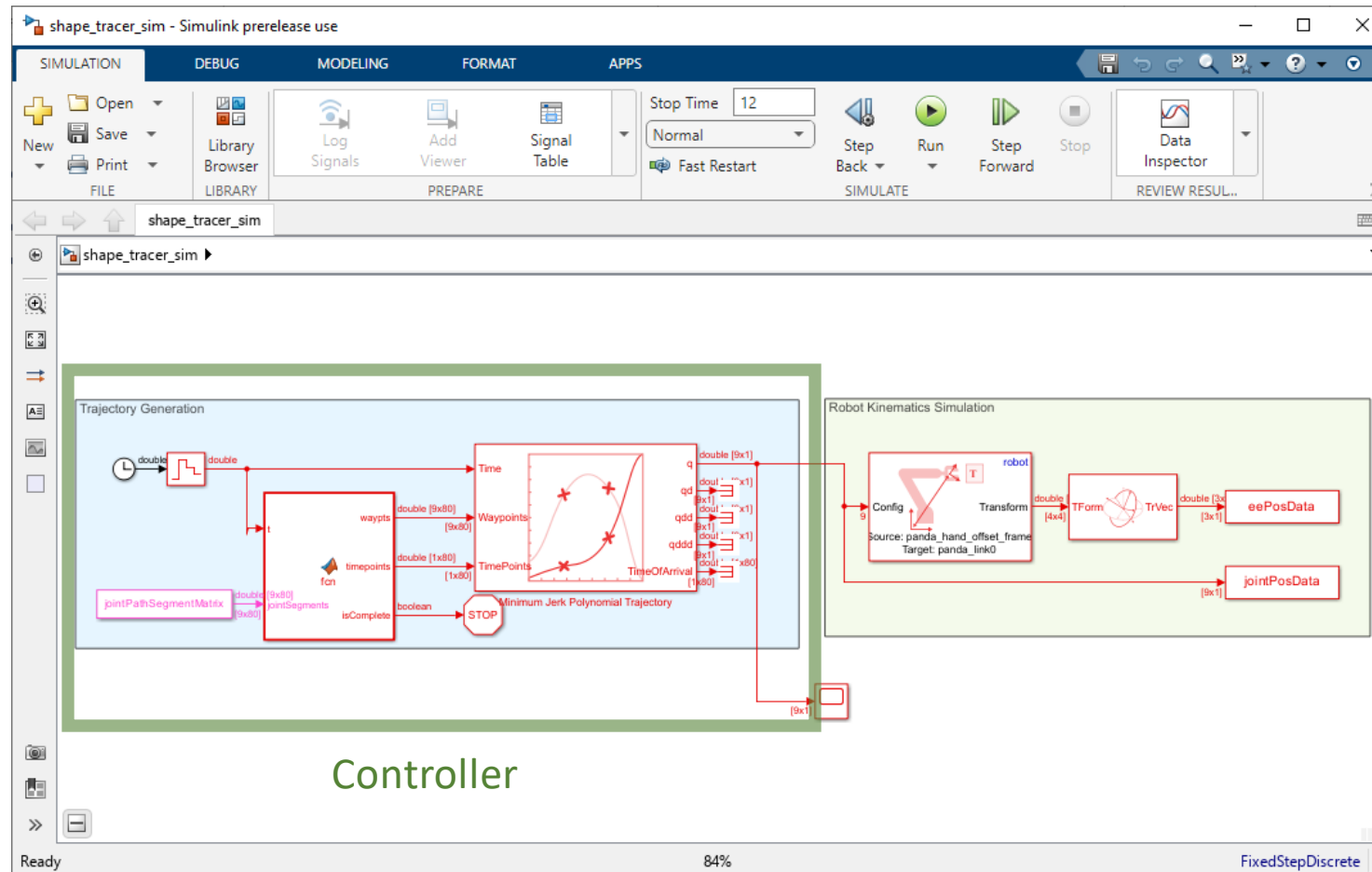


Recap (ros_control architecture)

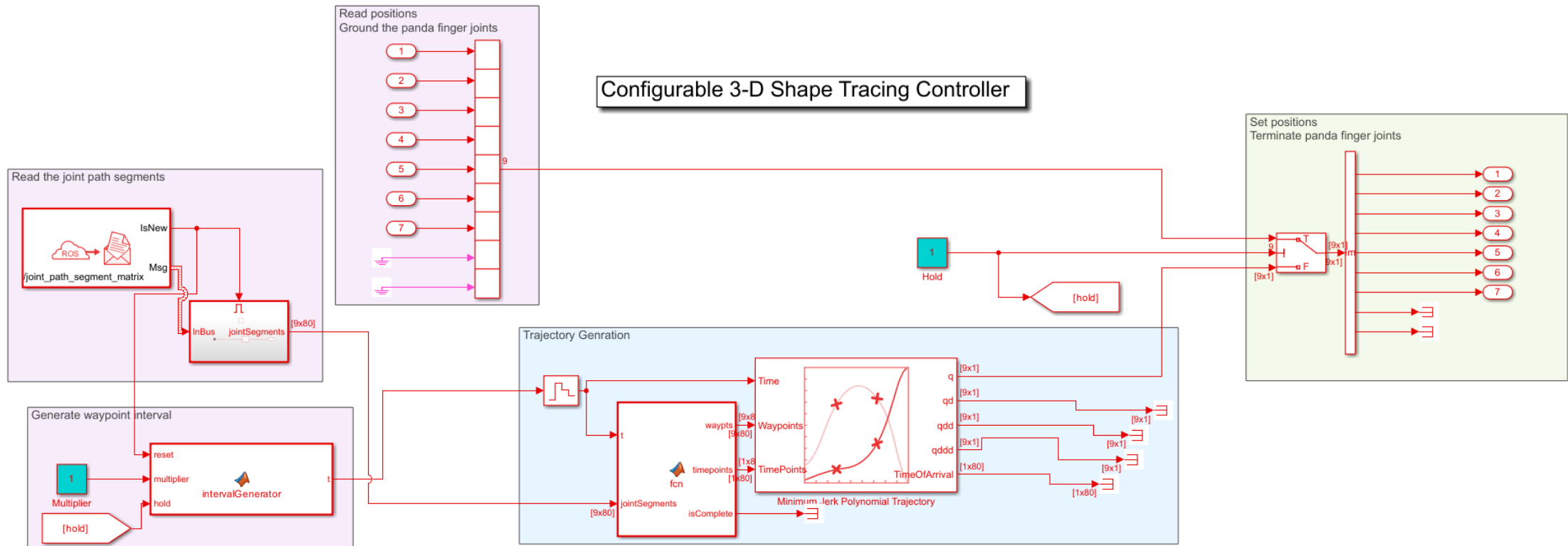
- Layered architecture
- Single process (multi-threaded)
- Determinism *within* node (execution)
- OEM provides up to the *interfaces layer*
- RobotHW transforms data:
 - From HW to ROS (ex: enc ticks → rad)
 - From ROS to HW (ex: rad → enc ticks)
- Combine RobotHW (OEM1, OEM2, ...)
- Controllers are user-facing



Example – 3D Shape Tracing

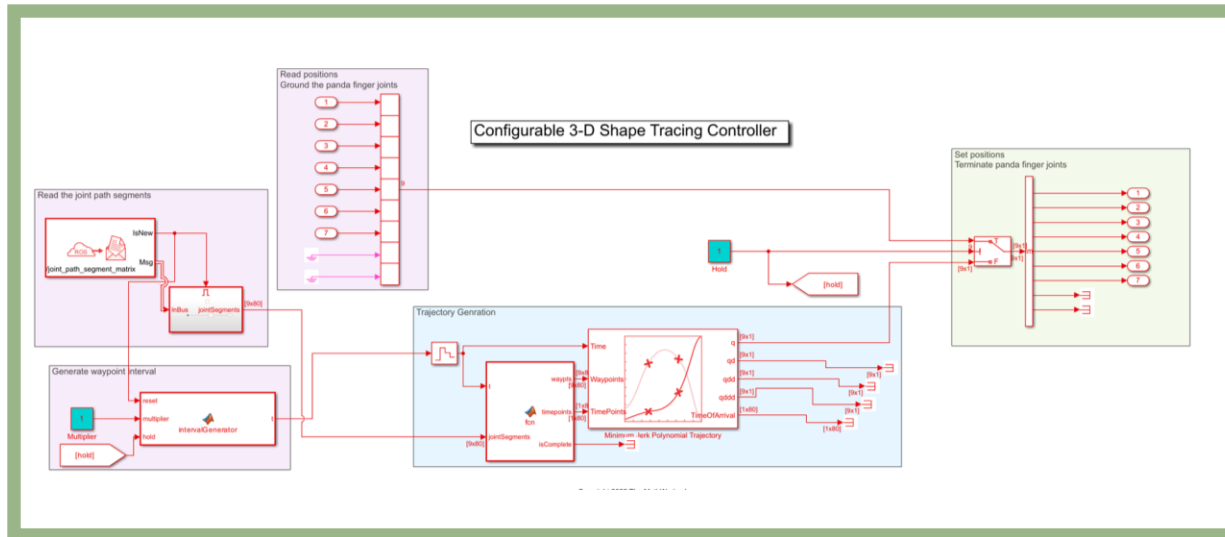


Example – Shape Tracing Controller Model

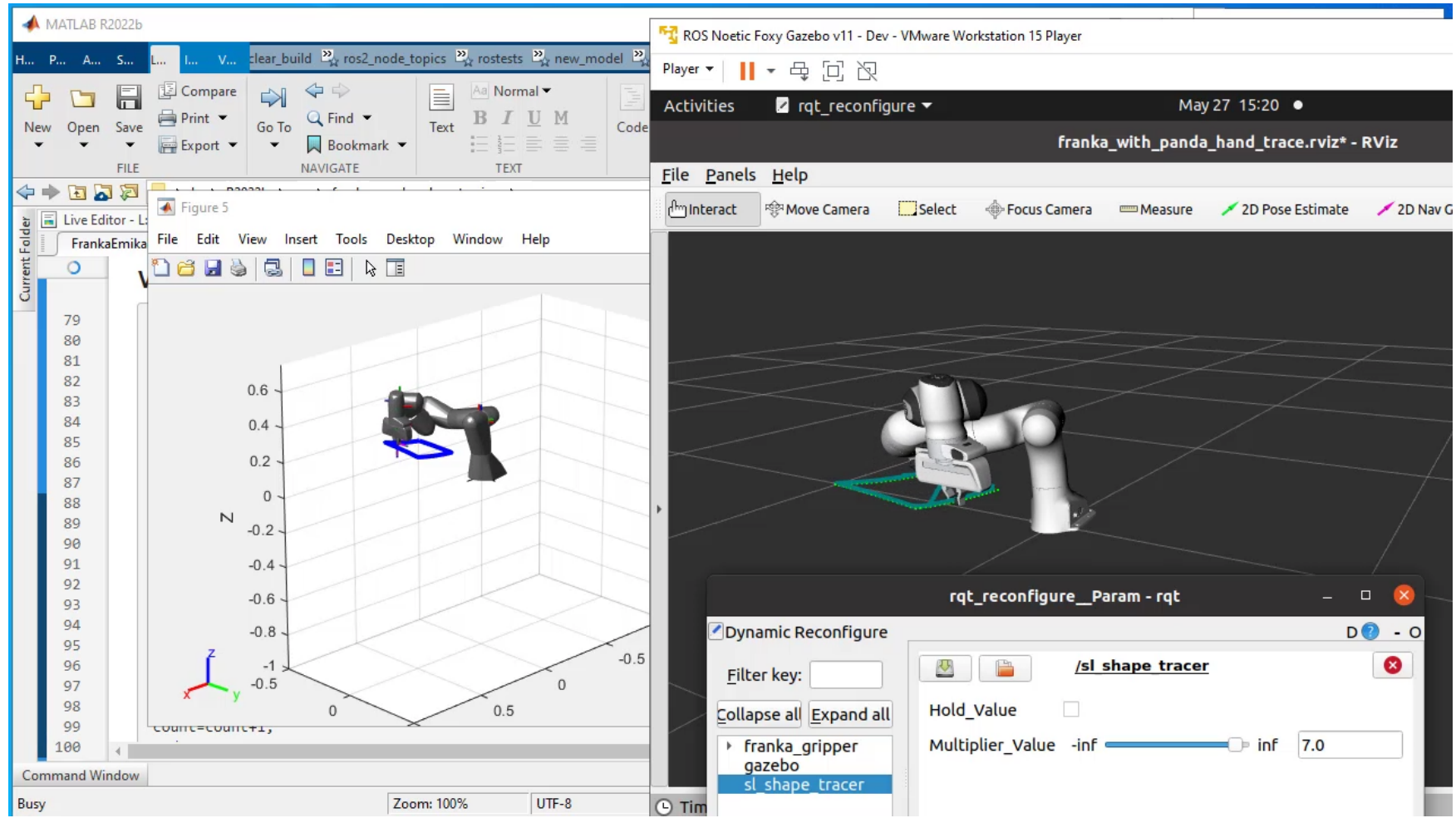


Copyright 2022 The MathWorks, Inc.

Example – 3D Shape Tracing



Video demo



Typical Applications using Shape Tracing

- Industrial Applications
 - Welding
 - Sanding
 - Painting
- Curved 3-D surfaces
 - Aircrafts
 - Ships



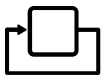
Agenda



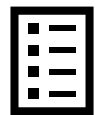
Introduction



Automated ROS code generation



Automated ros_control generation

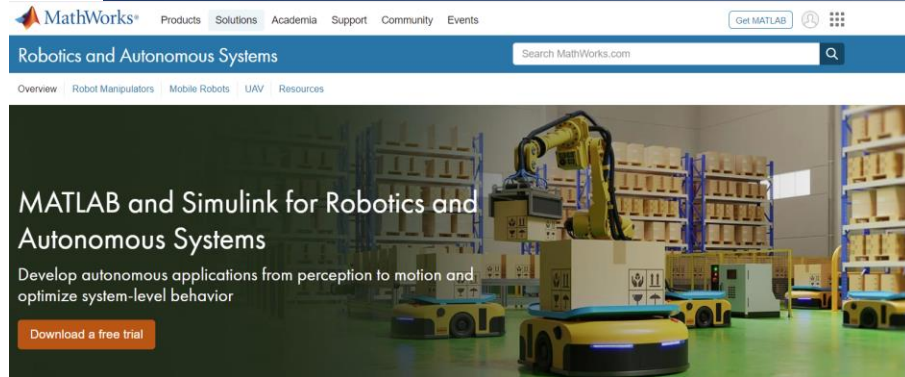


Summary

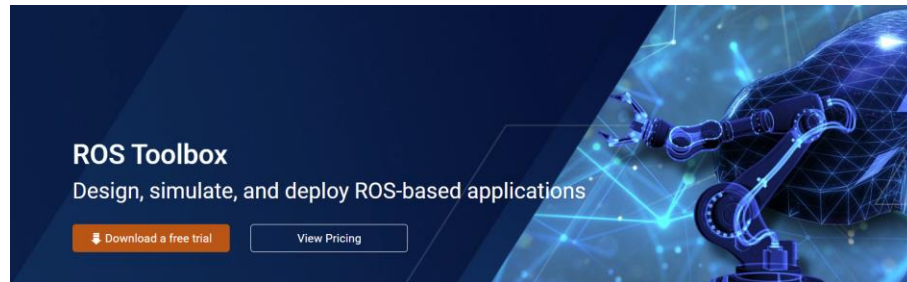
Key Takeaways

- Automated **C++ ROS code** and **ros_control** generation from MATLAB and Simulink
 - ▶ Incorporate ROS framework for implementation
 - ▶ Speed up the development process
 - ▶ Remove manual implementation errors
 - ▶ Go directly from algorithm prototyping to implementation
 - ▶ Easily incorporate Simulink controllers into ros_control framework
- **Call-To-Action:**
 - ▶ Try out the reference examples from [ROS Toolbox](#)
 - ▶ Reach out to us to work on real-world industrial applications

Learn More



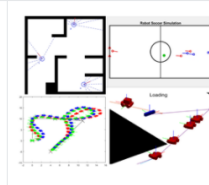
[Robotics Solutions Page](#)



[ROS Toolbox](#)

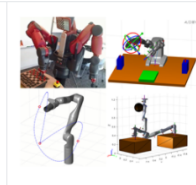
Ground Vehicles and Mobile Robotics

- Kinematic motion models for simulation
- Control and simulation of warehouse robots
- Programming of soccer robot behavior (Video)
- Simulation and programming of robot swarm (Video)
- Mapping, Localization and SLAM (See Section Below)
- Motion Planning and Path Planning (See Section Below)
- Mobile Robotics Simulation Toolbox (Video)
- Robotics Playground (Robotics Education - Video)



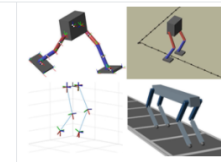
Manipulation

- Tools for rigid body tree dynamics and analysis
- Inverse Kinematics (Blog and GitHub Repo)
- Inverse kinematics with spatial constraints
- Interactive Inverse Kinematics
- Collision checking (Self-Collisions, Environment Collisions)
- Trajectory Generation (Blog, GitHub Repo)
- Safe trajectory planning (Impedance based control)
- Pick and place workflows (Using Gazebo)



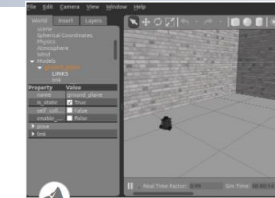
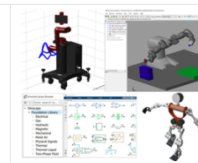
Legged Locomotion

- Modeling and simulation of walking robots (GitHub Repo)
- Pattern Generation for Walking Robots (Video)
- Linear Inverted Pendulum Model (LIPM) for humanoid walking (Video)
- Deep Reinforcement Learning for Walking Robots (Video)
- Modeling of quadruped robot running (Files)
- Quadruped Robot Locomotion Using DDPG Agent



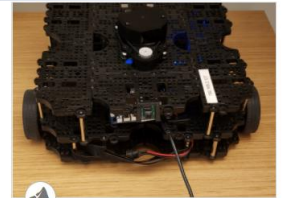
Robot Modeling

- Simscape Tools for Modeling and Simulation of Physical Systems
- Simulate Manipulator Actuators and Tune Control Parameters
- Algorithm Verification Using Robot Models
- Import Robots to MATLAB from URDF Files
- Import Robots from CAD and URDF Files



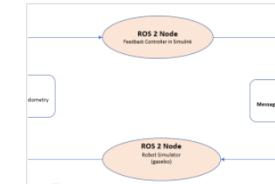
Test Robot Autonomy in Simulation

Explores MATLAB® control of the Gazebo® Simulator.



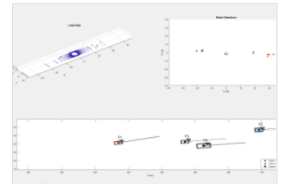
Get Started with a Real TurtleBot

Connect to a TurtleBot® using the MATLAB® ROS interface. You can use this interface to connect to a wide range of ROS-supported



Feedback Control of a ROS-Enabled Robot Over ROS 2

Use Simulink® to control a simulated robot running in a Gazebo® robot simulator over ROS 2 network.



Fusion of Radar and Lidar Data Using ROS

Perform track-level sensor fusion on recorded lidar sensor data for a driving scenario recorded on a rosbag. This example uses the same

MathWorks Robotics Solution Page

**Awesome-MATLAB-Robotics
GitHub Repo ([LINK](#))**

**ROS Examples
([LINK](#))**

Acknowledgement

We thank **Gijs van der Hoorn** for providing his guidance creating 'Automated ROS Control Plugin from Simulink'



Gijs van der Hoorn





YJ Lim

yjlim@mathworks.com



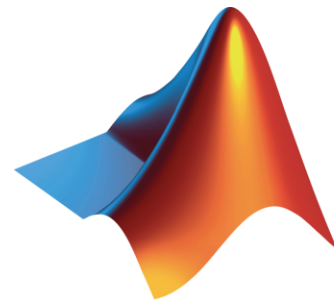
Abhijeet Gadkari

agadkari@mathworks.com



Murat Belge

mbelge@mathworks.com



MathWorks®

Accelerating the pace of engineering and science