Automated Code Generation of Simulink Models as ros_control Controllers

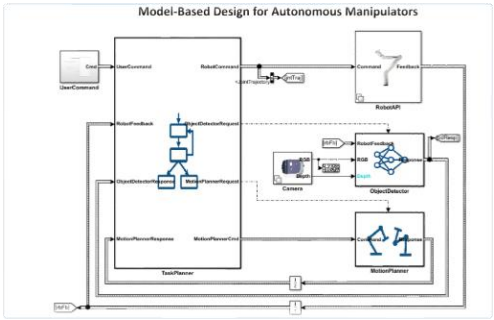
G.A. vd. Hoorn

Cognitive Robotics
Delft University of Technology
g.a.vanderhoorn@tudelft.nl

Murat Belge

Development Lead
MathWorks, Natick USA
mbelge@mathworks.com

Introduction



Code Generation

```
#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)
```

Motivation

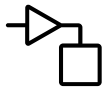
Idea: Design a controller in Simulink and use automated code generation to create a ros_control controller

- Many benefits
  - Model-based design
  - Virtual testing and validation in simulation environment
  - Automated C++ code generation from a graphical model of the controller eliminates hand-coding

- MathWorks and Delft University of Technology partnered to bring Simulink and ros_control together

Agenda

ros_control overview



Simulink C/C++ code generation

ros_control as a codegen target for Simulink models

Summary

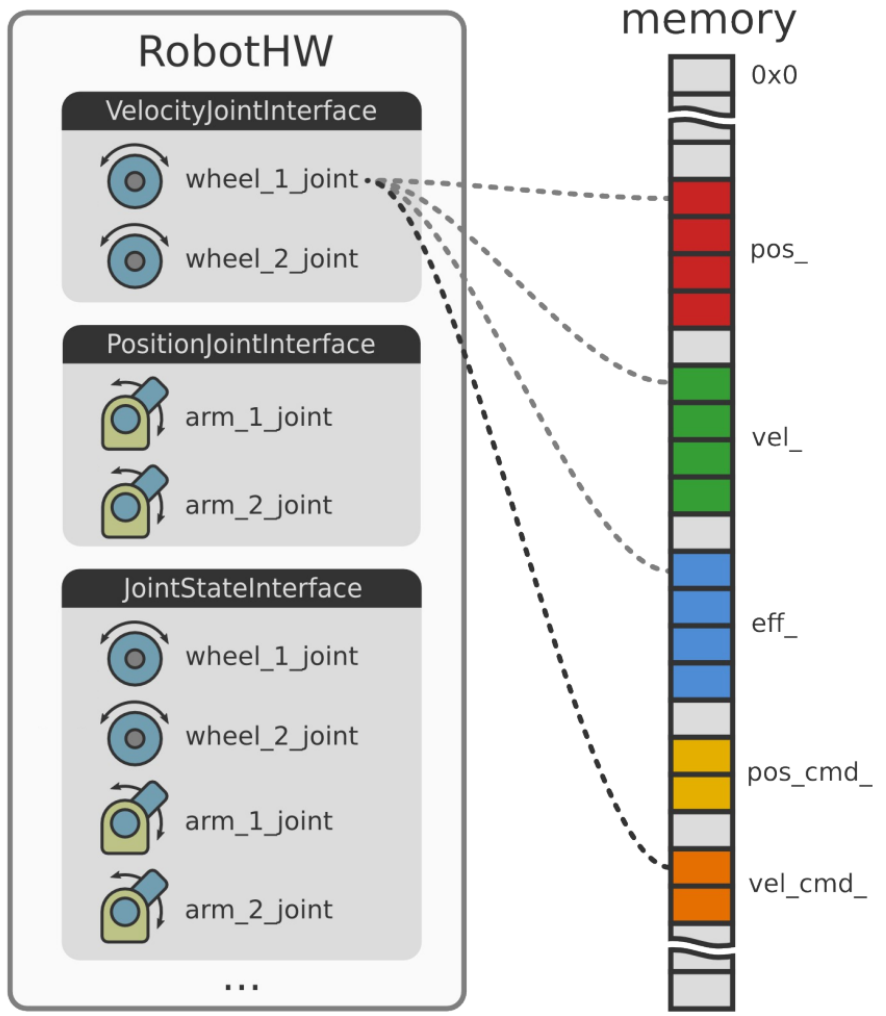
What is ros_control

- Generic set of tools for control of robots (mostly PID)
- Generalization of `pr2_mechanism` controllers (PR2)
- ROS 1 and ROS 2
- Used by enthusiasts and OEMs (Clearpath, PAL, UR, ABB, Franka Emika, ..)
- Two main parts:
  - Abstracted hardware access (including simulators)
  - Controllers with standard ROS interfaces (`FollowJointTrajectory`, etc)
- Decoupled execution policies
- Hard real-time compatible

Main goal: promote reuse of control code

ros_control: abstract hardware

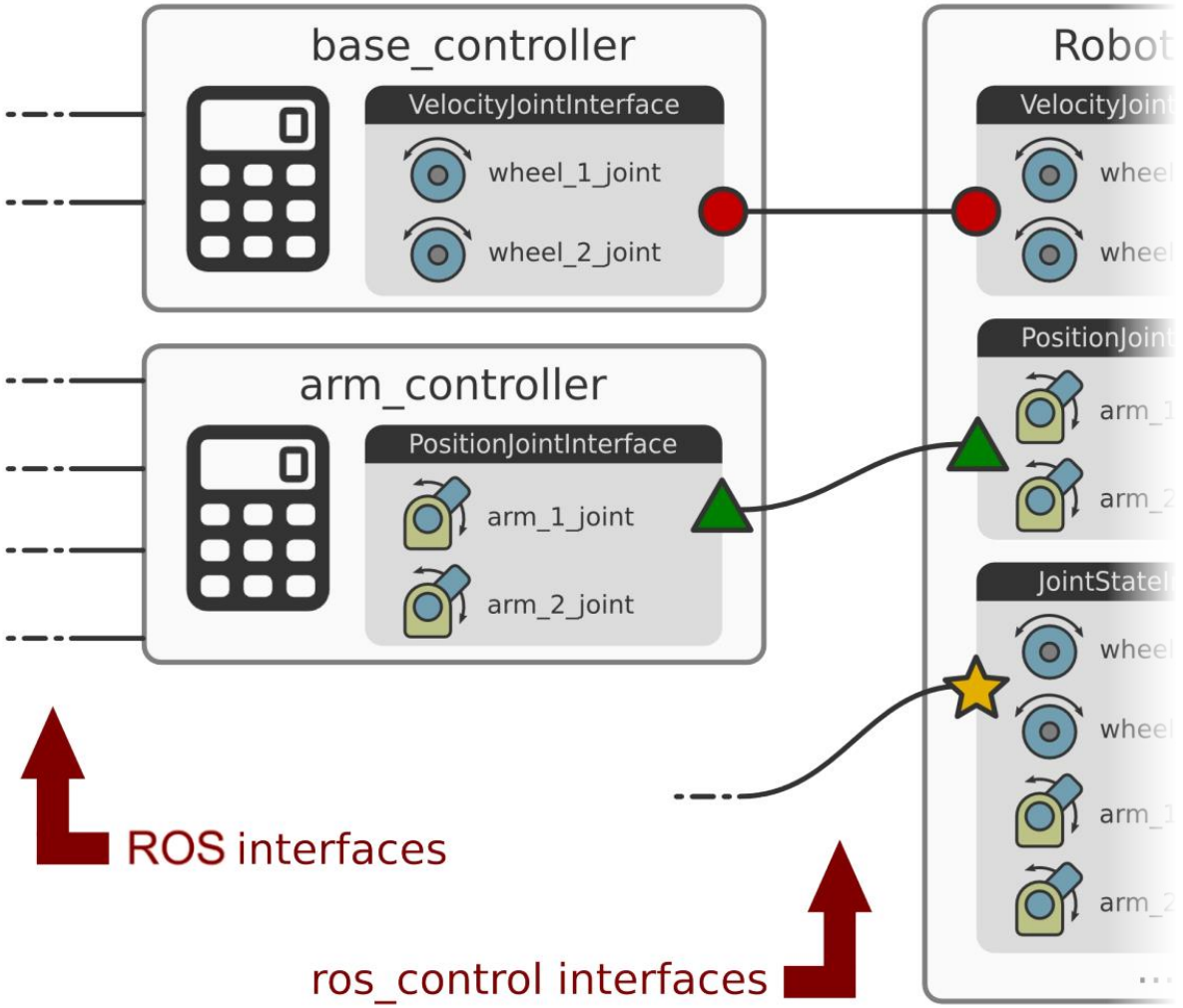
- No direct access to hardware
- `RobotHW` class wraps:
  - Registers / memory
  - OEM API / SDK
  - Fieldbus
  - TCP or UDP
  - ROS topics / services / actions
- Exposes abstract resources (*handles*):
  - OotB: position, velocity, effort, IMU, F/T
  - Plugin based (custom resources)
- Conflict detection (sometimes resolution)



Original diagram: Adolfo Rodríguez Tsouroukdissian

ros_control: controllers

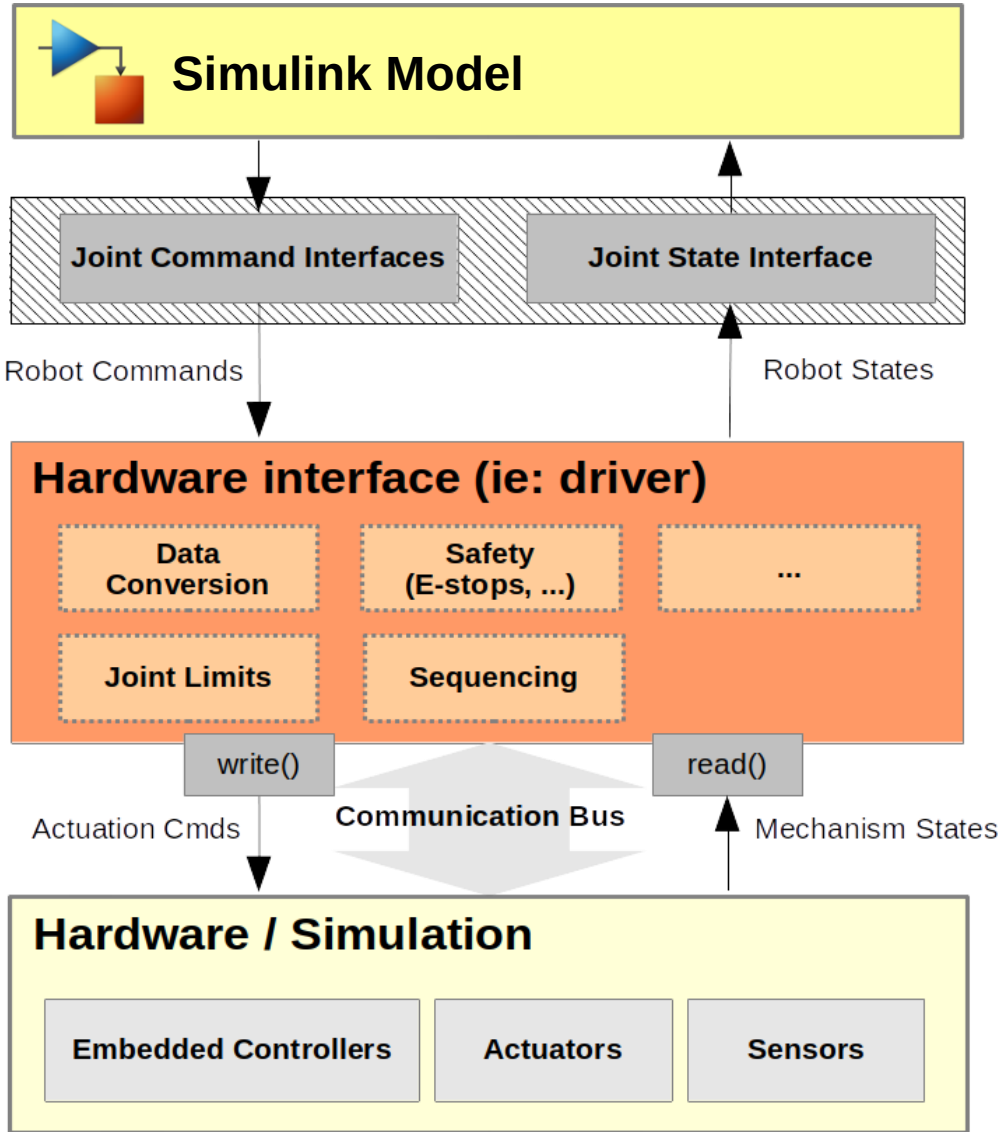
- Claim *resources*, not hardware
- Expose ROS interfaces
- Runtime load, init, (re)start, stop
- Runtime reconfigurable & switchable
- OotB: position, velocity, effort, trajectory execution
- Plugin based: custom controllers:
  - Whole-body (NASA Valkyrie, PAL)
  - Affordance
  - Cartesian F/T
  - Visual Servoing
  - Etc



Original diagram: Adolfo Rodríguez Tsouroukdissian

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



Original diagram: Dave Coleman

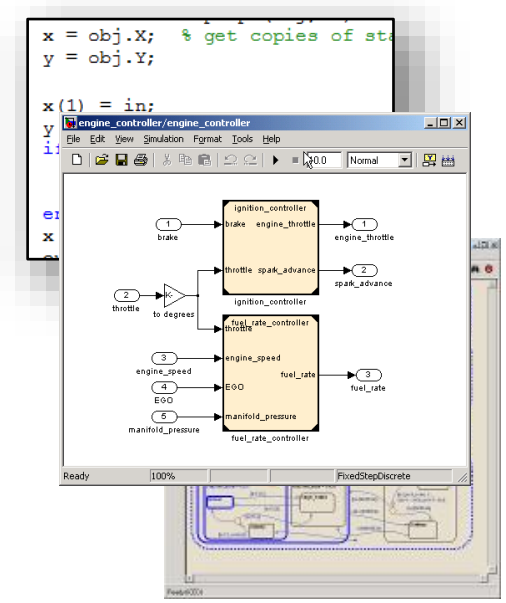
Agenda

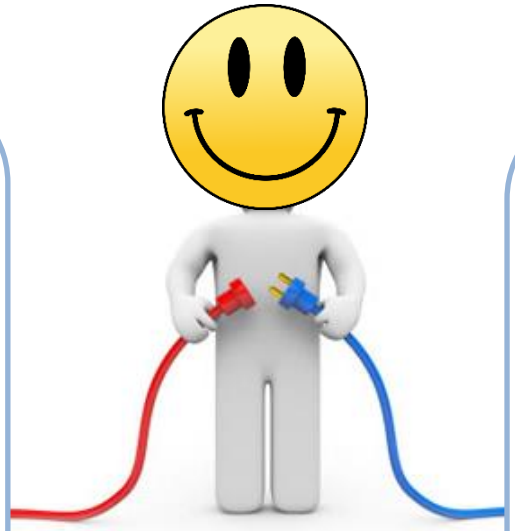
- ros_control Recap
- Simulink C/C++ Code Generation
- ros_control as a codegen target for Simulink models
- Summary

Simulink Code Generation

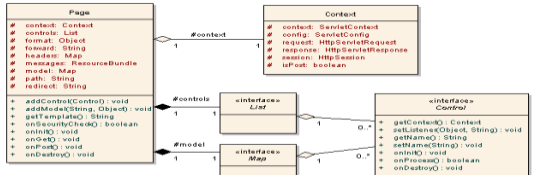
Generate *efficient* C++ code that *plugs easily* into your C++ architectures

Simulink components





C++ Software Architecture



DDS ROS AUTOSAR Qt OpenCV Linux

Roles and Goals of Code Generation

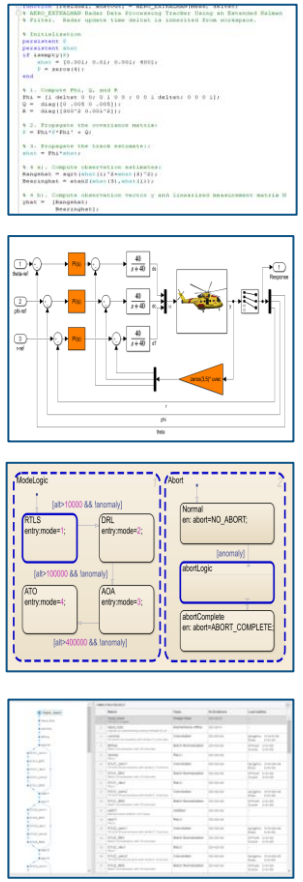
Roles

- Acceleration
- Prototyping
- Production
- Verification

Goals

- Correct
- Efficient
- Customizable
- Certifiable
- Readable
- Scalable
- …

User design in high level / domain specific languages



Code Generator

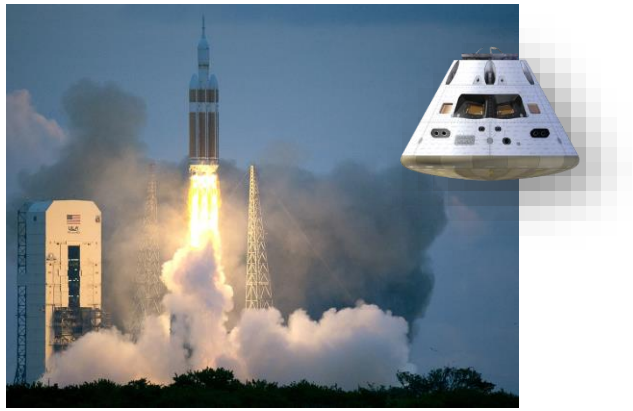
General purpose programming languages

- **C code**
- **C++ code**
- **HDL code**
- **PLC code**
- **GPU code**
- ⋮

Success Stories

NASA Orion Program

“All of our GN&C flight software is automatically coded into C++ for deployment onto the vehicle.”



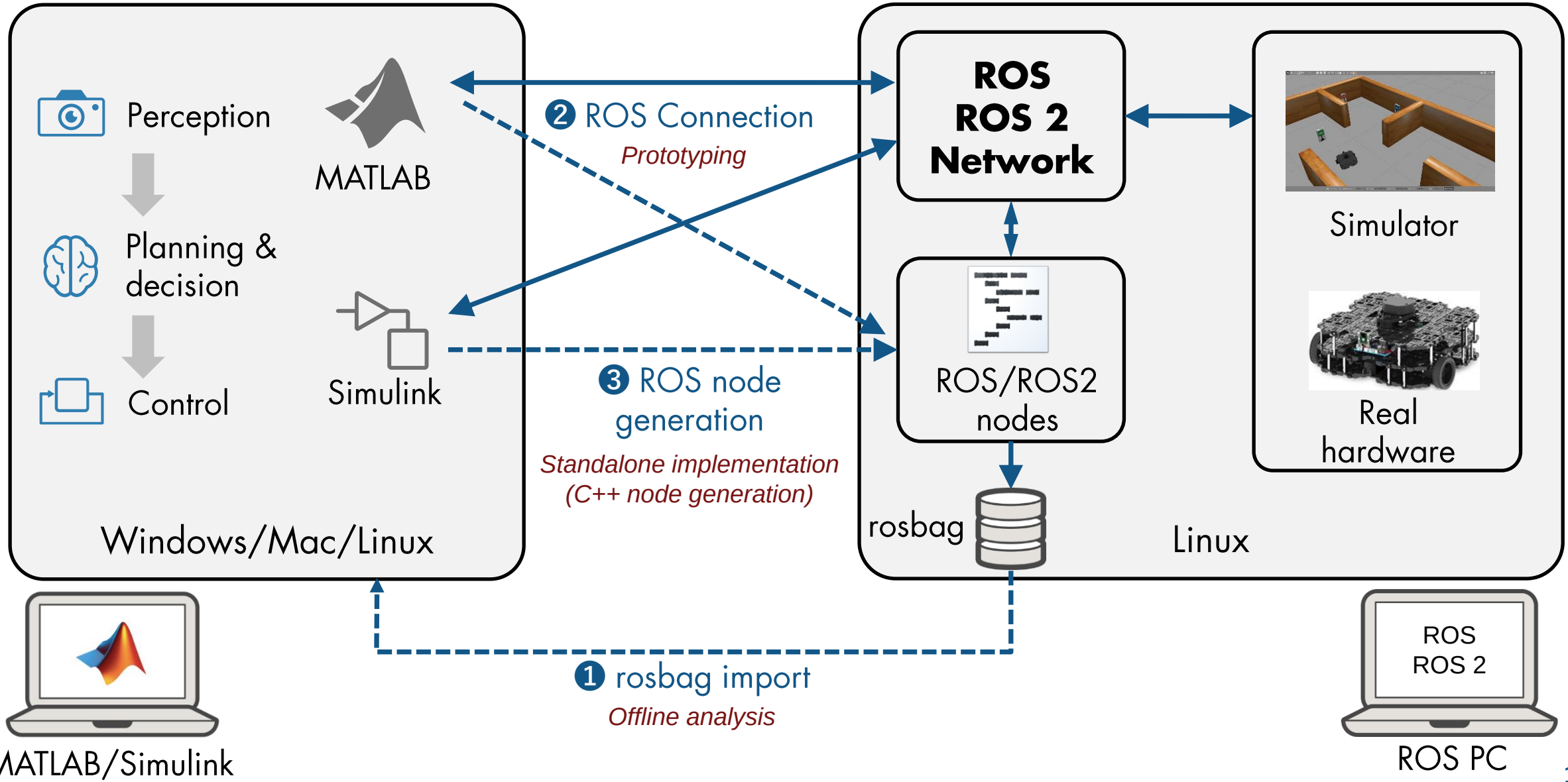
Voyage

“Simulink + ROS allowed us to deploy a Level 3 Autonomous vehicle in less than 3 months”

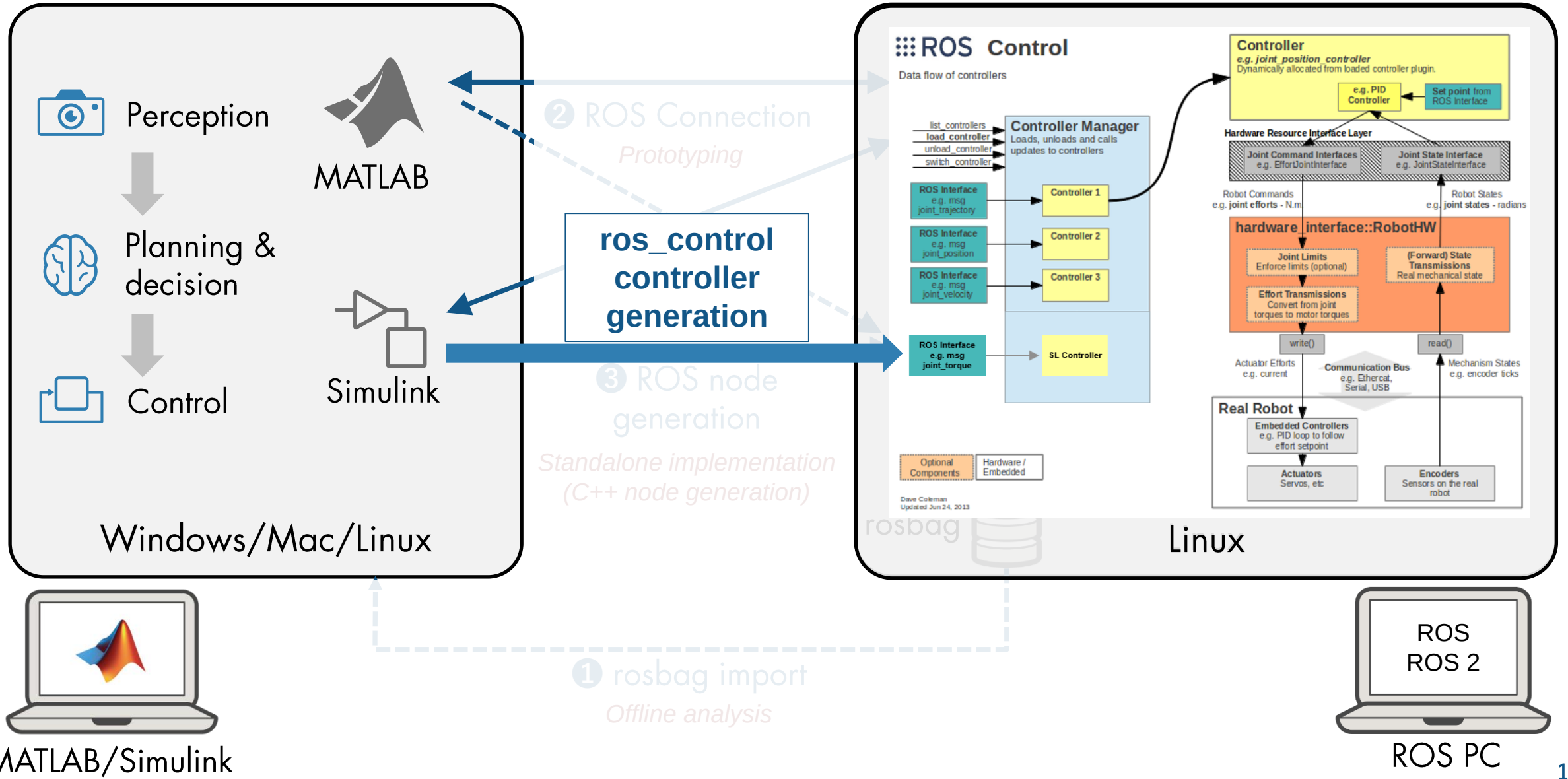
Agenda

- ros_control Recap
- Simulink C/C++ Code Generation
- ros_control as a codegen target for Simulink models
- Summary

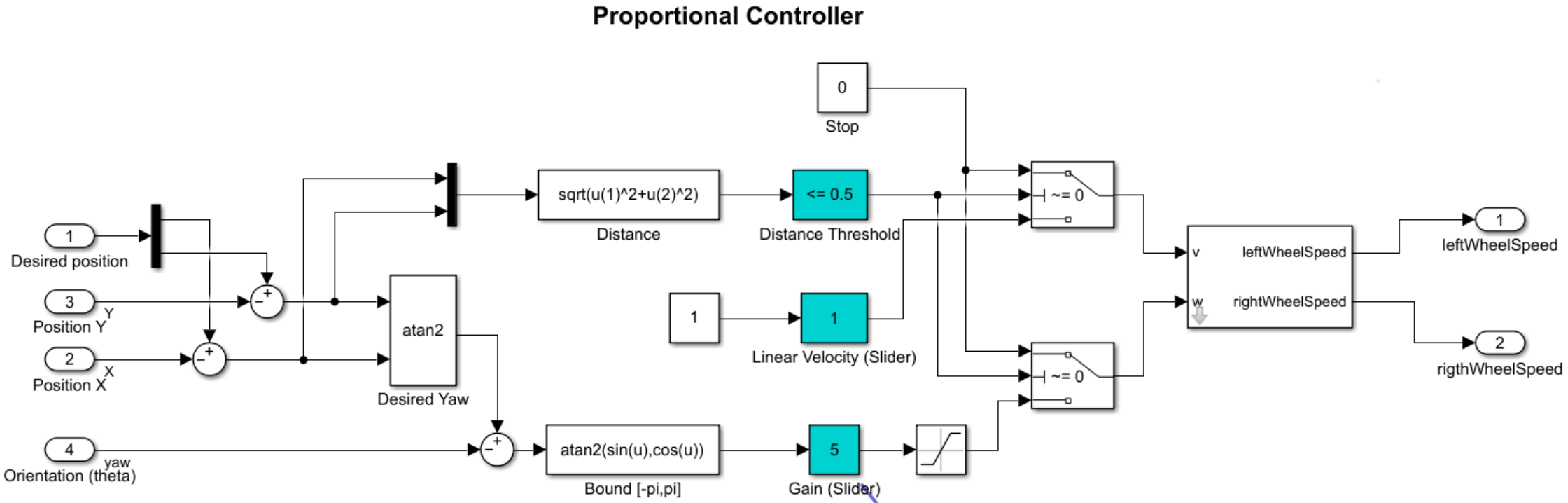
Connect MATLAB and Simulink with ROS



Connect MATLAB and Simulink with ROS

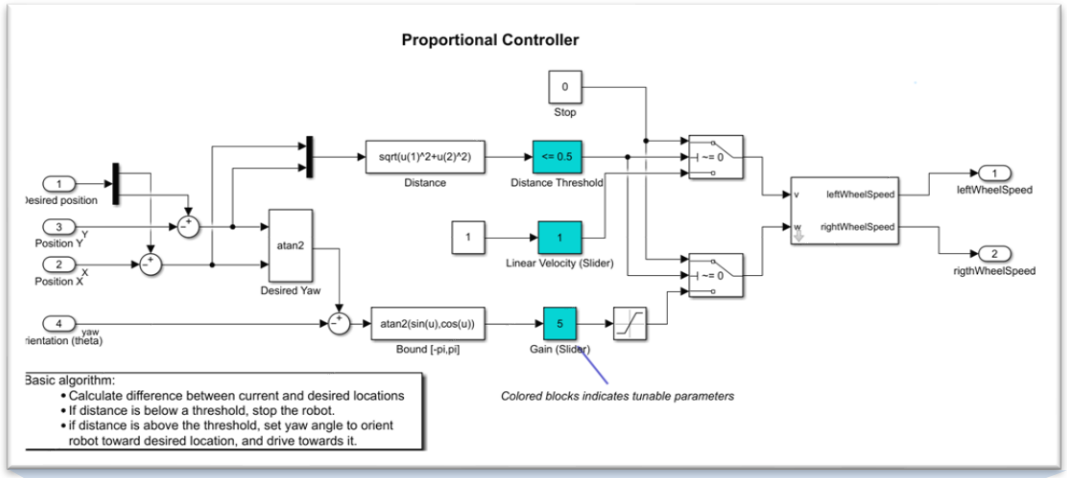


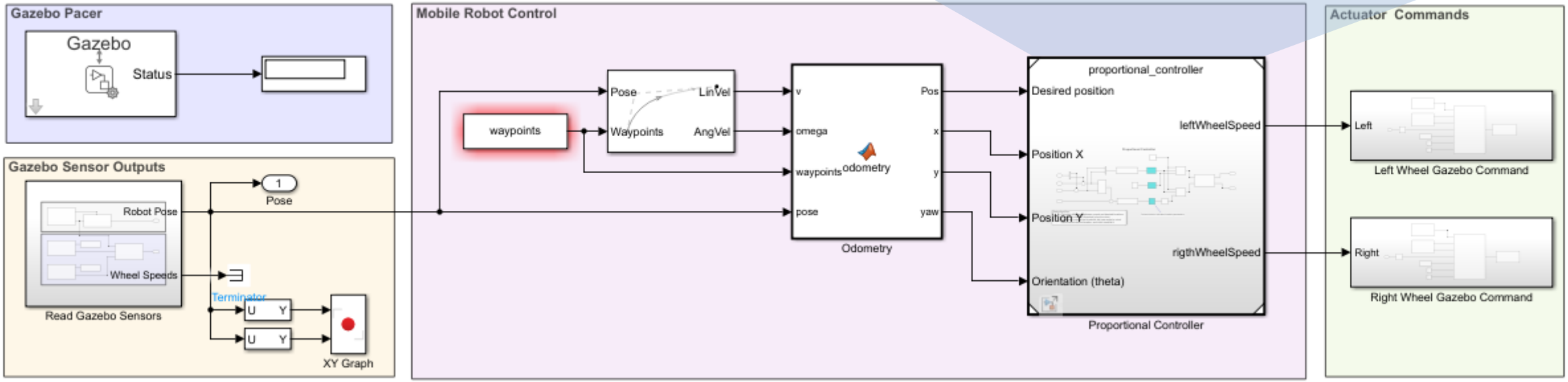
Proof-of-Concept: Differential Drive Proportional Controller



Proof-of-Concept: Differential Drive Proportional Controller

- Test & validate controller in Simulink environment

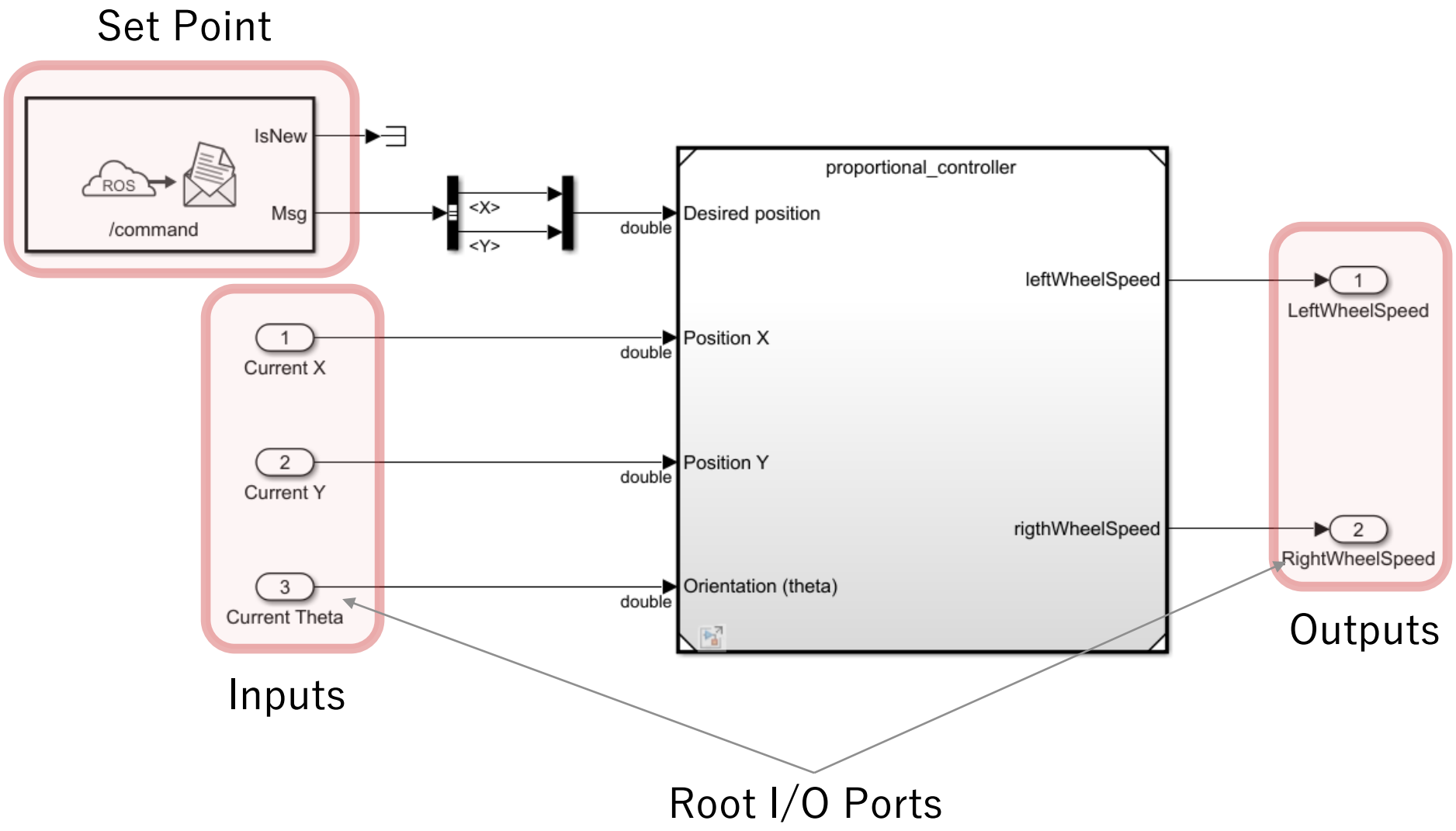




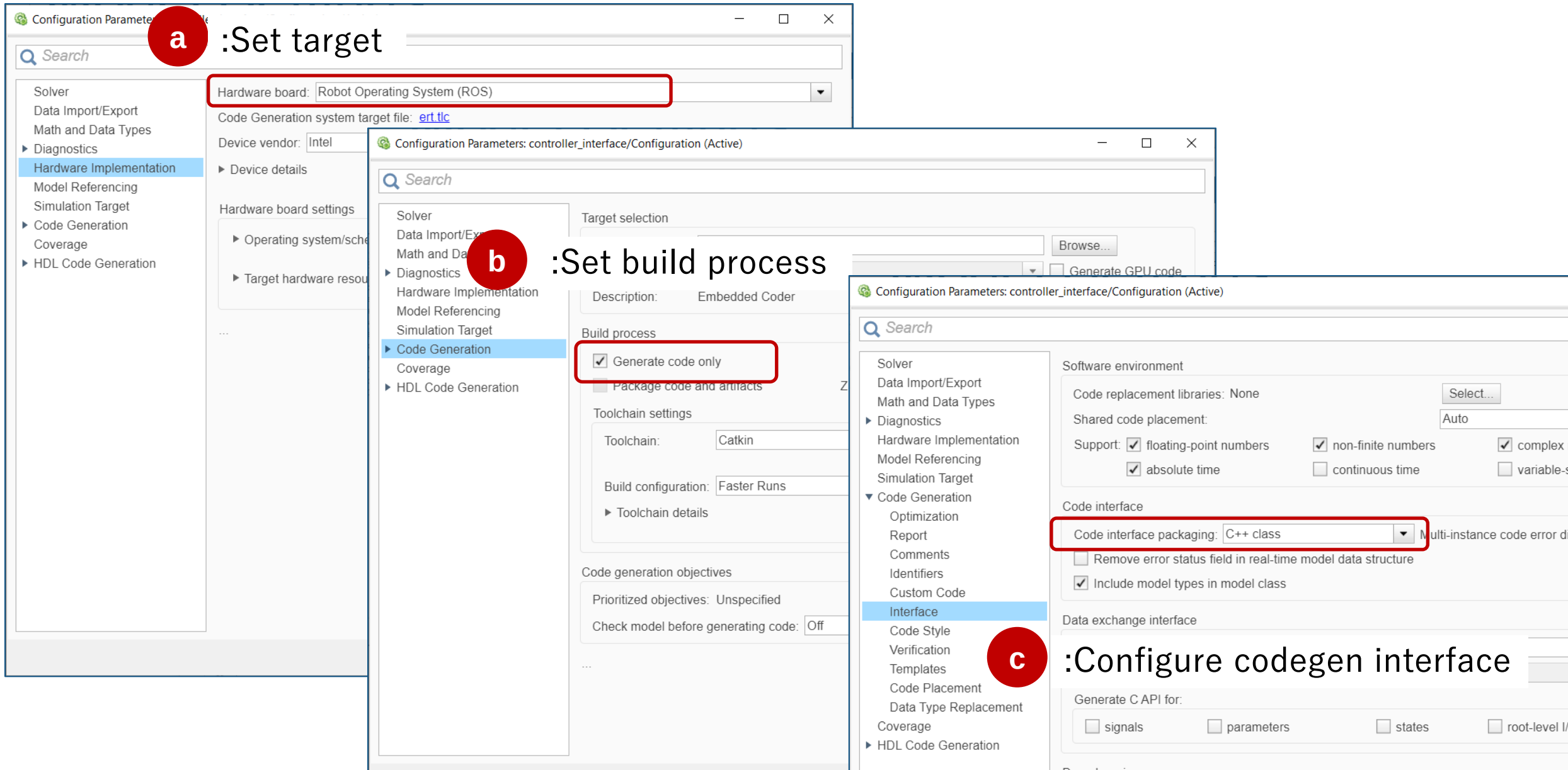
Steps to Deploy Simulink Controller as a **ros_control** Controller

1. Create interface model
2. Adjust model settings
3. Generate code
4. Integrate generated code into ros_control framework

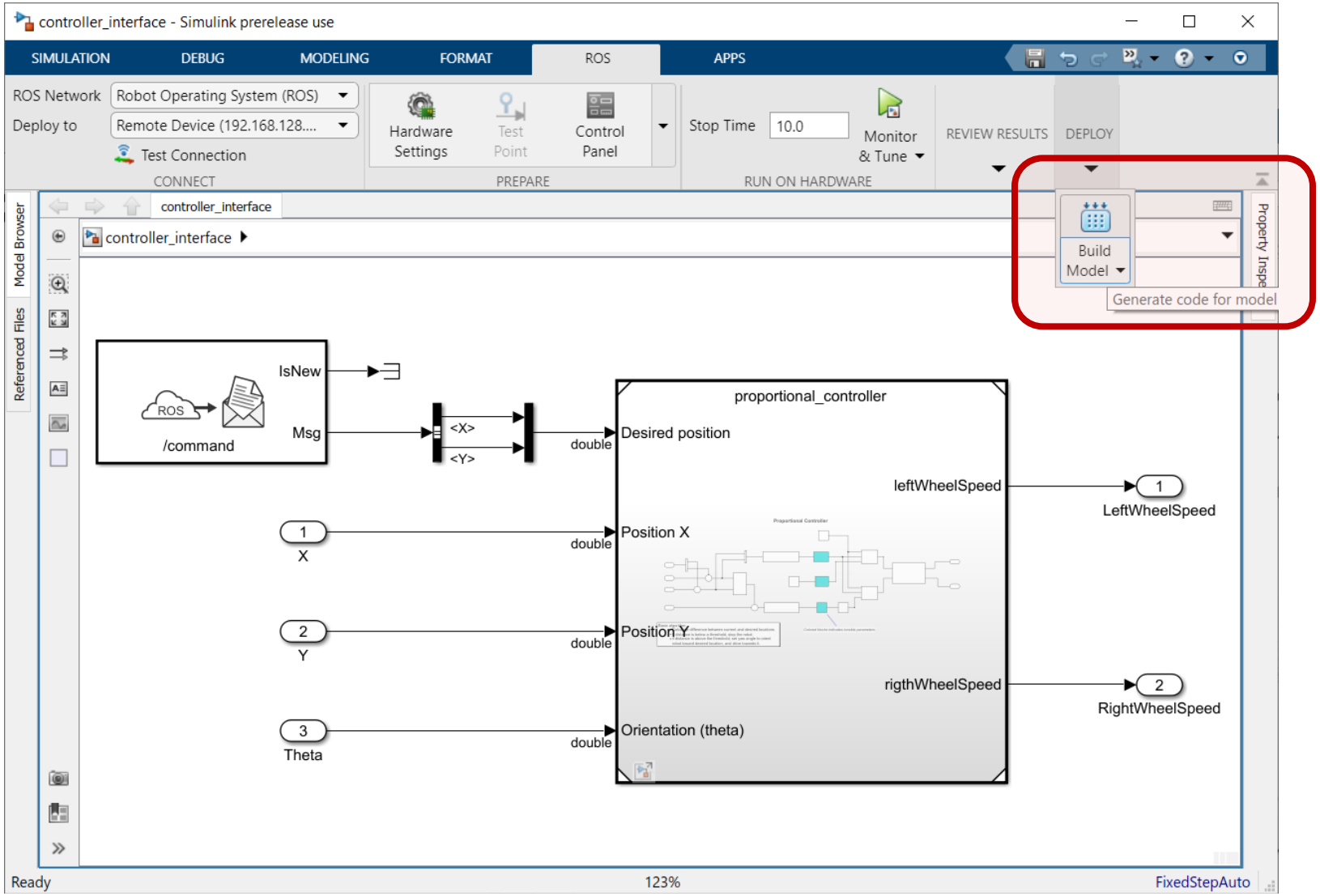
❶ Create Interface Model



❷ Adjust Model Settings



❸ Generate Code



❹ Integrate Generated Code into **ros_control** Framework

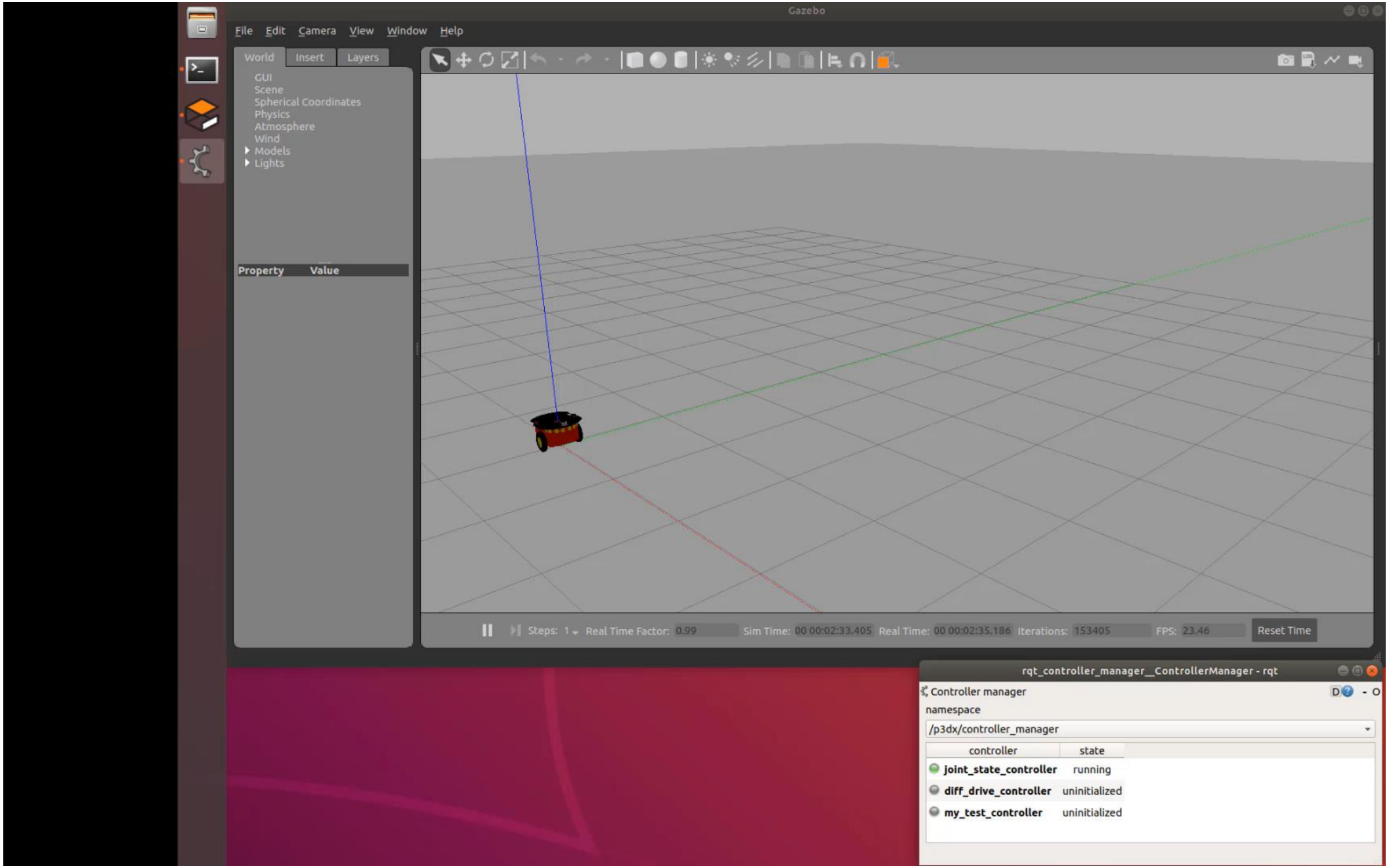
```
namespace SL {

class modelClass {

 public:
  void initialize();
  void step();
  const ExtY &getExternalOutputs() const;
  void terminate();

 protected:
  // protected data

 private:
  // private member methods
  // private data
};
}
```

Simulink Model Class

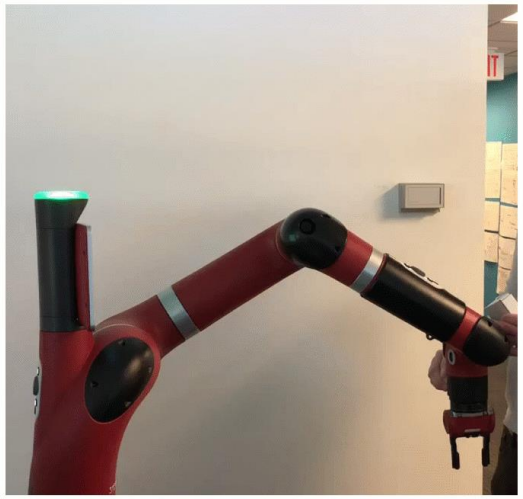
```
namespace ros {

class controller{

 public:
  void init(…);
  void update(…);
  void starting(…);
  void stop();
  ~controller();

 private:
  SL::modelClass slModel;
};
}
```

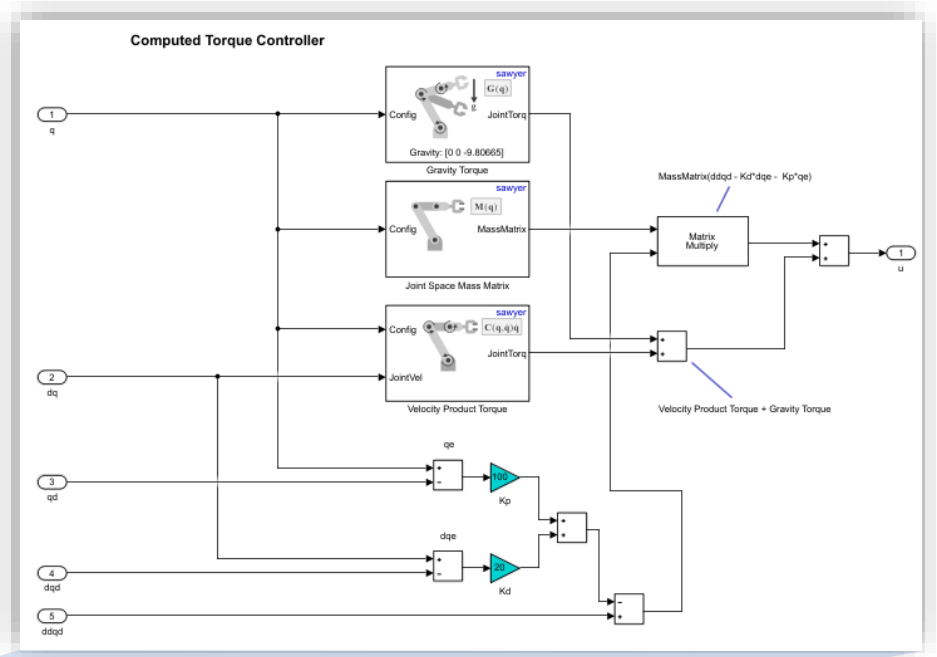
ros_control Class

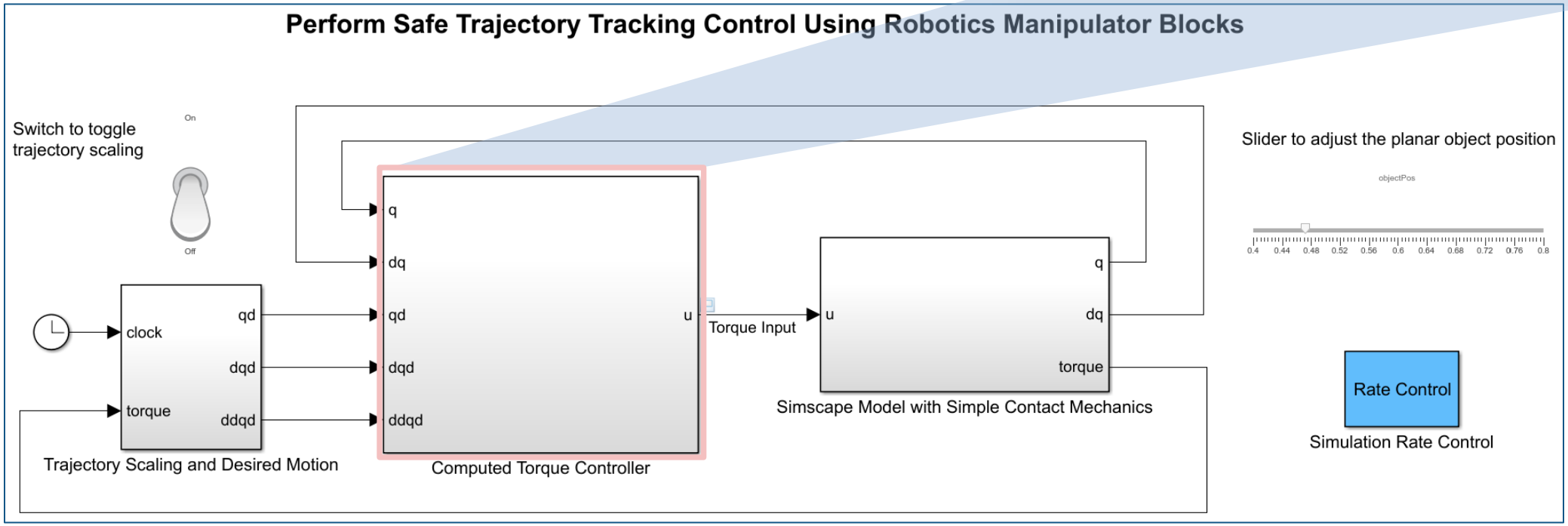
Demo: Differential Drive Robot Control in Gazebo



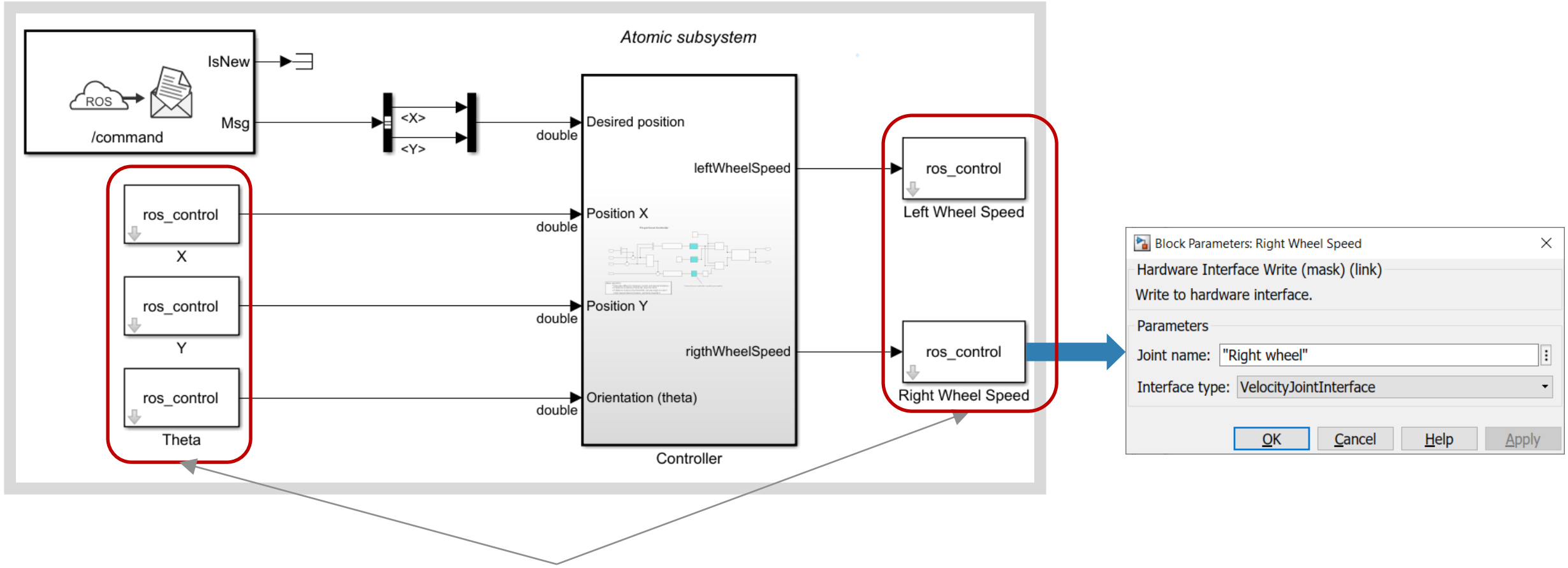
Future: Extend to Manipulator Controllers







Future: Generate ros_control Package **Automatically**



Robot Hardware Interface Blocks

Key Takeaways

- Automated code generation from Simulink to create ros_control controllers
  - ► Simulink controllers can easily be incorporated into ros_control framework for real-time controller implementation
- Same automated code generation infrastructure can be used for **ros2_control**
- Call-To-Action:
  - ► **Reach out to us to work on real-world industrial applications**

% Thank you!

Gijs van der Hoorn

Robot Dynamics, Cognitive Robotics

Delft University of Technology

the Netherlands

GitHub : **gavanderhoorn**
Email : **g.a.vanderhoorn@tudelft.nl**

Murat Belge

Development Lead of ROS Toolbox

MathWorks Inc.

Natick MA, USA

Email : **mbelge@mathworks.com**