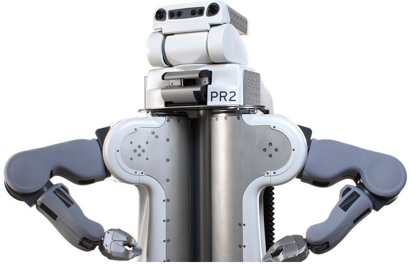
# Evolving ROS for Safety Critical Systems

Embedded Linux Conference 2022

Tully Foote

June 23rd 2022

# Who am I?

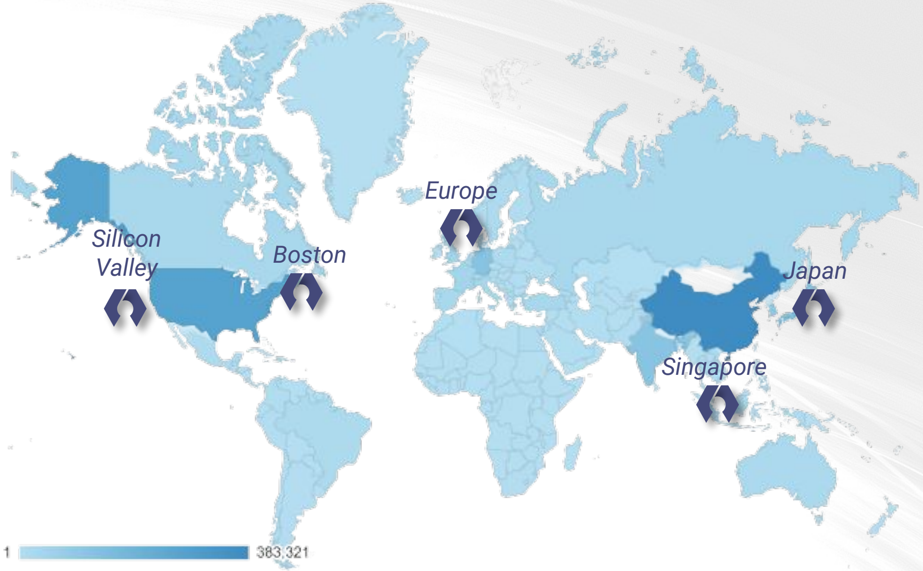


Willow Garage

open robotics

# Who We Are
## About Us

- Open Robotics est. in 2012
  - Originally Open Source Robotics Foundation
- Privately-held
- 50 employees
- Global team of engineers and scientists
- Founders, key contributors, and curators of world's most widely-used robot software



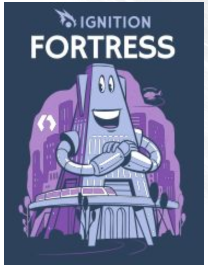
# Who We Are

## ROS and Ignition

ROS

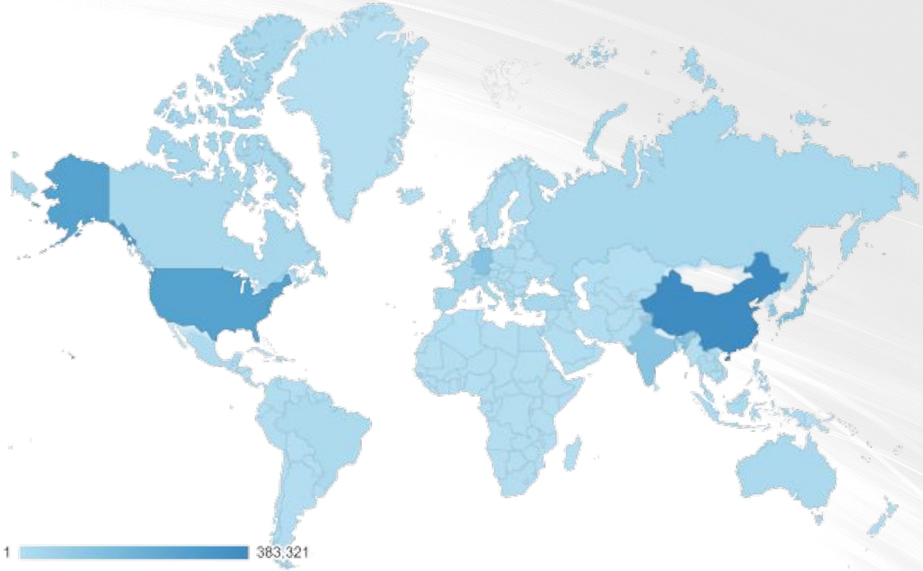
- World's most widely-used open-source SDK for robotics
- ROS is to robotics as Android is to mobility

- Open-source robot simulation software
- Ignition is to robotics what AutoCAD is to architecture



# Who We Are
## Community

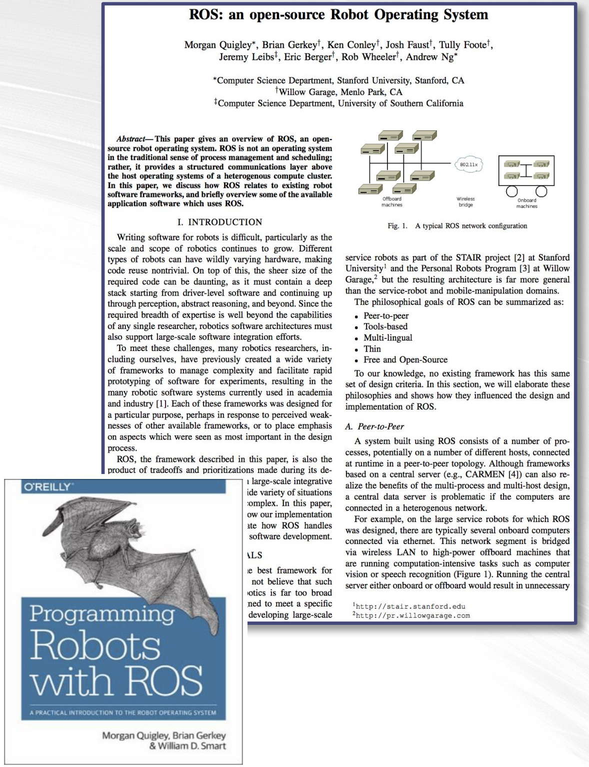
- Users
  - 178K/month*
  - 1.49M/year
    - 35% YoY increase
- Page Views
  - 24.48M/year
- Global Impact
  - U.S. constitutes 19% of users
- Translations
  - Partial translations into 14 languages



*2021 statistics

# Who We Are
## History of ROS

- First commit to ROS on SourceForge
  - November 7, 2007
- ROS: An Open-Source Operating System
  - Presented at ICRA May 17, 2009
- 2012: Debut of ROSCon
- 2013: ROS_Answers surpasses 10,000 questions
- 2015: DARPA Robotics Challenge
  - 18 teams using ROS; 14 using Gazebo
- 2015: Publication of *Programming Robots With ROS*
- 2022: ROS 2: Design, architecture, and uses in the wild
  - Science Robotics Vol 7 Issue 66



### ROS: an open-source Robot Operating System

Morgan Quigley*, Brian Gerkey†, Ken Conley†, Josh Faust†, Tully Foote†,
Jeremy Leibs‡, Eric Berger†, Rob Wheeler†, Andrew Ng*

*Computer Science Department, Stanford University, Stanford, CA
†Willow Garage, Menlo Park, CA
‡Computer Science Department, University of Southern California

***Abstract*— This paper gives an overview of ROS, an open-source robot operating system. ROS is not an operating system in the traditional sense of process management and scheduling; rather, it provides a structured communications layer above the host operating systems of a heterogenous compute cluster. In this paper, we discuss how ROS relates to existing robot software frameworks, and briefly overview some of the available application software which uses ROS.**

Fig. 1. A typical ROS network configuration

#### I. INTRODUCTION

Writing software for robots is difficult, particularly as the scale and scope of robotics continues to grow. Different types of robots can have wildly varying hardware, making code reuse nontrivial. On top of this, the sheer size of the required code can be daunting, as it must contain a deep stack starting from driver-level software and continuing up through perception, abstract reasoning, and beyond. Since the required breadth of expertise is well beyond the capabilities of any single researcher, robotics software architectures must also support large-scale software integration efforts.

To meet these challenges, many robotics researchers, including ourselves, have previously created a wide variety of frameworks to manage complexity and facilitate rapid prototyping of software for experiments, resulting in the many robotic software systems currently used in academia and industry [1]. Each of these frameworks was designed for a particular purpose, perhaps in response to perceived weaknesses of other available frameworks, or to place emphasis on aspects which were seen as most important in the design process.

ROS, the framework described in this paper, is also the product of tradeoffs and prioritizations made during its de-
ı large-scale integrative
ide variety of situations
:omplex. In this paper,
ow our implementation
ıte how ROS handles
software development.

ALS

ıe best framework for
not believe that such
ıotics is far too broad
ned to meet a specific
developing large-scale

service robots as part of the STAIR project [2] at Stanford University[1] and the Personal Robots Program [3] at Willow Garage,[2] but the resulting architecture is far more general than the service-robot and mobile-manipulation domains.

The philosophical goals of ROS can be summarized as:

- Peer-to-peer
- Tools-based
- Multi-lingual
- Thin
- Free and Open-Source

To our knowledge, no existing framework has this same set of design criteria. In this section, we will elaborate these philosophies and shows how they influenced the design and implementation of ROS.

#### A. *Peer-to-Peer*

A system built using ROS consists of a number of processes, potentially on a number of different hosts, connected at runtime in a peer-to-peer topology. Although frameworks based on a central server (e.g., CARMEN [4]) can also realize the benefits of the multi-process and multi-host design, a central data server is problematic if the computers are connected in a heterogenous network.

For example, on the large service robots for which ROS was designed, there are typically several onboard computers connected via ethernet. This network segment is bridged via wireless LAN to high-power offboard machines that are running computation-intensive tasks such as computer vision or speech recognition (Figure 1). Running the central server either onboard or offboard would result in unnecessary

[1] http://stair.stanford.edu
[2] http://pr.willowgarage.com

O'REILLY

Programming Robots with ROS

A PRACTICAL INTRODUCTION TO THE ROBOT OPERATING SYSTEM

Morgan Quigley, Brian Gerkey & William D. Smart

# What We Do
## Open Source

- Drivers
  - Robotics industry booming
    - Three-fold increase in professional service robots from 2019-2023
  - Robotics applications are exploding
    - Service robots set to overtake industrial robots
  - Demand exceeds supply for robotics expertise
  - Open-source robotics software is leading the industry
- Benefits
  - Focus your efforts on value-add and differentiation vs. basic robotics
  - Faster time to market
    - *18 months* to build Simbe’s Tally, rather than *22 years*
  - Better products with lower maintenance and support
  - Participation in broad and growing ecosystem

# What We Do
## ROS

- Core Services for Robotics
  - Message passing, robotics primitives, sensor abstractions, development tools
- Multi-Platform
  - Linux, Windows, OS X, RTOS, etc.

- ROS 2 Enhanced Capabilities
  - Real-time
  - Multi-threaded
  - Multi-robot; fleets
  - Legacy/lossy network

ROS = Plumbing + Tools + Capabilities + Ecosystem

# What We Do
## Consulting

- Recognized Robotics Expertise
  - Stewards of ROS/ROS 2 and Gazebo
  - Exposure to hundreds of robotics projects (commercial and research)
  - Direct line to top ROS developers
- Proven Results
  - Dozens of projects completed for major organizations
- Easy to Onboard
  - Available on demand for engagements from days to months to years
- High-Velocity Development Approach
  - Unique experience using simulation to shorten development cycle and improve outcomes

# What We Do
## ROS 2

### ROS 2 Corporate Sponsors

amazon.com

arm

Google

intel

Microsoft

TOYOTA RESEARCH INSTITUTE

UNIVERSAL ROBOTS

### ROS 2 Technical Steering Committee Members

ADLINK TECHNOLOGY INC.

amazon.com

Apex.AI

arm

BOSCH

CANONICAL

EPROSIMA The Middleware Experts

intel

LG Electronics

Microsoft

open robotics

ROBOTIS

TARDEC

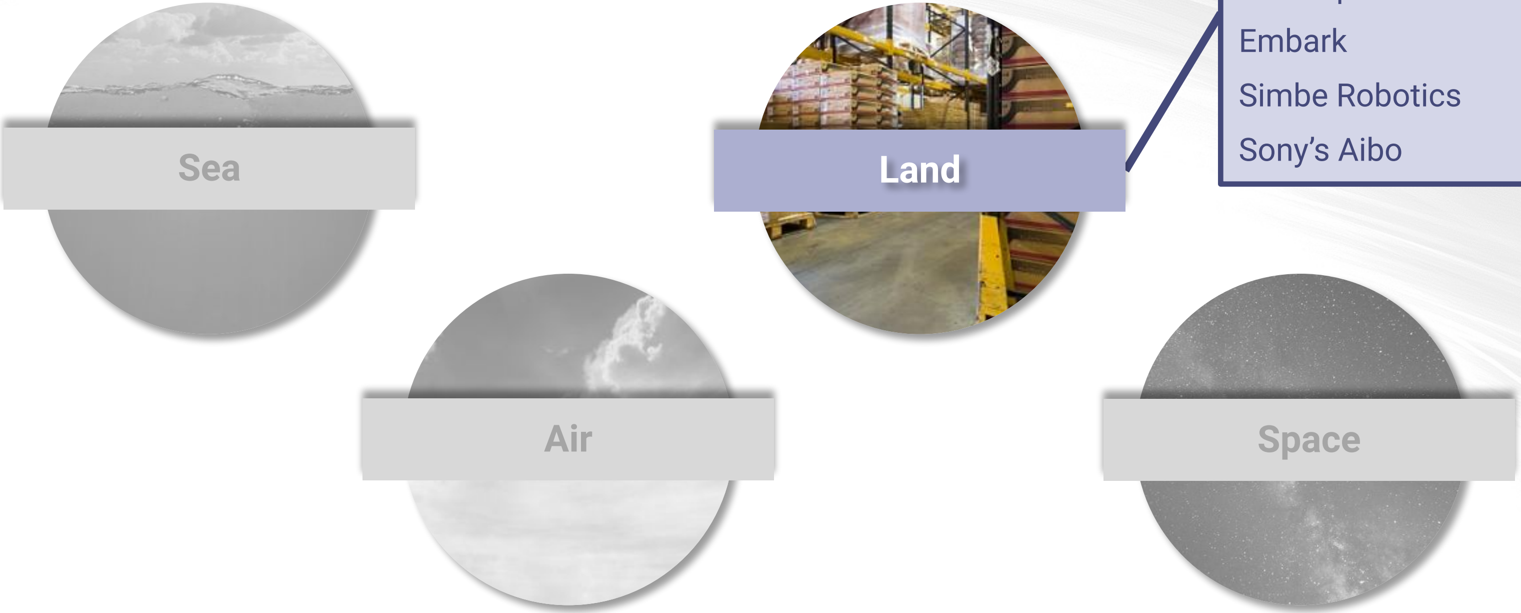
TOYOTA RESEARCH INSTITUTE

# Clients and Users

## Commercial Adoption

# Clients and Users

## Commercial Adoption



Sea

Land

Air

Space

- Apex.AI
- Cobalt Robotics
- BDI’s Spot
- Embark
- Simbe Robotics
- Sony’s Aibo

# Clients and Users

## Commercial Adoption

Agriculture

Consumer

Healthcare

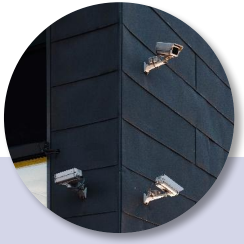
Inventory

Material Transport

Medicine

Research

Retail



Security

Transportation

# Clients and Users

## Commercial Adoption

**Agriculture**

American Robotics
Blue River Technology/John Deere
Iron Ox
Root.AI

Inventory

Material Transport

Medicine

Research

Retail

Security

Transportation

# Clients and Users
## Commercial Adoption

Agriculture

Consumer

6 River Systems
Fetch Robotics
OTTO Motors
Rapyuta Robotics
Locus Robotics

**Material Transport**

Medicine

Research

Retail

Security

Transportation

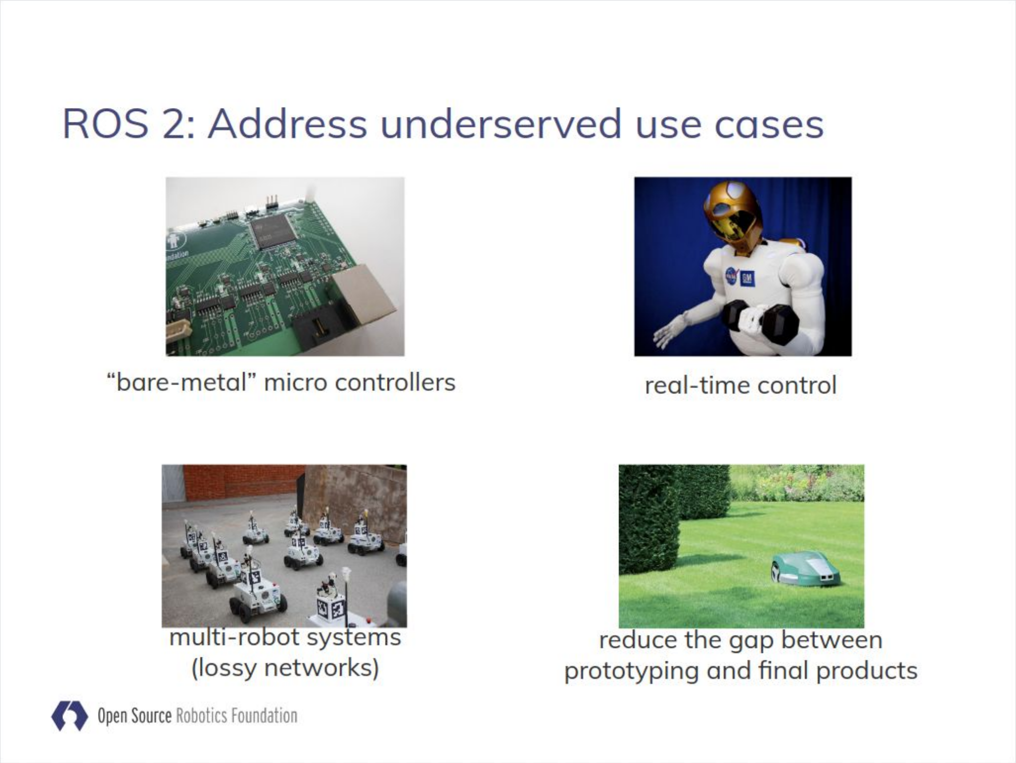
# ROS For Products Workshop
## Vision

Motivation:

ROS used in research but not in commercial applications

Results of 2013 workshop

- Embedded
- Multi-robot
- Realtime
- Commercial development
  - Cross Platform



## ROS 2: Address underserved use cases

"bare-metal" micro controllers

real-time control

multi-robot systems
(lossy networks)

reduce the gap between
prototyping and final products

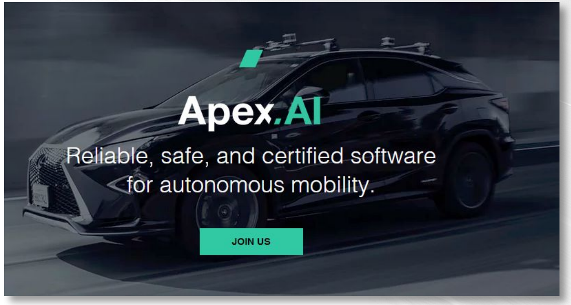
Open Source Robotics Foundation

# Clients and Users
## Case Study: iRobot

- Problem
  - Continue to innovate and develop new systems
  - Need to share developments across all their platforms
  - Accelerate research and development cycles
- Solution
  - Validate ROS 2 running on smaller embedded systems
  - Joined ROS 2 TSC and has helped drive development
- Results
  - All Roombas i Series and higher now use ROS 2 internally as of 2021
  - iRobot Create 3 has a ROS 2 interface for users

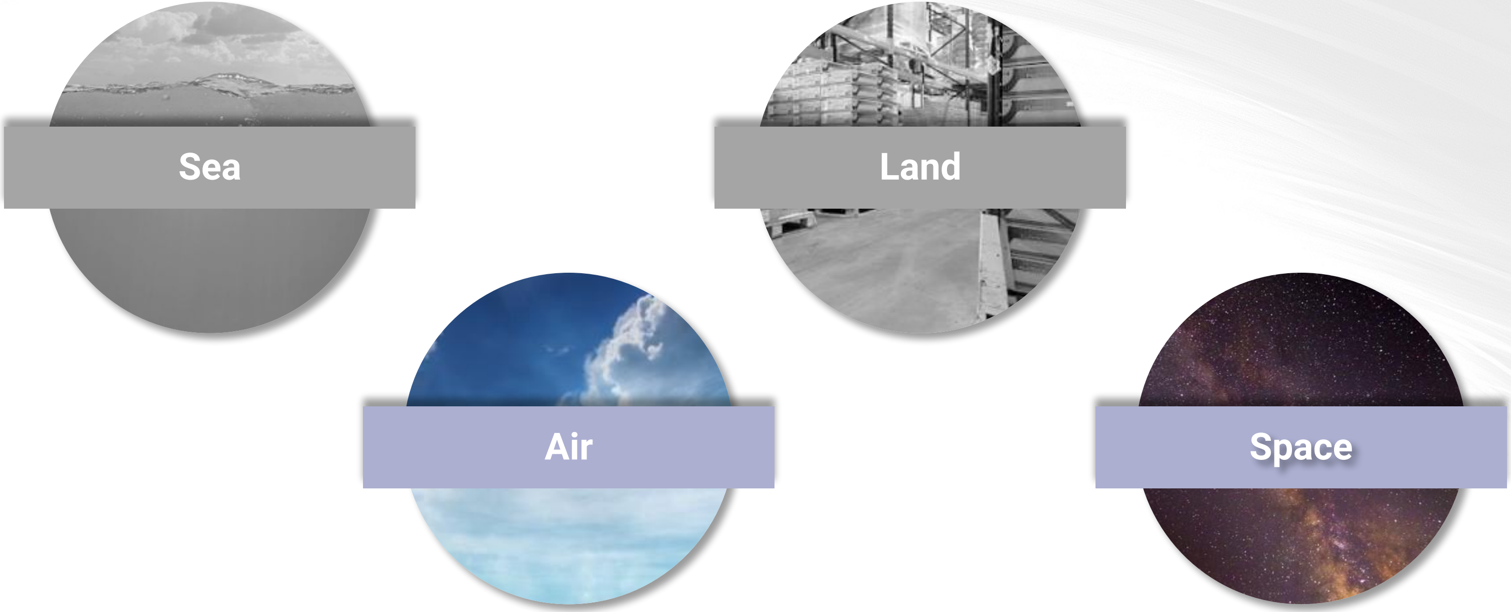
# Clients and Users
## Case Study:  Apex.AI

- Problem
  - Establish a rock-solid platform for new autonomous mobility solution
  - Meet safety and reliability standards for both automakers and consumers within new Apex.AI “Android-like” software stack
  - Architect application-specific, mission-critical enhancements within ROS 2
- Solution
  - Designed ROS 2 features for enabling safety critical applications
  - Prioritized real-time-safe APIs
  - Merged code changes upstream, include code in open source releases
  - Provided expert advice through iterative design review
- Results
  - ROS 2 system redundancies ensure single failures don’t trigger system-wide failures
  - Application agnostic approach for any autonomous system — cars to drones to flying taxis


Apex.AI
Reliable, safe, and certified software for autonomous mobility.
JOIN US

# Clients and Users

## Commercial Adoption



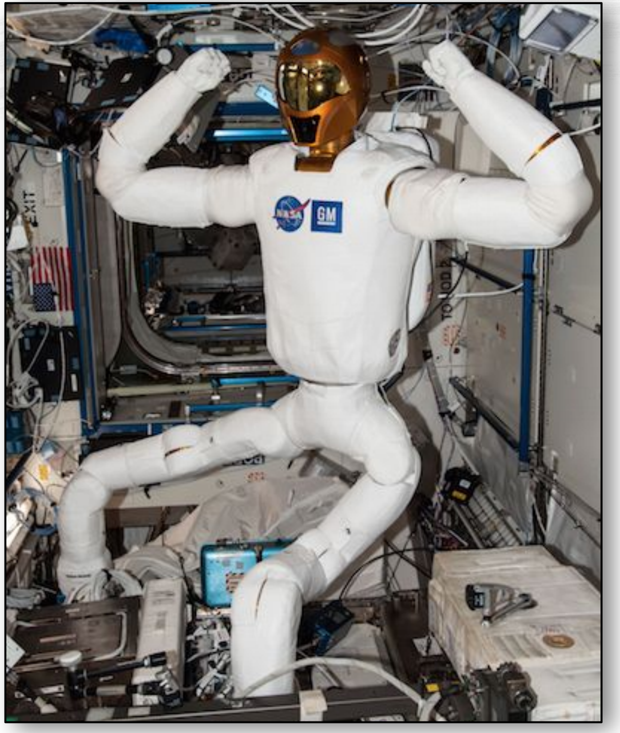
Sea

Land

Air

Space

# ROS In Space



2014: Robonaut 2

2019: Astrobee

# ROS In Flight

PX4 autopilot

# VIPER

*Prospecting for lunar resources in permanently shadowed regions of the lunar south pole*

- **ROS** used in ground software systems
- **Gazebo** simulation used in mission development, testing, planning, operator training, etc.
- Other open source software
  - cFS/ROS bridge
  - Yamcs
  - VERVE
- NASA requires software used in *flight missions* to be space qualified

# Feature of ROS 2 to support Certification Process

- Modularity
  - Allowing paring down
  - Separate parts requiring certification versus tooling
- Good abstractions
  - Decoupling independent elements
  - Support for custom allocators
- Designed to be pared down
  - Optional elements can be compiled out such as logging
  - Support for custom allocators

spaceROS

# What we need: A version of ROS for space applications!

- A space-certifiable and reusable robotic framework
  - Designed to meet flight software standards
  - Aligned with NASA so that it can be adopted for Class A missions
- Support characteristic space robotics applications
- Enable rapid development of new robotic capabilities
- Based on open community, frameworks, and standards

spaceROS

## An open source, “qualification-ready,” ROS-based distribution

- Support certification with DO178C and NPR7150.2
- Allow space flight projects gain a head start on their certification efforts
- Facilitate reuse across missions, reducing development effort and costs
- Focus on source code quality; use code analysis tools to assess/improve the code
    - Static analysis, testing, dispositioning and addressing issues, requirements & traceability
- Sandboxing certain language features (e.g., memory allocation, exceptions)

spaceROS

## Provide space-specific functionality, such as

- Additional modules like star tracker, terrain hazard analysis
- Operator tools (e.g., UIs like VERVE)
- Common telemetry and command formats as ROS messages
- Bridging to existing tools and frameworks, like cFS
- Integration with CCSDS Asynchronous Message Service

# SpaceROS

## Progress to Date

- Space ROS github organization
- A core set of ROS packages
- Continuous Integration (CI) system for Space ROS builds
- Docker scripts for Space ROS (and MoveIt2 built against Space ROS)
- Additional static analysis tools integrated, including NASA’s Cobra and IKOS
  - AutoSAR C++ 14 coding guidelines support in the works for Cobra
  - Addressed many issues identified by static analysis tools
- SARIF output for all static analysis tools

# SpaceROS
## Projects in Flight

- Avoidance of dynamic memory allocation
  - Requirements, design, and implementation underway
- Eventing and Logging Subsystem
  - Requirements defined
- Dashboard
  - Based on SARIF input data collected from the analysis tools
  - Requirements defined

# SpaceROS

## Next Steps

- Contact me (tfoote@openrobotics.org)
- Work with the new static analysis tool integration
- Help us refine the Space ROS roadmap
- Make code contributions
  - https://github.com/space-ros

# Medical ROS

A new domain for ROS

Strong interest following Automotive and Space proof of concept

# Medical ROS
## Vision

A subset of ROS which can be used in medical devices

- IEC 62304 Level B or C for runtime
- IEC 62304 Level A for tools
- Wire compatible with mainline ROS for faster development and debugging
  - Researchers & Developers can use full capability

### Tools to Level A

- colcon
- ament*
- Ignition simulation
- osrf_testing_tools_cpp

### Runtime Libraries to Level B/C

- rcl
- rcutils
- rccputils
- rosidl_runtime_c
- rcl_interfaces
- rcl_logging*
- rmw

### Compatible for R&D

- rviz
- rqt
- tracetools
- ros2 cli
- Rclpy

# Medical ROS
## Process

Open Source Distribution

- Push back as much as possible to the mainline to avoid forking

Improved QA process

- Setup a certified Quality Management System
- Improve requirements and development traceability
- Improve our testing and coverage

To enable the production of the necessary

- Safety Document
- Safety Justification Document

# Medical ROS
## Business Model

Building up a set of partners interested in using the product.

- Many are already using ROS 2 in their R&D and want to use it in their products.
  - They can directly contribute to the effort or contract us for new features at levels A B or C

Open Source Distribution

- The code remains open source
  - Push back anything that is valuable to the mainline to avoid divergance
- All process improvements results are visible and integrated into the core package development process

Products

- Safety Document
- Safety Justification Document

# Medical ROS
## Get Involved

Open development process looking forward

More partners will mean smaller individual investments

Early involvement will enable helping define the scope and process

We have anchor partners and have established client bases in healthcare.

If you’re interested please reach out to me:

Tully Foote tfoote@openrobotics.org

# Thank You

- Tully Foote tfoote@openrobotics.org

www.ros.org

www.openrobotics.org

www.gazebosim.org