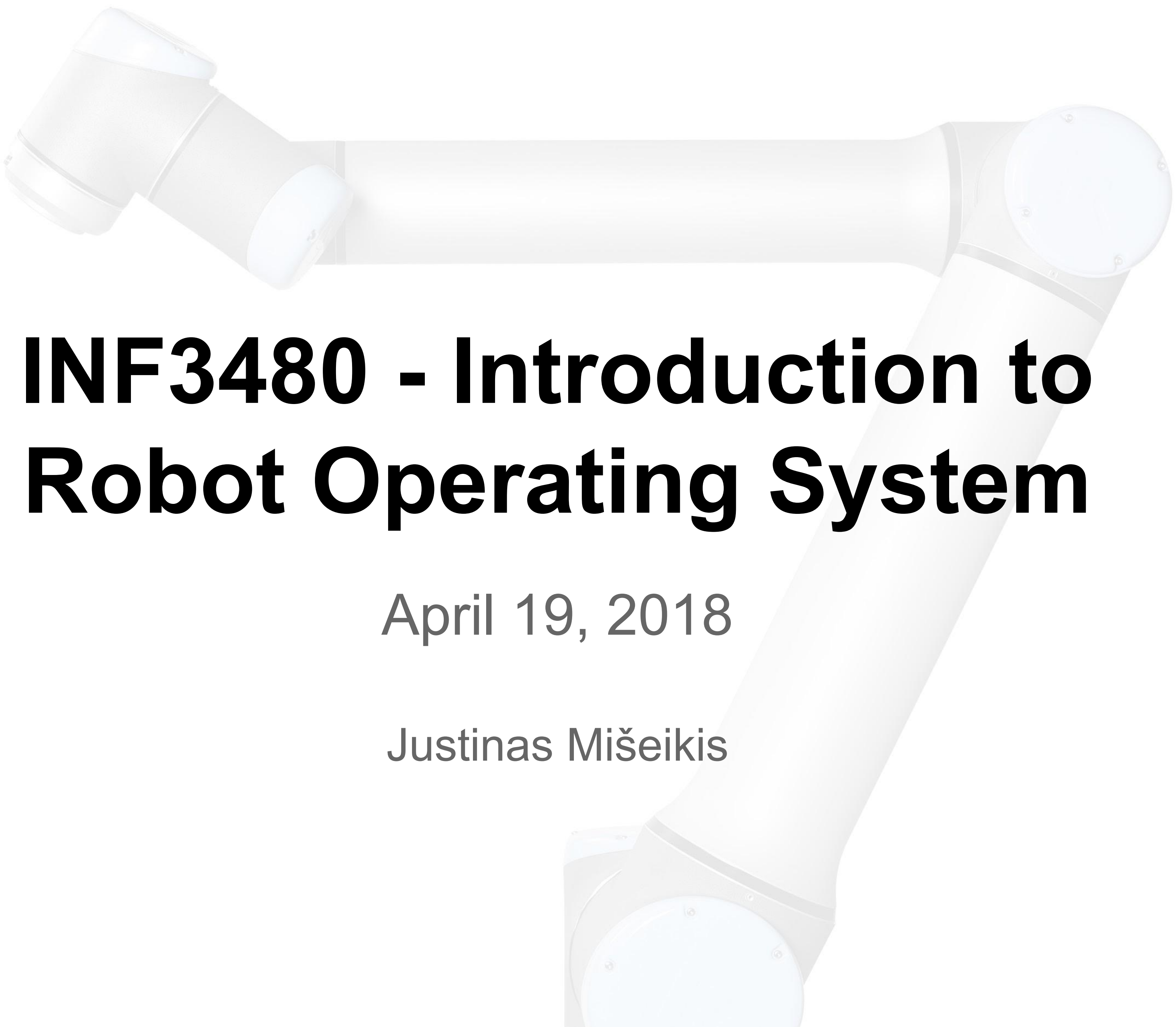
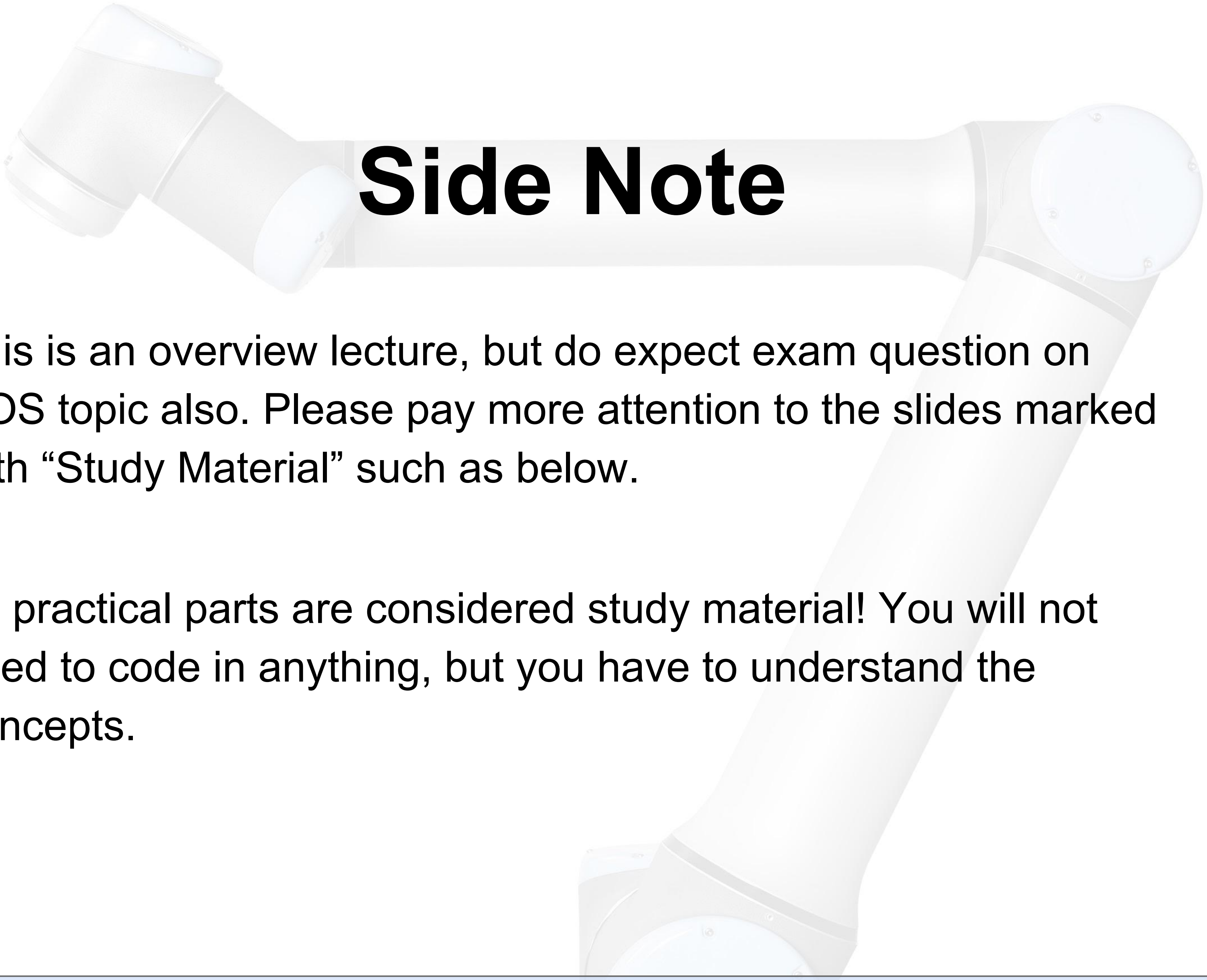


INF3480 - Introduction to Robot Operating System

April 19, 2018

Justinas Mišeikis

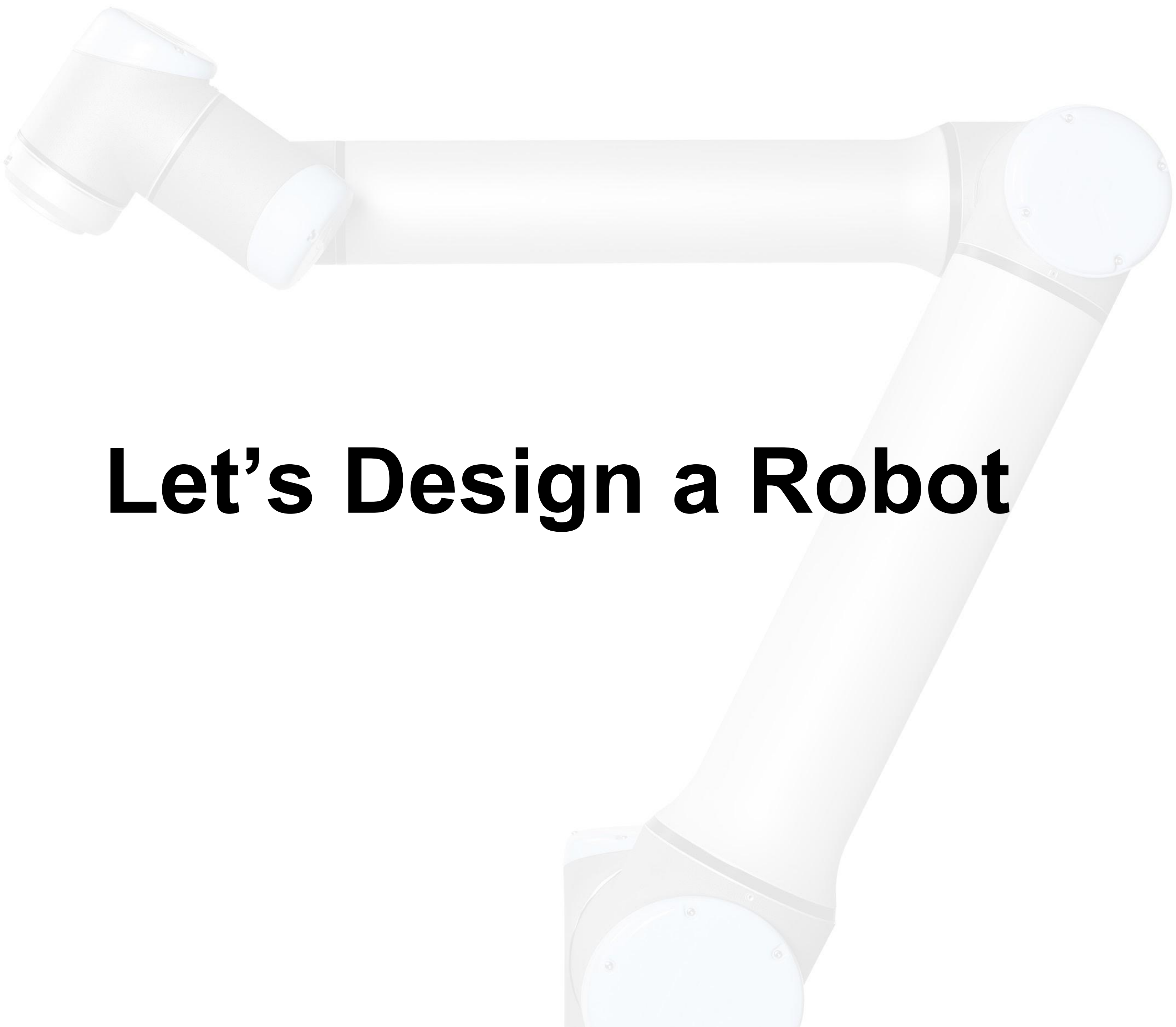


Side Note

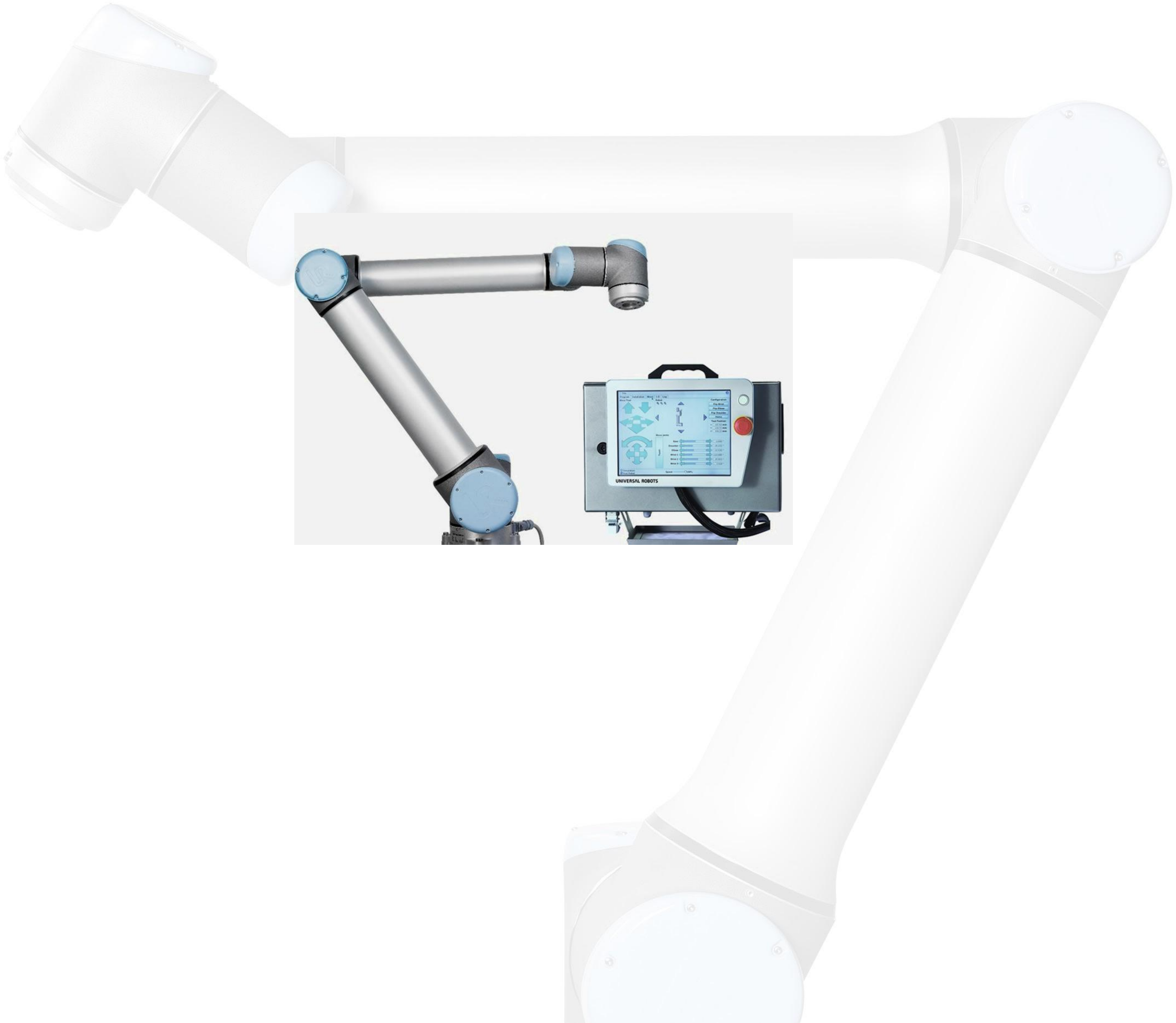
This is an overview lecture, but do expect exam question on ROS topic also. Please pay more attention to the slides marked with “Study Material” such as below.

All practical parts are considered study material! You will not need to code in anything, but you have to understand the concepts.

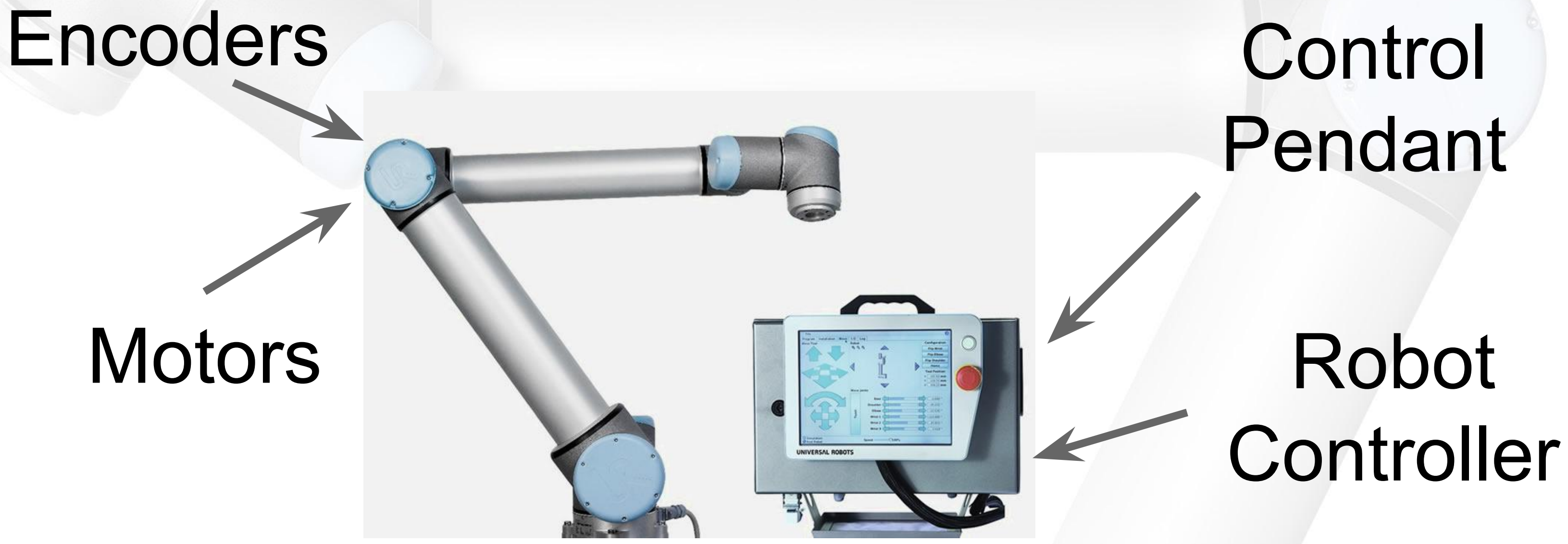
Study Material

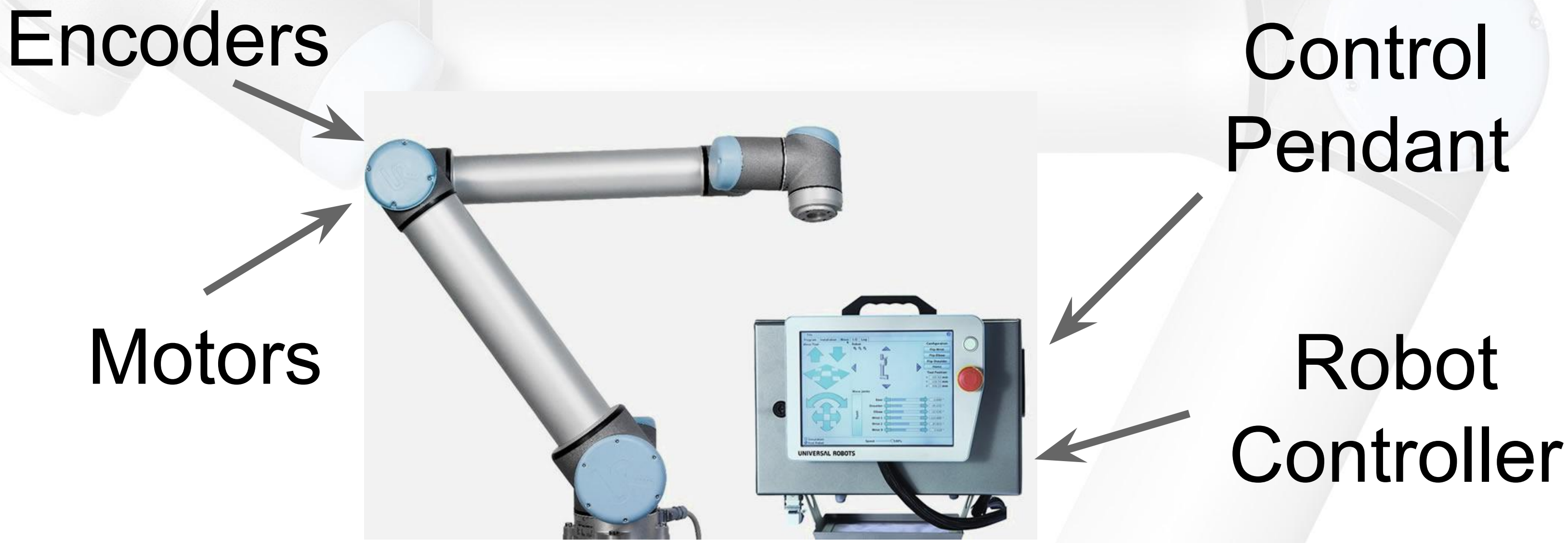


Let’s Design a Robot



UNIVERSAL ROBOTS


Encoders
Motors
Control Pendant
Robot Controller
UNIVERSAL ROBOTS



Motor controllers

Inverse Kinematics

GUI

Collision Detection

Trajectory Planner

PID Controller

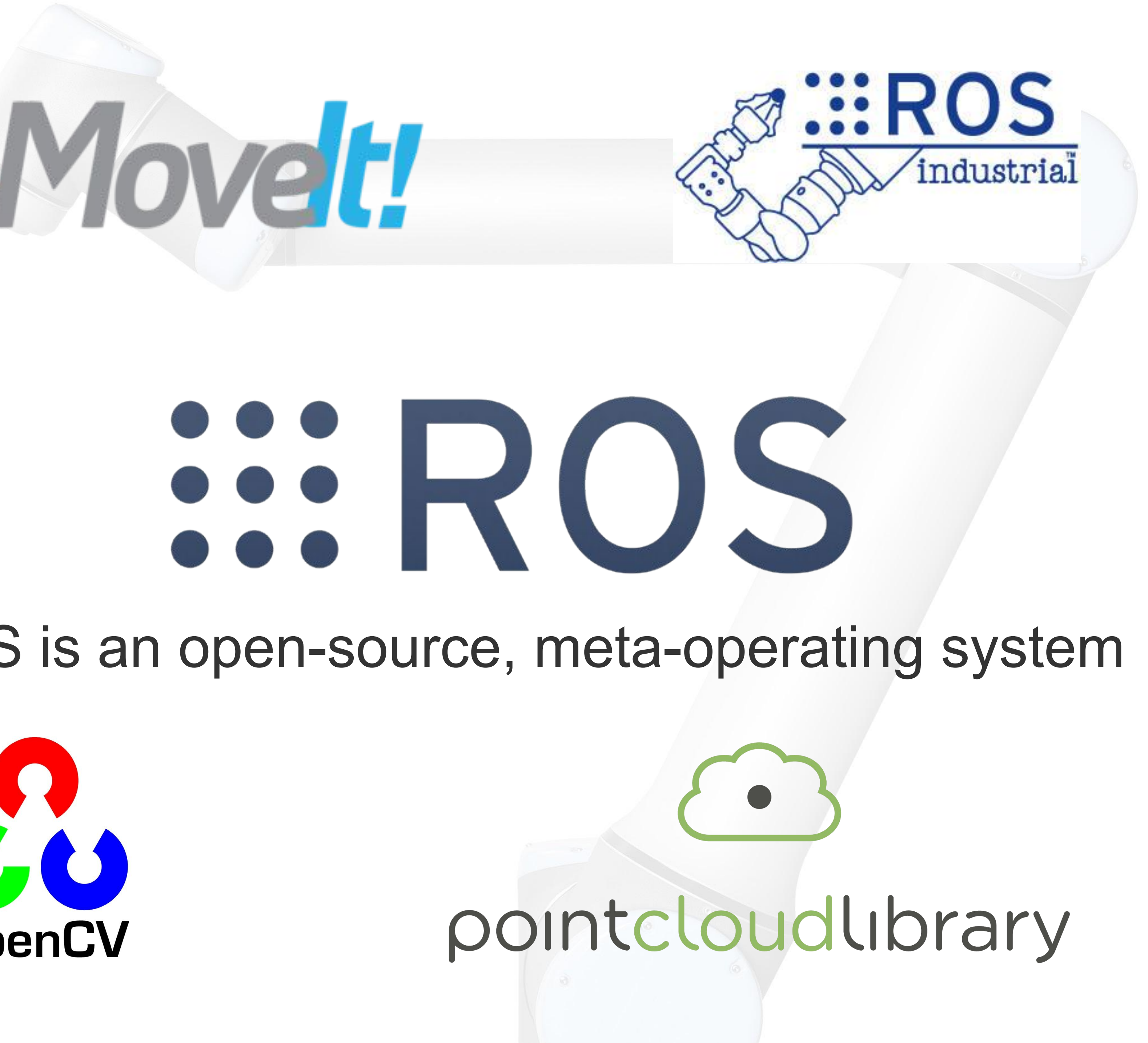
Error Monitoring

Teaching Mode

What a mess!

How can we deal with it?

ROS is an open-source, meta-operating system



MoveIt!

ROS industrial

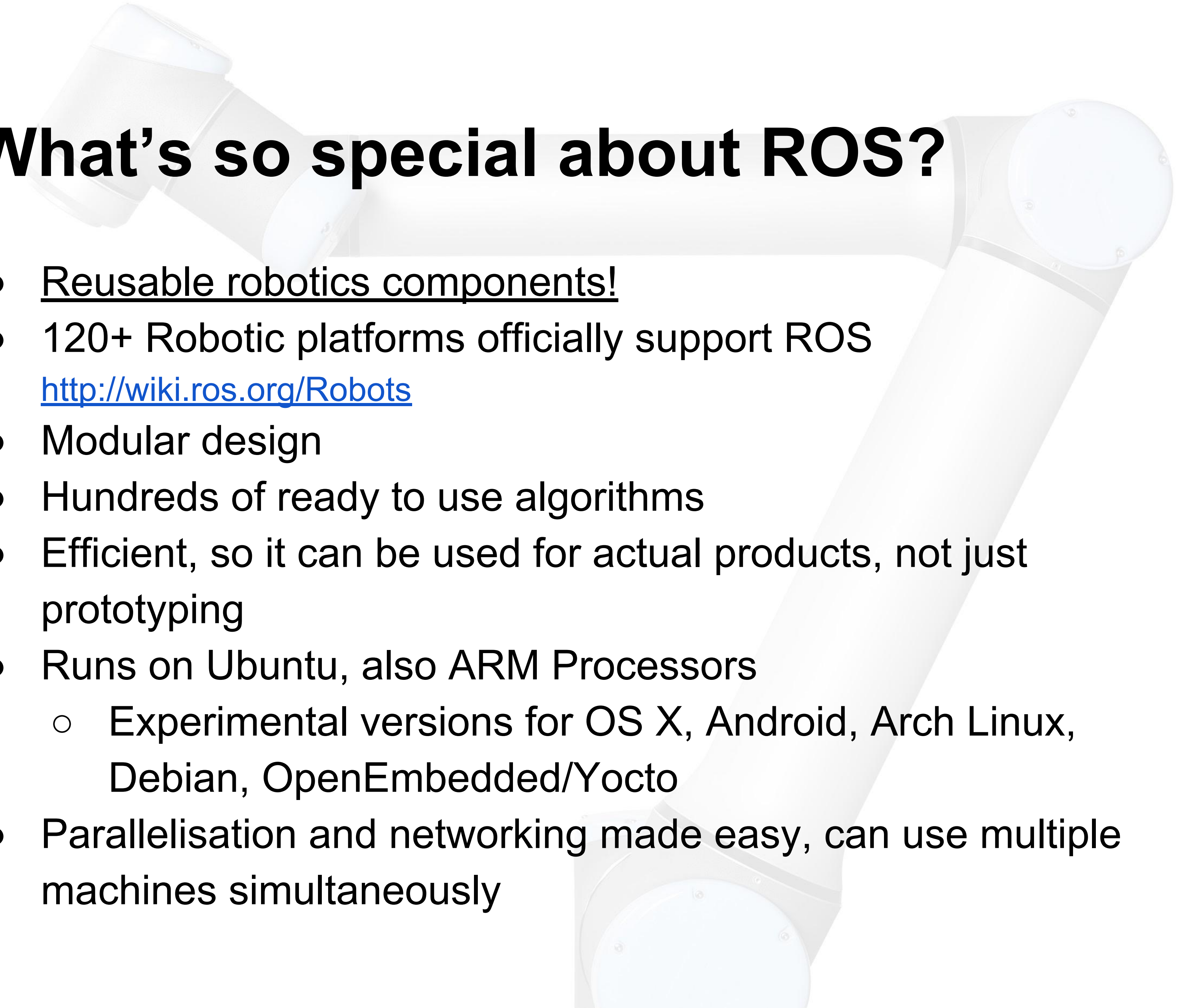
ROS

ROS is an open-source, meta-operating system

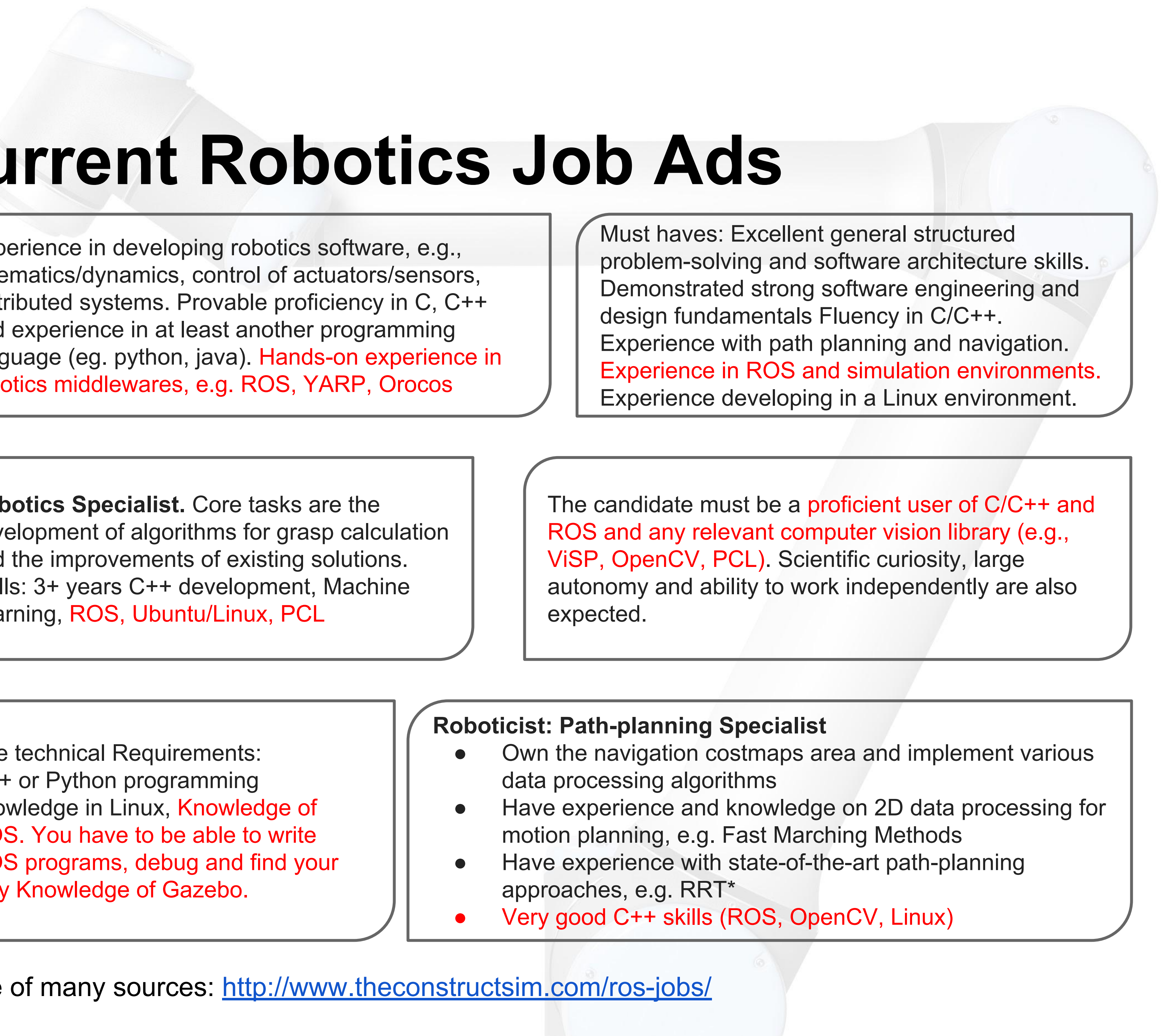
OpenCV

pointcloudlibrary

What's so special about ROS?



- <u>Reusable robotics components!</u>
- 120+ Robotic platforms officially support ROS
  http://wiki.ros.org/Robots
- Modular design
- Hundreds of ready to use algorithms
- Efficient, so it can be used for actual products, not just prototyping
- Runs on Ubuntu, also ARM Processors
  - Experimental versions for OS X, Android, Arch Linux, Debian, OpenEmbedded/Yocto
- Parallelisation and networking made easy, can use multiple machines simultaneously



Current Robotics Job Ads

Experience in developing robotics software, e.g., kinematics/dynamics, control of actuators/sensors, distributed systems. Provable proficiency in C, C++ and experience in at least another programming language (eg. python, java). Hands-on experience in robotics middlewares, e.g. ROS, YARP, Orocos

Must haves: Excellent general structured problem-solving and software architecture skills. Demonstrated strong software engineering and design fundamentals Fluency in C/C++. Experience with path planning and navigation. Experience in ROS and simulation environments. Experience developing in a Linux environment.

Robotics Specialist. Core tasks are the development of algorithms for grasp calculation and the improvements of existing solutions. Skills: 3+ years C++ development, Machine Learning, ROS, Ubuntu/Linux, PCL

The candidate must be a proficient user of C/C++ and ROS and any relevant computer vision library (e.g., ViSP, OpenCV, PCL). Scientific curiosity, large autonomy and ability to work independently are also expected.

The technical Requirements:
C++ or Python programming knowledge in Linux, Knowledge of ROS. You have to be able to write ROS programs, debug and find your way Knowledge of Gazebo.

Roboticist: Path-planning Specialist

- Own the navigation costmaps area and implement various data processing algorithms
- Have experience and knowledge on 2D data processing for motion planning, e.g. Fast Marching Methods
- Have experience with state-of-the-art path-planning approaches, e.g. RRT*
- Very good C++ skills (ROS, OpenCV, Linux)

One of many sources: http://www.theconstructsim.com/ros-jobs/

ROS

Plumbing

Tools

Capabilities

Ecosystem

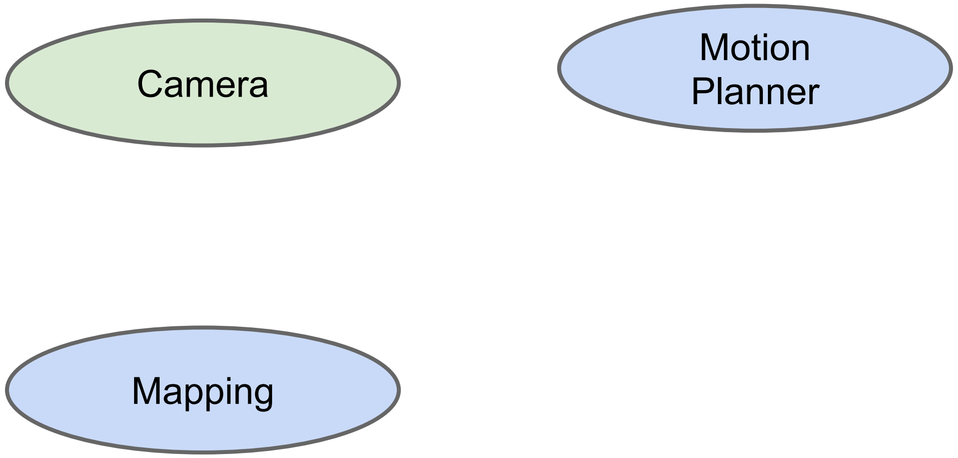
Let’s see how it works!

“Plumbing”

Nodes

http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes

Nodes are processes that perform computation, “executables”. Each node performs a specific processing part, usually a part of the algorithm.



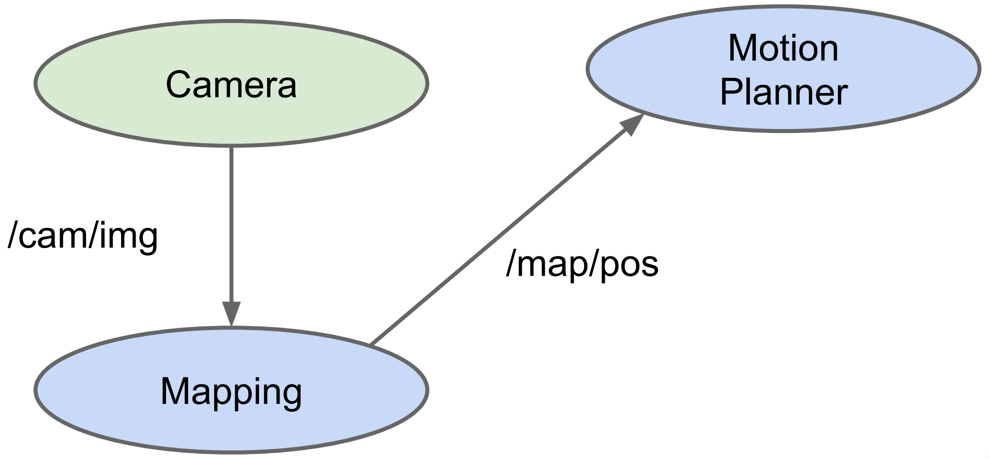
Study Material - highly recommend to check the URL

Topics

http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics

Topics are streams of data with publish / subscribe semantics.

They are uniquely identifiable by its name. Nodes can publish and subscribe to topic in order to transfer data.

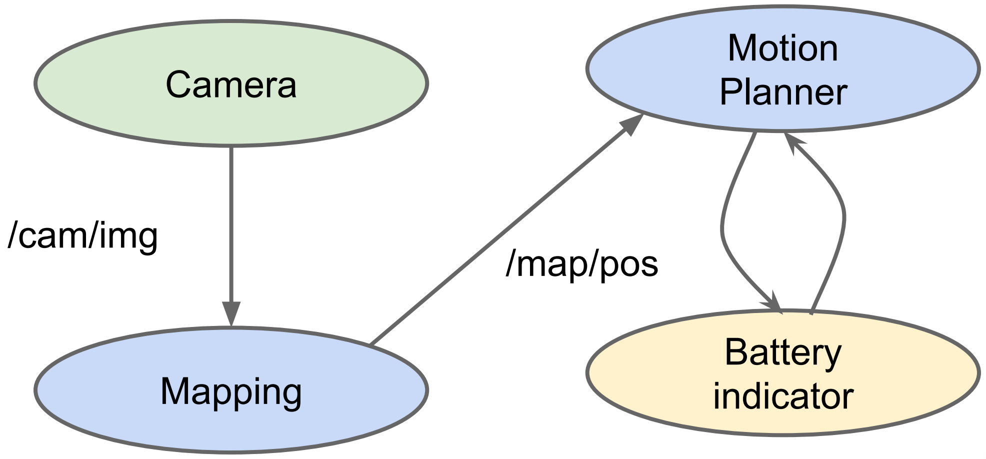


Study Material - highly recommend to check the URL

Services

http://wiki.ros.org/ROS/Tutorials/UnderstandingServicesParams

Request / reply is done via services, which are defined by a pair of message structures: one for the request and one for the reply.



Study Material - highly recommend to check the URL

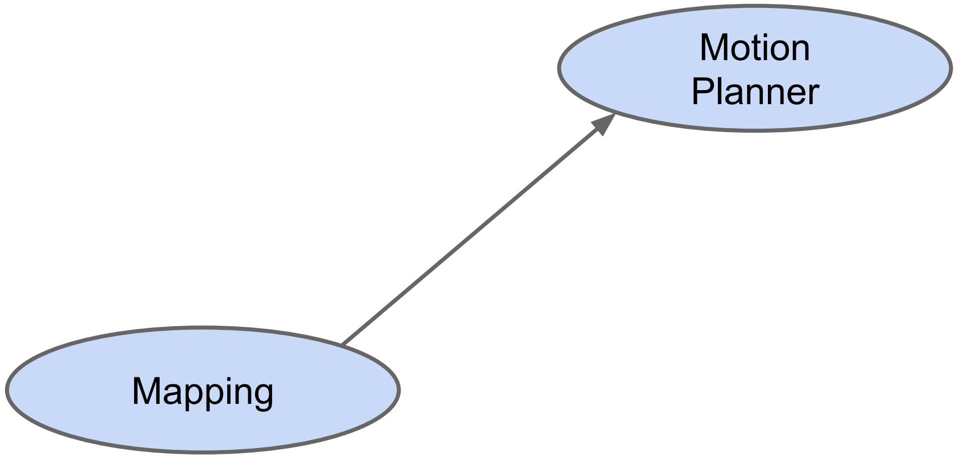
Messages

http://wiki.ros.org/ROS/Tutorials/CreatingMsgAndSrv

A message is simply a data structure, comprising typed fields.

Language agnostic data representation. C++ can talk to Python.

Messages are sent on defined topics.

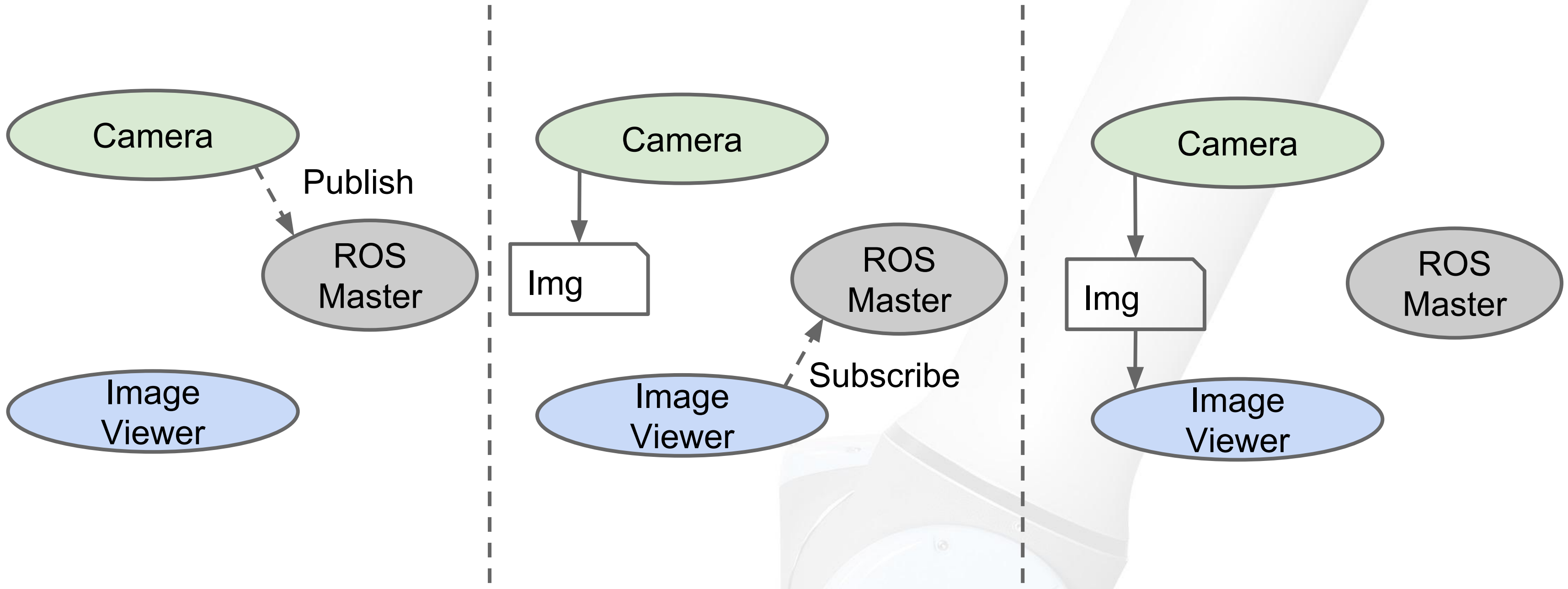


File: pos.msg

```
string robotName
uint32 posX
uint32 posY
uint32 goalX
uint32 goalY
```

ROS Master

The ROS Master provides name registration and lookup to nodes. Without the Master, nodes would not be able to find each other, exchange messages, or invoke services.

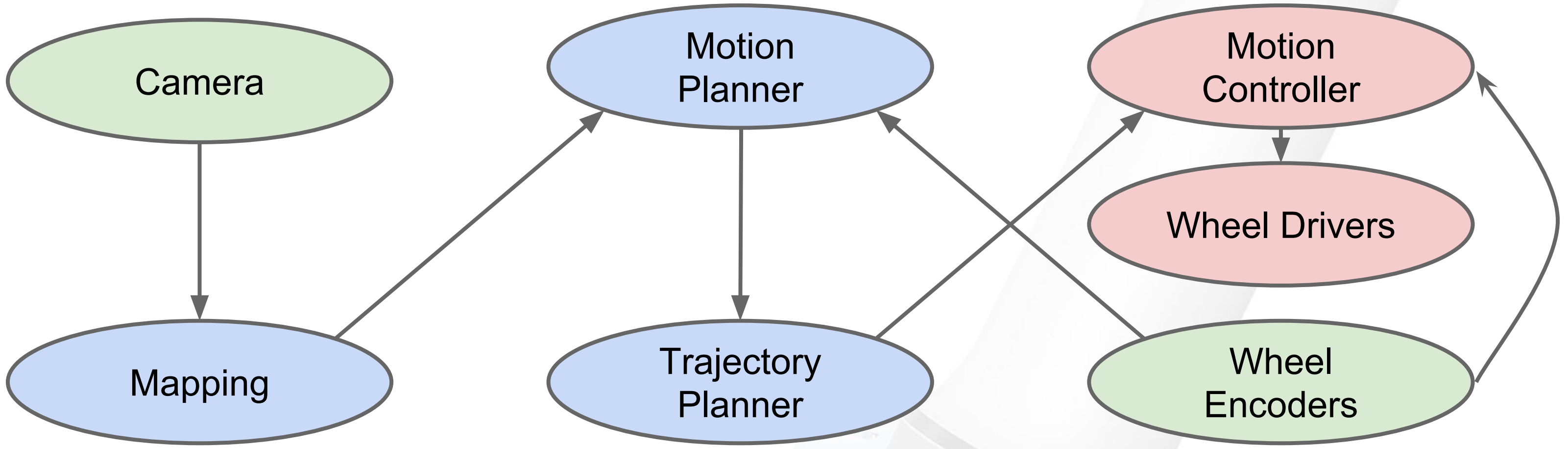


Example System - Mobile Robot

Green - Sensors

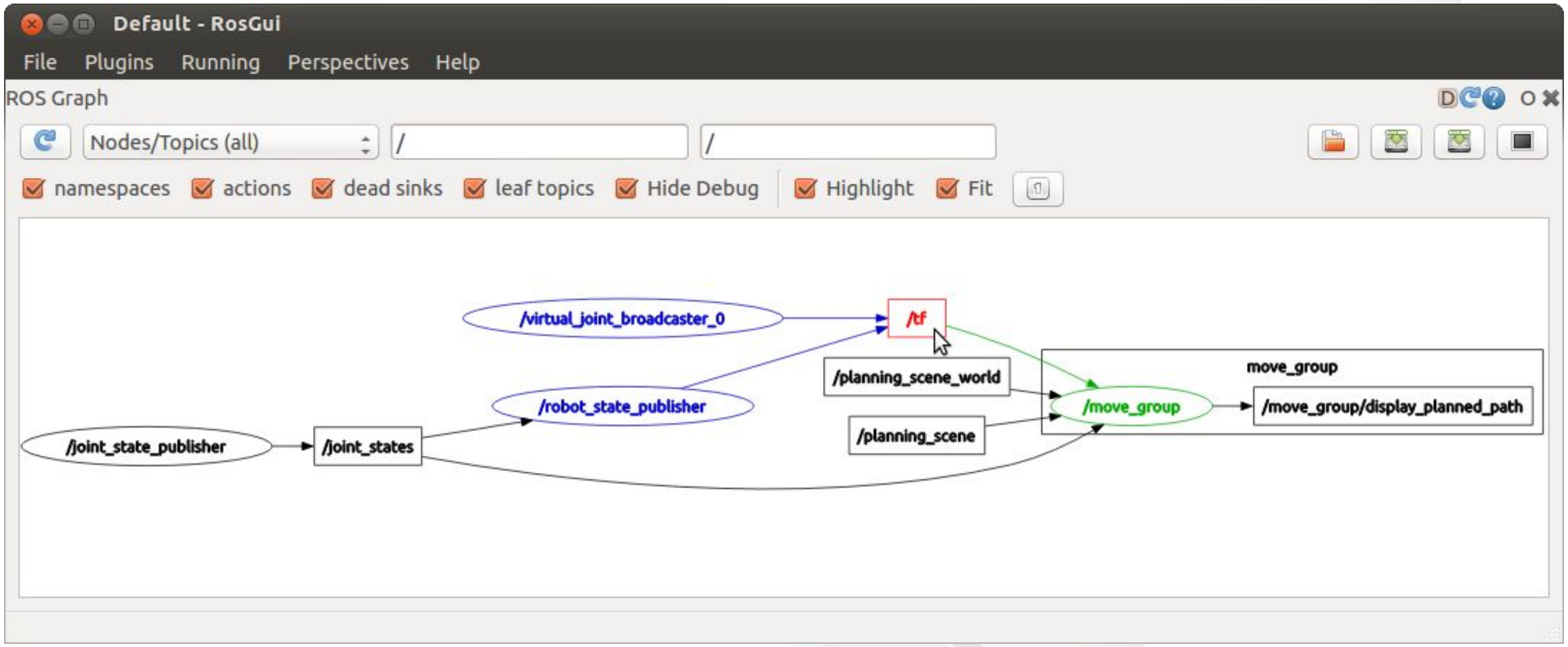
Blue - Planning algorithms

Red - Hardware integration

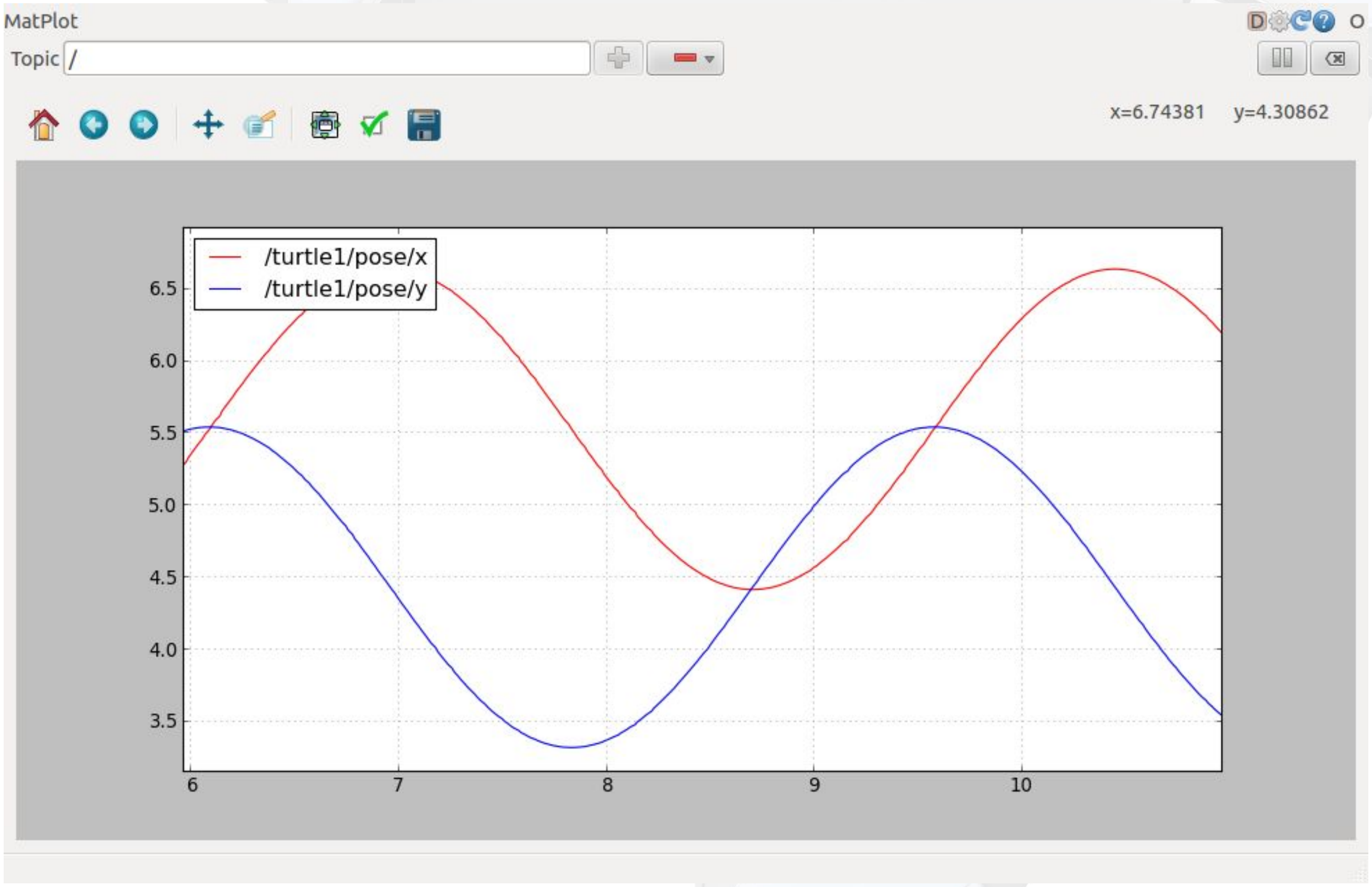


“Tools”

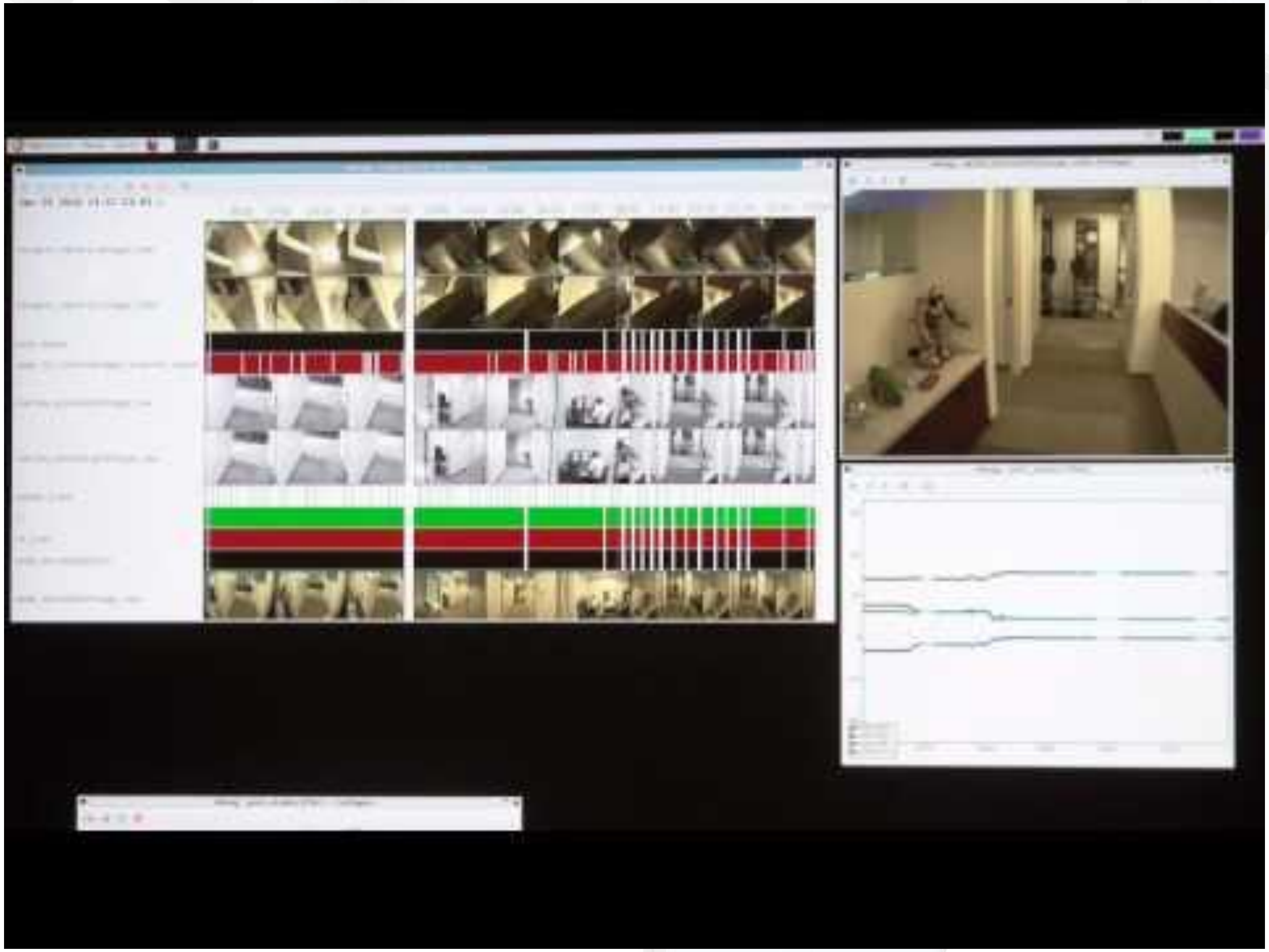
System Visualisation: rqt_graph



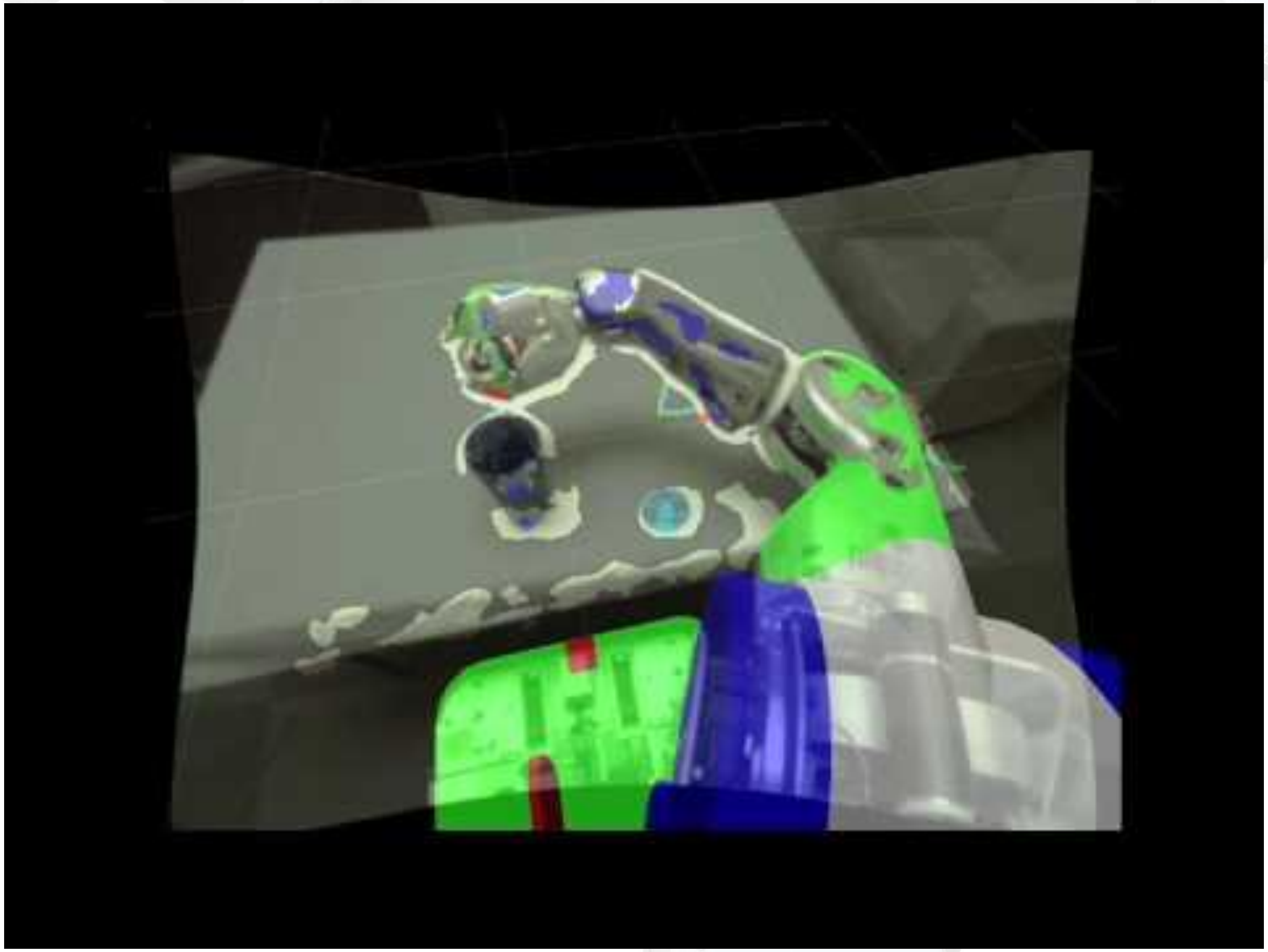
Live Plotting: rqt_plot



Logging and Visualization Sensor Data: rosbag and rqt_bag



3D Visualisation: RVIZ



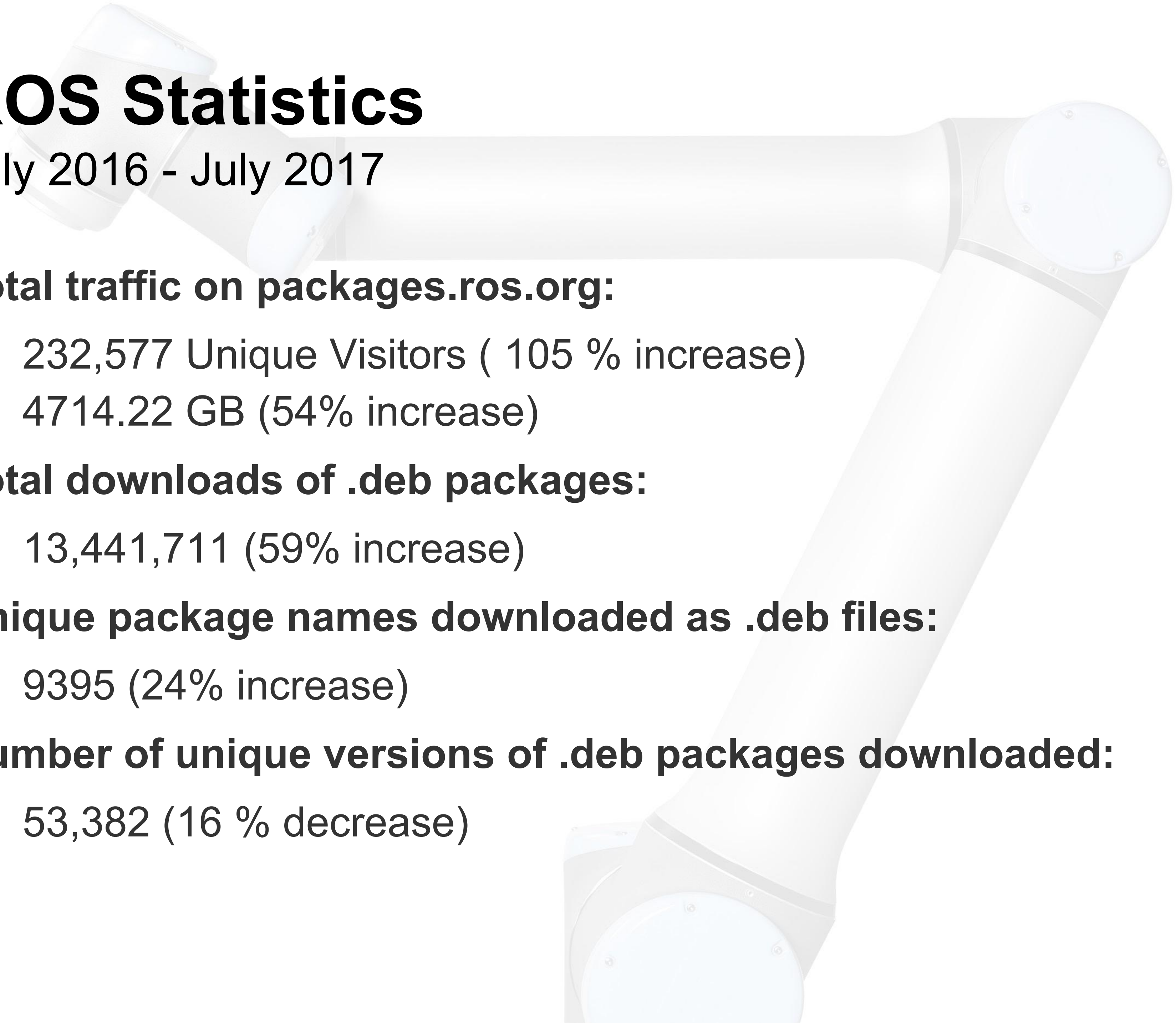
“Capabilities”

Team MAXed Out

The switching between tasks is safe, stable, and smooth, and requires minimal human effort.

Learning Algorithms and Systems Laboratory (LASA) | EPFL

“Ecosystem”



ROS Statistics

July 2016 - July 2017

Total traffic on packages.ros.org:

- 232,577 Unique Visitors (105 % increase)
- 4714.22 GB (54% increase)

Total downloads of .deb packages:

- 13,441,711 (59% increase)

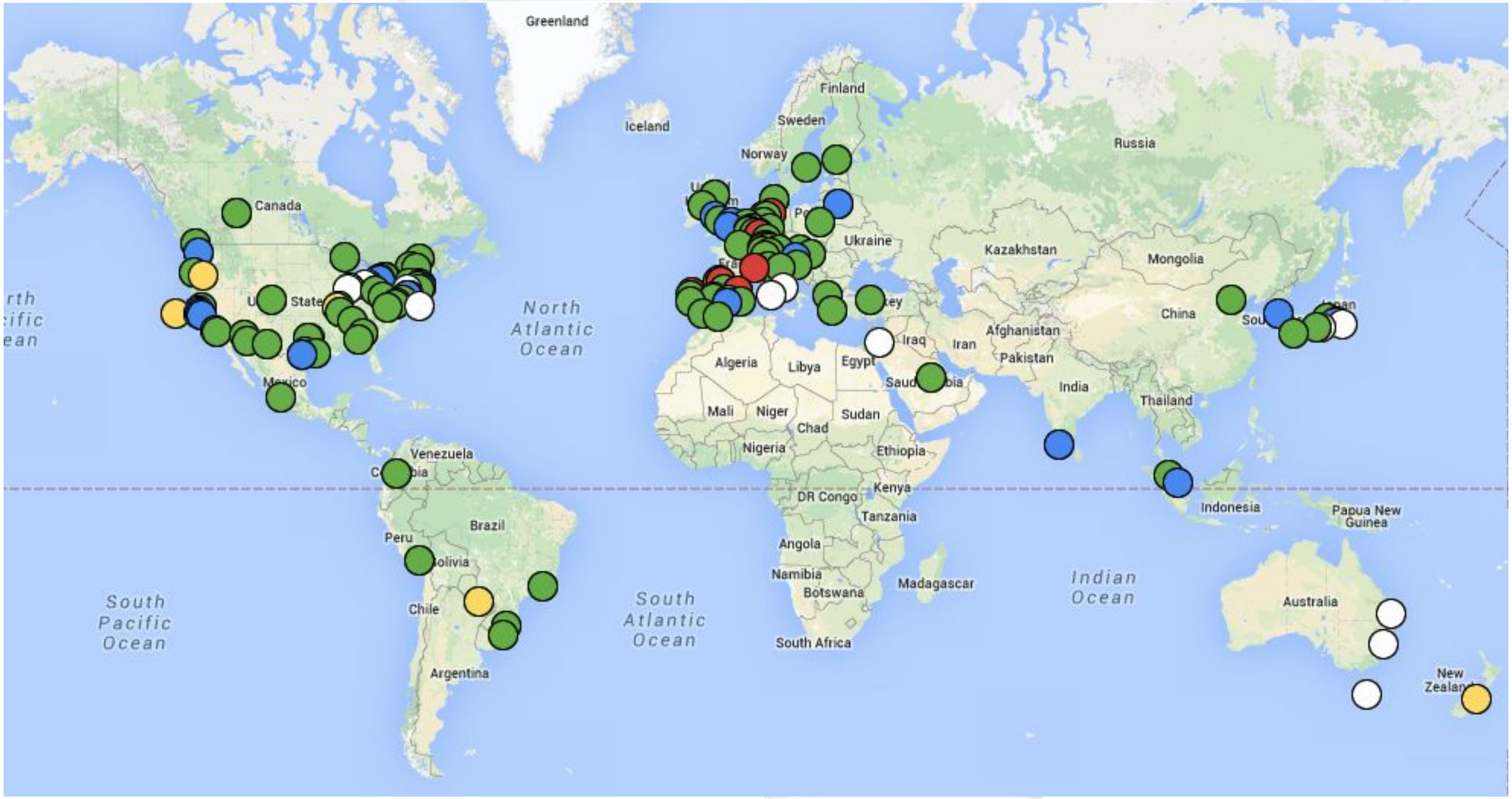
Unique package names downloaded as .deb files:

- 9395 (24% increase)

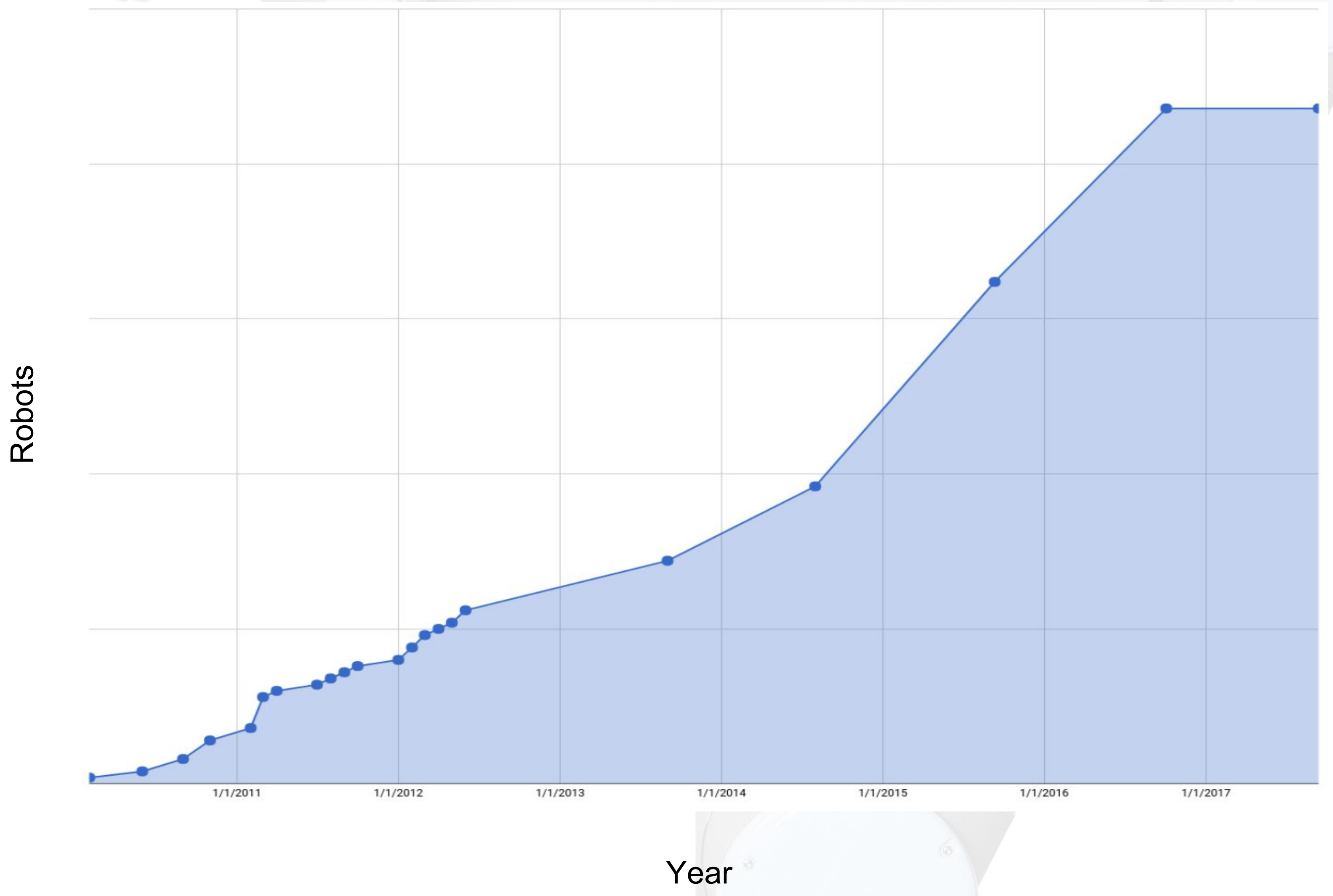
Number of unique versions of .deb packages downloaded:

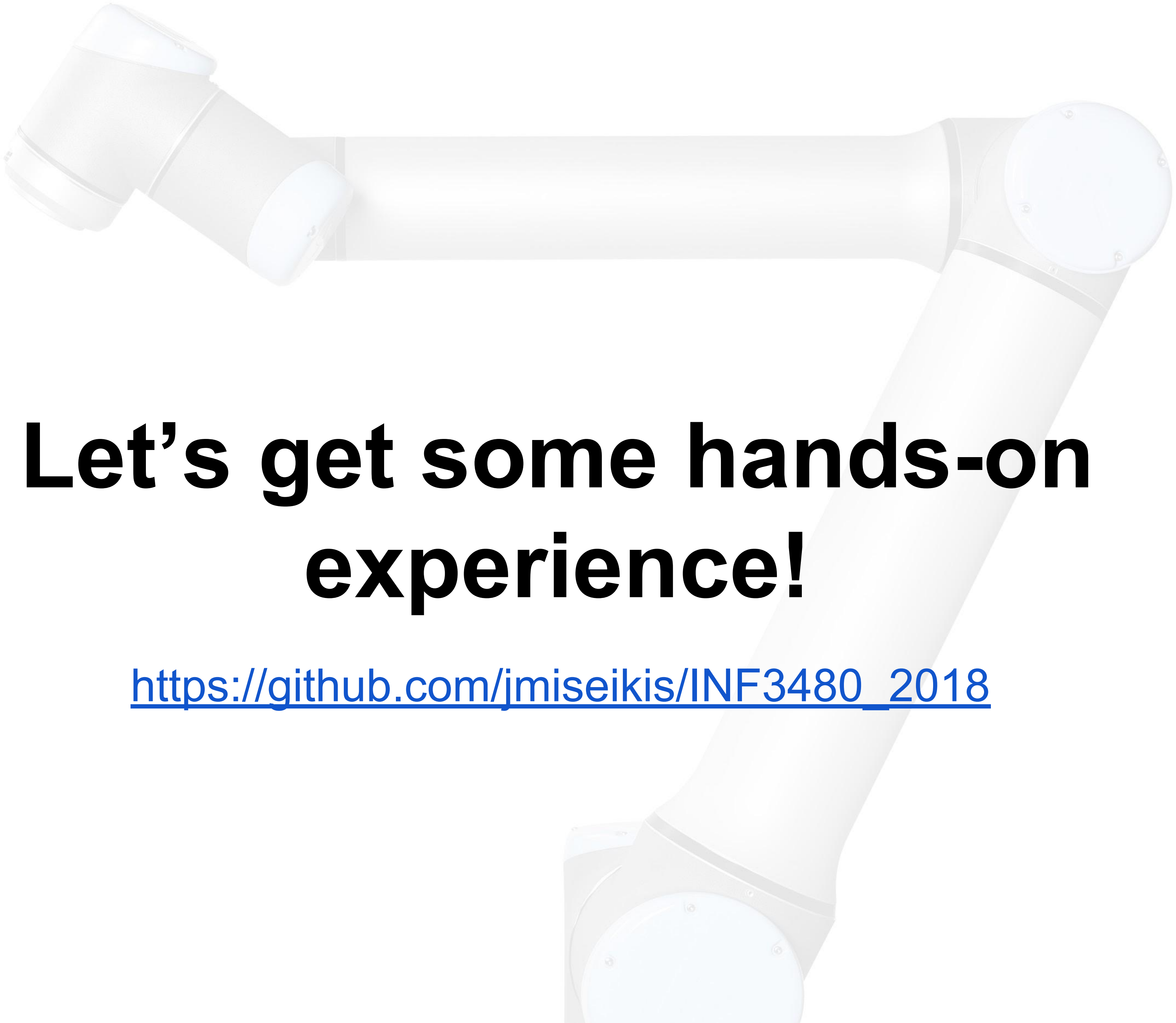
- 53,382 (16 % decrease)

Worldwide User Base



Number of Robots Supporting ROS





Let’s get some hands-on experience!

https://github.com/jmiseikis/INF3480_2018

Thank You!

Any Questions?