MANCHESTER 1824
The University of Manchester

1st SmartArmStack Workshop

Hopefully not the last

Part 1 - Introduction

Murilo M. Marinho
Lecturer in Robotics
EEE, UoM

November 29th, Eagle Labs (Top Deck) & RAICo1 Rig Hall/Glovebox Bay

Motivation

MANCHESTER 1824
The University of Manchester

Why was I ~~forced~~invited to be here?

- Project Sustainability: Person leaves, projects die off
  - Research stalled
  - Follow-up projects must reinvent the wheel.
- Project Collaboration: no mutual benefit
  - Person A creates code with their preferred settings
  - This code is indecipherable to any person $B \neq A$.
- Time efficiency: On-topic procrastination
  - Person C spends 6 months making yet another robot driver unusable for any $D \neq C$.
  - This cannot be used directly for a research paper, just as support for experiments.
  - This cannot (should not) be part of their PhD thesis.

Keep legacy alive

- DQ Robotics
  - The de facto common language among many groups.
  - Contribute here always when suitable.
- SmartArmStack
  - Complementary to DQ Robotics.
  - Contribute here always when suitable.
- Whatever you propose next
  - Complementary to DQ Robotics and SmartArmStack.

Use SmartArmStack!

The company's repo has a ROS driver already!!!!

Existing ROS drivers are pure

RUBBISH

**95% of the drivers available online for ROS are pure rubbish. Number is extremely accurate for sure.*

MANCHESTER 1824
The University of Manchester

Why are existing drivers rubbish?

- Not maintained
  - [redacted]
- ROS code is mixed up with the driver code 🤮.
- Overreliance on MoveIt!
  - Most drivers have a buffer to smooth out the trajectory
  - Only meant for pre-computed trajectories
  - Poor reactive behavior

But company!!!
Industry!!!
Amazing!!!

IS

NO MATTER BY WHOM

What is SAS?

SmartArmStack in a glance

- Arguably not rubbish
- Abstract away driver code from ROS code
- Abstract away kinematic controller code from ROS code
- Many utility functions
  - Datalogger (export data to .mat)
  - Clock (a proper clock with data-keeping and statistics)
  - Conversions between ROS interfaces, DQ Robotics, Eigen…
- C++ and Python (pybind11) support

For existing robots

- Use and rejoice in C++ and Python!
  - You can benefit from kinematic controllers
  - Combine them with other robots
  - Benefit from teleoperation packages
  - And so on

MANCHESTER 1824
The University of Manchester

For any new robot

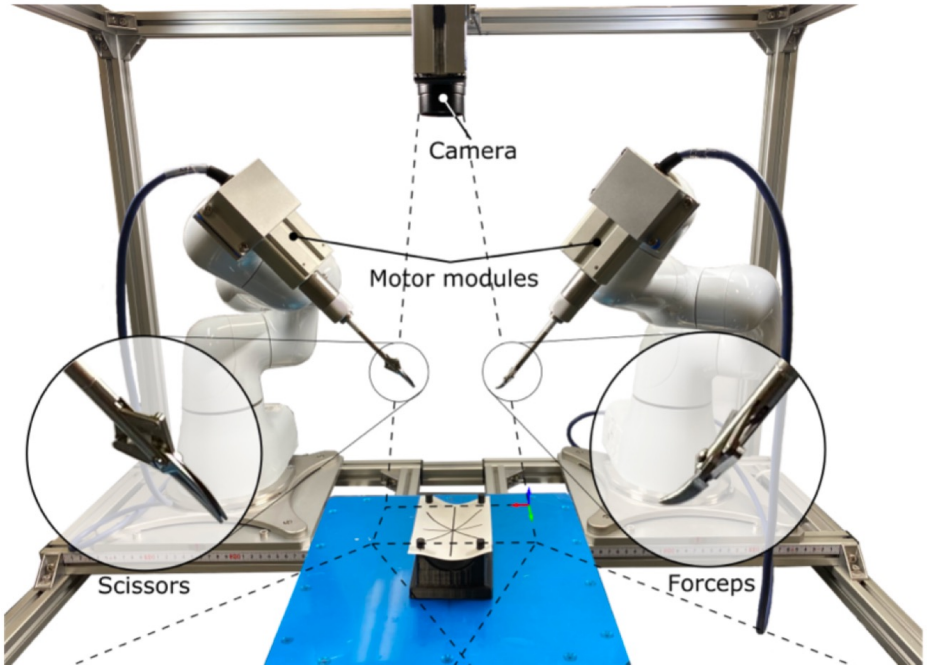
- Two-step magic

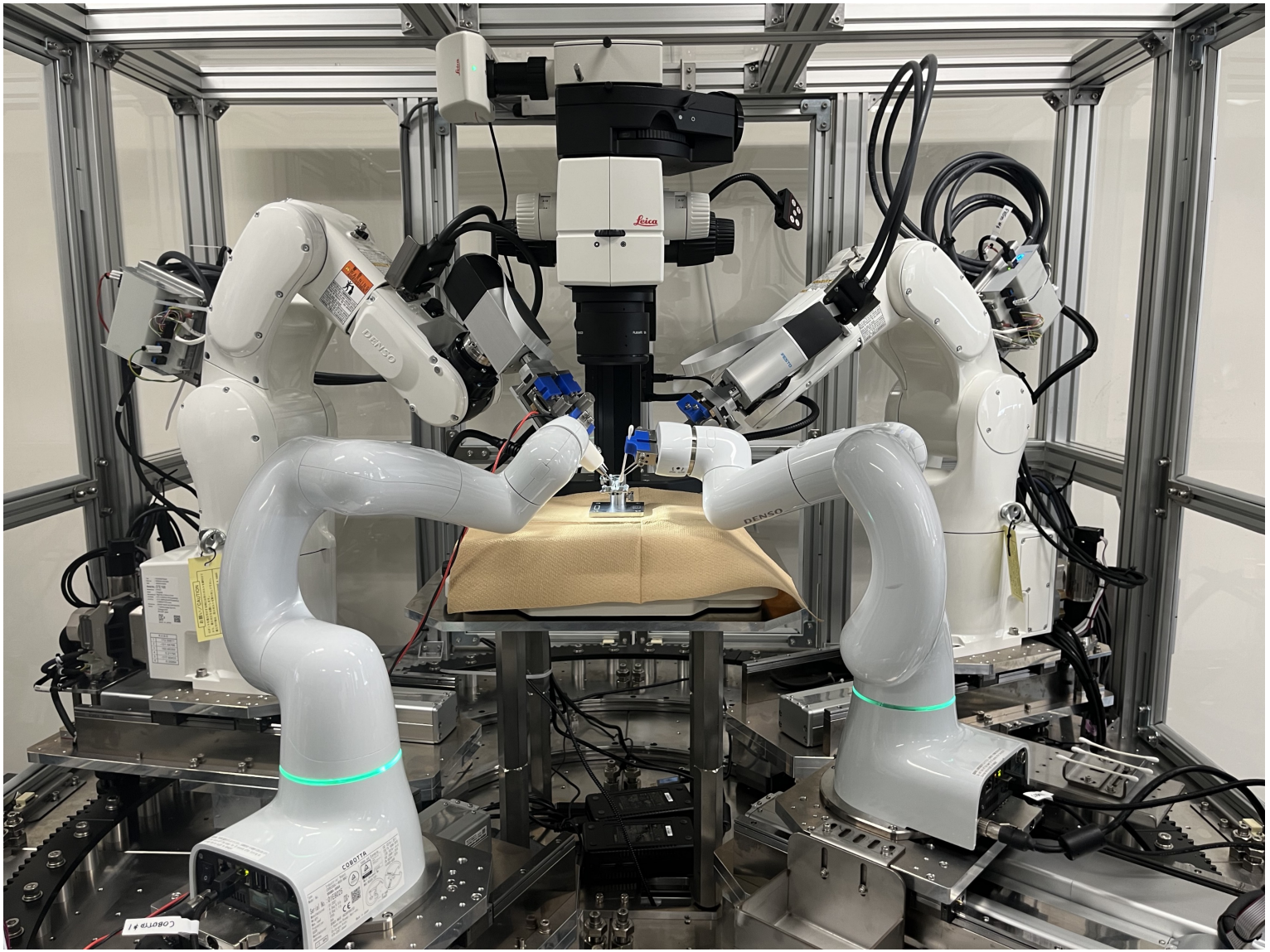
1. Make the driver a subclass of **sas_robot_driver**
2. Done!

- New robot can benefit from kinematic controllers
- New robot can be combined with other robots
- New robot can benefit from teleoperation packages
- And so on

SAS Showcase


SONY


Camera
Motor modules
Scissors
Forceps


DENSO
DENSO
Leica
COBOTTA
COBOTTA #1


KUKA
KUKA
RAIC
Synergy in Action

What is in SAS?

LGPL packages

General purpose	**sas_core**	LGPL	The part of SAS that does not depend on ROS2. Planned to be moved away completely.
	sas_common	LGPL	Common ROS2 code used throughout SAS.
	sas_datalogger	LGPL	Log information at execution time and output as .mat file.
	sas_msgs	LGPL	ROS messages that were made redundant in ROS2.
	sas_conversions	LGPL	Conversions among ROS2 messages, DQ Robotics, Eigen, std::vector etc
	sas_robot_driver	LGPL	Client—Server library for configuration-space monitoring and control. Generic ROS node for any sas_robot_driver subclasses.
	sas_robot_kinematics	LGPL	Client—Server library for task-space monitoring and control.
Robot specific	**sas_robot_driver_ur**	LGPL	A sas_robot_driver implementation for UR's bCap controlled robots
	sas_robot_driver_kuka	LGPL	A sas_robot_driver implementation for KUKA’s FRI controlled robots
	sas_robot_driver_denso	LGPL	A sas_robot_driver implementation for DensoWave's bCap controlled robots

I heard some packages are closed source?

- Unfortunately, people do not respect licenses
  - **LGPL does not give anyone freedom to plagiarize.**
  - Previous [redacted] unspecified individuals were
    - [redacted]
    - Claiming code they had not developed as their own
- Solution
  - Commercializable packages have Non-Commercial licenses
  - Those are distributed only in **binary** format.
  - You can use them if you don't get money from it.
  - Whenever you run it, it tells you the copyright is mine.

Which ones are closed source?

- Source is closed but you can use the binary for Non-Commercial Purposes

sas_operator_side_receiver	NonCommercial	Receive any number of master device data from external PC and expose it to ROS.
sas_patient_side_manager	NonCommercial	Manage master device behaviour, including clutch switching and expose to the SAS Client—Server paradigm.
sas_robot_kinematics_constrained_multiarm	NonCommercial	Centralized control of any number of arms with configurable constraints obtained from CoppeliaSim.
sas_robot_driver_escon	NonCommercial	Possibly commercializable in projects such as ImpACT, Moonshot, and JAXA
sas_robot_driver_aia	NonCommercial	Possibly commercialise as part of the Moonshot multi-arm platform
sas_robot_driver_festo	NonCommercial	

But company!!!
Industry!!!
Amazing!!!

Who would buy this?

Case study: Collaboration with XYZW

- Circa 2019:
  - Collaborative project with XYZW.
  - “We do not need your code; we’ll hire company A.”
- 2019~2021
  - Company A fails miserably. “We’ll hire company B.”
- 2022~2024
  - Company B fails miserably.
  - “Can we please have your code for free?”

Not that easy, was it?!

SAS Principles

MANAGEABLE DEPENDENCIES

Manageable dependencies

- C++17
  - The same as ROS2 [for now at least]
- pybind11
  - For Python bindings of all the C++ code
- Your favorite libraries (only when needed)
  - DQ Robotics, Eigen3, Qt and so on

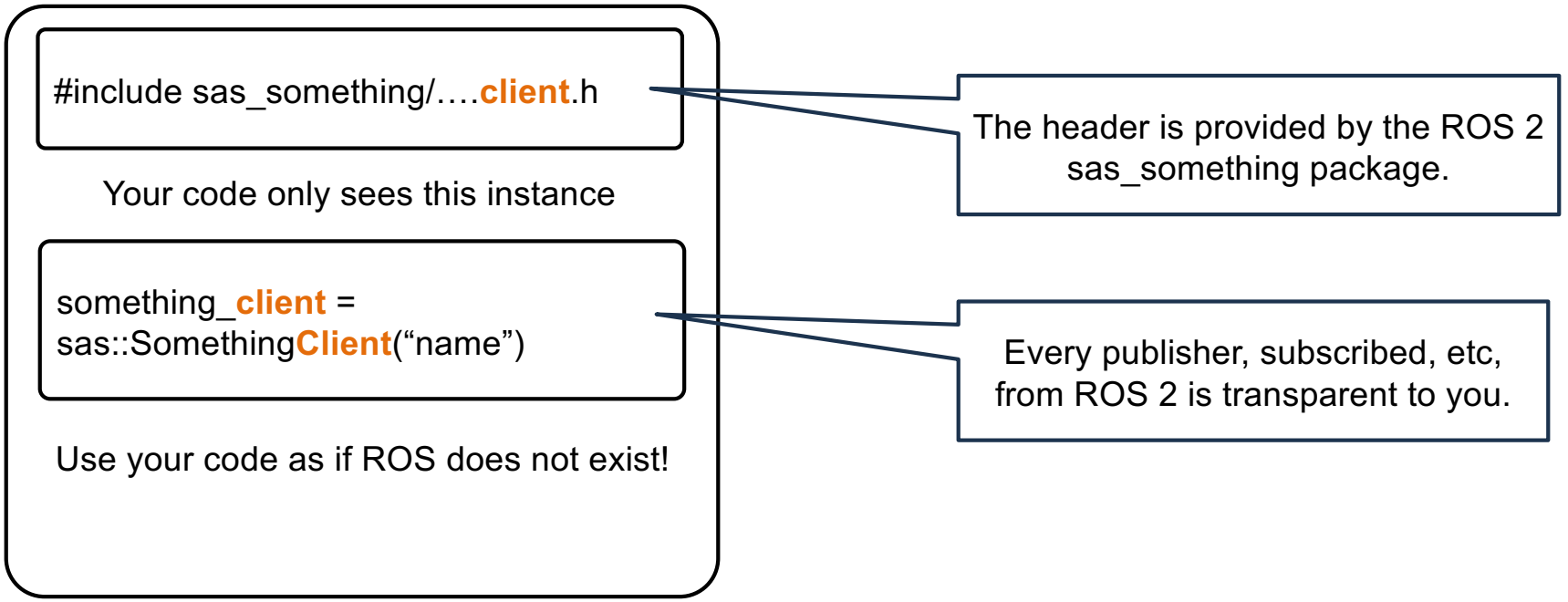
ISOLATE ROS CODE

Isolate ROS Code

- But why?
  - (circa 2010) ROS1 Electric used **rosbuild**
    - Learn
  - (circa 2015) At some point, it switched to **catkin**
    - Learn, rewrite all code
  - Now ROS 2 uses **colcon**
    - Learn, rewrite all code
    - ROS 2 changed every single header to .hpp, plus other changes.
- Lesson
  - **Only what needeth ROS shall dependeth on ROS.**

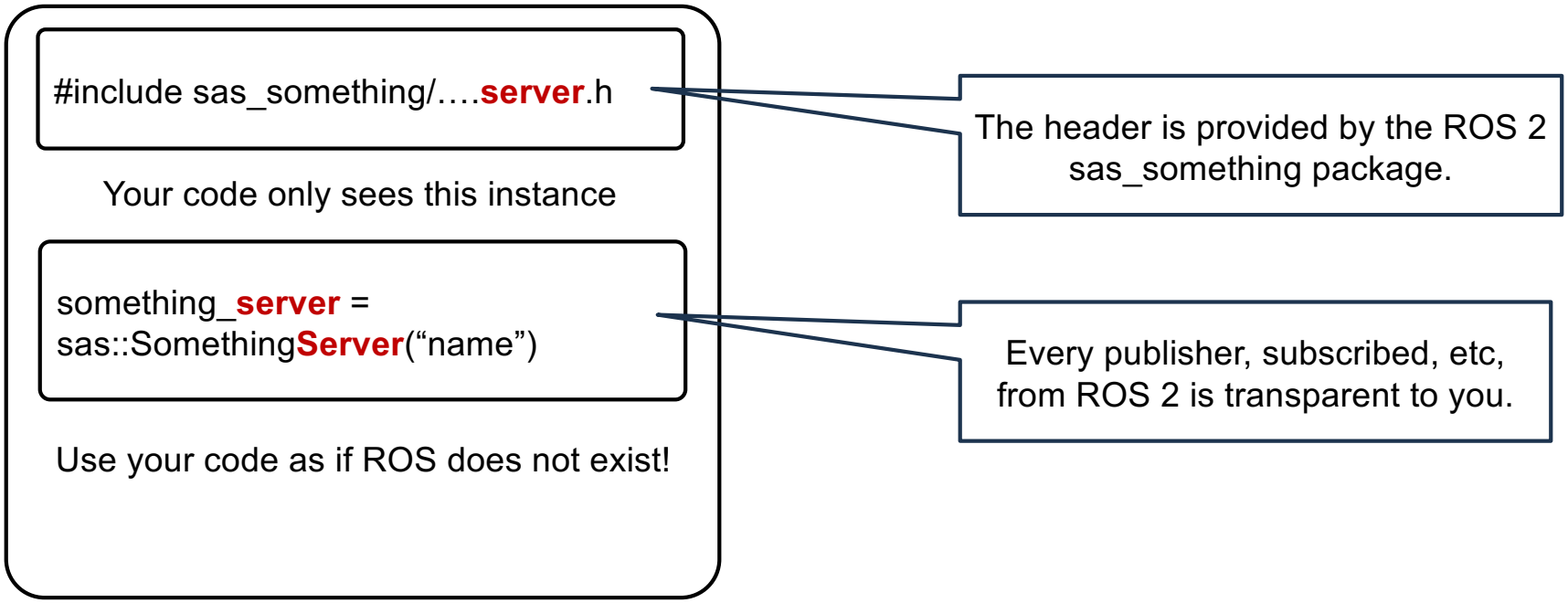
Client—Server Topology

- Client

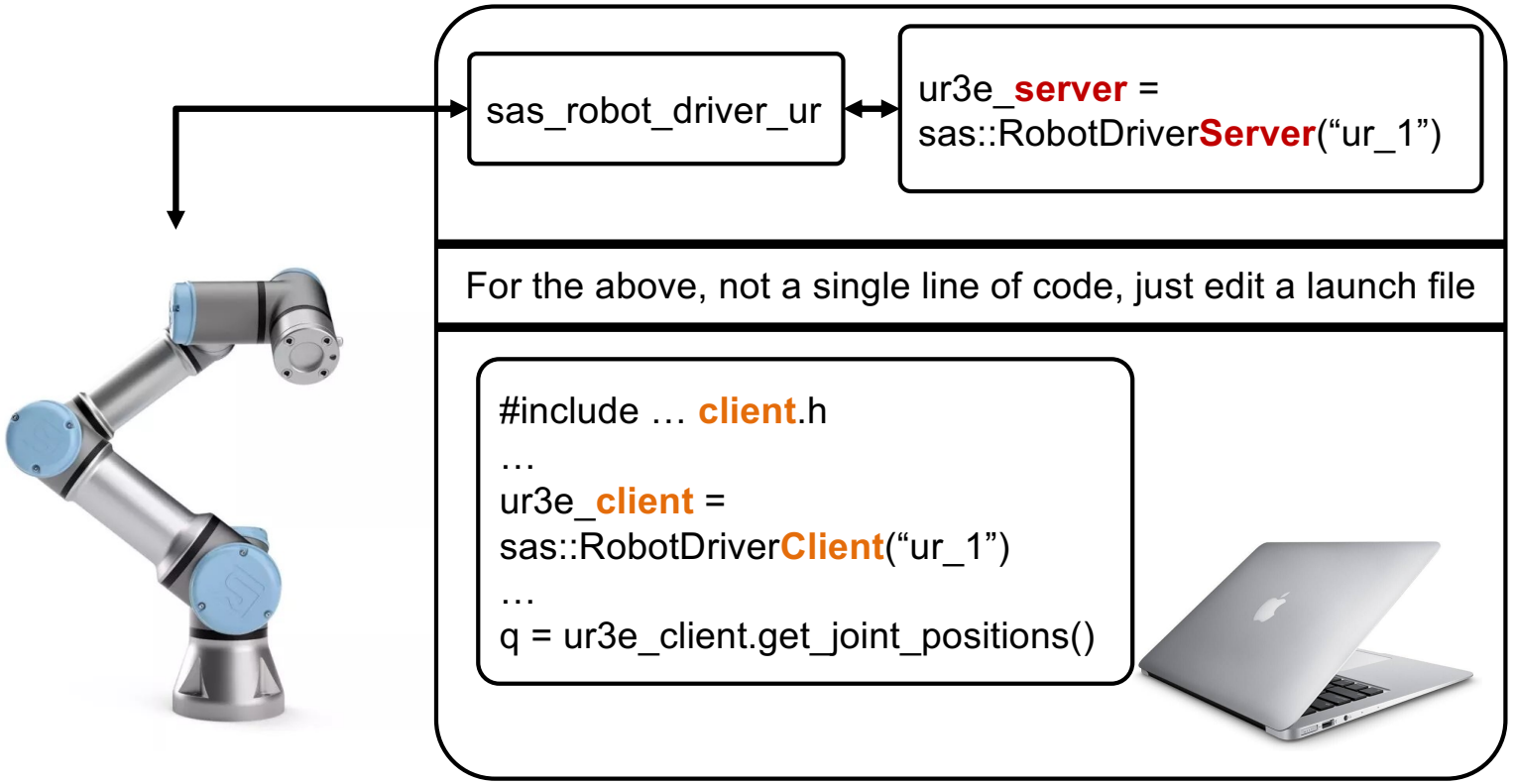


Client—Server Topology

- Server



Server—Client Example C++



Server—Client Example Python

sas_robot_driver_ur

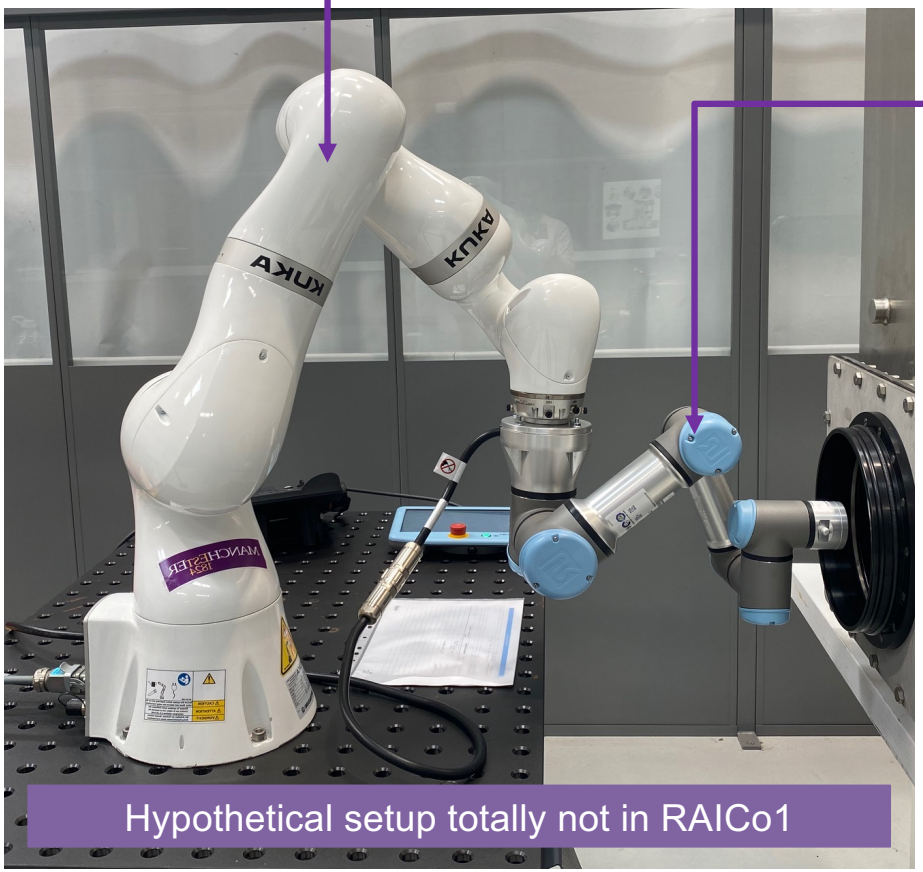
ur3e_**server** =
sas::RobotDriver**Server**(“ur_1”)

For the above, not a single line of code, just edit a launch file

```
from … import RobotDriverClient
…
ur3e_client =
RobotDriverClient(“ur_1”)
…
q = ur3e_client.get_joint_positions()
```

SEAMLESS SERIAL COMBINATION

SAS ROS Composer



sas_robot_driver_ur

sas_robot_driver_kuka

sas_robot_driver_ros_composer_node

For the above, not a single line of code, just edit a launch file

```
longboy_client =
RobotDriverClient("longboy_1")
…
q = longboy_client.get_joint_positions()
```

Summary

MANCHESTER 1824
The University of Manchester

Summary

- SmartArmStack for research sustainability
- The company’s ROS drivers are rubbish
- Some cool robots use SmartArmStack
- LGPL licensed for most
- Manageable dependencies
- ROS code isolated
- Seamless serial combination