

# A ROS based architecture for an autonomous chemistry laboratory

David Marquez-Gamez

Leverhulme Research  
Centre for Functional  
Materials Design



UNIVERSITY OF  
LIVERPOOL

# What is this talk about?

“Developing a self-driving autonomous scientific laboratory powered by artificial intelligence and robotics.”

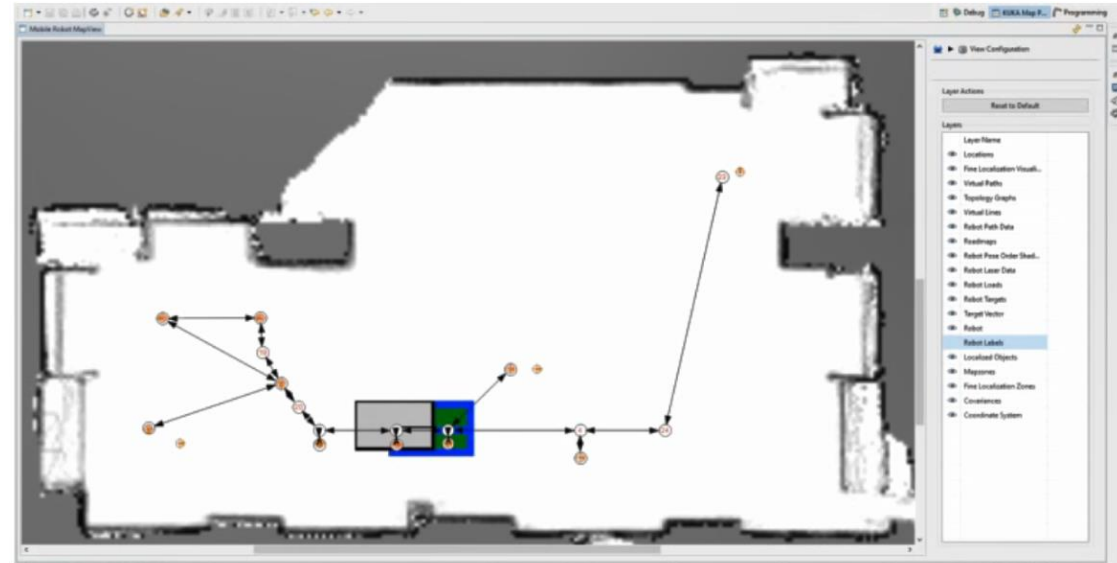
*Cooperative, heterogeneous and adaptive robots for an autonomous chemistry laboratory: recent advances and open challenges towards the scientific lab of the future*

# Motivation: Scientists at work in laboratory



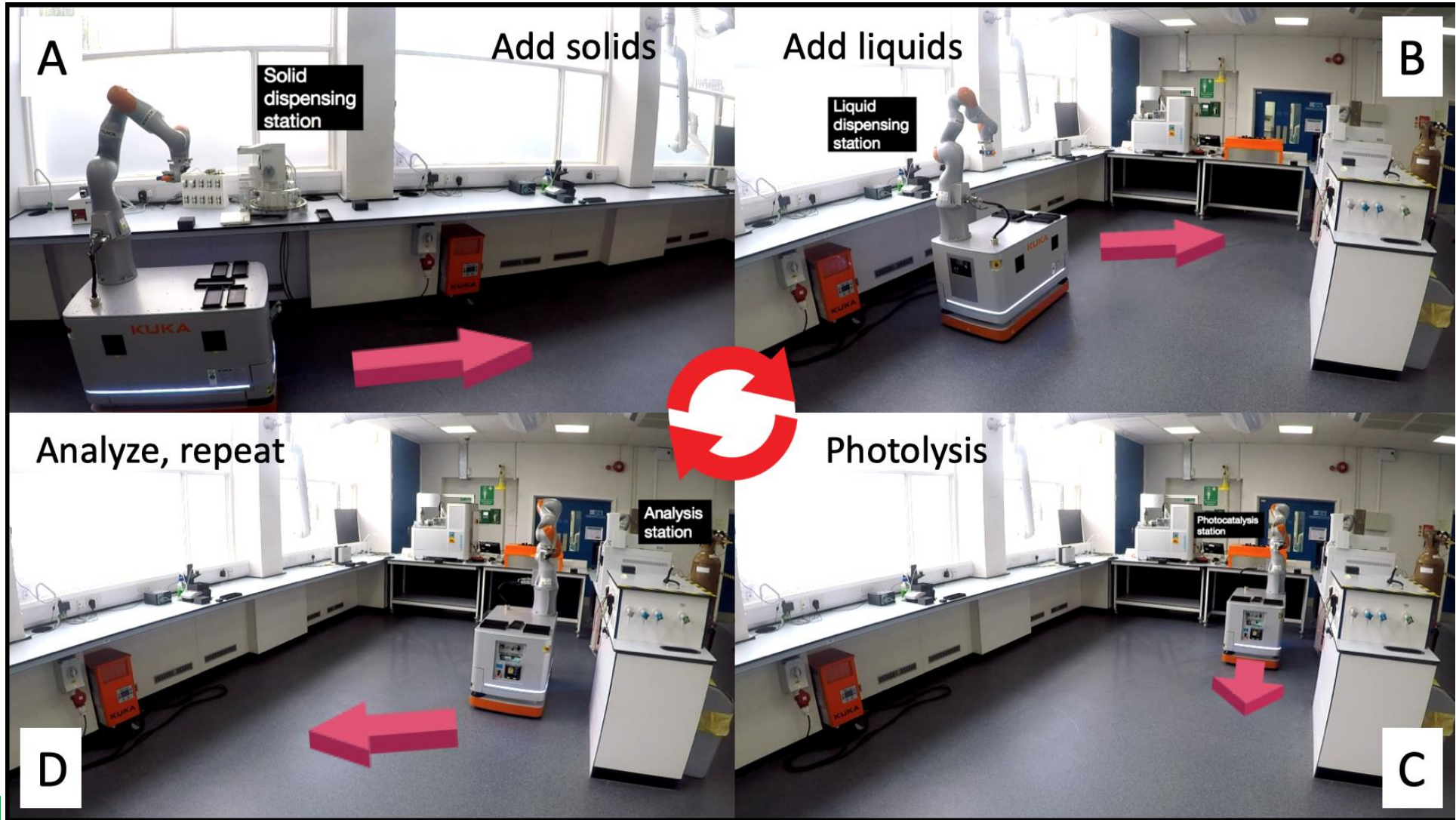
<https://youtu.be/X-HyirHulo>

- Autonomous mobile robot scientist
- Dexterous - can work with equipment / scales that are relevant to scientific labs



# Robotic Scientist

Autonomous mobile robot - Autonomous discovery

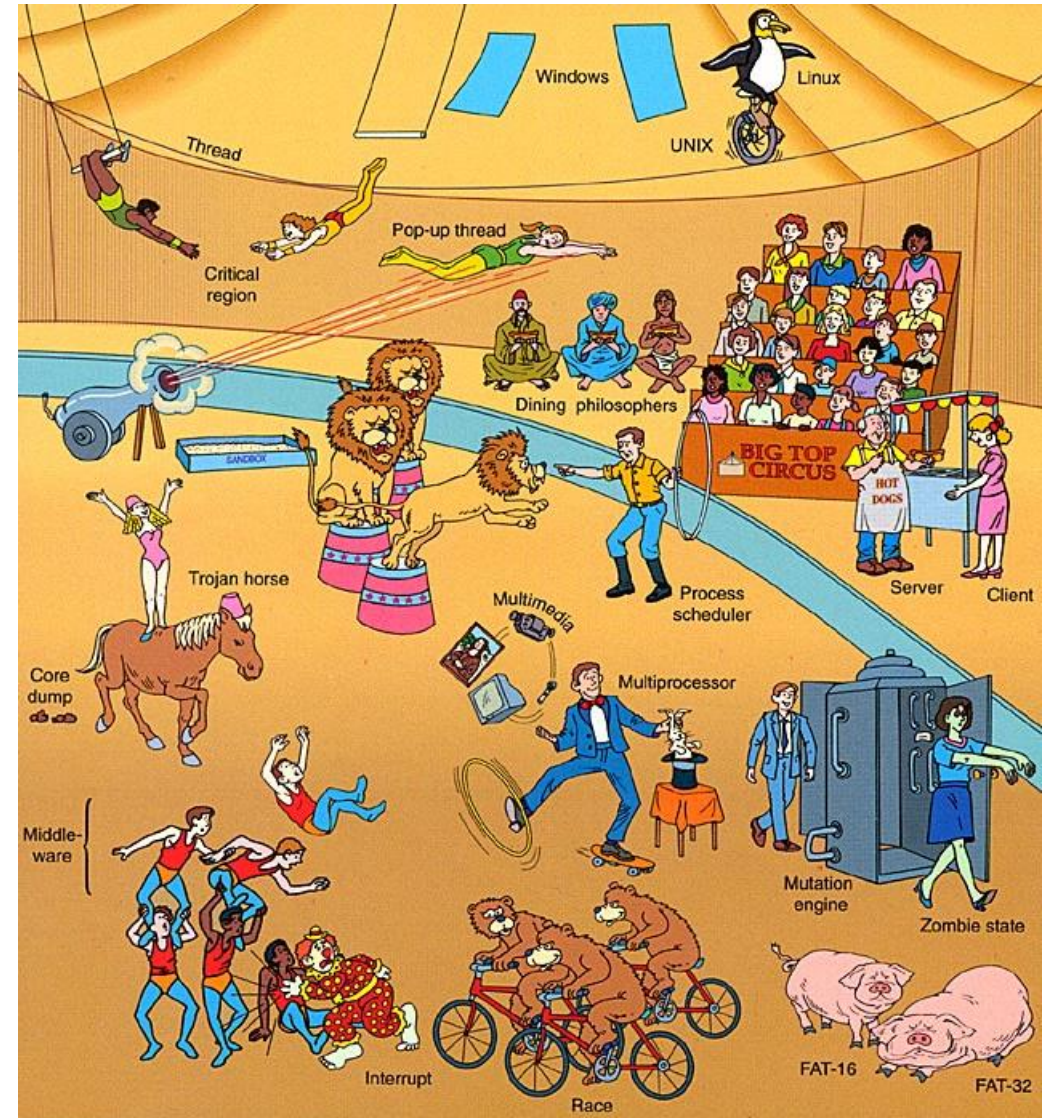


# Self-driving Autonomous Lab: Communication “Middleware”

## Robot Operating System (ROS)



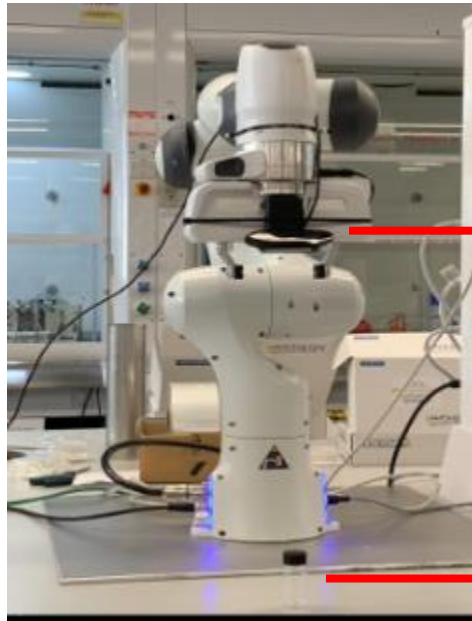
Goal:  
abstract away interprocess communication (IPC) and cross-network communication details



# Self-driving Autonomous Lab: challenges

ROS

Challenge: perception  
OpenCV



2D camera

Vial

<https://youtu.be/rdPkkdZ7leE>

```

1 #!/usr/bin/env python
2 import rospy
3 from sensor_msgs.msg import Image
4 import cv2
5 from cv_bridge import CvBridge, CvBridgeError
6 rospy.loginfo('sensor_msgs::Image')
7 bridge = CvBridge()
8
9 # Define a function to show the image in an OpenCV window
10 def show_image(img):
11     cv2.imshow("Image Window", img)
12     cv2.waitKey(1)
13
14 def img_callback(msg):
15     # Try to convert the ROS image message to a CV2 image
16     try:
17         cv_image = bridge.imgmsg_to_cv2(msg, 'bgr8')
18     except CvBridgeError as e:
19         rospy.logerr('CvBridge Error: (%s)' % str(e))
20         return
21     color_image = cv_image
22
23     # Convert the image to color image
24     img = cv2.cvtColor(color_image, cv2.COLOR_BGR2RGB)
25     gray_img = cv2.cvtColor(color_image, cv2.COLOR_BGR2GRAY)
26     img = cv2.medianBlur(gray_img, 5)
27     circles = cv2.HoughCircles(img, cv2.HOUGH_GRABBY, 1.4,
28                             param1=100, param2=100, minRadius=20, maxRadius=30)
29     numbers = circles.shape
30     print('Number of circles: %d' % numbers)
31     for i in range(1, numbers):
32         (x, y, radius) = circles[i, 0]
33         cv2.circle(img, (int(x), int(y)), int(radius), (0, 255, 0), 2)
34         cv2.circle(img, (int(x), int(y)), int(radius), (0, 255, 0), 1)
35         print('Circle: %s' % str((x, y, radius)))
36
37     show_image(img)
38     rospy.loginfo('Image')
39
40 # Main function
41 def main():
42     rospy.init_node('img_callback')
43     sub = rospy.Subscriber('robot/camera_raw', Image, img_callback)
44     rospy.spin()
45
46 if __name__ == '__main__':
47     main()

```

<https://youtu.be/4HAKkNNp9FE>

```

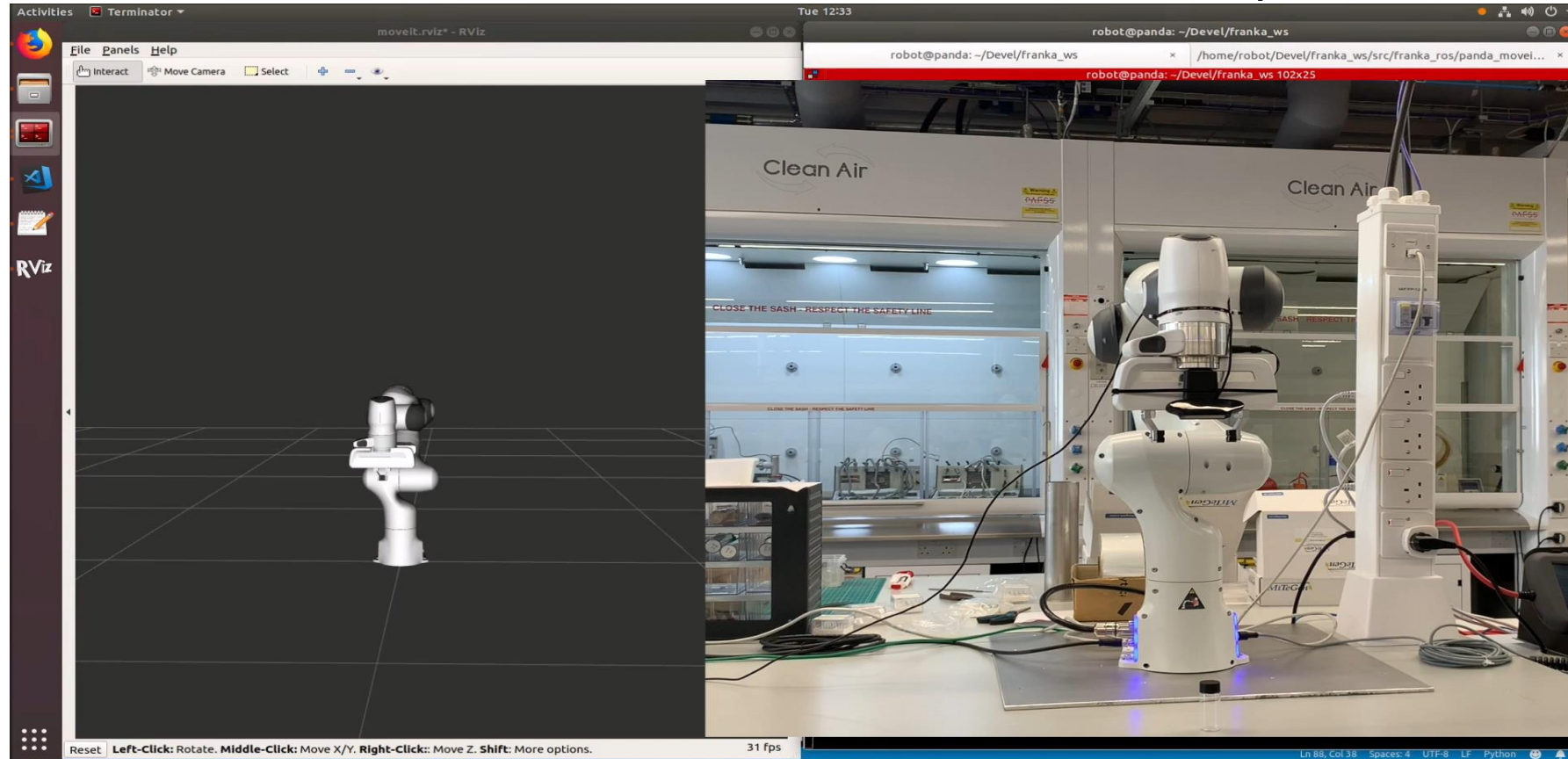
1 #!/usr/bin/env python
2 import rospy
3 from sensor_msgs.msg import Image
4 import cv2
5 from cv_bridge import CvBridge, CvBridgeError
6 rospy.loginfo('sensor_msgs::Image')
7 bridge = CvBridge()
8
9 # Define a function to show the image in an OpenCV window
10 def show_image(img):
11     cv2.imshow("Image Window", img)
12     cv2.waitKey(1)
13
14 def img_callback(msg):
15     # Try to convert the ROS image message to a CV2 image
16     try:
17         cv_image = bridge.imgmsg_to_cv2(msg, 'bgr8')
18     except CvBridgeError as e:
19         rospy.logerr('CvBridge Error: (%s)' % str(e))
20         return
21     color_image = cv_image
22
23     # Convert the image to color image
24     img = cv2.cvtColor(color_image, cv2.COLOR_BGR2RGB)
25     gray_img = cv2.cvtColor(color_image, cv2.COLOR_BGR2GRAY)
26     img = cv2.medianBlur(gray_img, 5)
27     circles = cv2.HoughCircles(img, cv2.HOUGH_GRABBY, 1.4,
28                             param1=100, param2=100, minRadius=20, maxRadius=30)
29     numbers = circles.shape
30     print('Number of circles: %d' % numbers)
31     for i in range(1, numbers):
32         (x, y, radius) = circles[i, 0]
33         cv2.circle(img, (int(x), int(y)), int(radius), (0, 255, 0), 2)
34         cv2.circle(img, (int(x), int(y)), int(radius), (0, 255, 0), 1)
35         print('Circle: %s' % str((x, y, radius)))
36
37     show_image(img)
38     rospy.loginfo('Image')
39
40 # Main function
41 def main():
42     rospy.init_node('img_callback')
43     sub = rospy.Subscriber('robot/camera_raw', Image, img_callback)
44     rospy.spin()
45
46 if __name__ == '__main__':
47     main()

```

Challenge: Perception and control (Visual servoing)  
OpenCV + MoveIt!



ROS



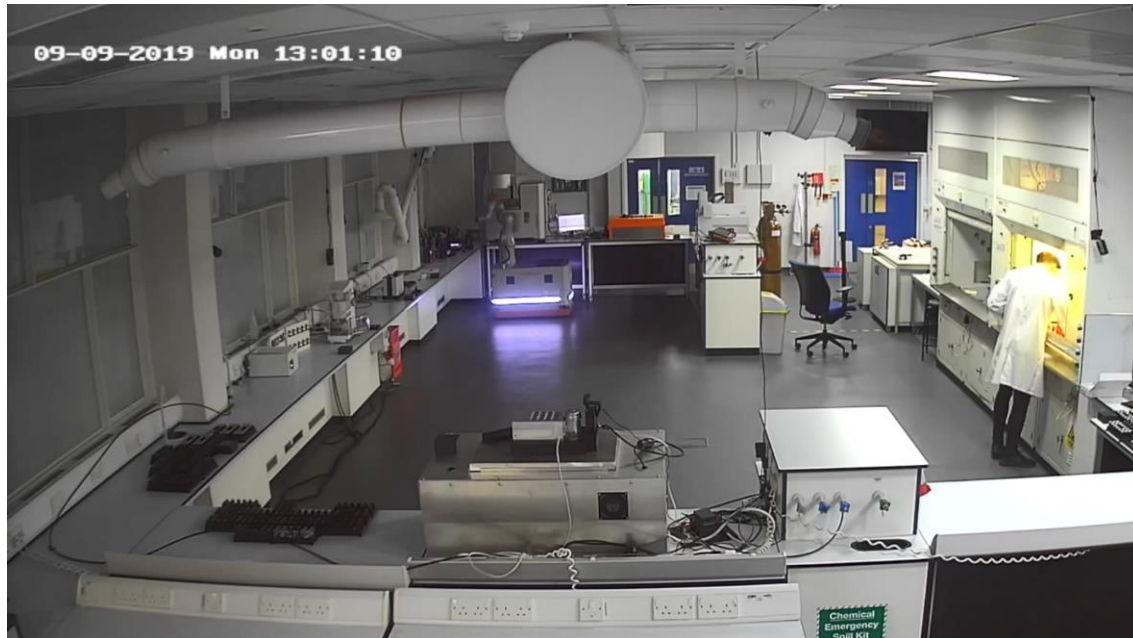
<https://youtu.be/iwKtckYdE2M>





Challenge: Robust autonomous navigation -> Navigation stack

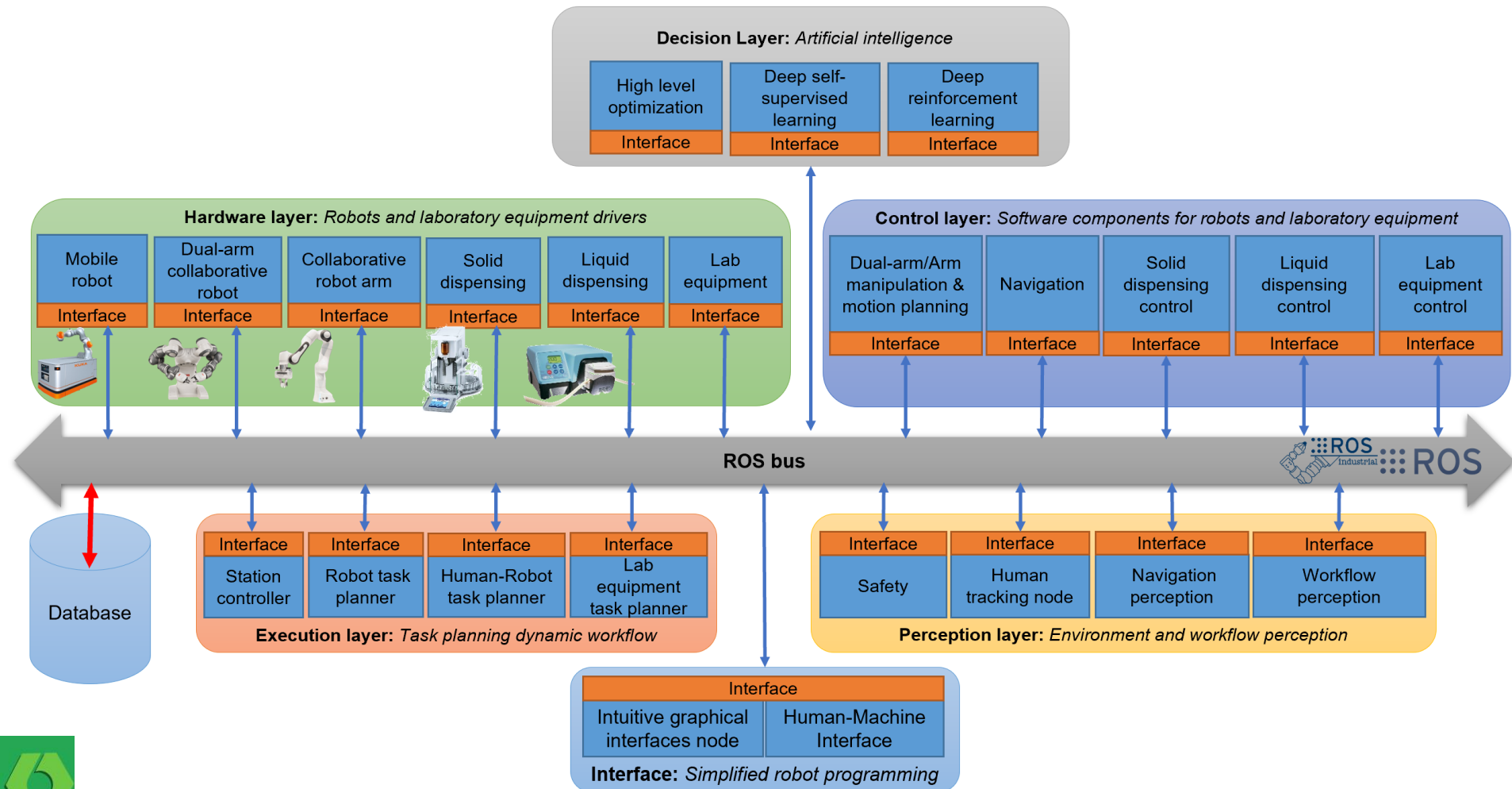
Need for adaptive and *intelligent navigation*



## Dynamically reconfigurable workflows – Modular and Flexible systems

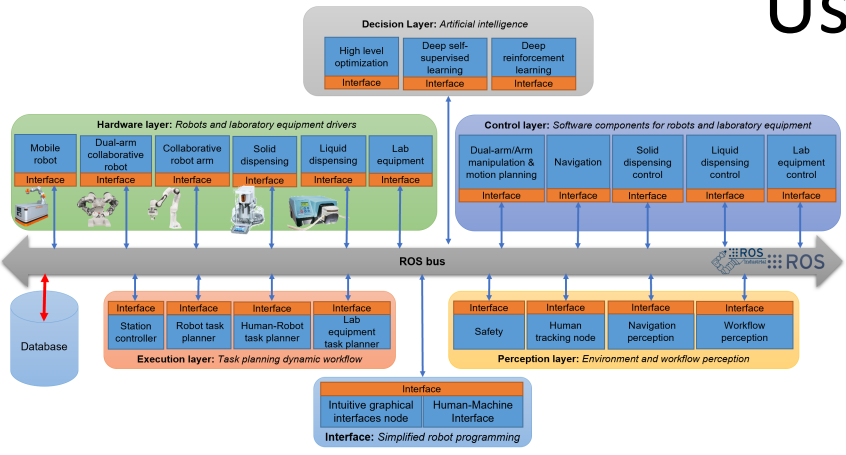
- Simplification of the integration and networking of the control and sensor data utilizing web based, and ontology services
- Use of common integration architecture to monitor the execution of the task and dynamically redistribute the workflow
- Using the status reporting of the autonomous robots, the system will be able to generate alternative allocations for robots (and humans).

## Dynamically reconfigurable workflows – Modular and Flexible systems - Modular architecture



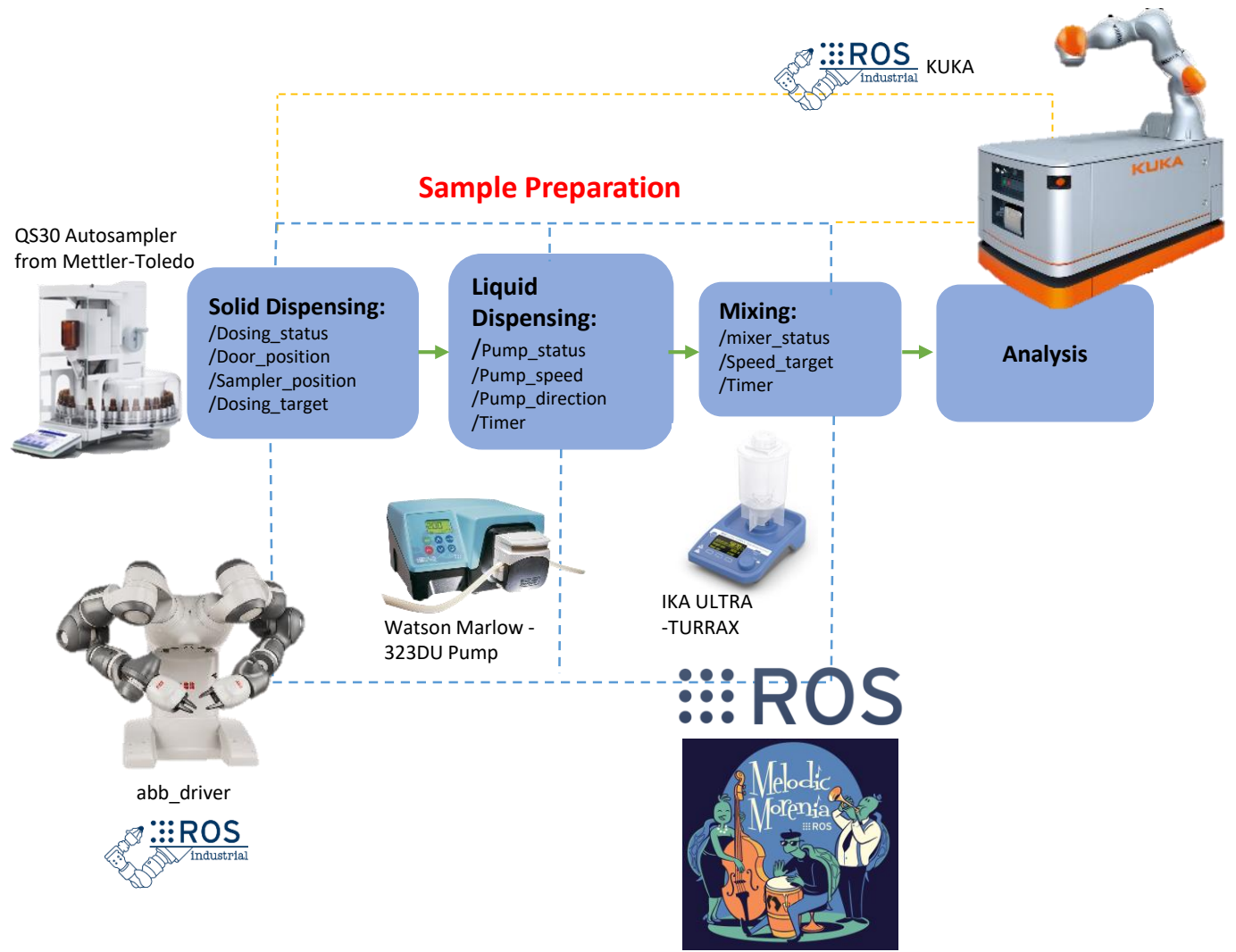
# Robot Architecture:

## Use Case: sample preparation



Dynamically reconfigurable workflows

Modular and Flexible systems



# Robot Architecture: Lab automation/interfacing

ROS



## ROS package

QS30 Autosampler  
from Mettler-Toledo



**Solid Dispensing:**  
/Dosing\_status  
/Door\_position  
/Sampler\_position  
/Dosing\_target

Watson Marlow -  
323DU Pump



**Liquid Dispensing:**  
/Pump\_status  
/Pump\_speed  
/Pump\_direction  
/Timer

IKA ULTRA -TURRAX




**Mixing:**  
/mixer\_status  
/Speed\_target  
/Timer

## RS232 - Serial Interface

**solid**


QS30 Autosampler  
from Mettler-Toledo



**Solid Dispensing:**  
/Dosing\_status  
/Door\_position  
/Sampler\_position  
/Dosing\_target

**liquid**


Watson Marlow -  
323DU Pump



**Liquid Dispensing:**  
/Pump\_status  
/Pump\_speed  
/Pump\_direction  
/Timer

**mixing**

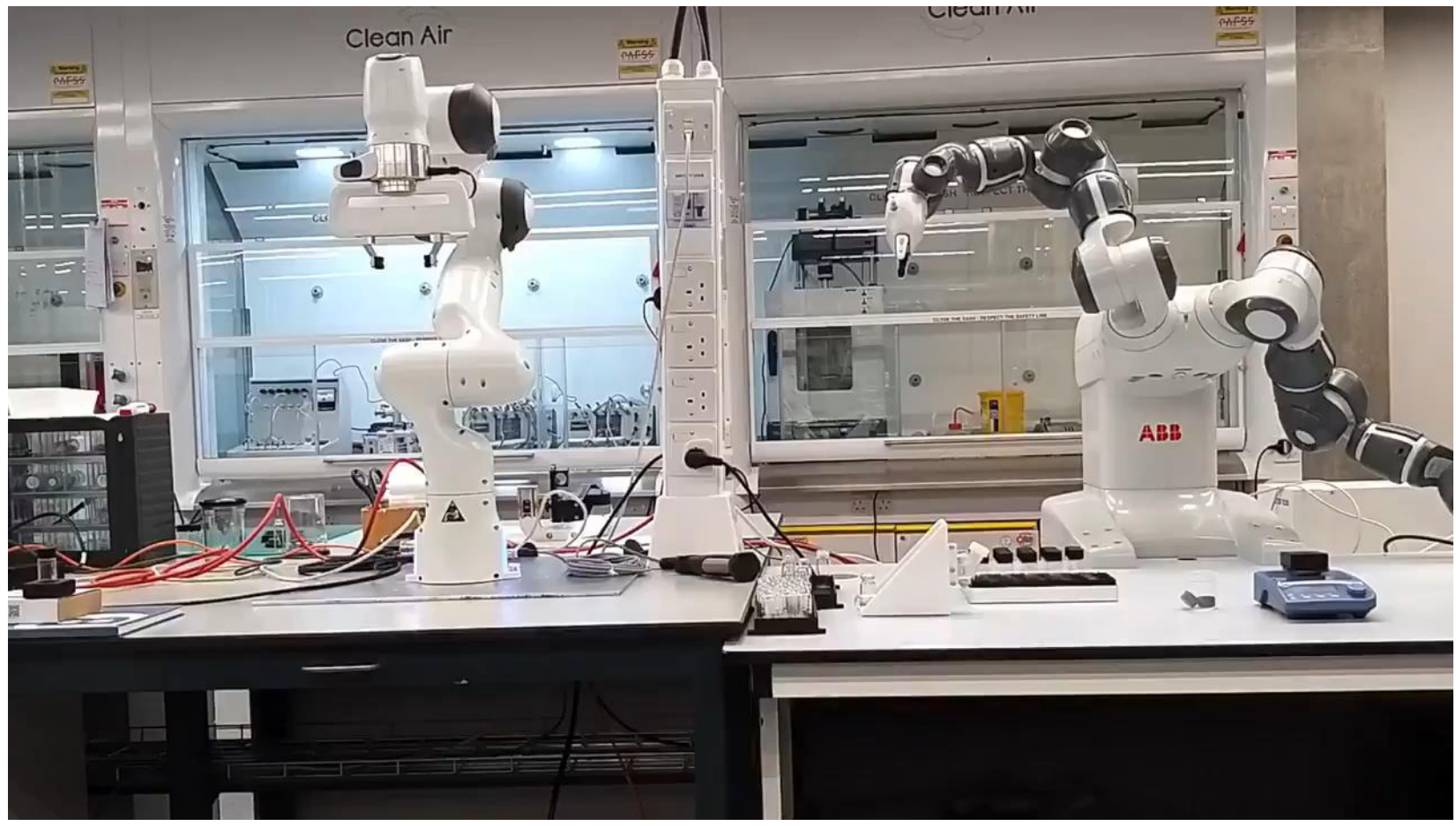
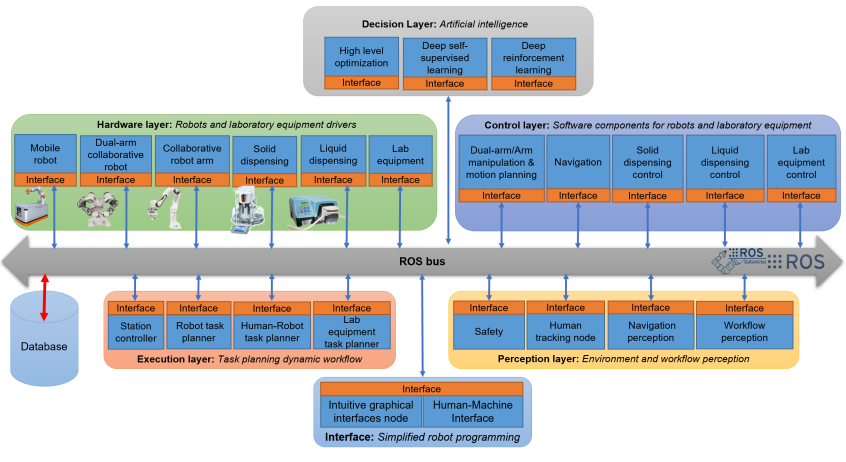
IKA ULTRA -TURRAX



**Mixing:**  
/mixer\_status  
/Speed\_target  
/Timer

# Robot Architecture

Heterogenous and collaborative robots  
Collaborative task: vial capper/decapper and transfer



Dynamically reconfigurable  
workflows



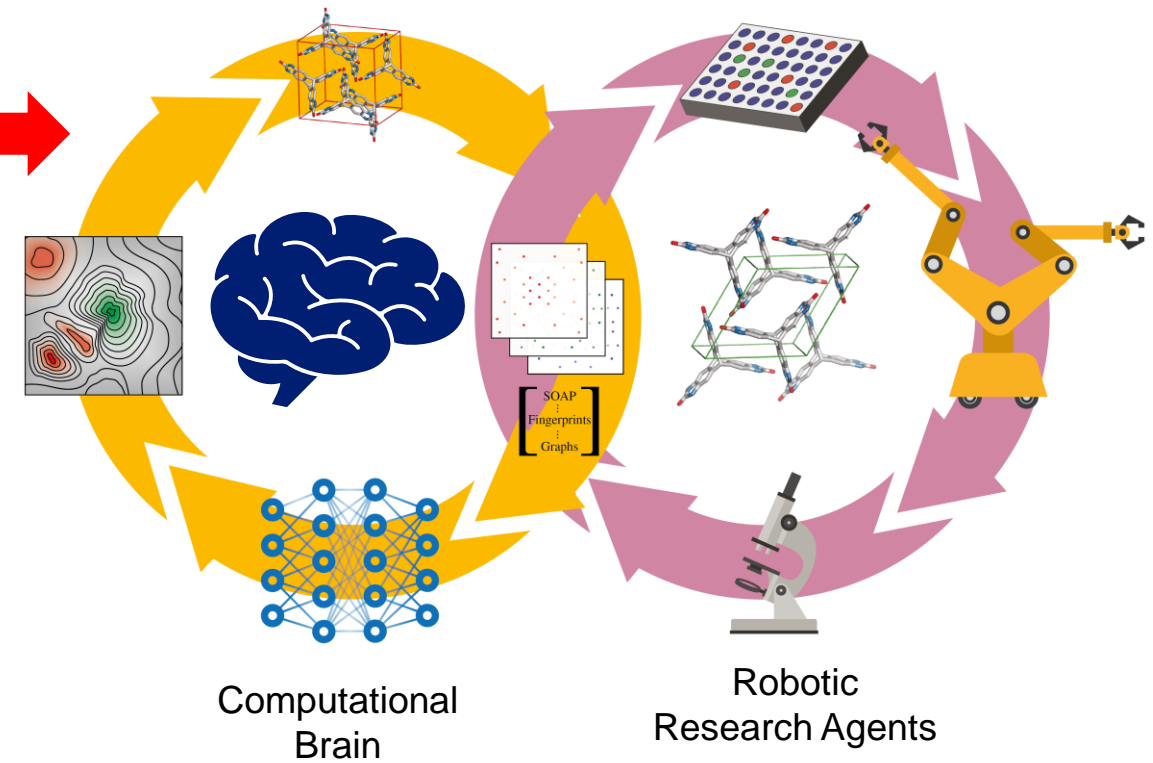
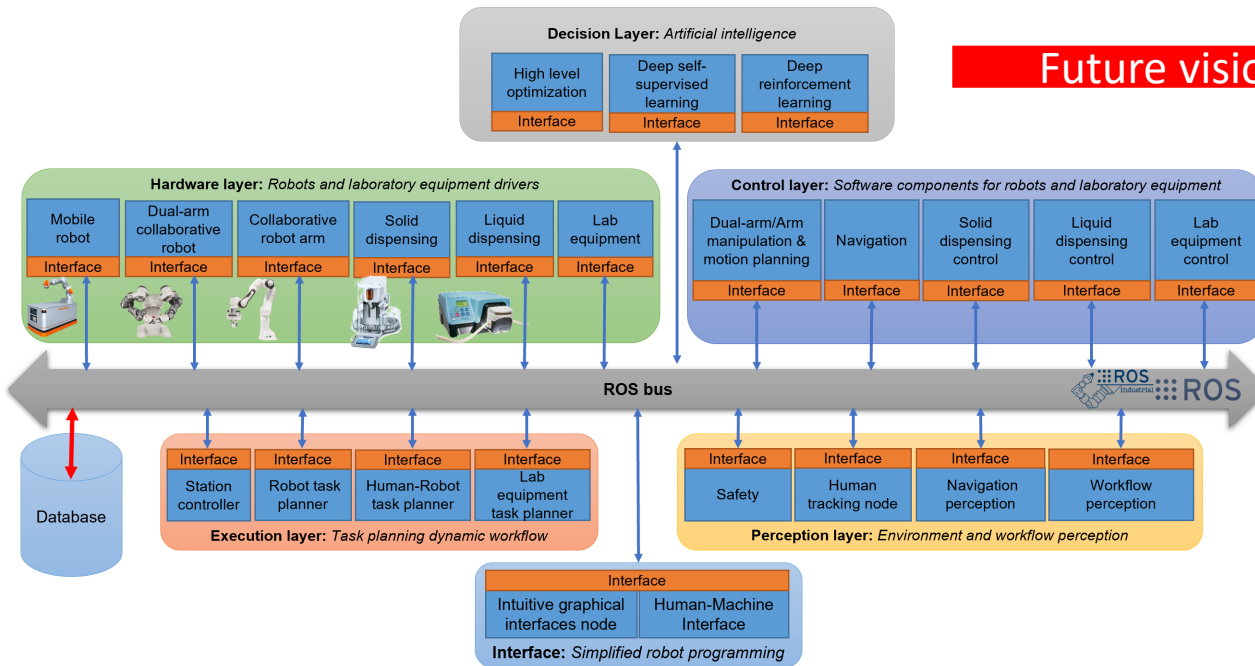
Modular and Flexible systems

ROS



<https://youtu.be/tgWzPMPSeOM>

# Autonomous and Intelligent Labs: Future Vision



ROS

# Thank you

## Let's Make Robots!!!



[dmarquez@liverpool.ac.uk](mailto:dmarquez@liverpool.ac.uk)



@damarquezg