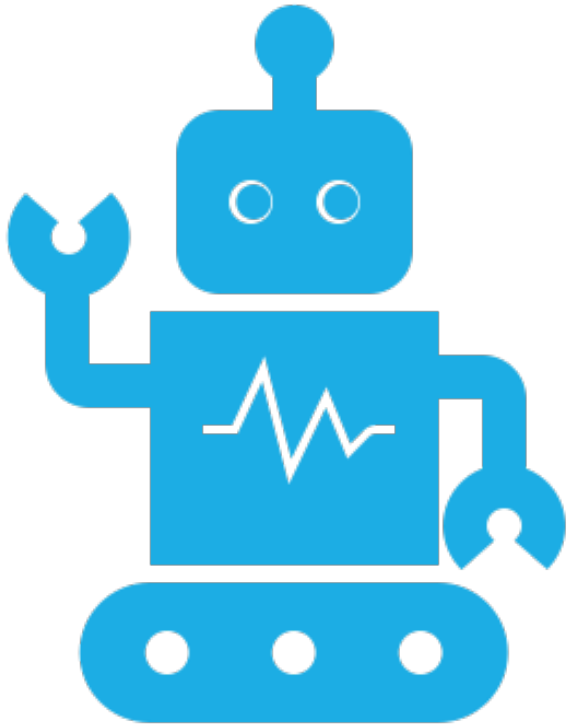
# Lecture 7
# Robot Software (PEPPER)

Samuele Vinanzi

Department of Computer Science, Kilburn Building,

The University of Manchester



# CODING FOR HUMANOID ROBOTS

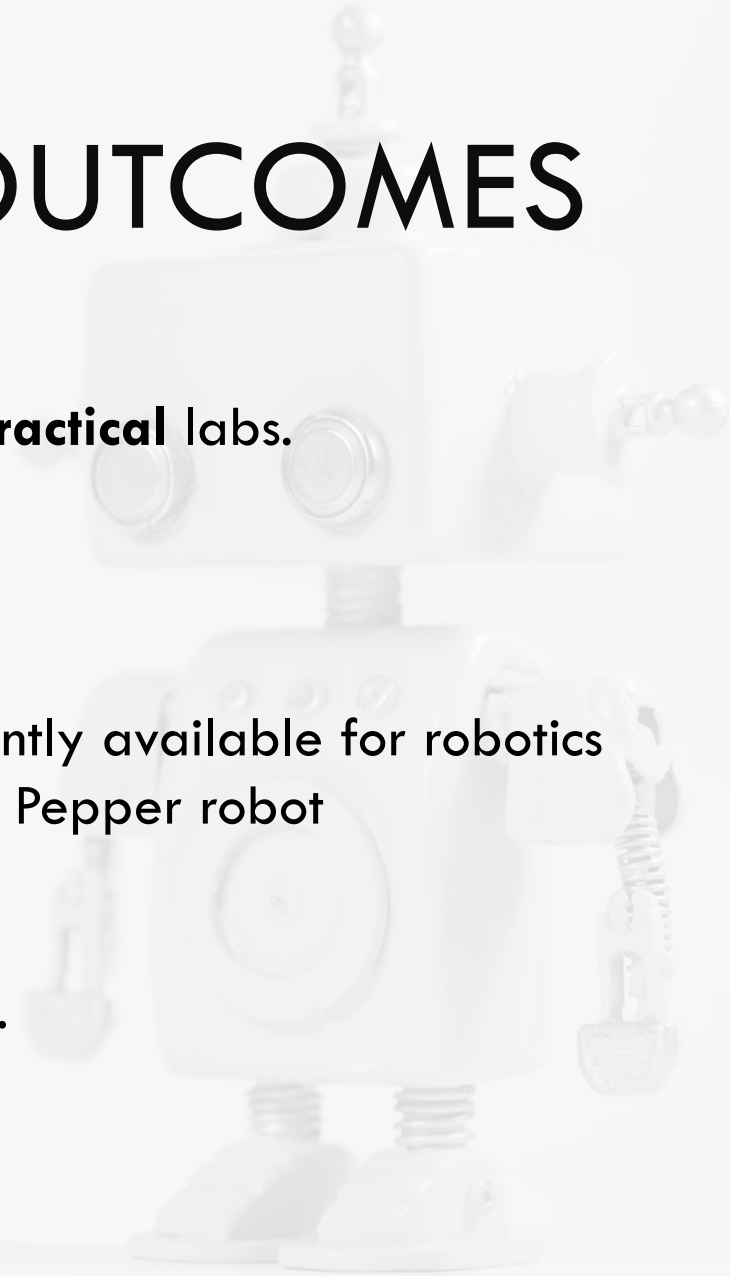
# EXPECTED LEARNING OUTCOMES

One 1-hour **technical** lecture followed by two 2-hours **practical** labs.

What are you going to learn from them?

- Anatomy of a common humanoid robot
- What is a *middleware* and which libraries are currently available for robotics
- How to use Choregraphe to graphically program a Pepper robot
- How to code for Pepper using Python and NAOqi

You are expected to develop **hands-on** skills in robotics.

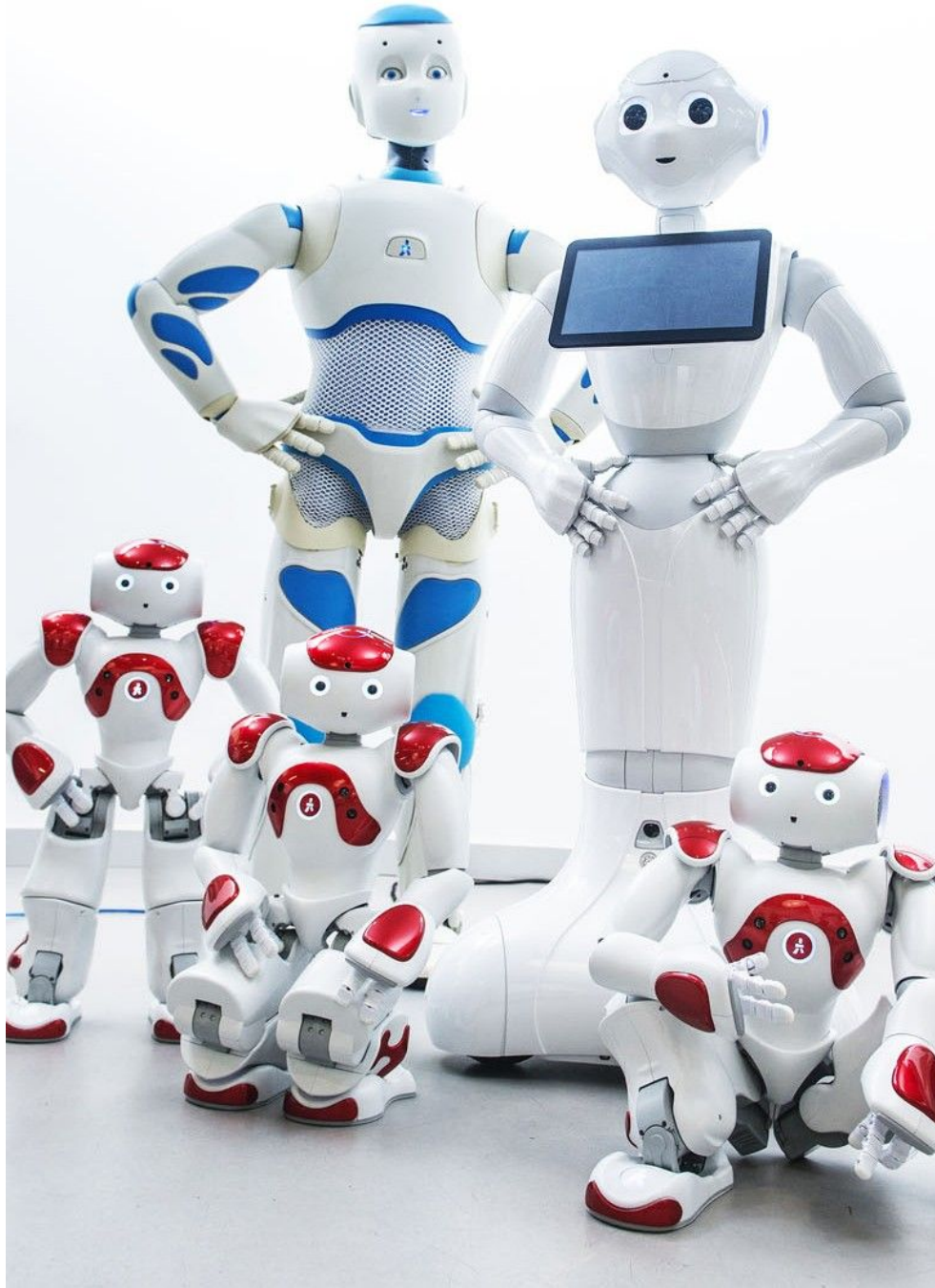


# WHY HUMANOID ROBOTS?

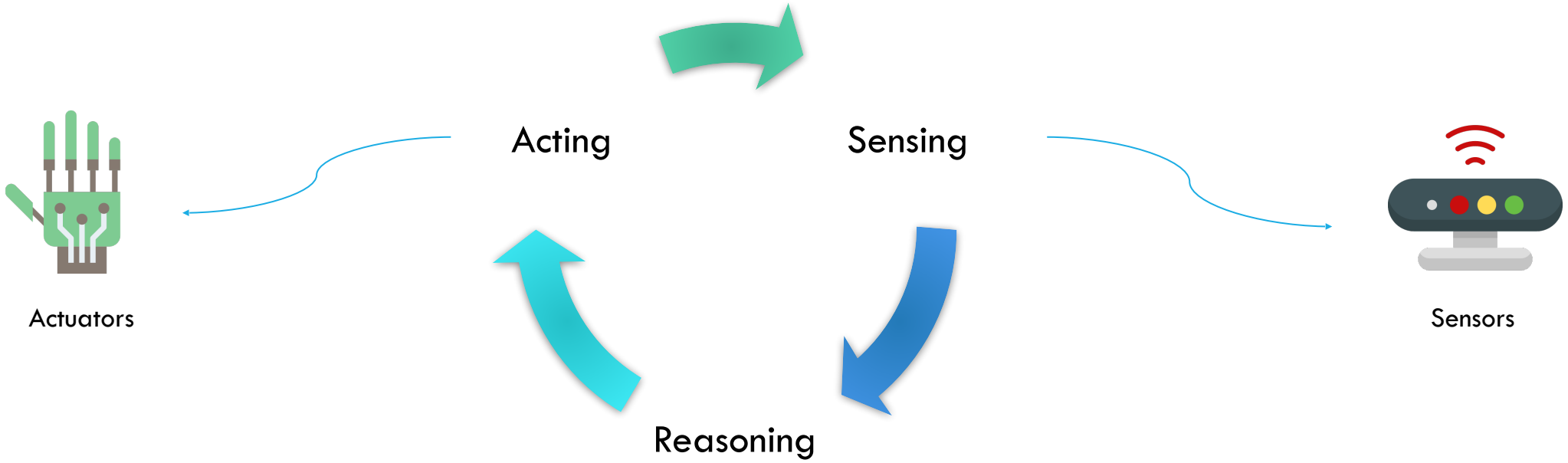
A humanoid robot is a machine that resembles the human body structure and capabilities.

But… the human body is complex!

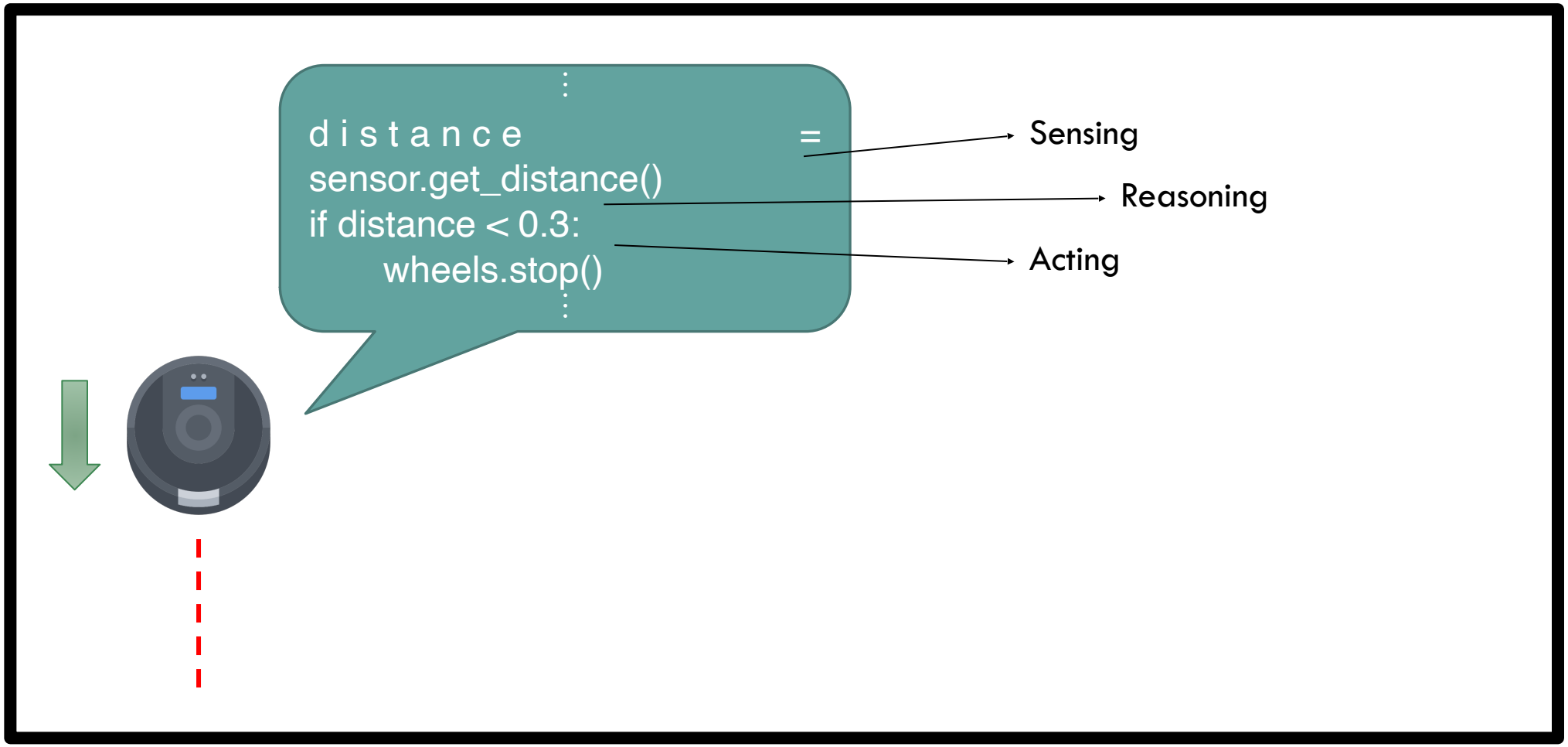
So why do we bother?



# AUTONOMOUS ROBOT WORKFLOW



# AUTONOMOUS ROBOT WORKFLOW

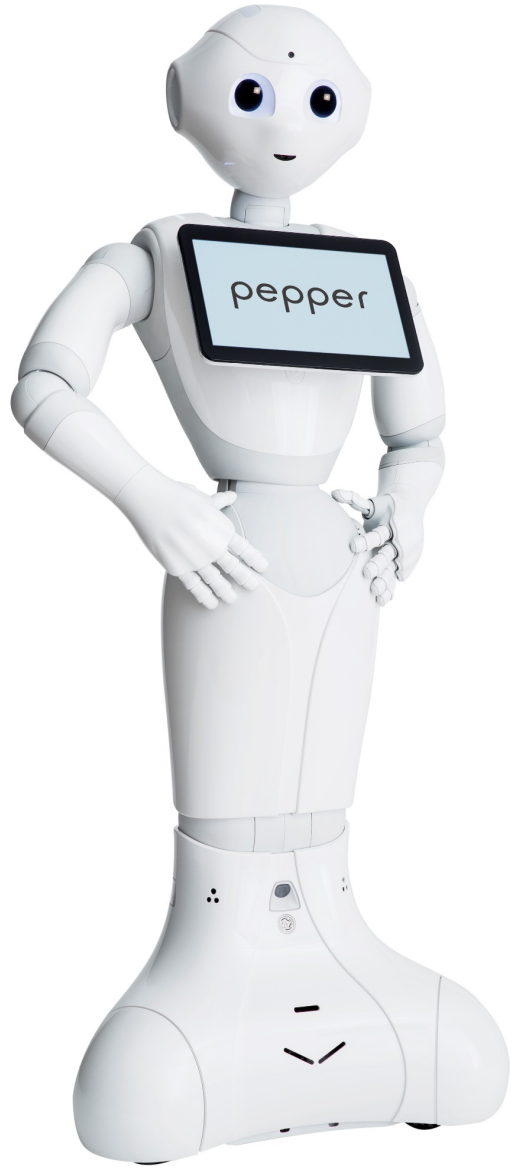


# SOFTBANK PEPPER ROBOT

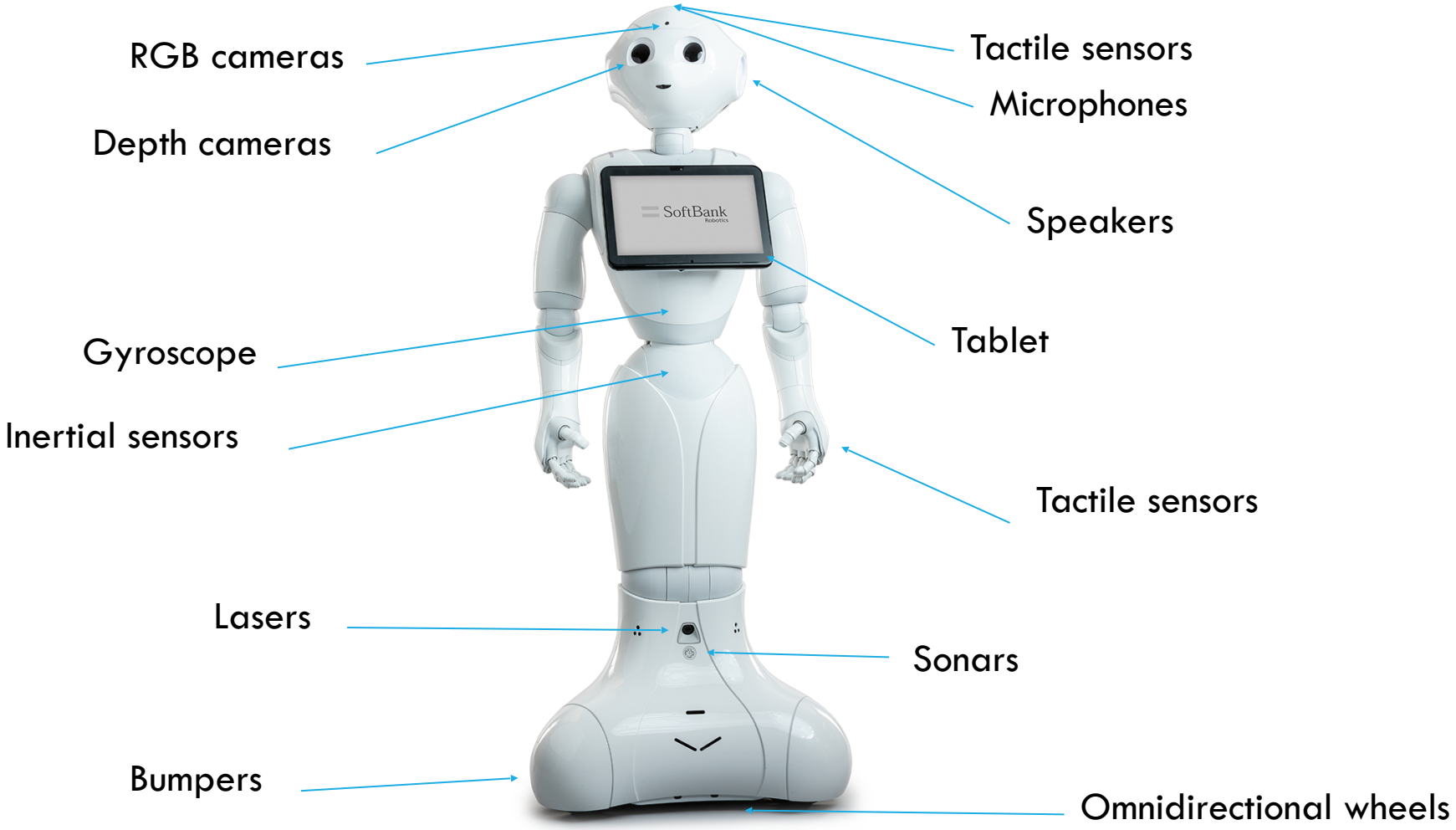
Manufactured in 2014 by SoftBank (formerly Aldebaran), a Japanese company.

«Pepper is intended to make people enjoy life, enhance people's lives, facilitate relationships, entertain and connect people with the outside world»

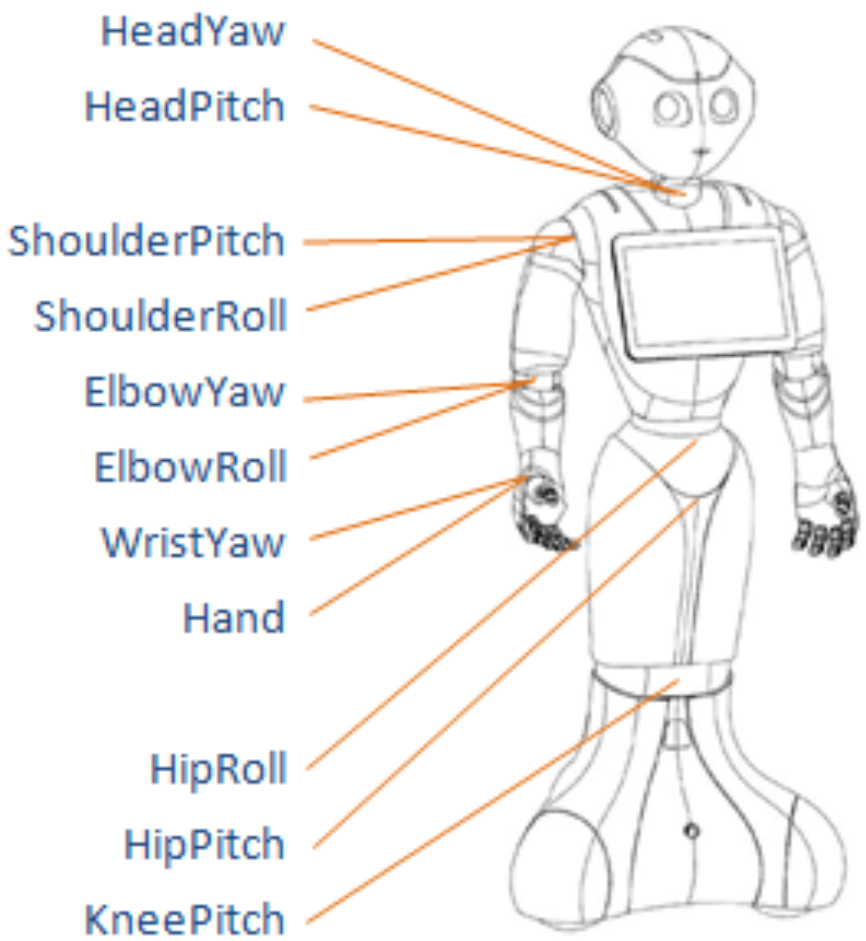
| | |
|---|---|
| PROCESSOR | Atom E3845 |
| CPU | Quad core |
| Clock speed | 1.91 GHz |
| RAM | 4 GB DDR3 |
| GPU | Intel HD graphics 792 MHz |

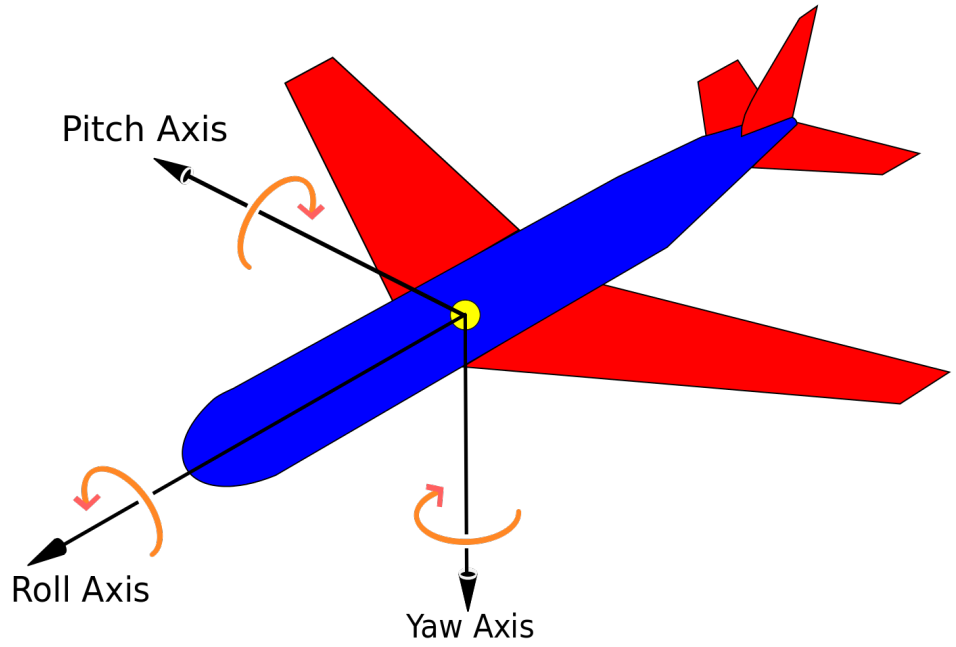


# SOFTBANK PEPPER ROBOT



# SOFTBANK PEPPER ROBOT




3D rotational axes (roll, pitch yaw)

# ROBOTICS MIDDLEWARE

Sensors and actuators are not standardized but produced by different makers.

A robot is a collection of heterogeneous hardware pieces.

Problem: how do we control all these devices from the application level?

# ROBOTICS MIDDLEWARE

*Middleware* is quite a fuzzy term.

Broadly, it's a software layer that connects several different systems.

In robotics, it's a layer between hardware and software.

"Software glue".

# ROBOTICS MIDDLEWARE

"*...robotic middleware is designed to* ***manage the complexity and heterogeneity*** *of the hardware and applications, promote the* ***integration*** *of new technologies,* ***simplify software*** *design,* ***hide the complexity*** *of low-level communication and the sensor heterogeneity of the sensors, improve software quality,* ***reuse robotic software*** *infrastructure across multiple research efforts, and to reduce production costs.*"

Elkady (2012)

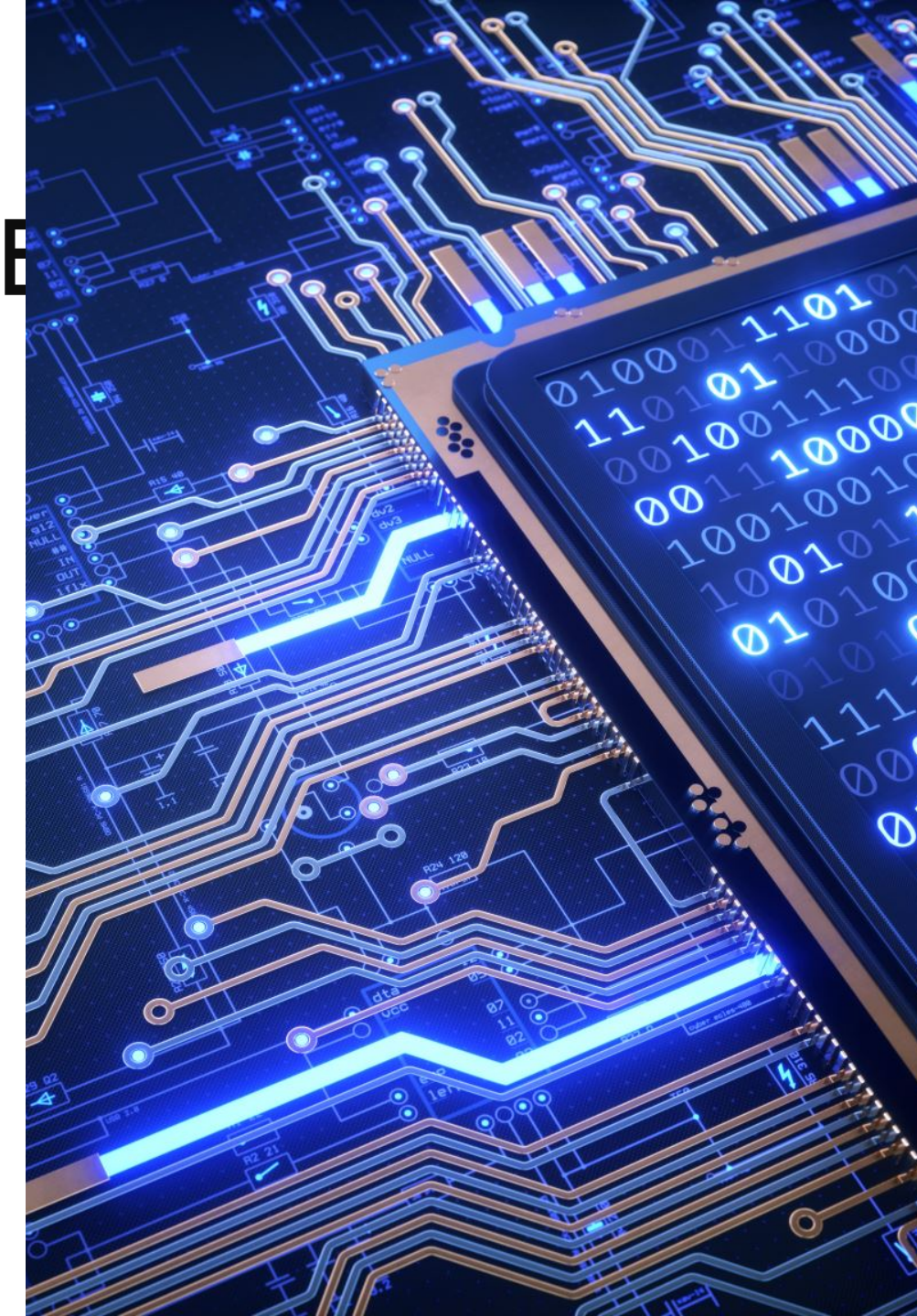
# ROBOTICS MIDDLEWARE

Advantages:

- Modularity
- Reconfigurability
- Reduced coupling
- Language independent

Middleware takes care of:

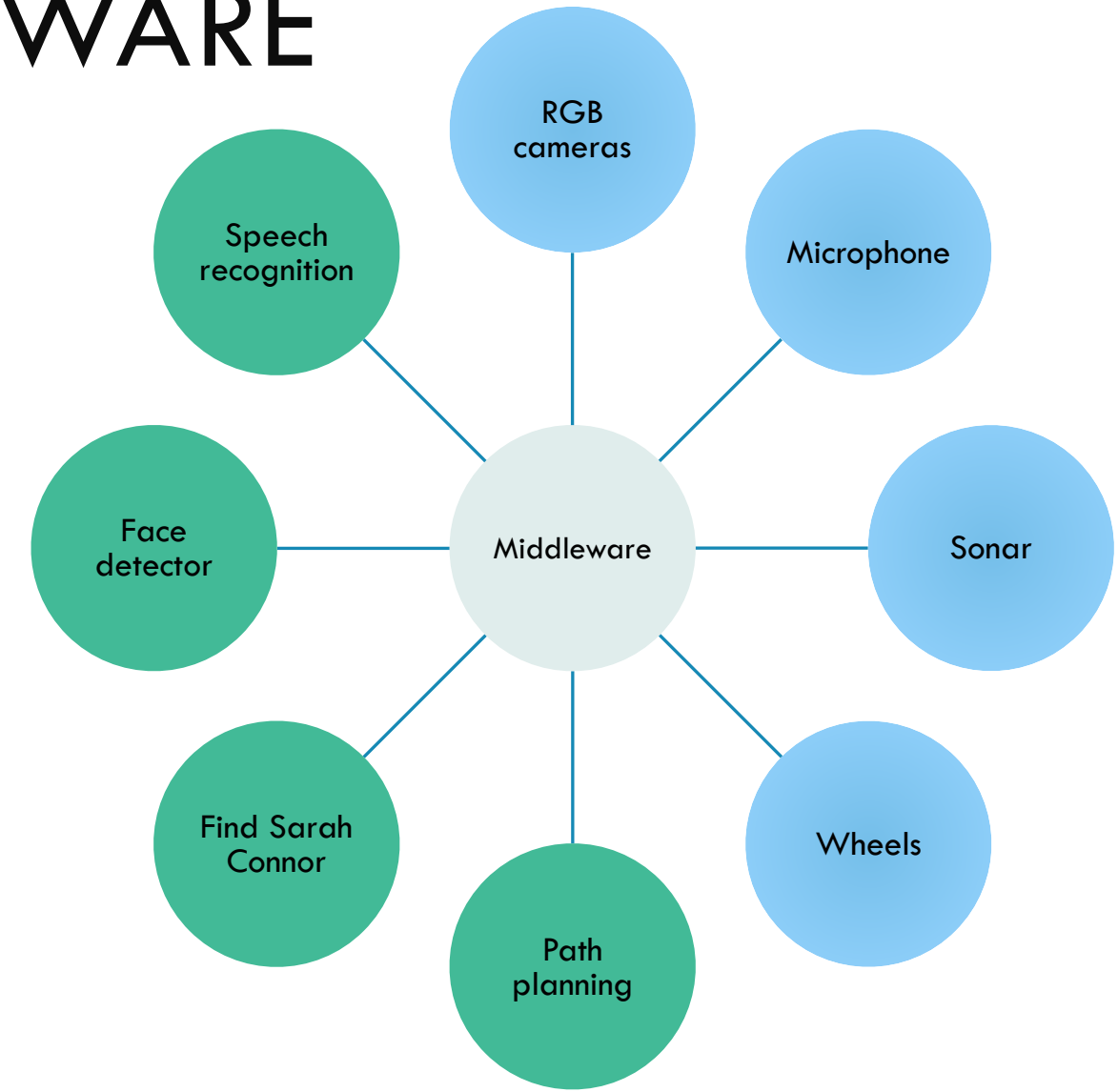
- Information sharing
- Timing
- Data buffering
- Hardware abstraction



# ROBOTICS MIDDLEWARE

Both application and hardware elements are considered "*nodes*" which are connected by the middleware infrastructure.

Each node specifies its inputs and outputs.



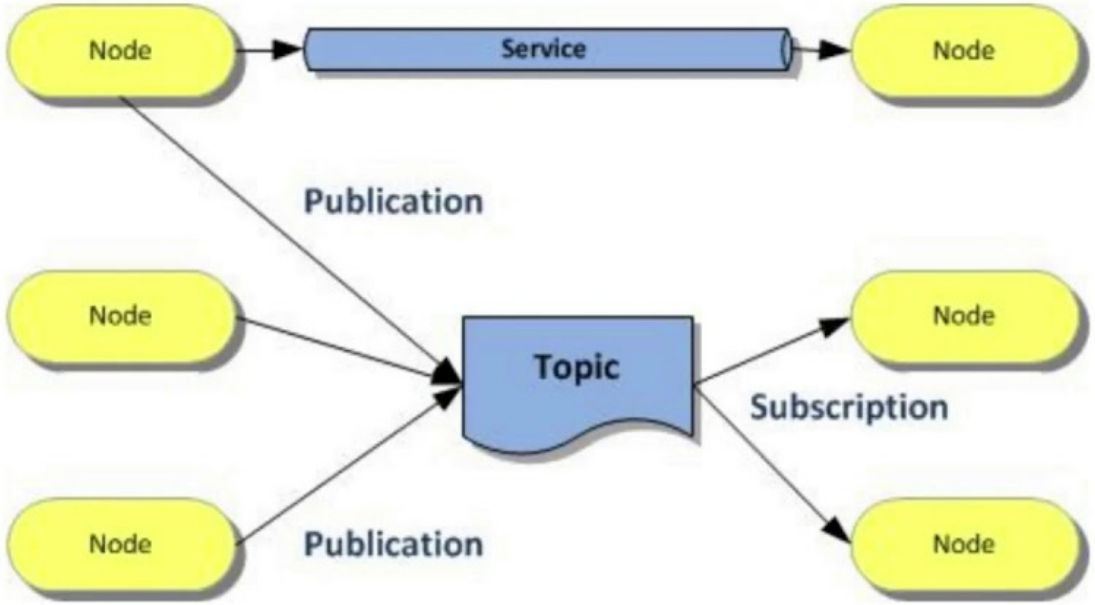
# MIDDLEWARE CASE STUDY: ROS

**R**obot **O**perating **S**ystem (not an OS though!)

Most widespread robotics middleware. Open source, thousands of projects, documentation and supported by the community.

Nodes **publish** on **topics** which are read by **subscriber** nodes (e.g.: a camera streaming images).

Other communication mechanisms (e.g. on-demand "services").



# MIDDLEWARE CASE STUDY: ROS

```python
import rospy
from std_msgs.msg import String

def talker():
    pub = rospy.Publisher('chatter-topic', String, queue_size=10)
    rospy.init_node('talker')
    rate = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        hello_str = "hello world!"
        pub.publish(hello_str)
        rate.sleep()

if __name__ == '__main__':
    talker()
```

Publisher

Subscriber

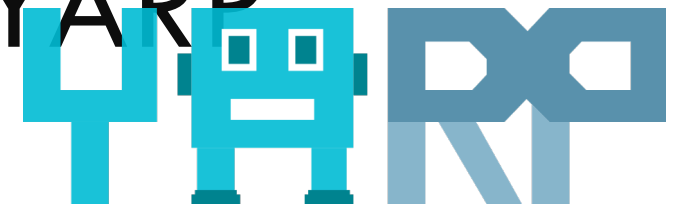
```python
import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo("I heard: %s", data.data)

def listener():
    rospy.init_node('listener')
    rospy.Subscriber("chatter-topic", String, callback)
    rospy.spin()    # loops until killed

if __name__ == '__main__':
    listener()
```

# MIDDLEWARE CASE STUDY: YARP



**Y**et **A**nother **R**obot **P**latform

Very similar to ROS, uses the same publisher/subscriber protocol.

Smaller community, used mainly for certain robots (e.g. iCub).

Mainly C++, bindings for other languages.

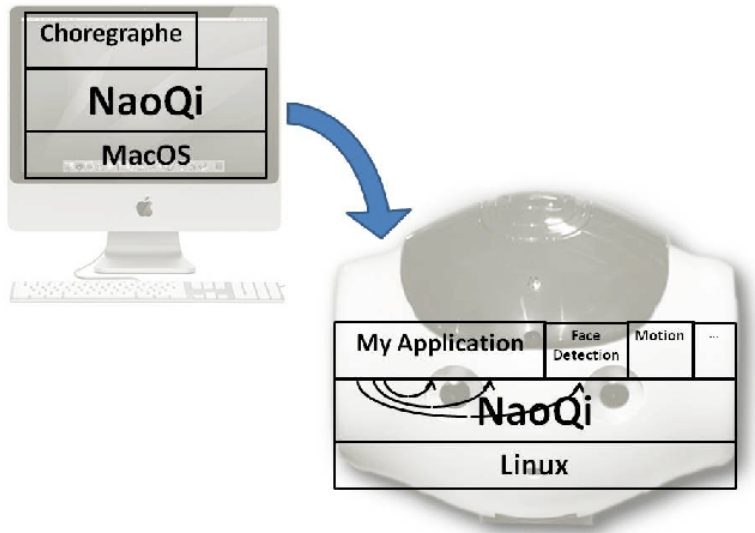
# MIDDLEWARE CASE STUDY: NAOqi

Proprietary software of SoftBank Robotics.

Used across their robots: Nao, Pepper and Romeo.

Interfaces in Python and C++.

More about this later on…




# PROGRAMMING PEPPER

ROS

Choregraphe

Python
C++

# CHOREGRAPHE


Easiest way of programming the robot.

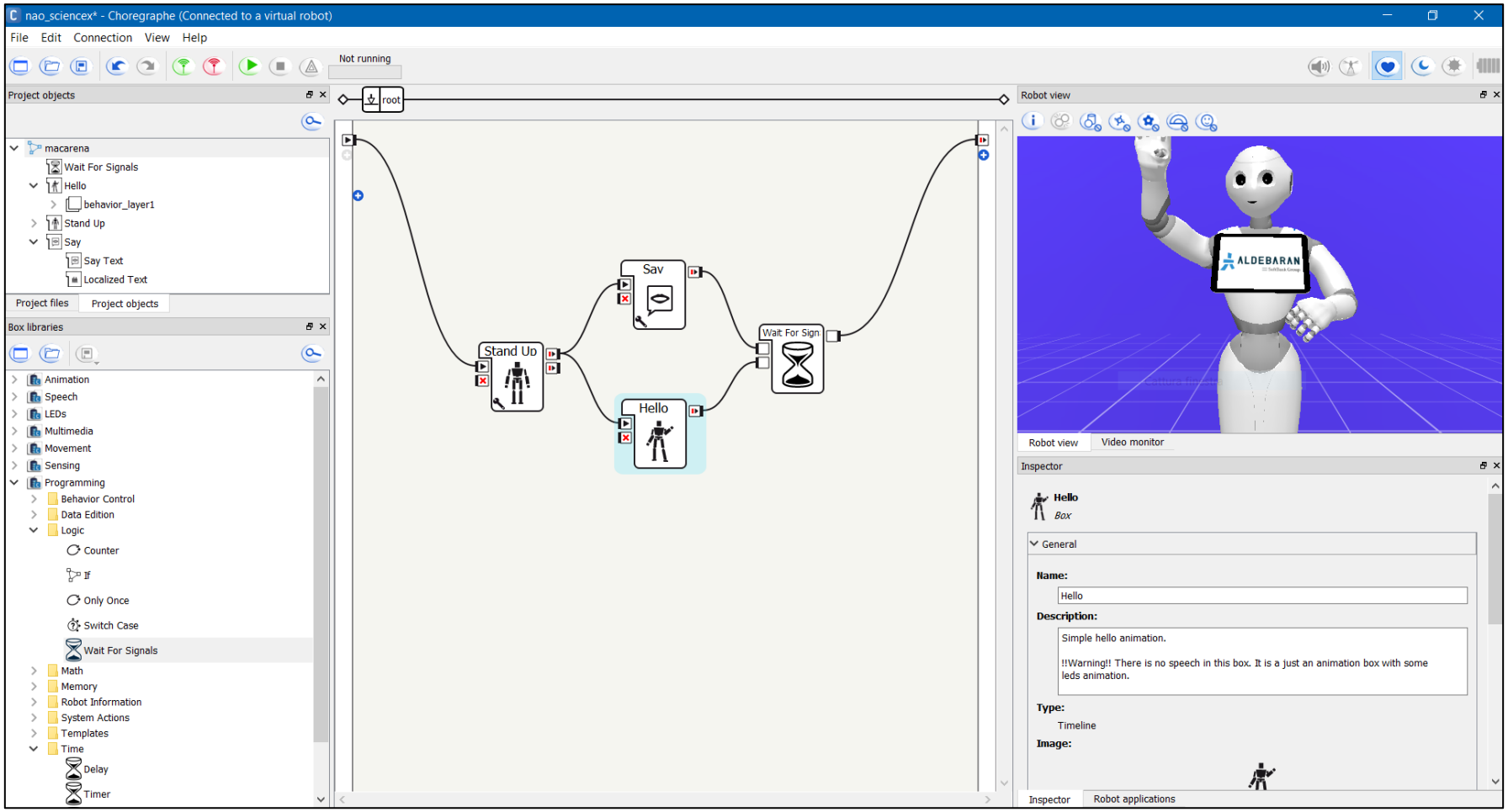
No code: only graphical blocks that are combined to form behaviors.

Useful for:

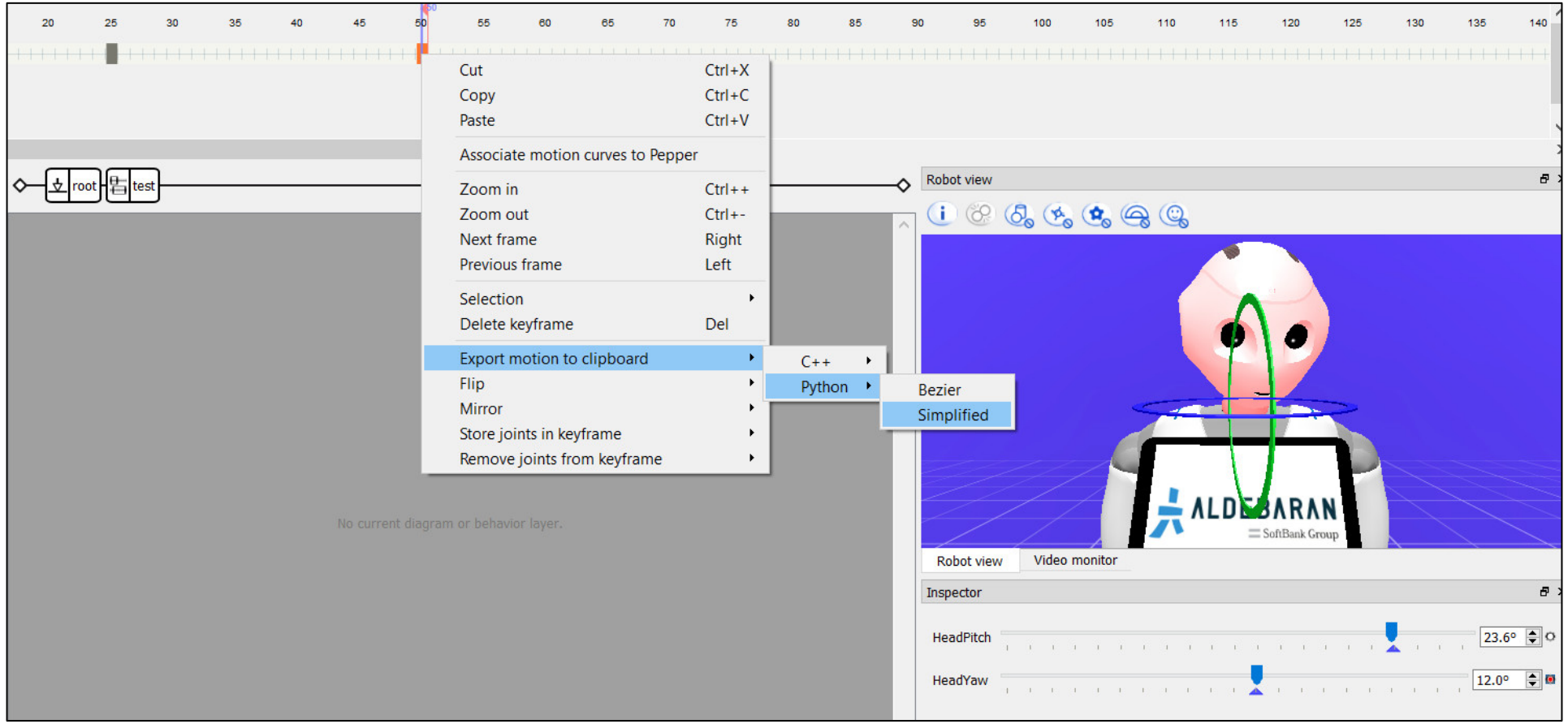
- Learning robot programming
- Developing simple behaviors
- Simulation
- Monitoring the robot
- Designing animations

http://doc.aldebaran.com/2-5/software/choregraphe/tutos/index.html

# CHOREGRAPHE OVERVIEW



# CHOREGRAPHE OVERVIEW

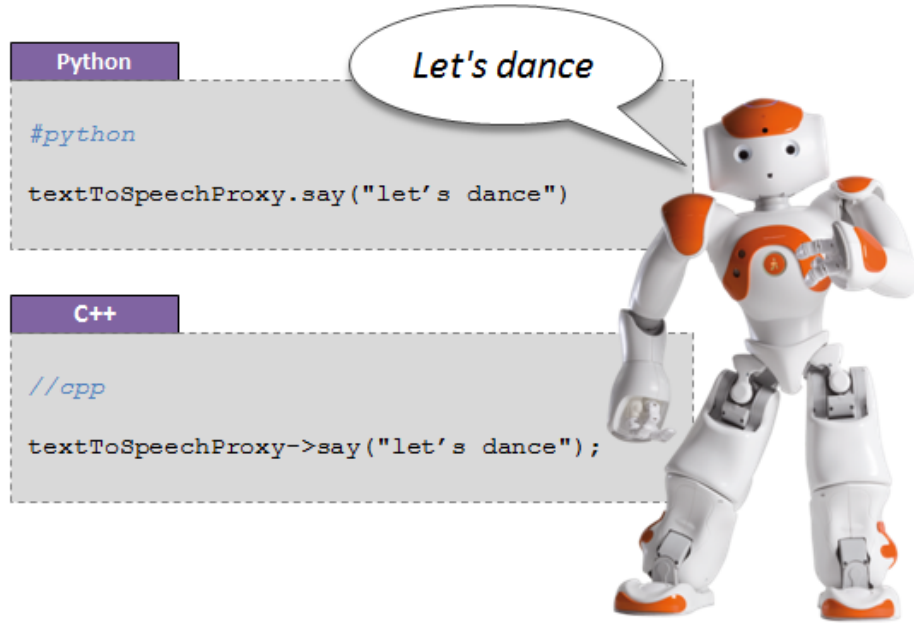


# PYTHON PROGRAMMING

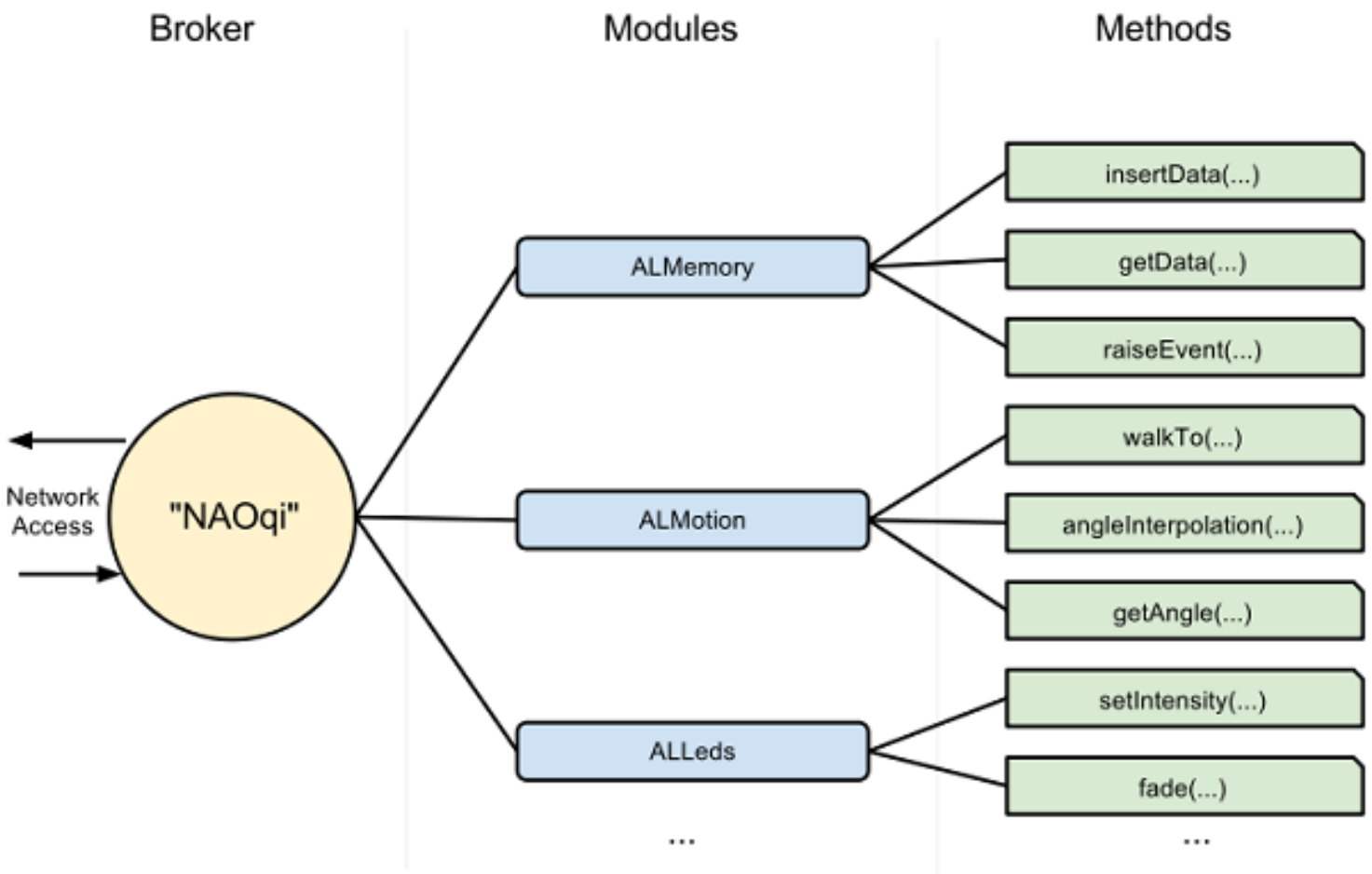
**NAOqi** is the name of the main software that runs on the robot and controls it.

Interfaces in Python and C++.

http://doc.aldebaran.com/2-5/dev/naoqi/index.html

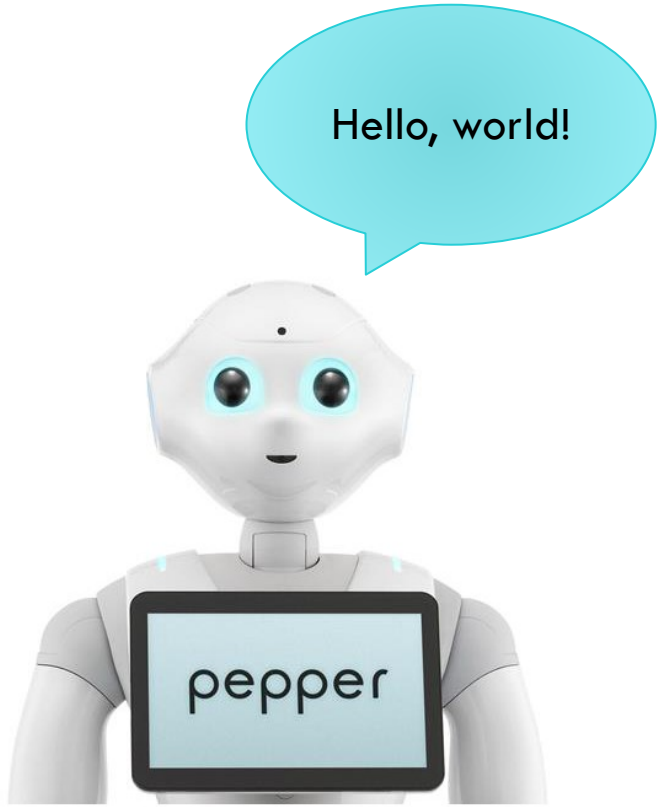


# PYTHON PROGRAMMING



# PYTHON PROGRAMMING

```
from naoqi import ALProxy
IP = "pepper.local"
PORT = 9559
tts = ALProxy("ALTextToSpeech", IP, PORT)
tts.setLanguage("English")
tts.say("Hello, world!")
```



# PYTHON PROGRAMMING

```
from naoqi import ALProxy
IP = "pepper.local"
PORT = 9559
motion = ALProxy("ALMotion", IP, PORT)
tts    = ALProxy("ALTextToSpeech", IP, PORT)
motion.moveInit()
motion.moveTo(0.5, 0, 0)            # (x; y; theta)
tts.say("I am walking")
```

Talks while walking

Talks after walking

```
from naoqi import ALProxy
IP = "pepper.local"
PORT = 9559
motion = ALProxy("ALMotion", IP, PORT)
tts    = ALProxy("ALTextToSpeech", IP, PORT)
motion.moveInit()
id = motion.post.moveTo(0.5, 0, 0)          # (x; y; theta)
motion.wait(id, 0)
tts.say("I have walked")
```

# PYTHON PROGRAMMING

Useful modules to keep in mind (check documentation for full API description):

- ALVideoDevice
- ALTextToSpeech
- ALAnimatedSpeech
- ALMotion
- ALFaceTracker
- ALLeds
- ALLandmarkDetection
- ALMemory
- ALSpeechRecognition

API documentation: http://doc.aldebaran.com/2-5/naoqi/index.html (you'll need it!)

# PREPARING FOR THE PRACTICAL LABS

1. Download **VirtualBox** from:
   https://www.virtualbox.org/
2. Download the **virtual machine** (~5Gb) with all the required software already pre-installed from:
   https://mega.nz/file/11jgCBb#LgIEon7rWvKYGGK92VoRZXg26afFEJ2LcxcR1VDeWH0
3. Launch VirtualBox. In the top menu, click on “File”, then “Import Appliance…” and select the OVA file you just downloaded. Click “Next”, then “Import”.
4. (For Mac users) Be sure to follow this guide to allow the program to run correctly:
   https://medium.com/@Aenon/mac-virtualbox-kernel-driver-error-df39e7e10cd8