Evangelos Boukas

34761 – Robot Autonomy

Introduction to Robot Autonomy

Outline

- <u>**Overview of 34761 – Robot Autonomy**</u>
  - Course format, topics, learning objectives, Exam
- Guest presentation from NNE + Robot challenge!!
- Software architecture
  - Operation paradigms
  - Hardware abstraction
- Exercises
  - Setting up a docker container with ROS2 installed

What is this course about? What is Robot Autonomy?

Robot localization

Mapping of environments

Autonomy for Robots

Behavior definitions

Navigating environments

Exploring new environments

Overview of 34761 – Robot Autonomy

- 3 lectures on software frameworks
- 7 lectures on building your own autonomy stack for a mobile robot
- 1 lecture on DL/RL – an overview of black-box approaches to what you have done
- 2 lectures of project work before hand in + guest lectures

Software frameworks	30th Jan	Software architecture
	6th Feb	Intro to ROS2
	13th Feb	More on ROS2
Robot autonomy stack	20th Feb	Localization
	27th Feb	Mapping
	6th Mar	Localization wrt. maps
	13th Mar	SLAM
	20th Mar	Navigation
	27th Mar	Easter
	3rd Apr	Exploration
	10th Apr	Behavior Trees
	17th Apr	Black Box Approaches
Project work + guest lectures	24th Apr	Group work
	1st May	Group work

Today

Overview of 34761 – Robot Autonomy

- 5 ECTS

- Lecturer:
  - Rasmus Andersen, PostDoc, building 326, Room 030. 
  - Evangelos Boukas, Building 326, Room 020. 

- Teaching assistants:
  - Dimosthenis Angelis, building 326, Room 030. 
  - Dimitrios Syrrafos, building 326, Room 030. 

Overview of 34761 – Robot Autonomy

- Lectures take place Tuesdays 13.00 – 17.00, Building 341, aud. 21
- Approximately 2-hour lectures
- Approximately 2-hour exercises
  - Work on course material in groups
    - Group formation:
      - 5-6 students in each group
      - There's a spreadsheet on Learn where all group members sign up
  - Course TAs will be available during the course hours

Learning objectives

- A student who has met the objectives of the course will be able to:
  - Integrate the building blocks of robot operation, situational awareness, and robot introspection using ROS
  - Explain what autonomy is in a robotics context
  - Design and implement a behavioral system for robots operating in unknown environments
  - Describe mission planning, fault detection, environment changes, and re-planning
  - Interpret high-level mission goals to autonomous system architectures
  - Formulate and convert high-level mission goals to quantifiable objective functions
  - Explain the different steps in mobile robot localization and implement the related algorithms
  - Suggest and implement algorithms for robot situational awareness using multi-modal sensing

Exam / Evaluation

- To successfully complete the course, you need to:
  1. Submit a group report about your final project
     - Students are expected to work in their groups
     - Implement your own version of a set of selected topics from the course material
     - Hand in an approximately 4-6-page paper of the work including a (video) demonstration
       – Use standard IEEE template format
  2. Passing the report will be a prerequisite to join the final exam questionnaire

- Important dates:
  – **Deadline for forming groups: 13th February 2024**
  – **Deadline for handing in report: 1st May at 23:55 2024**
  – **Notification of failed reports: 8th May at 16:00 2024**
  – **Examination: 23rd May 2024**

NNE / DTU RoboTech Challenge

Presented by the Fill Finish Automation Team from NNE

Outline

- Overview of 34761 – Robot Autonomy
  - Course format, topics, learning objectives, Exam
- Guest presentation from NNE + Robot challenge!!
- **<u>Software architecture</u>**
  - Operation paradigms
  - Hardware abstraction
- Exercises
  - Setting up a docker container with ROS2 installed

Operation Paradigms / Software Architectures

- A high-level model of how an Autonomous System operates

- Example from Robotics: **Robotic Paradigm**
  – Robot software architectures can be built by combining 3 primitive operations:
    - Sense
    - Plan
    - Act

Operation Paradigms / Software Architectures

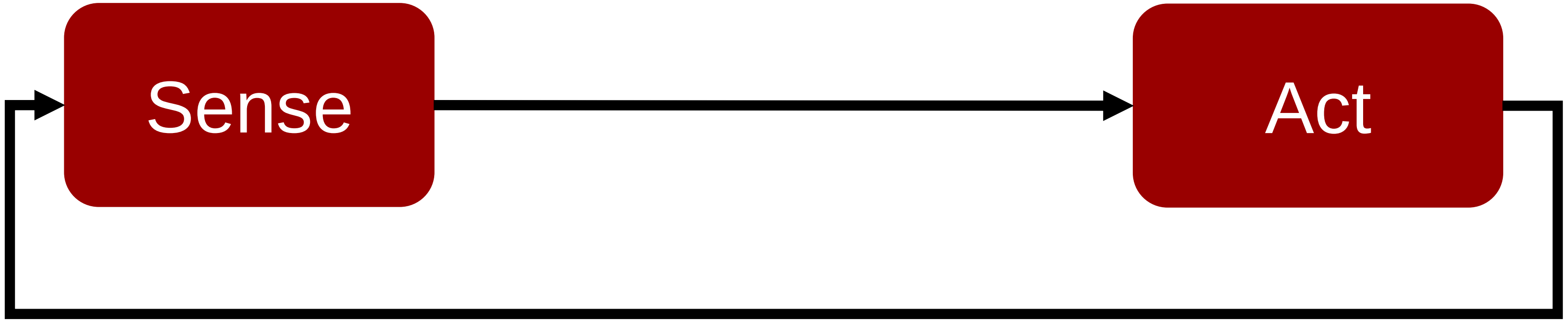
- Hierarchical (deliberative) paradigm

Operation Paradigms / Software Architectures

- Hierarchical (deliberative) paradigm
  - First introduced with the Shakey Robot in 1966-1972.
  - A “world model” is needed that fully describes the operation environment.
  - Planners can be formalized and used to provide next actions.

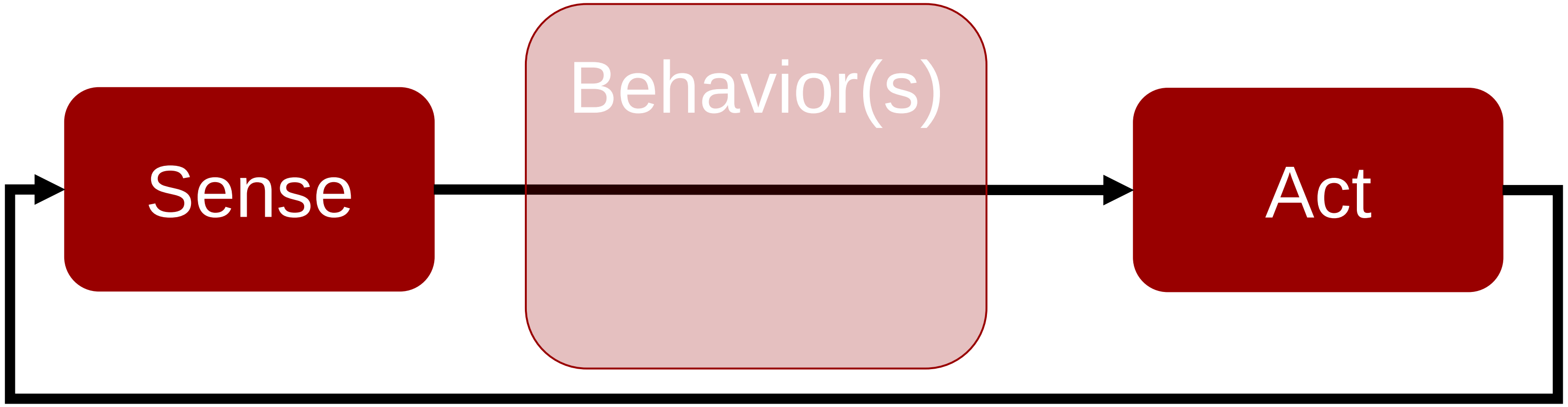
Operation Paradigms / Software Architectures

- Reactive paradigm



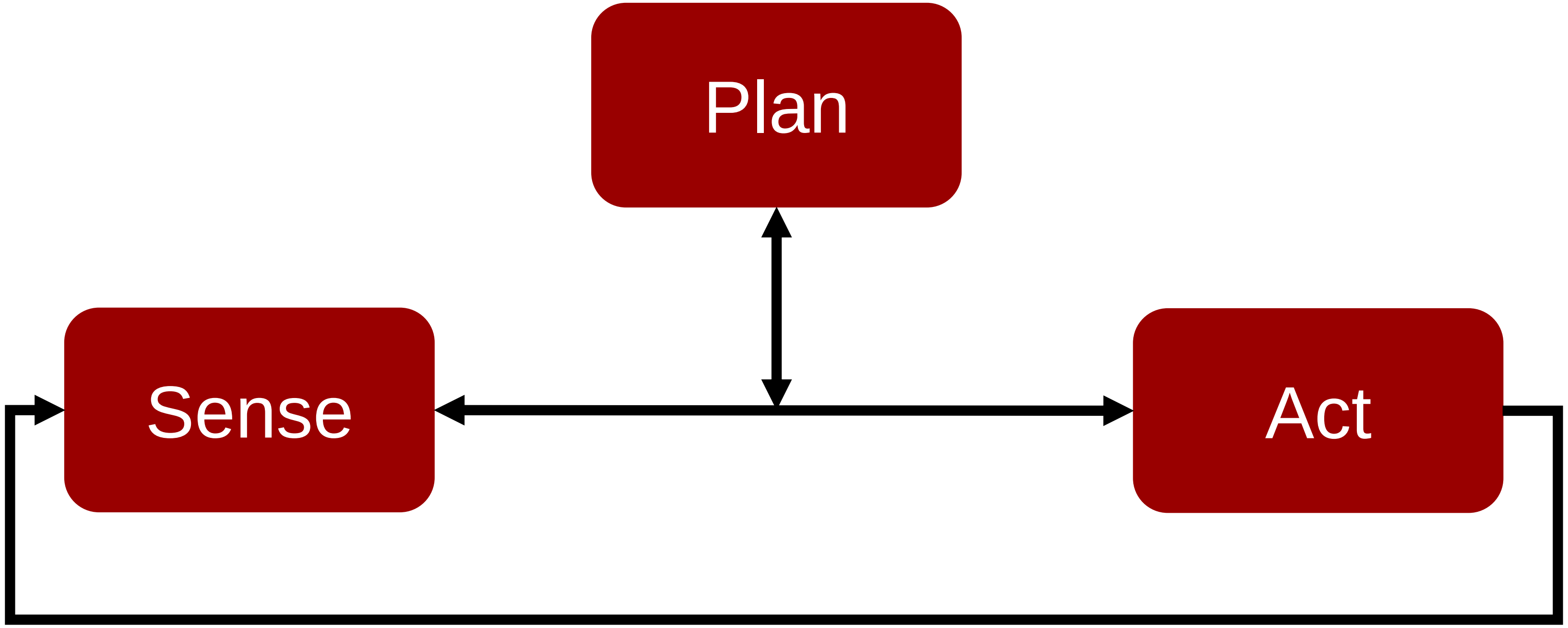
Operation Paradigms / Software Architectures

- Reactive paradigm
  - Biologically inspired like instincts/reflexes
  - No need for world models and model updates
  - Behavior-based robotics (there can be multiple behaviors)



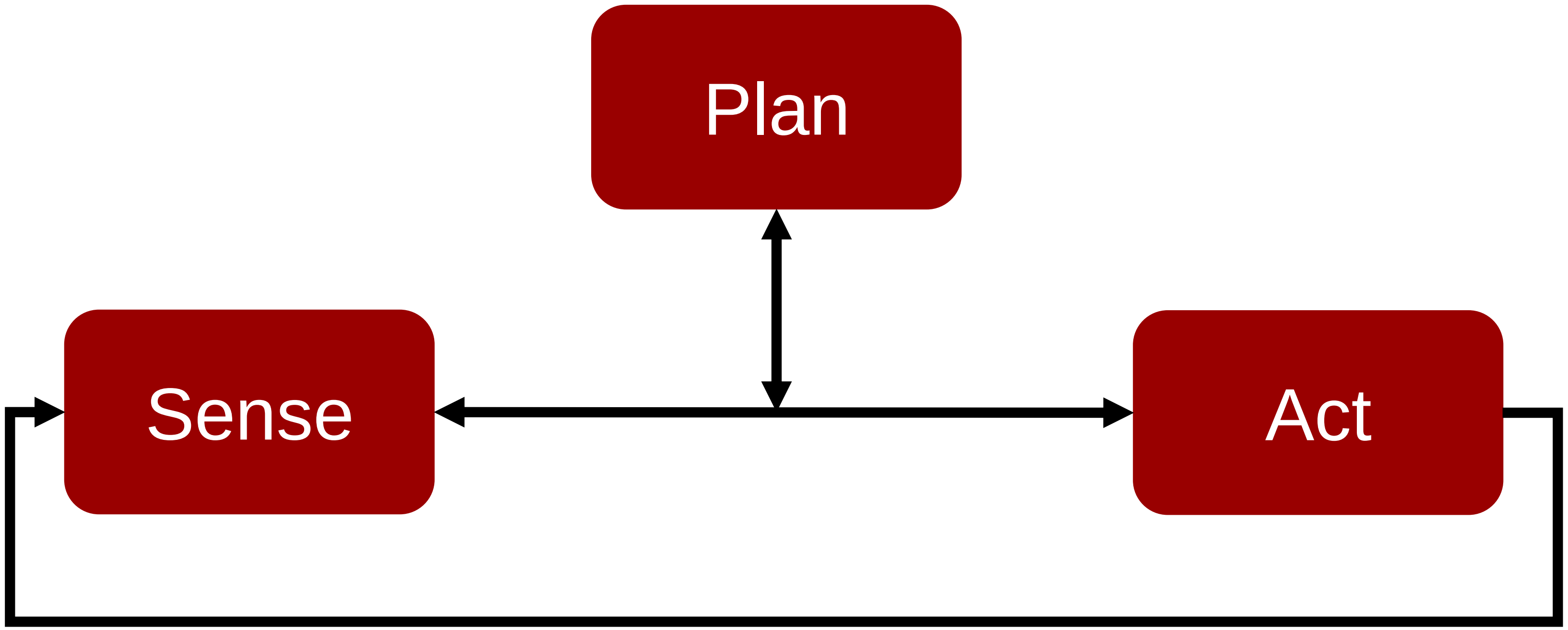
Operation Paradigms / Software Architectures

- Hybrid paradigm / Three Layer Architecture



Operation Paradigms / Software Architectures

- Hybrid paradigm / Three Layer Architecture
  - Best of the 2 first paradigms
    - Planning operates on a long horizon
    - Reactive behaviors act on the short horizon



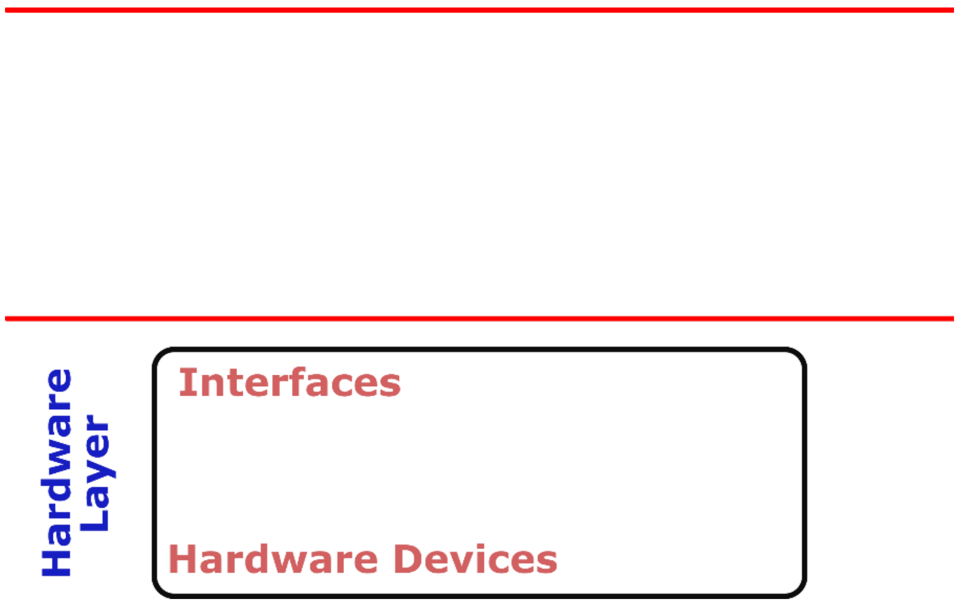
Introduction

- What is a ”middleware”?

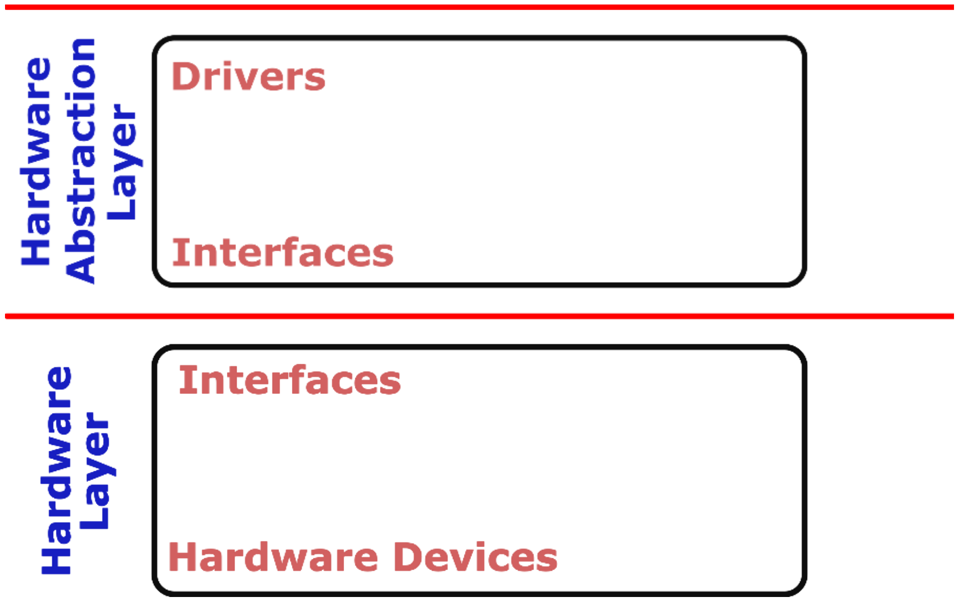
Introduction

- What is a ”middleware”?
- Why do we need it?

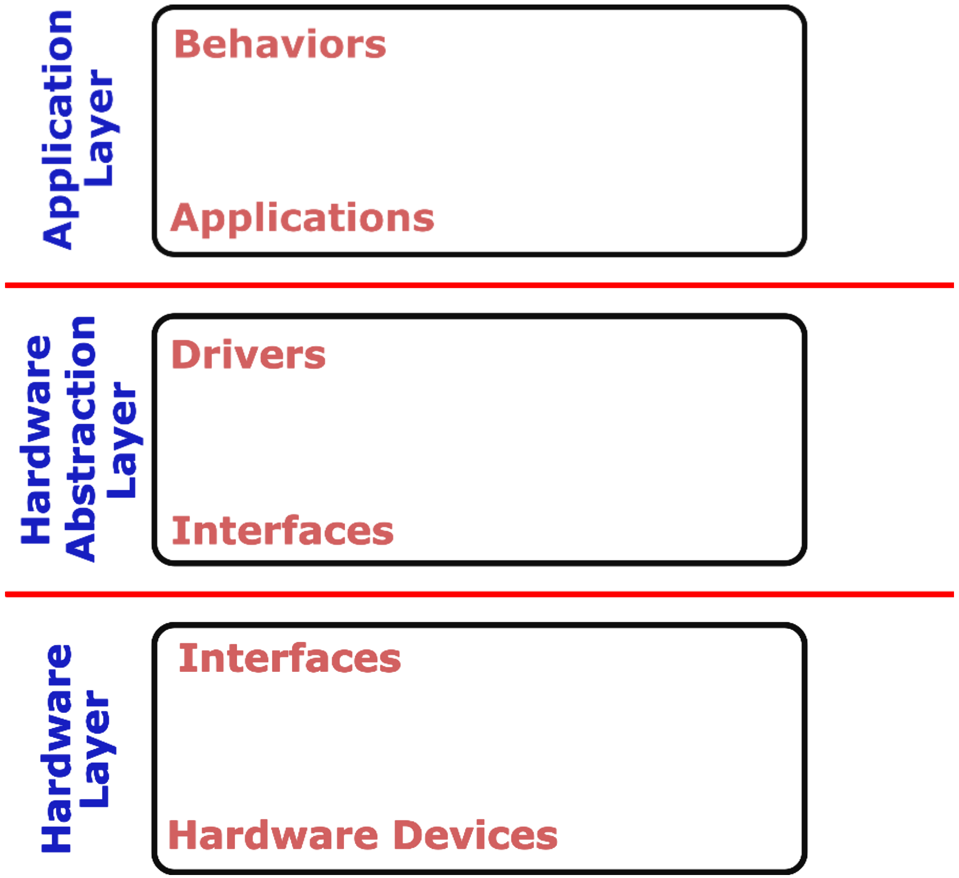
Hardware Abstraction Layers #1



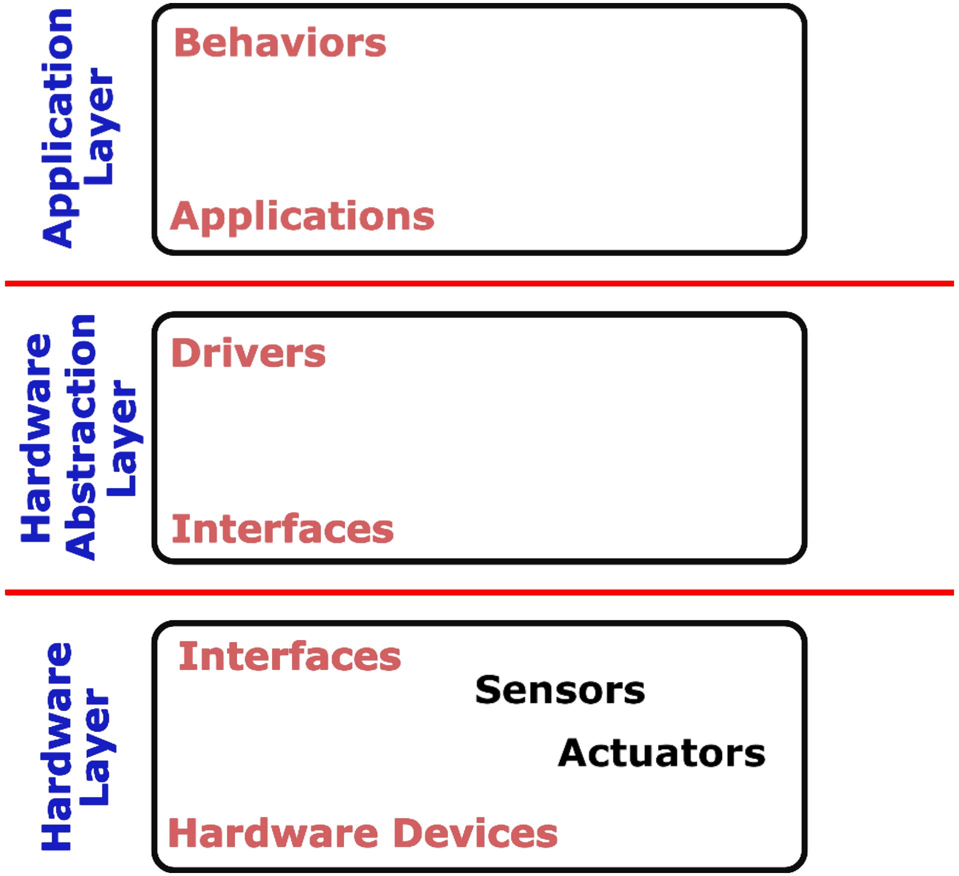
Hardware Abstraction Layers #1



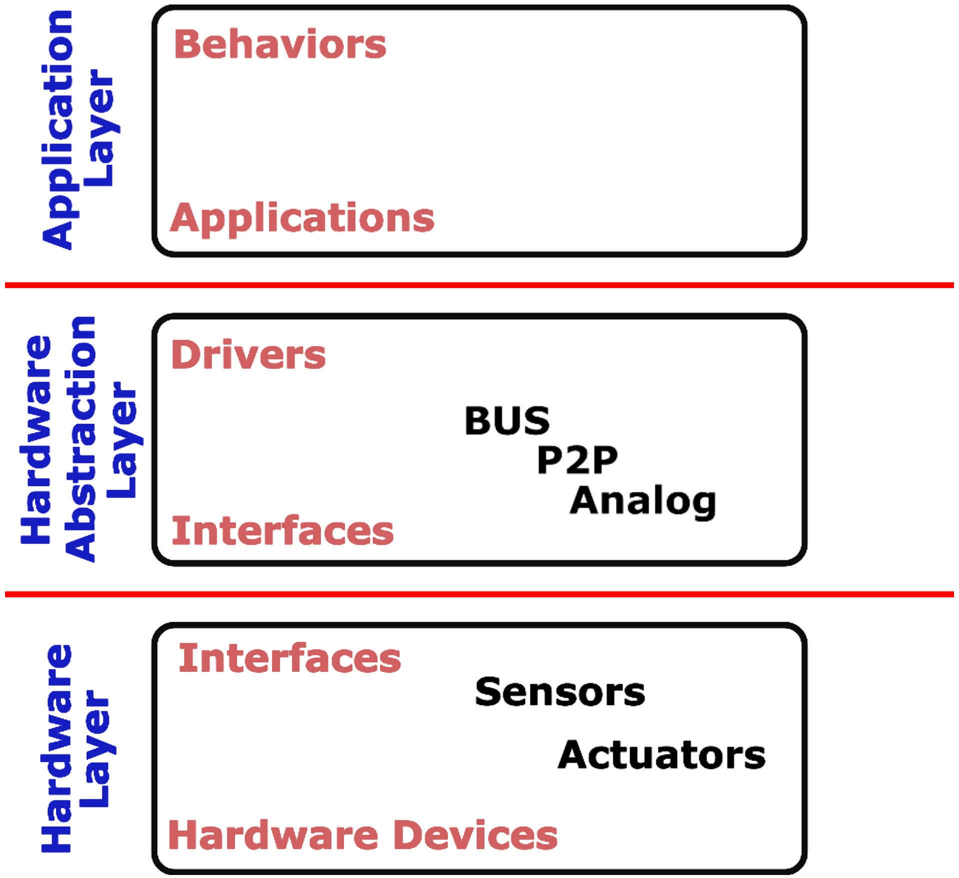
Hardware Abstraction Layers #1



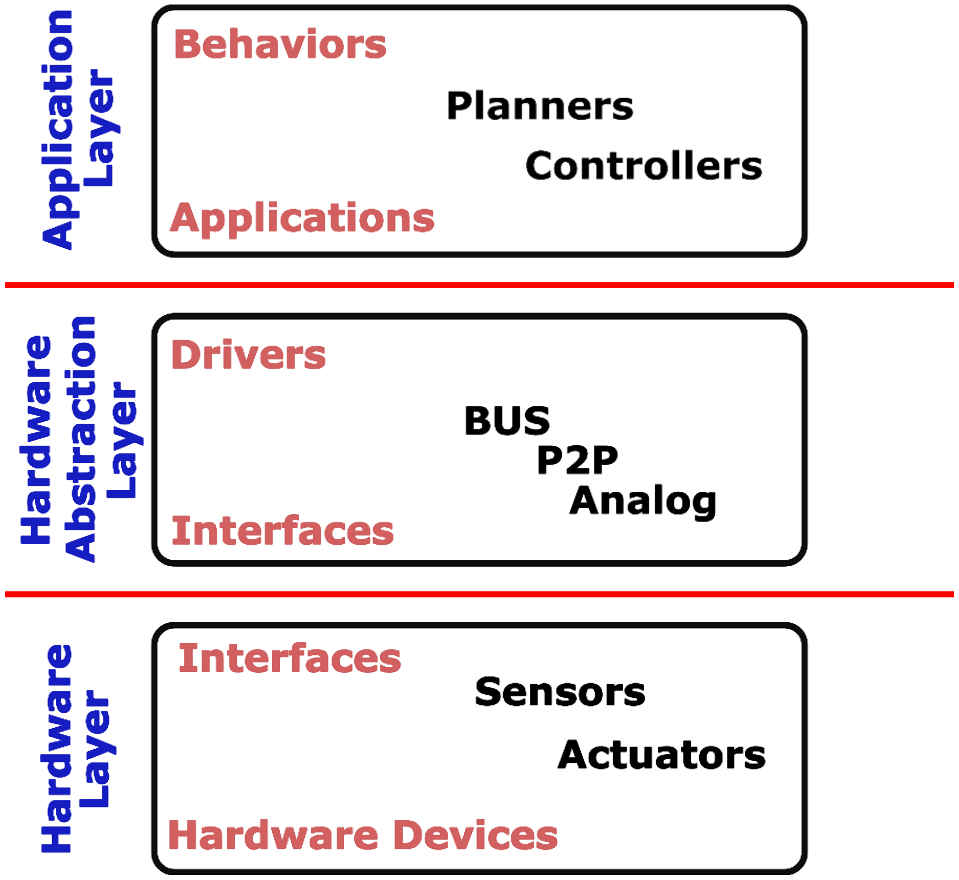
Hardware Abstraction Layers #1



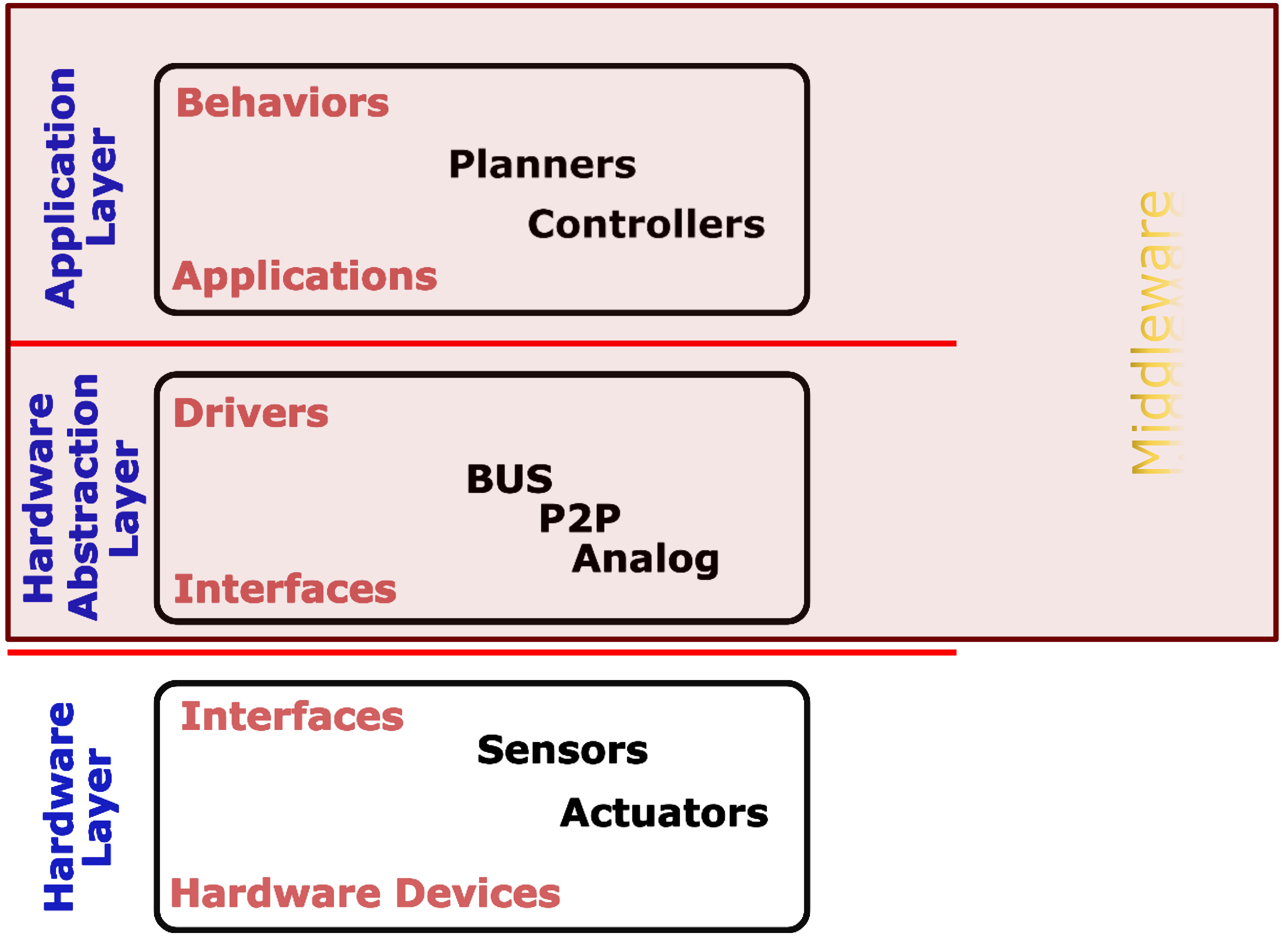
Hardware Abstraction Layers #1



Hardware Abstraction Layers #1



Hardware Abstraction Layers #1



Hardware Abstraction Layers #2

- Let’s add a little detail..

Hardware Abstraction Layers #2

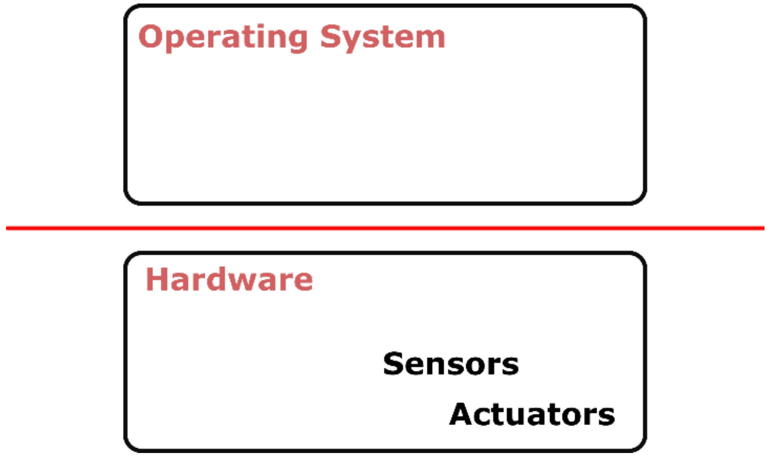
- Let’s add a little detail..

Hardware Abstraction Layers #2

- Let's add a little detail..

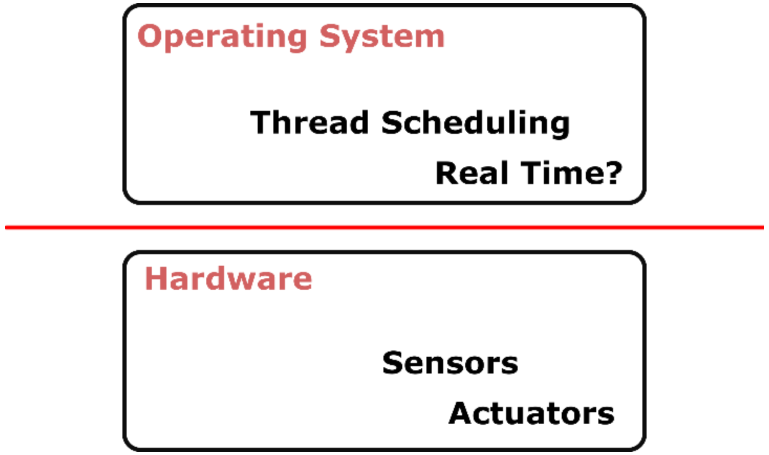
Hardware Abstraction Layers #2

- Let’s add a little detail..



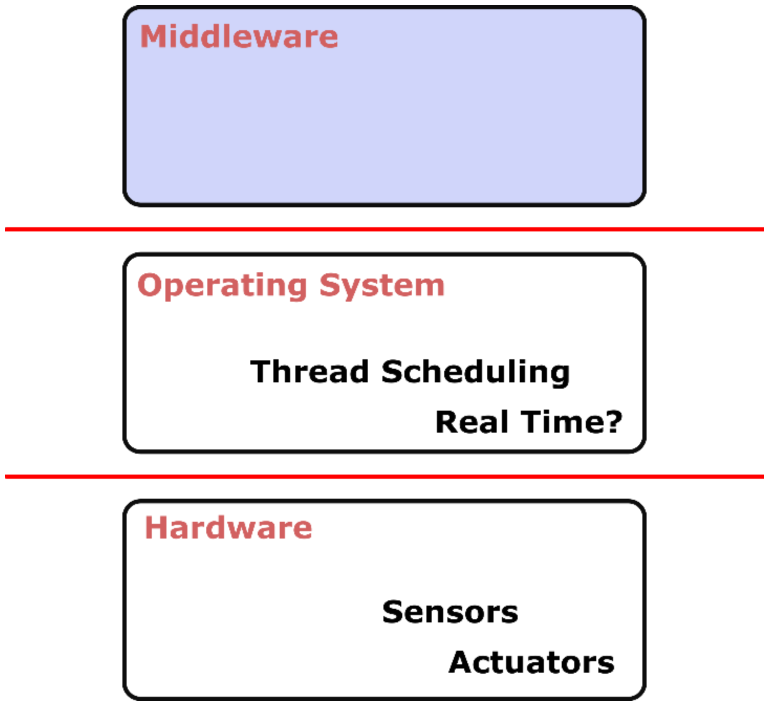
Hardware Abstraction Layers #2

- Let's add a little detail..



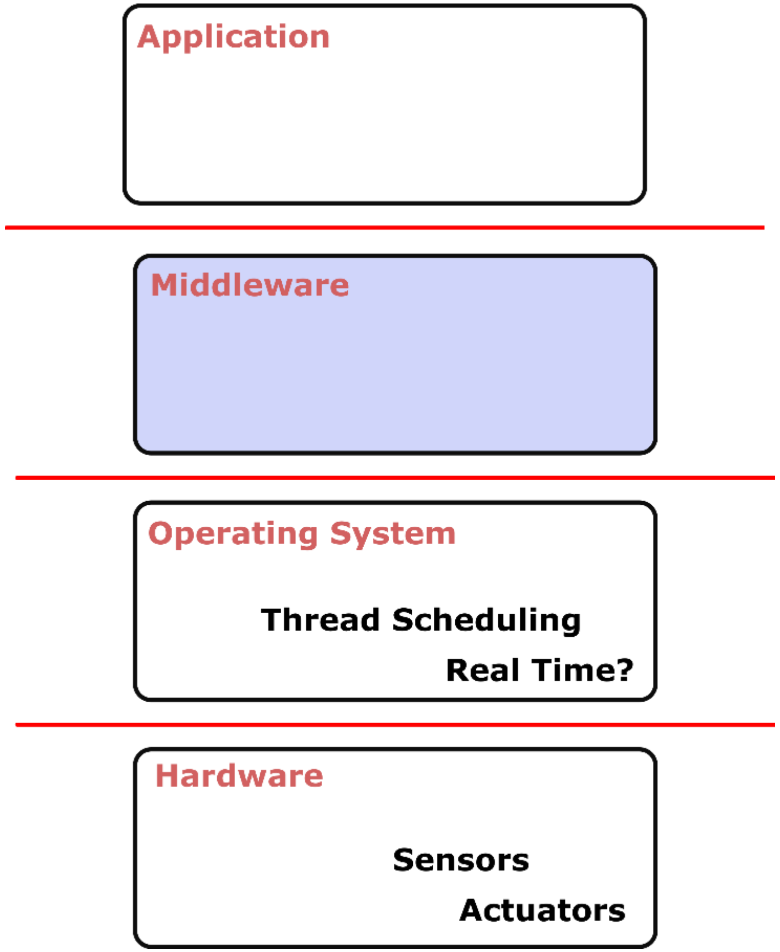
Hardware Abstraction Layers #2

- Let's add a little detail..



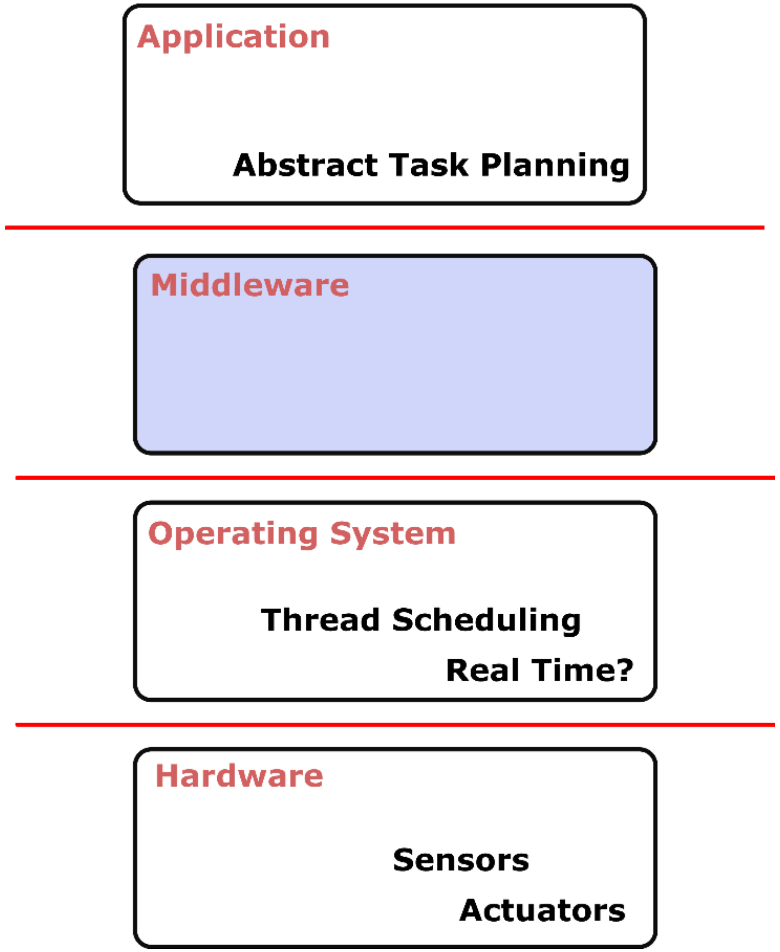
Hardware Abstraction Layers #2

- Let’s add a little detail..



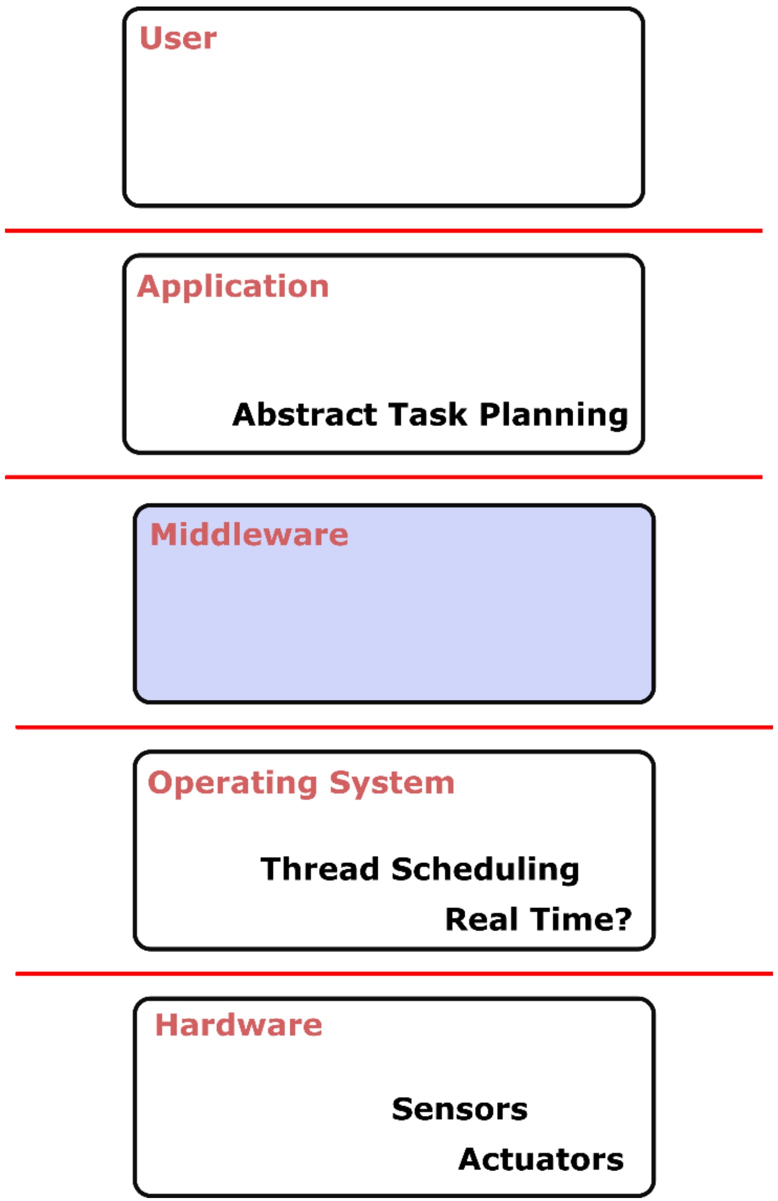
Hardware Abstraction Layers #2

- Let’s add a little detail..



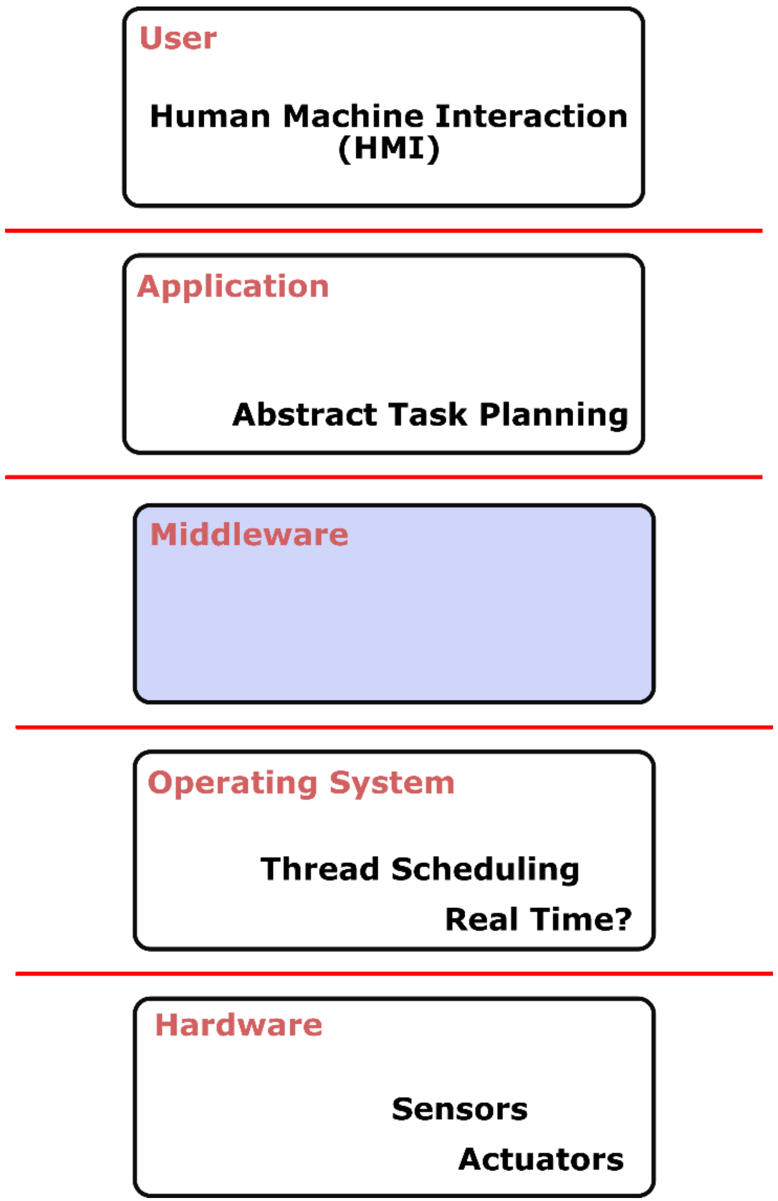
Hardware Abstraction Layers #2

- Let’s add a little detail..



Hardware Abstraction Layers #2

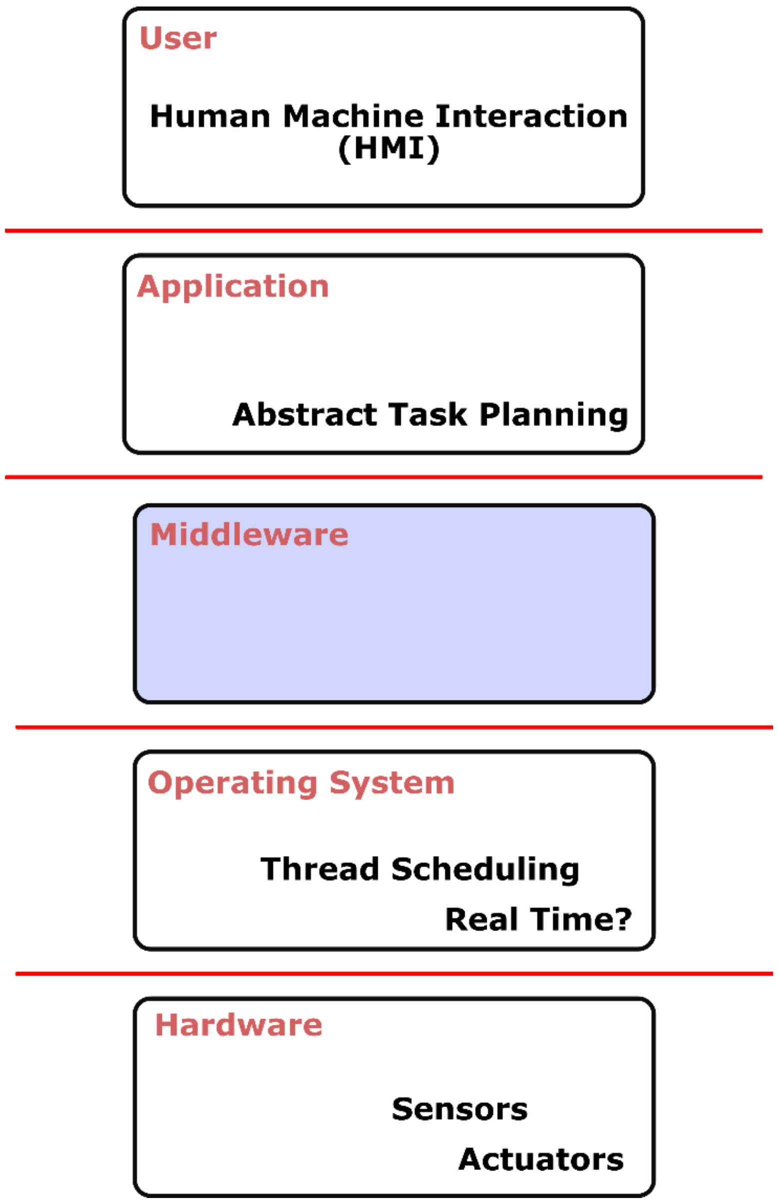
- Let’s add a little detail..



Hardware Abstraction Layers #2

- Let’s add a little detail..

- A class of technologies in order to handle the complexity of distributed systems



What are some Middleware?

What are some Middleware?

What are some Middleware?

What are some Middleware?

USC University of Southern California

Player Stage

NASA

Microsoft Robotics Developer Studio

Microsoft

What are some Middleware?

USC University of Southern California

Player Stage

NASA

Microsoft Robotics Developer Studio

Microsoft

ROS

Willow Garage

What are some Middleware?

USC University of Southern California

Player Stage

NASA

Microsoft Robotics Developer Studio

Microsoft

ROS

Willow Garage

THE OROCOS PROJECT

Open Robot Control Software

EURON Interest Group

Modular Architecture

DTU MOBOTWARE

KATHOLIEKE UNIVERSITEIT LEUVEN

What are some Middleware?

- Orocos
- Pyro
- Player
- Orca
- Miro
- OpenRTMaist
- ASEBA
- MARIE
- RSCA

- MRDS
- OPROS
- CLARAty
- ROS
- SmartSoft
- ERSP
- Webots
- RoboFrame

What’s UP with this?

What's UP with this?

- Is there any reason there're so many?

What’s UP with this?

- Is there any reason there’re so many?
- Different Scope

What’s UP with this?

- Is there any reason there’re so many?

- Different Scope
- Different Functional Architecture

What’s UP with this?

- Is there any reason there’re so many?

- Different Scope
- Different Functional Architecture
- Different Communication Architectures

Let's see the most important ones!

- CLARAty

Let’s see the most important ones!

- CLARAty
  - NASA Jet Propulsion Laboratory

Let’s see the most important ones!

- CLARAty
  - NASA Jet Propulsion Laboratory
  - Functional Layer:
    - Navigation, Mapping
    - Terrain evaluation, path planning
    - Estimation, simulation

Let’s see the most important ones!

- CLARAty
  - NASA Jet Propulsion Laboratory
  - Functional Layer:
    - Navigation, Mapping
    - Terrain evaluation, path planning
    - Estimation, simulation
  - Decision Layer:
    - General Planners
      Schedulers, Databases

Let’s see the most important ones!

- CLARAty
  - NASA Jet Propulsion Laboratory
  - Functional Layer:
    - Navigation, Mapping
    - Terrain evaluation, path planning
    - Estimation, simulation
  - Decision Layer:
    - General Planners
      Schedulers, Databases
  - Client ←→Server Scheme

Let’s see the most important ones!

- CLARAty
- Orocos

Let’s see the most important ones!

- CLARAty
- Orocos
  – RealTime

Let’s see the most important ones!

- CLARAty
- Orocos
  - RealTime
  - Orocos Components Library (OCL)
  - Orocos Kinematics and Dynamics Library (KDL)
  - Orocos Bayesian Filtering Library (BFL)

Let’s see the most important ones!

- CLARAty
- Orocos
  - RealTime
  - Orocos Components Library (OCL)
  - Orocos Kinematics and Dynamics Library (KDL)
  - Orocos Bayesian Filtering Library (BFL)
  - OMG’s CORBA

Let’s see the most important ones!

- CLARAty
- Orocos
- Player

Let’s see the most important ones!

- CLARAty
- Orocos
- Player

Kind of Opposite

Let's see the most important ones!

- CLARAty
- Orocos
- Player
  - Shared Libraries among devices

Kind of Opposite

Let’s see the most important ones!

- CLARAty
- Orocos
- Player
  - – Shared Libraries among devices
  - – Player Core
    - Drivers, Libraries
    - Configuration Parsing

Kind of Opposite

Let’s see the most important ones!

- CLARAty
- Orocos
- Player
  - Shared Libraries among devices
  - Player Core
    - Drivers, Libraries
    - Configuration Parsing
  - Transport Layer
    - Independent of Drivers
    - TCP communication using web sockets

Kind of Opposite

Let’s see the most important ones!

- CLARAty
- Orocos
- Player
- ROS

Let’s see the most important ones!

- CLARAty
- Orocos
- Player
- ROS

Logical Extension

Let’s see the most important ones!

- CLARAty
- Orocos
- Player
- ROS
  - Master

Logical Extension

Let's see the most important ones!

- CLARAty
- Orocos
- Player
- ROS
  - Master
  - Nodes

Logical Extension

Let’s see the most important ones!

- CLARAty
- Orocos
- Player
- ROS
  - Master
  - Nodes
  - Topics
    - Messages

Logical Extension

Let’s see the most important ones!

- CLARAty
- Orocos
- Player
- ROS
  - Nodes
  - Topics
    - Messages
  - Publish/Subscribe Scheme

Logical Extension

Let’s see the most important ones!

- CLARAty
- Orocos
- Player
- ROS
- ROS2

Let's see the most important ones!

- CLARAty
- Orocos
- Player
- ROS
- ROS2
  - Data Distribution Service (DDS)

No master anymore!!

Let’s see the most important ones!

- CLARAty
- Orocos
- Player
- ROS
- ROS2
  - Data Distribution Service (DDS)
  - Nodes
  - Topics
    - Messages
  - Publish/Subscribe Scheme

Let’s see the most important ones!

- CLARAty
- Orocos
- Player
- ROS
- ROS2
  - Data Distribution Service (DDS)
  - Nodes
  - Topics
    - Messages
  - Publish/Subscribe Scheme
  - In general, tries to solve the same thing in a very similar way, but better

Exercises (Find them on LEARN)

- Install docker on your system and build a container with ROS2 installed