An Erlang Framework for Autonomous Mobile Robots

Corrado Santoro, (presenter Vincenzo Nicosia)

University of Catania, Engineering Faculty, ITALY

Department of Computer and Telecommunication Engineering

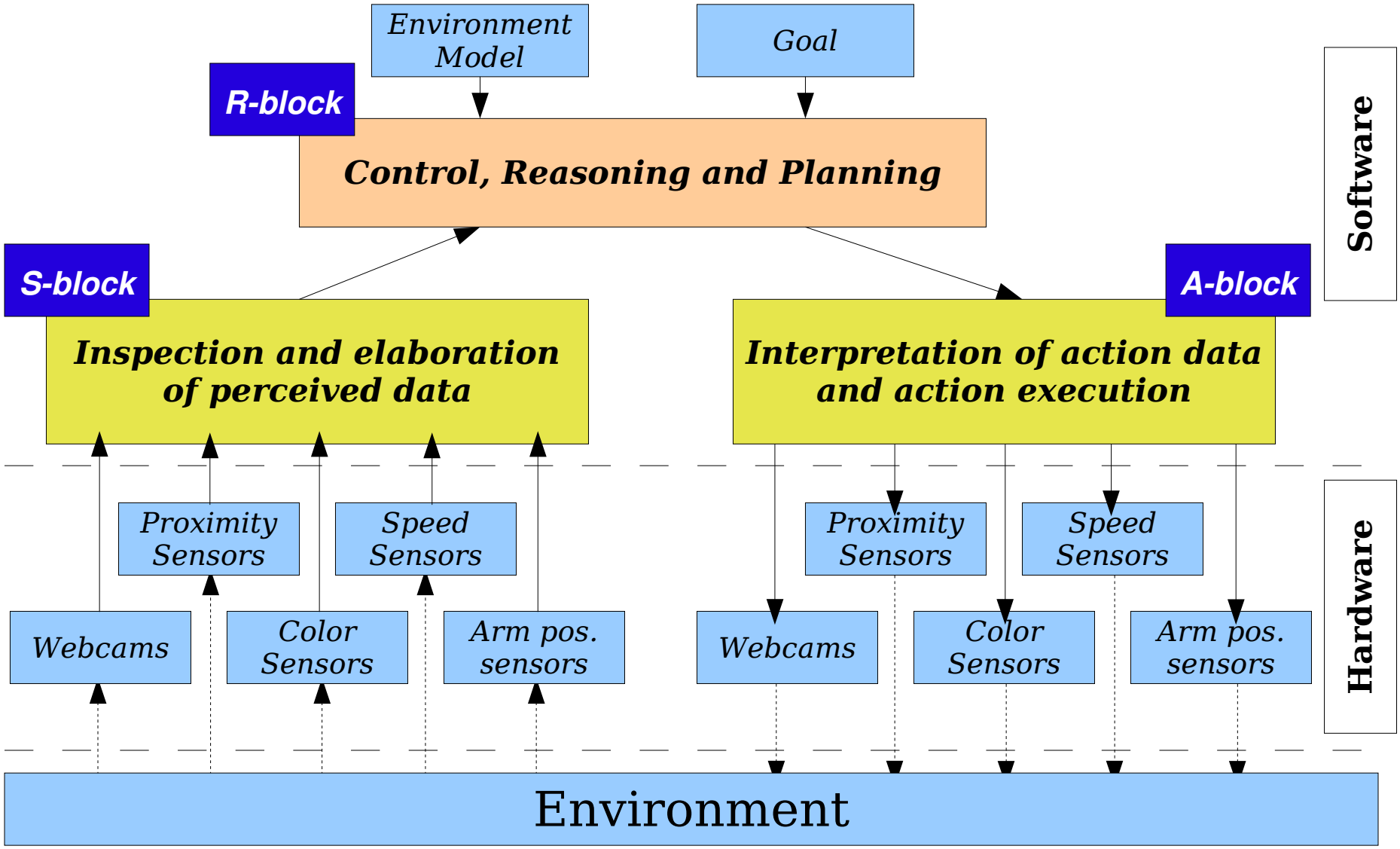
Agenda

- **Background**
- **Problem Statement**
- **The Robotic Framework**
- **Case Study: the “Caesar” robot**
- **Conclusions**

Background

- Autonomous Mobile Robot concerns
  - **AI:** perception, adaptation, planning, reasoning
    - LISP, Prolog
  - **Control systems:** environment sensing, motion driving, FSM-based loops, etc.
    - C, C++, Java
- **Erlang is suitable for both!**

HW/SW Robot Structure

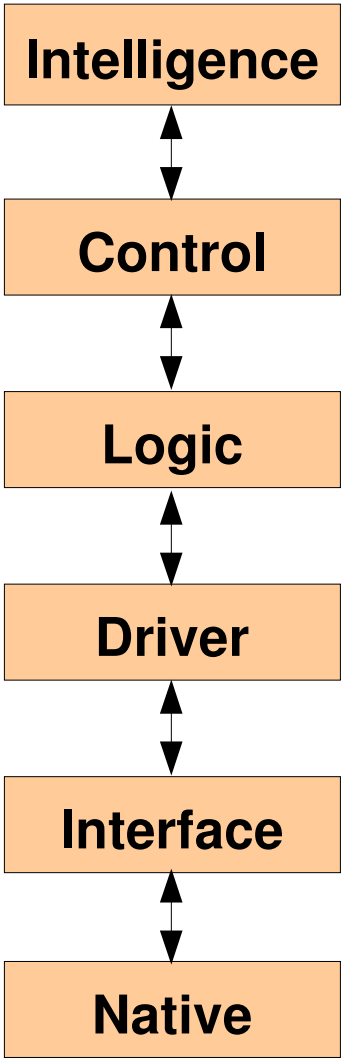


Issues to be faced

- A sensor/actuator may be replaced
  - different hardware and interface/same functionality
  - different functionality
- Control loops are interdependent
  - speed sensing/wheel control
  - position sensing/trajectory control
  - environment sensing/trajectory planning
- **Solution: use a layered architecture to separate each concern**

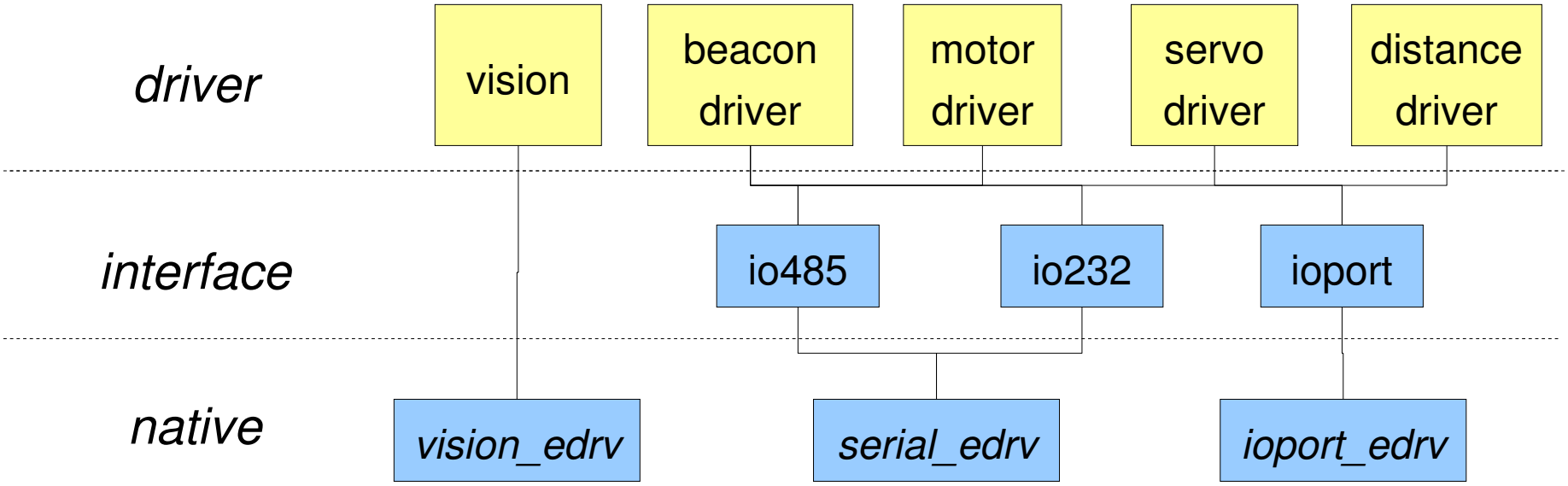
Basic Architecture

- **Planning and reasoning**
- **Control actions**
- **Sensor/actuator functionalities**
- **Specific sensor/actuator driving**
- **Physical interface**
- **Physical interface (native code)**



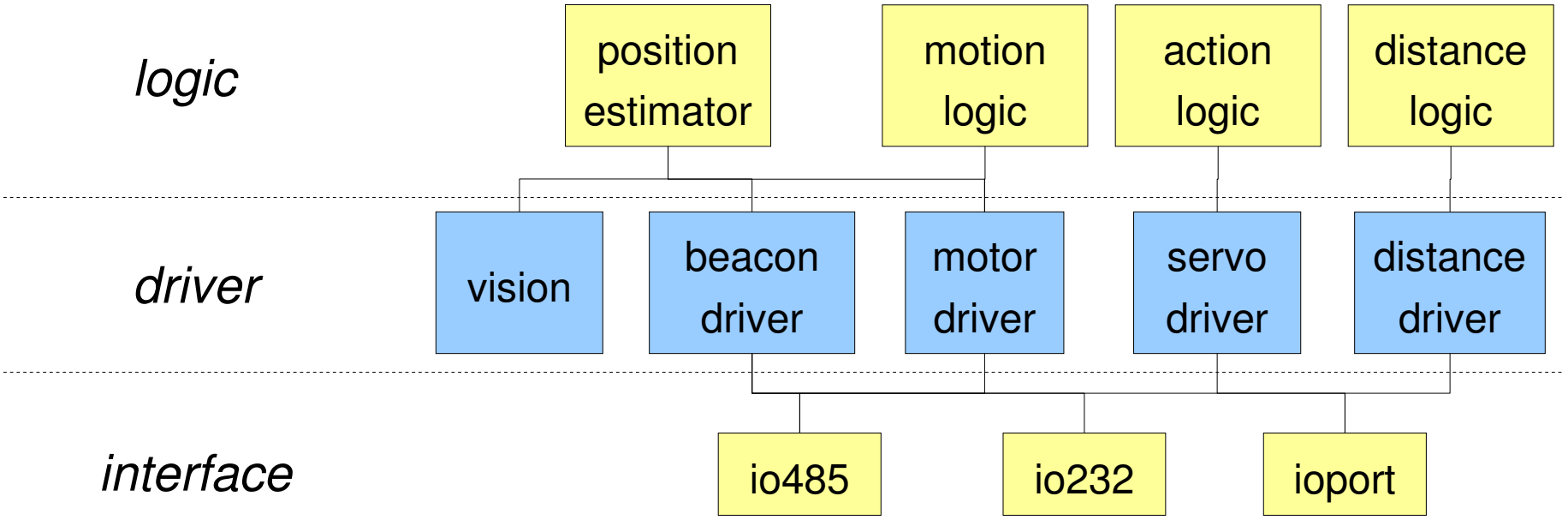
Native & Interface Layers

- Low-level details
- I/O lines interface
- Serial protocols (when needed)
- Services for the driver layer



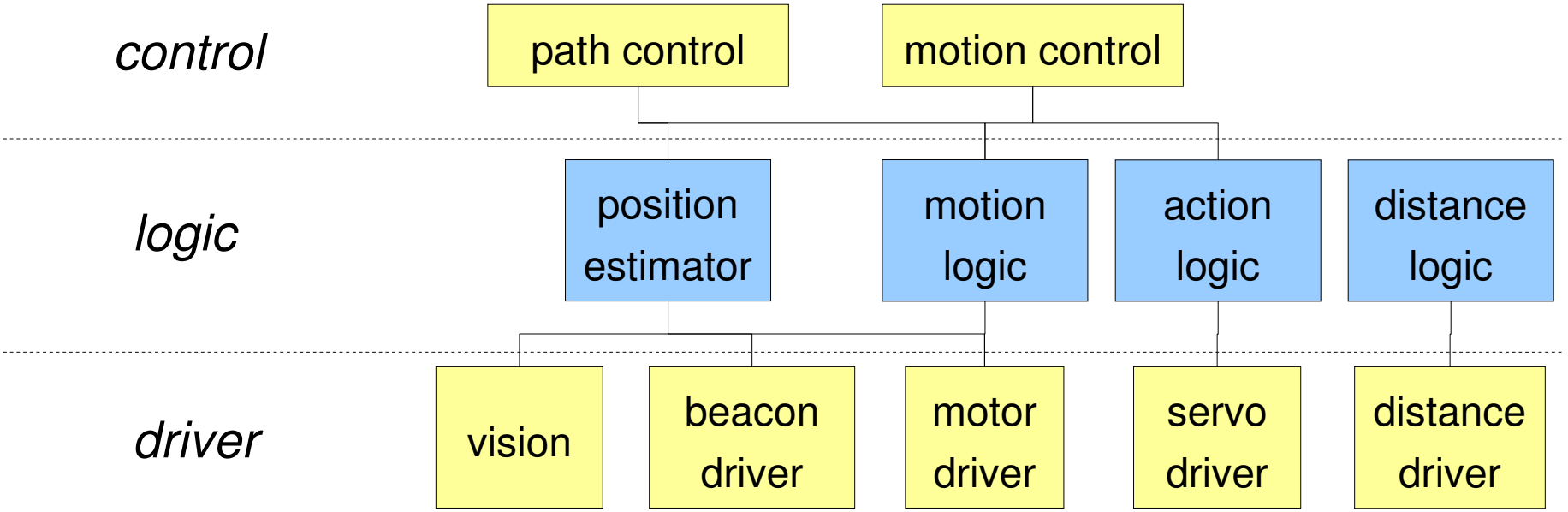
Driver Layer

- Device-specific details
- Different modules for each type of sensor/actuator
- Services for the logic layer



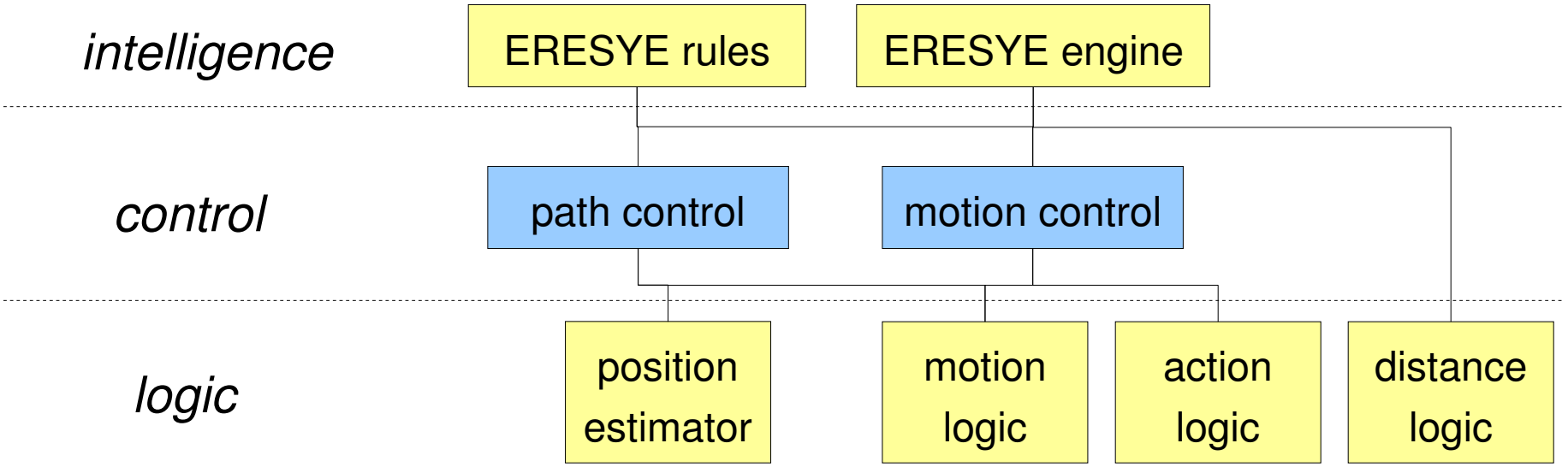
Logic Layer

- Device-independent sensing/action functionalities
- Different modules for each functionality
- Services for the control layer



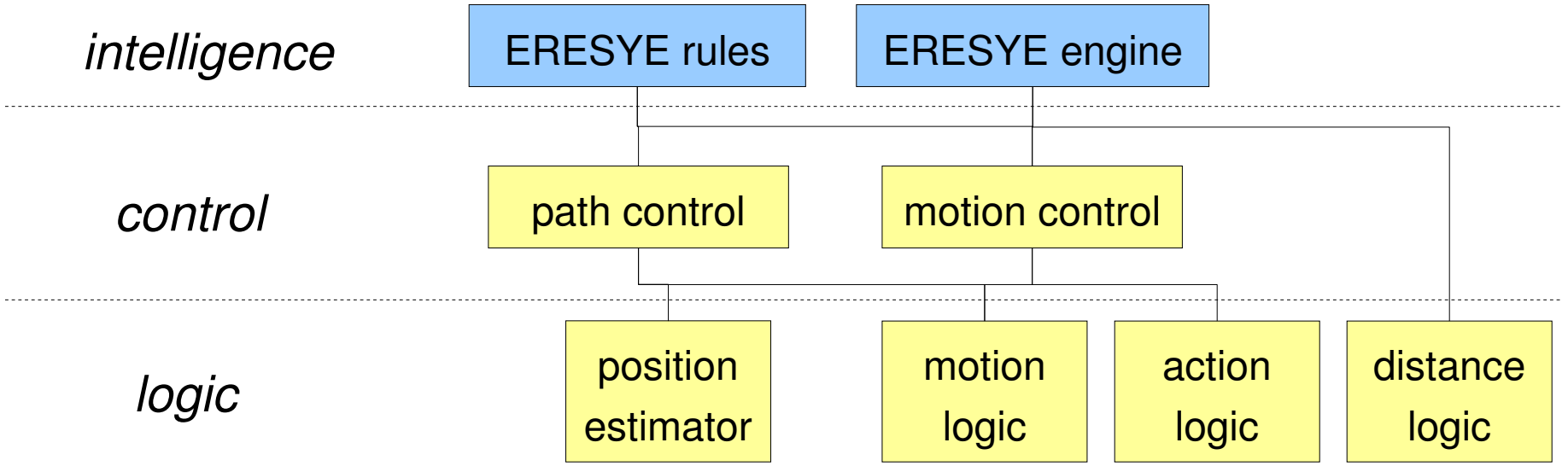
Control Layer

- Action control loops
- Trajectory control
- Mechanical Arms control
- Services for the intelligence layer



Intelligence Layer

- ERESYE: based on inference rules
- Interaction through fact assertion (KB)
- Environment Map
- Goal specification
- Planning and reasoning

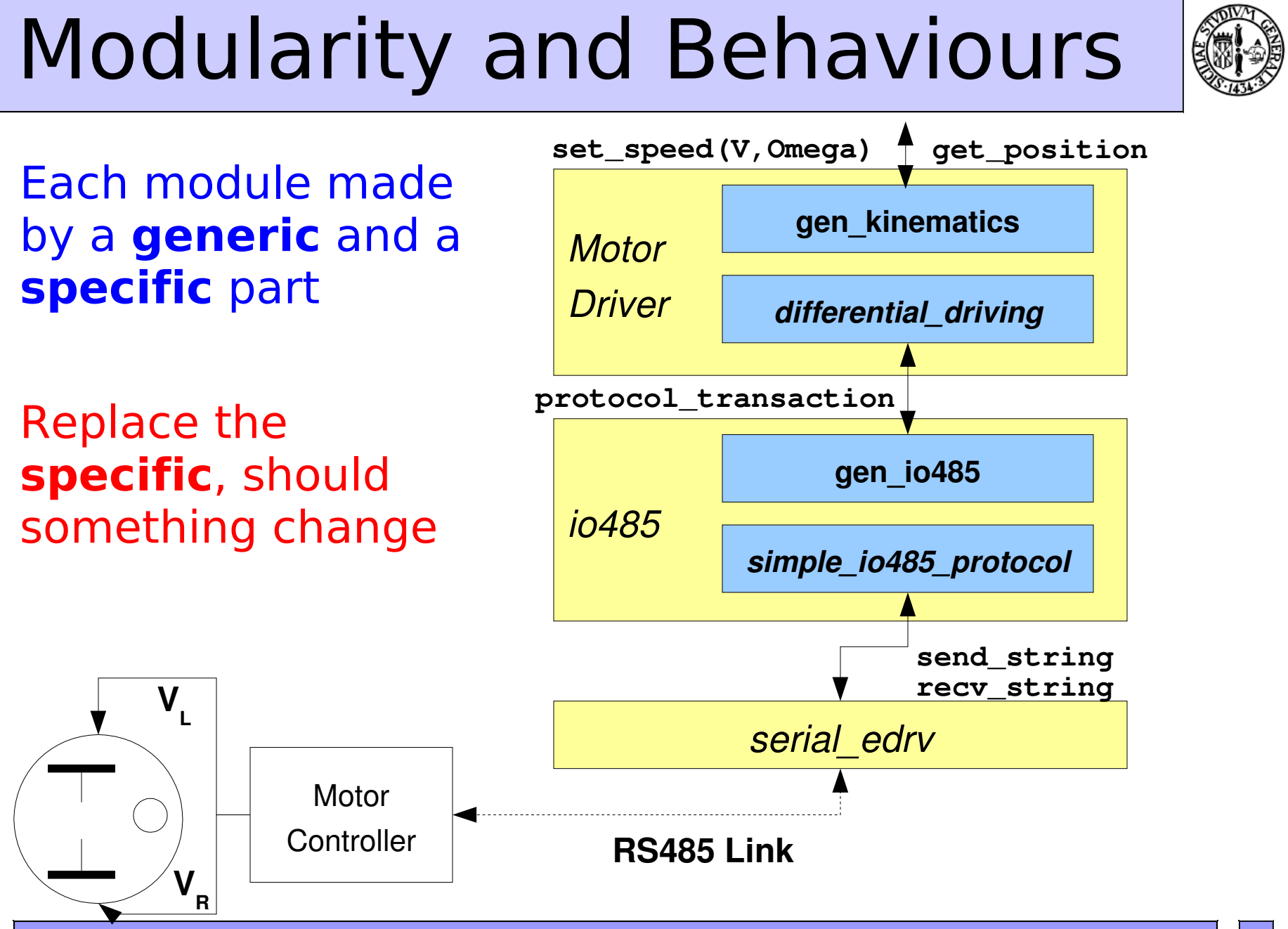


Modular Structure

- ... makes easier
  - hardware refactoring
    - intervene only at the *driver* level
  - software refactoring
    - only involved modules have to be patched
  - software reuse
    - many modules can be used “as-is” for different mobile robot applications

- **Each module is a “behaviour”, according to OTP**

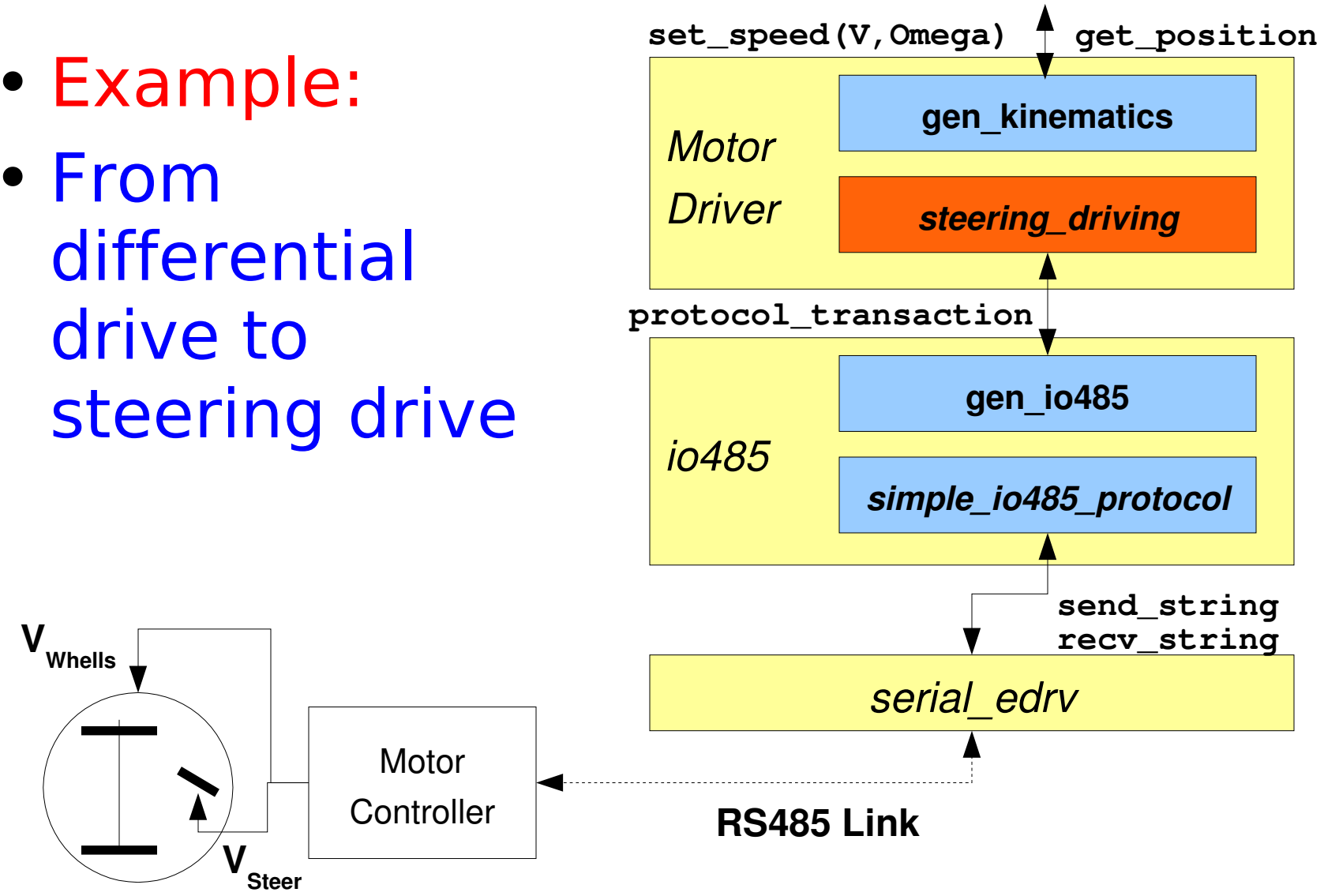
Modularity and Behaviours



- Each module made by a **generic** and a **specific** part
- Replace the **specific**, should something change

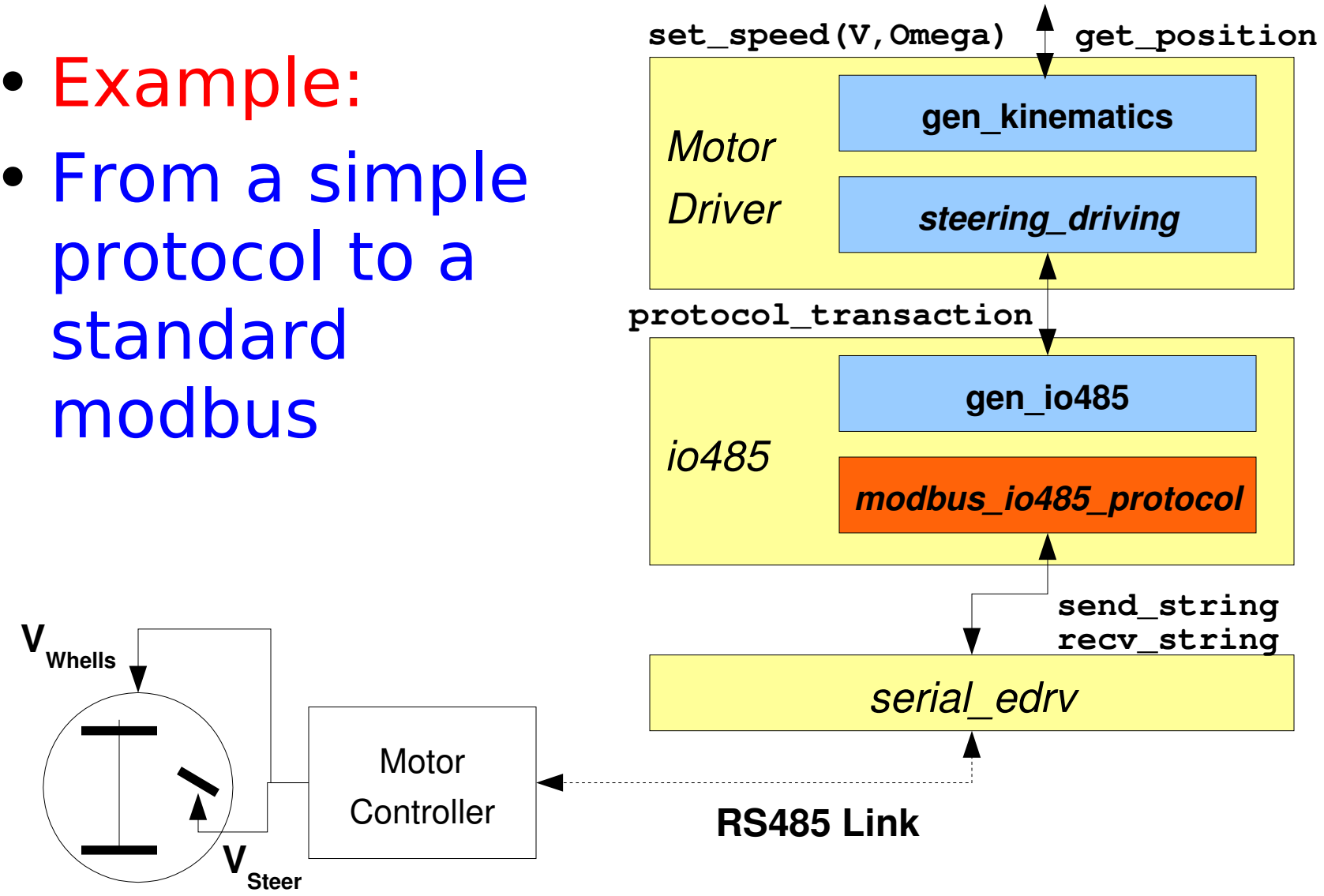
Modularity and Behaviours

- Example:
- From differential drive to steering drive

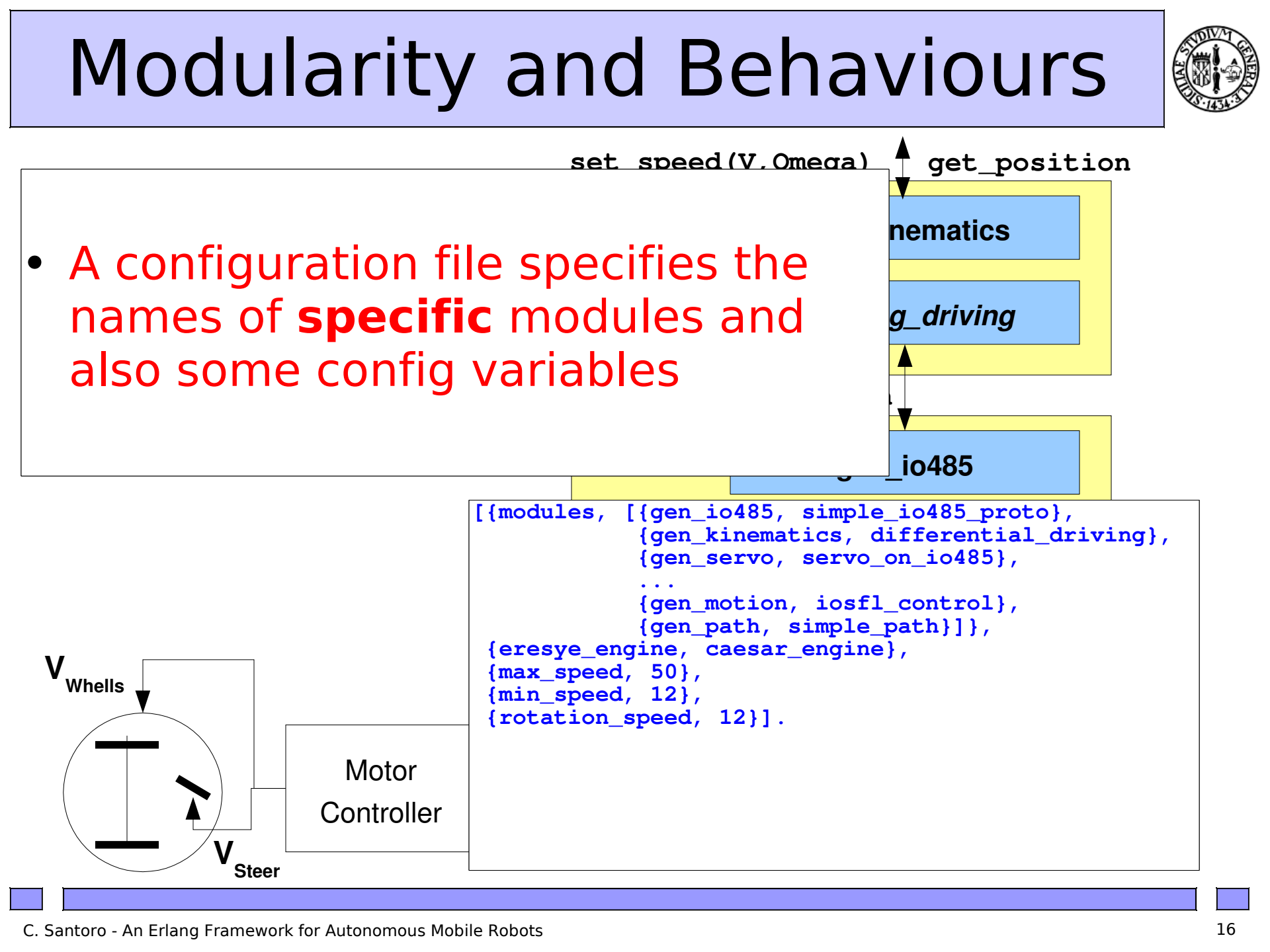


Modularity and Behaviours

- Example:
- From a simple protocol to a standard modbus



Modularity and Behaviours



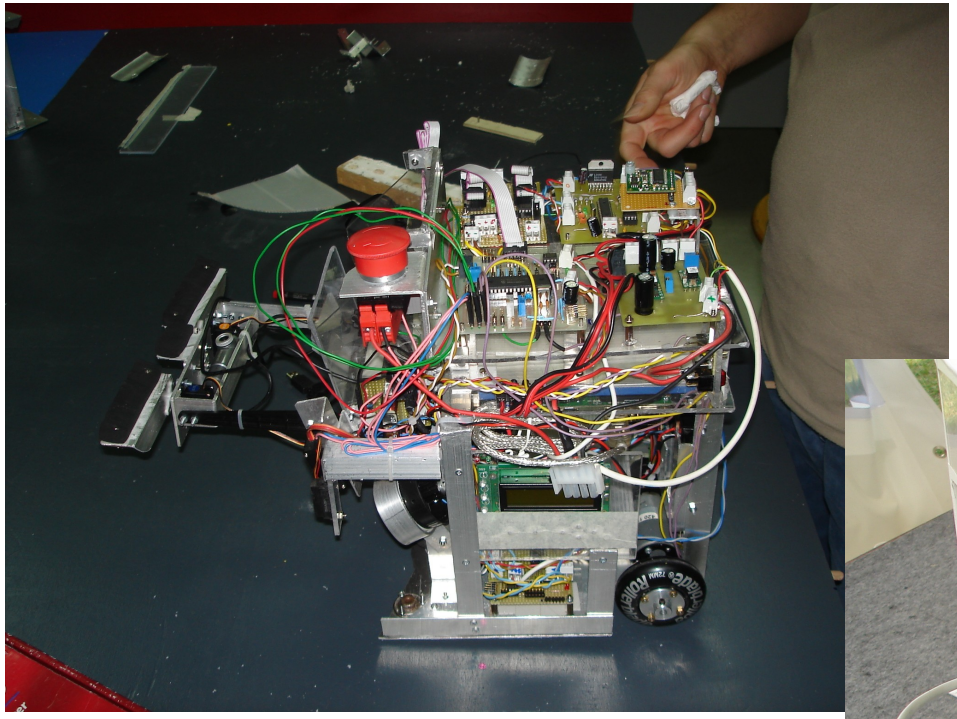
- A configuration file specifies the names of **specific** modules and also some config variables

Case Study: “Caesar”, the Robot

- Designed for **Eurobot 2007 Competition**

- Two robots collecting waste
- Three types of waste:
  - Cans (yellow)
  - Bottles (transparent with a green stripe)
  - Batteries (red and blue)

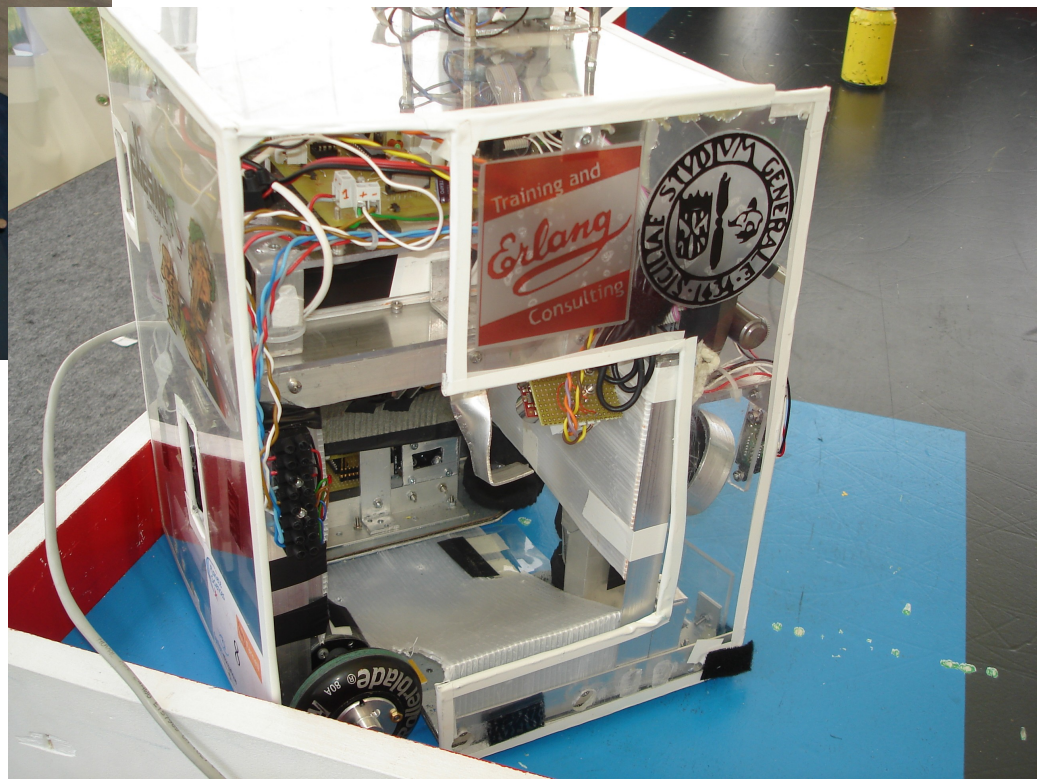
The Robot



Training and Erlang Consulting

Si.a.tel
Sistemi Automazione Telecontrollo

CAESAR
DIIT TEAM - ITALY



The Team!

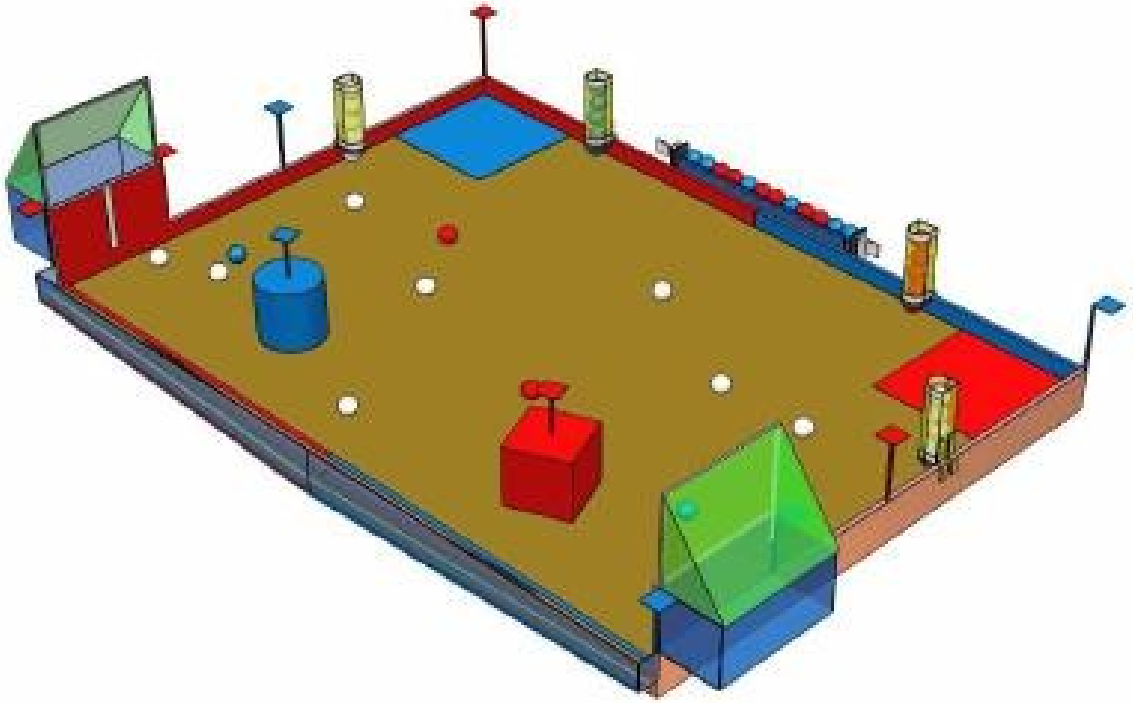

SORTIE DE SECOURS
CAESAR
DIIT Team
1st among Italian teams
14th (on 37) in International Championship
:34

Conclusions

- Erlang symbolic/functional programming is very effective.

- Thanks to the modular structure of the robotic framework, it was very simple to do:
  - fast prototyping
  - to change (also substantially) just-in-time
    - the strategy
    - the robot's structure
    - the behaviour of sensors and actuators

Ready for Eurobot 2008



- **Mission to Mars!**
- Find proofs of life (coloured balls) and bring them to Earth (baskets)