XBotCore: A Real-Time Cross-Robot Software Platform

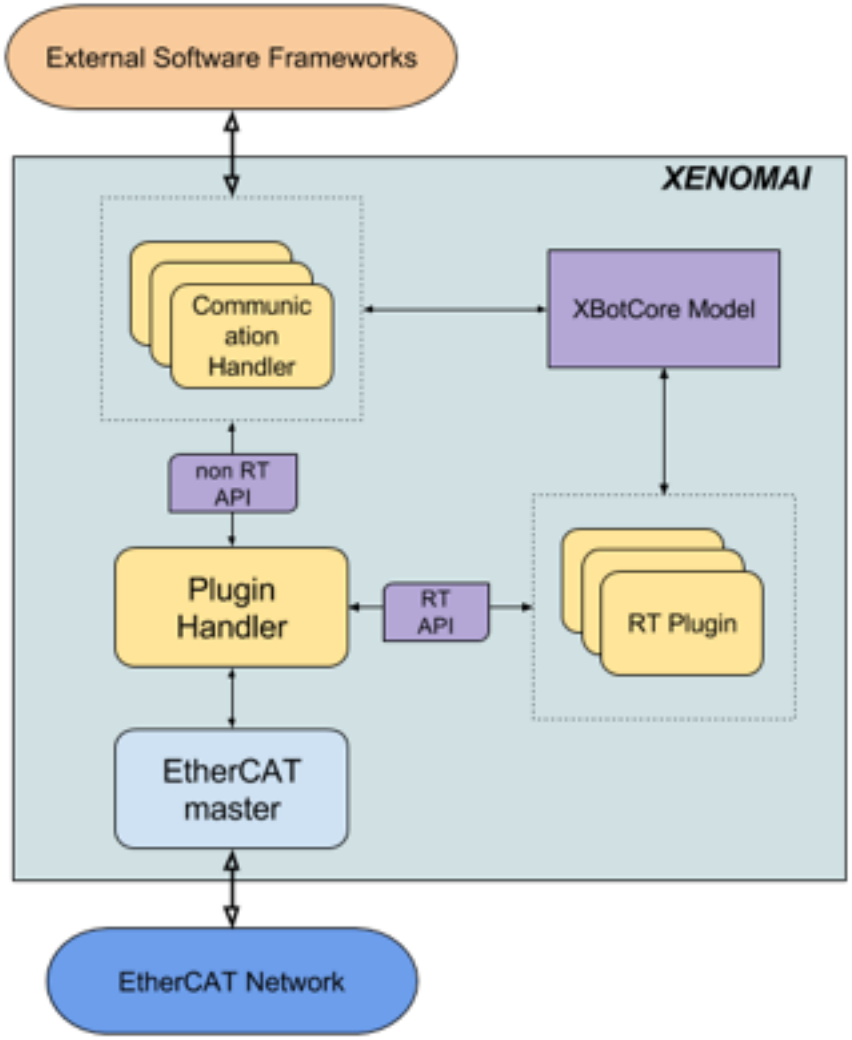
Towards Humanoid Robots OS -
Full-day Workshop
Humanoids 2016,

Luca Muratore (IIT)

The Westin Resort & Spa, Cancun, Mexico,
15th November 2016

iit ADVR
ISTITUTO ITALIANO DI TECNOLOGIA

XBotCore

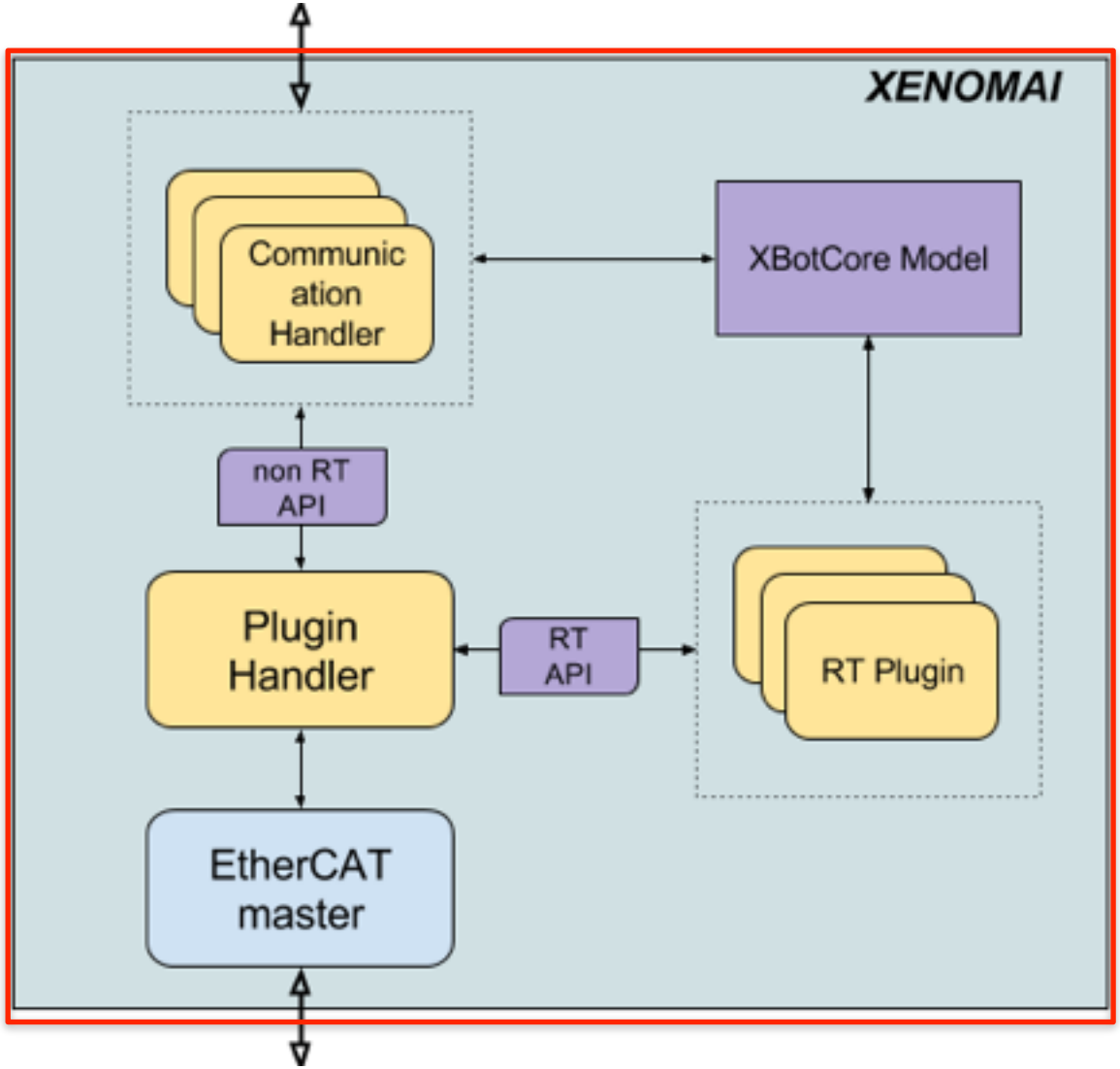


- **XBotCore**
  - XBotCore = XENOMAI RT development framework.
  - XBotCore = cross-robot
  - XBotCore = Core libraries and middleware functionalities.

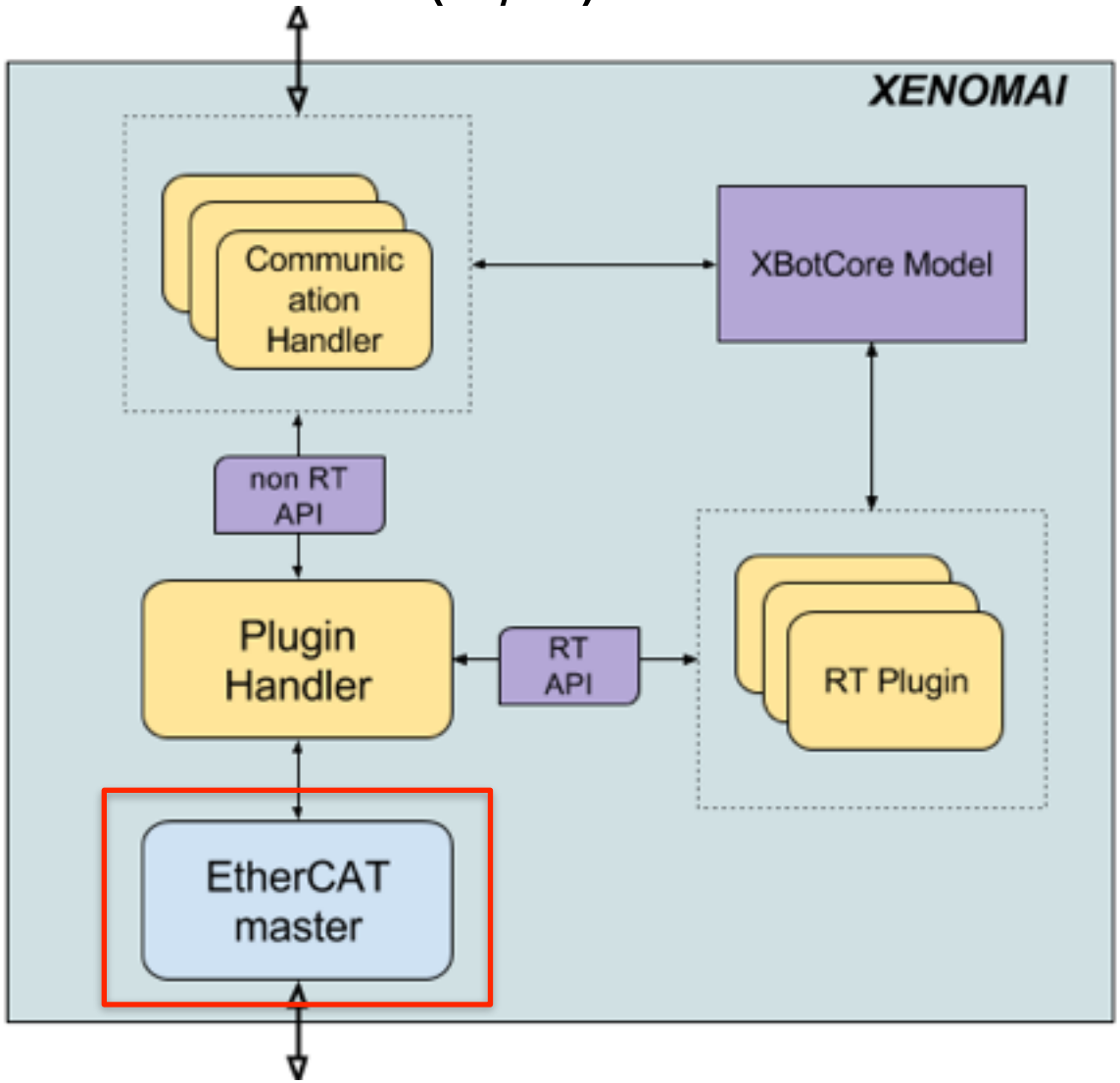
XBotCore Design Goals

- Hard RT control system
- 1 KHz control frequency
- Cross-Robot compatibility
- External framework integration
- Plug-in architecture
- Light-weight
- Simplicity
- Flexibility
- Open-source

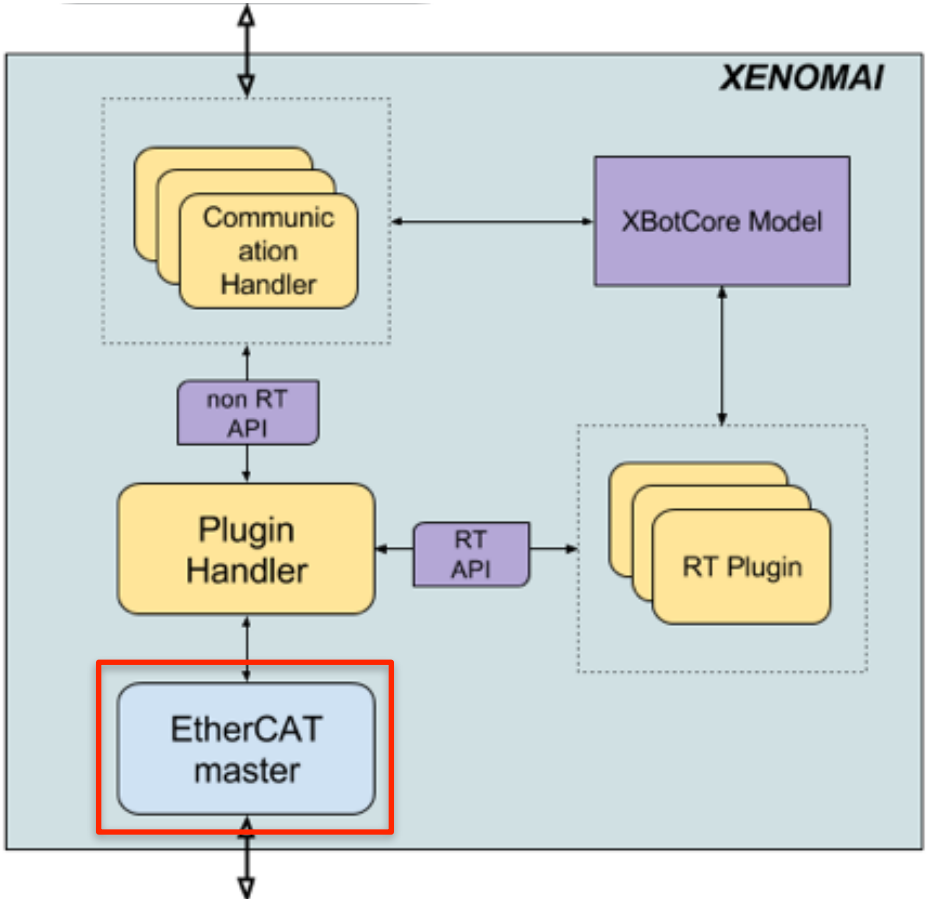
XBotCore


XENOMAI
Communic
ation
Handler
XBotCore Model
non RT
API
Plugin
Handler
RT
API
RT Plugin
EtherCAT
master

EtherCAT master (1/3)

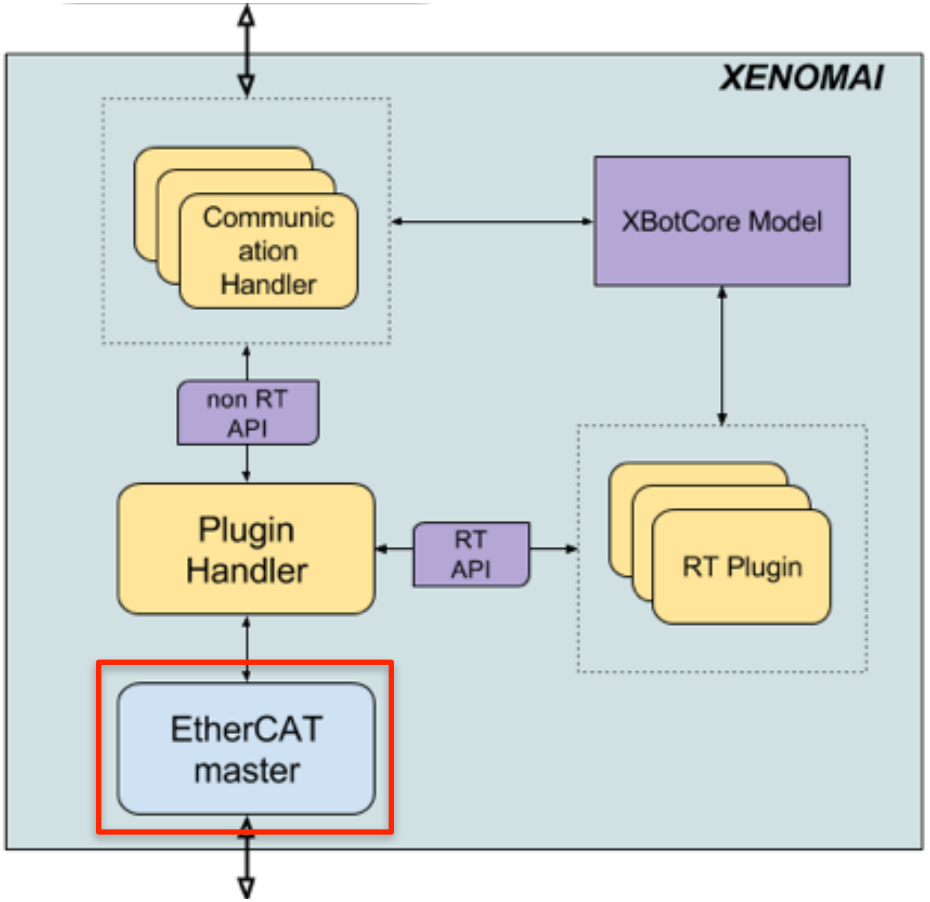


EtherCAT master (2/3)



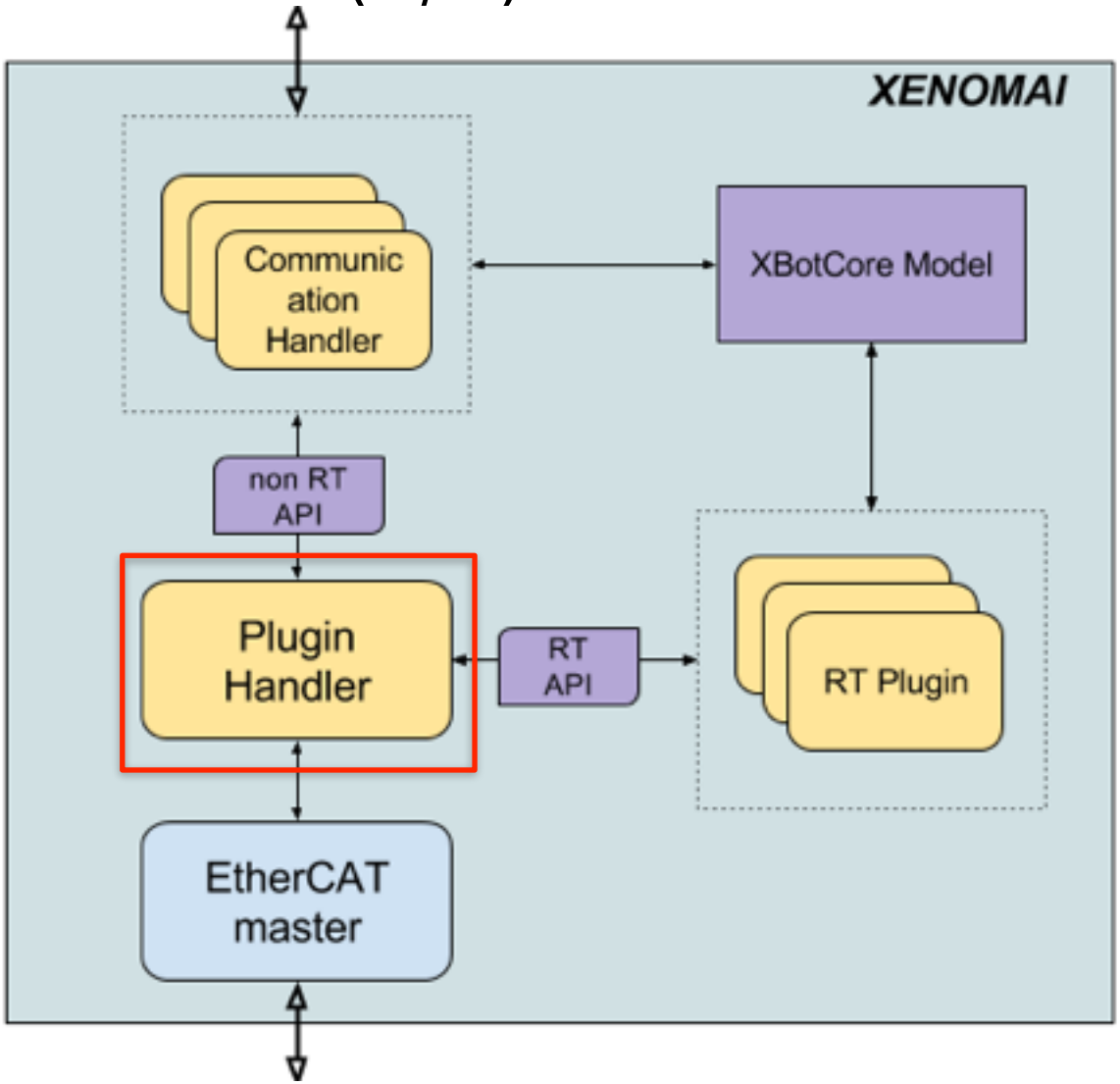
- **EtherCAT master**
  - Manages the EtherCAT slaves (i.e. electronic boards) and provides an asynchronous API to the higher level.
  - Developed starting from *SOEM* (*Simple Open EtherCAT Master*) library.
  - It can be used in not Real-Time or Real-Time mode.
  - EtherCAT State Machine: mailbox communication (SDO) vs process data streaming communication (PDO).

EtherCAT master (3/3)

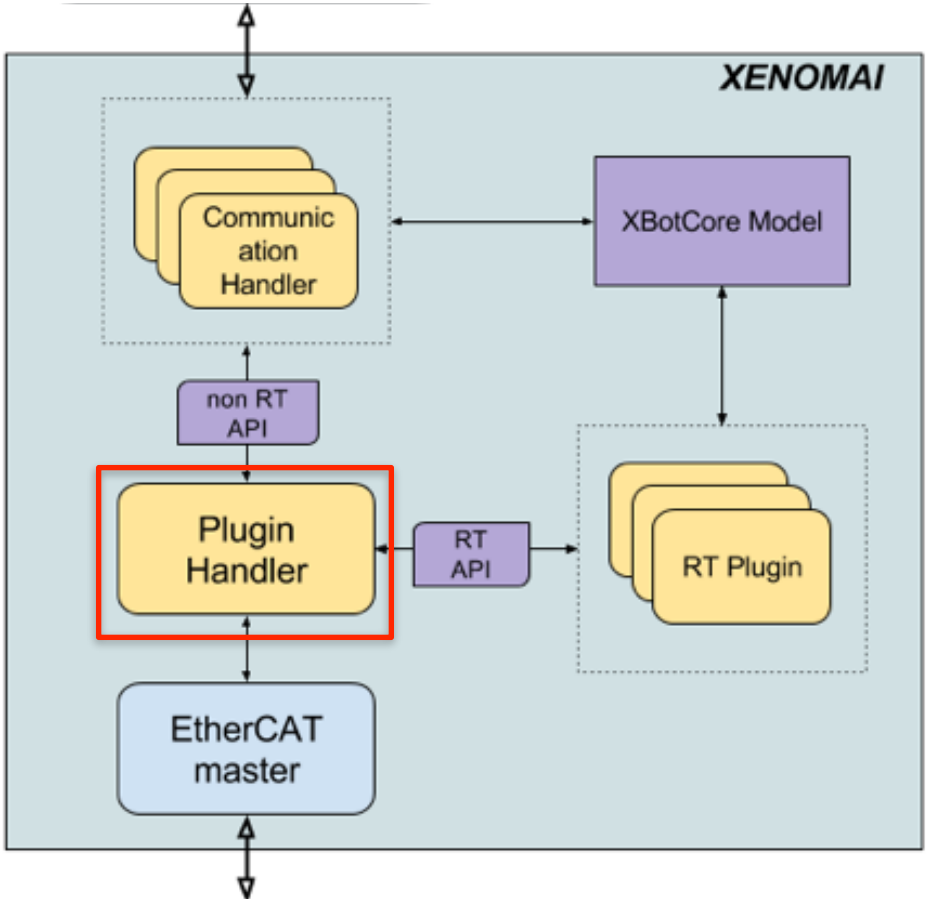



```
struct McEscPdoTypes {
    // TX  slave_input -- master output
    struct pdo_tx {
        float       pos_ref;     //link
        int16_t     vel_ref;     //link
        int16_t     tor_ref;     //link
        uint16_t    gains[5];
        uint16_t    fault_ack;
        uint16_t    ts;
        uint16_t    op_idx_aux; // op [get/set] , idx
        float       aux;        // set value

    }  __attribute__((__packed__));  // 28 bytes

    // RX  slave_output -- master input
    struct pdo_rx {

        float       link_pos;           // rad
        float       motor_pos;          // rad
        float       link_vel;           // rad TBD on the
        int16_t     motor_vel;          // rad/s
        int16_t     torque;             // Nm
        uint16_t    max_temperature;    // C
        uint16_t    fault;
        uint16_t    rtt;                // us
        uint16_t    op_idx_ack;         // op [ack/nack]
        float       aux;                // get value or n

    }  __attribute__((__packed__)); // 28 bytes

}; // 56 bytes total
```

Plugin Handler (1/3)

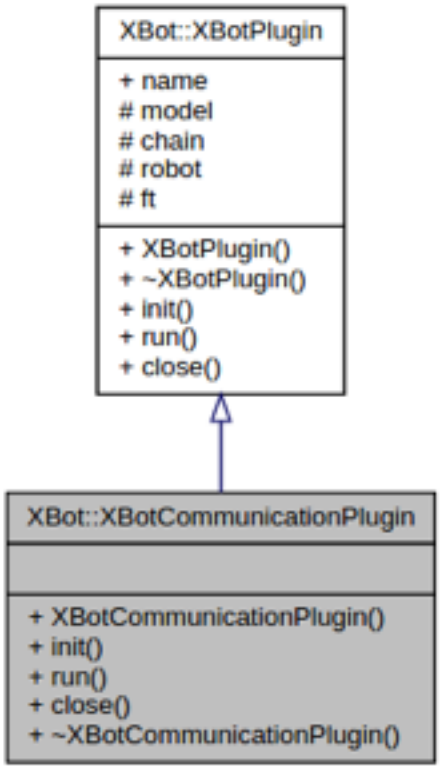


Plugin Handler (2/3)



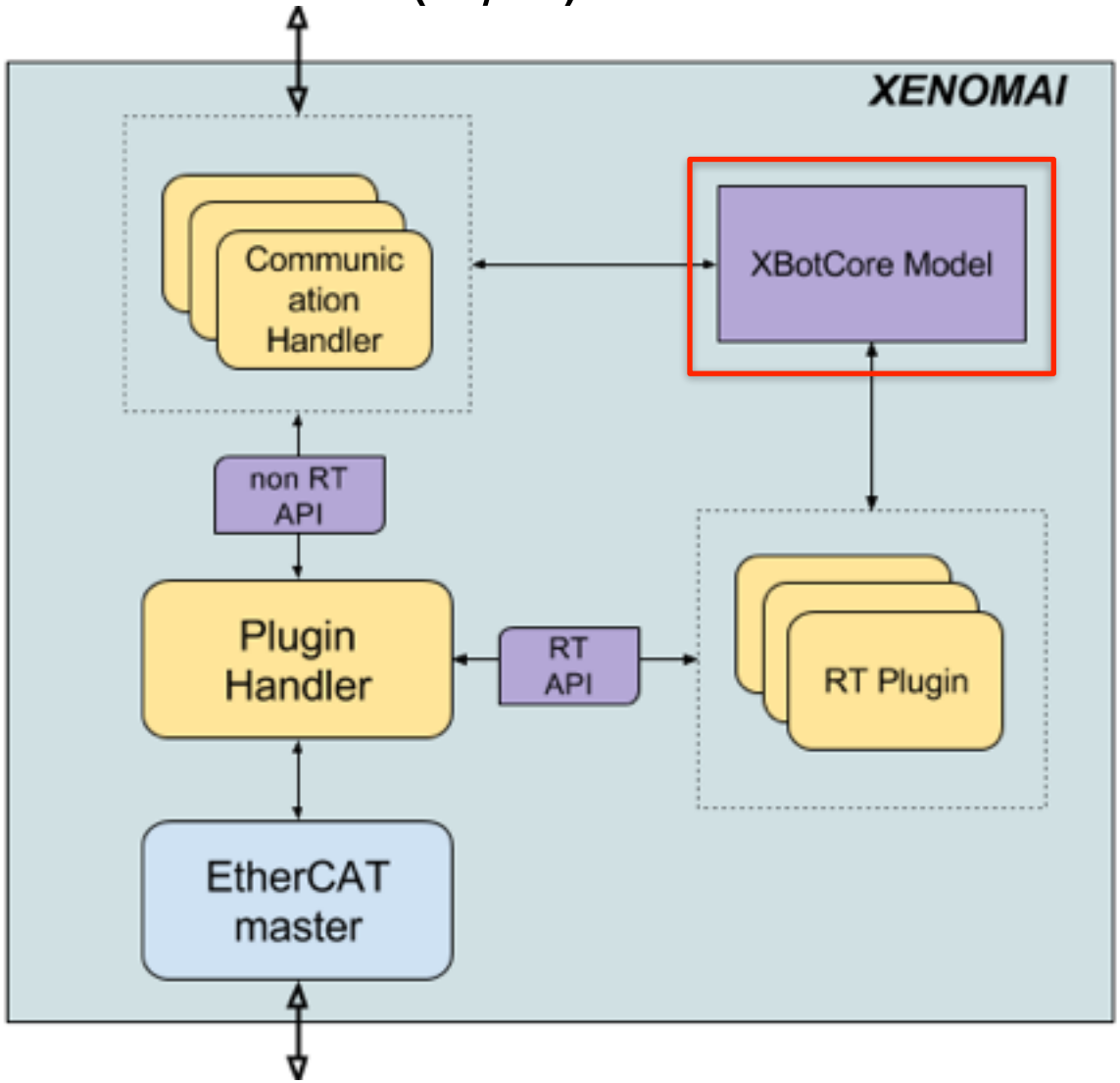
- **Plugin** = simple s.o.
  - Instance of an abstract class with init(), run() and close() methods.
  - It links against the R-T API to control the robot.
- **Plugin Handler**
  - a Real-Time thread that executes sequentially a set of Plugins.
  - it is possible to dynamically load and unload one or more plugins: it is responsible to start all the loaded plugins, execute them sequentially and close them before unloading them.

Plugin Handler (3/3)

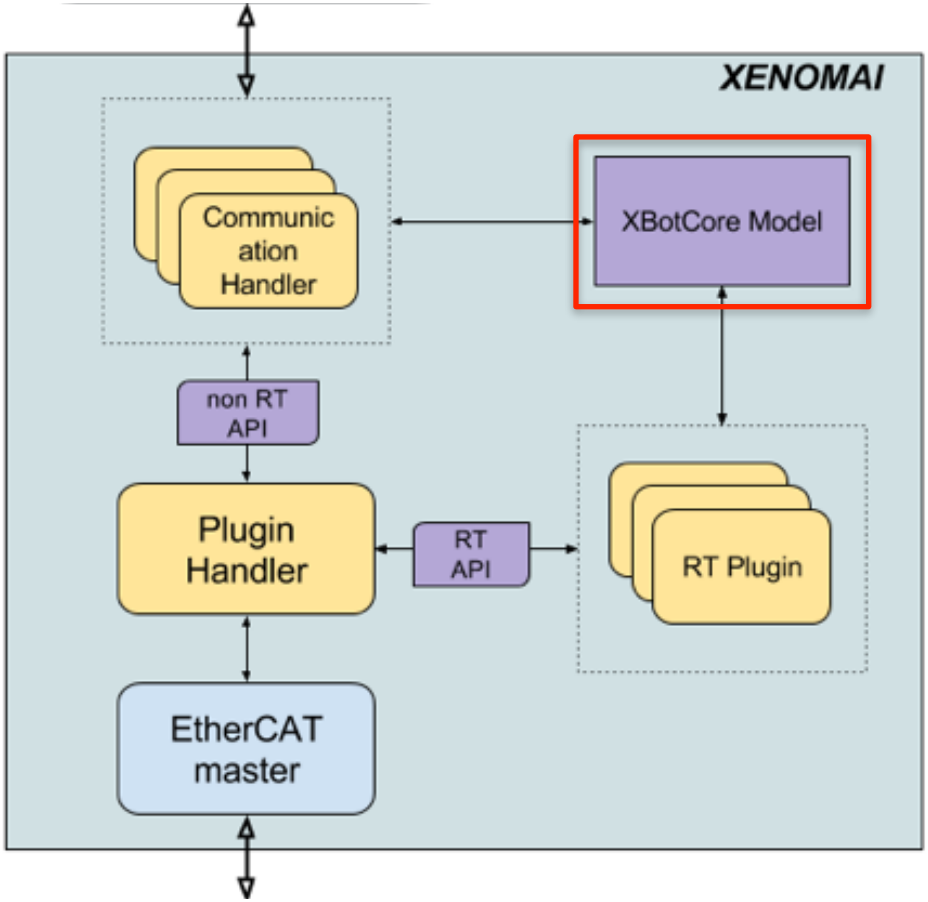


- RT Plugin example:
  - Implement init(), run(), close() functions using the RT API.
  - Test the plugin using the *GazeboXBotPlugin*.
  - Run it on the robot.

XBotCore Model (1/2)

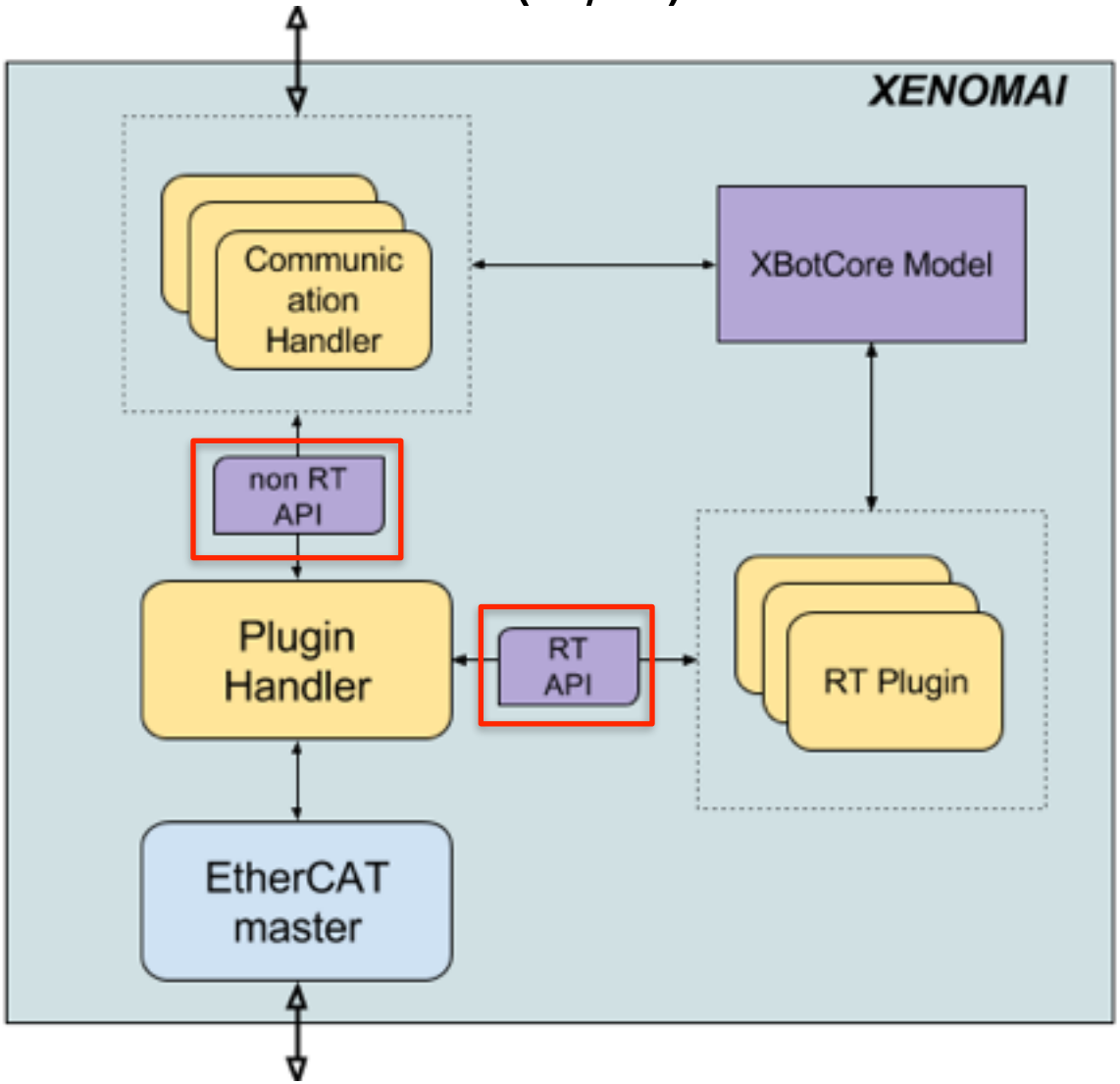


XBotCore Model (2/2)

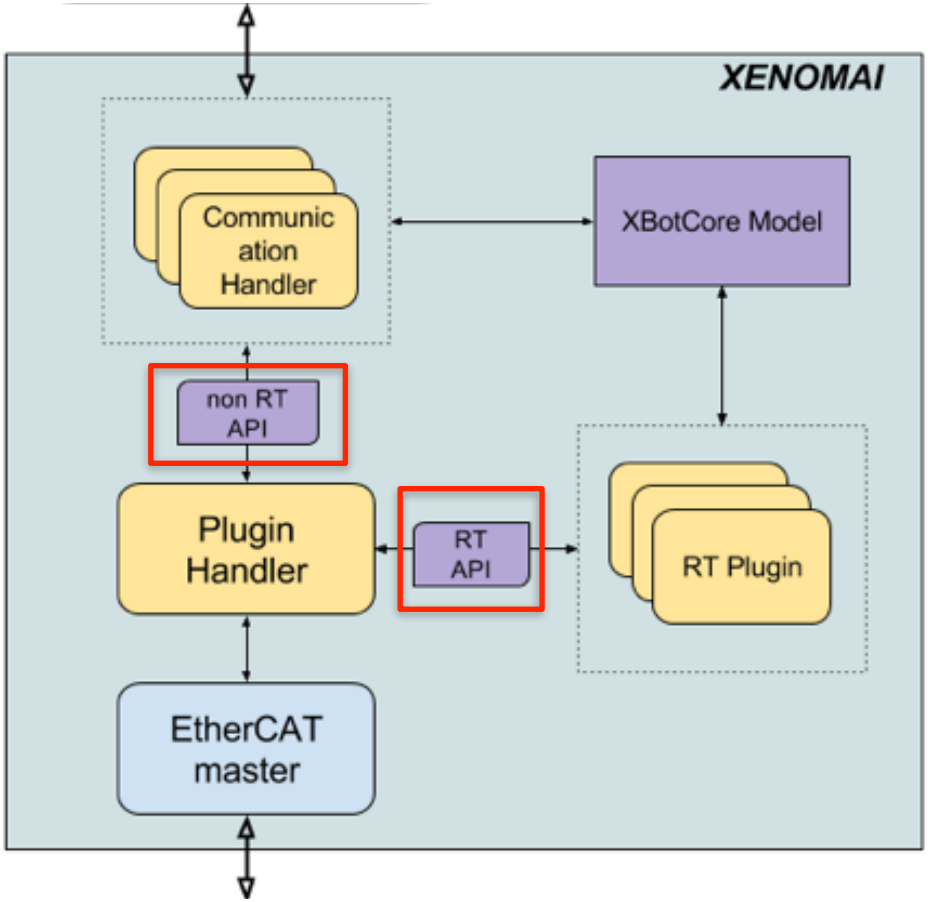


- **XBotCore Model**
  - Novel approach to the configuration of low-level control system by using:
    - URDF (Universal Robotics Description Format)
    - SRDF (Semantic Robotic Description Format)
  - Cross-Robot software platform
    - Robot API dynamically built starting from the input URDF and SRDF.

RT and non-RT API (1/3)

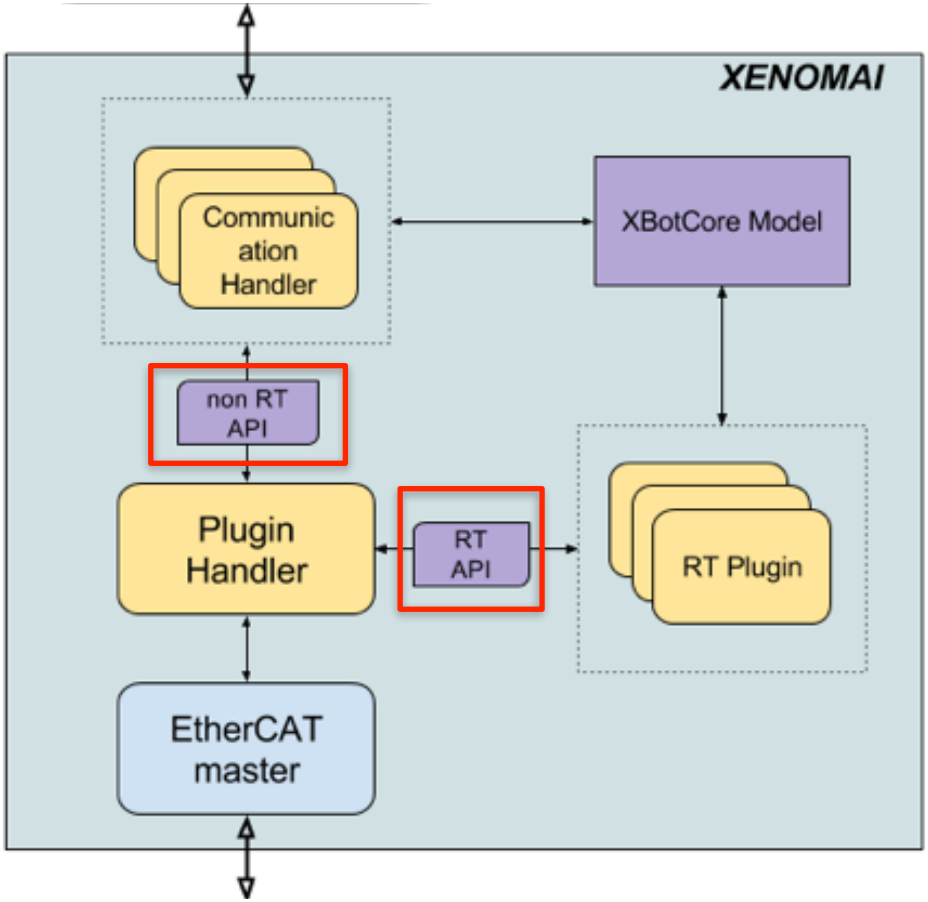


RT and non-RT API (2/3)



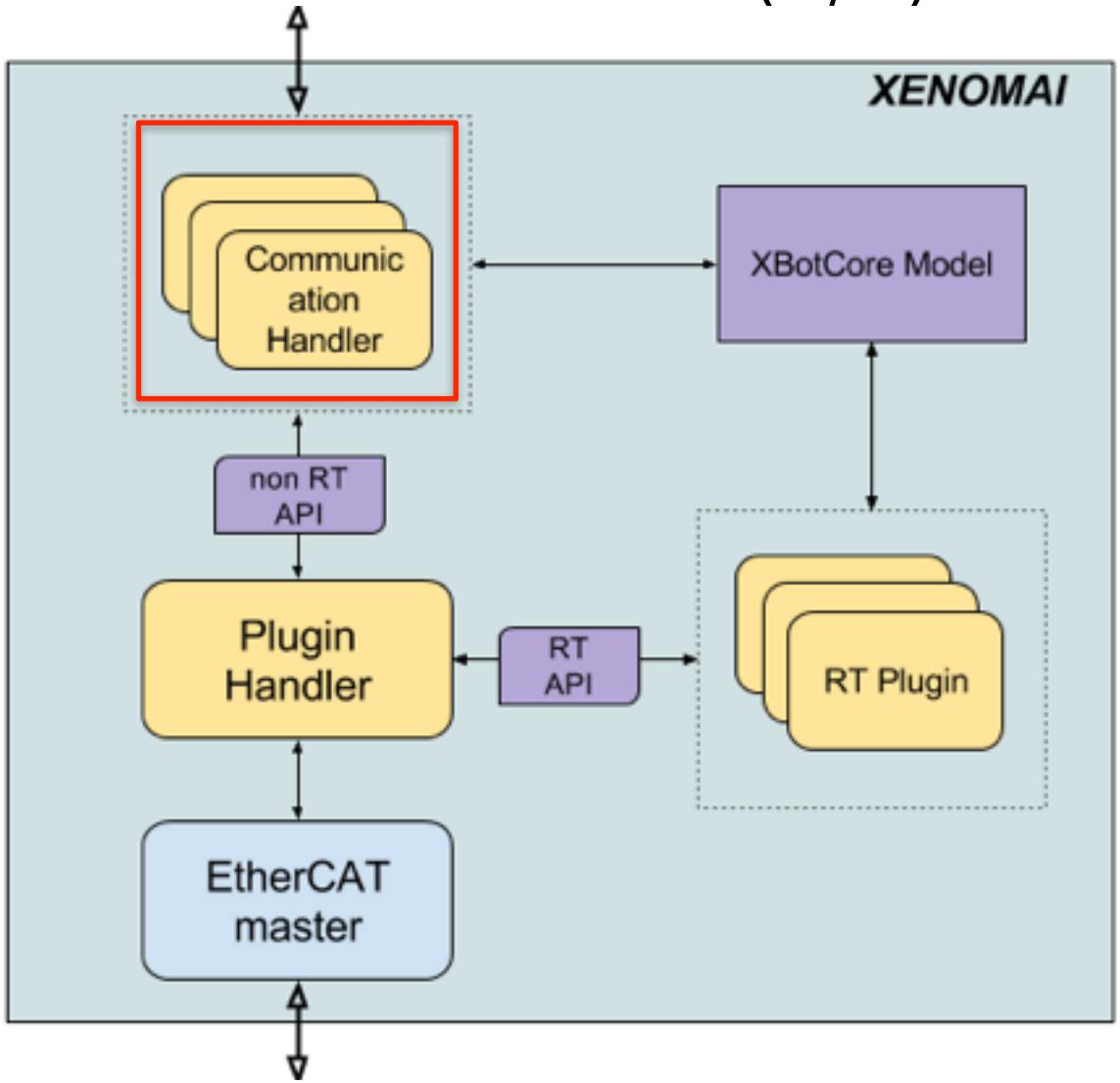
- **API interfaces**
  - IXBotJoint
    - Abstraction of the robot joints with the getters and setters related to the single joint element.
  - IXBotFT
    - Abstraction of the robot Force-Torque sensors.
  - IXBotChain
    - Abstraction of the robot kinematic chain with getters and setters related to a collection of joints.
  - IXBotRobot

RT and non-RT API (3/3)

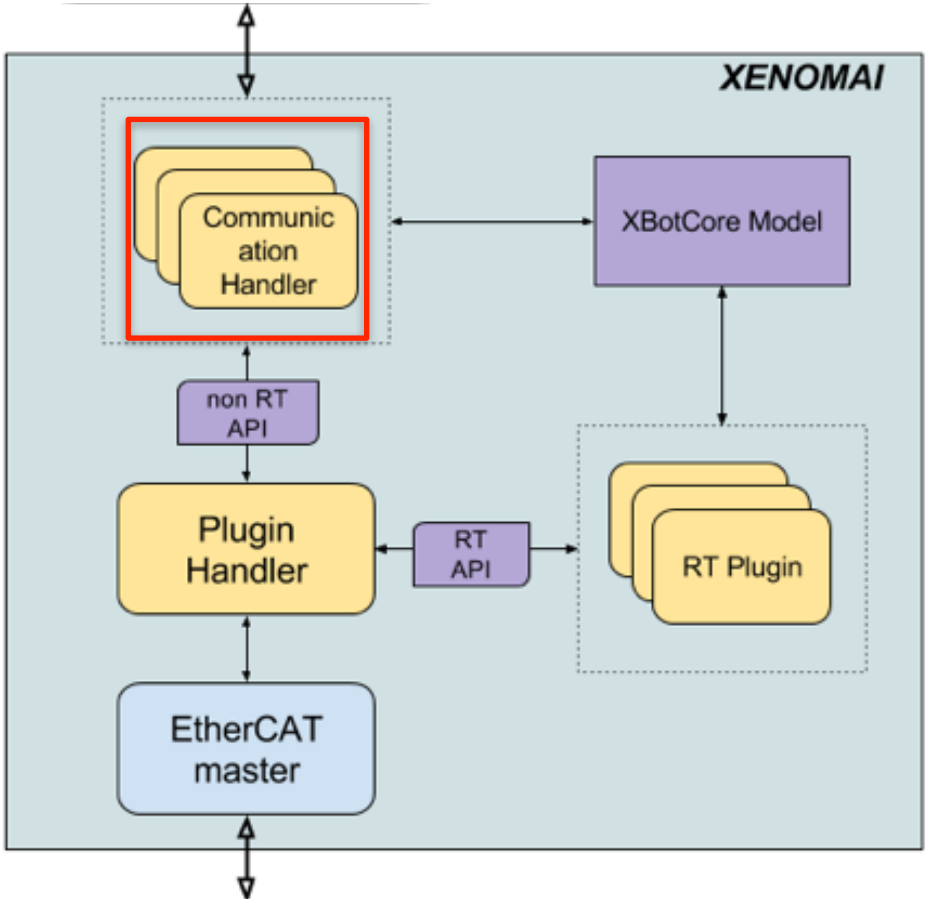


- **RT API**
  - Suitable for the RT plugins that will run in the Plugin Handler.
  - Shared memory communication mechanism.
- **non-RT API**
  - XDDP(Cross Domain Datagram Protocol) Xenomai pipes
    - Asynchronous communication between RT and N-RT threads.
    - Lock-free IPC (Inter-Process Communication)

Communication Handlers (1/2)



Communication Handlers (2/2)



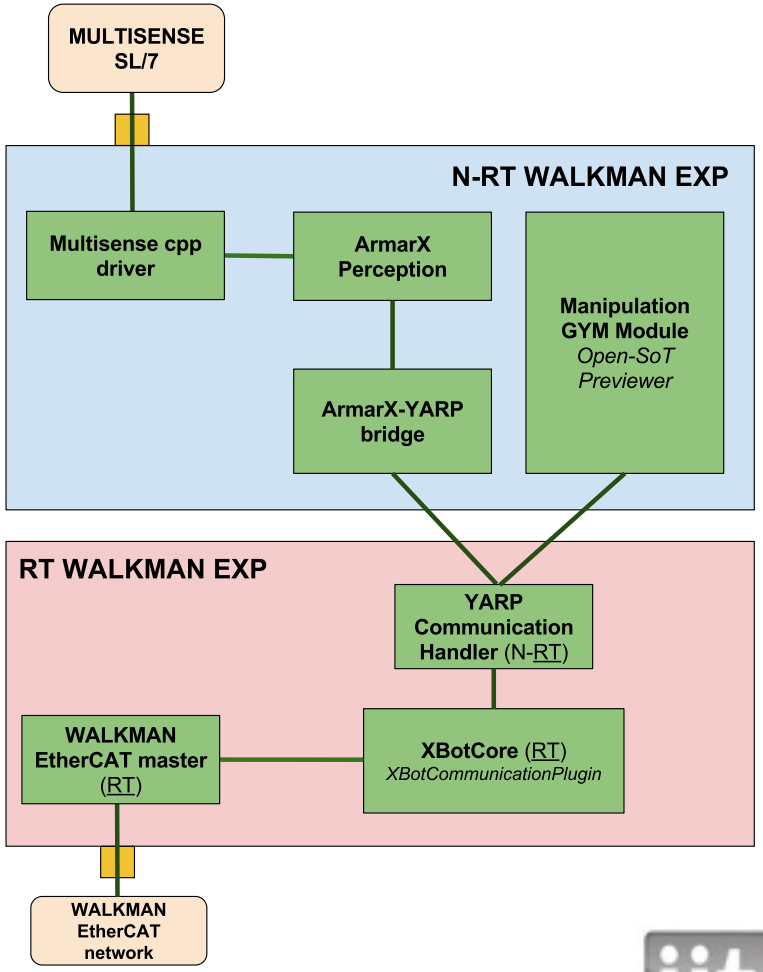
- **Communication Handlers**
  - A set of non-RT threads used to communicate with the external software frameworks.
  - *XBotCommunicationHandler* class is provided with ready-to-use non-RT API functions.
  - Built-in support for the YARP communication framework.
- **XBotYARP**
  - Dynamic control board wrappers and analog sensors allocation.
  - Same YARP interfaces (ports) for the simulation environment and the real robot.
- **XBotROS**

Experiments description (1/2)

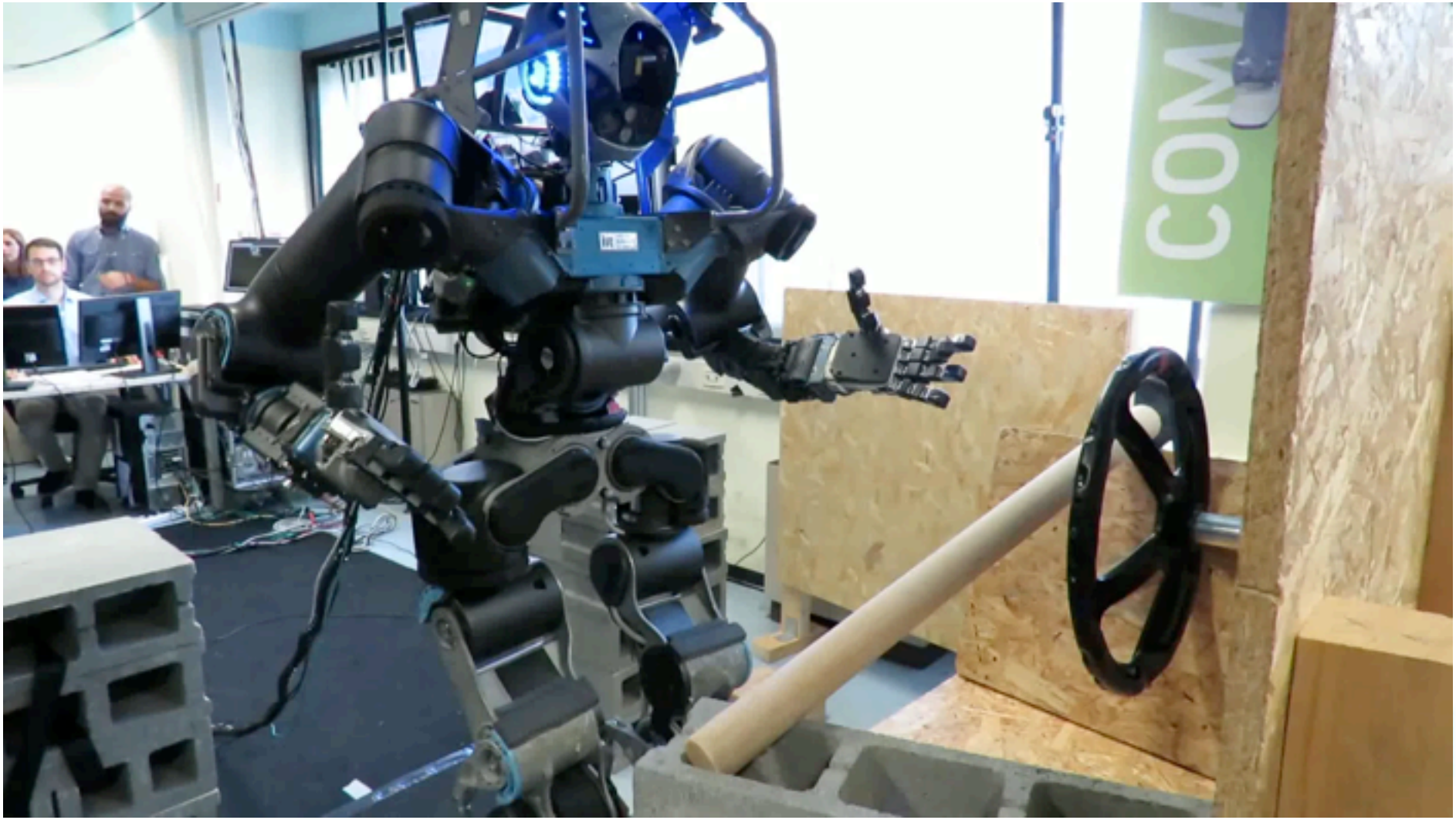
- Set of experiments on the WALK-MAN robot were performed:
  - full-size humanoid with 33 DOFs
  - 4 custom F/T sensors
  - CMU Multisense-SL sensor
- The experiments were carried out in a DRC-inspired scenario, targeting the removal of debris in front of a valve to turn.

Experiments description (2/2)

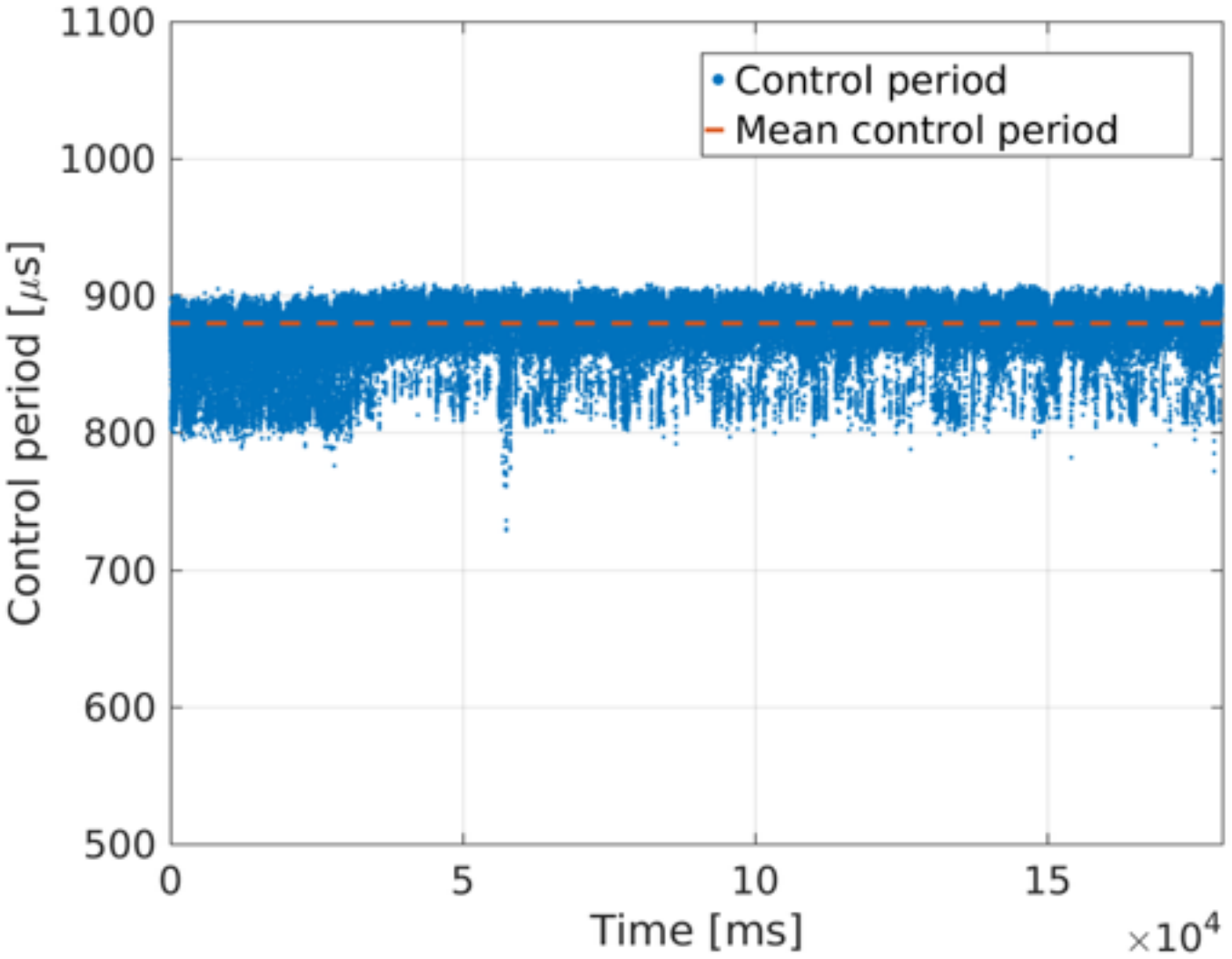
- In the evaluation different high-level software frameworks were successfully integrated on top of *XBotCore*:
  - Perception: ArmarX
  - Motion feasibility analysis and collision checking: Open-SoT previewer based on MoveIt!
  - Control module: YARP



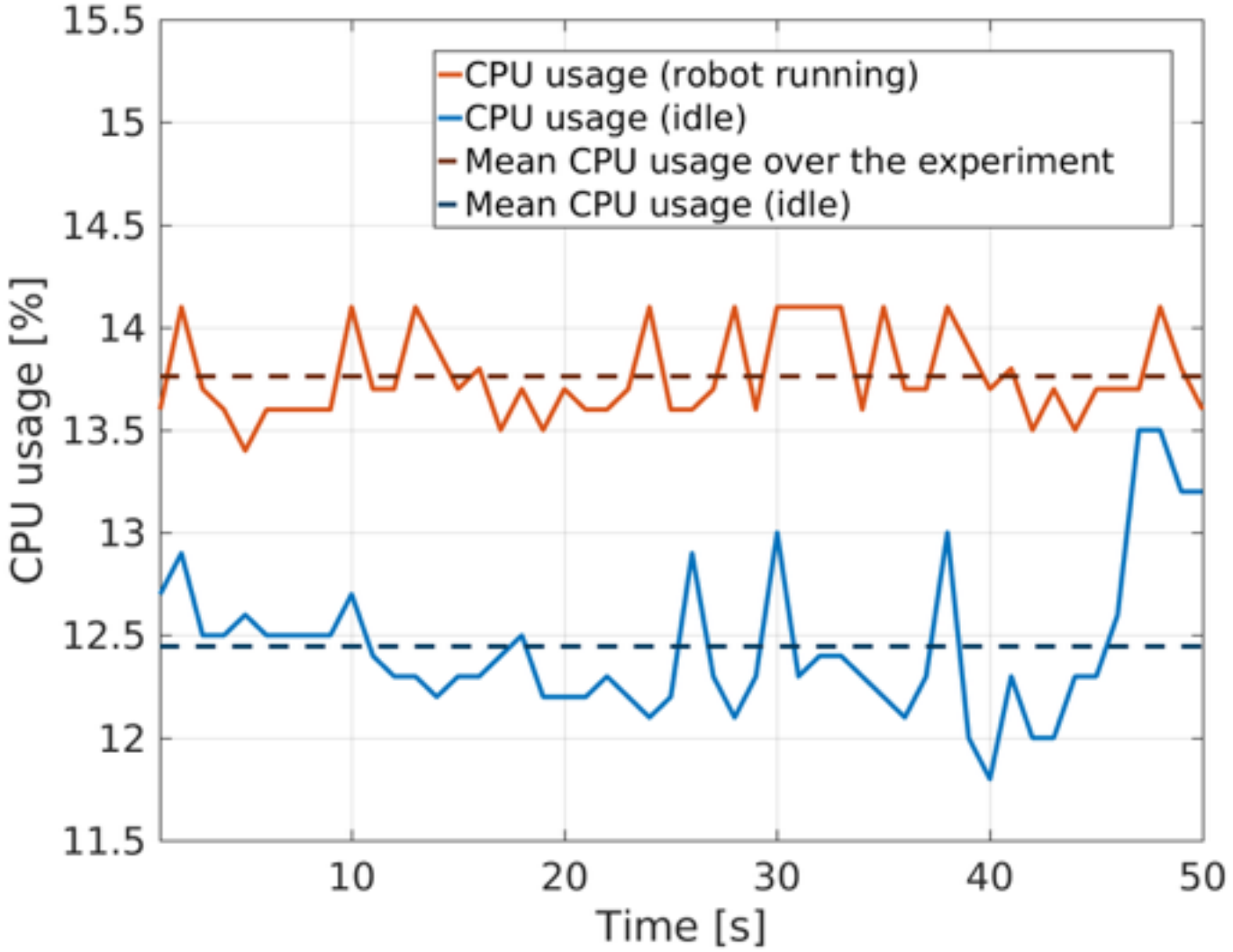
Experiments video



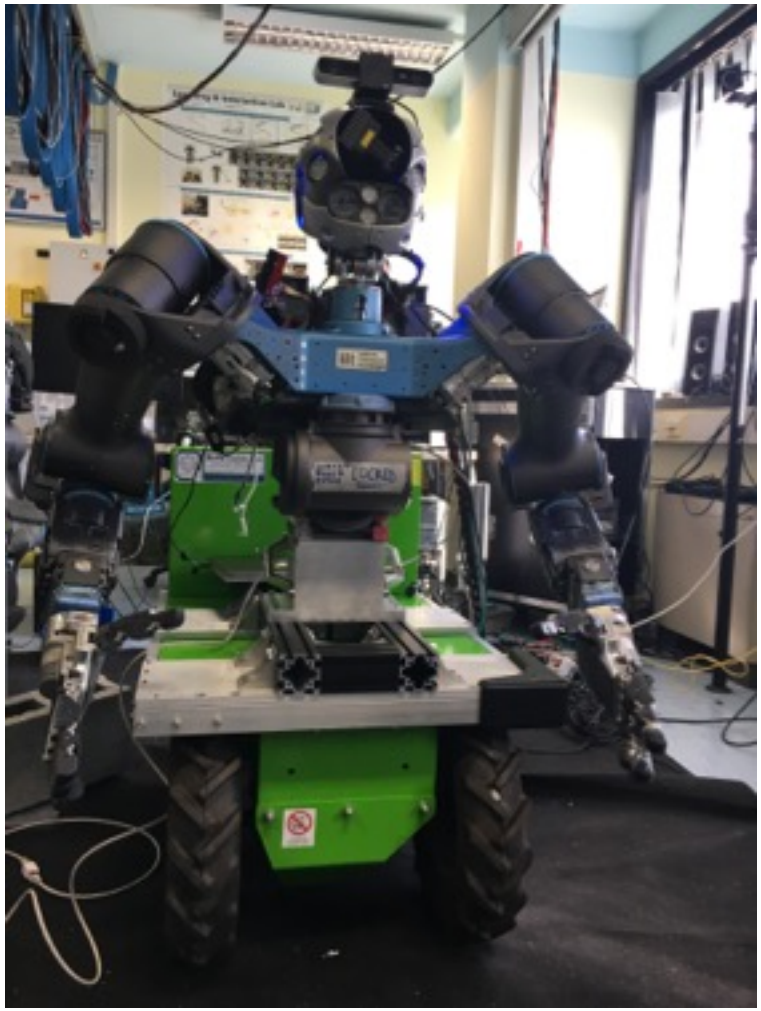
Experiments Results (1/2)



Experiments Results (2/2)



Other examples



CENTAURO

Q&A

Thank you for your attention