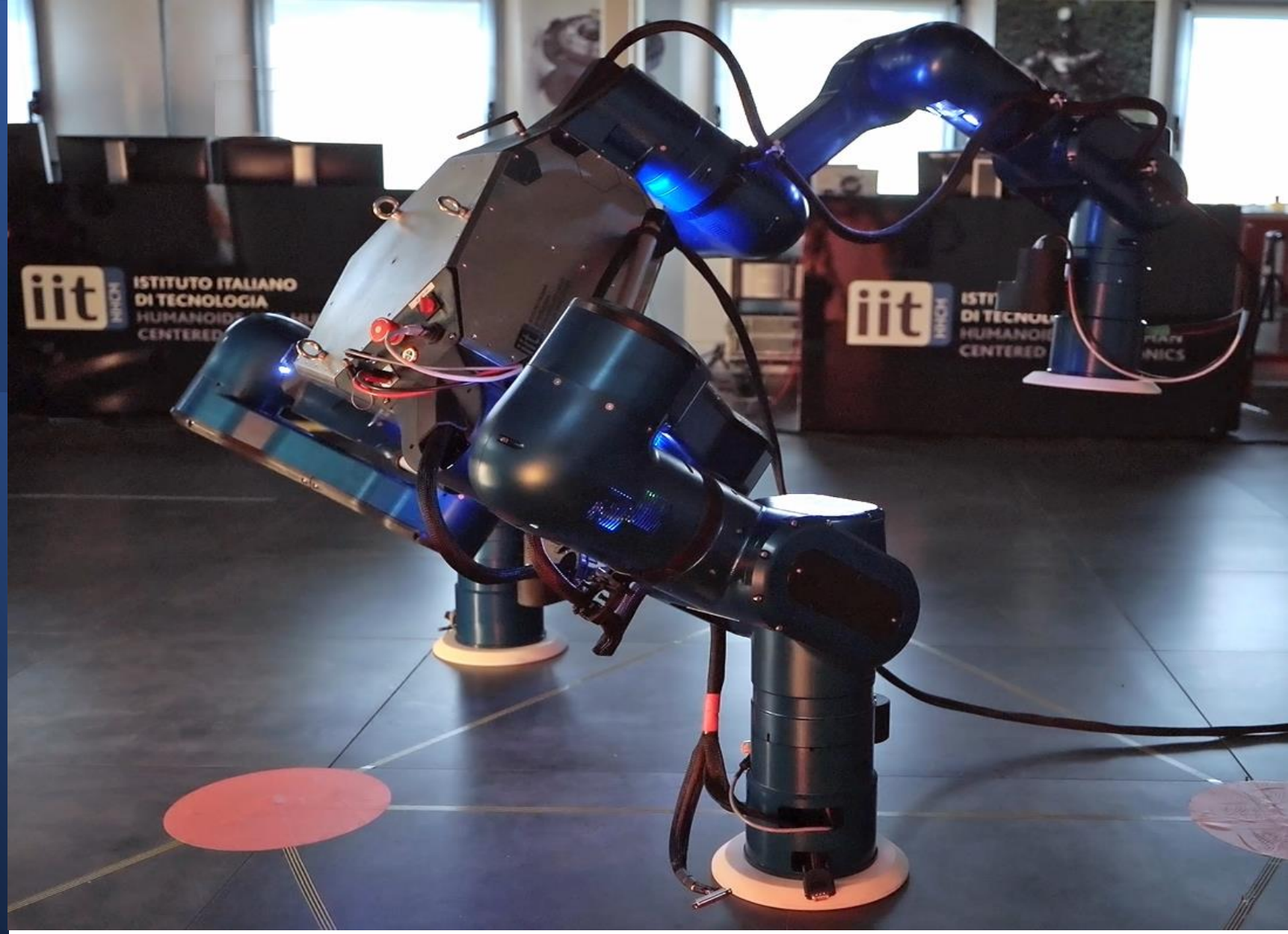
D. Antonucci , A. Margan, A. Laurenzi, A. Rodriguez, P. Romeo, J. Barrientos, J. Estremera, A. Rusconi,G. Sangiovanni, N.G. Tsagarakis , and S. Cordasco

A REAL-TIME COMPUTER ARCHITECTURE BASED ON A CLIENT-SERVER APPROACH FOR A MULTI-ARM ROBOT MANIPULATION (MARM) PLATFORM



ISTITUTO ITALIANO DI TECNOLOGIA

INNOVATING SOLUTIONS

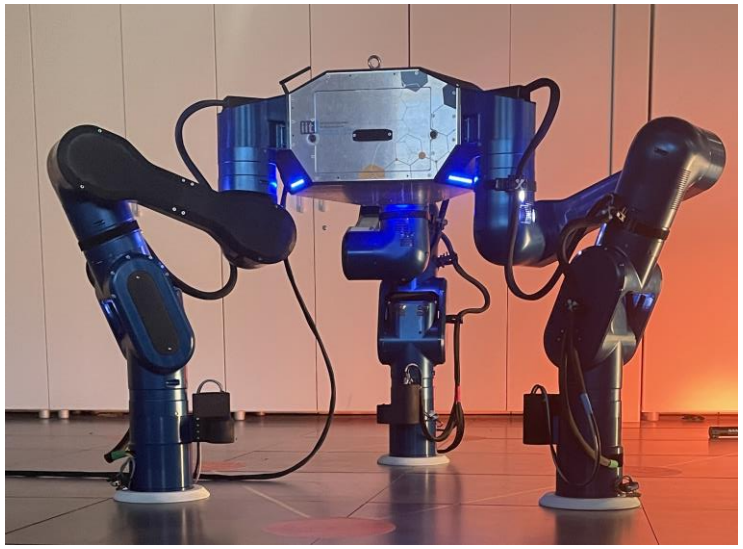
LEONARDO

esa

Overview

- Mirror project
- Software overview and control architecture
- Architecture design
- Safety
- Use cases and validation
- Discussion and conclusion

MIRROR project

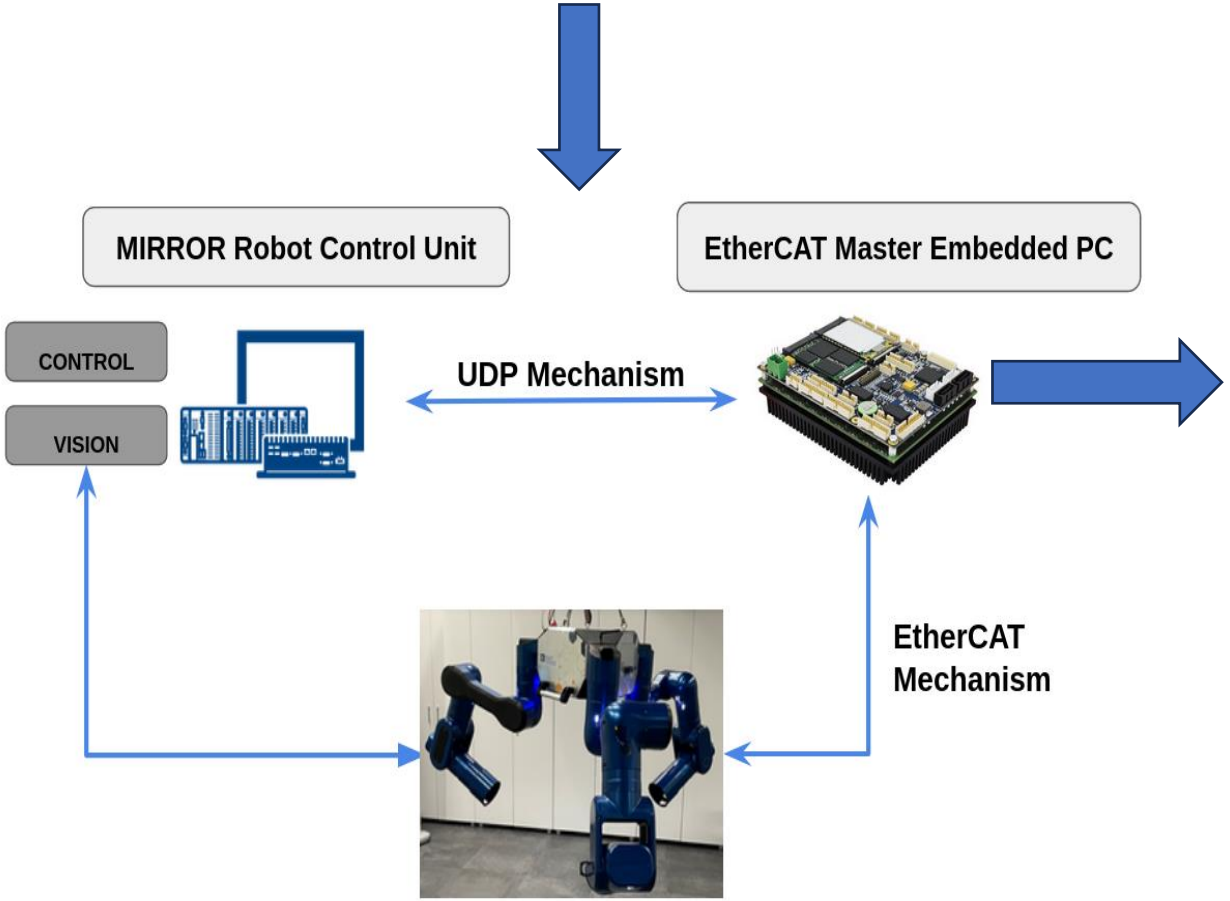


MIRROR: Multi-Arm Installation Robot for Reading ORUS and Reflectors:

- 3 x robotic arm/leg loco manipulation platform
  - 6DOF , 1.2m long
  - 14kg payload capacity on each arm
  - High performance torque sensing and control actuation
  - Semi-modular design
- Capable of performing loco-manipulation to relocate itself while carrying a payload and transporting it from one position to another
- The robot can use the arms to grapple standard interconnects (SIs), installed on re-configurable tiles over the station's surfaces for locomotion purposes, tile assembly, and handling of Orbital Replacement Units (ORUs).

Software overview and Control architecture

MIRROR project: Multi-Arm Installation Robot for Reading ORUS and Reflectors.



Physical separation in term of embedded systems.

Client-Server approach.

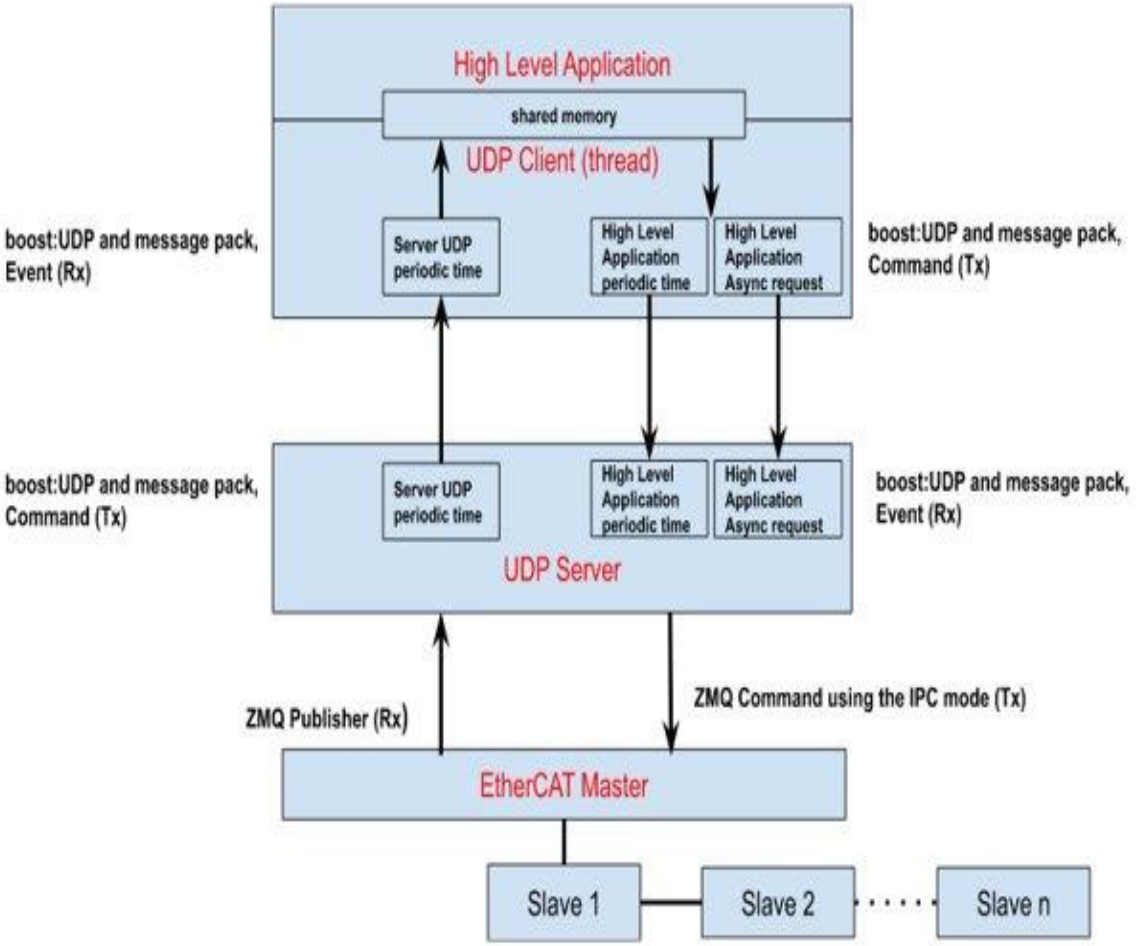
APIs provided:

- client/server protocol initialization (connect, disconnect, quit server, etc..);
- setting of control modes;
- start/stop actuation;
- release/engage brakes;
- read actuation states, force-torque sensor, IMU, power board date and the states of other devices providing full telemetry of the robot state
- write actuation set point references;
- read/write service data object (calibration);

Motor controllers available:

- Position mode: Position controller + Current controller;
- Impedance mode: Impedance controller + Torque controller + Current controller;

Architecture design



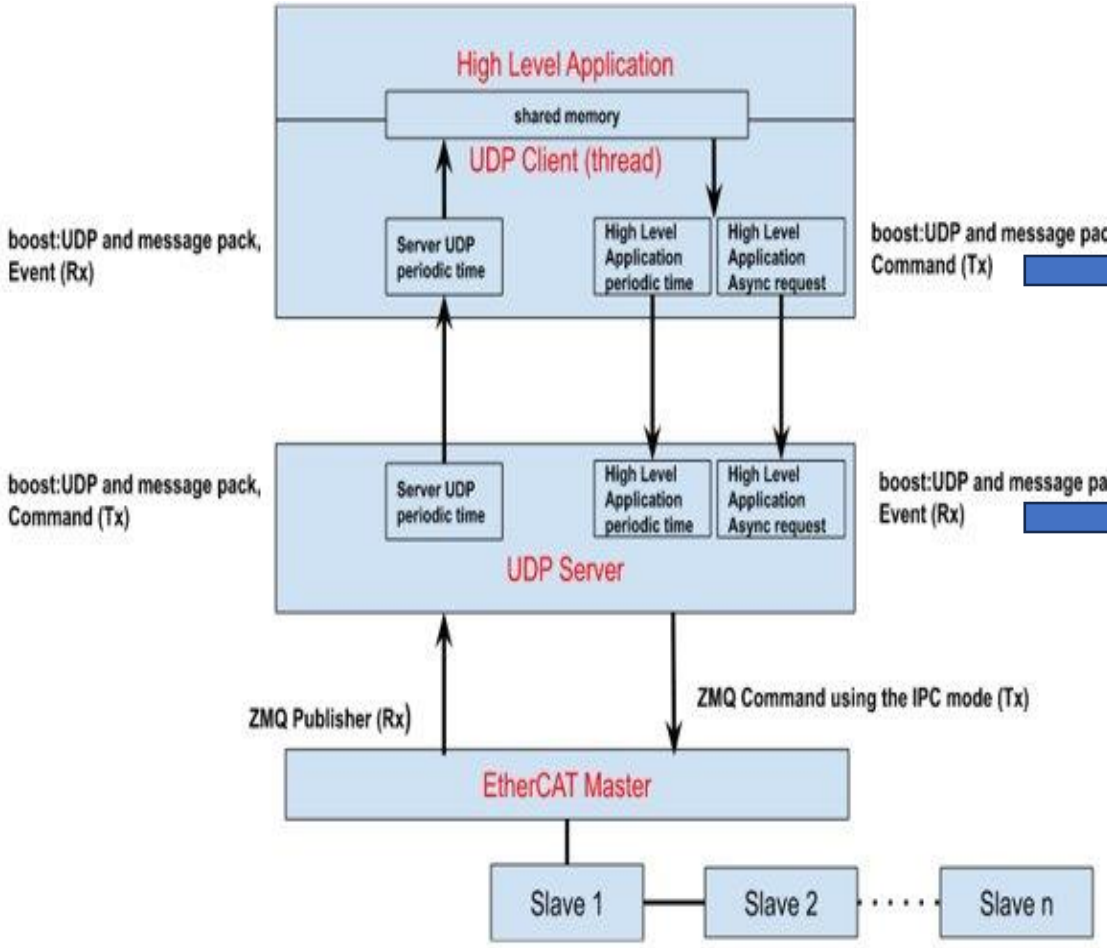
Msgpack library for packing and unpacking data.

UDP protocol to exchange data, in particular boost::asio library

Command handler.

Event handler.

Architecture design

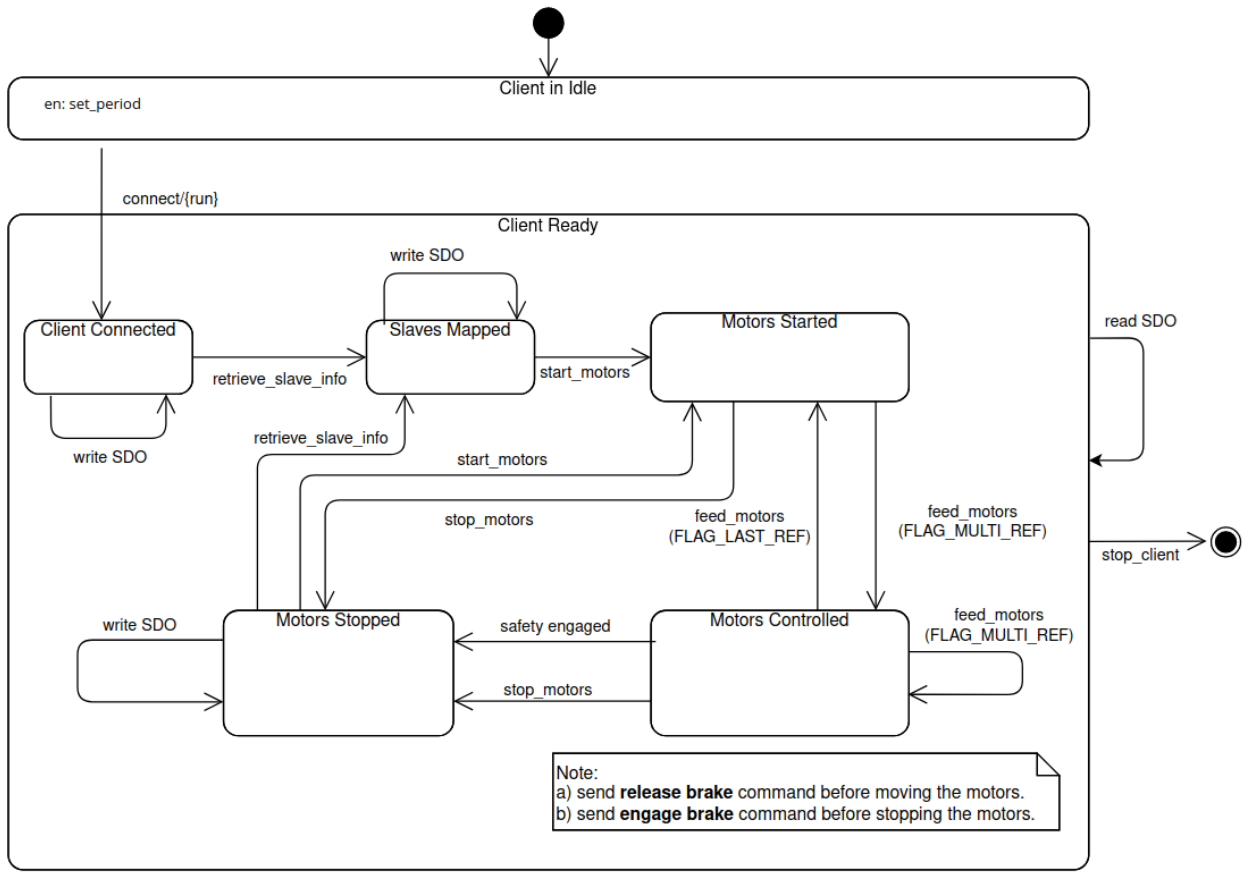


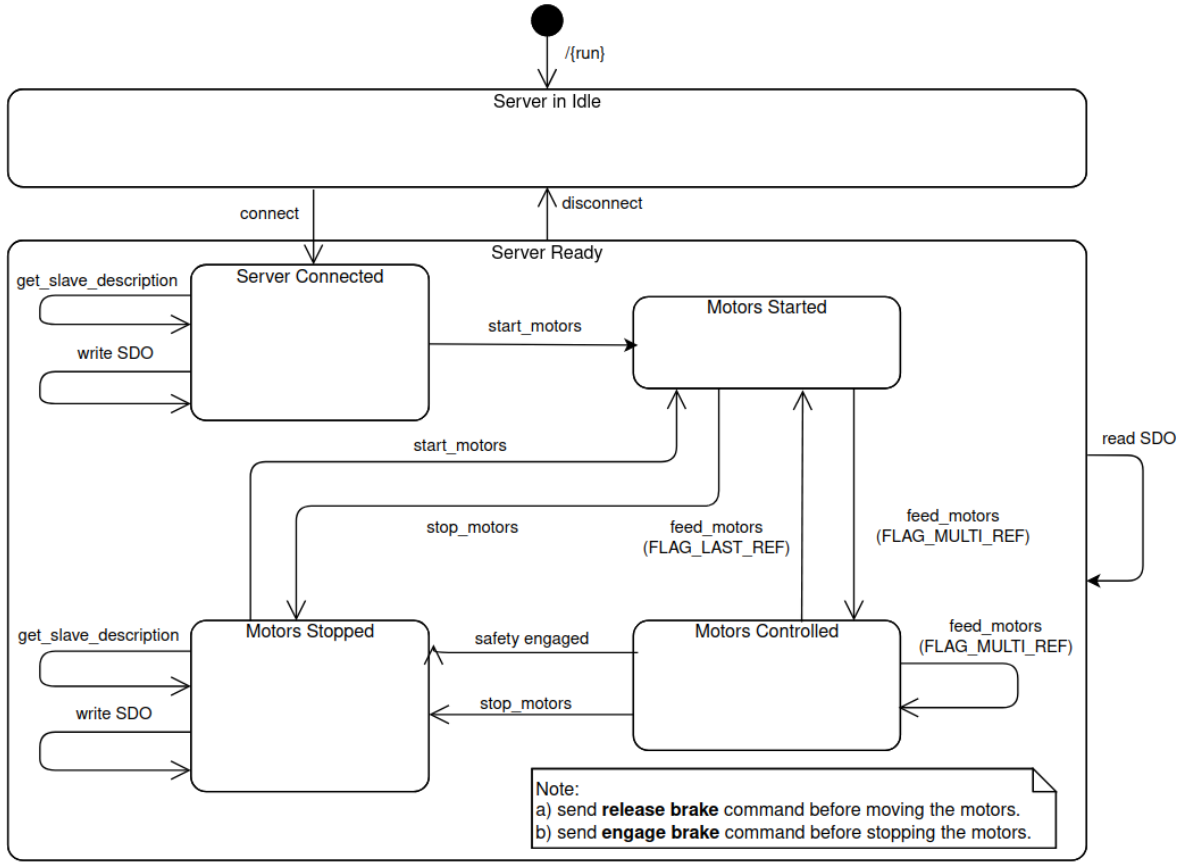

```
bool Client::start_motors(const MST &motors_start)
{
    CBuffT<4096u> sendBuffer{};
    bool ret_cmd_status=false;
    // packing
    auto sizet= proto.packReplRequestMotorsStart(sendBuffer, motors_start);
   // send
   do_send(sendBuffer.data(), sendBuffer.size());
   // ACK/NACK information
   ret_cmd_status= get_reply_from_server( ReplReqRep::START_MOTOR,repl_msg);
   return ret_cmd_status;
}
```

```
// Register Message Handler
registerHandler(ServerMsg::MSG_MOTOR_STATUS,&Client::motor_status_handler);

void Client::motor_status_handler(char *buf,size_t size)
{
    _mutex_motor_status->lock();
   static MSS motors_status;
   // unpacking
   auto ret
=  proto.getEscStatus(buf, size, ServerMsg::MSG_MOTOR_STATUS,motors_status);
   // manipulation
    .......
   _mutex_motor_status->unlock();
}
```

Architecture design

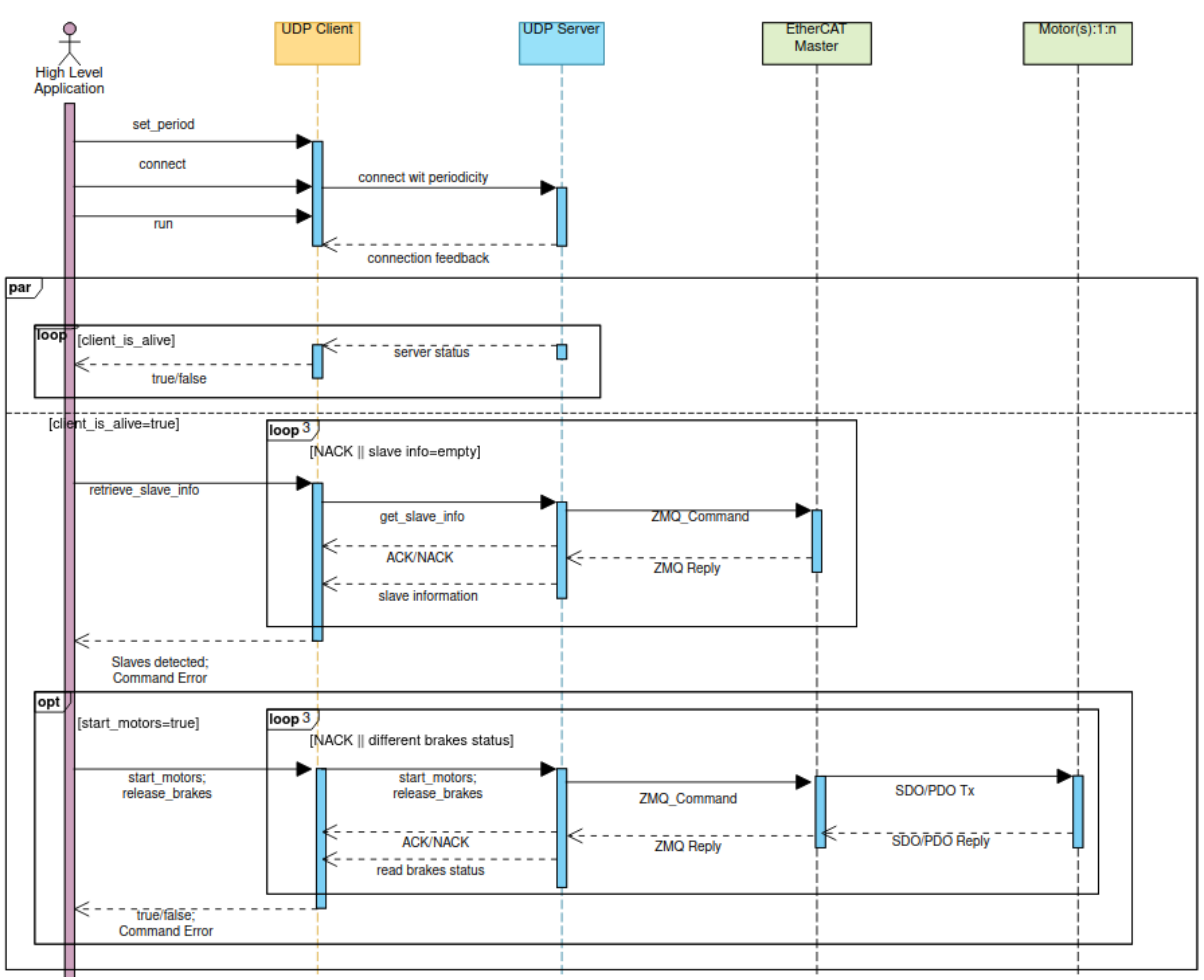
Two state machines were implemented to make the mechanism more consistent due to the asynchronous protocol, helping high level robotic controller to send the commands in the right way, getting right feedback.

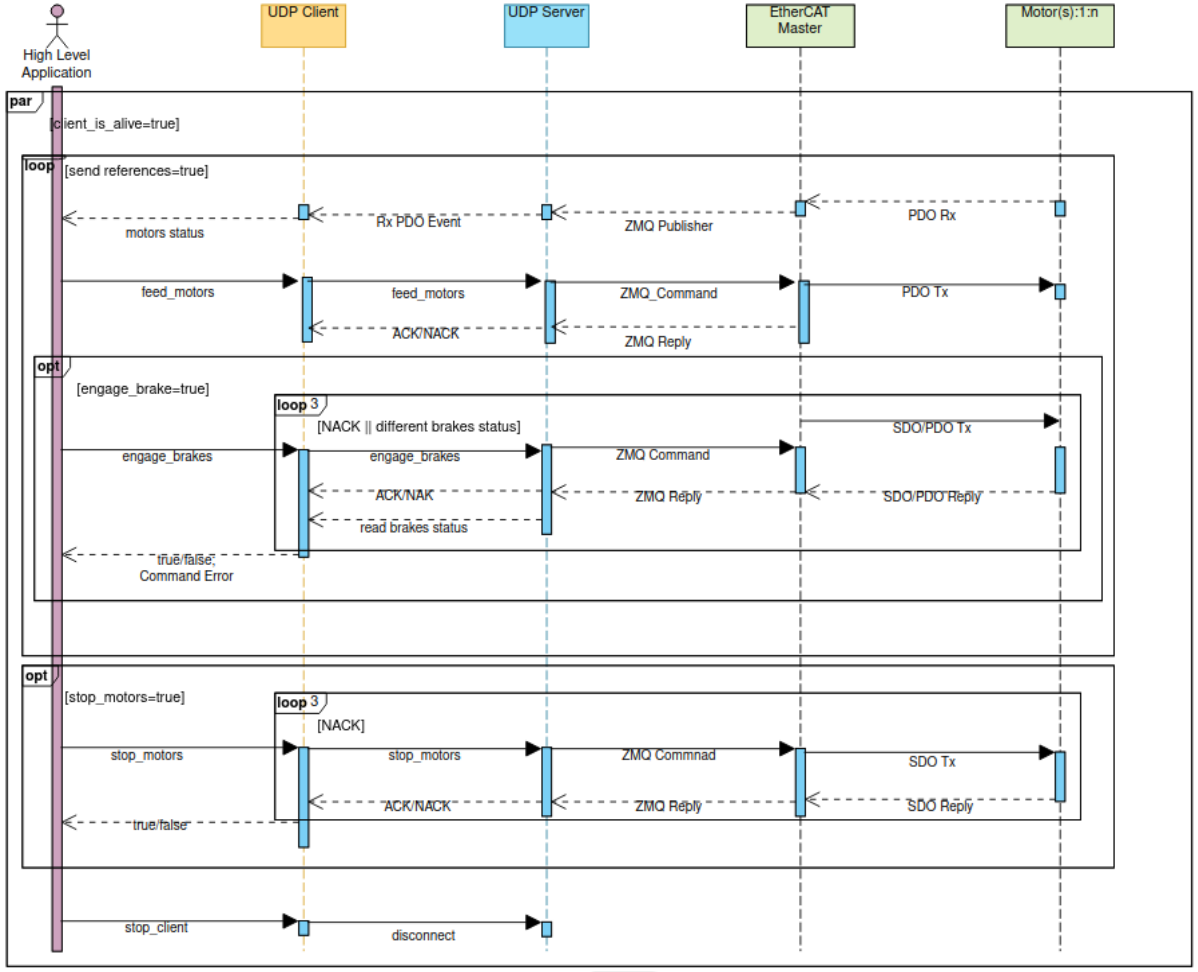




Architecture design

This sequence communication diagram of the overall system shows a typical use case to operate the robot:





Safety

Two safety controls were also developed to verify:

Communication brakedown.

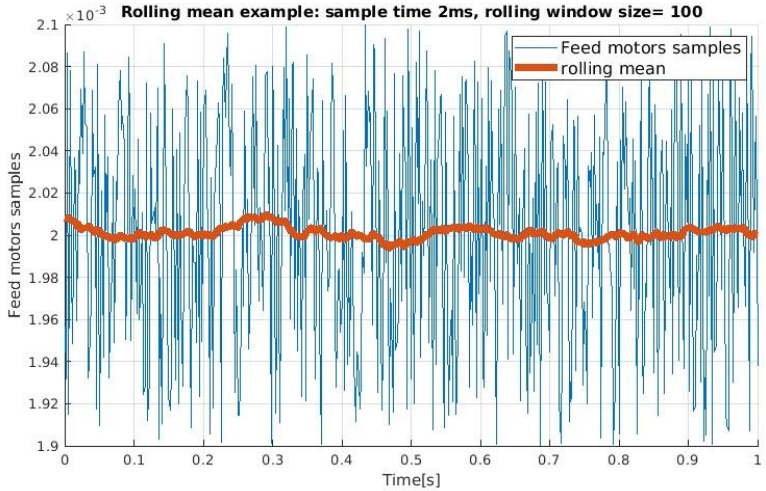
Communication **degradation.**

A recovery **action is activated to safely terminate the operation of the system and bring the robot actuation in idle mode engaging their brakes.**

Safety

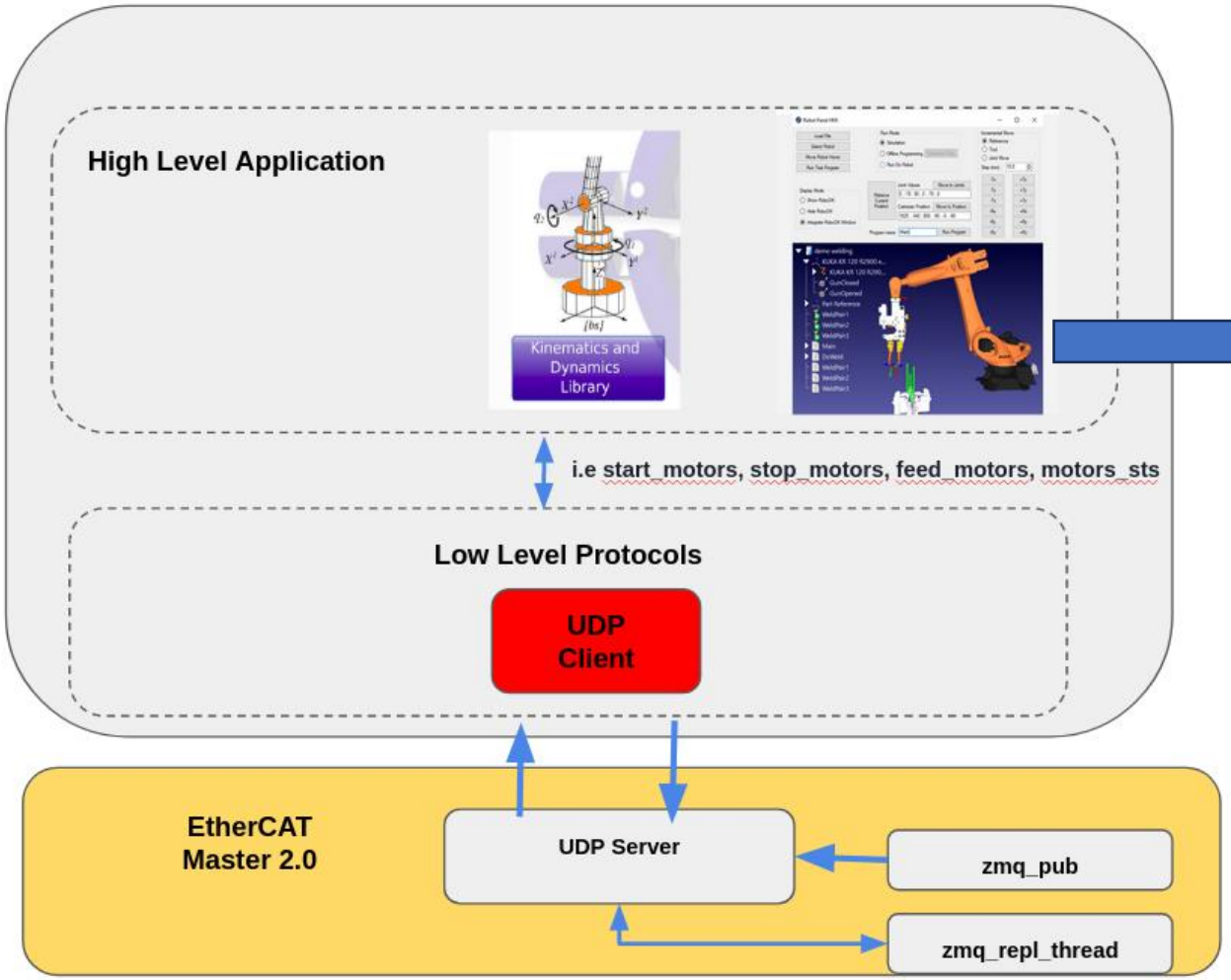
Communication degradation

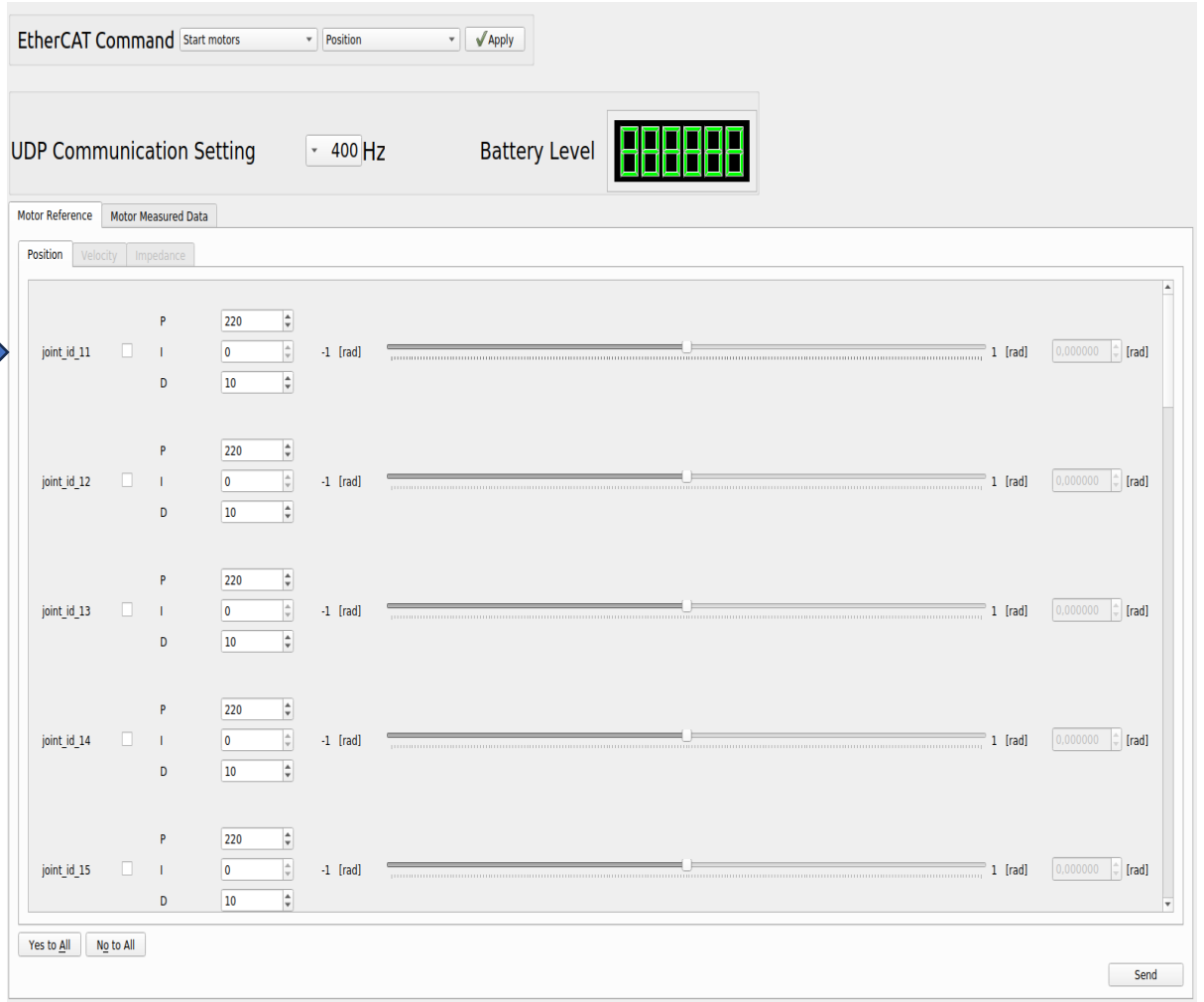
- The rolling_mean functions in the boost library are used to evaluate the communication quality in a specific window. The mean in that window is controlled verifying the desired communication frequency for the data exchange along the communication pipeline.
- Safety frequencies range allowed: greater a minimum frequency level of 225 Hz or less than or equal to 500 Hz.
- The rolling window size can be set before running the server process, the default size of which is equal to 100.




```
2ms --> rolling window 100

************************LOG****
------------------------------------------------------------------
10:30:43.467] [server] [info]  REPL REQ : SET_MOTOR_REFS
[10:30:43.467] [server] [info]  REPL REQ : SET_MOTOR_REFS
[10:30:43.467] [server] [info] mean refs freq mean 0.002000171470000014
[0MQ Req] connect to ipc:///tmp/ecat_master:5555
[10:30:43.468] [server] [info]  send request reply SET_MOTOR_REFS
[10:30:43.469] [server] [info]  REPL REQ : SET_MOTOR_REFS
[10:30:43.469] [server] [info]  REPL REQ : SET_MOTOR_REFS
[10:30:43.469] [server] [info] mean refs freq mean 0.002001261530000001
[0MQ Req] connect to ipc:///tmp/ecat_master:5555
[10:30:43.470] [server] [info]  send request reply SET_MOTOR_REFS
[10:30:43.471] [server] [info]  REPL REQ : SET_MOTOR_REFS
[10:30:43.471] [server] [info]  REPL REQ : SET_MOTOR_REFS
[10:30:43.471] [server] [info] mean refs freq mean 0.002000142350000001
[0MQ Req] connect to ipc:///tmp/ecat_master:5555
[10:30:43.472] [server] [info]  send request reply SET_MOTOR_REFS
[10:30:43.473] [server] [info]  REPL REQ : SET_MOTOR_REFS
[10:30:43.473] [server] [info]  REPL REQ : SET_MOTOR_REFS
[10:30:43.473] [server] [info] mean refs freq mean 0.001999978140000001
[0MQ Req] connect to ipc:///tmp/ecat_master:5555
[10:30:43.474] [server] [info]  send request reply SET_MOTOR_REFS
[10:30:43.475] [server] [info] periodicActivity tDiff 0.001836719
[10:30:43.475] [server] [info]  REPL REQ : SET_MOTOR_REFS
[10:30:43.475] [server] [info]  REPL REQ : SET_MOTOR_REFS
[10:30:43.475] [server] [info] mean refs freq mean 0.002000213480000001
[0MQ Req] connect to ipc:///tmp/ecat_master:5555
[10:30:43.476] [server] [info]  send request reply SET_MOTOR_REFS
[10:30:43.477] [server] [info]  REPL REQ : SET_MOTOR_REFS
[10:30:43.477] [server] [info]  REPL REQ : SET_MOTOR_REFS
[10:30:43.477] [server] [info] mean refs freq mean 0.002000003240000001
[0MQ Req] connect to ipc:///tmp/ecat_master:5555
[10:30:43.478] [server] [info]  send request reply SET_MOTOR_REFS
[10:30:43.479] [server] [info]  REPL REQ : SET_MOTOR_REFS
[10:30:43.479] [server] [info]  REPL REQ : SET_MOTOR_REFS
[10:30:43.479] [server] [info] mean refs freq mean 0.002000028500000001
[0MQ Req] connect to ipc:///tmp/ecat_master:5555
[10:30:43.480] [server] [info]  send request reply SET_MOTOR_REFS
[10:30:43.481] [server] [info]  REPL REQ : SET_MOTOR_REFS
[10:30:43.481] [server] [info]  REPL REQ : SET_MOTOR_REFS
[10:30:43.481] [server] [info] mean refs freq mean 0.002000024840000001
[0MQ Req] connect to ipc:///tmp/ecat_master:5555
[10:30:43.482] [server] [info]  send request reply SET_MOTOR_REFS
[10:30:43.483] [server] [info]  REPL REQ : SET_MOTOR_REFS
[10:30:43.483] [server] [info]  REPL REQ : SET_MOTOR_REFS
[10:30:43.483] [server] [info] mean refs freq mean 0.002000026470000001
[0MQ Req] connect to ipc:///tmp/ecat_master:5555
[10:30:43.484] [server] [info]  send request reply SET_MOTOR_REFS
```

Use cases and validation

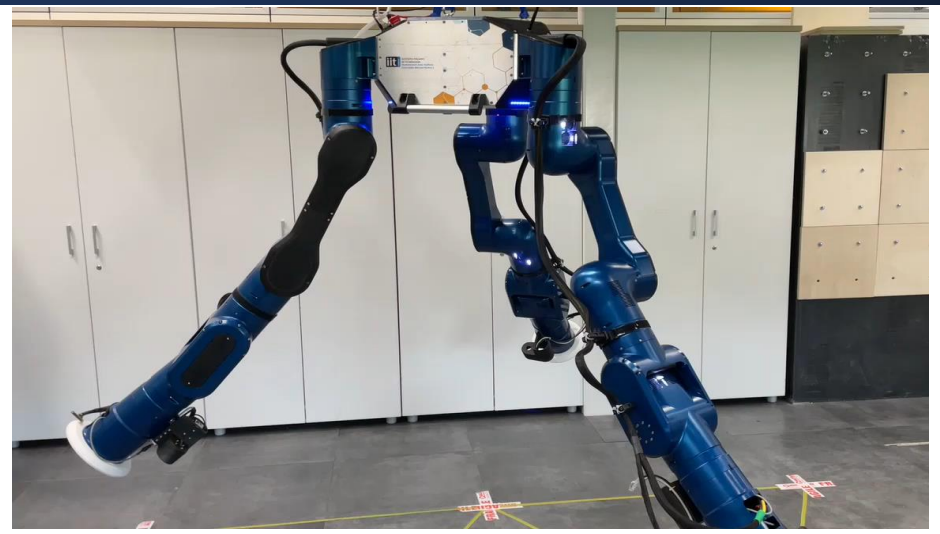




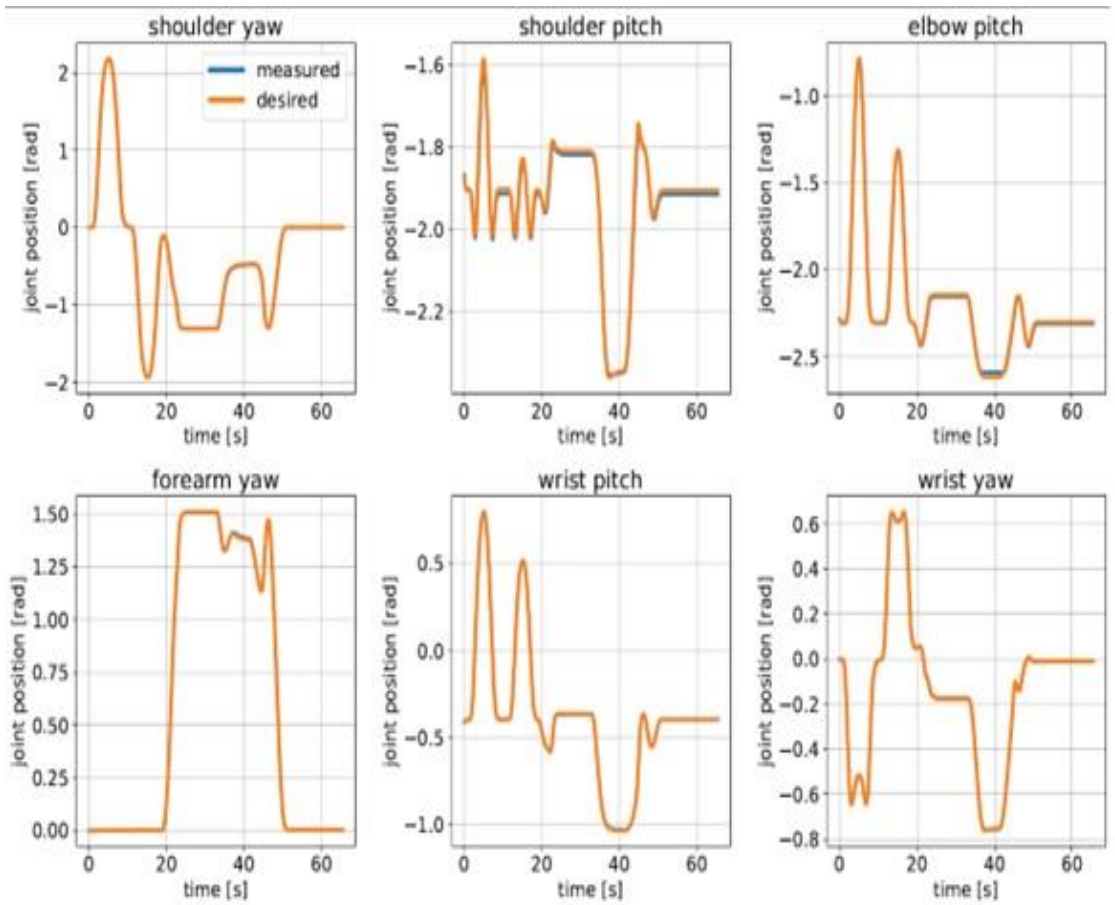
Use cases and validation

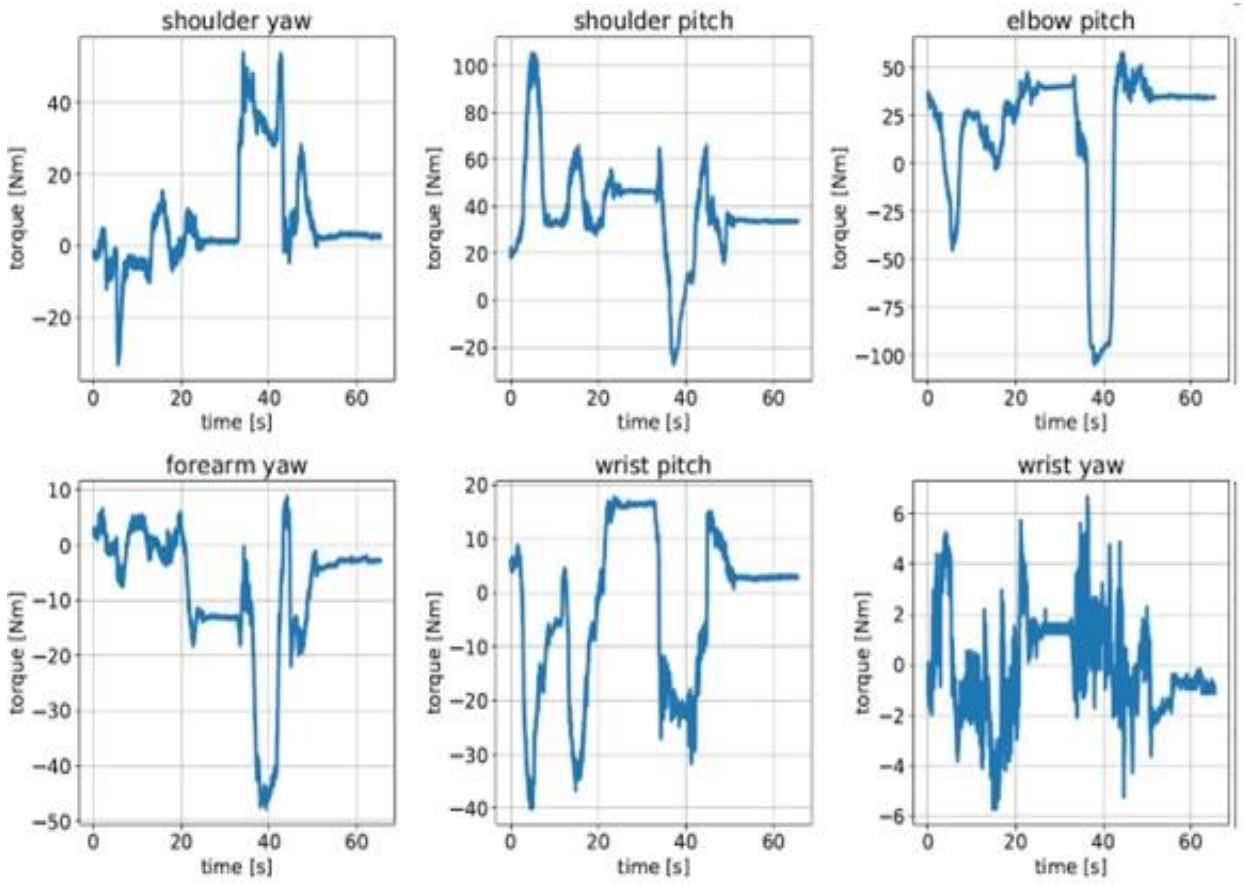
```
network:
hostname: localhost
port: 54321
timeout: 1000

###############position###############
#control_mode: position
#position gains
#gains: [200.0,0.0,10.0,0.0,0.0]
###############position###############

###############impedance###############
control_mode: impedance
#impedance gains
# stiffness, damping, tau_p, tau_f, tau_d
gains: [1000.0,10.0,1.0,0.7,0.007]
###############impedance###############


UDP_period_ms: 4
homing_position: {11: -0.75, 12: -1.0, 13: -1.0, 14: -0.75, 15: 1.0, 16: -0.75,
                  21: -0.75, 22: -1.0, 23: -1.0, 24: -0.75, 25: 1.0, 26: -0.75,
                  31: -0.75, 32: -1.0, 33: -1.0, 34: -0.75, 35: 1.0, 36: -0.75}
homing_time_sec: 3
trajectory: {11: 0.0, 12: -1.9, 13: -2.3, 14: 0.0, 15: -0.4, 16: 0.0,
             21: 0.0, 22: -1.9, 23: -2.3, 24: 0.0, 25: -0.4, 26: 0.0,
             31: 0.0, 32: -1.9, 33: -2.3, 34: 0.0, 35: -0.4, 36: 0.0}
trajectory_time_sec: 3
repeat_trj: 3


slave_id_led: [16,26,35]

XBotInterface:
urdf_path: ${PWD}/urdf/mirror.urdf
srdf_path: ${PWD}/srdf/mirror.srdf
joint_map_path: ${PWD}/joint_map/mirror_joint_map.yaml

ModelInterface:
model_type: "RBDL"
is_model_floating_base: "true"
```

https://advrhumanoids.github.io/XBotInterface/

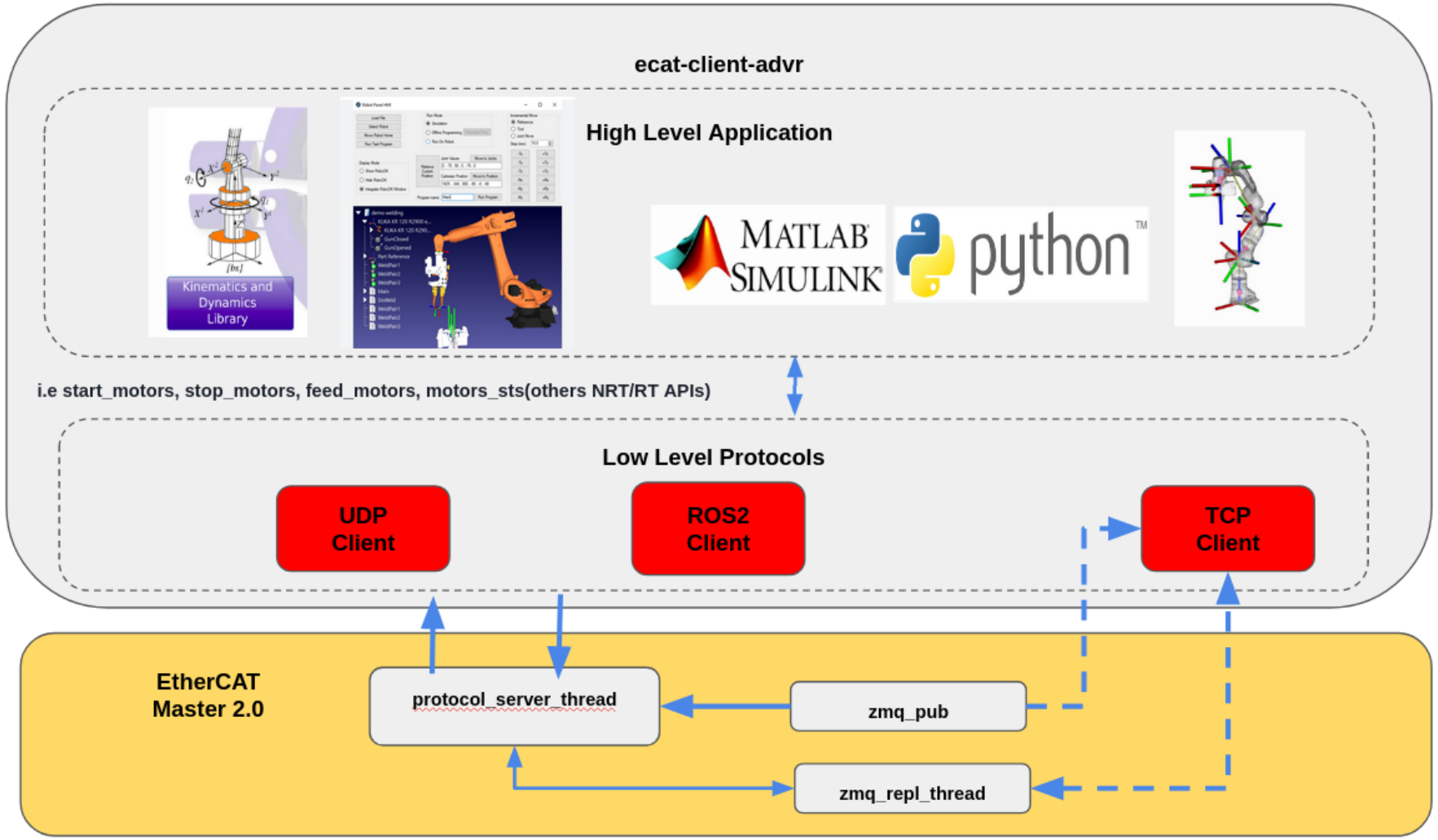


Use cases and validation





Discussion and conclusion



..some extra demonstrations

Ackowledgement

Davide Antonucci

Stefano Cordasco

Arturo Laurenzi

Alessio Margan

Nikolaos Tsagarakis

Pablo Romeo

Andres Rodriguez

Jorge Barrientos

Joaquin Estremera

Andrea Rusconi

Guido Sangiovanni

Thank you

Davide Antonucci

davide.antonucci@iit.it