Cross-Robot Core Software Architecture for Real-Time Humanoid Control

Ava Lakmazaheri
Olin College | Mechanical Engineering

Alex Moran
Virginia Tech | Computer Engineering

Dr. Alexander Leonessa
Virginia Tech | Mechanical Engineering

BACKGROUND

Humanoid robots' potential for dexterity and intelligence uniquely enables them to perform life-critical tasks such as assisting people with mobility impairment or performing search-and-rescue in hazardous environments.

The DARPA Robotics Challenge (DRC) sought to promote innovation in this area by inviting teams to develop a humanoid robot that can perform disaster-response operations (drive a utility vehicle, traverse rubble, enter a building, close a valve to a leaking pipe, etc.) semi-autonomously and in real-time.

[1]

In this work, we sought to identify and adapt a software architecture that supports such advanced locomotion and manipulation tasks for the TREC Lab's own humanoid robot.

CONTROL ARCHITECTURE

Our first goal was to conceptualize a robust operating logic for a generic humanoid. After studying various strategies from the DRC, we synthesized the following system flow that maps abstract tasks to low-level control.

Operators specify high-level tasks *e.g. walk to target* using a graphical interface

A simulated environment allows selecting target objects and locations

Custom widgets break down complex tasks step-by-step

An action string outputted from the GUI is used to traverse a finite state machine (FSM)

Multiple FSMs can be used to capture entire DRC tasks or background behaviors

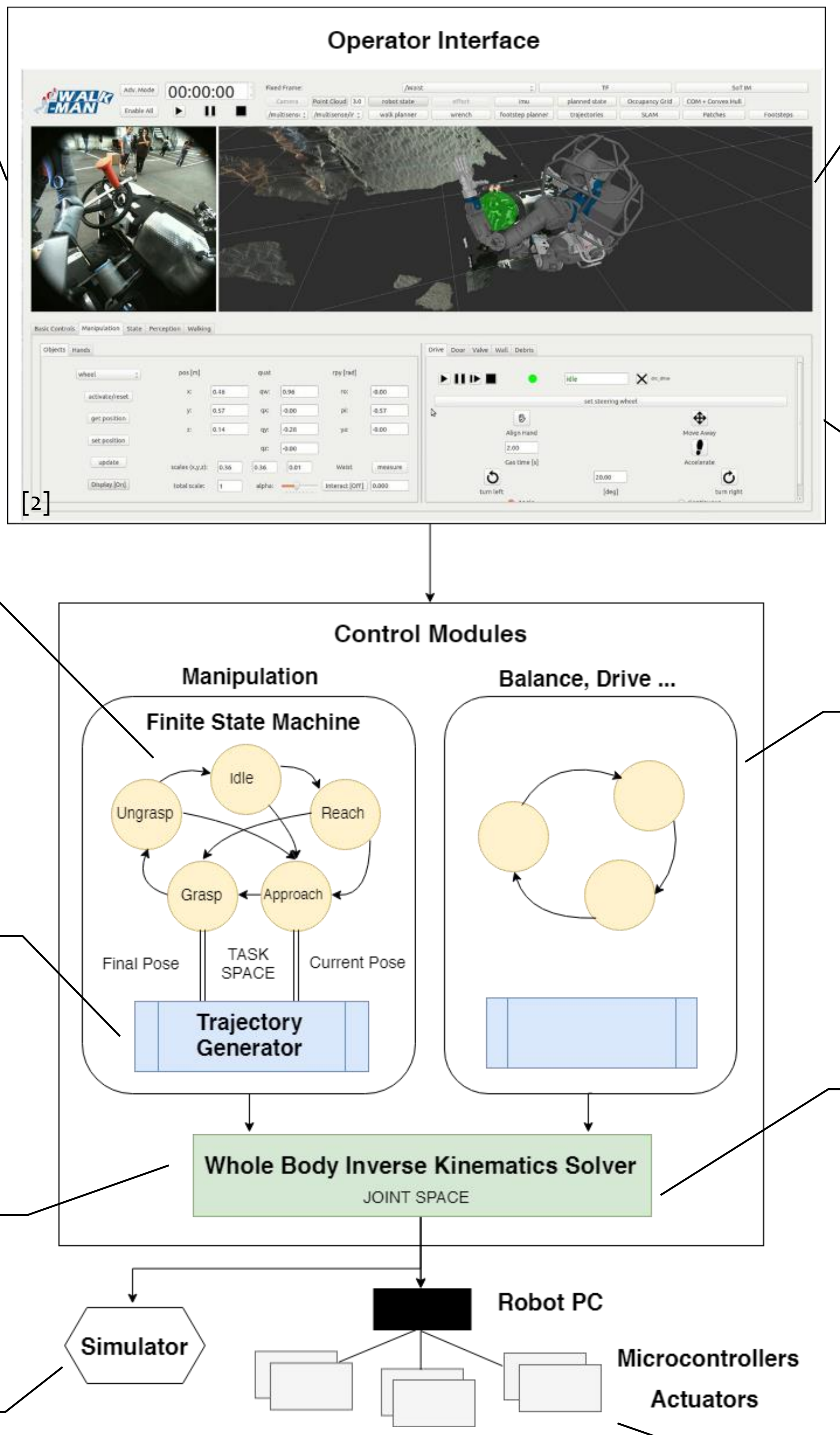
Trajectories interpolate a smooth path between target states

FSMs may generate conflicting target positions for one joint; WBIK solves a hierarchy of tasks with priorities and constraints

Inverse kinematics maps the end-effector state to target joint positions

A third-party simulator visualizes robot behavior with built-in joint controllers

Joint states are sent to a main computer then distributed to multiple microcontrollers via EtherCAT



SOFTWARE REVIEW

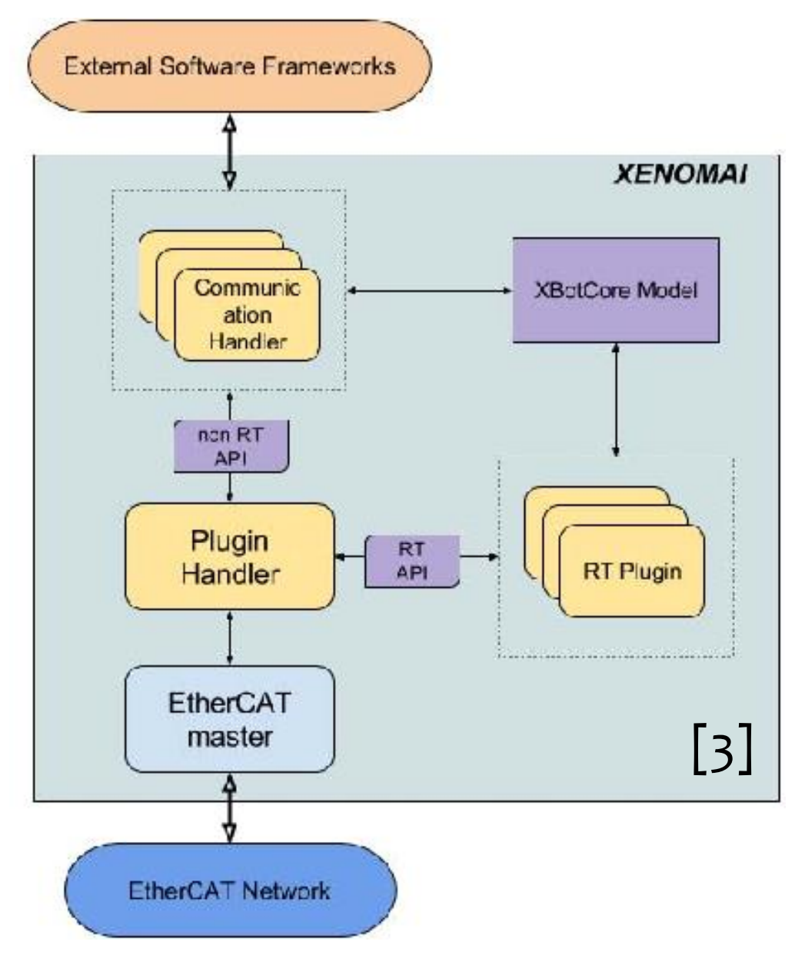
In order to realize our software architecture, we began by evaluating existing open-source control applications for functionality, modularity, and simplicity. Our primary findings were: 1) *XBotCore* - a software platform that supports real-time humanoid control and 2) *IHMC Open Robotics Software* - a set of libraries for out-of-the-box planning and robot operation. Both allow modelling abstract behaviors using finite state machines, multi-tasking via a whole-body inverse kinematics engine, and simplifying complex structures into a universal robot description format.

XBOTCORE

XBotCore extends the Robot Operating System (ROS) middleware to enable communication between user-developed control modules, third-party software, and robot hardware.

Its plugin architecture ensures a 1 kHz control loop with the robot even during intensive background computations.

We installed *XBotCore* and created non-real time communication plugins to test the software. These modules enable robot control via either direct joint trajectories or an inverse kinematics solver. A ROS node then visualizes whole-body robot behavior in the third-party simulator *Gazebo*.



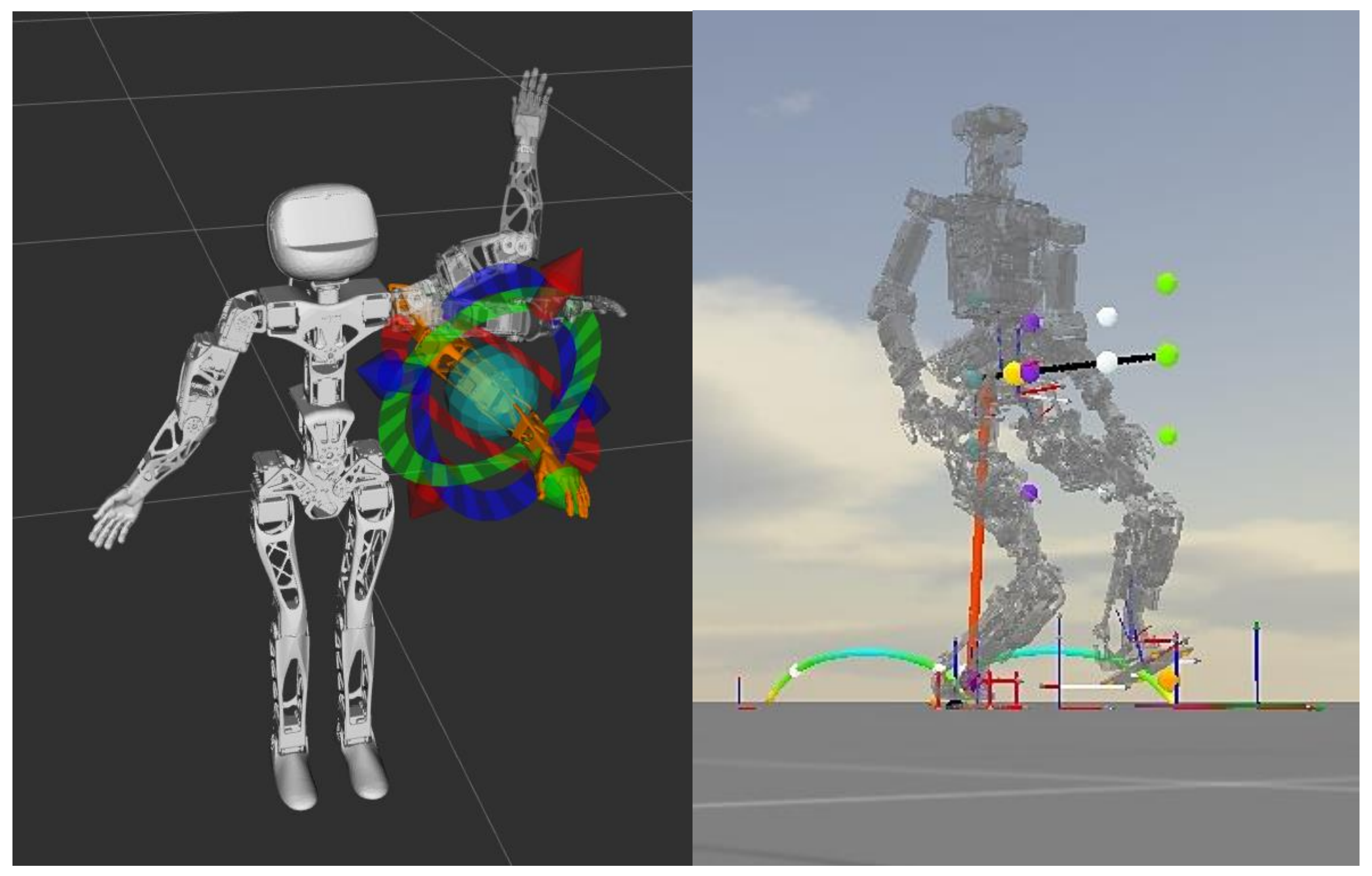
[3]

IHMC OPEN ROBOTICS SOFTWARE

As an open-source initiative, the Institute for Human and Machine Cognition (IHMC) has published the complete control code from their DRC entry. This robotics software contains legged locomotion algorithms for walking and an optimized momentum-based controller core for managing multiple tasks.

To test the software, the TREC Lab adapted an IHMC walking planner for simulated navigation on its Tactical Hazardous Operations Robot (THOR).

RESULTS & CONCLUSIONS



We conclude this phase with two software platforms ready and operational. To decide which to use moving forward, we consider the following:

1) The IHMC platform provides robust control algorithms, but is inflexible. Generating new high-level operations or adapting their control algorithms will be a primary challenge.

2) XBotCore is not self-contained and requires a significant amount of algorithm development and coding for the intended purpose. However, it will ultimately provide more customizability.

3) Without an operational robot to test the solutions, it is difficult to draw solid conclusions about the real-time efficacy of either software.

Our future plans include developing a full simulation of THOR in the DRC environment, testing and further developing real-time commands on a microcontroller, and ultimately controlling a full physical robot to perform high-level tasks.

REFERENCES

[1] https://www.darpa.mil/program/darpa-robotics-challenge
[2] Ferrati M, Settimi A, Muratore L, Tsagarakis NG, Natale L and Pallottino L (2016) The Walk-Man Robot Software Architecture.
[3] Muratore L, Laurenzi A, Mingo Hoffman E, Rocchi A, Caldwell D & Tsagarakis N. (2017). On the Design and Evaluation of XBotCore, a Cross-Robot Real-Time Software Framework