XBotCore: A Real-Time Cross-Robot Software Platform

Luca Muratore[1] [2]

Abstract

In this work, **XBotCore** (*Cross-Bot-Core*), a light-weight, Real-Time (RT) software platform for EtherCAT-based robots will be presented.

XBotCore is open-source and is designed to be both an RT robot control framework and a software middleware. It satisfies hard Real-Time requirements, performing computation within predictable timing constraints e.g. ensuring 1 KHz control loop even in complex Multi-Degree-Of-Freedom systems. Moreover, it provides a simple and easy-to-use middleware Application Programming Interface (API), for both RT and non-RT control frameworks. The *XBotCore* API is completely flexible with respect to the external control framework (RT or non RT) a user wants to utilize.

One of the main *XBotCore* design goals is the cross-robot compatibility: it has to work with any kind of EtherCAT-based robot, without any code modification; it is crucial to be able to reuse the software platform with different robots, or different part of the same robot.

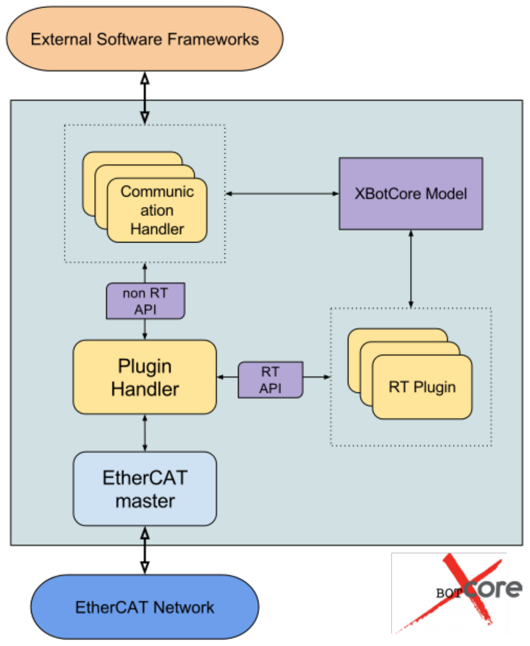

Figure 1. *XBotCore*: a Cross-Robot software platform based on Xenomai RT Linux extension and EtherCAT fieldbus

As shown in Figure 1, *XBotCore* consists of 5 main components: *EtherCAT master*, *Plugin Handler*, *XBotCore-Model*, RT and non RT *middleware API* and *Communication Handlers*.

Real-time scheduling is essential for precise robot control period, especially for high-frequency (e.g. 1 KHz): there are several operating systems or platforms which support real-time operation, like Windows CE, INtime, RTLinux, RTAI, Xenomai, QNS, VXWorks [1]. We selected a Linux based RTOS because we want to avoid a licensed product that does not give us the possibility to modify the source code depending on our system. Xenomai is our choice because its design considers extensibility, portability and maintainability as well as low latency [2], [10].

XBotCore is developed for EtherCAT based robots: we expect a network of EtherCAT slaves in the system, e.g. the electronic boards responsible for motor control and sensors data acquisition. The EtherCAT protocol has a master/slave medium access control policy. *EtherCAT master* is a RT thread and its implementation is developed starting from the *SOEM* (Simple Open EtherCAT Master) library: an open source EtherCAT master implementation, meant to be highly portable on a variety of embedded platforms (HW and RTOSes) [4]. The structure of the data flowing in the EtherCAT network is called *PDO* (Process Data Object) and it has two different sub-structures:

- *PDO RX*: master input, slave output e.g. link position, motor position, motor velocity, torque, temperature etc.
- *PDO TX*: master output, slave input e.g. position reference, torque reference, gains etc.

Furthermore the *EtherCAT master* provides an asynchronous API to the higher level components in order to read/write the PDO data.

The *Plugin Handler* is the main component of the RT plugin architecture: it is an RT thread that executes sequentially a set of plugins, as in [9]. A *Plugin* is a class inheriting from the abstract class *XBotPlugin*. Writing a *Plugin* is straightforward for the user, as he needs to implement three functions:

- an `init()` function that will be called only once by the *Plugin Handler* in order to initialize the variables of the *Plugin*
- a `run()` function which will be executed in the control loop of the *Plugin Handler*
- a `close()` function, called when the *Plugin Handler* wants to remove the plugin

The *Plugin* implementation is compiled as a shared object (.so). It is possible to dynamically load and unload one or more plugins in the *Plugin Handler* that is responsible to start all the loaded plugins, executes them sequentially in the `plugin_handler_loop()` function, and closes them

[1]Advanced Robotics Department (ADVR), Istituto Italiano di Tecnologia, Genova, Italy.
[2]School of Electrical and Electronic Engineering, The University of Manchester, M13 9PL, UK.

before unloading them.

XBotCore implies a novel approach to the configuration of low-level control systems by using modern description formats such URDF[1] (Universal Robotics Description Format) and SRDF[2] (Semantic Robotic Description Format), traditionally used for high-level software components (e.g. ROS nodes). Its main feature is to be a cross-robot software platform: thanks to the abstractions provided by the `XBotCoreModel` class it is possible to control different robots or different parts of the same robot without code modifications. In fact the API provided to control the robot is dynamically built starting from the URDF and SRDF of the robot. Modifying the SRDF, removing for example a kinematic chain (e.g. the torso of the robot), results in a different API for the user that is compatible with the available/desired parts of the robot to control. The same happens when the URDF is modified, e.g. when working with a different robot.

XBotCore is also a middleware that provides the user with both RT and non-RT flexible and easy-to-use APIs. The *RT API* is suitable for the RT plugins that will run in the *Plugin Handler*: it works using a shared memory communication mechanism with the low level RT EtherCAT thread. The interfaces implemented by the RT API are: `IXBotJoint`, `IXBotChain`, `IXBotRobot` and `IXBotFT`.

The *non-RT API* has similar interfaces, but the implementation of the functions uses XDDP (Cross Domain Datagram Protocol) Xenomai pipes [8] in order to have asynchronous communication between RT and non-RT threads. It is crucial to have a lock-free IPC (Inter-Process Communication) in a robotic system. RT control threads and non-RT communication threads may want to exchange data in a way that does not require the former to leave the RT domain; Xenomai provides XDDP pipes for this purpose.

A robotic system has to communicate with the external world using a set of non-RT threads: in *XBotCore* the `XBotCommunicationHandler` class is provided; instances of classes inheriting from `XBotCommunicationHandler` run in non-RT threads, from which developers have access to ready-to-use non-RT API functions. It is pretty straightforward to implement a new set of Communication Handlers: *XBotCore* provides built-in support for the YARP [3] communication framework. Built-in ROS [5] support is under development.

To validate and evaluate the performance of the *XBotCore* software platform, we performed a set of experiments on the WALK-MAN [6] robot, a full-size humanoid with 33 DOFs (Degree-Of-Freedoms) and 4 custom F/T sensors, developed at the Istituto Italiano di Tecnologia (IIT). The experiments were carried out in a DRC-inspired scenario targeting the removal of debris in front of a valve. In the evaluation different high-level software frameworks were successfully integrated on top of *XBotCore*: *ArmarX* [7], *MoveIT!* and *YARP*. Only one RT plugin, responsible for the communication with the robot and the high-level external frameworks, was loaded during the experiments. We analyzed *XBotCore* performance in terms of control period of the RT plugin and CPU usage: during the experiments, each millisecond, we recorded all the data flowing from the EtherCAT master to the EtherCAT slaves and vice versa, thanks to an *XBotCore* low-level logging tool. The control period measured during the experiments in the worst-case scenario, i.e. while the robot was performing a set of manipulation actions, was always below the 1000μs (i.e. 1 Khz control frequency) even if the RT system is communicating with the high-level software components through YARP `XBotCommunicationHandler` non-RT threads. Moreover we compared *XBotCore* CPU usage while the robot was idle (i.e. not moving, nor communicating with external software frameworks) and when the manipulation experiments were running: the CPU core usage overhead introduced by *XBotCore* when the robot was performing the manipulation task as described above, was only 1.2% (in average). The CPU usage of *XBotCore* is very low (always ranging from 11.7% to 14.2%).

XBotCore is released free and open source at `https://gitlab.robotology.eu/luca.muratore/xbotcore`

The research leading to these results has received funding from the European Union Seventh Framework Programme [FP7-ICT-2013-10] under grant agreements n.611832 WALKMAN.

References

[1] A. Barbalace, A. Luchetta, G. Manduchi, M. Moro, A. Soppelsa, and C. Taliercio. Performance comparison of VxWorks, linux, RTAI, and xenomai in a hard Real-Time application. *IEEE Trans. Nucl. Sci.*, 55(1):435–439, Feb. 2008.

[2] J. H. Brown. How fast is fast enough? choosing between xenomai and linux for real-time applications. *Twelfth Real-Time Linux Workshop*, 2012.

[3] G. Metta, P. Fitzpatrick, and L. Natale. Yarp: Yet another robot platform. *International Journal on Advanced Robotics Systems*, 2006.

[4] D. Orfanus, R. Indergaard, G. Prytz, and T. Wien. EtherCAT-based platform for distributed control in high-performance industrial applications. In *2013 IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA)*, pages 1–8. ieeexplore.ieee.org, 2013.

[5] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[6] N. G. Tsagarakis, D. G. Caldwell, A. Bicchi, F. Negrello, M. Garabini, W. Choi, L. Baccelliere, V. Loc, J. Noorden, M. Catalano, M. Ferrati, L. Muratore, A. Margan, L. Natale, E. Mingo, H. Dallali, J. Malzahn, A. Settimi, A. Rocchi, V. Varricchio, L. Pallottino, C. Pavan, A. Ajoudani, J. Lee, P. Kryczka, and D. Kanoulas. WALK-MAN: A High Performance Humanoid Platform for Realistic Environments. *Journal of Field Robotics (JFR)*, 2016.

[7] N. Vahrenkamp, M. Wächter, M. Kröhnert, K. Welke, and T. Asfour. The robot software framework ArmarX. *it - Information Technology*, 57(2), 2015.

[8] Xenomai.org. Xenomai rtdm skin api. `http://xenomai.org/documentation/xenomai-2.6/pdf/rtdm-api.pdf`.

[9] K. Yokoi, F. Kanehiro, K. Kaneko, S. Kajita, K. Fujiwara, and H. Hirukawa. Experimental study of humanoid robot HRP-1S. *Int. J. Rob. Res.*, 23(4-5):351–362, 1 Apr. 2004.

[10] H. Yoon, J. Song, and J. Lee. Real-time performance analysis in linux-based robotic systems. *Proceedings of the 11th Linux Symposium*, 2009.

[1]http://wiki.ros.org/urdf
[2]http://wiki.ros.org/srdf