CANopen Robot Controller (CORC): An open software stack for human robot interaction development

Justin Fong*, Emek Barış Küçüktabak[$†], Vincent Crocher*, Ying Tan*, Kevin M. Lynch[$], Jose L. Pons[$†], Denny Oetomo*

Motivation

- Development of control strategies and algorithms is becoming increasingly important in wearable robotics
- CANopen is a well-established industrial protocol commonly used in many robotics platforms

This work introduces **CANopen Robot Controller (CORC)** – an open-source software stack designed to accelerate algorithm development.

CORC Overview

- Linux-based for portability and wide hardware platform support
- C (low level) and C++ (high level) implementation to leverage efficiency and object oriented programming
- Designed for application on any CANopen-based hardware
- Modular 3-layer design to maximise code portability:

Application Layer
- Overall program logic and control strategies
- State Machine structure
- Application-specific libraries (e.g. ROS)

(C++)

← Changes with program or algorithm

Robot Layer
- Describes robot structure abstracting I/O

(C++)

← Modification with new hardware only

CANopen Layer
- CANopen stack based on CANOpenSocket [1]

(C)

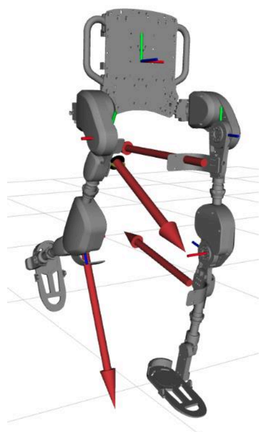
← Rarely requires change

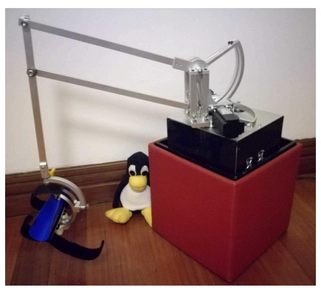
CORC is published under the permissive Apache-2.0 license.

Tested Hardware Platforms

X2 Exoskeleton (Fourier Intelligence) [2]

Input/Output: 4 Copley Accelnet ACK-055-06 motor drives, 4 custom force sensors
Computer: Laptop (Intel Core i7-9750H CPU, 16.0GB RAM with a PCAN-USB adapter)
OS: Ubuntu 18.04, ROS Melodic





EMU Upper-Limb Rehabilitation Robot [3]

Input/Output: 3 Kinco FD123-CA motor drives
Computer: Beaglebone AI (Dual ArmCortex-A15, 1.0GB RAM)
OS: Debian, Linux Kernel v4.14 with PREEMPT-RT patch

Demonstrative Software Applications

X2-P: Position control with inverse kinematics between sitting and standing postures; ROS Node broadcasting joint position and force to RViz visualisation
EMU-I: Impedance control with position-dependent gravity compensation
EMU-P: Position control minimum jerk task space trajectory

Implementation Notes

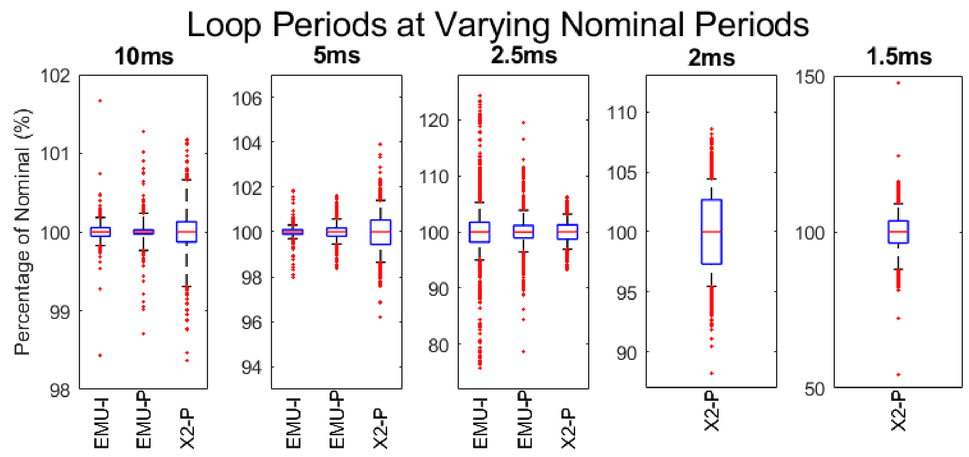
- Applications did not require any modification at the CANopen Layer, despite no common hardware
- EMU-I and EMU-P differed only at the Application Layer

Timing Evaluation

- Software applications were run with different nominal loop periods for at least 60 seconds each
- Actual loop periods were recorded, and are reported as percentage of nominal

Results

- Mean loop period within 0.001% of nominal in all cases
- EMU platform incapable of running at <2ms period



*Extents of box capture exactly 80% of datapoints. Extents of whiskers capture exactly 99%

Conclusions and Future Work

- Consistent loop rates with low jitter at all frequencies
- Planned developments include robust logging module and further integration with ROS

We welcome the community to download, use and contribute to CORC at:
github.com/UniMelbHumanRoboticsLab/CANOpenRobotController

References

[1] CANOpenSocket: https://github.com/CANopenNode/CANopenSocket
[2] Fourier Intelligence, X2 http://www.fftai-global.com/lower-extremity/
[3] Fong, J, et al. ”EMU: A transparent 3D robotic manipulandum for upper-limb rehabilitation.” ICORR 2017. pp. 771-776

*University of Melbourne and Fourier Intelligence Joint Laboratory, The University of Melbourne
[$]McCormick School of Engineering, Northwestern University
[†]Legs + Walking Lab, Shirley Ryan Ability Lab
Correspondence: fong.j@unimelb.edu.au

WeRob 2020