



Interfacing ErgoJr with Creative Coding Platforms

Downloaded from: <https://research.chalmers.se>, 2026-05-10 20:34 UTC

Citation for the original published paper (version of record):

Caravati, M., Tatar, K. (2024). Interfacing ErgoJr with Creative Coding Platforms. ACM International Conference Proceeding Series. <http://dx.doi.org/10.1145/3658852.3659082>

N.B. When citing this work, cite the original published paper.



Interfacing ErgoJr with Creative Coding Platforms

Matteo Caravati
ENSEIRB-MATMECA
Talence, France
mcaravati@bordeaux-inp.fr

Kıvanç Tatar
Chalmers University of Technology
Gothenburg, Sweden
tatar@chalmers.se

ABSTRACT

This paper introduces a project enabling non-coders to control a Poppy Ergo Jr. robotic arm with Dynamixel servomotors. Originally using a Raspberry Pi and Pixl board, various constraints related to importation led to adopting a ROBOTIS OpenCM9.04 board. A client-server architecture was implemented for remote control, with creative coding platforms (p5.js, Processing, Pure Data, Python) as clients.

The server, utilizing a two-layer architecture, manages communication and interfaces with the ROBOTIS OpenCM9.04 board. The OSC and WebSocket protocols were chosen for communication due to their flexibility and their ease of use. Clients were developed for each platform, leveraging compatibility layers.

CCS CONCEPTS

• Human-centered computing → Interaction design.

KEYWORDS

Robot control, Creative coding, OSC, WebSocket, C++, Python, p5.js, Pure Data, Processing, OpenCM9.04

ACM Reference Format:

Matteo Caravati and Kıvanç Tatar. 2024. Interfacing ErgoJr with Creative Coding Platforms. In *9th International Conference on Movement and Computing (MOCO '24)*, May 30–June 02, 2024, Utrecht, Netherlands. ACM, New York, NY, USA, 4 pages. <https://doi.org/10.1145/3658852.3659082>

1 INTRODUCTION

The idea kickstarting this project was to allow people who are not really code-literate to control and program a robot built using ROBOTIS Dynamixel servomotors, in the current case the Ergo Jr. [16], a low-cost and open-source 6-DOF robotic arm for education developed by the Poppy Project (see figure 1 for a mechanical description of the robot).

The Poppy Ergo Jr. is composed of 6 x Dynamixel XL-320 servomotors [10], of a Raspberry 3B+ computer [20] to compute the movements and to work as a human interaction device (HID), and a PCB (Printed Circuit Board) that acts as the servomotors' control card and current regulator, called the Pixl board, that plugs directly on the GPIO pins of the Raspberry Pi board. The Poppy Ergo Jr. provides a custom Linux image for the Raspberry Pi with all the required software for the programming and the control of the robot.

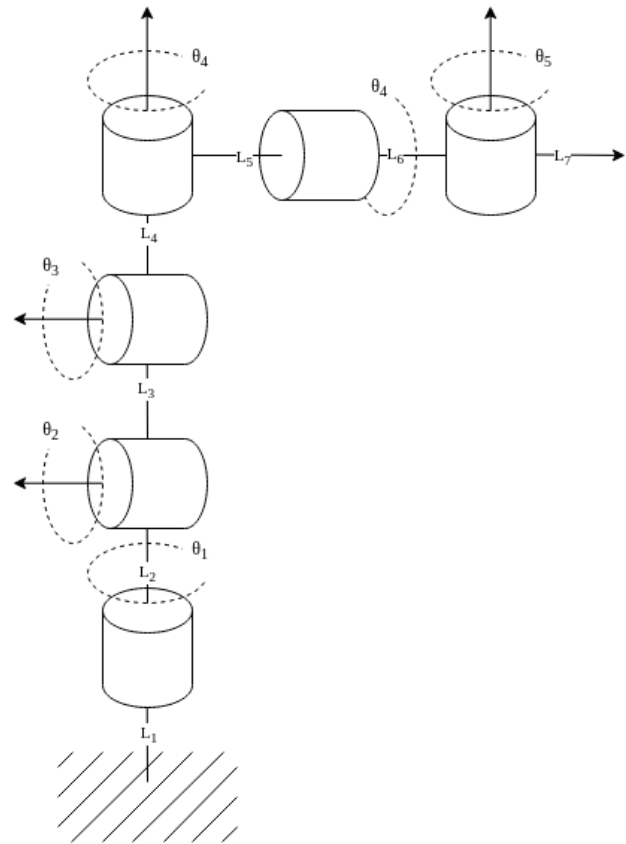


Figure 1: Poppy Ergo Jr. axis representation

Due to constraints regarding the importation of the Pixl board, the Raspberry Pi and the Pixl board were replaced by a ROBOTIS OpenCM9.04 embedded development board [11], which is based on an ARM Cortex-M3 32-bits processor and possesses 4 x MOLEX 53253-0370 connectors, whose are the connectors used by the Dynamixel XL-320 servomotors. The ROBOTIS OpenCM9.04 is compatible with the Arduino IDE, allowing the use of the Arduino ecosystem and its libraries.

To interface the ROBOTIS OpenCM9.04 that controls the Poppy Ergo Jr. with the multiple creative coding platforms, a suitable software has been developed, based on a client-server architecture for the users to be able to control the robotic arm remotely.

Computational creative tools are known to expand creative possibilities [19] and can have applications in numerous domains such as medical rehabilitation [18] or psychological therapy [17]. They favour art and expression rather than functionality, and creative



This work is licensed under a Creative Commons Attribution International 4.0 License.

MOCO '24, May 30–June 02, 2024, Utrecht, Netherlands

© 2024 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-0994-4/24/05

<https://doi.org/10.1145/3658852.3659082>

platforms provide a simplified syntax for the programming language they use as well as multi-media interfaces such as a graphic interface or a sound interface. That is why they were chosen as the client software.

The target platforms are:

- p5.js: built around JavaScript and primarily made to run on a web page.
- Processing: standalone IDE (Integrated Development Environment) with its own overlay for the Java language.
- Pure Data: a visual programming language for multimedia that allows the development of custom features through extensions called *externals*.
- Python: a general-purpose, high-level, interpreted programming language.

2 CONTROL SERVER

The server has been conceived using a two-layers architecture (see class diagram 2):

- The first layer takes care of the communication with the clients through the local network: it is composed of a simple abstract class allowing the implementation of a communication layer for multiple protocols by using class inheritance.
- Once received through the network, the commands are passed to the second layer which is represented by an interface containing the signatures of the operations that will be implemented by the ROBOTIS OpenCM9.04 development board. In our case, two classes implement this layer:
 - A first class that is able to communicate with the ROBOTIS OpenCM9.04 development board using a USB serial connection, allowing direct control of the robotic arm.
 - A second class that is a straightforward dummy implementation of the robot, simply printing the commands it receives on the standard output, for testing purposes.

3 FIRMWARE DEVELOPMENT

To be able to develop an efficient firmware, the Dynamixel2Arduino [12] library was used for the communication and control of the Dynamixel XL-320 servomotors. For a complete list of operations, that were implemented by the firmware, see table 1.

A consideration in the design of the server / devboard protocol was that the ROBOTIS OpenCM9.04's processor (an ARM Cortex-M3 32-bits processor) operates in little-endian format, so the decision was made to transmit data in this particular order, allowing us to directly transfer "complex types" through the serial connection. By "complex types", we mean here 32-bits integers and 32-bits floating-point numbers, as they are coded on more than one byte. From the server side, the data is sent through the serial connection by encoding it using the Python struct library [15] in little endian format[6]. The Arduino standard library allows the boards to use the memcpy function, copying directly the data from the serial stream in raw form to a variable's address, this way the data can be transferred as it is, without the need for manipulation or conversion to make it compatible between the two devices.

4 CREATIVE CODING PLATFORMS LIBRARIES

The core library is composed of a shared interface that is implemented by a class for each protocol that can be used to communicate with the server (see class diagram 3).

To benefit from a robust language that has a strong typing system and that has bindings to almost any popular and modern language, C++ was chosen to write the core library. C++ also allows object-oriented programming, providing the possibility to control multiple arms at the same time.

The go-to protocol for the core library to use while communicating with the server is the OSC (Open Sound Control) protocol, because of its flexibility, its use in the art world, and because it is a protocol running via UDP (User Datagram Protocol), allowing faster transfer and processing times. For the core library to use the OSC protocol, the tinyOSC library was used [3].

A Python client was developed by using a C/C++ to Python compatibility layer. Here, the Boost's Python [4] library was used, allowing a simple seamless integration of the C++ class to Python's environment, as opposed to other existing layers such as the ctypes [13] or the Python C API [14].

Developing a client for Processing was as straightforward; the creative coding platform's programming language is mainly a Java overlay, so almost any vanilla Java code can be used with Processing. In this case, javacpp was used as the compatibility layer between C++ and Java [7] and Maven was used to wrap the building process and dependencies management [9].

As explained earlier in this paper, Pure Data allows the creation of custom features through externals. Multiple libraries and frameworks exist for anyone to write externals, flext was used there [1], allowing for the compilation of C++ code in Pure Data externals.

A JavaScript client was compiled for p5.js using Emscripten [2], a C++ to WebAssembly compiler using LLVM [8]. WebAssembly is a low-level and platform-independent binary instruction format aimed for a stack-based virtual machine, supported by most modern browsers and JavaScript runtimes such as Chrome, Firefox, Safari or Node.js.

When a p5.js sketch is run on a web page, the code is executed through the web browser's JavaScript runtime. In this case, the OSC protocol can not be used as modern web browsers impose security constraints regarding the creation of UDP connections; to allow p5.js to communicate with the server anyway, WebSocket will be used.

WebSocket allows the creation of lightweight bidirectional connections via TCP and do not send a packet header at each request as opposed to other web-based TCP protocols such as HTTP, allowing devices and programs to process messages faster and to achieve a *real-time* control of the arm.

The peer-to-peer protocol WebRTC was also considered, but while its API is well-documented and well-developed, the protocol itself is a bit more complex than WebSocket and can it can be really hard to grasp the internal concepts. WebRTC also uses third-party web servers for the peers to establish a connection between one another, such as signaling servers [5], complexifying the connection process beyond what is needed here.

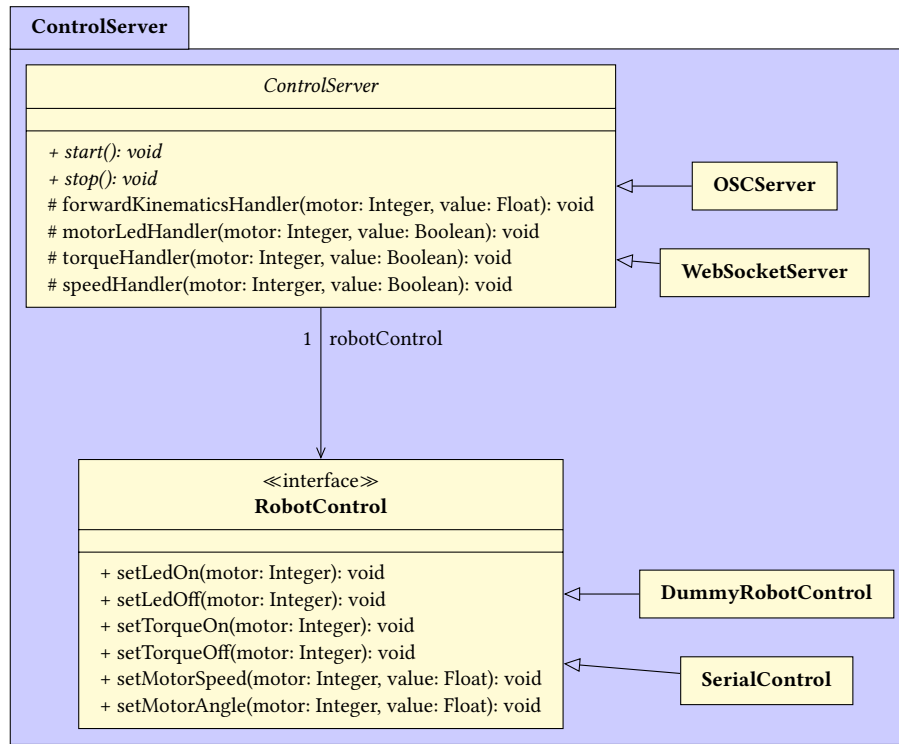


Figure 2: The server's class diagram

Operation name	OP code	Nb of arguments	Arguments types
Toggle LED	0x01	2	Integer (motor ID) / Integer (1 for On, 0 for Off)
Toggle Torque	0x02	2	Integer (motor ID) / Integer (1 for On, 0 for Off)
Set speed	0x03	2	Integer (motor ID) / Integer (speed)
Set position	0x04	2	Integer (motor ID) / Float (angle)

Table 1: Operations implemented by the firmware

5 CONCLUSION

In summary, this project aimed to make easier the control and programming of a low-cost robotic arm such as the Poppy Ergo Jr by interfacing it with creative coding platforms. The original design using a Raspberry Pi was adapted to the ROBOTIS OpenCM9.04 due to import constraints. A client-server architecture facilitated remote control through the creative coding platforms like p5.js, Processing, Pure Data, and Python.

A firmware was developed using the Dynamixel2Arduino library to ensure efficient communication with the multiple Dynamixel XL-320 servomotors. C++ was chosen to develop the core library, using the OSC and WebSocket protocols for communication.

Client libraries were created for Python, Processing, Pure Data, and p5.js, offering a range of programming options. This project successfully merged creative coding with robotic control, making it accessible to a wider audience. Future work could explore additional platforms and enhance the server's capabilities for broader usability.

ACKNOWLEDGMENTS

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program – Humanities and Society (WASP-HS) funded by the Marcus and Amalia Wallenberg Foundation.

REFERENCES

- [1] 2004. Flex. <https://svn.grrrr.org/public/pub/grill-2004-flex.pdf>
- [2] 2015. Main – Emscripten 3.1.52-git (dev) documentation. <https://emscripten.org/>
- [3] 2018. TinyOSC. <https://github.com/mhroth/tinyosc>
- [4] 2019. Boost.Python - 1.72.0. https://www.boost.org/doc/libs/1_72_0/libs/python/doc/html/index.html
- [5] 2021. Getting started with peer connections | WebRTC. <https://webrtc.org/getting-started/peer-connections>
- [6] 2023. Endianness. <https://en.wikipedia.org/w/index.php?title=Endianness&oldid=1191109382> Page Version ID: 1191109382.
- [7] 2023. javacpp. <https://github.com/bytedeco/javacpp>
- [8] 2023. The LLVM Compiler Infrastructure Project. <https://llvm.org/>
- [9] 2023. Maven. <https://maven.apache.org>
- [10] 2023. ROBOTIS e-Manual. <https://emanual.robotis.com/docs/en/dxl/x/xl320/>
- [11] 2023. ROBOTIS e-Manual. <https://emanual.robotis.com/docs/en/parts/controller/opencm904/>
- [12] 2023. ROBOTIS-GIT/Dynamixel2Arduino. <https://github.com/ROBOTIS-GIT/Dynamixel2Arduino> original-date: 2019-04-30T05:08:15Z.

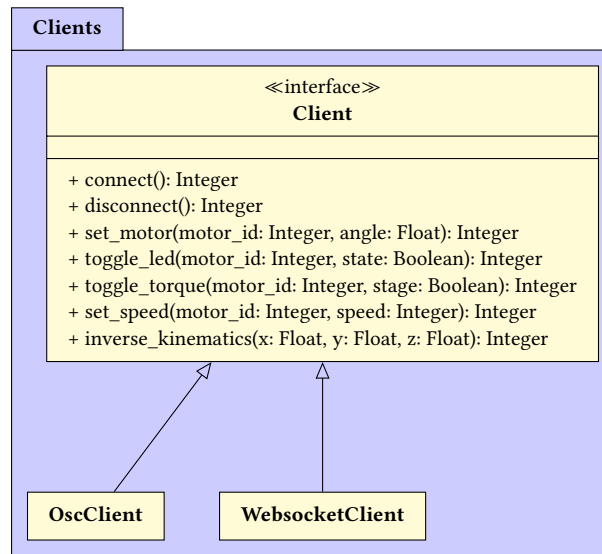


Figure 3: Clients' class diagram

- [13] 2024. ctypes – A foreign function library for Python. <https://docs.python.org/3/library/ctypes.html>
- [14] 2024. Extending Python with C or C++. <https://docs.python.org/3/extending/extending.html>
- [15] 2024. struct – Interpret bytes as packed binary data. <https://docs.python.org/3/library/struct.html>
- [16] Thibault Desprez, Stéphanie Noipoudre, Théo Segonds, Damien Caselli, Didier Roy, and Pierre-Yves Oudeyer. 2018. Poppy Ergo Jr : un kit robotique au coeur du dispositif Poppy Éducation. 1. <https://inria.hal.science/hal-01753111>
- [17] Sooyeon Jeong, Laura Aymerich-Franch, Sharifa Alghowinem, Rosalind W. Picard, Cynthia L. Breazeal, and Hae Won Park. 2023. A Robotic Companion for Psychological Well-Being: A Long-Term Investigation of Companionship and Therapeutic Alliance. In *Proceedings of the 2023 ACM/IEEE International*

Conference on Human-Robot Interaction (Stockholm, Sweden) (HRI '23). Association for Computing Machinery, New York, NY, USA, 485–494. <https://doi.org/10.1145/3568162.3578625>

- [18] Elizabeth Jochum, Andreas Kornmaaler Hansen, and TD Murphey. 2023. Drawing Robots for Rehabilitation. In *Ergonomics in Robotics: Advances and Innovations (ERGOROB)*. Springer.
- [19] Daniel Lopes, Jéssica Parente, Pedro Silva, Licinio Roque, and Penousal Machado. 2023. Can Creativity be Enhanced by Computational Tools?. In *International Conference on Computational Creativity, Waterloo, Canada*. https://computationalcreativity.net/iccc23/papers/ICCC-2023_paper_72.pdf.
- [20] Raspberry Pi Ltd. 2023. Raspberry Pi. <https://www.raspberrypi.com/>

Received May 14, 2024