Report on the D1 Mechanical Arm Project

Proposing General Applications of the D1 Mechanical Arm for Educational Purposes

Author: Anas Malhouq | Ali Mohamed Benachou

Institution: Polytech Nice-Sophia, University Côte d'Azur

Course: Robotics & Autonomous Systems

Date: 10/04/2025

1. Abstract

This report documents the development and application of the D1 Mechanical Arm, designed for educational purposes as part of the Robotics & Autonomous Systems formation at Polytech Nice-Sophia. The project demonstrates key concepts in robotics, including motor control, sensor integration, and ROS-based software development. Through case studies and code implementations, the report highlights the methodology behind controlling a brushless motor, integrating software libraries, and leveraging hardware interfaces provided by Unitree. The project's objective is to provide students and researchers with a practical platform for exploring advanced robotics techniques.

2. Introduction

Robotics is an interdisciplinary field that combines mechanics, electronics, and software engineering. The D1 Mechanical Arm project was initiated to bridge theoretical knowledge with practical application in an educational environment. The arm serves as a demonstrator of how modern robotic systems can be controlled using state-of-the-art frameworks such as ROS (Robot Operating System). This project was carried out under the auspices of Polytech Nice-Sophia, with particular emphasis on exploring general applications of robotic arms in both academic and real-world scenarios.

Key objectives of the project include:

- Demonstrating the control of the arm manipulator via ROS.
- Showcasing modularity in design to adapt to various application cases.
- Providing a reproducible platform for further research in robotic systems.

3. Progress Timeline

Making use of our two-months-long project time-span, we divided our work into multiple sections to increase efficiency. The work proceeded as follow:

Project Sessions	Tasks
Session 1~3	Documentation: Due to the lack of guidelines and resources concerning the D1-550, we used these sessions to collect as much data, from both Unitree's official documentation and particular individuals' websites, githubs, and such.
Session 4~5	The supplied code was not compiling and executing correctly. This time was spent debugging and re-arranging the code to suit our needs.
Session 6~7	This time was spent troubleshooting both software and hardware problems concerning the servo motors, as some necessary services and kits were not made available by default. A bit of research was needed in order to pinpoint which services were necessary.
Session 8	After making sure the robotic arm worked as intended, we dedicated this time writing the report, making the visual support for the oral presentation, as well as fine-tuning our GitHub repository.

4. Project Overview

The project repository (GitHub Repository) is structured into several key components:

- **Guide:** Documentation that explains the steps necessary to set up and operate the mechanical arm, ranging from full documentation list, working environment setup, all the way to the different codes needed for a \n accurate control of the arm. A section for faced issues and future improvement is also in place.

- **Technical Sheet:** A detailed description of the arm's specifications, hardware interfaces, and design considerations., according to the manufacturer. It is useful for determining the arm's power supply requirement and the range of the end-effector, as well as the different ways of communication with the robot.

- **Code Section:** Contains the source code and scripts required for controlling the arm's actuators. A concise explanation of the pieces of code needed for the arm control. We can distinguish between the Unitree's Interface code, which is in Json format and can be fully understood by the robot's integrated controller, and the Unitree's SDK code, which is written in C++ and can be rearranged to suit our needs. It is compiled and executed through the ROS environment in an Ubuntu terminal, and serves to send the Json commands to the arm.

The repository also provides references to several Unitree resources, including:

- The Unitree D1 Mechanical Arm Services Interface.

- Unitree's ROS environment setup.

- The Software Development Kit (SDK).

- Documentation for the Open Manipulator X ROS Control.

These resources were integral to both the conceptual design and practical implementation of the project.

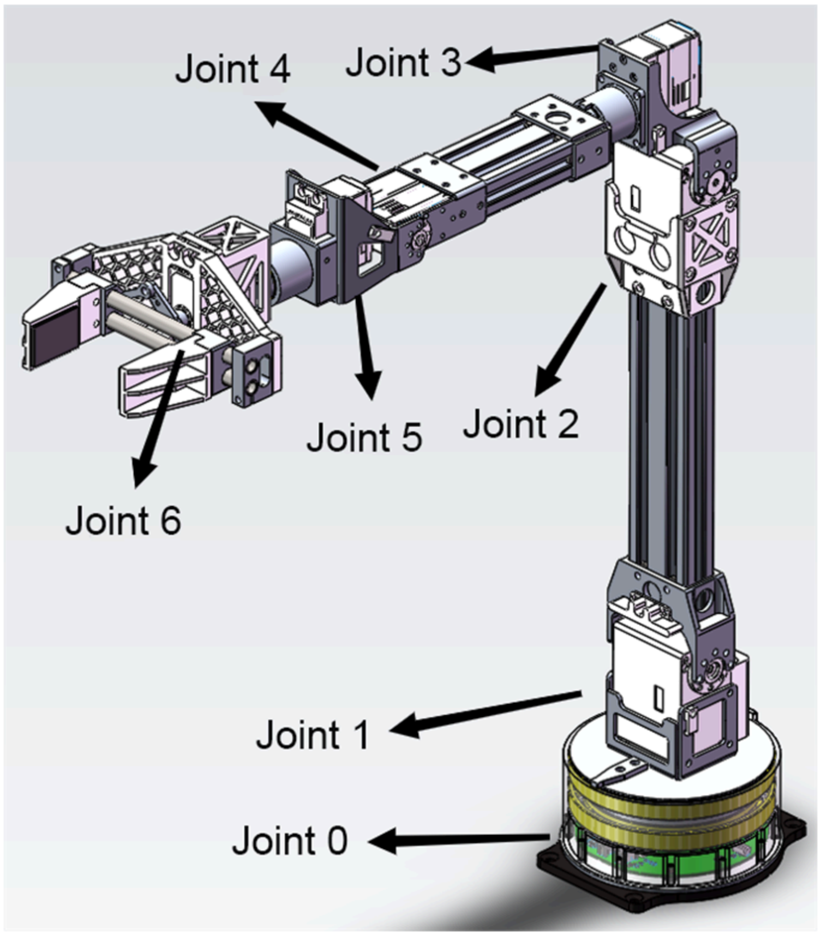
5. System Architecture and Design

4.1. Hardware Components

The D1 Mechanical Arm comprises several hardware components, including:

- **Bus-Servo Motors:** 7 in total. These provide high-efficiency actuation with precise control, essential for smooth arm movements. They allow for a rotation of up to 90º~135º and a torque of 1.7~3.3N.M depending on their position on the arm. They can support a maximum payload of 500g. They have integrated encoders for the current angle and position feedback.

- **Controller Units:** Unitree's embedded controllers that interface with the motors, facilitating communication between hardware and software layers.
- **Gripper:** The end-effector of the arm manipulator is a gripper controlled with one of the servos.



4.2. Software Environment

The software side of the project relies heavily on ROS, as well as Unitree's own SDK (Software Development Kit). Key features include:

- **ROS1 Noetic:** Individual nodes are responsible for specific tasks such as sensor data processing and actuator control.

- **Message Passing:** ROS1 topics and services allow for seamless communication between nodes, ensuring synchronized operation of the mechanical arm.

- **Code Organization:** The code base is organized to separate low-level motor control routines from higher-level planning algorithms. This separation enhances readability and maintainability.

The integration with ROS1 not only provides robust communication protocols but also allows the project to leverage a wide range of existing ROS packages and tools.

It must be noted that the D1-550 arm is only capable of receiving .json type commands.

6. Implementation Details

6.1. Control of Arm Movement

A central feature of the project is the precise control of the brushless motors that drive the arm's movements. The control algorithm was designed to ensure smooth transitions and accurate positioning. Key aspects of the implementation include:

- **Feedback Loops:** Servos' encoders continuously provide data to correct deviations in real-time.

- **Command Interfaces:** Specific ROS1 messages are defined for sending control commands, which are then translated into motor actions.

These components work in tandem to create a responsive and reliable control system that is crucial for educational demonstrations and experimental setups.

6.2. ROS1 Integration

The choice of ROS1 was driven by its advanced features and improved real-time performance over its predecessor. The integration process involved:

- **Setting up the ROS Environment:** Configuring the development environment to include necessary packages and dependencies, as outlined in Unitree's ROS environment setup.

- **Node Communication:** Establishing a robust communication framework using ROS topics and services to manage sensor data and control signals.

- **Modular Code Development:** Writing reusable code modules that handle different aspects of the arm’s operation. This modular approach facilitates testing, debugging, and future enhancements.

The repository’s “Guide” section provides step-by-step instructions on setting up the ROS environment, ensuring that other researchers can replicate the setup with minimal effort.

7. Testing and Validation

Testing played a crucial role in the development of the D1 Mechanical Arm project. A combination of simulation and real-world trials was employed:

- **Integration Testing:** Once individual modules were verified, the entire system was integrated and tested to ensure all components interacted correctly. Using feedback from the robotic arm on the state of its motor, and through iteration, we succeeded in controlling precisely the trajectory of the arm.

These tests confirmed that the system meets the design specifications and can serve as a reliable platform for educational experiments.

8. Challenges and Future Work

Challenges

During the course of the project, several challenges were encountered:

- **Integration Complexity:** Synchronizing hardware and software components required careful planning and numerous iterations. Some pieces of code were not compiling correctly due to syntax error or errors in environment setup.

- **Real-time Constraints:** Ensuring the system's responsiveness, particularly when dealing with high-speed actuators, was a constant challenge. A problem we faced was related to the motors being unresponsive. Even when supplying them with a 24V, they were only demanding about 0.5A, when the normal value would be around 10 according to the Unitree supplier.

- **Documentation:** There were not enough resources to work with at the beginning of the project. We dedicated a lot of our time working on collecting many ressources from various documentation sites, some of them coming from Unitree, others from individuals.

Future Work

Based on the insights gained, several areas for future research and development have been identified:

- **Enhanced Sensor Fusion:** Integrating additional sensors could further improve the arm's precision and adaptability. Some potential upgrades could include implementing touch sensors for the end-effector and cameras for interacting with the environment.

- **Advanced Control Algorithms:** Implementing more sophisticated control strategies (e.g., model predictive control) could enhance performance. Also, it might be wise to simplify some of the codes as they might prove challenging for newcomers.

- **Extended Applications:** Exploring additional applications, such as collaborative robotics or adaptive learning environments, all making use of our educational background, could broaden the project's impact.

- **User Interface Improvements:** Developing a more intuitive interface for controlling the arm would increase its usability in educational settings.

9. Conclusion

The D1 Mechanical Arm project successfully demonstrates the integration of hardware and software in a modern robotics application. Through the use of ROS, Unitree's own SDK, and our own technological knowledge, the project provides a solid foundation for both academic and research-oriented exploration in robotics. The hands-on experience gained from addressing real-world challenges—such as sensor integration, motor control, and system synchronization—highlights the value of applied learning in advanced robotics education.

The report has outlined the project's objectives, system architecture, implementation details, testing procedures, and the challenges encountered. With proposed avenues for future work, the project not only stands as an educational tool but also opens up new possibilities for research and development in robotic systems.

10. References

1. Unitree's D1 Mechanical Arm Services Interface. Available at: support.unitree.com

2. Unitree's ROS Environment Setup. Available at: Unitree ROS Setup GitHub

3. Unitree's Software Development Kit (SDK). Available at: Unitree SDK

4. Open Manipulator X ROS Control. Available at: Open Manipulator Documentation