National Technical University of Athens
MSc in Control Systems & Robotics

Robotics Laboratory

myCobot 280 NVIDIA's Collaborative Robot

Ronaldo Tsela (02124205), Theodoros Fragos (02124208), Andreas Skoufis (02124201)

April 22, 2025

Contents

Abstract

We live in an era where industrial automation is revolutionizing industries around the world. Two types of autonomous devices are at the forefront of this transformation: Cobots and Robots. While both are widely used to enhance efficiency, productivity, and safety, their distinct characteristics and applications make them suitable for different tasks. Classical robotic manipulators are typically large and powerful machines, capable of handling repetitive and often hazardous tasks, replacing humans. While working close to an industrial manipulator is dangerous, collaborative robots, or commonly known as 'cobots', are designed to co-exist safely alongside humans in a shared workspace.

In this report paper we introduce the NVIDIA's myCobot 280 Jetson Nano, a 6-DOF robotic arm used as part of the "Robotics Laboratory" lectures at NTUA for the MSc program in "Control Systems and Robotics" for the 2024/2025 academic year. Specifically we will explore the NVIDIA myCobot 280 Jetson Nano platform, highlighting its key features, characteristics and explanatory use cases. We will familiarize with the programming methods available for the device, including its Python API, intuitive high-level command passing, and kinesthetic teaching. Finally, we will examine the forward kinematics of the manipulator and assess its actual performance and accuracy compared with the analytical solutions.

Keywords: myCobot Jetson Nano, collaborative, control, kinesthetic teaching, forward kinematics, command

1 Introduction

We are living in an era where industrial automation is revolutionizing industries around the world by accelerating processes, performing tasks that most humans are physically unable to do, and optimizing production in terms of both speed and quality. The key feature of this industrial automation of-curse are autonomous machining tools and robots. Specifically, industrial robots are autonomous machines designed to perform tasks without direct human intervention. They are commonly used for repetitive, high-precision, or/ and hazardous tasks that would be difficult or dangerous for humans to handle [4, 1]. You can find these robots in production lines, such as in car manufacturing, packaging processes, and the management of heavy equipment. Common applications of industrial robots include welding, painting, material handling, and packaging, among many other applications and across multiple industrial domains [4].

Industrial robotic manipulators are typically large, fast, and highly powerful machines with a high payload capacity, a wide range of motion and capable of produce high forces and torques. Due to their size and capabilities, they are often enclosed in safety cages or barriers to protect human workers from potential hazards when they need to co-exist in the same environment [4].

Although robotic automation is widespread, and many industries benefit from automating several tasks and processes replacing human effort, the human factor remains essential in industrial settings. There are certain tasks that robots cannot perform independently, and thus requiring human collaboration [4]. This new trend of robot-human collaboration is commonly known as "human-in-the-loop".

This is where cobots come into play. The term "cobot" stands for "collaborative robot" [1]. These type of robots are specifically designed to work alongside humans in a shared workspace. Cobots are built with features that prioritize safety, flexibility, and ease of use as opposed to the classical large and "rigid" industrial robotic manipulators [4, 1]. They are equipped with force and torque sensors that enable them to detect and respond to contact with humans or objects, ensuring a safe working environment for them. Their usually lightweight and compact design allows for easy movement and reconfiguration, providing greater adaptability for a variety of tasks compared to large industrial robots. Common applications of cobots include assembly, pick-and-place, machine tending, quality inspection and of-curse in medical applications (e.g. The da Vinci Surgical System) [4].

Cobots are generally more affordable than industrial robots, with lower upfront costs. Their ease of programming and integration, combined with their flexibility and adaptability, helps reduce setup and training expenses. Additionally, cobots often eliminate the need for costly safety infrastructure, such as cages or barriers, which further lowers overall costs. These advantages make cobots a popular choice in the academic community for research and educational purposes [4].

In this report, we will explore a small academic robotic manipulator designed exactly for collaborative tasks, called the myCobot 280, powered by NVIDIA's Jetson Nano system-on-module (SoM) single-board computer. The next section will introduce the key characteristics of the NVIDIA myCobot 280 Jetson Nano cobot platform, as well as the setup used for the purposes of this lab exercise. In the third section, we will familiarize with the programming interface of the myCobot and examine the simple programming of a pick-and-place task. In the fourth section we will explore kinesthetic teaching as an alternative programming method for the myCobot 280. Next in the fifth section will focus on joint space, providing a forward

kinematic analysis of the robotic manipulator using Denavit-Hartenberg parameters to describe the robot, along with an evaluation of the actual positioning error. This report concludes with some thoughts and comments on the lab experience.

2 Hardware Setup

2.1 Introduction to NVIDIA's myCobot 280 Jetson Nano

The myCobot 280 Jetson Nano is a compact, 6-axis (6-DOF) collaborative robotic arm developed by Elephant Robotics, powered by NVIDIA's Jetson Nano single-board module. This platform combines robotic control with edge computing in a small footprint, making it ideal for education, research, and prototyping purposes. The arm features a payload capacity of 250 grams, a reach of 280 mm, and offers a repeatability of 5 mm, according to the manufacturer.

The Jetson Nano is an advanced edge computing module based on NVIDIA's Jetson Nano SoM, specifically designed for on-board machine learning, computer vision, and robotics control. It enables efficient AI processing directly on the device, eliminating the need for cloud-based computation (of-curse constrained to certain limitations!) [5].

The Jetson Nano board is compact (70×45 mm) and power-efficient, consuming up to only 10 watts at full operation, which makes it well-suited for embedded systems and stand-alone edge devices. Its popularity in onboard AI applications comes from its Maxwell-based GPU featuring 128 CUDA cores, capable of delivering up to 0.5 TFLOPs of performance. Furthermore the main processing system is powered by a quad-core ARM Cortex-A57 CPU and equipped with 4 GB of LPDDR4 RAM [5].

The board also supports a wide range of I/O interfaces and peripherals, including Gigabit Ethernet, HDMI 2.0, DisplayPort 1.4, two DSI connectors, an M.2 PCIe slot, four USB 3.0 ports, and two CSI camera connectors [5].

Among the many benefits of using the myCobot 280 Jetson Nano, another important feature is that myCobot platform is completely open-source, from its software stack (with the exception of proprietary NVIDIA drivers) to its hardware design (again except for the NVIDIA's stack). This openness allows users to access, modify, and extend both software and hardware components, making it a versatile tool for hands-on development, system integration, experimentation and research.

Additionally, as an open-source and relatively popular platform, myCobot benefits from an active community which subsequently means extensive support resources available. This makes it accessible to users of varying proficiency levels, offering a soft learning curve and encouraging the development of innovative applications in robotics.

2.2 Mechanical Characteristics of myCobot 280 Jetson Nano

The mechanical configuration of the myCobot 280 Jetson Nano robotic arm and its operating workspace is illustrated in Fig. 1. The robot is mounted on a stable desktop platform, and the workspace includes the reachable volume within which the arm can operate without violating its mechanical constraints as depicted in the aforementioned Fig. 1.

The reference frame is at the center of the robot's base, following a global cartesian coordinate system.

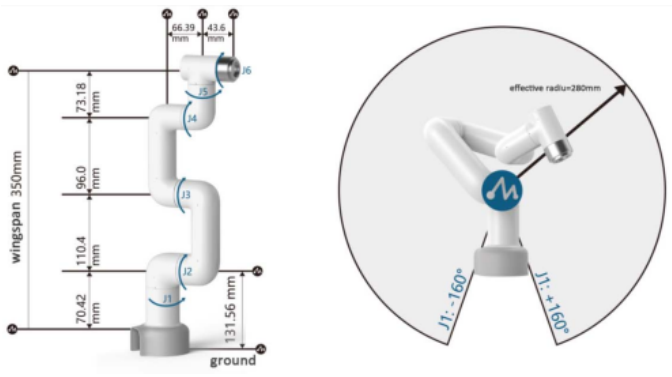

Figure 1: Mechanical configuration and workspace of the myCobot 280 Jetson Nano

The robot's orientation and motion are defined using Euler angles (roll, pitch, and yaw) alongside the axes of the reference frame.

Mechanical limitations define the invalid configurations of the robot. These occur primarily when the arm is fully extended along the x and/ or y axes, or when it attempts to exceed the lower z-axis boundary defined by the table surface. Although the nominal maximum reach is approximately 280 mm, in practice, the arm cannot fully extend due to its own weight and torque limitations, as noted in the lab instructor.

The joint-level motion constraints for each degree of freedom are provided in Table 1. Specifically, all joints are constrained to a range of motion slightly less than a full 360-degree rotation, from -165 to +165 degrees, with the exception of the final joint (J6), which has a slightly wider range, operating between -175 to +175 degrees respectively.

Table 1: Joint range of motion in degrees

Joint	**Range** (°)
J1	-165 to +165
J2	-165 to +165
J3	-165 to +165
J4	-165 to +165
J5	-165 to +165
J6	-175 to +175

2.3 Programming and Interfacing myCobot 280 Jetson Nano

The integration of the Jetson Nano-based platform enables the myCobot 280 to perform high-performance tasks such as real-time object detection, gesture recognition, and autonomous interaction, while simultaneously providing efficient control of the robot's joints. Its embedded Maxwell-based NVIDIA GPU makes it particularly well-suited for edge AI applications, where low-latency processing is crucial for responsive robotic behavior.

The robot supports a wide range of programming environments, from low-level C and C++ development to high-level Python scripting for both control logic and task execution. One of its key advantages is its native support for the Robot Operating System (ROS), which simplifies the development of complex, modular robotics projects. For beginners or those without prior programming experience, myBlockly—a visual programming framework—offers an intuitive alternative to text-based coding. Additionally, the robot's compatibility with various sensors and peripheral modules further enhances its adaptability for diverse experimental and research applications.

Furthermore the Jetson Nano runs a Linux distribution based on Ubuntu MATE, on a dual-core ARM Cortex A57 processing system, which provides a familiar desktop-like environment and relatively fast response times although it is an embedded device. This makes interfacing and interactions with the device simple, and executing programmed tasks and accessing system resources is relatively straightforward—especially for users with basic Linux experience.

Although there are multiple ways of interfacing, programming and interacting with the device, the most common method is through Python scripting. Specifically Elephant Robotics provides an official Python package called pymycobot, available via their GitHub repository (URL). This package offers support for multiple myCobot models, including the 260, 270, 280, and 320 variants respectivelly.

The API for the myCobot 280 can be found here (although in Chinese...). In the following section, we will explore a selection of these commands as part of our hands-on familiarization with the device as part of the lab exercise.

2.4 Lab Setup Configuration

The lab setup for this exercise consists of the myCobot robotic manipulator mounted on a desktop surface and connected to a monitor, keyboard, mouse, and, of course, a power source. The robot's end effector is capable of integrating various tools; in our case, a simple gripper is used. The complete setup is shown in Fig. 2. In this figure, several objects designated for manipulation are visible-the green cube is the target object-, along with the global reference frame at base and the coordinate frame of the end effector.

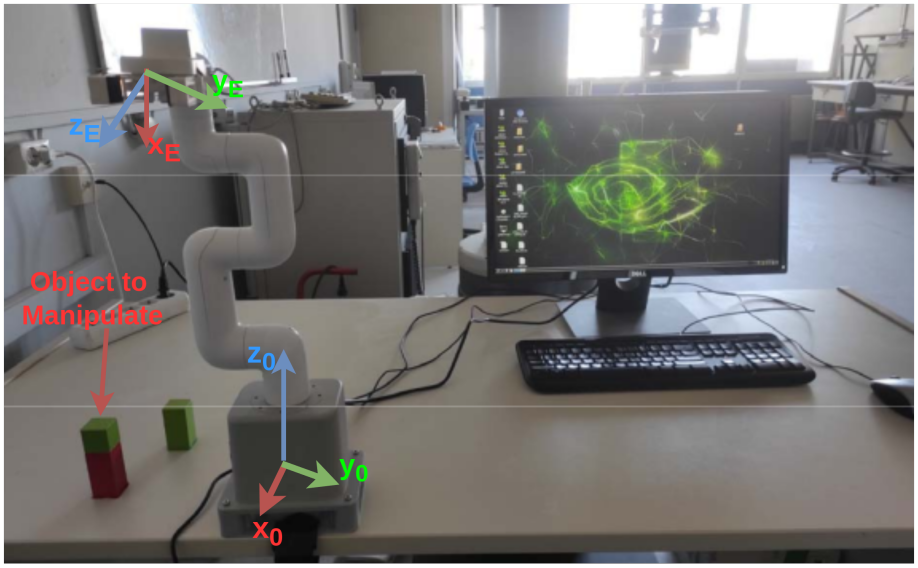

Figure 2: The lab setup for familiarizing with the myCobot 280 Jetson Nano.

3 Familiarizing With myCobot 280

The first part of the lab exercise involves programming the robot to perform a simple pick-and-place task from position A to position B. This task helped us become familiar with the programming environment.

The robot is controlled using the Python API mentioned earlier, but here it is accessed through higher-level task-specific commands provided from a user-defined TXT file. The commands utilized in this example are shown bellow:

Table 2: High-Level Command Syntax for Robot Programming

Command Syntax	**Description**
`set_coords: X, Y, Z`	Sets a target position for the robot to move to. The coordinates (X, Y, Z) represent the position of the end effector with respect to the robot's base reference frame, in mm.
`set_gripper_state: S`	Opens (`S=0`) or closes (`S=1`) the gripper attached to the end effector.
`set_coords_pick_pos: X, Y, Z`	Defines the objects pick position. Coordinates (X, Y, Z) are with respect to the robot's base reference frame, in mm.
`set_coords_place_pos: X, Y, Z`	Defines the target place position for the object. Coordinates (X, Y, Z) are with respect to the robot's base reference frame, in mm.

When coordinates (X, Y, Z) are provided, the robot computes the inverse kinematics and follows a linear path-planning approach to move through intermediate positions. As a result given the pick position is (-42.7, -232.3, 115.1) and the place position is (-147.1, -174.2, 117.7) we can naively prescribe the robot to execute the following program:

```
set_coords_pick_pos: -42.7, -232.3, 115.1
set_gripper_state: 1
set_coords_place_pos: -147.1, -174.2, 117.7
set_gripper_state: 0
```

However, this program does not ensure a proper pick-and-place action. Although the robot can successfully move the end effector to the defined pick position, executing the program as-is will likely result in the object being pushed rather than picked up. Even if the object is successfully placed at the new location, the robot may unintentionally drag the object when moving away from the target placement position.

To safely execute the pick-and-place task, intermediate positions (waypoints) must be used. The first waypoint should be chosen to place the gripper at a safe distance above the object, aligned and facing it. For example we can move the robot's end effector at the pick position's X and Y but keep Z axis for a few millimeters above the target Z position. From there, the robot can descend to the pick position and close the gripper to secure the object on a subsequent move. Next, the gripper—now holding the object—should move to another intermediate position above the target placement location, following exactly the previous approach. Once there, the robot can descend to the final place coordinates and release the object by opening the gripper. Finally, to avoid disturbing the placed object during the return motion, the robot should move with the open gripper back to a safe height above the object before returning to its home position. The sequence of the movement is depicted in Fig. 3.

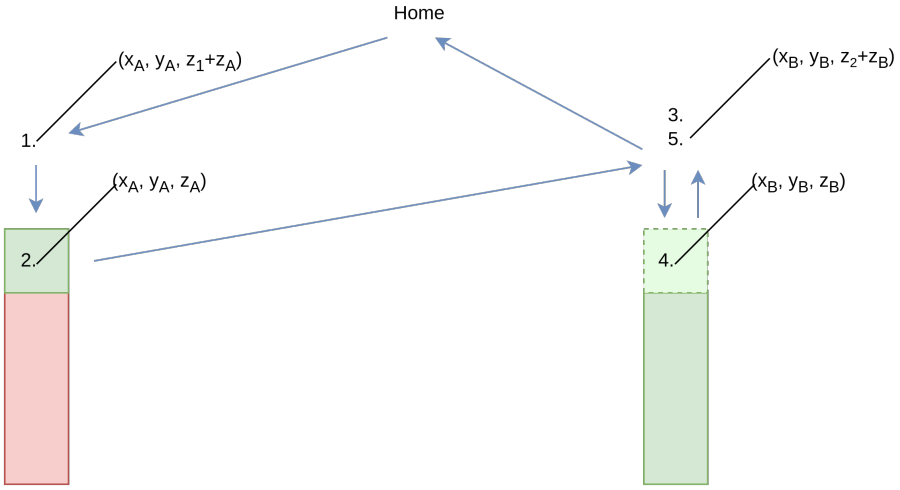

Figure 3: Sequence of movements for the proper pick-and-place task execution.

The program used in the lab to safely move the object from position A to position B as described above is shown below. A video demonstrating the successful execution of this task can be found in the accompanying material at "material/video-for-part-1.mp4", while a screenshot of the configured program's TXT file is shown in "material/image-for-part-1.png".

```
set_coords: -42.7, -232.3, 140.0
set_coords_pick_pos: -42.7, -232.3, 115.1
set_gripper_state: 1
set_coords: -42.7, -232.3, 140.0
set_coords_place_pos: -147.1, -174.2, 117.7
set_gripper_state: 0
set_coords: -42.7, -232.3, 140.0
```

4 Kinesthetic Teaching

One of the most notable features of small cobots, such as the myCobot 280 Jetson Nano that we introduced here, is their ability to learn tasks directly from human demonstration. In this process, the robot is physically guided by a human operator who "shows" it how to perform a desired motion, allowing the robot to mimic and repeat the motion afterwards. This technique is known as kinesthetic teaching and is based on direct physical interaction between the user and the cobot. Specifically in our occasion the user should manually moves the robot arm through the desired positions, and record the motion [3].

Kinesthetic teaching is especially advantageous in scenarios involving complex geometries or tasks that would be difficult to program explicitly through numerical methods as we previously did. By physically demonstrating the required motion, users can quickly test different movement sequences, enabling rapid prototyping. Moreover, this approach eliminates the need for complex coding making it accessible to users without advanced-or any at all- programming skills required [2, 3].

This teaching method complements traditional programming by enabling quick setup of basic tasks such as pick-and-place. It can also serve as the foundation for more advanced and refined motion sequences, which can later be optimized through coding or simulation.

In the context of the previously demonstrated pick-and-place task, kinesthetic teaching was used to guide the robot to the necessary positions without manually specifying spatial coordinates as the third part of the lab requests. A simple and intuitive graphical user interface was provided and utilized to record the end effector's joint positions. Thus we captured and stored the robot's joint encoder readings at each waypoint depicted in Fig 3.

After replaying the recorded waypoints, the robot was able to successfully repeat the pick-and-place task. The video `material/video-for-part-3.mp4` demonstrates the successful completion of this task. This video is provided in the accompanying material.

5 Forward Kinematic Analysis

The kinematics of the myCobot 280 Jetson Nano manipulator are described using the Denavit–Hartenberg convention with the corresponding kinematic diagram shown in Fig. 4, and the resulting DH parameter table presented in Table 3.

Using the DH parameter table, the forward kinematics of the robotic arm at hand can be derived using Eq. 1 which defines the homogeneous transform from frame $i-1$ to frame i for each link.

$$A_i^{i-1} = Rot(z,\theta_i) \cdot Tra(z,d_i) \cdot Tra(x,a_i) \cdot Rot(x,\alpha_i) \tag{1}$$

where, $Rot(k,\theta)$ defines a rotation by angle θ about axis k, and $Tra(k,d)$ defines a translation d along axis k. These elementary homogeneous transformations are defined as follows:

$$Rot(z,\theta_z) = \begin{bmatrix} cos(\theta_z) & -sin(\theta_z) & 0 & 0 \\ sin(\theta_z) & cos(\theta_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Tra(z,d_z) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Rot(x,\theta_x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\theta_x) & -sin(\theta_x) & 0 \\ 0 & sin(\theta_x) & cos(\theta_x) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Tra(x,d_x) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

By setting $q_4 = q_5 = q_6 = 0$ and defining $s_i \equiv sin(q_i)$ and $c_i \equiv cos(q_i)$, the homogeneous transform matrices simplify to:

$$A_1^0(q_1) = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & 131.22 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_2^1(q_2) = \begin{bmatrix} -s_2 & -c_2 & 0 & -110.4 \cdot s_2 \\ c_2 & -s_2 & 0 & 110.4 \cdot c_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2^3(q_3) = \begin{bmatrix} c_3 & -s_3 & 0 & 96\,c_3 \\ s_3 & c_3 & 0 & 96\,s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_3^E = \begin{bmatrix} 0 & -1 & 0 & 75.05 \\ 0 & 0 & -1 & -45.6 \\ 1 & 0 & 0 & 63.4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Therefore, the overall homogeneous transformation from the base frame to the end-effector is obtained by multiplying the individual aforementioned transforms:

$$T(q_1,q_2,q_3) = A_0^1(q_1) \cdot A_2^1(q_2) \cdot A_3^2(q_3) \cdot A_3^E = \begin{bmatrix} s_1 & s_{23}c_1 & c_{23}c_1 & p_x \\ -c_1 & s_{23}s_1 & c_{23}s_1 & p_y \\ 0 & -c_{23} & s_{23} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

where the cartesian position of the end-effector in space is given by the vector:

$$p_e = \begin{bmatrix} 63.4s_1 - 110.4c_1s_2 + 45.6c_{23}c_1 - 171.05s_{23}c_1 \\ 45.6c_{23}s_1 - 110.4s_1s_2 - 63.4c_1 - 171.05s_{23}s_1 \\ 171.05c_{23} + 45.6s_{23} + 110.4c_2 + 131.22 \end{bmatrix} \tag{3}$$

Table 3: The Denavit-Hartenberg parameter table for the myCobot 280 Jetson Nano robotic manipulator.

Link i	θ_i (rad)	d_i (mm)	a_i (mm)	α_i (rad)
1	q_1	131.22	0.0	$\pi/2$
2	$q_2 + \pi/2$	0.0	110.4	0.0
3	q_3	0.0	96.0	0
4	$q_4 - \pi/2$	63.4	0.0	$-\pi/2$
5	$q_5 - \pi/2$	75.05	0.0	$-\pi/2$
6	q_6	45.6	0.0	0.0

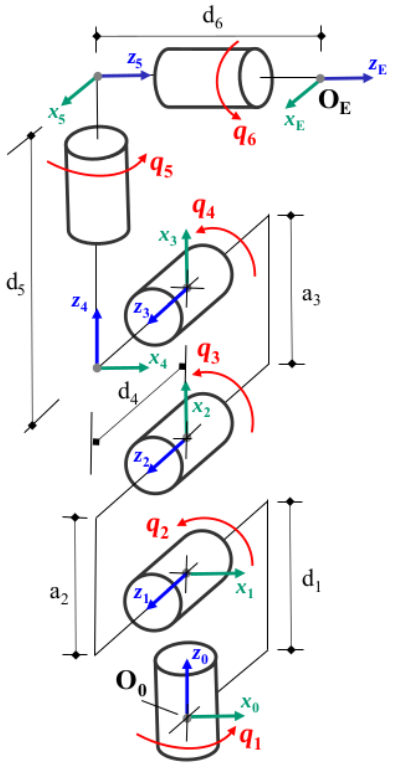

Figure 4: The kinematic diagram of the myCobot 280 Jetson Nano robotic arm. The frames are positioned according to Denavit-Hartenberg convention.

Therefore, in this experiment we evaluate the accuracy of the robotic arm in reaching user-defined positions. To do so, we use the interface program as mentioned before in this report and the high-level command *set_angles: A, B, C*, which sets the joints J1, J2 and J3 to positions defined by arguments A, B and C respectivelly. The values of A, B and C are given in degrees.

Table 4 lists the commanded joint configurations, the analytically computed end-effector positions (via (3)), the experimentally measured positions reported by the robot in millimeters, and the encoder readings in degrees for the three aforementioned joints respectively.

Table 4: Command inputs (in degrees) for joints J1, J2 and J3, theoretical end-effector positions computed via (3) (in mm), measured positions returned by the robot (in mm), and encoder readings (in degrees).

Command	Theoretical Position	Actual Position	Encoder Readings
set_angle: 30.0, -25.0, 30.0	98.536, -16.31, 405.65	100.834, -15.89, 404.61	29.44, -24.78, 29.35
set_angle: 30.0, -40.0, 40.0	132.64, 3.37, 386.84	138.03, 5.594, 384.01	29.61, -40.51, 39.28
set_angle: 30.0, -55.0, 50.0	162.27, 20.48, 360.96	168.0, 22.84, 357.1	29.7, -55.63, 49.21

Using the data from the table above, we can compute the position deviation from the expected end-effector position as well as the encoder readings error with respect to the commanded joint configurations. Results are concentrated in Table 5.

Table 5: Position and joint configuration errors.

Command	EE Absolute Position Error	Joint Absolute Position Error
set_angle: 30.0, -25.0, 30.0	2.29, 0.42, 1.04	0.56, 0.22, 0.65
set_angle: 30.0, -40.0, 40.0	5.39, 2.22, 2.83	0.39, 0.51, 0.72
set_angle: 30.0, -55.0, 50.0	5.73, 2.36, 3.86	0.3, 0.63, 0.79

It appears that the encoder of the third joint has relatively low accuracy, with a deviation of nearly 1 degree from the expected joint angle. The cumulative errors from all three joints cause the end effector to deviate from the expected position by almost 6 mm, which exceeds the manufacturer's specified tolerance of 5 mm repetition.

6 Conclusion

In this report, we studied the myCobot 280 Jetson Nano, NVIDIA's collaborative robotic manipulator platform, primarily designed for research and educational purposes. We explored the programming interface of the platform and familiarized ourselves with performing a pick-and-place task. This task was executed in two different ways: the first method involved defining the robot's path from point A to point B, and the second method was kinesthetic teaching, where we physically guided the robot to learn the task and then executed it. Finally, we conducted an accuracy assessment of the robot's movement. We conclude that the robot exhibits a significant error margin, making it unsuitable for tasks that require extreme accuracy and precision. The robot's accuracy is approximately 6 mm, more that the specified by the manufacturer tolerance.

References

[1] Claudio Taesi, Francesco Aggogeri, Nicola Pellegrini, *"COBOT Applications - Recent Advances and Challenges"*, MDPI, Robotics, 2023.

[2] Baris Akgun, Maya Cakmak, Jae Wook Yoo, Andrea Lockerd Thomaz, *"Trajectories and keyframes for kinesthetic teaching: a human-robot interaction perspective"*, ACM, HRI, 2012.

[3] Petar Kormushev, Sylvain Calinon, Darwin G. Galdwell, *"Imitation Learning of Positional and Force Skills Demonstrated via Kinesthetic Teaching and Haptic Input"*, Taylor & Francis, Advanced Robotics, 2011.

[4] Andrius Dzedzickis, Jurga Subaciute-Zemaitiene, Ernestas Sutinys, Urte Samukaite-Bubniene, Vytautas Bucinskas, *"Advanced Applicataions of Industrial Robotics: New Trends and Possibilities"*, MDPI, Applied Sciences, 2021.

[5] NVIDIA Developers, Jetson Nano, online: `https://developer.nvidia.com/embedded/jetson-nano`, (accessed: April 2025).