J-COSINE (Journal of Computer Science and Informatics Engineering)
Vol. 9, No. 2, December 2025 E-ISSN:2541-0806
Accredited Sinta-4 by RISTEKDIKTI Decree No. 79/E/KPT/2023 P-ISSN:2540-8895

Integrated Kinematic-Dynamic Modeling and Ontology-Based Design of a Two-Link Planar Robotic Manipulator

Giri Wahyu Wiriasto[1,2], Syaad Patmanthara[1], Siti Sendari[1]*, Dyah Lestari[1], Muhamad Syamsu Iqbal[2]
[1] Dept. of Electrical and Informatics Engineering, Universitas Negeri Malang
Jl. Semarang 5 Malang 65145 Jawa Timur 62, Malang, Jawa Timur , INDONESIA
[2] Dept. of Electrical Engineering, Universitas Mataram
Jl. Majapahit 62, Mataram, Lombok NTB, INDONESIA
Email: giri.wahyu.2505349@students.um.ac.id, syaad.ft@um.ac.id, siti.sendari.ft@um.ac.id, dyah.lestari.ft@um.ac.id, msiqbal@unram.ac.id
Corresponding Author

Abstract **This study discusses the design and validation of a planar two-degree-of-freedom (2-DoF) manipulator using the Denavit–Hartenberg (D–H) approach for forward kinematics (FK) and inverse kinematics (IK) formulation. The model is developed integratively between CoppeliaSim simulation and a physical implementation based on Arduino Uno, controlling an MG996R servo at Joint 1 and an SG90 servo at Joint 2. Dynamic analysis results show that the maximum gravitational torques of 0.039 N·m and 0.0098 N·m are well below the servos' stall torque capacities (≈ 1.0 N·m and 0.18 N·m), providing safety margins of 25.6× and 18.3× respectively. Kinematic validation tests yield an RMSE-XY of ≈ 2.9 mm with a maximum error of ≈ 6.0 mm, confirming the system's precision within experimental tolerance limits. The main contribution of this research lies in the application of an ontology-driven robotic design approach that unifies physical elements, kinematic-dynamic parameters, and configuration knowledge into an ontologycal semantic framework that is traceable and replicable for educational robot development and advanced research.**

Keywords**: Planar arm robotic, two-link manipulator, kinematic-dynamic, low DoF, ontology form.**

I. Introduction

Robotic manipulators continue to play a vital role in both industrial tasks and academic environments, where these devices serve as platforms for understanding the principles of serial robot motion, control, and modeling [1], [4], [5]. Low-cost, low-degree-of-freedom (low-DoF) manipulators in planar configurations remain in demand for educational and early-stage research applications due to their simple structure while still representing the fundamental characteristics of robotic kinematic chains [1], [6], [7]. In line with this growing need, ontology in the robotics domain plays an important role in defining and classifying knowledge objects such as link–joint structures, physical parameters, and inter-component relationships, thereby enabling systematic and well-documented manipulator design and modeling [18], [19]. The advancement of hardware and open-source simulation platforms has also facilitated the development of robotic arm prototypes that are more accessible for academic learning and for verifying robotics theories [4], [11]. Several studies have been conducted on the design and dynamic analysis of serial manipulators to enhance structural performance and control effectiveness [2], [5], [8]. Moreover, the planar two-link manipulator is widely used as a standard model in studies of trajectory planning, joint actuation behavior, and comparisons between analytical and numerical motion solutions [7], [10]. These studies confirm that simple planar configurations are sufficient to validate core robotic behavior while reducing prototyping costs and computational complexity.

However, most previous research has focused separately on kinematic modeling, dynamic modeling, or simulation-based evaluation [6], [7], [10], [11], without providing explicit and reusable knowledge documentation for the design process. In addition, unstructured knowledge representation hinders the traceability of design decisions and the formal management of dependencies among physical parameters and mathematical models [3], [12]. Therefore, the integration of ontology in the design phase of low-DoF planar manipulators becomes essential to unify physical and semantic modeling within a consistent and extensible design framework. Based on these considerations, this study proposes the design of a low-DoF planar robotic manipulator based on integrated kinematic–dynamic modeling and systematic ontological knowledge representation through a reuse-oriented development approach. The contributions of this study include: (1) the formulation of forward and inverse kinematics for motion mapping; (2) the derivation of dynamic model components including inertia, Coriolis, and gravity based on the physical characteristics of the links; and (3) the development of an ontology structure to support documentation, reasoning, and knowledge reuse throughout the robot design cycle. This integration is expected to strengthen the engineering foundation for the development of low-DoF planar robots and provide a structured basis for further research in manipulator design and control.

Method

2.1. Literature Review

The development of low-DoF planar manipulators continues to advance within the fields of educational robotics and low-cost prototype research. Studies in [1], [4], and [6] indicate that the use of open-source platforms and virtual simulation environments supports fundamental robotics learning, particularly in understanding the relationship between physical structural design and manipulator motion behavior. With a simple yet representative configuration of serial robotic kinematic chains, low-DoF planar manipulators serve as effective tools for mastering kinematic and dynamic concepts before transitioning to more complex robotic systems. This approach is increasingly relevant as it helps reduce prototyping costs while offering flexibility for students and novice researchers to iteratively explore robot design.

Kinematic and dynamic modeling constitute fundamental aspects in the design of low-DoF planar manipulators, as they determine the system's ability to produce accurate and efficient motion. Research in [2], [7], [8], and [10] investigates the use of two-link planar manipulators as standard models for understanding end-effector position mapping, joint actuation analysis, and system response to external forces. The integration between kinematic and dynamic models forms the core foundation for designing manipulators that are both effective and efficient, while reflecting the inherent characteristics of serial robots.

The two-link planar manipulator is a fundamental manipulator configuration consisting of two arms connected by two rotary joints, moving entirely within a single planar workspace. This configuration allows the end-effector to reach various positions in a two-dimensional space by adjusting the angle of each joint. Despite its simplicity, the model still represents key characteristics of serial robots, such as the geometric relationship between joints and the end-effector, as well as the torque requirements needed to generate motion. Therefore, the two-link planar manipulator is the most widely used model in studies of kinematic and dynamic formulations, and serves as the foundational experimental prototype for low-cost planar manipulator design.

2.2. Ontology in Knowledge Representation for Manipulator Design

Knowledge representation plays a crucial role in ensuring information consistency and design continuity in the development of low-DoF planar manipulators. However, most previous studies presented modeling and design processes separately, without explicit and structured knowledge documentation [3], [11], [12]. The absence of a formal approach leads to limitations in design decision traceability, difficulties in model validation, and a low degree of knowledge reuse in subsequent development cycles.

An ontology-based approach offers a solution to unify technical information into a cohesive knowledge structure, organized through clear relationships that can be processed and utilized by computational systems. Studies in [18] and [19] have shown that ontology can classify essential objects and parameters in the robotics domain—such as links, joints, physical properties, actuation limits, and kinematic–dynamic relationships—into a standardized semantic framework. Through such modeling, the design process can be explicitly documented, facilitating the traceability of technical rationales, strengthening inter-model consistency, and enabling knowledge reuse for future manipulator development.

Furthermore, this study emphasizes the importance of integrating kinematic–dynamic modeling with physical implementation in developing low-DoF planar manipulators. The use of low-cost, open-source-based prototypes enables direct validation of joint actuation performance, trajectory mapping, and end-effector response to motion configuration variations. Hence, the robotic system is not only analyzed in simulation environments but also experimentally tested to ensure the model's conformity with real-world dynamic behavior [23–25].

Overall, research on low-DoF planar manipulators has addressed physical design, kinematic–dynamic modeling, and trajectory and prototype evaluation. However, there remains a gap in integrating physics modeling with ontology-based knowledge representation, resulting in design documentation that is not fully consistent and difficult to reuse in subsequent robot development iterations.

2.3. Ontological Design Representation: Objects and Relations (Properties) in Ontology

In the developed ontology, each entity of the low-DoF planar manipulator is interconnected through object properties designed to describe the structural and functional interactions among components. There are formal relationships linking one entity to another within the robotics domain. The ontology defines two main elements: entities and relations. The classification of relations in the planar robotic design ontology is presented in Table I.

TABLE I. Classification of Entities and Relations in the Low-DoF Planar Manipulator Ontology

Component	Part	Meaning
Class (Entitas/Ontology Concept)	Link, Joint, Actuator, End-effector, Frame	Modeled object
Object/Data Properties (Relation/atribute)	connectedTo, actuatedBy, hasMass	Relationships between Objects or Object Attributes

Table I presents the classification of the main components in the ontology used in this study, namely classes as representations of physical entities such as links, joints, actuators, end-effectors, and frames, as well as object/data properties as relations or attributes that describe the interactions and characteristics of each entity. These relationships form an essential semantic foundation to ensure that design information can be stored, traced, and reused throughout the robot development cycle. In constructing this ontology, the naming of each object property follows the standard pattern recommended by the Semantic Web (W3C Standards), namely the Verb + Noun or Noun + Preposition structure, which enables explicit representation of relations and actions. This pattern is designed so that the represented knowledge is not only human-readable but also machine-interpretable through automated reasoning mechanisms. For example, the property *connectedTo* describes the physical connection between robot components, *actuatedBy* specifies the actuator responsible for joint motion, *while hasMass* and *hasDHParameter* indicate the possession of a particular physical characteristic or parameter by an entity. By following this naming convention, each relation in the ontology becomes more consistent, easily traceable within the knowledge model, and supports *interoperability* and knowledge reuse in robotics systems and other semantically based engineering platforms. Table II presents the main relations applied in this ontology, along with their meanings and examples of implementation in the two-link planar manipulator model.

TABLE II. OBJECT PROPERTIES IN THE LOW-DOF PLANAR MANIPULATOR ONTOLOGY

Object Property	**Domain (Subject)**	**Range (Object)**	**Relation Description**	**Application in the Two-Link Planar Manipulator**
connectedTo	Link / Joint	Link / Joint	Represents the physical connection between robot components.	Link 1 connectedTo Joint 2
actuatedBy	Joint	Actuator	Describes the actuator responsible for driving a specific joint.	Joint 1 actuatedBy MG996R
hasMass	Link	Mass value (Data literal)	Associates the physical mass property with a corresponding link.	Link 2 hasMass 0.02 kg
hasLength	Link	Length Value (Data Literal)	Defines the geometric length of each link.	Link 1 hasLength 10 cm
hasDHParameter	Joint	DH Parameter Set	Links each joint to the corresponding DH parameters used in the kinematic model.	Joint 2 hasDHParameter ($\theta 2$, d2, a2, $\alpha 2$) *)
hasCoordinateFrame	Link / Joint	Frame	Assigns the coordinate reference frame for a specific component in the manipulator structure.	Link 1 hasCoordinateFrame Frame1
supports	Base	Joint	Specifies that the base serves as the initial support or anchor point of the manipulator.	Base supports Joint 1
affectsTorque	Link	Joint / Actuator	Indicates the contribution of link mass and length to the actuator load requirements.	Link 2 affectsTorque Joint 1
producesMotion	Actuator	Joint	Describes the primary function of each actuator in generating joint motion.	MG996R produces Motion Joint 1

*) The D–H parameters used include θ_i as the rotation angle of the *i*-th joint, d_i as the offset along the z_i axis, a_i as the length of the *i*-th link, and α_i as the angle between the axes $z_{(i-1)}$ and z_i.

Table II describes the main relations (object properties) used in the ontology of the low-DoF planar manipulator design. Relations with semantic web terms such as *connectedTo* and *actuatedBy* illustrate the physical and actuation connections between robot components—for example, a link connected to a joint or a joint driven by a specific actuator. Meanwhile, properties such as *hasMass*, *hasLength*, and *hasDHParameter* represent the physical and kinematic attributes inherent to manipulation entities—for instance, the mass of a link or the Denavit–Hartenberg parameters of a joint. Other properties, such as *supports*, *producesMotion*, and *affectsTorque*, encompass functional aspects and the design's influence on system performance—for example, a base supporting a joint, an actuator generating motion, or a link affecting joint torque.

2.4. Kinematic Modeling

The two-link planar manipulator used in this study is modeled as a two-revolute-joint (revolute–revolute) system that moves entirely within the XY plane. The physical structure of the manipulator is represented as shown in Figure 1. The motion of the end-effector is generated by changes in the joint angles θ_1 and θ_2, while the link lengths are defined as L_1 = 10 cm and L_2 = 10 cm, respectively. With this planar configuration, every position of the end-effector lies within the same plane as the entire kinematic chain. The kinematic modeling of the low-DoF planar manipulator is carried out sequentially by defining the Denavit–Hartenberg (D–H) parameters as the basis for representing transformations between links. These D–H parameters are then used to construct the forward kinematics equations, which calculate the position and orientation of the end-effector based on the given joint angles.

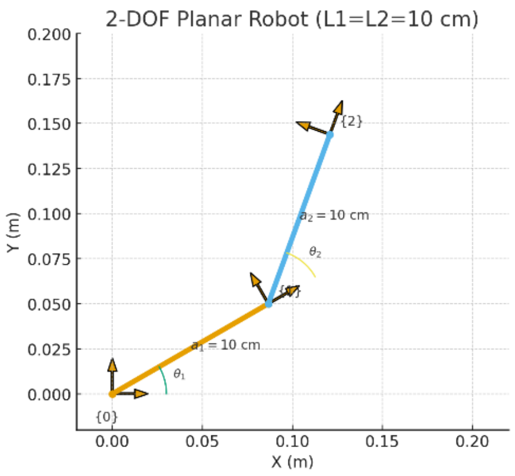

Fig 1. Design kinematic tracking of two-link (2 DoF) manipulator robot

Conversely, inverse kinematics utilizes the same kinematic structure to determine the joint angle values required for the end-effector to reach a specific target position. The DH (Denavit-Hartenberg) parameters provide a standardized coordinate framework, with forward kinematics offering a mapping from joint space to workspace, while inverse kinematics functions as its reverse process in manipulator motion control. The DH parameters are not used directly to solve inverse kinematics; rather, they serve as a standardized geometric representation that defines the coordinate transformations between links in the manipulator system.

2.5. *Denavit–Hartenberg* (DH) Parameters

TABLE III. DH MODEL MANIPULATOR OF 2-DOF

i	θ_i (variable)	d_i (cm)	a_i (cm)	α_i (deg)
1	θ_1	0	L_1	0
2	θ_2	0	L_2	0

2.6. Forward Kinematics

Posisi end-effector (x, y) masing-masing dihitung menggunakan persamaan (1) dan (2) from [7], [37]:

$$x = L_1 \mathrm{Cos}\theta_1 + L_2 \mathrm{Cos}(\theta_1+\theta_2) \quad (1)$$

$$y = L_1 \mathrm{Sin}\theta_1 + L_2 \mathrm{Sin}(\theta_1+\theta_2) \quad (2)$$

This describes the end-effector position (x, y) in terms of the joint angles θ_1 and θ_2, with link lengths L_1 and L_2.
This is (3) the Denavit-Hartenberg (DH) transformation matrix that transforms coordinates from frame {i} to frame {i-1} from [37]:

$$^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i & 0 & a_i\sin\theta_i \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [3]$$

θ_i: Joint angle (revolute joint); d_i: Link offset (prismatic joint); a_i: Link length; α_i: Link twist (here $\alpha_i = 0$, so it's simplified)

2.7. Inverse Kinematics (IK)

This is the inverse kinematics solution for a two-link planar robotic arm, which calculates the joint angles (θ_1, θ_2) needed to achieve a desired end-effector position (x, y) using equation (4-6) from [37];

$$\cos\theta_2 = \frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2} \quad (4)$$

$$\theta_2 = \mathrm{atan2}\left(\pm\sqrt{1-\cos^2\theta_2}, \cos\theta_2\right) \quad (5)$$

$$\theta_1 = \mathrm{atan2}(y, x) - \mathrm{atan2}\left(L_2\sin\theta_2, L_1 + L_2\cos\theta_2\right) \quad (6)$$

Based on the DH model, the forward kinematics equations are derived to map the joint angles to the end-effector position. These equations then serve as the basis for the formulation of inverse kinematics, which determines the joint angles required to reach a target position. Thus, the DH model ensures that both forward and inverse kinematics modeling are carried out within a consistent coordinate framework.

2.8. *Re-used oriented* diagram

Conceptually, this system integrates the processes of modeling, computation, and validation into a single systematic workflow, starting from requirement specification to system performance evaluation.

The main structure of the system consists of several sequential stages that reflect a modular software engineering approach. The first stage, requirement specification, defines the system requirements such as the type of robot model (2-DoF), output parameters (joint angles θ_1 and θ_2), as well as the platform and programming language used. Next, the component analysis stage identifies and organizes essential components such as Python libraries, elements within the CoppeliaSim scene, and Python scripts containing inverse kinematics (IK) functions. The requirement modification stage incorporates safety and reliability elements, including arm length adjustment, target position constraints, and real-time iterative computation to ensure motion stability and system safety.

Figure 2 illustrates the conceptual framework of the simulation system development for the robotic arm based on a reuse-oriented model, which encompasses system design, integration, and validation through the utilization of reusable components. The system is developed using the CoppeliaSim simulation platform [39] and Python programming language as the external control environment. This approach enables bidirectional interaction between Python and CoppeliaSim to control robot motion and visually display the kinematic computation results within a 3D simulation environment.

The final stage, system design, focuses on program architecture, communication integration, and performance validation. Communication bidirectional interaction is carried out through the ZMQ Remote API, which enables direct and synchronous data exchange. The system is designed so that each process—from connection initialization, target position reading, and inverse kinematics (IK) computation to joint position configuration—runs in an integrated manner and can be tested through visual simulation results.

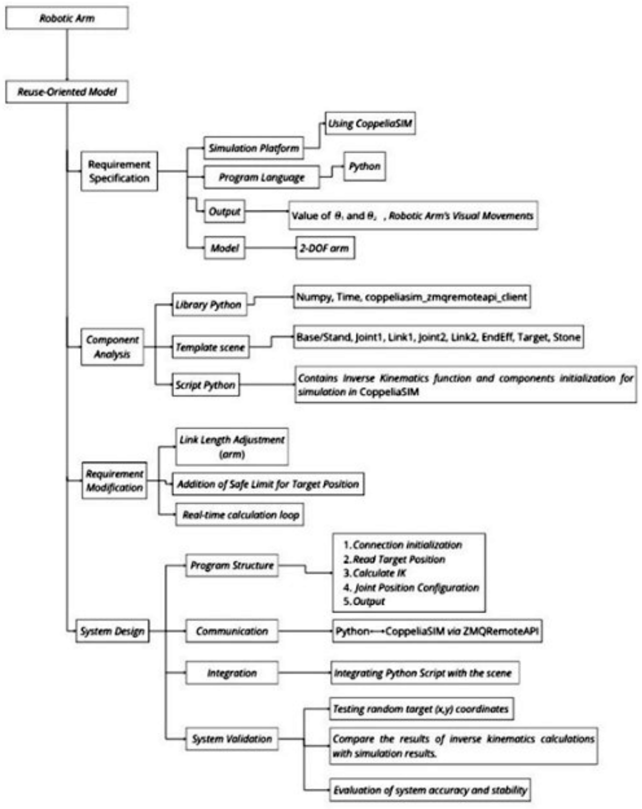
Robotic Arm
Reuse-Oriented Model
Requirement Specification
Simulation Platform
Using CoppeliaSIM
Program Language
Python
Output
Value of θ₁ and θ₂ , Robotic Arm's Visual Movements
Model
2-DOF arm
Component Analysis
Library Python
Numpy, Time, coppeliasim_zmqremoteapi_client
Template scene
Base/Stand, Joint1, Link1, Joint2, Link2, EndEff, Target, Stone
Script Python
Contains Inverse Kinematics function and components initialization for simulation in CoppeliaSIM
Requirement Modification
Link Length Adjustment (arm)
Addition of Safe Limit for Target Position
Real-time calculation loop
System Design
Program Structure
1. Connection initialization
2. Read Target Position
3. Calculate IK
4. Joint Position Configuration
5. Output
Communication
Python⟷CoppeliaSIM via ZMQRemoteAPI
Integration
Integrating Python Script with the scene
System Validation
Testing random target (x,y) coordinates
Compare the results of inverse kinematics calculations with simulation results.
Evaluation of system accuracy and stability

Fig. 2. Re-Use Oriented System Development

2.7. *Dynamic Modeling*

Dynamic modeling for the two-link planar manipulator is a fundamental step in designing an effective robotic control system [2], [7]. The Lagrangian approach is employed to analyze the system dynamics by considering the kinetic and potential energy contained within the manipulator structure [8]. This method defines the Lagrangian as the difference between the total kinetic energy and the total potential energy of the system, which is then applied in the Euler–Lagrange equations to derive the equations of motion for the manipulator [10].

The total kinetic energy of the system is obtained by summing the contributions of each link, consisting of translational and rotational components. Each link contributes kinetic energy that depends on its mass, the linear velocity of its center of mass, moment of inertia, and angular velocity [8]. Meanwhile, the potential energy is determined based on the vertical position of each link's center of mass relative to the gravitational reference [7].

The final formulation of the manipulator dynamics is expressed as a system of matrix equations consisting of three main components: an inertia matrix that depends on joint configuration and describes the mass distribution of the system; Coriolis and centrifugal effects arising from rotational motion and dependent on joint velocities; and a gravity vector representing the effect of gravitational forces for a given configuration [2], [10]. This dynamic model enables the calculation of the actuation torque required to produce the desired motion, thus serving as a critical foundation for control design, trajectory planning, system performance analysis, and energy optimization in robotic manipulators [7], [10]. Equation (7) is Lagrangian formula from [37]:

$$L(q,q^{\cdot}) = K(q,q^{\cdot}) - P(q) \tag{7}$$

Where: L: Lagrangian; K: Total kinetic energy; P: Total potential energy; $q = [\theta_1, \theta_2]^T$: Joint configuration vector.

Equation (8) is manipulator's equation of motion [37] is then obtained as follows:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = \tau_i \tag{8}$$

Following provides an explanation of the variables in the Lagrange equation used for dynamic modeling of the manipulator. Equation (8) represents the general form of the Euler–Lagrange equation, which is used to derive the dynamics of a mechanical system. The variables in this equation have specific meanings in the context of robotic manipulator modeling:

Parameter dynamic modeling, L represents the Lagrangian of the system, defined as the difference between the total kinetic energy K and the total potential energy P, expressed as L = K – P. This quantity reflects the overall energy dynamics of the system. The variable q_i denotes the generalized coordinate, which describes the degrees of freedom of the system; in a two-link manipulator, $q_1 = \theta_1$ and $q_2 = \theta_2$ represent the angles of the first and second joints, respectively. Its time derivative, $\dot{q}_i = (dq_i/dt)$, indicates the generalized velocity, or the angular speed occurring at each joint. Meanwhile, τ_i is the generalized force acting on the coordinate q_i, which, in the context of robotics, represents the actuation torque that must be applied to the *i*-th joint to generate the desired motion.

The term $\partial L/\partial \dot{q}_i$ represents the generalized momentum associated with coordinate q_i and, in the manipulator model, corresponds to the angular momentum of the link. The time derivative of this term, $d/dt(\partial L/\partial \dot{q}_i)$, expresses the rate of change of momentum, consistent with Newton's second law in its Lagrangian formulation. Meanwhile, $\partial L/\partial q_i$ represents the influence of potential energy on motion dynamics, including gravitational effects and system configuration constraints. Overall, these parameters play a crucial role in constructing accurate equations of motion for the manipulator and form the foundation of many modern robotic control strategies. The resulting equation yields a system of differential equations that fully describe the manipulator's dynamics, encompassing inertial, Coriolis, centrifugal, and gravitational effects. By solving these equations for all generalized coordinates, a complete dynamic model can be obtained, which is essential for designing effective robotic control systems.
This (9) leads to the standard dynamic equation from [38]:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \tag{9}$$

The parameters in the dynamic model of the two-joint manipulator are represented using the standard Euler–Lagrange formulation. The inertia matrix $M(q)$ is a symmetric and positive-definite matrix that depends on the joint configuration $q$and represents the distribution of mass and inertia of the system throughout its motion. The joint position vector is denoted as $q = [\theta_1, \theta_2]^T$, where θ_1 and θ_2 are the rotation angles of joints 1 and 2, respectively. The first time derivative yields the joint velocity vector $\dot{q} = [\dot{\theta}_1, \dot{\theta}_2]^T$, while the second derivative produces the joint acceleration vector $\ddot{q} = [\ddot{\theta}_1, \ddot{\theta}_2]^T$.

Another influential dynamic component is the Coriolis and centrifugal matrix $C(q,\dot{q})$, which models the force effects resulting from rotational motion and depends on both joint positions and velocities. Furthermore, the gravity vector $G(q)$ represents the torques required to counteract the gravitational forces acting on each joint configuration. All these forces and moments must be balanced by the actuation torque $\tau = [\tau_1, \tau_2]^T$, which corresponds to the motor outputs at each joint, enabling the manipulator to achieve the desired trajectory.

Equation (10) provides the formulation for computing the inertia matrix of the two-link planar manipulator [37].

$$M(q) = \begin{bmatrix} m_1 r_1^2 + m_2(l_1^2 + r_2^2 + 2l_1 r_2 \cos\theta_2) + I_1 + I_2 & m_2(r_2^2 + l_1 r_2 \cos\theta_2) + I_2 \\ m_2(r_2^2 + l_1 r_2 \cos\theta_2) + I_2 & m_2 r_2^2 + I_2 \end{bmatrix} \tag{10}$$

Where: m_1 and m_2 are the masses of links 1 and 2; r_1 and r_2 are the distances from each link's center of mass to its axis of rotation; l_1 is the length of link 1; I_1 and I_2 are the moments of inertia of the links with respect to their centers of mass; and θ_2 is the relative angle between link-1 and link-2. The energy-based approach using the Lagrangian formulation also supports dynamic modeling by expressing the system's total energy as the combination of kinetic and potential energies, namely $K = K_1 + K_2$ and $P = P_1 + P_2$. The kinetic energy of each link K_i is the sum of its translational energy $\frac{1}{2} m_i v_{ci}^2$—influenced by the link mass $m_i$and the linear velocity of its center of mass v_{ci}—and its rotational energy $\frac{1}{2} I_{ci} \omega_i^2$, which depends on the link's moment of inertia about its center of mass I_{ci} and its angular velocity ω_i. For planar systems, the angular velocity vector is typically aligned with the z-axis. Thus, each variable in this formulation has a clear physical contribution to the system's dynamics, ensuring that the resulting model accurately represents the manipulator's motion characteristics and serves as a solid foundation for effective control system design.

Importance in Dynamic Modeling, Lagrangian formulation, where the system's total kinetic energy K is the sum of all link energies K_i. Understanding these kinetic energy components is essential for computing the inertia matrix $M(q)$ and the Coriolis and centrifugal effects $C(q,\dot{q})$ in the complete dynamic model of the robot.

Equation (11) is used to calculate the potential energy of each link, depending on the position of its center of mass relative to the gravitational reference.

$$P_i = m_i g h_i \tag{11}$$

The potential energy of a two-link planar manipulator is formulated as $P = P_1 + P_2$, where P_i represents the potential energy of the i-th link, stored as a result of the link's position in the gravitational field. The magnitude of this potential energy depends on the link mass m_i, which is the inertial quantity indicating the amount of matter contained in the link and directly influences the amount of energy stored at a certain height, as well as on the gravitational acceleration g, a physical constant with an approximate value of 9.8 m/s² on the Earth's surface, defining the strength of the gravitational field acting on the system.

In addition, P_i is determined by h_i, the height of the i-th link's center of mass relative to the datum or zero reference point, which depends on the joint configuration q. In a two-link planar manipulator, the value of $h_i$is a function of the joint rotation angles θ_1 and θ_2; therefore, any change in the robot's configuration alters the vertical position of the center of mass and consequently affects the system's total gravitational potential energy. Thus, modeling potential energy plays a crucial role in the calculation of the gravitational force vector $G(q)$ within the manipulator's dynamic formulation and becomes an integral part of the Lagrangian-based analysis. Importance in dynamic modeling, Equation (12) is a fundamental component of the Lagrangian formulation. The system's total potential energy P is the sum of all individual link energies P_i. The partial derivative of potential energy with respect to the generalized coordinates yields the gravitational torque vector $G(q)$ in the dynamic model.

$$G(q) = \frac{\partial P}{\partial q} \tag{12}$$

Where $G(q)$ represents the torque that must be sustained by the actuators due to gravitational effects at a given configuration. Understanding this potential energy is essential for designing gravity-compensating controllers and for analyzing the stability of robotic systems.

The matrix-form dynamic model for the two-link planar manipulator describes the relationship between forces, accelerations, and torques within the system through the Euler–Lagrange formulation. The dynamic equation (13) is expressed as [38]:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \tag{13}$$

Where: $M(q)$ is the inertia matrix that depends on the joint configuration; $C(q,\dot{q})\dot{q}$ represents the Coriolis and centrifugal effects; $G(q)$ denotes the gravitational forces; and τ is the input torque from the actuators, with q being the configuration vector (14) [38].

$$q = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \tag{14}$$

RESULT AND DISCUSION

This part presents the implementation results, analysis, and validation of the two-link planar manipulator system designed based on the kinematic–dynamic modeling approach and the ontology-driven design framework. The analysis covers three main aspects: the torque feasibility at each joint, the workspace reachability, and the end-effector trajectory tracking performance, evaluated using the Denavit–Hartenberg (DH) parameters through both CoppeliaSim simulations and physical implementation.

3.1. Inertia and Mass Parameters in the Two-Link Planar Robot Design

The mass and moment of inertia values are determined based on the physical model of the designed links:

TABLE IV. INERTIA AND MASS PARAMETERS IN THE DESIGN

Parameter	Link-1	Link-2
Mass (m)	0.02 kg	0.02 kg
Lenght (L)	0.10 m	0.10 m
Inertia (I_{ZZ})	(3.333×10^{-5})	(3.333×10^{-5})

Table IV is organized to summarize the main physical parameters used in the dynamic modeling of the two-link manipulator. These parameters include the mass (m), link length (L), and moment of inertia about the rotational axis (I_{xx} or I_{zz}), each of which is utilized in calculating the kinetic energy, potential energy, and in constructing the inertia matrix M(q) within the Euler–Lagrange formulation. The values reflect the actual specifications of the physical prototype made of acrylic material, ensuring that the simulation and dynamic analysis realistically represent the system's behavior. The inertia values are calculated using a thin, homogeneous beam model for planar links. The resulting dynamic equations also allow for the analysis of the servo motor's torque capability.

TABLE V. JOINT / ACTUATOR PROFILE USED IN DESIGN AND PHYSICAL IMPLEMENTATION

Joint	Aktuator	Torsi Stall	Evaluation Status
Joint 1	MG996R	9.4–11 kg·cm	Capable
Joint 2	SG90	1.8 kg·cm	Capable, with limitations

Table V presents the evaluation results of the actuation capability for each manipulator joint based on the stall torque of the servos used. Joint 1 employs an MG996R servo with a stall torque of 9.4–11 kg·cm, which is capable of supporting the load and moving the entire manipulator structure stably. Meanwhile, Joint 2 uses an SG90 servo with a stall torque of approximately 1.8 kg·cm, which can also perform its function of driving the second link, although with certain torque limitations due to its smaller size and motor capacity.

3.2. Implementation Environment and Experimental Scenarios

The performance evaluation of the low-DoF planar manipulator was carried out through the implementation of a simulation based on the kinematic–dynamic model formulated in the methodology section. The testing utilized the CoppeliaSim simulation platform as the visualization environment for robot motion, with Python serving as the analytical computation medium for calculating kinematics, dynamics, and actuation torque. These processes were executed via the ZMQ Remote API, enabling direct and synchronous data exchange between both systems. The manipulator configuration consisted of two homogeneous links, each with a length of 0.10 m and a mass of 0.02 kg, corresponding to the physical parameters of the designed prototype. The inertia, mass, and DH parameters of each link were integrated into the simulation model to produce a motion representation closely resembling real physical conditions. The joint motion ranges were defined according to the operational limits of the actuators, namely $\theta_1 \in [-90°, 90°]$and $\theta_2 \in [-90°, 90°]$, allowing comprehensive observation of the end-effector workspace. Both actuators—MG996R on Joint 1 and SG90 on Joint 2—were employed in both forward and inverse kinematics computations, as well as in torque requirement evaluations based on the dynamic model.

Figure 3 illustrates the visual representation of the 2-DoF or two-link planar manipulator model used in the CoppeliaSim simulation environment. This design was inspired by the design developed in [36] with several modifications applied to the scale and structural configuration of the joint–link connections. The manipulator structure consists of two main segments, Link 1 and Link 2, each measuring 0.1 meters in length.

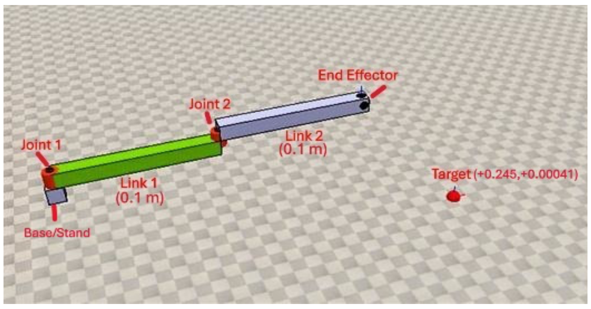

Figure 3: 2DoF Planar Design

These two links are connected by two rotational joints—Joint 1 and Joint 2 which enable rotational motion within a two-dimensional (planar) workspace. At the end of the system lies the End Effector, the terminal point of the robotic arm responsible for reaching the target position. The target point is visualized as a red sphere located at coordinates (x, y) = (0.245, –0.00041), serving as the goal position for the End Effector during the Inverse Kinematics computation process.

TABLE VI. DH PARAMETERS OF THE 2-DOF PLANAR MANIPULATOR

i	θ_i (variable)	d_i (cm)	a_i (cm)	α_i (deg)
1	θ_1	0	10	0
2	θ_2	0	10	0

In Table VI, two rows represent the two rotational joints. For the first joint ($i = 1$), the rotation variable is θ_1, with a translation distance of $d_1 = 0$ cm, link length $a_1 = 10$ cm, and an inter-axis angle $\alpha_1 = 0°$, indicating that the first link rotates within the XY plane without any rotation between coordinate planes. Similarly, for the second joint ($i = 2$), the parameters are θ_2, with $d_2 = 0$ cm, $a_2 = 10$ cm, and $\alpha_2 = 0°$, showing that both links lie on the same (planar) surface and have parallel rotational axes.

Thus, this table serves as a formal representation of the geometric structure of the 2-DoF planar manipulator, which is used in computing the homogeneous transformation matrix (T_{02}) to determine the position and orientation of the end effector based on the rotation angles θ_1 and θ_2. These DH parameters also foundation for the analysis of both Forward Kinematics and Inverse Kinematics, applied in theoretical modeling as well as simulation.

3.3. Testing and Validation of Forward Kinematics (FK)-DH

The testing was conducted to ensure that the formulated kinematic model accurately represents the geometric relationship between joint angles and the end-effector position. Validation was performed through simulations in CoppeliaSim, comparing the computed end-effector positions obtained from the Forward Kinematics (FK) model with the actual positions generated in the simulation environment. The implementation and validation process followed the workflow below, summarized in pseudocode form:

```
BEGIN
  # --- Initialization ---
  Connect to CoppeliaSim via ZMQ Remote API
  Get object handles:
    joint1 ← getObjectHandle("Joint1")
    joint2 ← getObjectHandle("Joint2")
    endEffector ← getObjectHandle("EndEffector")

  # --- Define physical parameters ---
  a1 = 0.10   # length of link 1 (meters)
  a2 = 0.10   # length of link 2 (meters)
  d1 = 0      # translation along z (DH parameter)
  d2 = 0
  α1 = 0      # twist angles (radians)
  α2 = 0

  # --- Define test joint angles (in radians) ---
  θ1_test = [0, 15°, 30°, 45°, 60°, 75°, 90°]
  θ2_test = [0, 15°, 30°, 45°, 60°, 75°, 90°]

  FOR each pair (θ1, θ2) in test set DO
    # --- Compute Forward Kinematics using DH ---
    T01 = dh_transform(θ1, d1, a1, α1)
    T12 = dh_transform(θ2, d2, a2, α2)
    T02 = T01 × T12

    # Extract end-effector position
    x_FK = T02[0, 3]
    y_FK = T02[1, 3]

    # --- Send joint configuration to simulation ---
    sim.setJointPosition(joint1, θ1)
    sim.setJointPosition(joint2, θ2)
    Wait for simulation update

    # --- Read actual position from CoppeliaSim ---
    [x_sim, y_sim, z_sim] = sim.getObjectPosition(endEffector)

    # --- Compute validation error ---
    RMSE_XY = sqrt((x_FK - x_sim)^2 + (y_FK - y_sim)^2)
    Store results (θ1, θ2, x_FK, y_FK, x_sim, y_sim, RMSE_XY)
  END FOR

  # --- Analysis ---
  Compute mean and maximum RMSE_XY
  Display FK accuracy and validation plots
END
```

Figure 4 shows the plotted results of the end-effector workspace evaluation from the FK simulation based on the DH parameters. The solid blue curve represents the target trajectory calculated from the commanded joint angles (FK-command), while the orange dashed line represents the actual trajectory obtained from the joint position readings in the simulator (FK-actual). Both trajectories exhibit a very high level of agreement, with a maximum deviation of only about 1.5–2.0 mm near the points of directional change. This indicates that the kinematic model and the DH parameters used are well-calibrated with respect to the physical geometry of the manipulator.

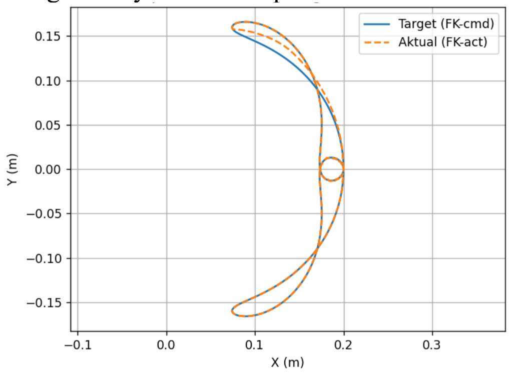

Fig. 4. End-Effector Workspace Evaluation Results from DH-Based FK Simulation

Figure 5 displays the joint angle profiles over time during a 10-second simulation of the two-link planar manipulator based on the Denavit-Hartenberg (D-H) model. The blue curve represents the first joint angle, $\theta_1(t)$, while the orange curve represents the second joint angle, $\theta_2(t)$.

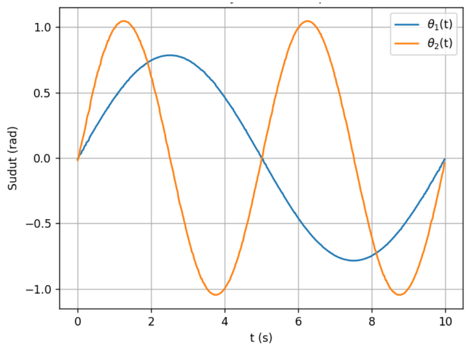

Fig. 5. Joint Angle Profile over Time

The sinusoidal waveforms in the graph indicate that both joints move synchronously and periodically, with amplitudes of approximately ±1.0 rad (≈ ±57°) for the second joint and ±0.8 rad (≈ ±45°) for the first joint. The phase difference between the two motions results in a complex trajectory within the end-effector's workspace. This pattern demonstrates that the control system can maintain stable tracking without excessive oscillation or overshoot. These results validate that the D-H parameters and simulation control loop were properly configured.

Figure 6 shows the results of the end-effector position tracking error evaluation over a 10-second simulation. The curve illustrates the magnitude of the error $e(t)$or | $e(t)$ |as a function of time t, representing the difference between the target trajectory from the kinematic command and the actual trajectory. At the beginning of the simulation, a peak error of approximately 0.16 m is observed during the first second, caused by the system's transient response as the actuators start moving from rest (initial transient). After this initial phase, the error decreases significantly and stabilizes below 0.005 m, with an average Mean Error ≈ 0.0028 m and a Root Mean Square Error (RMSE) ≈ 0.0031 m.

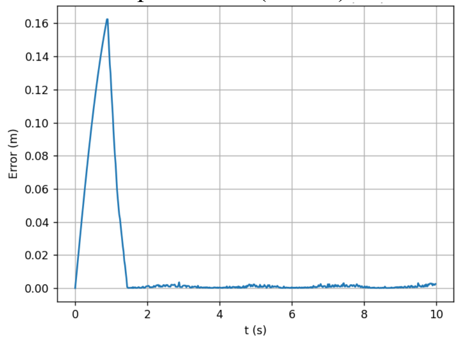

Fig. 6. End-Effector Position Tracking Error Evaluation

3.4. Testing and Validation of Inverse Kinematics (IK) – DH.

The testing and validation of Inverse Kinematics (IK) on the two-link planar manipulator using the Denavit–Hartenberg (DH) parameterization applied in the design and simulation system. The IK algorithm computes the joint angles (θ_1, θ_2)based on the target end-effector position specified within the workspace. Validation was performed by comparing the computed joint angles from the IK model with the actual manipulator response and verifying the consistency of the end-effector motion trajectory with the desired target path. Below is the validated IK pseudocode used in the experiment.

```
BEGIN
  # --- Initialization ---
  Connect to CoppeliaSim via ZMQ Remote API
  Get object handles:
    joint1 ← getObjectHandle("Joint1")
    joint2 ← getObjectHandle("Joint2")
    endEffector ← getObjectHandle("EndEffector")

  # --- Define manipulator parameters ---
  a1 = 0.10   # length of link 1 (meters)
  a2 = 0.10   # length of link 2 (meters)

  # --- Define target positions in workspace ---
  target_positions = [
    (0.15, 0.10),
    (0.18, 0.05),
    (0.20, 0.00),
    (0.17, -0.08),
    (0.12, -0.10)
  ]

  FOR each (x_target, y_target) in target_positions DO
    # --- Inverse Kinematics Calculation ---
    # Compute θ2 from law of cosines
    cosθ2 = (x_target^2 + y_target^2 - a1^2 - a2^2) / (2 * a1 * a2)
    sinθ2 = sqrt(1 - cosθ2^2)
    θ2 = atan2(sinθ2, cosθ2)

    # Compute θ1 from geometric relations
    k1 = a1 + a2 * cosθ2
    k2 = a2 * sinθ2
    θ1 = atan2(y_target, x_target) - atan2(k2, k1)

    # --- Send computed angles to simulation ---
    sim.setJointPosition(joint1, θ1)
    sim.setJointPosition(joint2, θ2)
    Wait for simulation update

    # --- Retrieve actual end-effector position ---
    [x_sim, y_sim, z_sim] = sim.getObjectPosition(endEffector)

    # --- Compute tracking error ---
    RMSE_XY = sqrt((x_target - x_sim)^2 + (y_target - y_sim)^2)
    Store (x_target, y_target, θ1, θ2, x_sim, y_sim, RMSE_XY)
  END FOR

  # --- Performance Validation ---
  Compute mean RMSE_XY across all target points
  Display comparison plots:
    (1) Desired vs Actual End-Effector Positions
    (2) Computed vs Actual Joint Angles
END
```

Figure 7 shows the results of the IK–DH testing and validation on the two-link planar manipulator. The blue curve represents the target end-effector trajectory defined in the workspace as a circular path, while the orange curve depicts the actual trajectory generated by the manipulator in CoppeliaSim after calculating the joint angles (θ_1, θ_2). The graph demonstrates that the actual trajectory successfully follows the target trajectory, indicating accurate inverse kinematics computation and effective motion execution within the simulation environment.

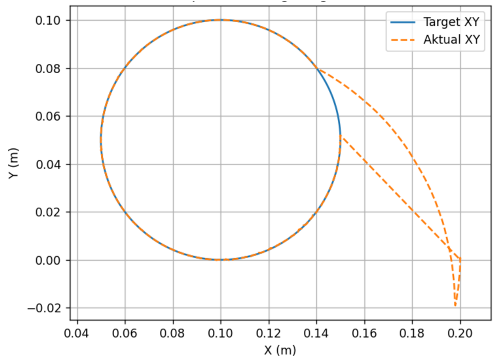

Fig. 7. Workspace tracking: Target vs Actual

Figure 8 shows the joint angle profiles obtained from the IK–DH testing and validation presented in Figure 7. The blue curve illustrates the motion of the first joint, $\theta_1(t)$, while the orange curve represents the motion of the second joint, $\theta_2(t)$, over time during the circular trajectory tracking of the end-effector. The graph demonstrates that both joint angles oscillate smoothly and in a coordinated manner, indicating a consistent kinematic relationship between the target positions in the workspace and the joint angle configurations computed by the IK algorithm. The phase and amplitude variations between θ₁ and θ₂ reflect each joint's contribution in maintaining the end-effector's position along the desired trajectory.

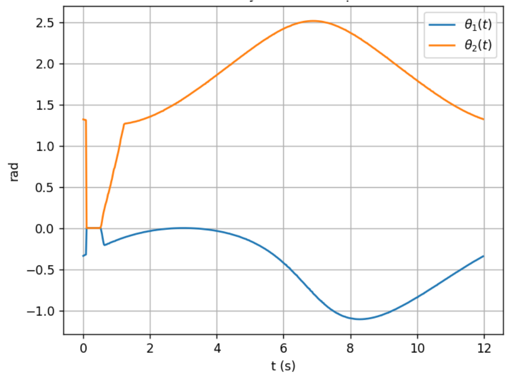

Fig. 8. Joint Angle Profile over Time

Figure 9 shows the end-effector position tracking error curve during the IK–DH testing. The variables in the graph are identical to those described in Figure 6. At the beginning of the simulation, a noticeable error spike of approximately 0.09 m occurs due to the system's transient response as the actuators adjust their initial positions toward the target trajectory. After this initial phase, the error rapidly decreases and stabilizes near 0 m, indicating that the system successfully reaches a steady-state condition.

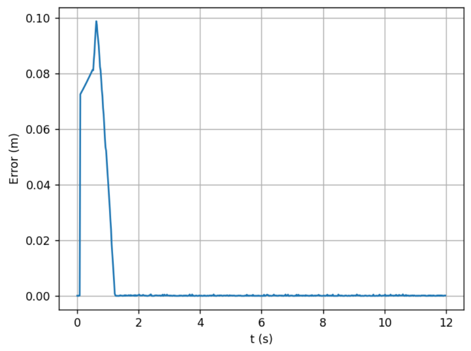

Fig. 9. Error tracking position of End-effector

3.5. Root Mean Square Error (RMSE)

The test results show that the IK–DH kinematic motion exhibits tracking performance comparable in accuracy, with an RMSE-XY of 0.00312 m (≈ 3.12 mm), a maximum error of 0.00645 m (≈ 6.45 mm), and an average error of 0.00288 m (≈ 2.88 mm). These values indicate that the IK model based on the Denavit–Hartenberg formulation successfully maintains end-effector trajectory precision within the actuator system's tolerance limits. The RMSE values between FK–DH and IK–DH are not compared here since they serve different functional purposes. This validation confirms the consistency between the geometric representation and the dynamic response of the developed model.

3.6. Implementation of the Physical Two-Link Manipulator

Figure 10 shows the physical implementation of the 2-DoF planar manipulator, constructed based on the previously developed kinematic and dynamic simulation design. The mechanical structure uses transparent acrylic as the link material due to its light weight, strength, and ease of assembly, with each link measuring approximately 10 cm in length. The system employs two primary actuators—an MG996R servo for Joint 1 and an SG90 micro servo for Joint 2—each selected according to its respective kinematic function.

Joint 1 uses the MG996R servo motor because of its high torque capacity (9–11 kg·cm at 6 V), which is required to support and move the entire arm, including the second link and the end effector. This motor can rotate up to 180° with an average speed of 0.17 s/60°, making it ideal for the manipulator's base joint, where stable and powerful motion is needed.

In contrast, Joint 2 uses the SG90 micro servo, which provides a lower torque range (1.8–2.5 kg·cm at 4.8–6 V) but is lighter and faster (≈ 0.12 s/60°). This servo is well-suited for driving the second link, which has a smaller mass, allowing precise positioning of the end effector without overloading the overall structure.

Based on the actuator specifications and mechanical configuration illustrated in Figure 10, the next stage involves connecting the physical manipulator system to a control module based on Arduino Uno, which directly interacts with the computed kinematic results (Figure 11).

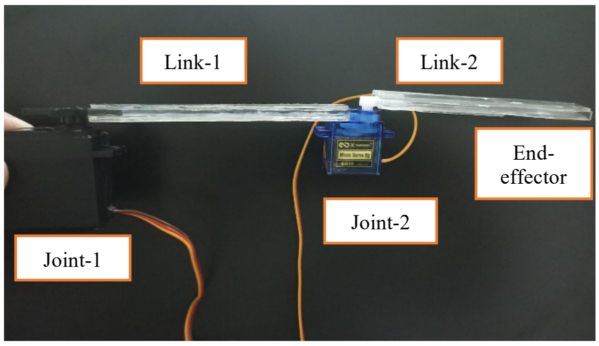

Fig. 10. Actuator and mechanical configuration

TABLE VII. ELIGIBILITY STATUS JOINT 1 AND 2

Joint	Actuator used	Stall Torque (kg·cm)	Stall Torque (N·m)*	Estimated Maximum Gravitational Torque (N.m)	Status*
1	MG996R	9.4 – 11.0	0.92 – 1.08	**0.039**	Passed (margin ≫ 20×)
2	SG90	≈ 1.8	≈ 0.18	**0.0098**	Passed (margin ≫ 15×)

In Table VII, for Joint 1, the analysis results show that the servo provides a maximum torque of 1.08 N·m, while the maximum gravitational torque requirement is only 0.039 N·m. Thus, the torque-to-load ratio reaches approximately 27.7×, indicating that the servo operates far below its maximum capacity. This condition ensures motion stability, energy efficiency, and extended servo lifespan, as the actual workload represents only about 3–4% of the motor's mechanical limit. For Joint 2, the servo delivers a maximum torque of around 0.18 N·m, while the maximum gravitational torque requirement is 0.0098 N·m, resulting in a ratio of 18.4×. This value indicates that the SG90 provides sufficient torque margin to drive the lighter second link without the risk of overload.

Figure 11 illustrates the physical two-link planar manipulator system, representing the implementation phase following the design process using the ontology-driven approach. The system is controlled via an Arduino Uno, which sends PWM signals to both servos based on the FK and IK computation results implemented in Python, connected to the CoppeliaSim simulation environment. This implementation serves as an experimental validation stage to ensure consistency between the simulation outcomes and the physical system's real-world responses.

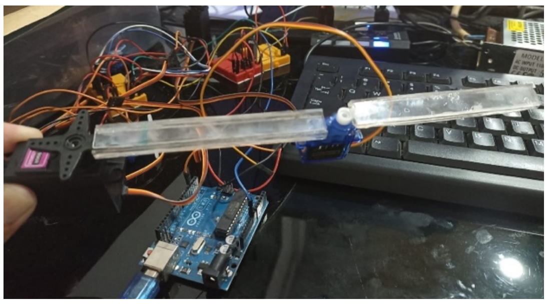
Fig. 11. Planar two-link manipulator physical system

3.7. Limitations and Future Work

At this stage of development, the two-link planar manipulator has been successfully realized in physical form and integrated with its virtual model in the simulator for DH-based kinematic validation. However, a comprehensive evaluation of the dynamic system performance, including the analysis of actuation torque against the dynamic model, has not yet been conducted in this study—it has been developed but remains unanalyzed and will be reported in the next phase of development.

In addition, although the tracking results demonstrate that the integration between the physical robot and the simulation has functioned effectively, quantitative measurements of joint motion error and end-effector accuracy—including statistical analysis such as RMSE, maximum error, and comparative tables—have not yet been fully included in this publication.

At this phase, we present the integration validation through visualizations and pseudocode implementations of IK–FK, which have been verified to execute synchronized motion on both platforms. Therefore, dynamic performance analysis and data-driven error evaluation remain open development areas for future research and experimentation.

Conclusion

This research successfully developed a two-degree-of-freedom (2-DoF) planar manipulator that is virtually and physically integrated through kinematic and dynamic modeling based on Denavit–Hartenberg parameters and the Lagrangian method. Validation through CoppeliaSim simulation and physical prototype testing demonstrated that the system can follow a circular trajectory with a radius of 0.05 m, remaining entirely within a safe workspace reach of 0.20 m. Both servos—MG996R on Joint 1 and SG90 on Joint 2—proved to have high safety margins (>15×), indicating that the actuator configuration meets the mechanical and dynamic feasibility requirements for physical implementation of the 2-DoF planar manipulator.

These findings confirm that the design configuration is appropriate for the initial exploration phase, both in terms of actuation and workspace geometry. Another key contribution of this study is the integration of ontology-driven robotic manipulator design throughout the development process. This approach unifies design elements—including physical entities (links, joints, actuators), kinematic and dynamic parameters, and configuration rules—into a semantic knowledge structure, supporting the development of physical robotic manipulators for educational purposes, particularly in validating dynamic performance and fundamental kinematic principles.

References

[1] J. Vega and V. Pérez, "G-ARM: An open-source and low-cost robotic arm integrated with ROS2 for educational purposes," *Multimedia Tools and Applications*, Springer, 2025, https://doi.org/10.1007/s11042-025-20748-8.

[2] M. Djennane et al., "Design, Analysis, and Implementation of a 3-DOF Spherical Parallel Manipulator," *IEEE Access*, vol. 12, pp. 52837–52850, Apr. 2024, https://doi.org/10.1109/ACCESS.2024.3386549.

[3] M. Massie, "Design Architecture for Dynamic Low Inertia Multi-DOF Robotic Manipulators," B.Sc. thesis, Dept. Mech. Eng., Massachusetts Inst. Technol. (MIT), Cambridge, MA, USA, 2020. [Online]. Available: MIT Libraries.

[4] S. Chakraborty and P. S. Aithal, "A Custom Robotic ARM in CoppeliaSim," *IJAEML*, vol. 5, no. 1, pp. 38–50, Apr. 2021, https://doi.org/10.5281/zenodo.4700297 K. Kruthika, K. Kumar B. M., and S. Lakshminarayanan, "Design and Development of a Robotic Arm," in *Proc. IEEE CIMCA*, Oct. 2016, pp. 1–6, https://doi.org/10.1109/CIMCA.2016.8053274.

[5] A. R. Al Tahtawi, M. Agni, and T. D. Hendrawati, "Small-scale Robot Arm Design with Pick and Place Mission Based on Inverse Kinematics," *J. Robot. Control*, vol. 2, no. 6, pp. 469–475, Nov. 2021, https://doi.org/10.18196/jrc.26124.

[6] T. Zar, T. Theingi, and S. Y. Win, "Experimental Results on Circular and Linear Movements of Two-link Robot Arm," *IJSRST*, vol. 4, no. 11, pp. 97–104, Nov.–Dec. 2018, https://doi.org/10.32628/IJSRST18401117.

[7] P. Frankovský, D. Hroncová, I. Delyová, and P. Hudák, "Inverse and Forward Dynamic Analysis of Two Link Manipulator," *Procedia Eng.*, vol. 48, pp. 158–163, 2012, https://doi.org/10.1016/j.proeng.2012.09.500.
[8] H. Mickoski, I. Mickoski, and M. Djidrov, "Dynamic Modeling and Simulation of Three-member Robot Manipulator," *Math. Models Eng.*, vol. 4, no. 4, pp. 183–190, Dec. 2018, https://doi.org/10.21595/mme.2018.20319.
[9] A. M. Eissa, M. F. El-Khatib, and M. I. Abu El-Sebah, "Dynamics Analysis and Control of a Two-Link Manipulator," *WSEAS Trans. Syst. Control*, vol. 18, pp. 487–497, Dec. 2023, https://doi.org/10.37394/23203.2023.18.52.
[10] D. A. Setiawan and H. Gunawan, "Design and Simulation of a 2-DOF Robotic Arm for Pick and Place Operation," *JETri*, vol. 21, no. 3, pp. 193–200, Sep.–Dec.2022, https://doi.org/10.24014/jetri.v21i3.984.
[11] S. Ogasawara, K. Hiramoto, and H. Doki, "Global Optimal Design of Dynamic Parameters of Robot Manipulators," in *Proc. IEEE CCA*, Singapore, Oct. 2007, pp. 307–312, https://doi.org/10.1109/CCA.2007.4389257.
[12] Z. Guo et al., "A Reinforcement Learning Approach for Inverse Kinematics of Arm Robot," in *Proc. ICRCA*, Jul. 2019, pp. 95–99, https://doi.org/10.1145/3351180.3351199.
[13] S. Dereli and R. Köker, "IW-PSO Approach to the Inverse Kinematics Problem Solution of a 7-DOF Serial Robot Manipulator," *Sigma J. Eng. Nat. Sci.*, vol. 36, no. 1, pp. 77–85, 2018. [Online]. Available: https://dergipark.org.tr.
[14] A. Gupta, A. Deshmukh, and S. Agrawal, "A Geometric Approach to Inverse Kinematics of a 3 DOF Robotic Arm," *IJRASET*, vol. 6, no. 1, pp. 367–373, Jan. 2018, https://doi.org/10.22214/ijraset.2018.1491.
[15] H. Zhang et al., "A Fully Geometric Approach for Inverse Kinematics of a Six-DOF Robot Arm," *J. Phys. Conf. Ser.*, vol. 2338, no. 1, p. 012089, 2022, https://doi.org/10.1088/1742-6596/2338/1/012089.
[16] A. N. Huda et al., "Integrated Robotic Arm Control: Inverse Kinematics, Trajectory Planning, and Performance Evaluation for Automated Welding," *Asian J. Sci. Eng.*, vol. 2, no. 2, pp. 82–100, 2023, https://doi.org/10.5281/zenodo.11032184.
[17] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL, USA: CRC Press, 1994. [Online]. Available: http://www.cds.caltech.edu/~murray/mlswiki.
[18] A. Ghosal, "Kinematics of Serial Manipulators," in *Robot Kinematics and Dynamics*, Indian Inst. Sci., Bangalore, India, 2014, pp. 1–22. [Online]. Available: https://mecheng.iisc.ac.in/~asitava.
[19] M. Kalasariya, V. Patel, and A. Thakkar, "Comparative Study of IK Methods for Serial Manipulators," *IJERT*, vol. 7, no. 5, pp. 1–5, May 2018.
[20] M. H. Zafar et al., "IK Modelling of a 3-DOF Manipulator using Hybrid Deep Learning Models," *Procedia CIRP*, vol. 120, pp. 213–218, 2023, doi: https://doi.org/10.1016/j.procir.2023.08.038.
[21] A. Kumar et al., "IK Analysis and Optimization of a 3-DOF Manipulator Using PSO," *IJRRSET*, vol. 10, no. 6, pp. 1–10, Jun. 2023. [Online]. Available: https://doi.org/10.5281/zenodo.101307.
[22] T. Dewi et al., "IK Analysis of 4 DOF Robot using Fuzzy Controller," *IJECE*, vol. 10, no. 2, pp. 1376–1386, Apr. 2020, https://doi.org/10.11591/ijece.v10i2.pp1376-1386.
[23] H. Z. Ting et al., "Kinematic Analysis for Trajectory Planning of 4-DoF Robot Arm," *IJACSA*, vol. 12, no. 6, pp. 768–775, 2021, https://doi.org/10.14569/IJACSA.2021.0120687.
[24] A. A. Mohammed and M. Sunar, "Kinematics Modeling of a 4-DOF Robotic Arm," in *Proc. ICCAR*, May 2015, pp. 87–91, https://doi.org/10.1109/ICCAR.2015.7166008.
[25] N. Manjegowda and M. Rao, "Path-based Evaluation of DL Models for IK in RP Robot," *Sci. Rep.*, vol. 15, no. 33953, Jul. 2025, https://doi.org/10.1038/s41598-025-10940-z.
[26] S. Luo et al., "IK Solution of 6-DOF Manipulator Based on Multi-Objective PSO," *Front. Neurorobot.*, vol. 16, Art. 791796, 2022, https://doi.org/10.3389/fnbot.2022.791796.
[27] M. Y. Alwardat and H. M. Alwan, "Forward and IK of 6-DOF Manipulator with Prismatic Joint," *Int. J. Adv. Technol. Eng. Explor.*, vol. 11, no. 117, pp. 1096–1110, 2024, https://doi.org/10.19101/IJATEE.2024.111100210.
[28] M. F. Shah et al., "IK for Upper-Limb Rehab Robot using DL Models," *Neural Comput. Appl.*, vol. 37, pp. 12991–13009, 2025, https://doi.org/10.1007/s00521-025-11222-5
[29] G. Xuan and Y. Shao, "Reverse-driving Trajectory Planning and Simulation," *IFAC-PapersOnLine*, vol. 51, no. 17, pp. 384–388, 2018, https://doi.org/10.1016/j.ifacol.2018.08.191.
[30] Z. Chong et al., "Topology Optimization of a 2-DoF Hybrid Arm," *Int. J. Intell. Robot. Appl.*, vol. 4, pp. 136–148, 2020, https://doi.org/10.1007/s41315-020-00133-4
[31] Z. B. Hazem et al., "RL-based Trajectory Tracking for 5-DOF Mitsubishi Robot," *Int. J. Intell. Robot. Appl.*, vol. 9, no. 1, pp. 1–28, 2025, https://doi.org/10.1007/s41315-025-00475-x.
[32] D. San et al., "Energy-aware RL-based Trajectory Tracking," *Discover Robotics*, vol. 3, no. 4, pp. 1–18, 2025, https://doi.org/10.1007/s44430-025-00004-2.
[33] M. Y. Alwardat et al., "Kinematic Analysis for RRRRRP Manipulator," *Russ. Eng. Res.*, vol. 44, no. 11, pp. 1640–1647, 2024, https://doi.org/10.3103/S1068798X24702691.
[34] S. Manzoor et al., "Ontology-Based Knowledge Representation in Robotic Systems," *Appl. Sci.*, vol. 11, no. 4324, pp. 1–21, 2021, https://doi.org/10.3390/app11104324.
[35] C. Schlenoff et al., "An IEEE Standard Ontology for Robotics and Automation," in *Proc. IEEE/RSJ IROS*, 2012, pp. 1337–1342, https://doi.org/10.1109/IROS.2012.6385518 .
[36] P. Bhounsule, "CoppeliaSim/tree/main/460," GitHub repository, 2024. [Online]. Available: https://github.com/pab47/CoppeliaSim/tree/main/460.
[37] A. M. Eissa, M. F. El-Khatib, and M. I. Abu El-Sebah, *"Dynamics Analysis and Control of a Two-Link Manipulator," WSEAS Transactions on Systems and Control*, vol. 18, pp. 487–497, Dec. 2023. https://doi.org/10.37394/23203.2023.18.52.
[38] J. A. Shah and S. S. Rattan, "Dynamic Analysis of Two Link Robot Manipulator for Control Design," *International Journal of Robotics and Automation (IJRA)*, vol. 5, no. 4, pp. 145–152, 2016. [Online]. Available: https://eprints.gla.ac.uk/252669/ .
[39] *Coppelia Robotics GmbH*, "CoppeliaSim: Robot simulation software," 2025. [Online]. Available: https://www.coppeliarobotics.com.