CEC Continuing Education Center

DesignNews

Modeling Robot Kinematics using Python and AI

DAY 5: Understanding and Modeling Translation and Rotation of a Robot Arm

Sponsored by

DigiKey

informa markets

Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Attendee Chat’ by maximizing the chat widget in your dock.

Dr. Don Wilcher

Visit ‘Lecturer Profile’ in your console for more details.

Course Kit and Materials

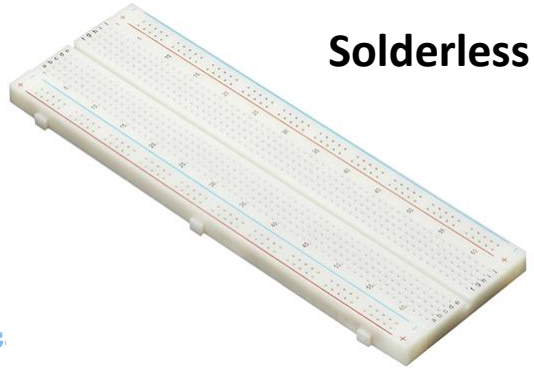
Adeept 5-DOF Robot Arm Kit

ESP32-DEVKITC-VIE

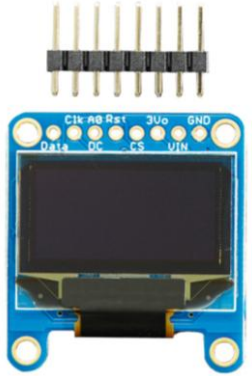
9G SERVO MOTOR KIT 180DEG

Adafruit Parts Pal Kit

Solderless Breadboard x2



OLED Display

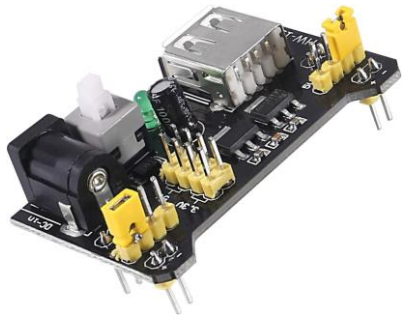


Course Kit and Materials

Jumper Wires: Male to Male

Jumper Wires: Male to Female

Solderless Breadboard Power Supply



18650 Rechargeable Battery

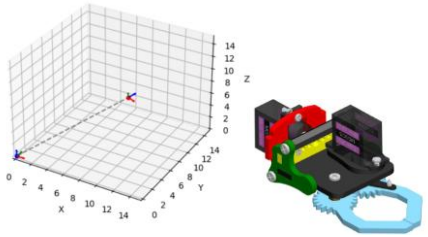
Research Perspective

- “Robotics is the art, knowledge base, and know-how of designing, and applying, and using robots in human endeavors (Niku, 2020).”
- “Using matrices, we first establish a method of describing objects, locations, orientations, and movements (Niku, 2020).”

Agenda:

- Understanding the Kinematics of a Robot Arm
- Homogeneous Transformations
- Using Homogeneous Transformation Matrices
- Lab: Building Rotation and Translation Robotic Arm Code Using an AI LLM

Understanding the Kinematics of a Robot Arm

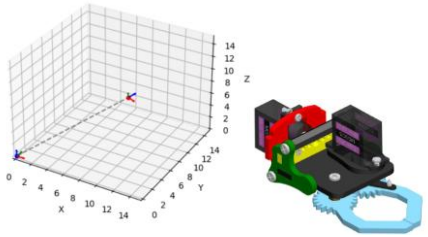


Kinematics is the study of motion without considering forces. For a **robot arm**, kinematics describes how the end effector moves based on the motion of its joints. Understanding kinematics is essential for **robotic control, planning movements, and simulation**.

There are two main types of kinematics for a robot arm:

1. **Forward Kinematics (FK)** – Determines the end effector's position and orientation given the joint parameters.
2. **Inverse Kinematics (IK)** – Determines the required joint parameters to achieve a desired end effector position and orientation.

Understanding the Kinematics of a Robot Arm. . .



1. Forward Kinematics (FK)

Forward kinematics is the process of computing the **end effector's position and orientation** based on the **joint angles and link lengths**.

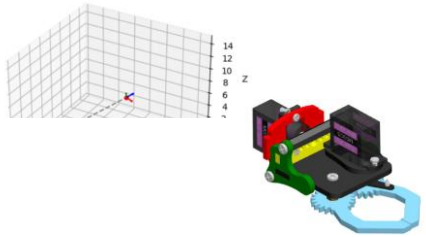
1.1. Representation of Robot Arm Motion

A **robot arm** consists of:

- **Joints** (Revolute or Prismatic)
- **Links** (Rigid bodies connecting joints)
- **Reference Frames** (Each joint and link has its coordinate system)

To systematically describe motion, we use **Homogeneous Transformation Matrices**.

Understanding the Kinematics of a Robot Arm. . .



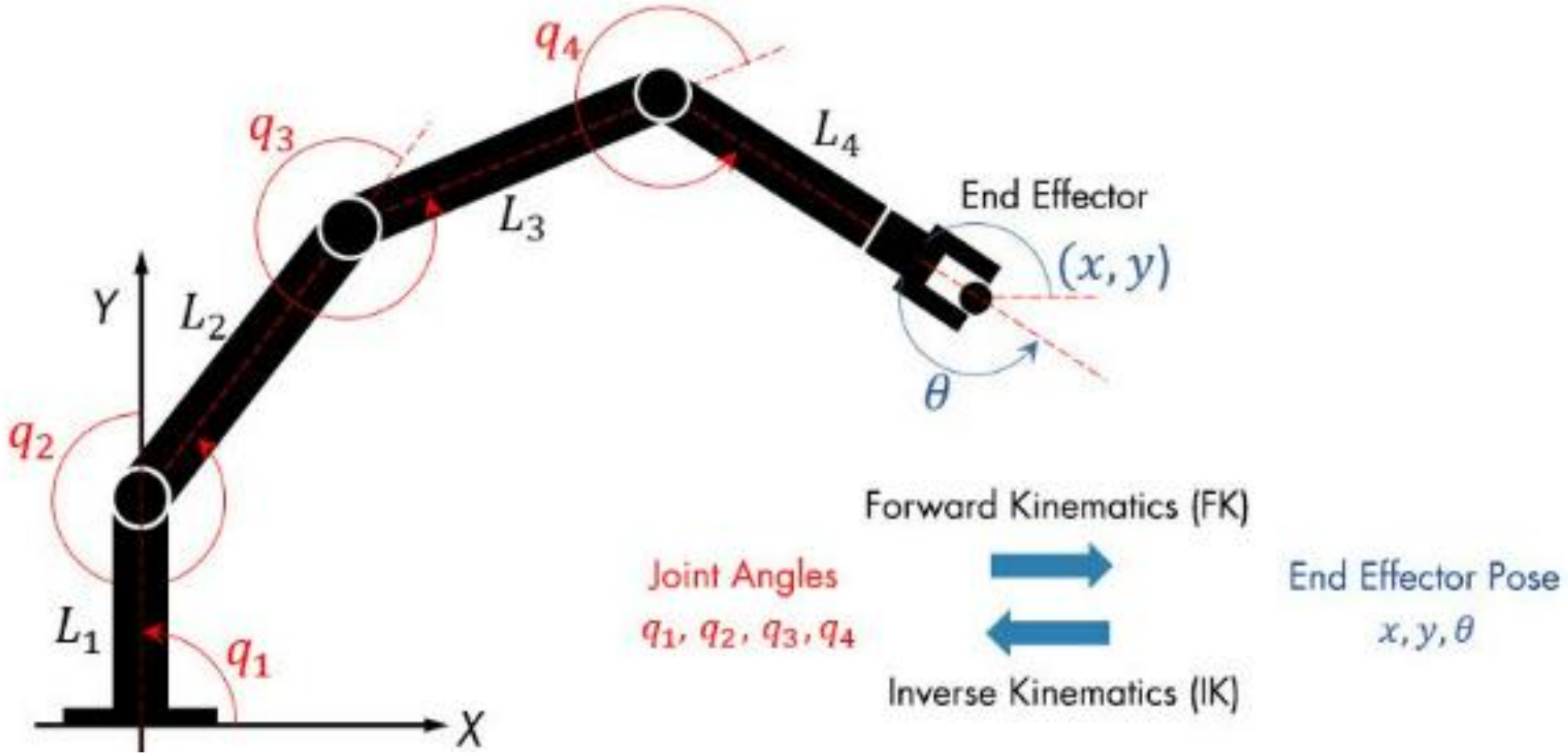
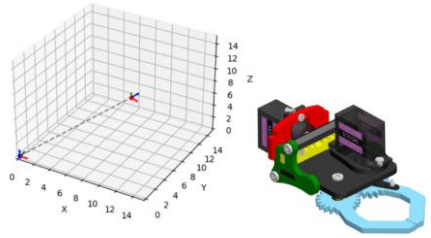
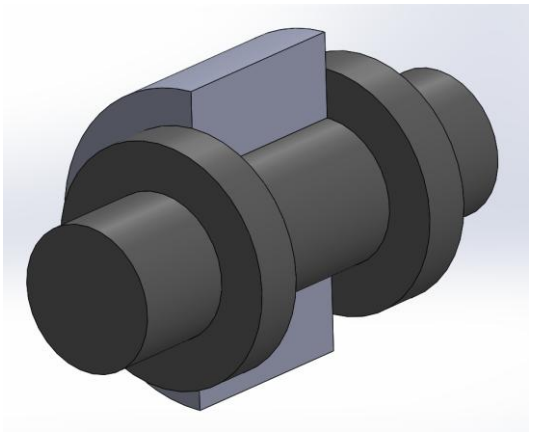


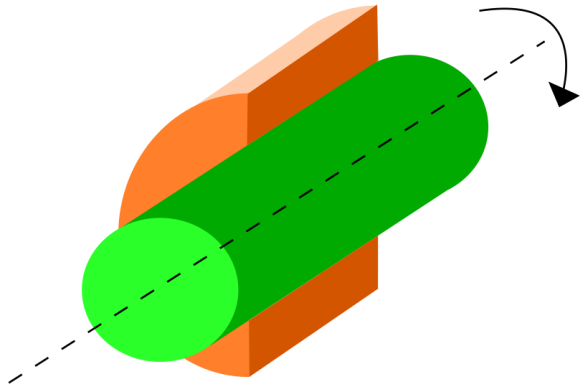
Image: Mathworks

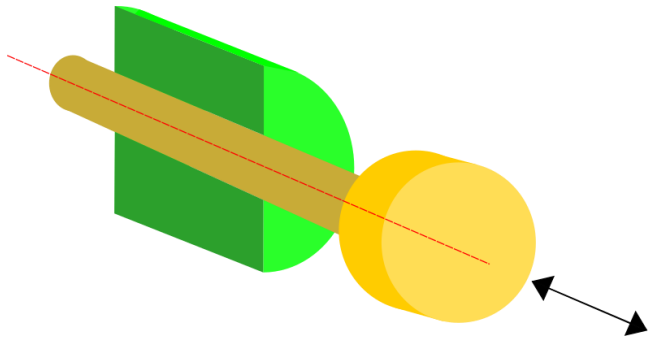
Forward and Inverse Robot Arm Kinematics Model

Understanding the Kinematics of a Robot Arm. . .









Prismatic Joint: Slider or Sliding Pair

Revolute Joint: Pin or Hinge Joint

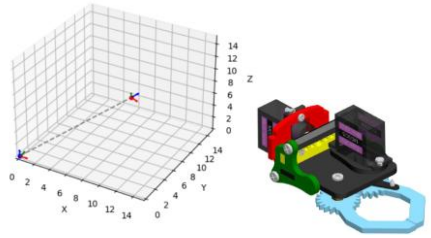
Image: Wikipedia

Question 1

What are the two main types of Kinematics?

a) Forward & Backward.
b) Backward & Reverse.
c) Forward & Inverse
d) none of the above

Understanding the Kinematics of a Robot Arm. . .



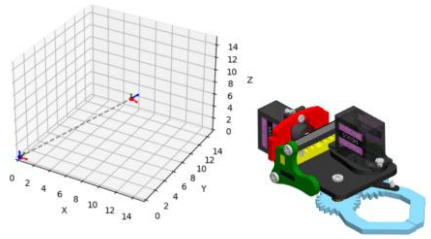
1.2. Denavit-Hartenberg (DH) Convention

To describe a **serial manipulator**, we use the **Denavit-Hartenberg (DH) convention**, which assigns a **coordinate frame** to each link.

Each link transformation is described using four **DH parameters**:

1. **θ (theta)** – Rotation around the Z-axis (joint angle for revolute joints)
2. **d** – Translation along the Z-axis (joint displacement for prismatic joints)
3. **a** – Link length (distance along the X-axis)
4. **α (alpha)** – Link twist (rotation around the X-axis)

Understanding the Kinematics of a Robot Arm. . .



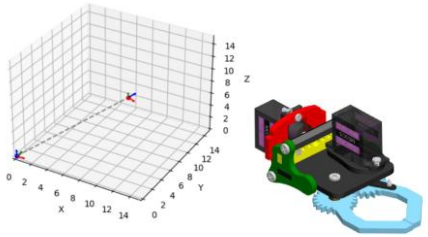
Each joint transformation is given by:

$$T_i^{i+1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The total transformation is obtained by multiplying all individual transformations:

$$T_0^n = T_0^1 \cdot T_1^2 \cdot \ldots \cdot T_{n-1}^n$$

Understanding the Kinematics of a Robot Arm. . .



1.3. Example: Two-Link Planar Robot Arm

Consider a **2-link planar manipulator** with **two revolute joints**. The transformation matrices are:

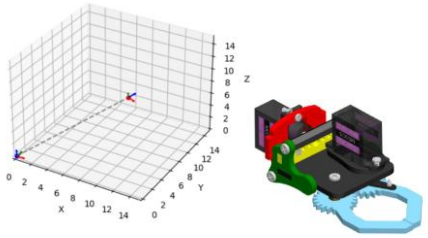
For **Joint 1**:

$$T_0^1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & L_1\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1 & 0 & L_1\sin\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For **Joint 2**:

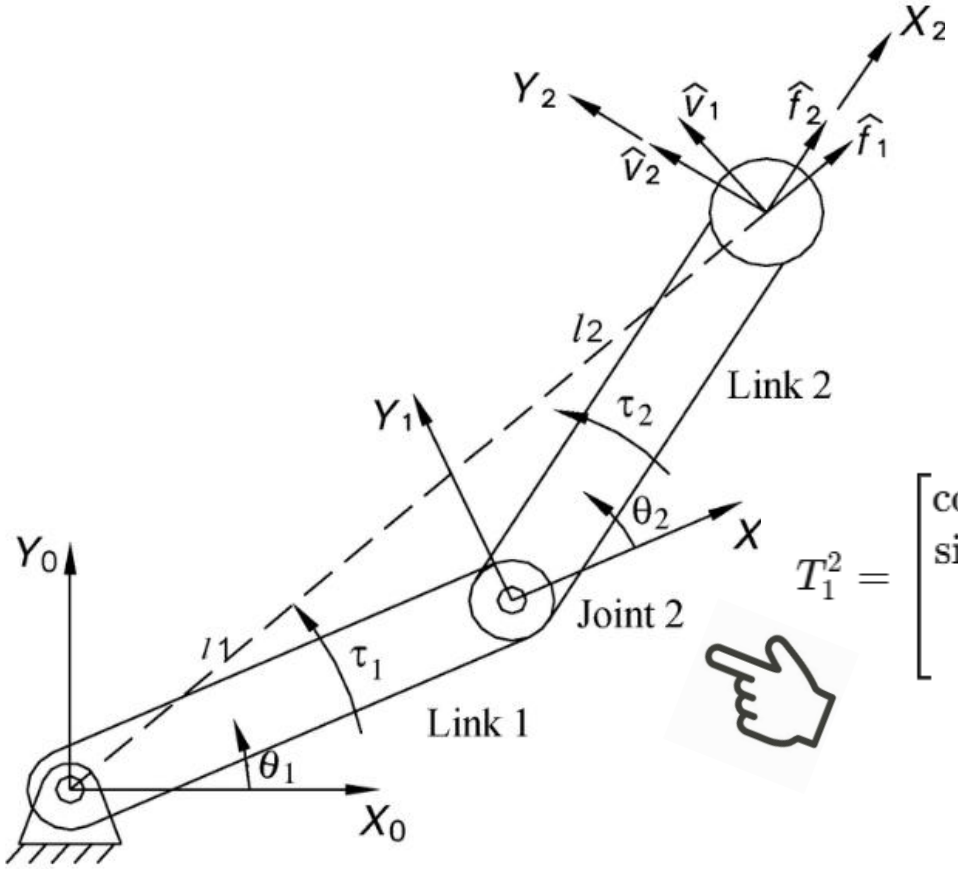
$$T_1^2 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & L_2\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & L_2\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Understanding the Kinematics of a Robot Arm. . .



A 2-Link Planar Manipulator Model

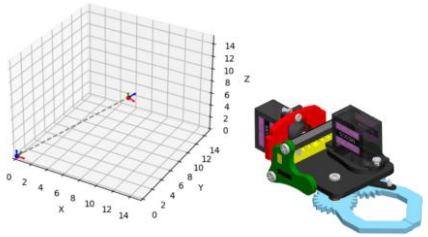
$$T_0^1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & L_1\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1 & 0 & L_1\sin\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$T_1^2 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & L_2\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & L_2\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Image: ResearchGate

Understanding the Kinematics of a Robot Arm. . .



The final-end effector position is:

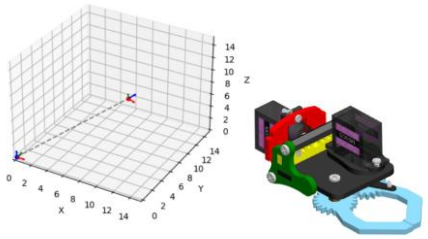
$$T_0^2 = T_0^1 \cdot T_1^2$$

Thus, the end effector position is:

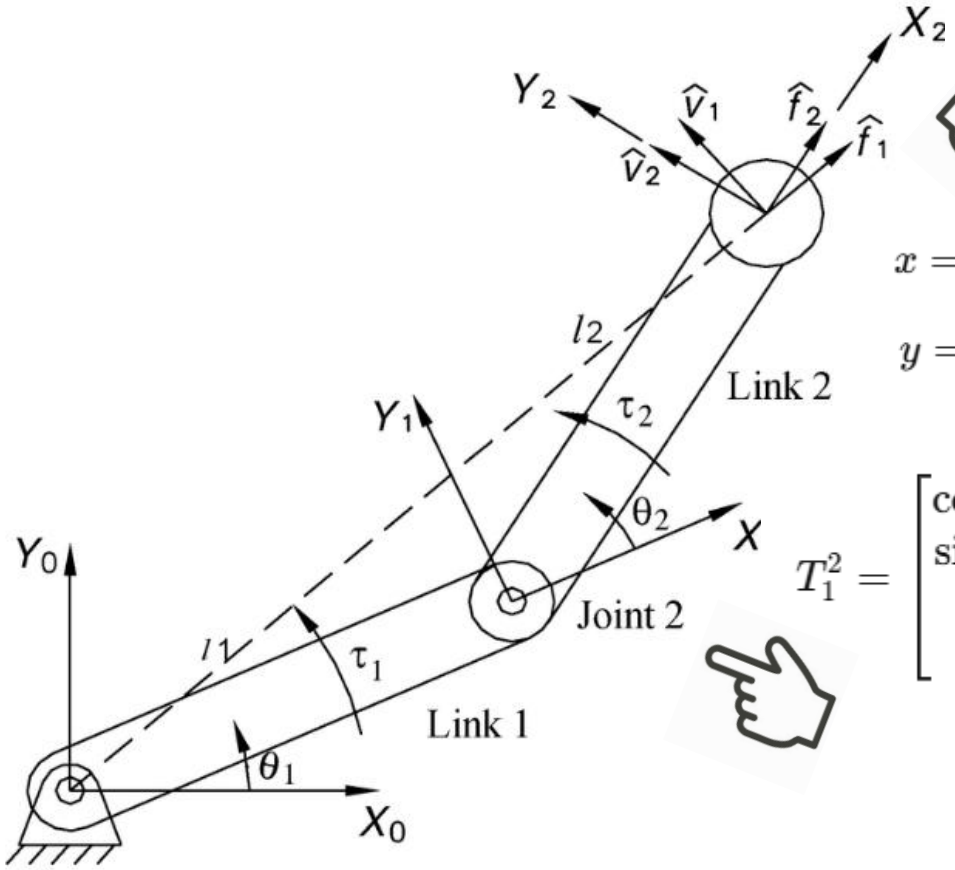
$$x = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2)$$

$$y = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2)$$

Understanding the Kinematics of a Robot Arm. . .



A 2-Link Planar Manipulator Model



$$x = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2)$$

$$y = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2)$$

$$T_0^1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & L_1\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1 & 0 & L_1\sin\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_1^2 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & L_2\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & L_2\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

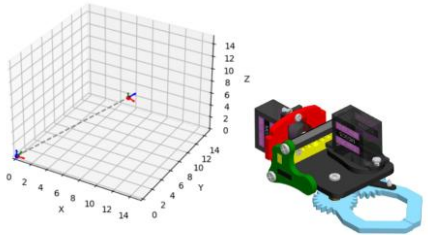
Image: ResearchGate

Question 2

What is the primary joint for a 2-link Planar Manipulator?

a) prismatic
b) evolute
c) revolute
d) none of the above

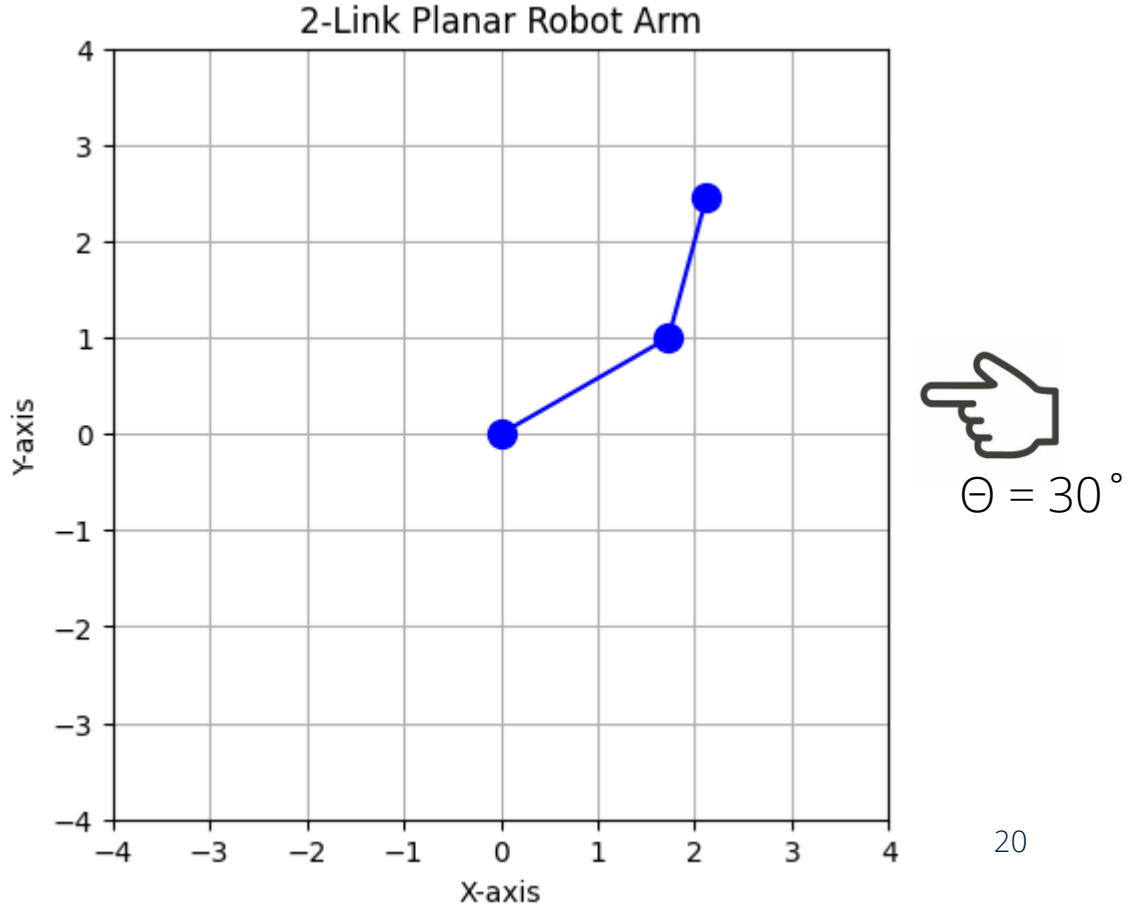
Understanding the Kinematics of a Robot Arm. . .



A 2-Link Planar Manipulator: Python Model

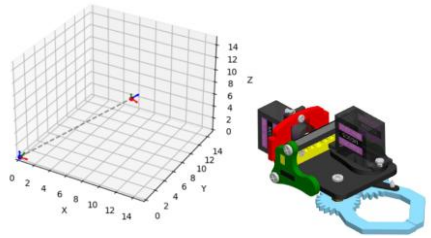
```python
def forward_kinematics(theta1, theta2, L1, L2):
    """Computes the end effector position for a 2-link planar robot arm"""
    x1 = L1 * np.cos(theta1)
    y1 = L1 * np.sin(theta1)

    x2 = x1 + L2 * np.cos(theta1 + theta2)
    y2 = y1 + L2 * np.sin(theta1 + theta2)

    return [(0, 0), (x1, y1), (x2, y2)]

# Robot arm parameters
L1, L2 = 2, 1.5  # Link lengths
theta1, theta2 = np.radians(30), np.radians(45)  # Joint angles
```

Θ = 30˚



Θ = 30˚

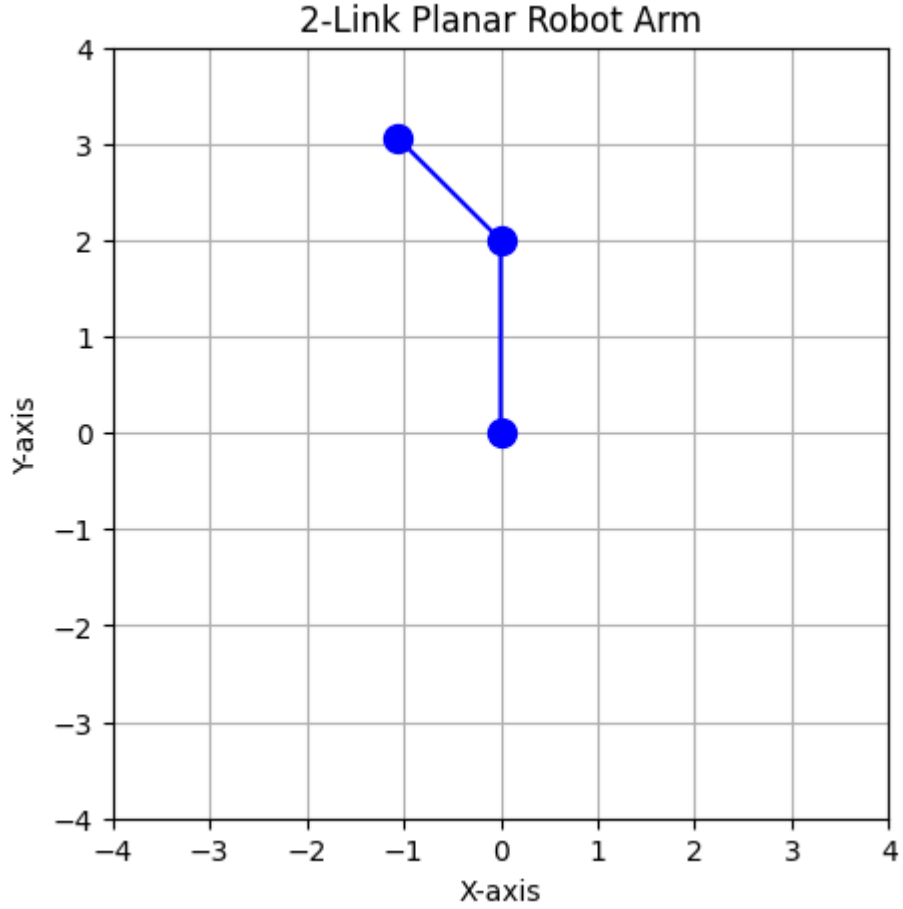
Understanding the Kinematics of a Robot Arm. . .



A 2-Link Planar Manipulator: Python Model

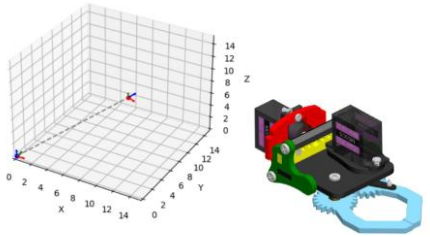
```
# Robot arm parameters
L1, L2 = 2, 1.5  # Link lengths
theta1, theta2 = np.radians(90), np.radians(45)  # Joint angles
```

Θ = 90˚



Θ = 90˚

Homogeneous Transformations



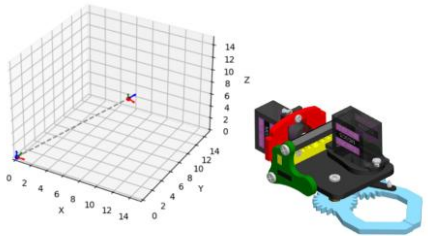
- Why use Homogeneous Coordinates?
  Combines rotation and translation into a single matrix operation
- Transformation Matrix Representation:

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

where:

- R: 3x3 Rotation Matrix
- t: 3x1 Translation Vector
- Last row ensures proper matrix multiplication

Homogeneous Transformations. . .



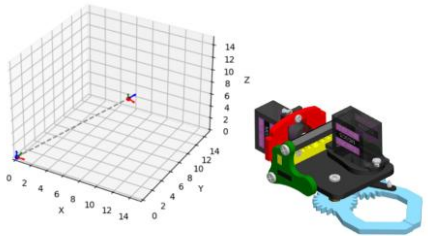
The **translation vector t** in rigid body motion represents the displacement of an object in space. It is a **3×1 column vector** that specifies how much the object moves along the **X, Y, and Z axes**.
Mathematically, it is written as:

$$t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

where:

- tx is the displacement along the X-axis.
- ty is the displacement along the Y-axis.
- tz is the displacement along the Z-axis.

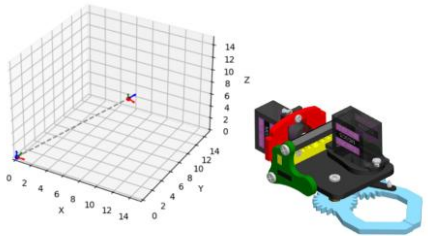
Homogeneous Transformations. . .



Key Points:

- Translation does not affect the shape or orientation of the object.
- It is a vector addition to the original position.
- It is essential in robotics to define the movements of robotic arms and mobile robots.

Using Homogeneous Transformation Matrices



Using Homogeneous Transformation Matrices

To represent both translation and rotation in a single mathematical structure, we use **homogeneous transformation matrices**:

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

where:

- R is the **rotation matrix** (3×3)
- t is the **translation vector** (3×1)
- The last row ensures proper transformation in homogeneous coordinates.

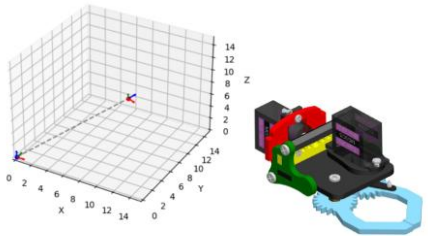
Each joint in the robot arm has its transformation matrix, and the overall motion is obtained by multiplying these matrices in sequence.

Question 3

Which statement is correct about the translation vector?

a) tx is the displacement along the z-axis
b) tx is the displacement along the y-axis
c) tx is the displacement along the x-axis
d) none of the above

Using Homogeneous Transformation Matrices. . .



Rotation Matrices for a 3D Robot Arm

Each joint can rotate around one of the three principal axes. The corresponding rotation matrices are:

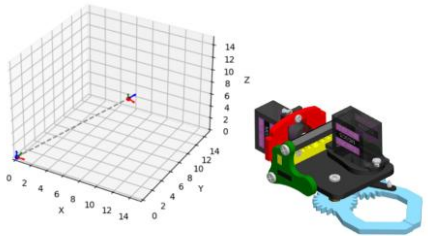
Rotation around the X-axis:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

Rotation around the Y-axis:

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

Using Homogeneous Transformation Matrices. . .

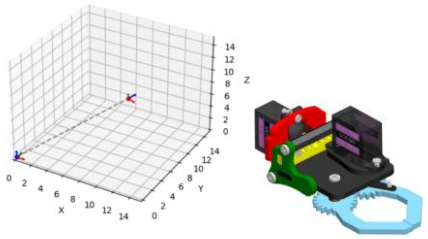


Rotation around the Z-axis:

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Each of these rotation matrices preserves the **lengths and angles** of objects, ensuring **rigid body motion**.

Using Homogeneous Transformation Matrices. . .

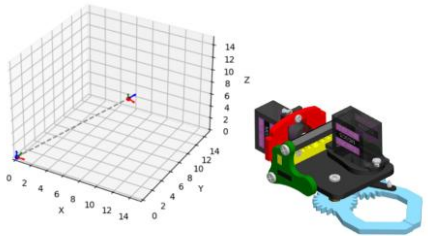


Rotation around the Z-axis:

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Each of these rotation matrices preserves the **lengths and angles** of objects, ensuring **rigid body motion**.

Using Homogeneous Transformation Matrices. . .

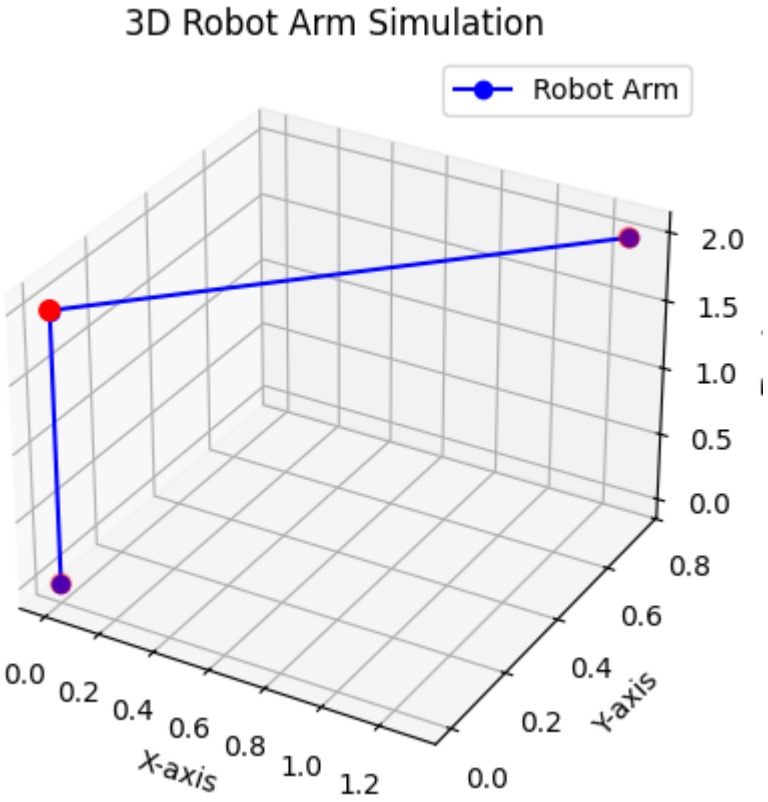


Implementing a Simple 3D Robot Arm in Python

Python code can demonstrate a **two-joint** robotic arm that can rotate and translate in 3D space.

```
# Define link lengths
L1 = 2
L2 = 1.5

# Define joint angles (in radians)
theta1 = np.pi / 6  # 30 degrees rotation about Z
theta2 = np.pi / 4  # 45 degrees rotation about Y

# Define translation vectors
t1 = np.array([0, 0, L1])  # First joint translation
t2 = np.array([L2, 0, 0])  # Second joint translation

# Compute transformation matrices
R1 = rotation_matrix('z', theta1)
R2 = rotation_matrix('y', theta2)

T1 = homogeneous_transform(R1, t1)
T2 = homogeneous_transform(R2, t2)

# Compute final end effector position
P0 = np.array([0, 0, 0, 1])  # Base position
P1 = T1 @ P0  # First joint position
P2 = T1 @ T2 @ P0  # End effector position
```

Equations transformed into a 3D Robotic Arm

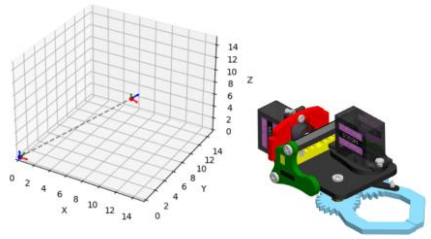


Question 4

In reviewing slide 30, what robotic device was being demonstrated in 3D space?

a) 2-link planar manipulator
b) 2-link prismatic manipulator
c) 2 DoF robot
d) 2-joint robotic arm

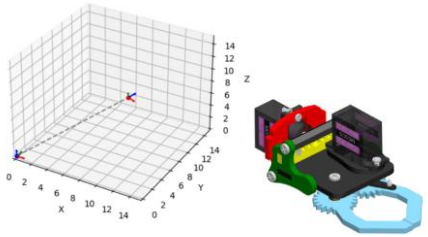
Using Homogeneous Transformation Matrices. . .

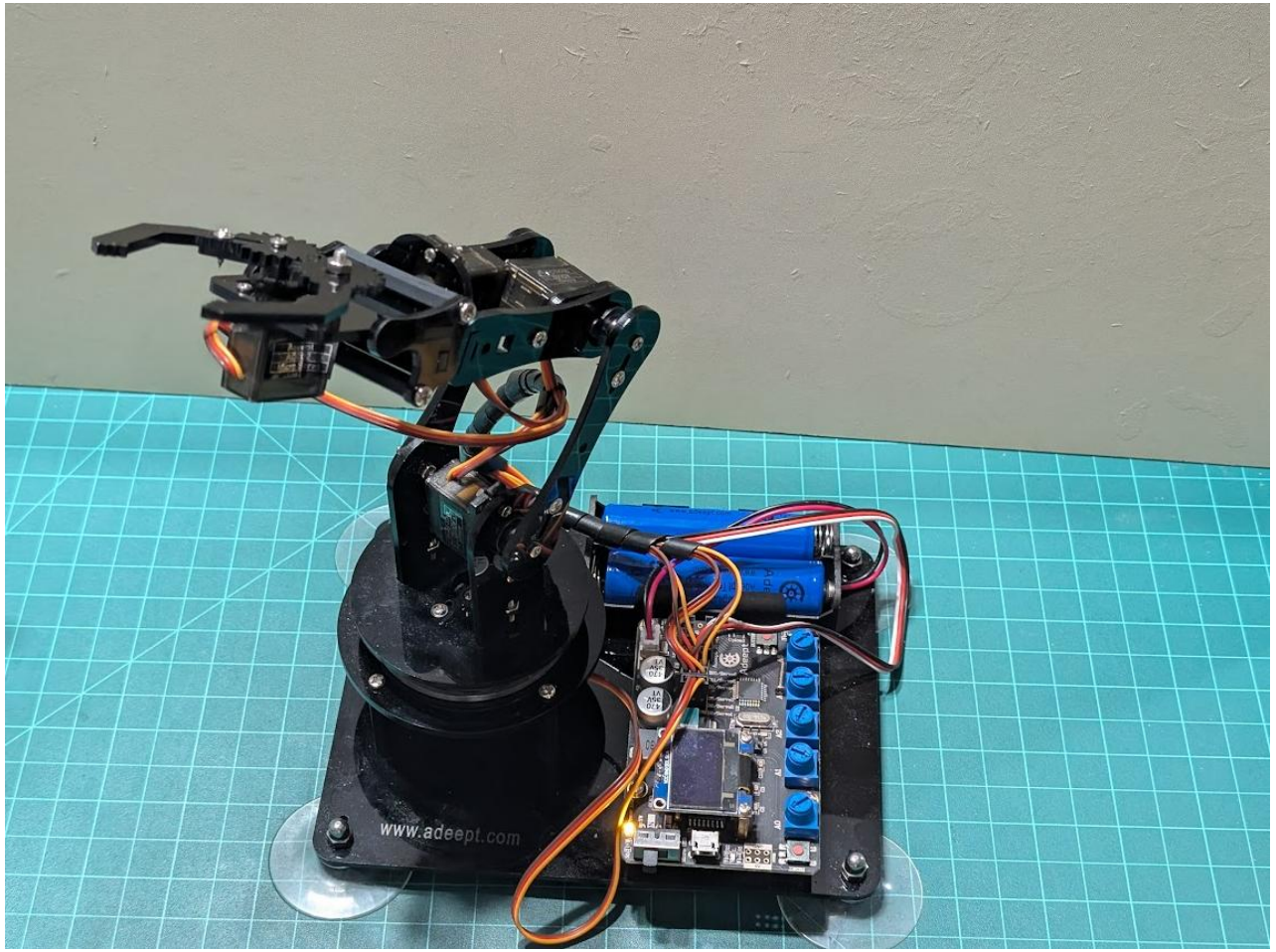


Interpretation of the Python Code (See Resources)

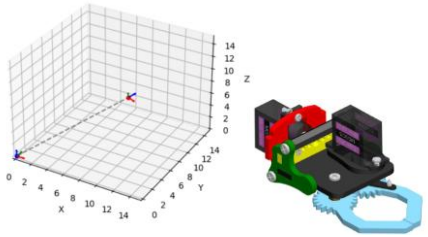
1. **Defines Rotation Matrices:** Uses rotation matrices for each joint.
2. **Computes Homogeneous Transformations:** Represents both rotation and translation in a matrix form.
3. **Applies Forward Kinematics:** Computes the position of each joint step-by-step.
4. **Visualizes the Robot Arm:** Uses Matplotlib to display the arm's movement.

Lab: Build Rotation and Translation Robotic Arm Code Using an AI LLM





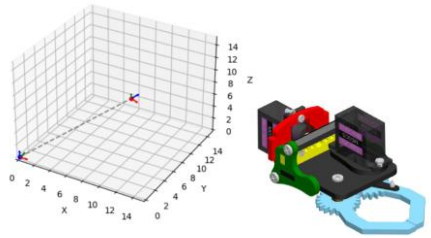
Lab: Build Rotation and Translation Robotic Arm Code Using an AI LLM. . .

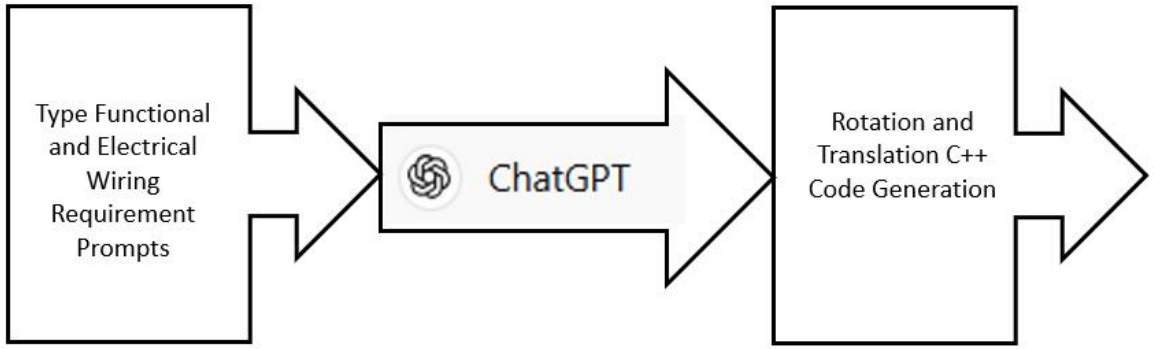


Participant Learning Objectives:

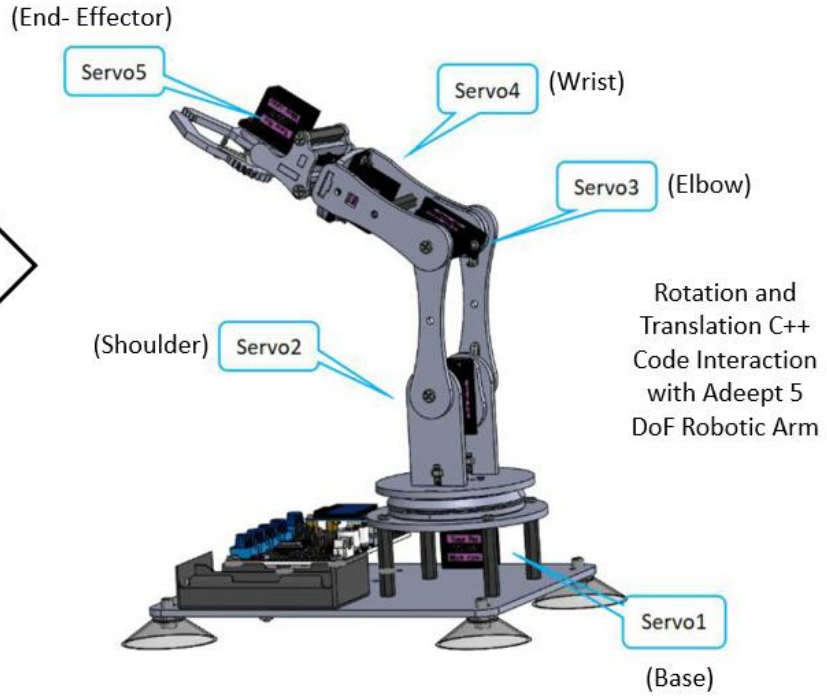
- Participants will learn to write prompts to an AI LLM with functional requirements and output wiring requirements.
- Participants will learn to program the Adeept Robotic Arm using the AI LLM-generated C++ code
- Participants will learn to calibrate the Adeept Robotic Arm using the AI LLM-generated C++ Code.

Lab: Build Rotation and Translation Robotic Arm Code Using an AI LLM. . .

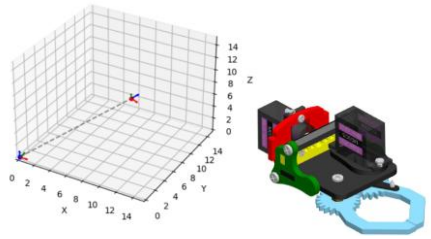




AI Code Generation to Robotic Arm Operation Concept Diagram



Lab: Build Rotation and Translation Robotic Arm Code Using an AI LLM. . .



The origin of the C++ Code is the Arduino Servo Motor sketch

ChatGPT AI LLM

Prompt 1

Functional and Electrical Wiring Requirements

Using the 2-Planar Robot Manipulator, please modify the following C++ code to rotate the base of an Adeept Robot and translate it's end-effector to an upward motion using the elbow. Here the specifics of the Adeept Robot's base and the elbow.

Servo Motor 3(Elbow)----------> Digital Pin 5
Servo Motor 1(Base)----------->Digital Pin 9

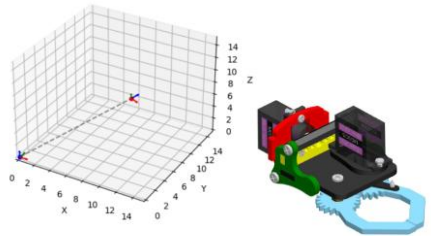
Test Result: Robot effector not raised to sufficient height.

Prompt 2

Please modify the code to include the shoulder. The shoulder is Servo Motor 2, attached to Digital Pin 6.

Test Result: Robot Arm rotated and translated properly.

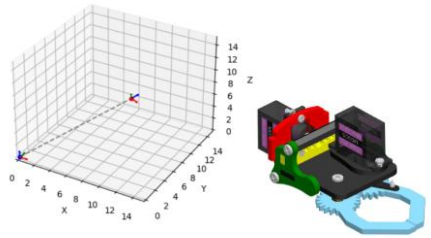
Lab: Build Rotation and Translation Robotic Arm Code Using an AI LLM. . .



Functional and Electrical Wiring Requirements Table

Robot Joint	Servo Motor	Digiital Pin	Potentiometer (L-R)	Analog Pin	Rotation (Degrees)
Base	1	9	1	A0	0 - 180
Shoulder	2	6	2	A1	0 - 180
Elbow	3	5	3	A2	0 - 180
Wrist	4	3	4	A3	0 - 180
End Effector	5	11	5	A6	35- 90

Lab: Build Rotation and Translation Robotic Arm Code Using an AI LLM. . .



Partial C++ Code Generated:

Note:
To prevent compilation errors, use Arduino IDE 1.8.19 version

Translation_Rotation_Elbow_Shoulder_Base | Arduino 1.8.19

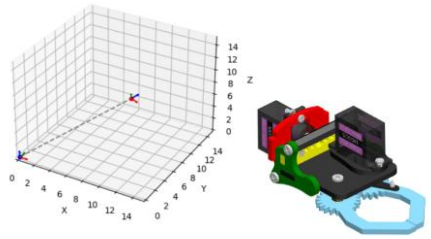
File Edit Sketch Tools Help

Translation_Rotation_Elbow_Shoulder_Base

```
// Pin Assignments
const int basePin = 9;      // Base servo -> Digital Pin 9
const int shoulderPin = 6; // Shoulder servo -> Digital Pin 6
const int elbowPin = 5;     // Elbow servo -> Digital Pin 5

// Calibration Angles (Adjust for your setup)
const int baseHomeAngle = 0;         // Neutral base position
const int baseRotateAngle = 45;       // No base rotation in this version
const int shoulderHomeAngle = 0;     // Default shoulder position
const int shoulderLiftAngle = 45;    // Move shoulder up
const int elbowHomeAngle = 0;        // Default elbow position
const int elbowUpwardAngle = 90;     // Move elbow up for full extension

const int servoDelay = 15; // Delay between movements
```

Sandbox Area for R-T Exploration

Lab: Build Rotation and Translation Robotic Arm Code Using an AI LLM. . .



Partial C++ Code Generated:

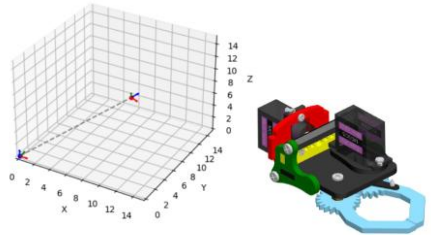
```
void loop() {
  // Step 1: Rotate the base (currently kept at 0°)
  rotateBase();
  delay(1000); // Pause before next movement

  // Step 2: Move the shoulder upward
  moveShoulderUp();
  delay(1000); // Pause before elbow movement

  // Step 3: Move the elbow upwards
  moveElbowUp();
  delay(2000); // Hold position

  // Step 4: Return to home position
  returnHome();
  delay(2000); // Pause before repeating
}
```

Rotate Event

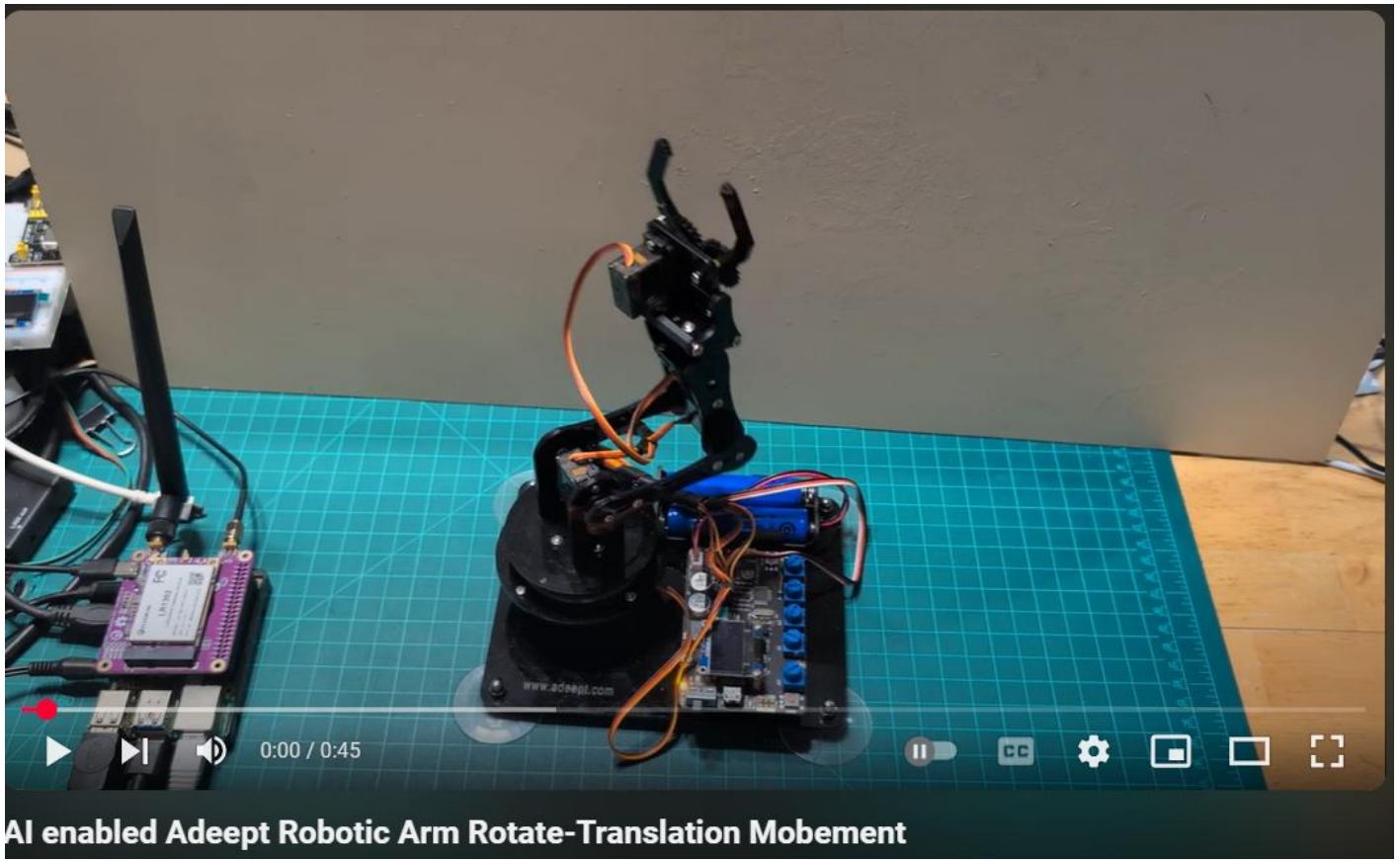
Translation Event

Final Note:
Review the Code critically, adverse – safety concerns exist!

Lab: Build Rotation and Translation Robotic Arm Code Using an AI LLM. . .



Adeept 5 DoF Robotic Arm In Action



Question 5

What version of the Arduino IDE was used to avoid software compilation errors?

a) 1.8.18
b) 1.8.20
c) 1.8.19
d) none of the above

Thank you for attending

Please consider the resources below:

Bravo, F. A., & Cruz-Bohorquez, J. M. (2024). Engineering education in the age of ai: Analysis of the impact of chatbots on learning in engineering. *Education Sciences 14*(484), 1-20, https://doi.org/10.3390/educsci14050484

Niku, S.B. (2020). Introduction to robotics: Analysis, control, applications (3rd ed). Wiley

Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2004). Robot dynamics and control (2nd ed), Wiley. https://www.kramirez.net/Robotica/Tareas/Kinematics.pdf

Walter, Y. (2024). Embracing the future of ai in the classroom: The relevance of ai literacy, prompt engineering, and critical thinking in modern education. *International Journal of Educational Technology in Higher Education 21*(15), *1*-29. https://doi.org/10.1186/s41239-024-00448-3

Wilcher, D. (2025). *Modeling robot kinematics using python and ai*. GitHub. https://github.com/DWilcher/DesignNews-WebinarCode/blob/main/March_25_Webinar_Code.zip

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

DigiKey

informa markets