CEC Continuing Education Center

DesignNews

Modeling Robot Kinematics using Python and AI

# DAY 2 : Understanding Basic Robot Movement of an End Effector

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

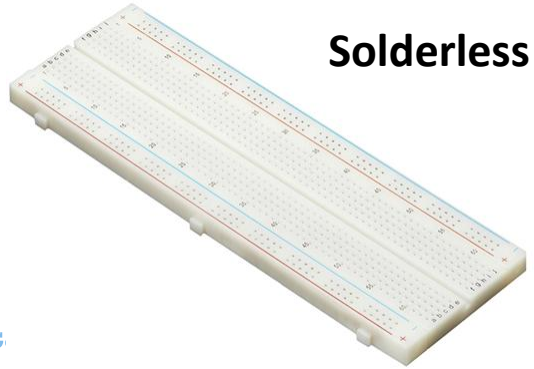
**Adeept 5-DOF Robot Arm Kit**

**ESP32-DEVKITC-VIE**

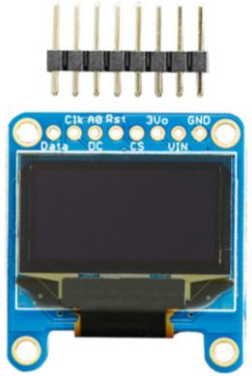
**9G SERVO MOTOR KIT 180DEG**


**Adafruit Parts Pal Kit**

**Solderless Breadboard x2**



**OLED Display**

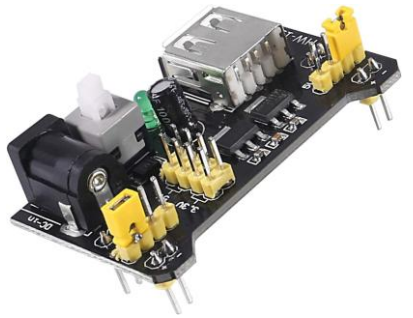


# Course Kit and Materials

**Jumper Wires: Male to Male**

**Jumper Wires: Male to Female**

**Solderless Breadboard Power Supply**



**18650 Rechargeable Battery**

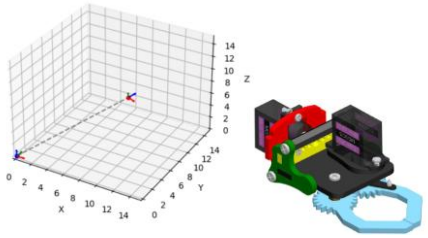
# Research Perspective

- “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:

- Robot Coordinate Systems
- The Point Matrix
- Translation Matrix
- Lab: Build a Translation Matrix In Python

# Robot Coordinate Systems

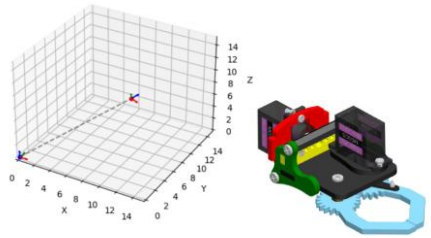


In robotics, different coordinate systems describe the position and orientation of a robot or its components in 3D space.

## World Coordinate System (Global Frame)

a) The world coordinate system is a fixed reference frame used to describe the robot's position in the environment.
b) It is typically defined with an origin at a known point in 3D space (e.g., the floor of a factory or the center of a workspace).
c) The robot's position and movements are measured relative to this global reference.

# Robot Coordinate Systems . . .



## World Coordinate System (Global Frame)

**Global Frame:** Coordinates based on the robot's location within the plant or manufacturing environment

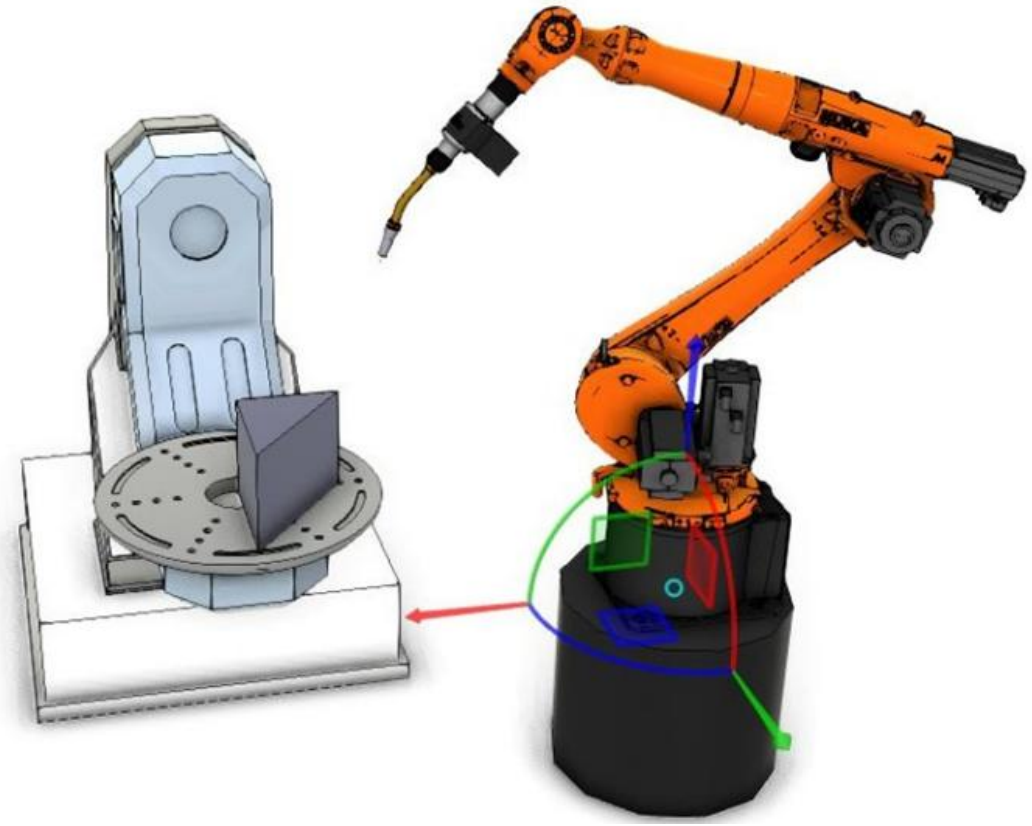
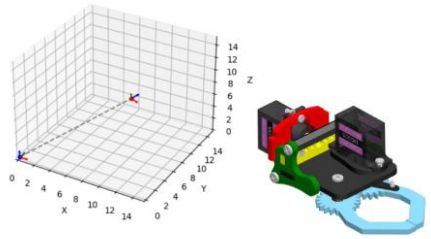


Image: SolisPLC

# Robot Coordinate Systems. . .



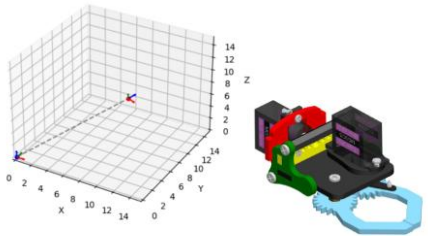
## Robot Base Coordinate System (World Frame)

a) This coordinate system is attached to the base of the robot.

b) The origin is usually at the robot's base or a predefined point on its body.

c) All other coordinate frames (e.g., tool, joint, or end-effector) are often defined relative to this system.

# Robot Coordinate Systems. . .



Robot Base Coordinate System: The coordinate system is attached to the base of the robot.

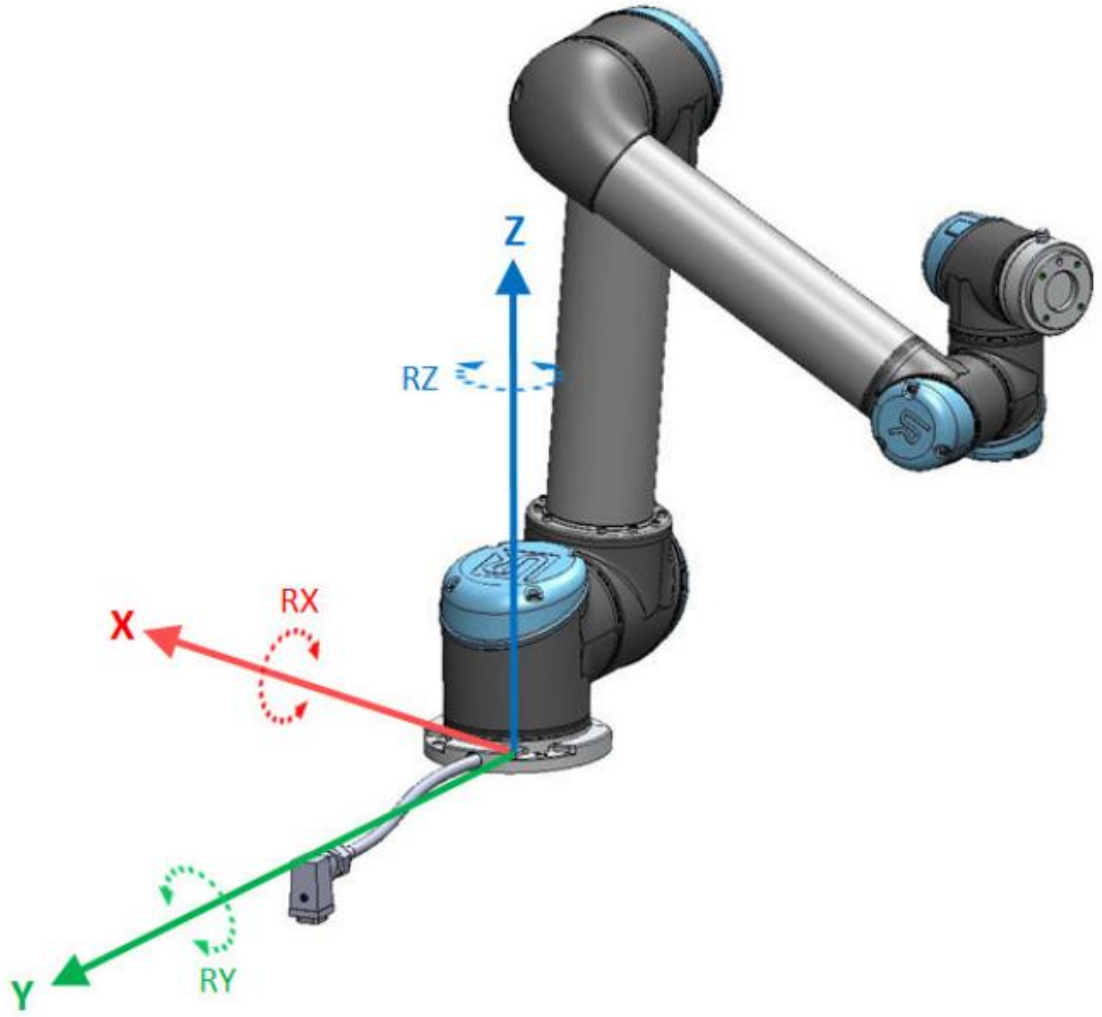
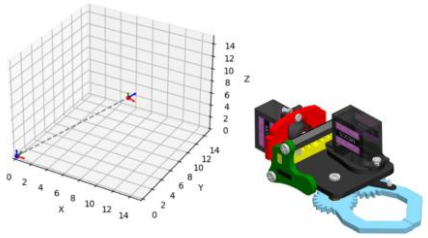



Image: Universal Robots

# Robot Coordinate Systems. . .



## Robot Base Coordinate System (World Frame)

**Right-Hand Rule:** Can aid in determining the World Frame

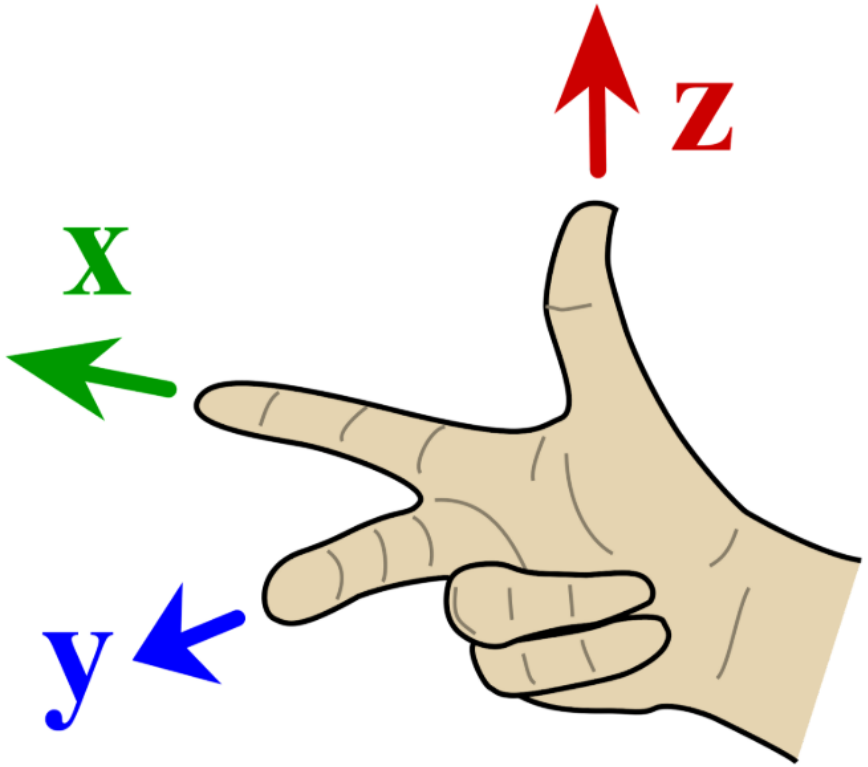


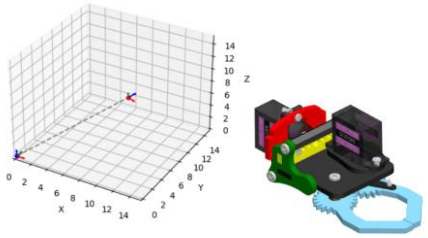

Image: SolisPLC

# Question 1

**What rule allows one to identify a Robot World Frame?**

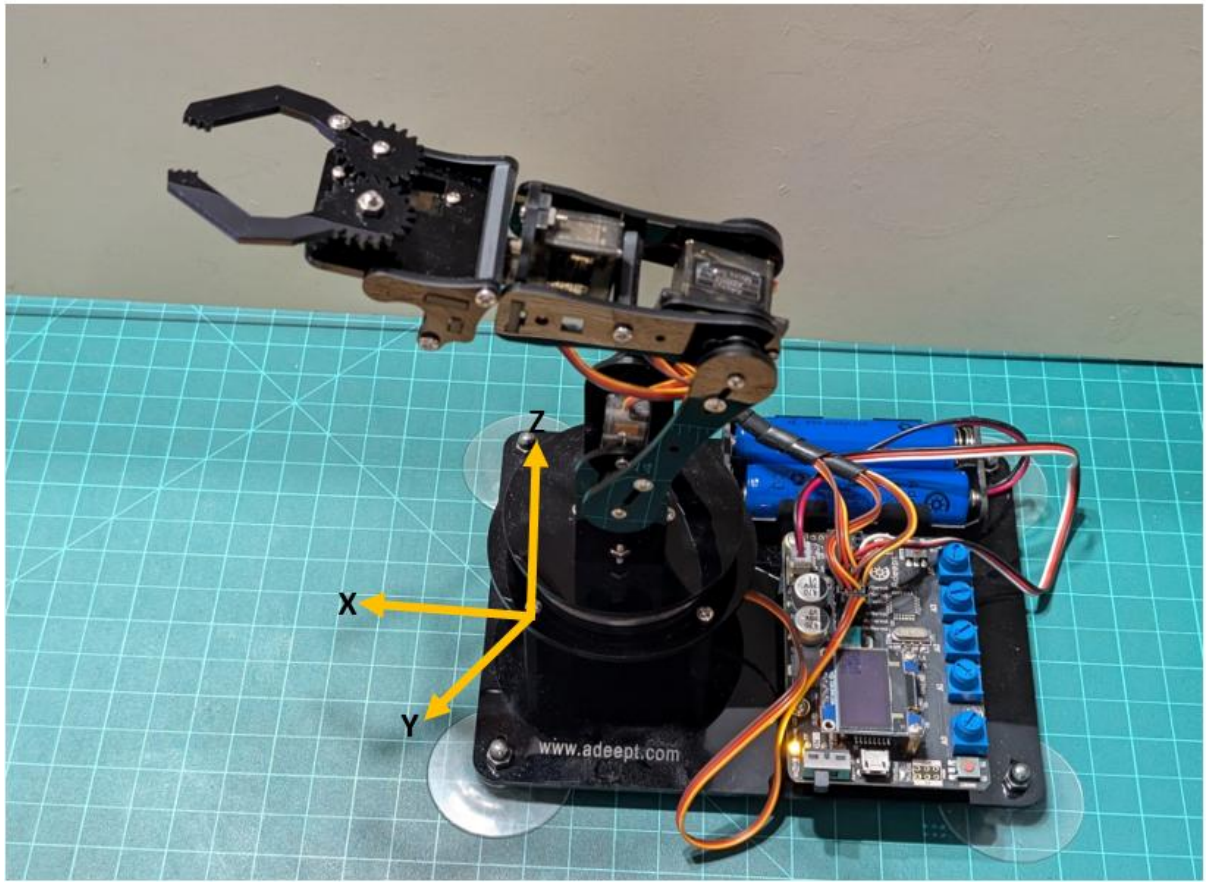
a) Third Hand
b) Left Hand
c) Right Hand
d) Two Hand

# Robot Coordinate Systems. . .

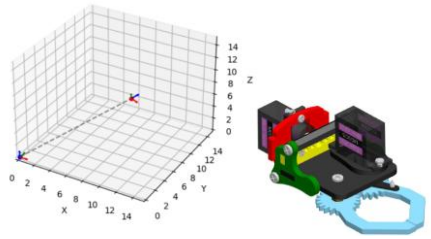


## Robot Base Coordinate System (World Frame)

**Right-Hand Rule:** Can aid in determining the World Frame



# Robot Coordinate Systems. . .



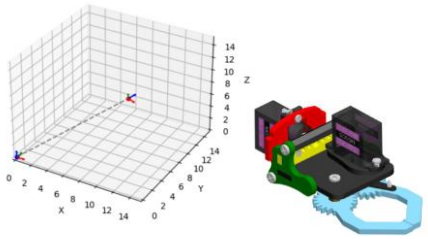
Joint Coordinate System

a) Each robotic joint has its coordinate system.

b) It is used in **joint space representation**, where the position of the robot is described in terms of joint angles rather than Cartesian coordinates.

c) This system is useful for defining the robot's kinematics and motion control.

# Robot Coordinate Systems. . .



Joint Coordinate System

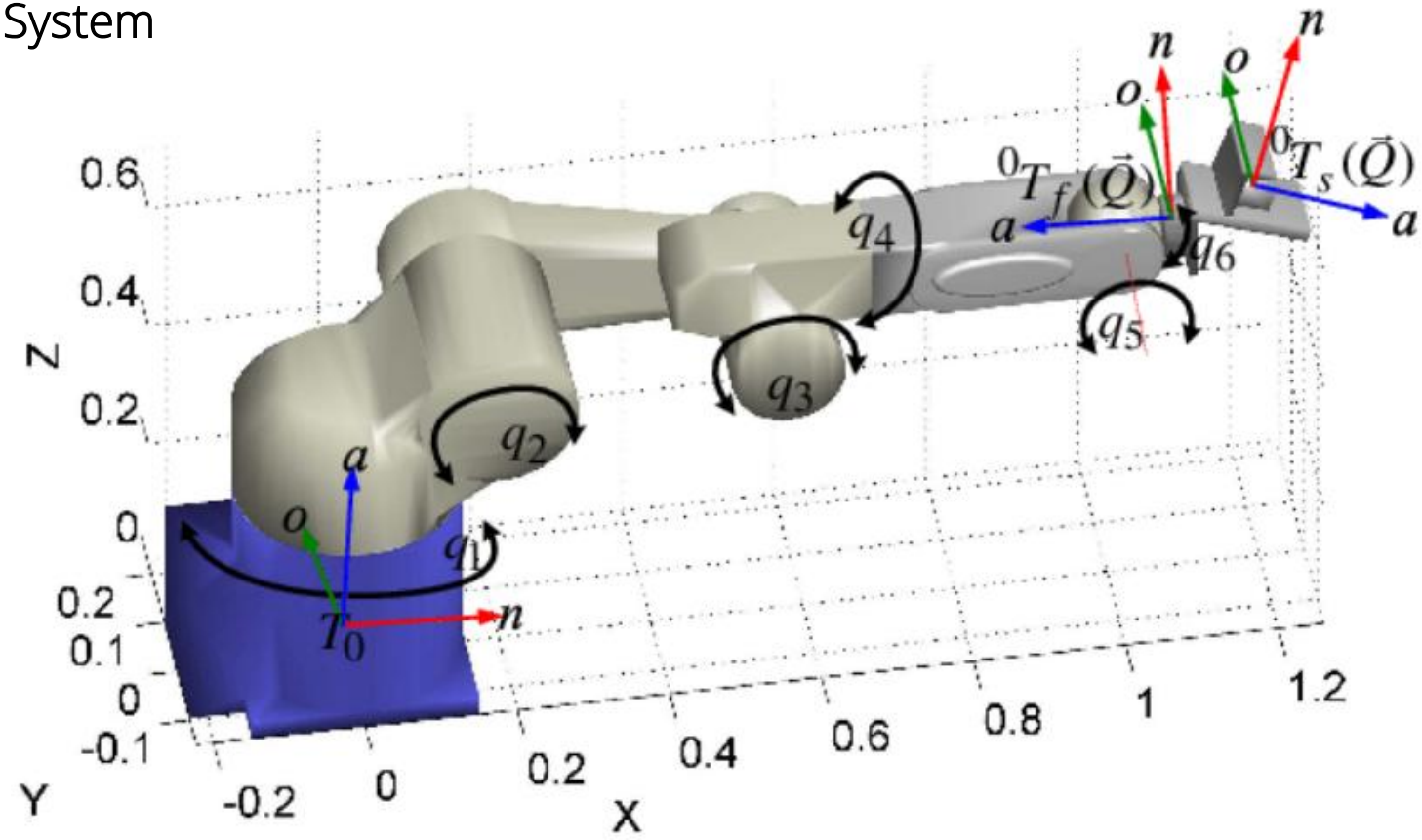
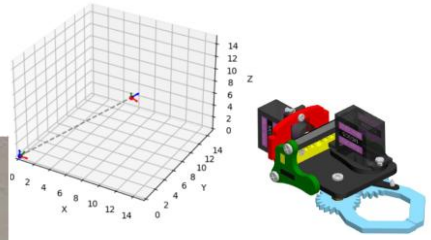
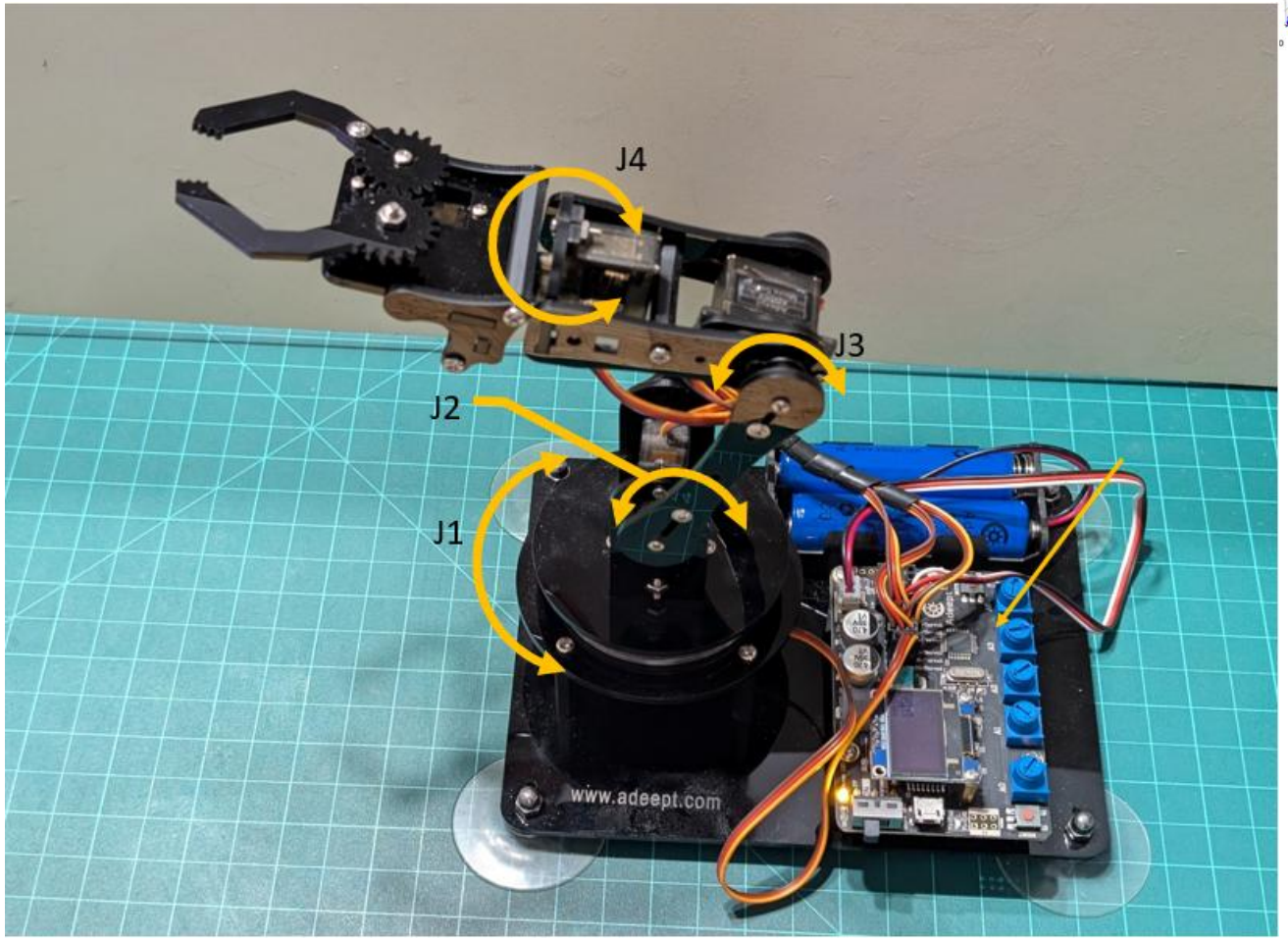



Image: ResearchGate

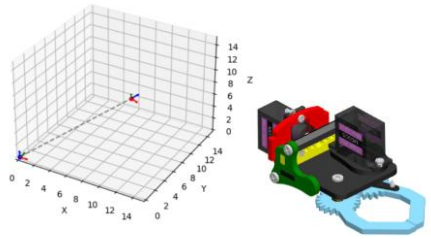
# Robot Coordinate Systems. . .



Joint Coordinate System:
Applied to Adeept Robotic Arm



# Robot Coordinate Systems



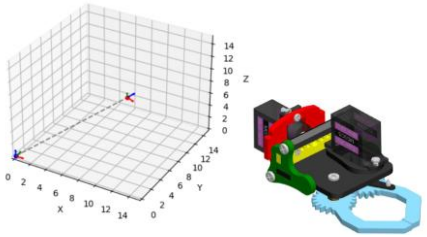
## Tool (End-Effector) Coordinate System

a) This system is attached to the robot's end-effector (e.g., a gripper or welding tool).

b) It defines the tool's position and orientation relative to the robot base.

c) Used for defining precise movements for tasks such as pick-and-place, welding, or assembly.

# Robot Coordinate Systems . . .



Tool (End-Effector) Coordinate System

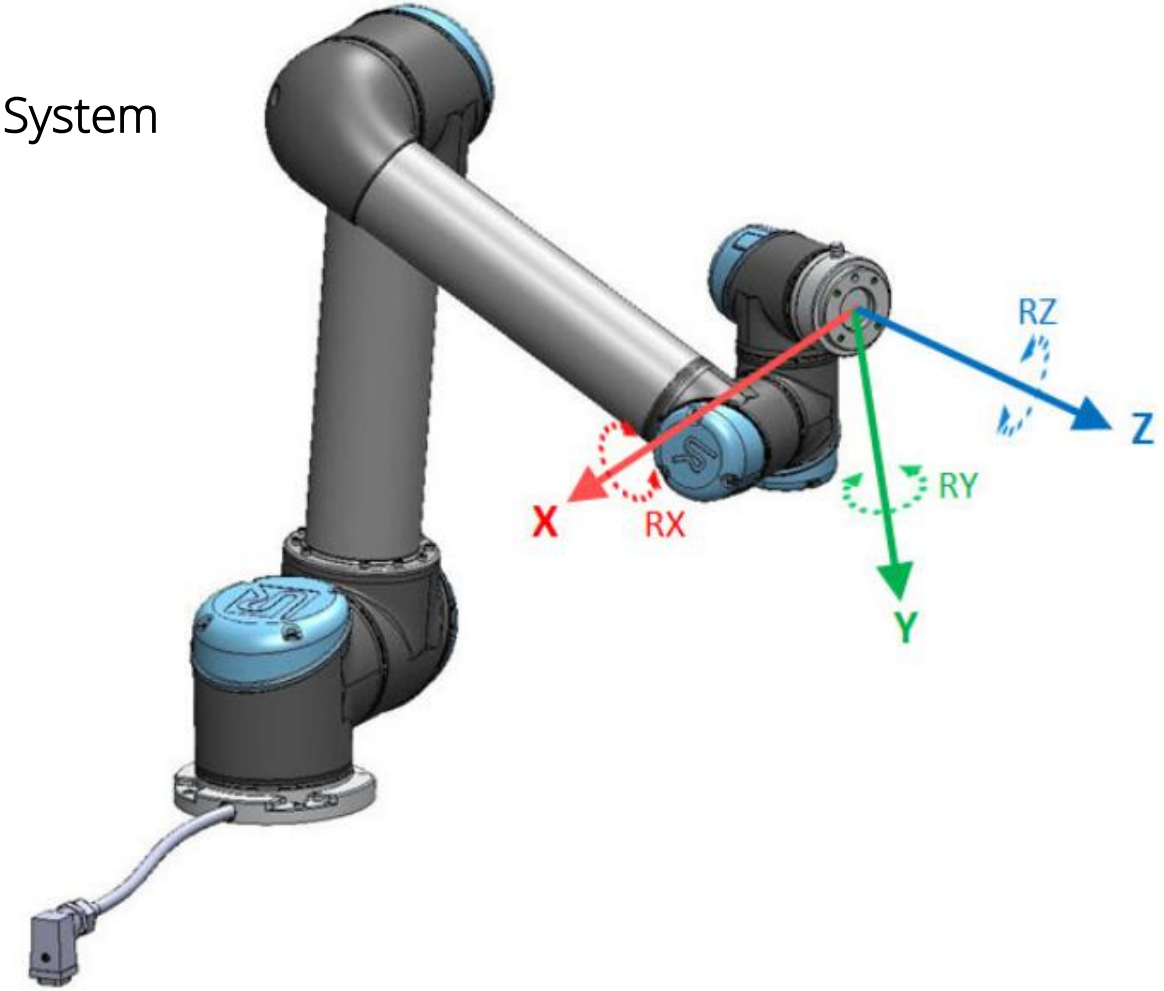
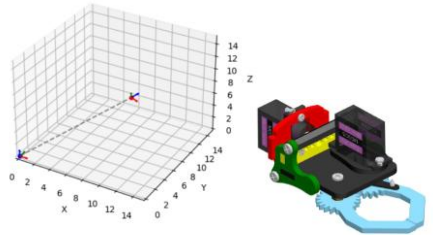
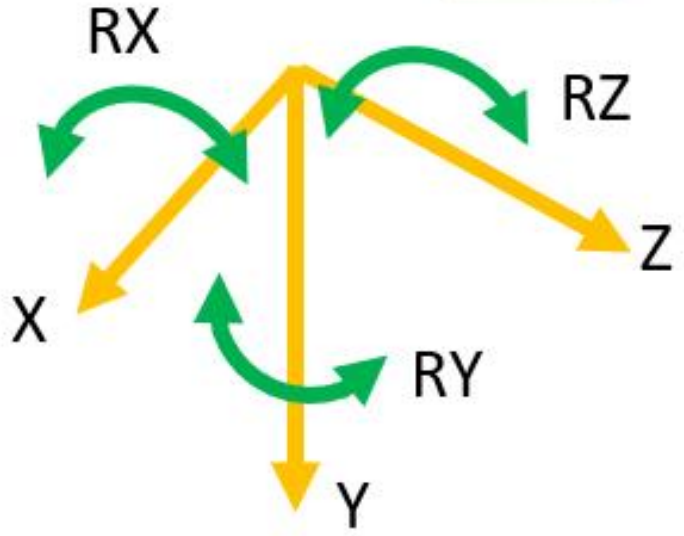



Image: Universal Robots

# Robot Coordinate Systems . . .



Tool (End-Effector) Coordinate System:
Applied to the Adeept End Effector

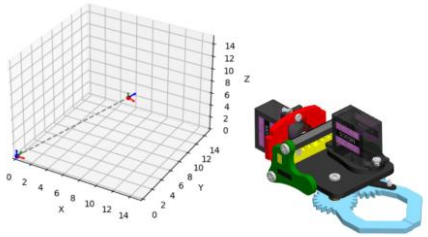


# Question 2

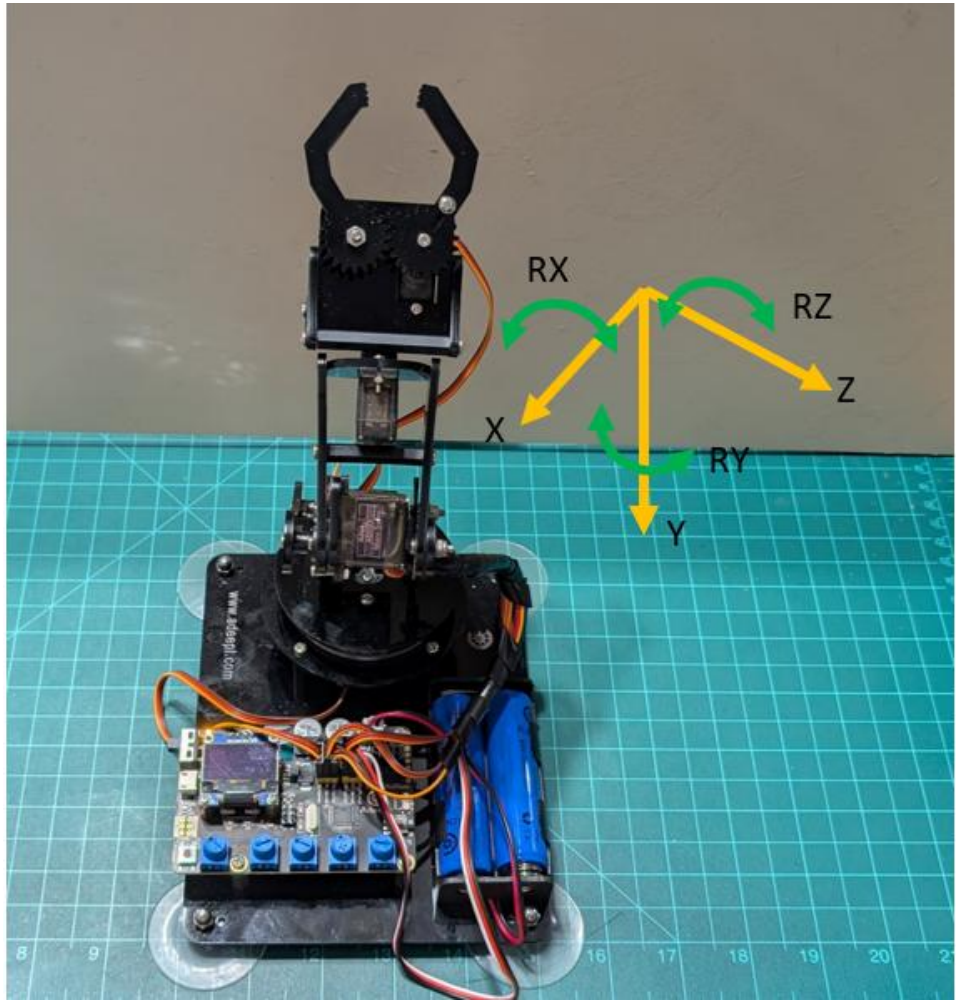
**Which coordinate system defines the tool position and orientation relative to the robot base?**

**a) Global**
**b) World**
**c) End Effector**
**d) none of the above**

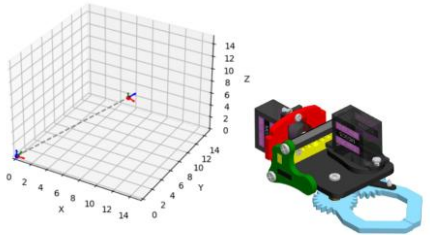
# Robot Coordinate Systems . . .



Tool (End-Effector) Coordinate System:
Applied to Adeept Robotic Arm



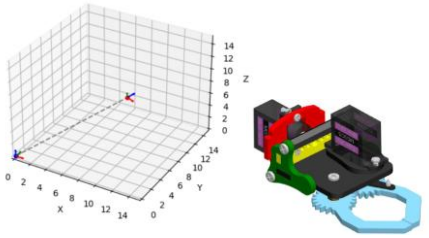
# The Point (P) Matrix



## Point Matrix in Robotics and Mathematics

A **Point Matrix** is a mathematical representation used to describe the position of a point in a coordinate system, often in **homogeneous coordinates**. In robotics, computer graphics, and engineering, point matrices help represent and transform points efficiently using matrix operations.

# The Point (P) Matrix. . .



## 1. Definition of a Point Matrix

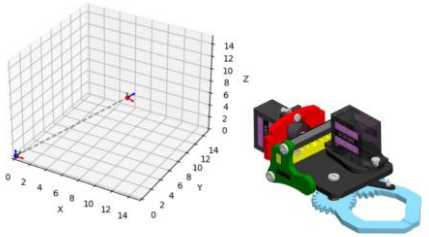
A **point matrix** represents the coordinates of a point in **3D space** as a column vector:

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

where:

- **x**, **y**, and **z** are the coordinates of the point in a Cartesian coordinate system.

# The Point (P) Matrix. . .



## 1. Definition of a Point Matrix

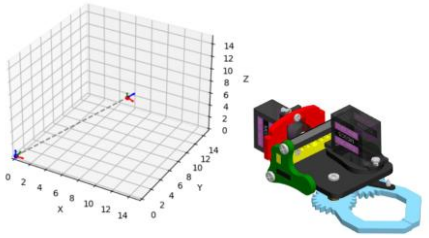
A **point matrix** represents the coordinates of a point in **3D space** as a column vector:

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

where:

- **x**, **y**, and **z** are the coordinates of the point in a Cartesian coordinate system.

# The Point (P) Matrix. . .



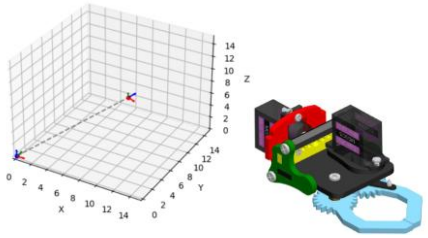
For **homogeneous coordinates**, a fourth component www is added:

$$P = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

where:

- w=1 for **position points**.
- w=0 for **direction vectors** (used for orientations).

# The Point (P) Matrix. . .



## 2. Usage of Point Matrices

### A. Representing Points in Space

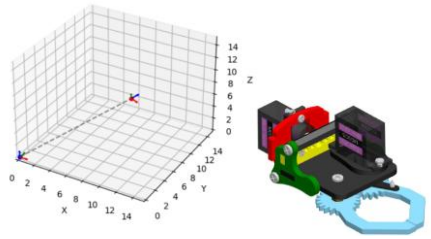
In robotics and kinematics, a **point matrix** represents a position in **3D space** relative to a specific coordinate frame.

### B. Transformation Using Matrices

Point matrices are used with **transformation matrices** to perform translations, rotations, and scaling.

For example, given a **homogeneous transformation matrix** T:

# Translation Matrix



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

where:

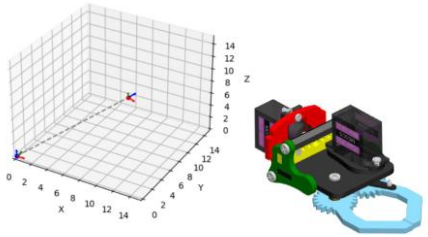
- R is the 3×3 rotation matrix.
- d is the 3×1 translation vector.

Applying T to a **point matrix** P:

$$P' = T \cdot P'$$

which gives the new transformed coordinates.

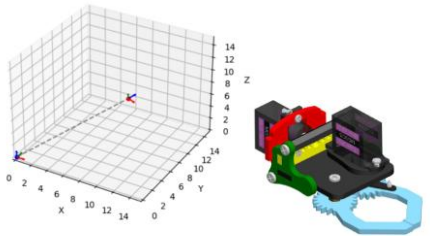
# Translation Matrix. . .



3. Example Calculation
Example 1: Translating a Point
Given the point:

$$P = \begin{bmatrix} 2 \\ 3 \\ 1 \\ 1 \end{bmatrix}$$

# Translation Matrix. . .



and a translation transformation:

$$T = \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Applying T·P:

Add these 2 columns

$$\begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2+5 \\ 3-2 \\ 1+3 \\ 1 \end{bmatrix} = \begin{bmatrix} 7 \\ 1 \\ 4 \\ 1 \end{bmatrix}$$

Coordinates for Translating a Point

Note:

**cos 90° = 0**

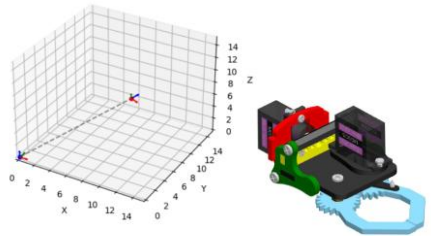
**sin 90° = 1**

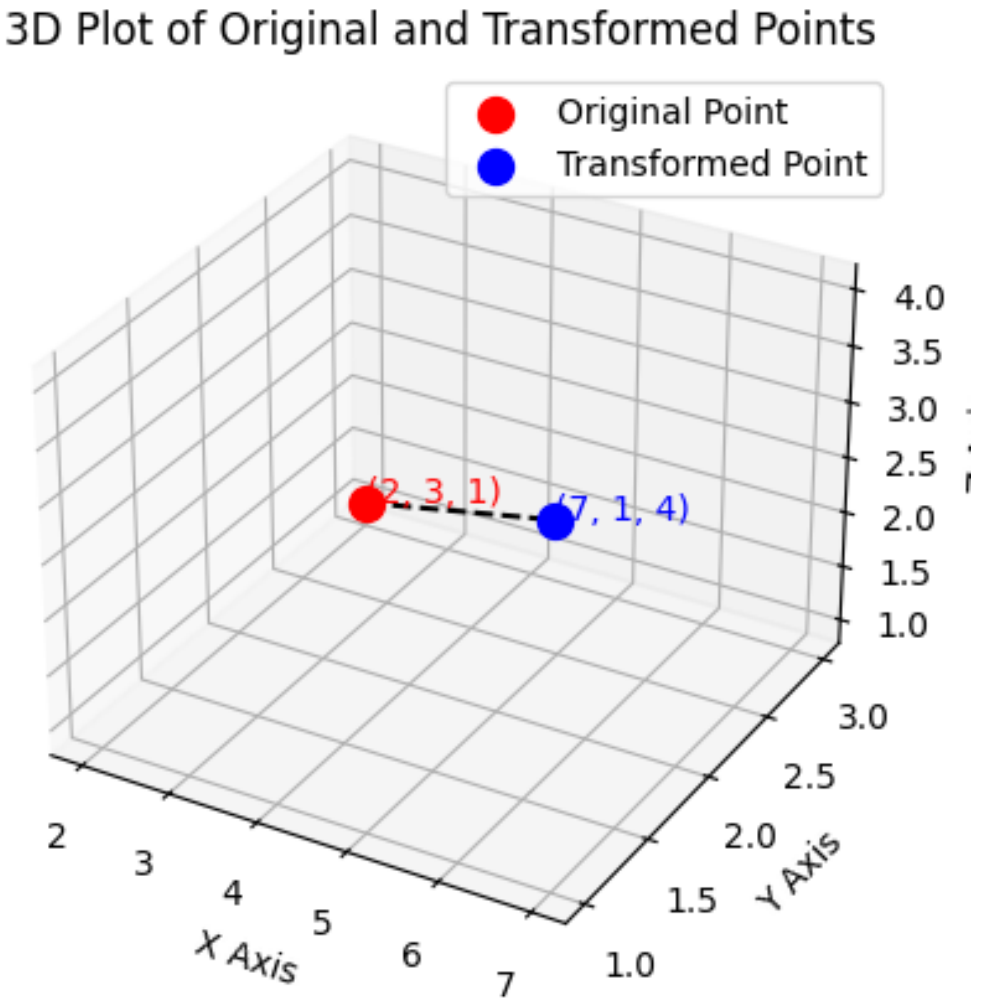
# Question 3

**What variable is used to define a 3x1 translation vector?**

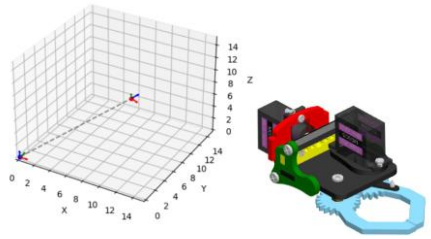
**a) t**
**b) r**
**c) d**
**d) s**

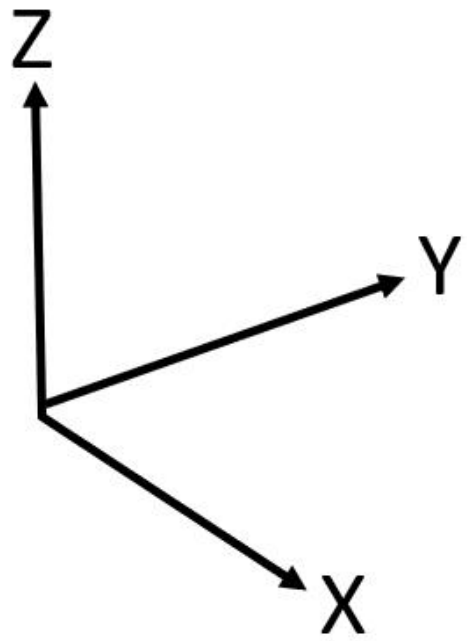
# Translation Matrix. . .

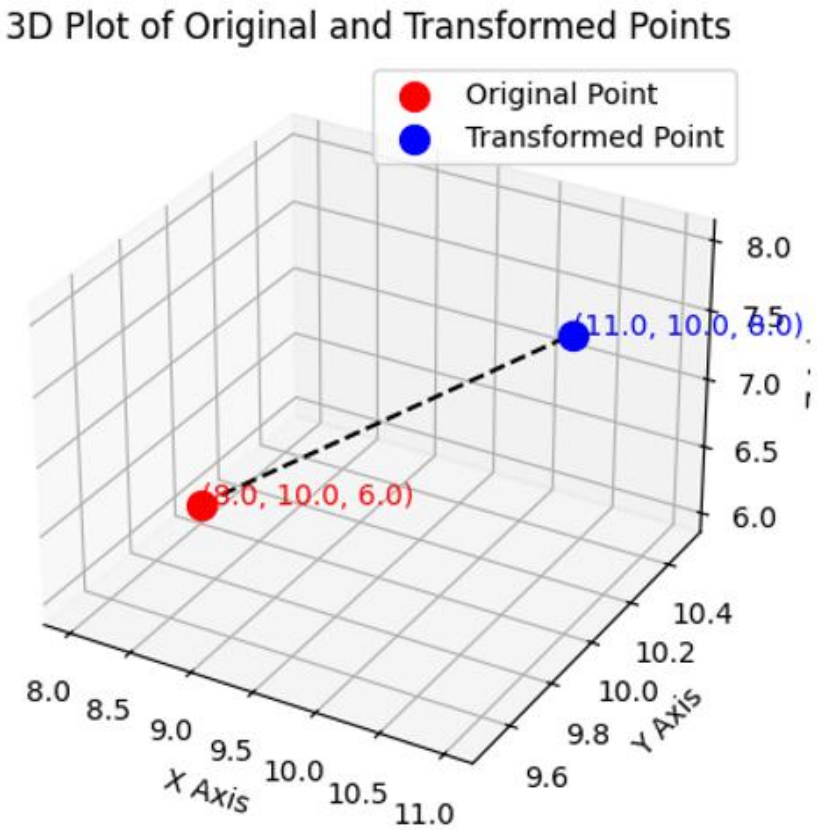




# Lab: Build A Translation Matrix in Python

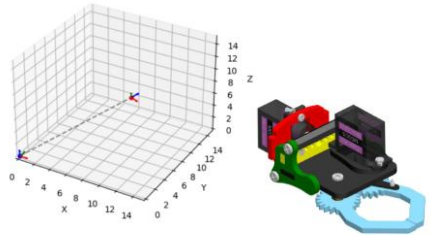




## Lab: Build A Translation Matrix in Python. .

.

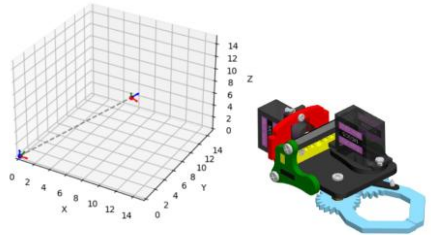


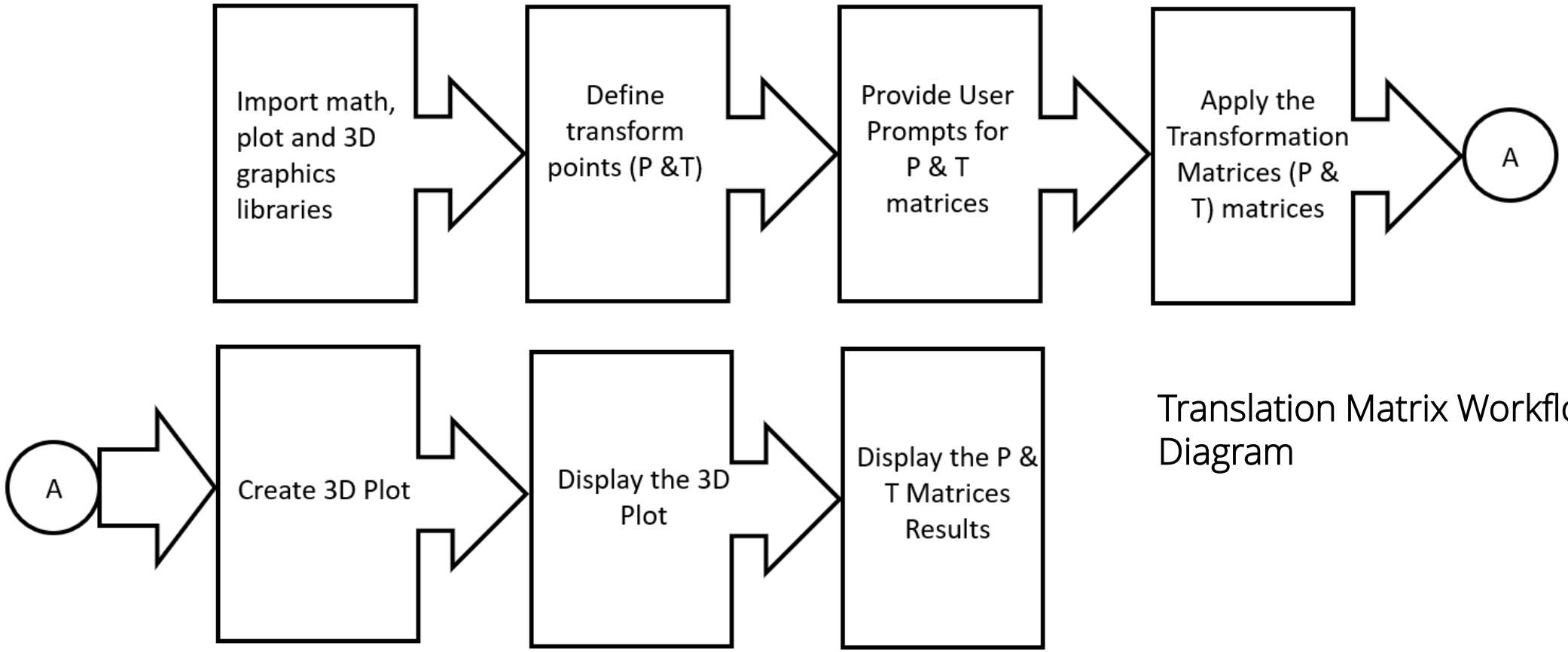
## Participant Learning Objectives:

- Participants will learn to build a Translation Matrix using a Workflow Diagram and Python programming language.
- Participants will learn to set up a Google Colaboratory Internet Notebook.
- Participants will learn to execute the Python Translation Matrix program in Google Colaboratory.

# Lab: Build A Translation Matrix in Python. .

.

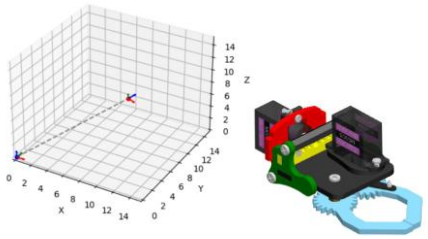





Translation Matrix Workflow Diagram

# Lab: Build A Translation Matrix in Python. .

.



Import math, plot and 3D graphics libraries

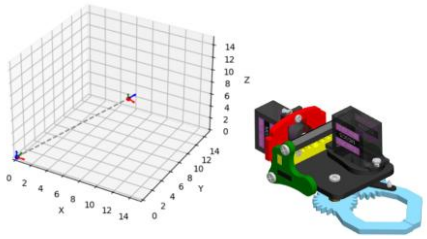
```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

Define transform points (P &T)

```python
def transform_point(T, P):
    """Apply a transformation matrix T to a point P using the dot product.
    Ensure that decimal values in T do not affect the Z transformation unexpectedly."""
    P_transformed = np.dot(T, P)

    # Explicitly ensure Z value is computed as expected
    P_transformed[2] = np.round(P_transformed[2], 6)  # Rounding to mitigate floating point errors

    return P_transformed
```

# Lab: Build A Translation Matrix in Python. .

.



Provide User Prompts for P & T matrices

```
15  # User input for the point
16  x = float(input("Enter x coordinate of the point: "))
17  y = float(input("Enter y coordinate of the point: "))
18  z = float(input("Enter z coordinate of the point: "))
19  w = float(input("Enter w coordinate of the point: "))  # Homogeneous coordinate
20  P = np.array([[x], [y], [z], [w]])  # (x, y, z, w)
21
22  # User input for the 4x4 transformation matrix
23  print("Enter the 4x4 transformation matrix row by row (decimal values allowed but should not affect Z result):")
24  T = np.array([[float(input(f"Row {i+1}, Col {j+1}: ")) for j in range(4)] for i in range(4)])
```

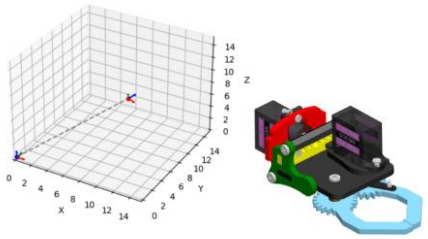
# Question 4

**In reviewing slide 37, which line of code builds the P-Matrix?**

**a) 24**
**b) 18**
**c) 16**
**d) 20**

# Lab: Build A Translation Matrix in Python. .

.

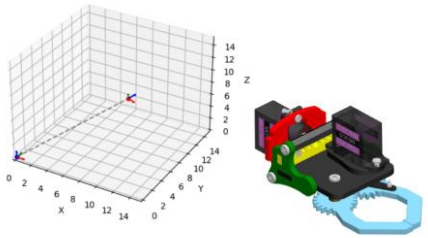


Apply the Transformation Matrices (P & T) matrices

```
26  # Apply the transformation
27  P_transformed = transform_point(T, P)
28
29  # Extract coordinates for plotting (ignoring the w coordinate since it's not part of 3D space)
30  x_vals = [P[0,0], P_transformed[0,0]]
31  y_vals = [P[1,0], P_transformed[1,0]]
32  z_vals = [P[2,0], P_transformed[2,0]]
```

# Lab: Build A Translation Matrix in Python. .

.



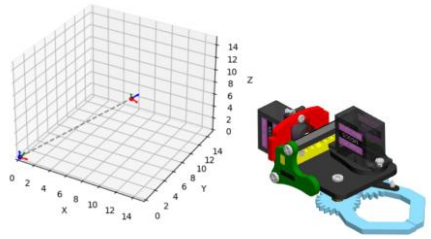
Create 3D Plot

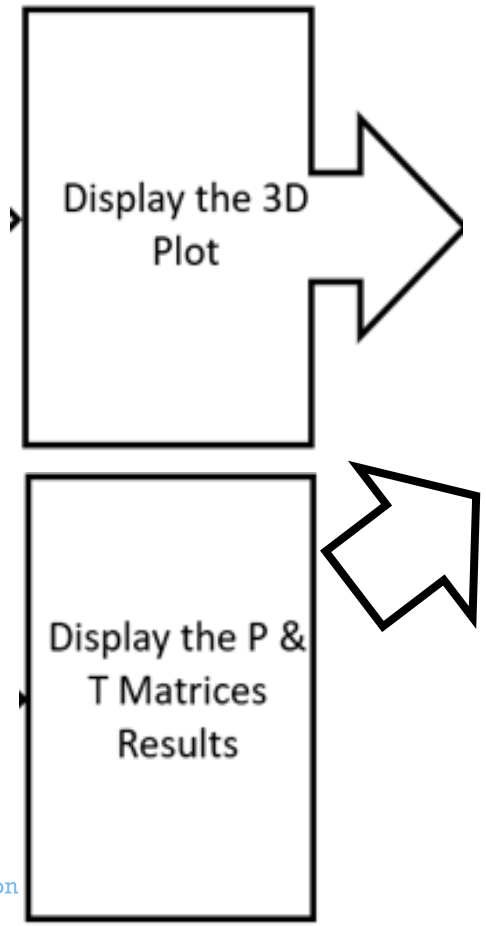
Display the 3D Plot

```
34  # Create a 3D plot
35  fig = plt.figure()
36  ax = fig.add_subplot(111, projection='3d')
37
38  # Plot original and transformed points with larger dots
39  ax.scatter(x_vals[0], y_vals[0], z_vals[0], color='r', s=100, label='Original Point')
40  ax.scatter(x_vals[1], y_vals[1], z_vals[1], color='b', s=100, label='Transformed Point')
41
42  # Draw a line connecting the points
43  ax.plot(x_vals, y_vals, z_vals, 'k--')
44
45  # Display coordinates as text near the points
46  ax.text(x_vals[0], y_vals[0], z_vals[0], f'({x_vals[0]}, {y_vals[0]}, {z_vals[0]})', color='red')
47  ax.text(x_vals[1], y_vals[1], z_vals[1], f'({x_vals[1]}, {y_vals[1]}, {z_vals[1]})', color='blue')
48
49  # Labels and title
50  ax.set_xlabel('X Axis')
51  ax.set_ylabel('Y Axis')
52  ax.set_zlabel('Z Axis')
53  ax.set_title('3D Plot of Original and Transformed Points')
54  ax.legend()
```

# Lab: Build A Translation Matrix in Python. .

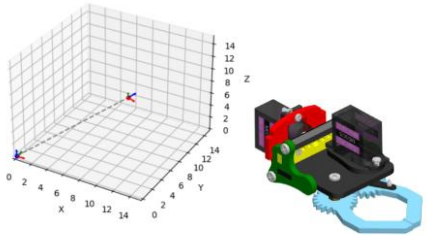
.






```
56  # Show the plot
57  plt.show()
58
59  # Display results
60  print("Original Point:\n", P)
61  print("Transformation Matrix:\n", T)
62  print("Transformed Point:\n", P_transformed)
63
```

# Lab: Build A Transformation Matrix in Python. . .



a) Click the “+Code” button then paste code into the cell.
b) Click the “Arrow” (Run) Icon to execute the code.

Design_New_CEC_Point_Matrix_Transform.ipynb

File Edit View Insert Runtime Tools Help

Share

Commands + Code + Text

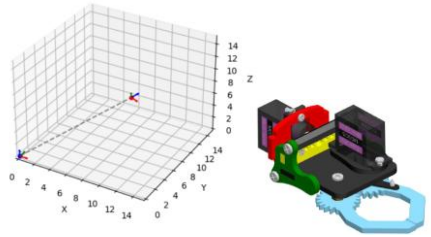
RAM Disk

```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

def transform_point(T, P):
    """Apply a transformation matrix T to a point P using the dot product.
    Ensure that decimal values in T do not affect the Z transformation unexpectedly."""
    P_transformed = np.dot(T, P)

    # Explicitly ensure Z value is computed as expected
    P_transformed[2] = np.round(P_transformed[2], 6)  # Rounding to mitigate floating point errors

    return P_transformed

# User input for the point
x = float(input("Enter x coordinate of the point: "))
y = float(input("Enter y coordinate of the point: "))
z = float(input("Enter z coordinate of the point: "))
w = float(input("Enter w coordinate of the point: "))  # Homogeneous coordinate
P = np.array([[x], [y], [z], [w]])  # (x, y, z, w)

# User input for the 4x4 transformation matrix
print("Enter the 4x4 transformation matrix row by row (decimal values allowed but should not affect Z result):")
T = np.array([[float(input(f"Row {i+1}, Col {j+1}: ")) for j in range(4)] for i in range(4)])

# Apply the transformation
P_transformed = transform_point(T, P)

```

Google Colaboratory Code Development Environment

# Lab: Build A Translation Matrix in Python. .

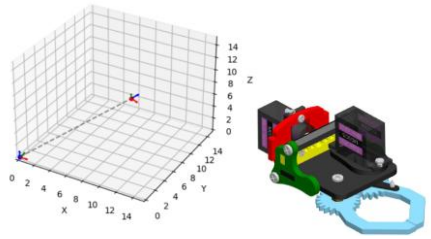
.



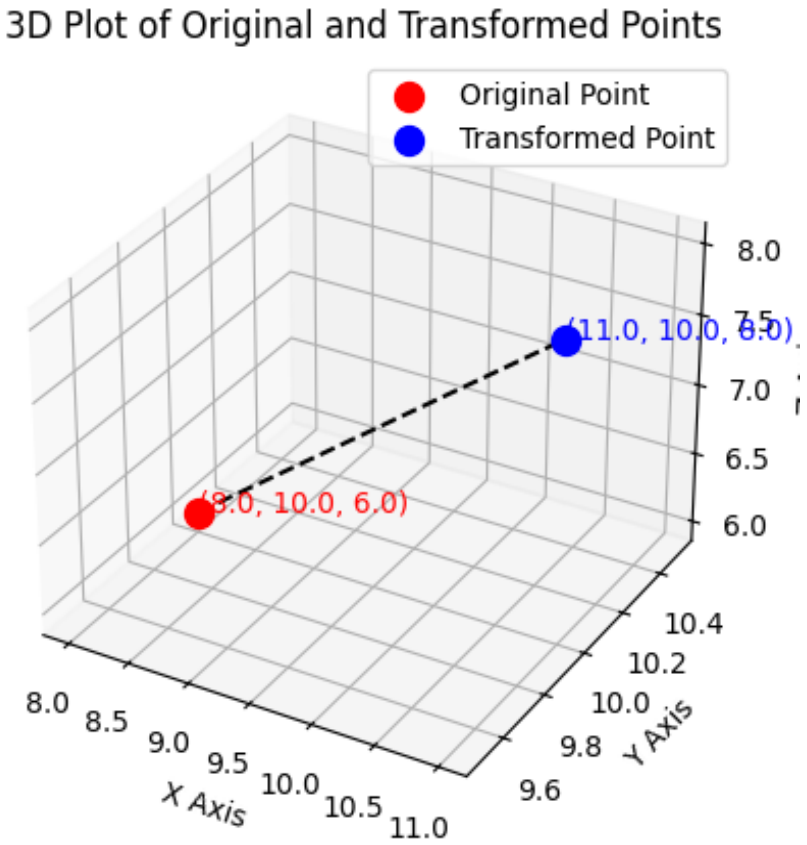
```
Enter x coordinate of the point: 8
Enter y coordinate of the point: 10
Enter z coordinate of the point: 6
Enter w coordinate of the point: 1
Enter the 4x4 transformation matrix row by row (decimal values allowed but should not affect Z result):
Row 1, Col 1: 1
Row 1, Col 2: 0
Row 1, Col 3: 0
Row 1, Col 4: 3
Row 2, Col 1: 0
Row 2, Col 2: 1
Row 2, Col 3: 0
Row 2, Col 4: 0
Row 3, Col 1: 0
Row 3, Col 2: 0
Row 3, Col 3: 1
Row 3, Col 4: 2
Row 4, Col 1: 0
Row 4, Col 2: 0
Row 4, Col 3: 0
```

Active –Interaction Session: Building the P and T Matrices

# Lab: Build A Translation Matrix in Python. . .



```
Original Point:
 [[ 8.]
 [10.]
 [ 6.]
 [ 1.]]
Transformation Matrix:
 [[1. 0. 0. 3.]
 [0. 1. 0. 0.]
 [0. 0. 1. 2.]
 [0. 0. 0. 1.]]
Transformed Point:
 [[11.]
 [10.]
 [ 8.]
 [ 1.]]
```




Results: P & T Matrices and Translation Point in 3D Plot

# Question 5

**In reviewing slide 44, which translated point is equal to the original point of 6?**

**a) 11**
**b) 10**
**c) 6**
**d) 8**

# Thank you for attending

Please consider the resources below:

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

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