# (EEEE3024 UNUK)

# Robotics, Dynamics, and Control

## Coursework 2: Exploration of Forward and Inverse Kinematic Using the DOBOT Magician Robotic Arm

| | |
|---|---|
| **STUDENT NAME:** | **Kexin Yu** |
| **ID NUMBER:** | **20320941** |
| **DATE:** | **09/05/2024** |

## 1. Present a possible set of inverse kinematic equations

A geometric approach was used in CW1 to solve the inverse kinematics equations of Dobot, and Eqn1 shows the formulas for calculating different angles, where $d_1$(135mm), $a_2$(138mm), and $a_3$(145mm) are the lengths of the different robotic arms respectively:

$$\begin{cases} \theta_1 = arctan\left(\frac{Y_p}{X_p}\right) \\ \theta_2 = 180^\circ - arctan\left(\frac{X_p}{cos(\theta_1)(d_1 - Z_p)}\right) - arccos\left(\frac{{a_2}^2 + (d_1 - Z_p)^2 + (\frac{Xp}{cos(\theta_1)})^2 - {a_3}^2}{2a_2\sqrt{(d_1 - Z_p)^2 + (\frac{Xp}{cos\,(\theta_1)})^2}}\right) \\ \theta_3 = 90^\circ - arccos\left(\frac{{a_2}^2 + {a_3}^2 - ((d_1 - Z_p)^2 + (\frac{Xp}{cos(\theta_1)})^2)}{2a_2a_3}\right) \end{cases} \qquad \textbf{Eqn 1}$$

## 2. Using the DOBOT Studio software, create a routine to pick and place an object within the workspace using the graphical programming environment.

Figures 1 (side view) and 2 (top view) show the key positional information of the Dobot's robotic arm used to calculate the coordinate settings for object pickup. The origin of the robotic arm is located 138mm up from the centre of the base, which is the reference height for controlling the end-effector.

**X-coordinate:** the object is located 120mm in front of the base of the arm. add the 79mm from the centre of the base to the edge makes the X-coordinate of the centre of the object 199mm.

**Y coordinate:** from the top view, the centre of the object is 40mm to the right of the base, so the end-effector needs to be moved 40mm to the right, setting the Y coordinate to -40mm.

**Z coordinate:** The default height of the end-effector is 138mm, considering the distance from the bottom of the actuator to the top of the suction cup is 13mm, plus the length of the suction cup is 60mm, and the thickness of the object is 2mm, the end-effector needs to be lowered by 63mm (138mm - 13mm - 60mm - 2mm = 63mm) in order to stably touch the top of the object. Therefore, the Z coordinate is set to -63mm. In practice, a Z coordinate of -64mm works equally well, either due to a certain tolerance range between the suction cup and the surface of the object, or due to small operational differences in the actuator. Suction cup designs often allow some flexibility to accommodate surfaces that are not perfectly flat or mechanical errors. Therefore, a Z-coordinate setting of either -63mm or -64mm ensures that the end-effector can safely and accurately contact the surface of the object.

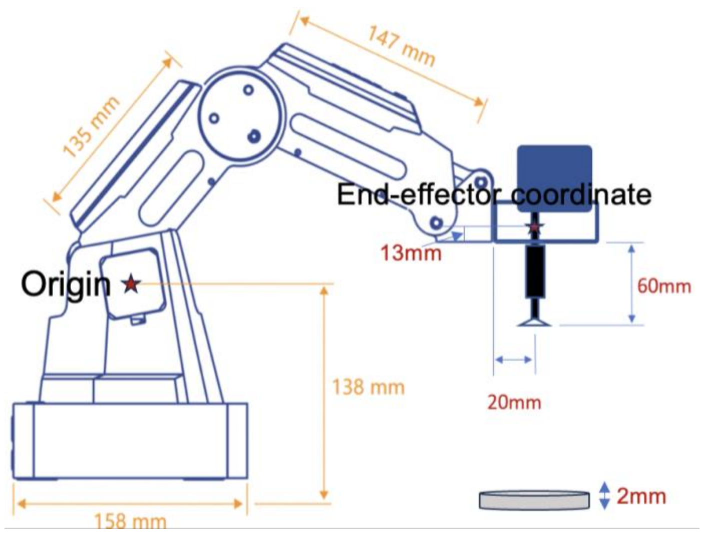


**Figure1:** Side view of Dobot

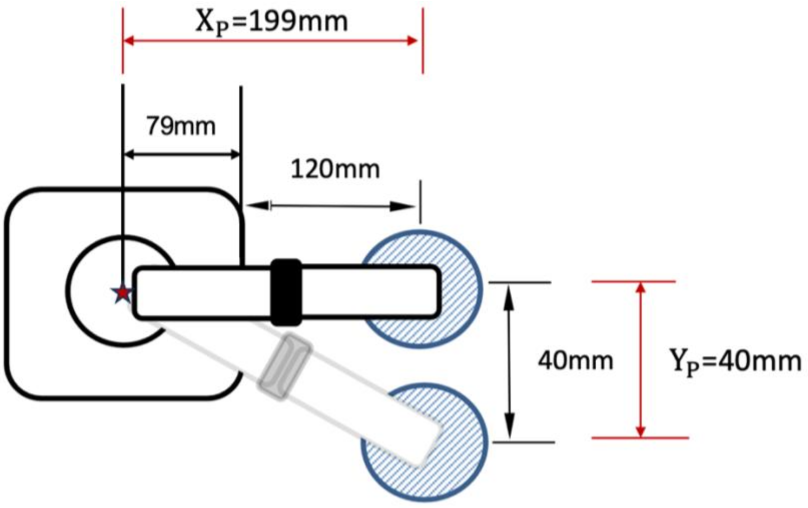
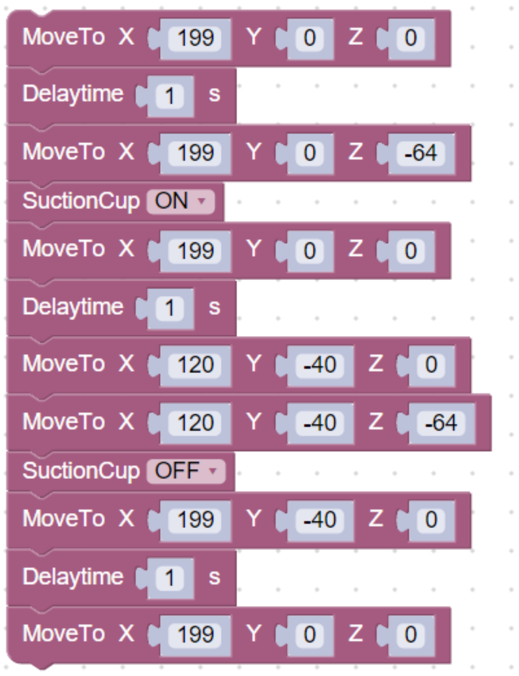


**Figure 2:** Top view of Dobot

Figure 3 shows the graphical blockly designed to allow the robotic arm to complete the pickup target based on the above calculations. The robotic arm was first instructed to move to the position X=199mm, Y=0mm, Z=0mm, which is usually the starting position directly above the object. Once the arm arrives, the program sets a 1 second delay to ensure stability. Next, the arm moves down to Z=-64mm to approach the object to be picked up. After activating the suction cups, the object was adsorbed. The arm was then raised back to its initial height of Z=0mm and paused again for 1 second to ensure stability. The arm then moved to a new position X=120mm, Y=-40mm, Z=0mm for object placement. A negative Y coordinate indicates that the arm is moving to the right side of the mechanical base. At the placement position, the arm drops to Z=-64mm and closes the suction cup to drop the object. After completing the placement, the robot arm returns to position X=199mm, Y=-40mm, Z=0mm with a 1 second delay. Finally, the arm returns to its original starting position of X=199mm, Y=0mm, Z=0mm ready for the next operation.



```
MoveTo X 199 Y 0 Z 0
Delaytime 1 s
MoveTo X 199 Y 0 Z -64
SuctionCup ON
MoveTo X 199 Y 0 Z 0
Delaytime 1 s
MoveTo X 120 Y -40 Z 0
MoveTo X 120 Y -40 Z -64
SuctionCup OFF
MoveTo X 199 Y -40 Z 0
Delaytime 1 s
MoveTo X 199 Y 0 Z 0
```

**Figure 3:** Graphical programming environment Blockly

The angles rotated by the three joints according to the completion of the task to each position are shown in Table 1 to validate Question 3.

**Table 1:** Rotation angles of joint1, 2, 3 at different positions

| X | Y | Z | $\theta_{J1}$ | $\theta_{J2}$ | $\theta_{J3}$ |
|---|---|---|---|---|---|
| 208 | 0 | 135 | 0° | 0° | 0° |
| 199 | 0 | 0 | 0° | 25.211° | 56.1902° |
| 199 | 0 | -64 | 0° | 53.755° | 78.0558° |
| 199 | -40 | 0 | -11.3653° | 26.3078° | 55.4113° |
| 199 | -40 | -64 | -11.3653° | 54.1574° | 76.6887° |

## 3. Develop an approach to solve the set of inverse kinematic equations developed in CW1, and apply them to task 2 to validate these equations

The coordinate system setup and associated angle calculations for a Dobot robotic arm are shown in Figure 4. In this system, the original coordinate system setup and angle calculations did not adequately account for the effect of end-effector addition on the actual coordinate points and the change in the origin position itself. The purple markers in the figure are used to indicate the coordinate corrections due to the addition of the End Effector and the change in the origin. These markers help explain the new variables in the calculation and the corrected coordinate positions. The following is a strategy for adjusting the angle calculation so that the new coordinates also apply to the use of Eqn1 (Inverse equations developed in CW1).

$$\begin{cases} \theta_1 = arctan\left(\frac{Y_p}{X_p}\right) \\ \theta_2 = 180^\circ - arctan\left(\frac{X_p}{cos(\theta_1)(d_1-Z_p)}\right) - arccos\left(\frac{a_2{}^2+(d_1-Z_p)^2+(\frac{X_p}{cos(\theta_1)})^2-a_3{}^2}{2a_2\sqrt{(d_1-Z_p)^2+(\frac{X_p}{cos\,(\theta_1)})^2}}\right) \\ \theta_3 = 90^\circ - arccos\left(\frac{a_2{}^2+a_3{}^2-((d_1-Z_p)^2+(\frac{X_p}{cos(\theta_1)})^2)}{2a_2a_3}\right) \end{cases}$$

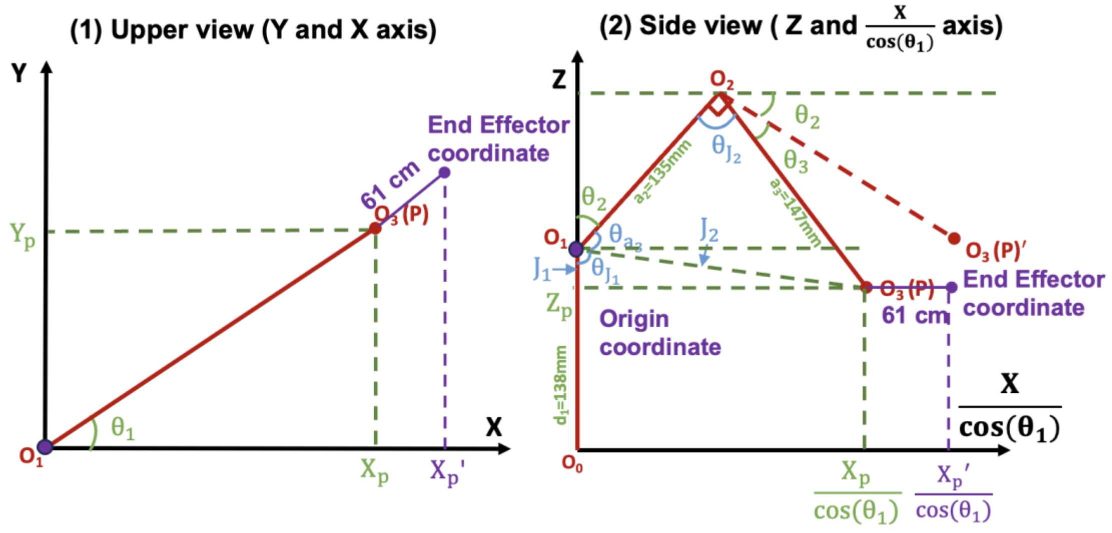


**Figure 4:** Coordinate system and angle calculation diagram for actuator at the end of the robot arm

**Calculation of Joint 1 rotation angle $\theta_1$:**

$\theta_1$ considers the effect of the end-effector on the X and Y axes, but since this effect is planar (XY plane), the original formula still applies, and nothing needs to be changed. The angle is obtained by calculating the angle between the origin to the projection of the end-effector on the XY plane and the X axis.

**Calculation of Joint 2 rotation angle $\theta_2$ and Joint 3 rotation angle $\theta_3$:**

The calculation of $\theta_2$ and $\theta_3$ involves a change in the z-axis and x-axis due to the addition of the end-effector and the upward shift of the origin coordinates. The specific calculation changes are as follows:

**Change in X-axis:** After the end-effector is added, the X-axis coordinates are shifted from $X_p$ to $X_p'$ , and this change is corrected by the Eqn 2, which considers the effect of the length of the end-effector.

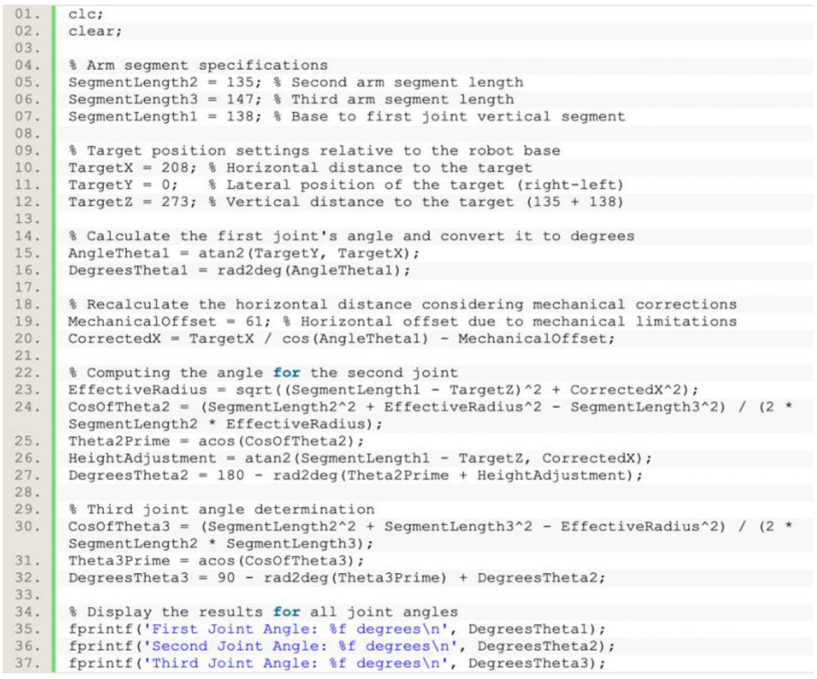
**Change in Z-axis:** The original Z-axis coordinates were adjusted by 138 mm to conform to the new starting point, this change was corrected by Eqn 3, where $Z_p$ is the original z-axis coordinate, and $Z_p'$ is the changed one.

$$\frac{X_p}{\cos(\theta_1)} = \frac{X_p'}{\cos(\theta_1)} - 61 \qquad \textbf{Eqn 2}$$

$$Z_p = Z_p' + 138 \qquad \textbf{Eqn 3}$$

The coordinates obtained from the Dobot robotic arm can still be adapted to Eqn 1 after the above adjustments to the x- and z-axes. It should also be noted that the $\theta_{J3}$ shown in the dobot studio panel is not $\theta_3$ as shown in the figure, but rather $\theta_2 + \theta_3$, which is listed in Eqn4. Based on these formulae adjustments, MATLAB code was used to validate the computation of the coordinates of the arrivals in Problem 2. Figure 5 shows the MATLAB code used to calculate the angles, and the results of the calculations with their corresponding coordinates are shown in Table 2.

$$\theta_{J3} = \theta_2 + \theta_3 \qquad \textbf{Eqn 4}$$



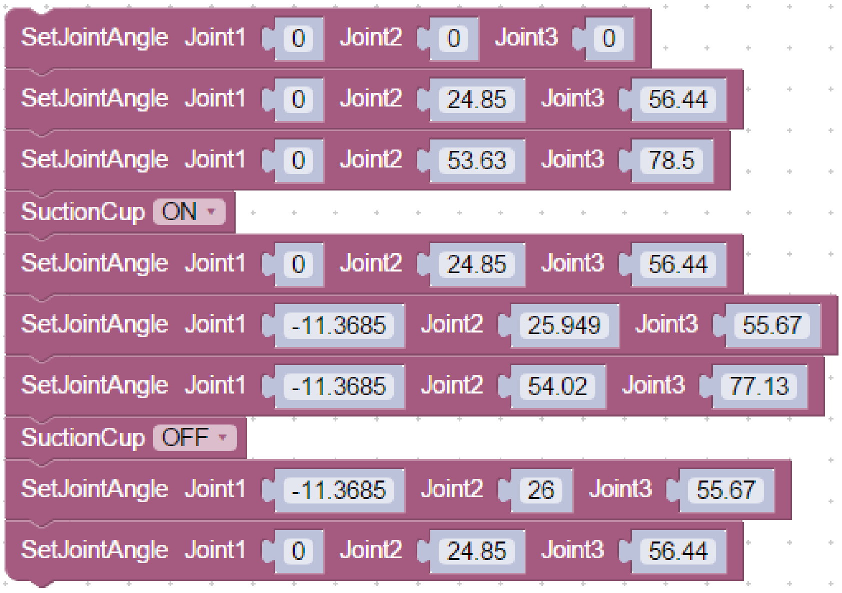
```matlab
clc;
clear;

% Arm segment specifications
SegmentLength2 = 135; % Second arm segment length
SegmentLength3 = 147; % Third arm segment length
SegmentLength1 = 138; % Base to first joint vertical segment

% Target position settings relative to the robot base
TargetX = 208; % Horizontal distance to the target
TargetY = 0;   % Lateral position of the target (right-left)
TargetZ = 273; % Vertical distance to the target (135 + 138)

% Calculate the first joint's angle and convert it to degrees
AngleTheta1 = atan2(TargetY, TargetX);
DegreesTheta1 = rad2deg(AngleTheta1);

% Recalculate the horizontal distance considering mechanical corrections
MechanicalOffset = 61; % Horizontal offset due to mechanical limitations
CorrectedX = TargetX / cos(AngleTheta1) - MechanicalOffset;

% Computing the angle for the second joint
EffectiveRadius = sqrt((SegmentLength1 - TargetZ)^2 + CorrectedX^2);
CosOfTheta2 = (SegmentLength2^2 + EffectiveRadius^2 - SegmentLength3^2) / (2 *
SegmentLength2 * EffectiveRadius);
Theta2Prime = acos(CosOfTheta2);
HeightAdjustment = atan2(SegmentLength1 - TargetZ, CorrectedX);
DegreesTheta2 = 180 - rad2deg(Theta2Prime + HeightAdjustment);

% Third joint angle determination
CosOfTheta3 = (SegmentLength2^2 + SegmentLength3^2 - EffectiveRadius^2) / (2 *
SegmentLength2 * SegmentLength3);
Theta3Prime = acos(CosOfTheta3);
DegreesTheta3 = 90 - rad2deg(Theta3Prime) + DegreesTheta2;

% Display the results for all joint angles
fprintf('First Joint Angle: %f degrees\n', DegreesTheta1);
fprintf('Second Joint Angle: %f degrees\n', DegreesTheta2);
fprintf('Third Joint Angle: %f degrees\n', DegreesTheta3);
```

**Figure 5:** Matlab code for calculating angles

**Table 2:** Calculation of the required rotation angles for three joints at different positions using MATLAB

| X | Y | Z | $\theta_1$ | $\theta_2$ | $\theta_2 + \theta_3$ |
|---|---|---|---|---|---|
| 208 | 0 | 135 | 0° | 0° | 0° |
| 199 | 0 | 0 | 0° | 24.85° | 56.44° |
| 199 | 0 | -64 | 0° | 53.63° | 78.50° |
| 199 | -40 | 0 | -11.3653° | 25.949° | 55.67° |
| 199 | -40 | -64 | -11.3653° | 54.02° | 77.13° |

The results in Tables 2 and 1 are almost identical except for some minor differences. To further verify that the calculation is accurate, Figure 6 shows the blockly of setting the calculated angles for the Dobot to perform the task in Question 2, it turned out that the object can still be picked up and put down at the designated location.



```
SetJointAngle Joint1 0 Joint2 0 Joint3 0
SetJointAngle Joint1 0 Joint2 24.85 Joint3 56.44
SetJointAngle Joint1 0 Joint2 53.63 Joint3 78.5
SuctionCup ON
SetJointAngle Joint1 0 Joint2 24.85 Joint3 56.44
SetJointAngle Joint1 -11.3685 Joint2 25.949 Joint3 55.67
SetJointAngle Joint1 -11.3685 Joint2 54.02 Joint3 77.13
SuctionCup OFF
SetJointAngle Joint1 -11.3685 Joint2 26 Joint3 55.67
SetJointAngle Joint1 0 Joint2 24.85 Joint3 56.44
```

**Figure 6:** Blockly used to validate the calculated angles

## 4. Understand the Tower of Hanoi puzzle – write pseudocode to plana path of picking and placing the discs from their initial 'rod' to target 'rod' avoiding obstructions in the way.

Figure 7 illustrates the design of this pseudo-code.

```
# Initialization
Set the total number of disks to n
Define the poles: Source Pole A, Auxiliary Pole B, Target Pole C
Define the minimum lifting height as minHeight necessary to clear any obstacles

# Solving Tower of Hanoi steps
For each disk from smallest (1) to largest (n):
    Calculate the number of moves required: move_count = $2^i - 1$

    For each move in move_count:
        Determine source and destination poles based on the move sequence:
            If (j % 3) == 1:
                move_from = Source Pole A
                move_to = Target Pole C
            If (j % 3) == 2:
                move_from = Source Pole A
                move_to = Auxiliary Pole B
            If (j % 3) == 0:
                move_from = Auxiliary Pole B
                move_to = Target Pole C

 # Detailed movement actions
  Elevate the disk from move_from to a height above minHeight to avoid collision
  Translate the disk horizontally from move_from to directly above move_to
  Descend the disk and place it securely on move_to
```
**Figure 7**: Pseudo-code for solving the Tower of Hanoi problem:

In the initialization phase of the Tower of Hanoi problem, the total number of discs to be manipulated is first determined, denoted by the variable n. The problem is solved by defining several rods. Next, define three poles, source pole A, auxiliary pole B, and target pole C. Source pole A is the starting location of the discs, where all discs are stacked in order of size, from smallest to largest; auxiliary pole B is used as a temporary storage location for the discs during the problem-solving process; and target pole C is the destination of the discs. In addition, a minimum lift height, minHeight, is set, which is high enough to prevent the discs from hitting other poles when moving from one pole to another, ensuring a safe and smooth movement. These preparatory steps provide the necessary basic settings for subsequent disk movements.

In solving the Tower of Hanoi problem, the number of moves of each disk is determined by the Eqn5, where i is the number of the disk, from 1 (the smallest disk) to n (largest disk). This formula determines the minimum number of moving steps required from the source rod to the target rod, via the auxiliary rod.

$$\text{move\_count} = 2^i - 1 \qquad \textbf{Eqn 5}$$

In the move logic, the direction of movement of each disk is determined by calculating the result of j % 3, where j represents the total number of moves “move_count” from 1 to that particular disk. Specifically, if the result is 1, the disk will move from the source pole A to the target pole C. If the result is 2, the disk will move from the source pole A to the auxiliary pole B. If the result is 0, this means that the disk should move from the auxiliary pole B to target pole C. This pattern ensures that each move conforms to the rules of the Tower of Hanoi problem, effectively utilizing the auxiliary poles to assist in the proper sequencing and movement of the discs.

Finally, there is a detailed movement action, which is divided into three main steps for each disk. First, the disk is lifted from its current position on the pole (move_from) above a preset minimum height (minHeight), a measure that ensures that the disk avoids colliding with other poles during its movement. Next, the disk is moved horizontally through the air until it is directly above the pole at its target position (move_to). Finally, the disk is lowered vertically and placed securely on the target pole to complete the move. This sequence of actions precisely controls the movement of the disk and ensures that it is safely and efficiently rearranged according to the rules of the Tower of Hanoi.

## 5. Create a routine to control the DOBOT arm to complete the Tower of Hanoi puzzle for 3 discs.

Since the maximum disc diameter is 62.4mm, the intermediate position is set here to be 100mm to the left (Y=100mm) and the final position to be 200mm to the left (Y=200mm) to prevent disc overlap from occurring. The specific values of the relative coordinate values represented by the heights of the different positions are marked in Fig. 8. Table 3 shows the positions that the robot arm needs to reach to complete the Tower of Hanoi puzzle with 3 discs each time, and its specific behavioural movements, where the terms can be understood in relation to the markings in Figure 8. The angles of rotation required by the robotic arm at different positions were calculated by the MATLAB code in Question 3 and recorded in Table 3.

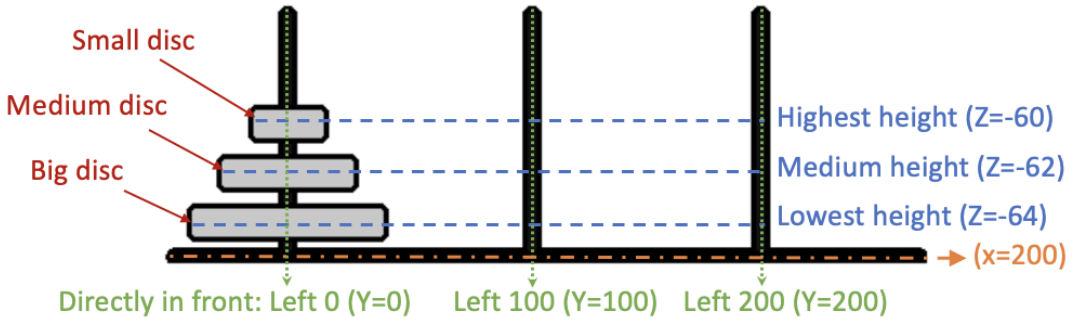


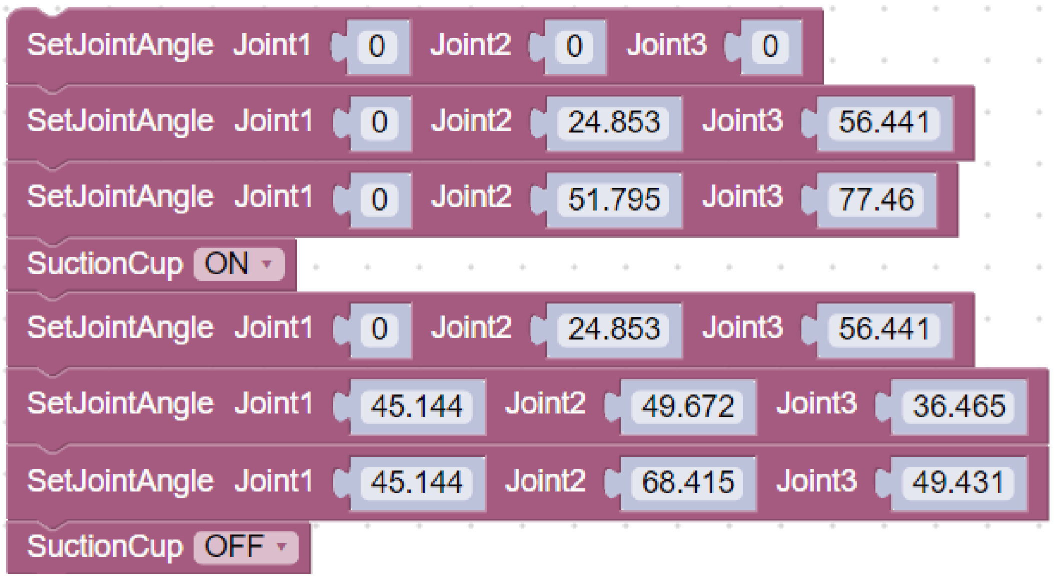
**Figure 8**: Marking of different elements of the Hanoi Tower completion process

**Table 3:** Calculation of the required rotation angles for three joints at different positions using MATLAB

| Action taken by the Dobot | Coordinate | | | MATLAB calculation | | |
|---|---|---|---|---|---|---|
| | X | Y | Z | $\theta_{J1}$ | $\theta_{J2}$ | $\theta_{J3}$ |
| Start at origin; Move forward above discs; Drop to highest height; Suction on; Pick small disc | 208 | 0 | 135 | 0 | 0 | 0 |
| | 199 | 0 | 0 | 0 | 24.853 | 56.441 |
| | 199 | 0 | -60 | 0 | 51.795 | 77.46 |
| Return to just above discs; Move to left 200; Drop to lowest height; Suction off; Drop small disc | 199 | 0 | 0 | 0 | 24.853 | 56.441 |
| | 199 | 200 | 0 | 45.144 | 49.672 | 36.465 |
| | 199 | 200 | -64 | 45.144 | 68.415 | 49.431 |
| Move back to the top of the left 200; Move to just above the front; Drop to medium height; Suction on; Pick medium disc | 199 | 200 | 0 | 45.144 | 49.672 | 36.465 |
| | 199 | 0 | 0 | 0 | 24.853 | 56.441 |
| | 199 | 0 | -62 | 0 | 52.714 | 77.989 |
| Return to just above discs; Move to left 100; Drop to lowest height; Suction off; Drop medium disc | 199 | 0 | 0 | 0 | 24.853 | 56.441 |
| | 199 | 100 | 0 | 26.68 | 31.428 | 51.595 |
| | 199 | 100 | -62 | 26.68 | 55.663 | 70.016 |
| Move back to the top of the left 100; Move to left 200; Drop to lowest height; Suction on; Pick small disc | 199 | 100 | 0 | 26.68 | 31.428 | 51.595 |
| | 199 | 200 | 0 | 45.144 | 49.672 | 36.465 |
| | 199 | 200 | -64 | 45.144 | 68.415 | 49.431 |
| Move back to the top of the left 200; Move to left 100; Drop to medium height; Suction off; Drop small disc | 199 | 200 | 0 | 45.144 | 49.672 | 36.465 |
| | 199 | 100 | 0 | 26.68 | 31.428 | 51.595 |
| | 199 | 100 | -62 | 26.68 | 55.663 | 70.016 |
| Move back to the top of the left 100; Move to directly above; Drop to lowest height; Suction on; Pick big disc | 199 | 100 | 0 | 26.68 | 31.428 | 51.595 |
| | 199 | 0 | 0 | 0 | 24.853 | 56.441 |
| | 199 | 0 | -64 | 0 | 53.632 | 78.505 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Move back to directly above; Move to left 200; Drop to lowest height; Suction off; Drop big disc | 199 | 0 | 0 | 0 | 24.853 | 56.441 |
| | 199 | 200 | 0 | 45.144 | 49.672 | 36.465 |
| | 199 | 200 | -64 | 45.144 | 68.415 | 49.431 |
| Move back to the top of the left 200; Move to left 100; Drop to medium height; Suction on; Pick small disc | 199 | 200 | 0 | 45.144 | 49.672 | 36.465 |
| | 199 | 100 | 0 | 26.68 | 31.428 | 51.595 |
| | 199 | 100 | -62 | 26.68 | 55.663 | 70.016 |
| Move back to the top of the left 100; Move to directly above; Drop to lowest height; Suction off; Drop small disc | 199 | 100 | 0 | 26.68 | 31.428 | 51.595 |
| | 199 | 0 | 0 | 0 | 24.853 | 56.441 |
| | 199 | 0 | -64 | 0 | 53.632 | 78.505 |
| Move back to directly above; Move to left 100; Drop to lowest height; Suction on; Pick medium disc | 199 | 0 | 0 | 0 | 24.853 | 56.441 |
| | 199 | 100 | 0 | 26.68 | 31.428 | 51.595 |
| | 199 | 100 | -64 | 26.68 | 56.487 | 70.464 |
| Move back to the top of the left 100; Move to left 200; Drop to medium height; Suction off; Drop medium disc | 199 | 100 | 0 | 26.68 | 31.428 | 51.595 |
| | 199 | 200 | 0 | 45.144 | 49.672 | 36.465 |
| | 199 | 200 | -62 | 45.144 | 68.415 | 49.43 |
| Move back to the top of the left 200; Move to directly above; Drop to lowest height; Suction on; Pick small disc | 199 | 200 | 0 | 45.144 | 49.672 | 36.465 |
| | 199 | 0 | 0 | 0 | 24.853 | 56.441 |
| | 199 | 0 | -64 | 0 | 53.632 | 78.505 |
| Move back to directly above; Move to left 200; Drop to highest height; Suction off; Drop small disc | 199 | 0 | 0 | 0 | 24.853 | 56.441 |
| | 199 | 200 | 0 | 45.144 | 49.672 | 36.465 |
| | 199 | 200 | -60 | 45.144 | 67.735 | 49.123 |
| Move back to the top of the left 200; Move to directly above; Back to origin | 199 | 200 | 0 | 45.144 | 49.672 | 36.465 |
| | 199 | 0 | 0 | 0 | 24.853 | 56.441 |
| | 208 | 0 | 135 | 0 | 0 | 0 |

The blocky content for completing this task is set in the order of the angles appearing in Table 3. Every three angles set, the suction cup on, and then every three angles set, the suction cup off. It is repeated according to such a cycle, and only one such cycle is shown in Figure 9, to avoid overly large images.



SetJointAngle Joint1 0 Joint2 0 Joint3 0
SetJointAngle Joint1 0 Joint2 24.853 Joint3 56.441
SetJointAngle Joint1 0 Joint2 51.795 Joint3 77.46
SuctionCup ON
SetJointAngle Joint1 0 Joint2 24.853 Joint3 56.441
SetJointAngle Joint1 45.144 Joint2 49.672 Joint3 36.465
SetJointAngle Joint1 45.144 Joint2 68.415 Joint3 49.431
SuctionCup OFF

**Figure 9**: Part of completing the Hanoi Tower mission blockly