**1**

**Robotic Laboratory:**

**Hanoi Tower ( Industrial Robot )**

# OMAR ABDULLAH SALEM BATIS (A19EE4037)

# 1.1 ABSTRACT

This lab is about the investigation on the trajectory path of the industrial robotic arm to give out a solution based on the problem given under suitable operating speed. In this laboratory, the industrial robot used is an ABB robot with the specification of IRB120\_3kg 0.58m. The problem statement given is to identify the trajectory and the shortest path for the ABB robot to solve the Hanoi Tower problem. At the end of this laboratory, the Hanoi Tower problem can be solved in 22 seconds.

# 1.2 INTRODUCTION AND METHODOLOGY

Nowadays, due to the rise of technology, the implementation, and usage of robots in the industry has become wider and important. This is because robots can perform some complex works in a short time and thus increase the efficiency and accuracy of the production line. For the robot to perform and operate automatically, it is very important for us to understand the path and trajectory of the robots before programming them.

In this laboratory, we are required to identify and design some paths and trajectories of the ABB robot to solve the 3 tier Hanoi Tower problem and compare the time taken. Our objective is to find out a path and trajectory which uses the minimum time.

ABBs studio is a robot programming software that enables us to program a robot in the virtual world before it operates in the real world. With RobotStudio®, programming can be done in the office without shutting down production on the factory floor, programs can be prepared in advance, training and optimization can be done without disturbing production, risk of damage or costly delays is reduced, the installation and commissioning of new systems is faster and changeover between production runs is faster. These increase productivity greatly. RAPID is a high-level programming language used to control ABB industrial robots.

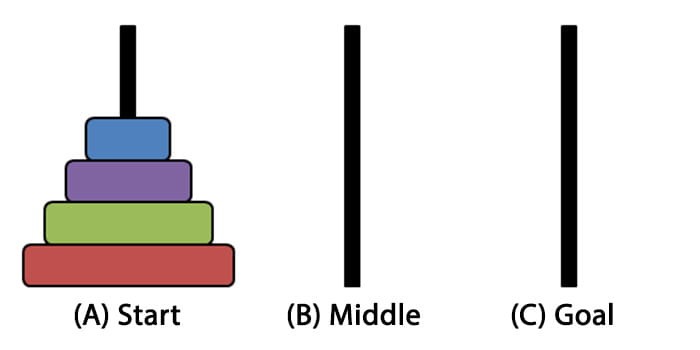
The tower of Hanoi is a mathematical game or puzzle which consists of three roads and several different sizes of disks. Initially, all the disks are placed on one rod, one over the other

in ascending order of size similar to a cone-shaped tower (refer to figure 1).

The objective of the puzzle is to move the entire stack to the last rod, obeying the following simple rules:

1. Only one disk can be moved at a time.
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod.
3. No larger disk may be placed on top of a smaller disk.

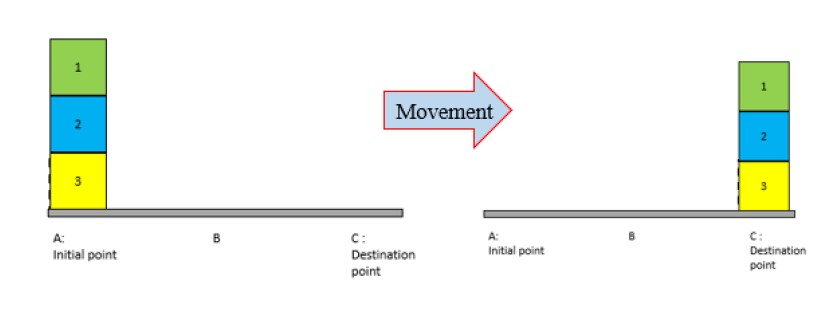
With 3 disks, the puzzle can be solved in 7 moves. The minimal number of moves required to solve a Tower of Hanoi puzzle is 2*n* − 1, where *n* is the number of disks.



*Figure 1*

In this laboratory, the same concept of the Hanoi Tower is used. The disks are replaced with 3 blocks labelled 1, 2, and 3 respectively. At the initial state, the blocks are stack up vertically. The robotic arm needs to solve the Hanoi tower problem in a minimum time. ( refer to

figure 2)

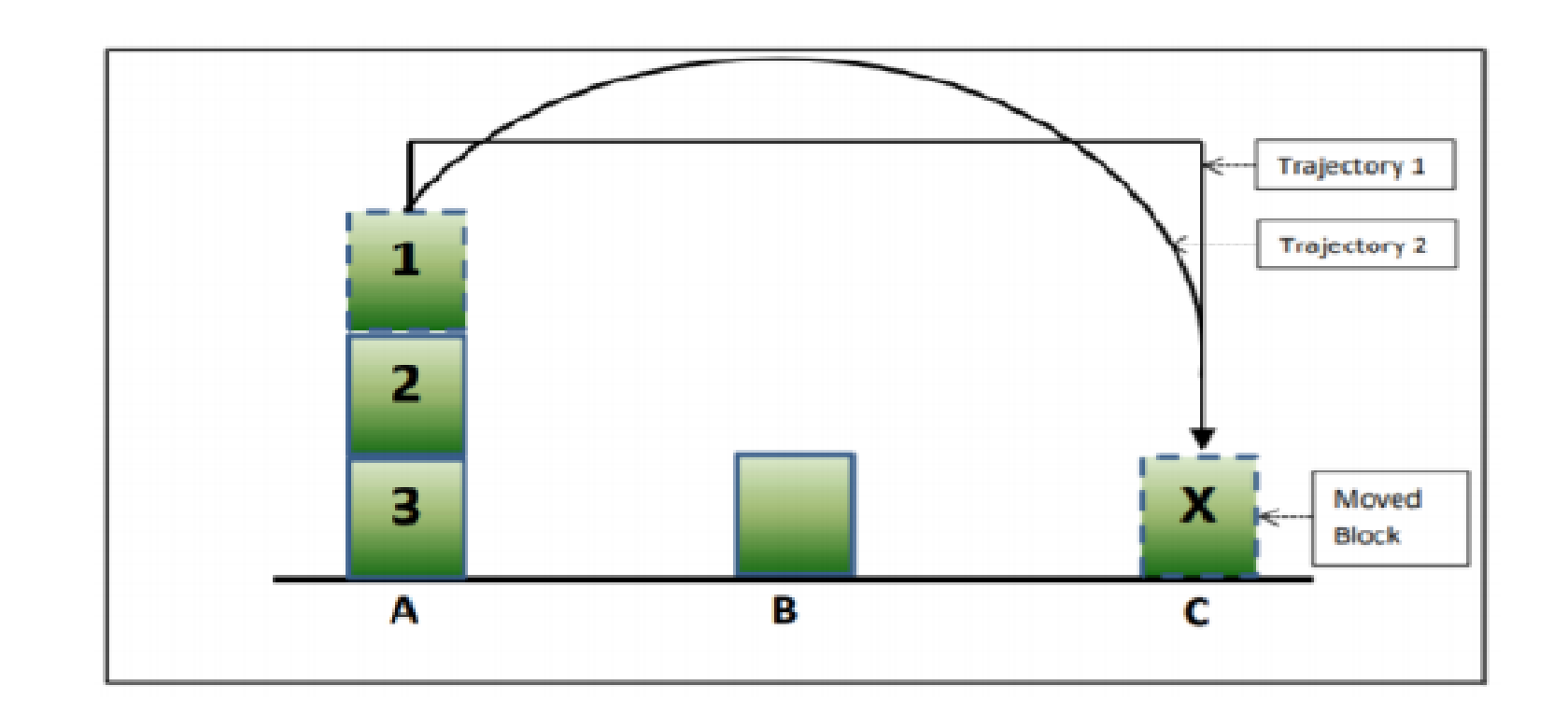


*Figure 2*

The trajectory that the blocks are moved can be done with 3 possible commands in RobotStudio:

|  |  |
| --- | --- |
| Move L | Move the block in a linear path |
| Move J | Move the block in a non-linear path based on the joint of axes |
| Move C | Move the block in a circular path |

We will program two types of trajectories as seen in Figure 3 (below): linear path (trajectory 1) and curved path (trajectory 2), and later compare which trajectory completes the block movement sequence in the shortest time.



*Figure 3*

# 1.3 OBJECTIVES

The purpose of this project is to solve the Hanoi Tower problem by using an ABB robot in the shortest time.

The objectives of this project are:

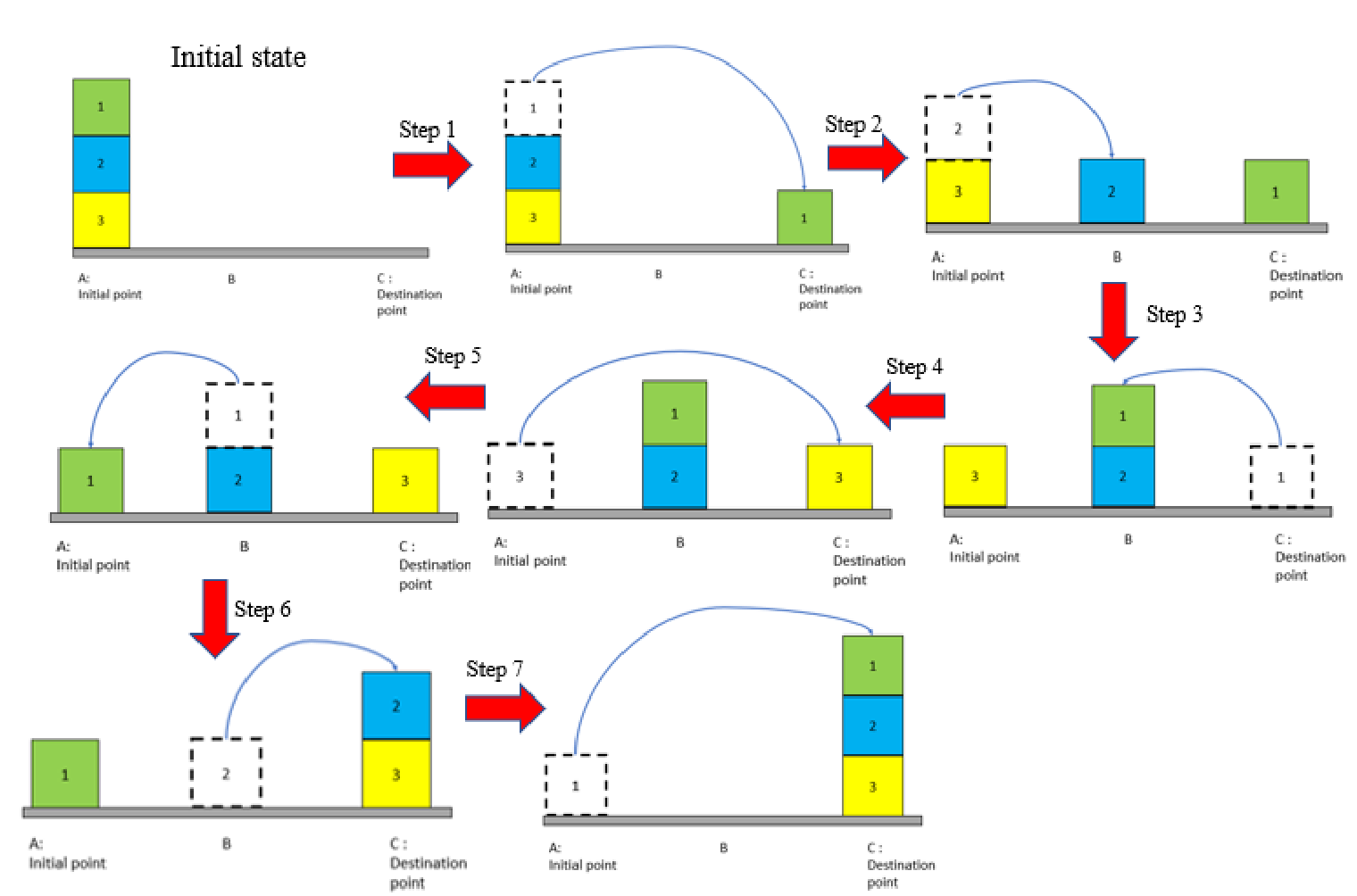
* To move the blocks to the destination point according to the predefined rules of the Hanoi

Tower Problem

* To program the proposed moving sequence of the ABB robot using the RAPID programming language in the RobotStudio.
* To find the path that takes the shortest time to solve the Hanoi Tower problem.

# 1.4 PROCEDURES

1. Three blocks labelled 1, 2, and 3 were arranged vertically on point A, starting with Block no 1 at the most top followed by block no 2, and block number 3 was positioned at the bottom.
2. The number of states required to solve this Hanoi Tower was calculated using the formula 2*n* − 1, where *n* is 3, and we get 6 steps.
3. The robot was calibrated to the initial state. The motion used was Linear motion.
4. A stopwatch was started immediately when the simulation start.
5. Block 1 was moved to point C.
6. Block 2 was moved to point B.
7. Block 1 was moved on tip of Block 2 at point B.
8. Block 3 was moved to point C.
9. Block 1 was moved to point A.
10. Block 2 was moved to point C, on top of Block 3.
11. Block 1 was moved to point C, on top of Block 2.
12. Time taken to move the block was recorded
13. Step 3 to step 12 was repeated using Circular motion.
14. Time taken to solve Hanoi tower between 2 methods was compared. *Please refer to figure 4 for the clear movement of the blocks*



*Figure 4 : Steps to move the blocks*

# 1.5 DATA, RESULT AND SIMULATION

The time taken for method 1(Linear motion) and method 2( circular motion ) was recorded in the table below.

|  |  |
| --- | --- |
| Method | Time taken (seconds) |
| Linear motion | 33 s |
| Circular motion | 22 s |
| Linear and Circular motion | 41.9 s |

**Simulation:**

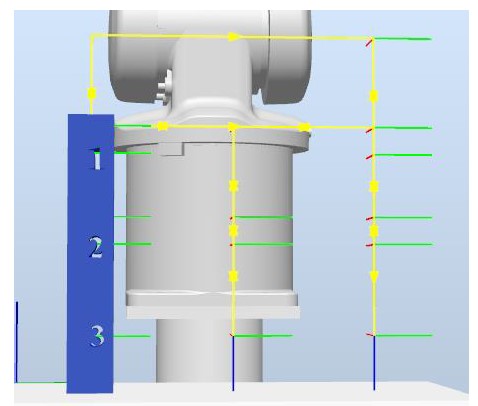
Below are links to the simulation of the robotic arm:

* Method 1 (Linear): <https://drive.google.com/file/d/1ZTcXbD0cyP5EK7n244YWTlML2aGgaion/view?usp=sharing>
* Method 2 (Circular): <https://youtu.be/cCxOez9dZYs>
* Method 2 (Linear and Circular): <https://drive.google.com/file/d/1qe5LnlL--GJ8yYprFYctpTtGPqAqEyE8/view?usp=sharing>

# 1.6 PATH AND RAPID PROGRAMMING

## 1.5.1 Linear motion

### 1.5.1.1 Linear motion path



*Figure 5 : Linear motion using move L instruction*

### 1.6.1.2 Linear motion RAPID programming

MODULE Module1

PROC main()

!Add your code here

Path\_start;

Path\_1;

Path\_2;

Path\_3;

Path\_4;

Path\_5;

Path\_6;

Path\_7;

Path\_start;

ENDPROC

PROC grip\_block1()

WaitTime 0.2;

SetDO grip1, 1;

ENDPROC

PROC ungrip\_block1()

WaitTime 0.2;

SetDO grip1, 0;

ENDPROC

PROC grip\_block2()

WaitTime 0.2;

SetDO grip2, 1;

ENDPROC

PROC ungrip\_block2()

WaitTime 0.2;

SetDO grip2, 0;

ENDPROC

PROC grip\_block3()

WaitTime 0.2;

SetDO grip3, 1;

ENDPROC

PROC ungrip\_block3()

WaitTime 0.2;

SetDO grip3, 0;

ENDPROC

PROC Path\_1() ungrip\_block1;

MoveL Step1\_Appr\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; MoveL Step1\_Target\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; grip\_block1;

MoveL Step1\_Appr\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1;

MoveL Step1\_Left\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; MoveL Step1\_Down\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; ungrip\_block1;

MoveL Step1\_Up\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; ENDPROC

PROC Path\_start()

MoveL pStartPos,v300,fine,tMyGripper\WObj:=Wobj\_StartPos;

ENDPROC

PROC Path\_2()

MoveL Step2\_Appr\_pBlock2,v300,fine,tMyGripper\WObj:=Wobj\_Block2; MoveL Step2\_Target\_pBlock2,v300,fine,tMyGripper\WObj:=Wobj\_Block2; grip\_block2;

MoveL Step2\_Appr\_pBlock2,v300,fine,tMyGripper\WObj:=Wobj\_Block2;

MoveL Step2\_Left\_pBlock2,v300,fine,tMyGripper\WObj:=Wobj\_Block2; MoveL Step2\_Down\_pBlock2,v300,fine,tMyGripper\WObj:=Wobj\_Block2; ungrip\_block2;

MoveL Step2\_Up\_pBlock2,v300,fine,tMyGripper\WObj:=Wobj\_Block2;

ENDPROC

PROC Path\_3()

MoveL Step3\_Appr\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; MoveL Step3\_Target\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; grip\_block1;

MoveL Step3\_Up1\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1;

MoveL Step3\_Right\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; MoveL Step3\_Down\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; ungrip\_block1;

MoveL Step3\_Up2\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1;

ENDPROC

PROC Path\_4()

MoveL Step4\_Appr\_pBlock3,v300,fine,tMyGripper\WObj:=Wobj\_Block3; MoveL Step4\_Target\_pBlock3,v300,fine,tMyGripper\WObj:=Wobj\_Block3; grip\_block3;

MoveL Step4\_Appr\_pBlock3,v300,fine,tMyGripper\WObj:=Wobj\_Block3;

MoveL Step4\_Left\_pBlock3,v300,fine,tMyGripper\WObj:=Wobj\_Block3; MoveL Step4\_Down\_pBlock3,v300,fine,tMyGripper\WObj:=Wobj\_Block3; ungrip\_block3;

MoveL Step4\_Up\_pBlock3,v300,fine,tMyGripper\WObj:=Wobj\_Block3;

ENDPROC

PROC Path\_5()

MoveL Step5\_Appr\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; MoveL Step5\_Target\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; grip\_block1;

MoveL Step5\_Appr\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1;

MoveL Step5\_Right\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; MoveL Step5\_Down\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; ungrip\_block1;

MoveL Step5\_Up\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; ENDPROC

PROC Path\_6()

MoveL Step6\_Appr\_pBlock2,v300,fine,tMyGripper\WObj:=Wobj\_Block2; MoveL Step6\_Target\_pBlock2,v300,fine,tMyGripper\WObj:=Wobj\_Block2; grip\_block2;

MoveL Step6\_Up1\_pBlock2,v300,fine,tMyGripper\WObj:=Wobj\_Block2;

MoveL Step6\_Left\_pBlock2,v300,fine,tMyGripper\WObj:=Wobj\_Block2; MoveL Step6\_Down\_pBlock2,v300,fine,tMyGripper\WObj:=Wobj\_Block2; ungrip\_block2;

MoveL Step6\_Up2\_pBlock2,v300,fine,tMyGripper\WObj:=Wobj\_Block2;

ENDPROC

PROC Path\_7()

MoveL Step7\_Appr\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; MoveL Step7\_Target\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; grip\_block1;

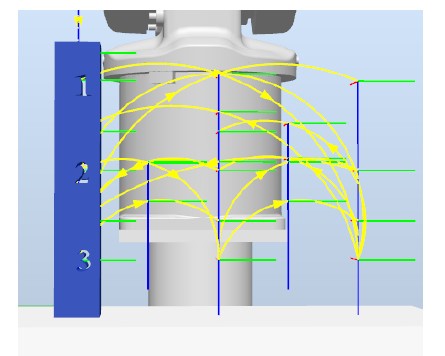
MoveL Step7\_Up1\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; MoveL Step7\_Left\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; MoveL Step7\_Down\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; ungrip\_block1;

MoveL Step7\_Up2\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; ENDPROC

ENDMODULE

## 1.6.2 Circular motion

### 1.6.2.1 Circular motion path



*Figure 6 : Circular motion using move J and move C instruction*

### 1.5.2.2 Circular path rapid programming

MODULE Module1

PROC main()

Path\_Start;

Path\_1;

Path\_2;

Path\_3;

Path\_4;

Path\_5;

Path\_6;

Path\_7;

Path\_Start;

ENDPROC

PROC grip\_block1()

WaitTime 0.2;

SetDO grip1, 1;

ENDPROC

PROC grip\_block2()

WaitTime 0.2;

SetDO grip2, 1;

ENDPROC

PROC grip\_block3()

WaitTime 0.2;

SetDO grip3, 1;

ENDPROC

PROC ungrip\_block1()

WaitTime 0.2;

SetDO grip1, 0;

ENDPROC

PROC ungrip\_block2()

WaitTime 0.2;

SetDO grip2, 0;

ENDPROC

PROC ungrip\_block3()

WaitTime 0.2;

SetDO grip3, 0;

ENDPROC

PROC Path\_Start()

MoveJ StartPos,v300,fine,tMyGripper\WObj:=Wobj\_StartPos;

ENDPROC

PROC Path\_1()

MoveJ Step1\_Appr\_Block1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; MoveJ Step1\_Target\_Block1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; grip\_block1; MoveC Step1\_Mid\_Block1,Step1\_Down\_Block1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; MoveJ Step1\_Down\_Block1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; ungrip\_block1; MoveC Step1\_Mid2\_Block1,Step1\_Target2\_Block1,v300,fine,tMyGripper\WObj:=Wobj\_Block1;

ENDPROC

PROC Path\_2()

MoveJ Step2\_Target\_Block2,v300,fine,tMyGripper\WObj:=Wobj\_Block2; grip\_block2; MoveC Step2\_Mid\_Block2,Step2\_Down\_Block2,v300,fine,tMyGripper\WObj:=Wobj\_Block2; MoveJ Step2\_Down\_Block2,v300,fine,tMyGripper\WObj:=Wobj\_Block2; ungrip\_block2; MoveC Step2\_Mid2\_Block2,Step2\_Target2\_Block2,v300,fine,tMyGripper\WObj:=Wobj\_Block2;

ENDPROC

PROC Path\_3()

MoveJ Step3\_Target\_Block1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; grip\_block1; MoveC Step3\_Mid\_Block1,Step3\_Down\_Block1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; MoveJ Step3\_Down\_Block1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; ungrip\_block1; MoveC Step3\_Mid2\_Block1,Step3\_Target2\_Block1,v300,fine,tMyGripper\WObj:=Wobj\_Block1;

ENDPROC

PROC Path\_4()

MoveJ Step4\_Target\_Block3,v300,fine,tMyGripper\WObj:=Wobj\_Block3; grip\_block3; MoveC Step4\_Mid\_Block3,Step4\_Down\_Block3,v300,fine,tMyGripper\WObj:=Wobj\_Block3; MoveJ Step4\_Down\_Block3,v300,fine,tMyGripper\WObj:=Wobj\_Block3; ungrip\_block3; MoveC Step4\_Mid2\_Block3,Step4\_Target2\_Block3,v300,fine,tMyGripper\WObj:=Wobj\_Block3;

ENDPROC

PROC Path\_5()

MoveJ Step5\_Target\_Block1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; grip\_block1; MoveC Step5\_Mid\_Block1,Step5\_Down\_Block1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; MoveJ Step5\_Down\_Block1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; ungrip\_block1; MoveC Step5\_Mid2\_Block1,Step5\_Target2\_Block1,v300,fine,tMyGripper\WObj:=Wobj\_Block1;

ENDPROC

PROC Path\_6()

MoveJ Step6\_Target\_Block2,v300,fine,tMyGripper\WObj:=Wobj\_Block2; grip\_block2; MoveC Step6\_Mid\_Block2,Step6\_Down\_Block2,v300,fine,tMyGripper\WObj:=Wobj\_Block2; MoveJ Step6\_Down\_Block2,v300,fine,tMyGripper\WObj:=Wobj\_Block2; ungrip\_block2; MoveC Step6\_Mid2\_Block2,Step6\_Target2\_Block2,v300,fine,tMyGripper\WObj:=Wobj\_Block2;

ENDPROC

PROC Path\_7()

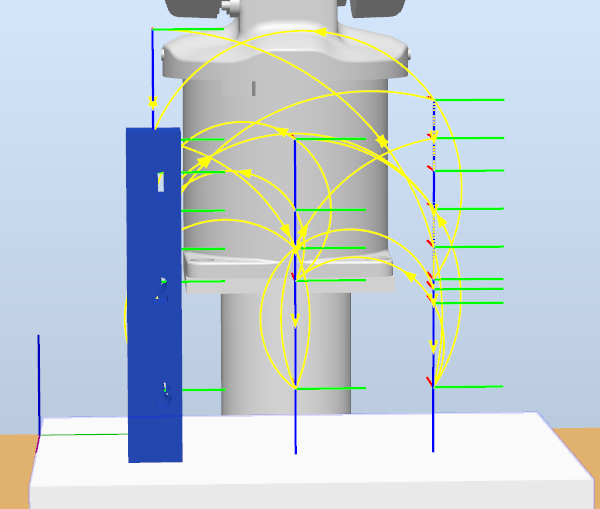
MoveJ Step7\_Target\_Block1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; grip\_block1; MoveC Step7\_Mid\_Block1,Step7\_Down\_Block1,v300,fine,tMyGripper\WObj:=Wobj\_Block1; ungrip\_block1;

ENDPROC

ENDMODULE

## 1.6.3 Linear and Circular motion

### 1.6.3.1 Linear and Circular motion path



### 1.6.3.2 Linear and Circular motion programming

PROC main()

Home;

Path\_start;

Path\_Block1;

Path\_Block2;

Path\_Block1\_20;

Path\_Block3;

Path\_Block1\_3;

Path\_Block2\_2;

Path\_Block1\_4;

Path\_start;

Home;

ENDPROC

PROC Home()

MoveJ HomePos,v300,fine,tMyGripper\WObj:=wobj0;

ENDPROC

PROC Path\_start()

MoveJ pStartPos,v300,fine,tMyGripper\WObj:=Wobj\_StartPos;

ENDPROC

PROC grip\_block1()

WaitTime 1;

SetDo grip1,1;

ENDPROC

PROC ungrip\_block1()

WaitTime 1;

SetDo grip1,0;

ENDPROC

PROC grip\_block2()

WaitTime 1;

SetDo grip2,1;

ENDPROC

PROC ungrip\_block2()

WaitTime 1;

SetDo grip2,0;

ENDPROC

PROC grip\_block3()

WaitTime 1;

SetDo grip3,1;

ENDPROC

PROC ungrip\_block3()

WaitTime 1;

SetDo grip3,0;

ENDPROC

PROC Path\_Block1()

ungrip\_block1;

MoveJ Appr\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1;

MoveL pBlock1\_10,v300,fine,tMyGripper\WObj:=Wobj\_Block1;

grip\_block1;

MoveL Appr\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1;

MoveC Appr\_pBlock1place,pBlock1\_10place,v300,fine,tMyGripper\WObj:=Wobj\_Block1;

MoveJ pBlock1\_10place,v300,fine,tMyGripper\WObj:=Wobj\_Block1;

ungrip\_block1;

MoveC Appr\_pBlock1place,Appr\_pBlock1,v300,fine,tMyGripper\WObj:=Wobj\_Block1;

ENDPROC

PROC Path\_Block2()

ungrip\_block2;

MoveJ pBlock2\_10,v300,fine,tMyGripper\WObj:=Wobj\_Block2;

MoveL Appr\_pBlock2,v300,fine,tMyGripper\WObj:=Wobj\_Block2;

grip\_block2;

MoveL pBlock2\_10,v300,fine,tMyGripper\WObj:=Wobj\_Block2;

MoveC Appr\_pBlock2place,pBlock2\_10place,v300,fine,tMyGripper\WObj:=Wobj\_Block2;

MoveJ pBlock2\_10place,v300,fine,tMyGripper\WObj:=Wobj\_Block2;

ungrip\_block2;

MoveC Appr\_pBlock2place,Appr\_pBlock1place\_3\_3,v300,fine,tMyGripper\WObj:=Wobj\_Block2;

ENDPROC

PROC Path\_Block1\_20()

ungrip\_block1;

MoveJ Appr\_pBlock1place\_3,v300,fine,tMyGripper\WObj:=Wobj\_Block1\_2;

MoveL pBlock1\_10place,v300,fine,tMyGripper\WObj:=Wobj\_Block1;

grip\_block1;

MoveC Appr\_pBlock1place\_3,pBlock1\_2place,v300,fine,tMyGripper\WObj:=Wobj\_Block1\_2;

MoveJ pBlock1\_2place,v300,fine,tMyGripper\WObj:=Wobj\_Block1\_2;

ungrip\_block1;

MoveC Appr\_pBlock1\_2place,Appr\_pBlock3\_2,v300,fine,tMyGripper\WObj:=Wobj\_Block1\_2;

ENDPROC

PROC Path\_Block3()

ungrip\_block3;

MoveJ Appr\_pBlock3,v300,fine,tMyGripper\WObj:=Wobj\_Block3;

MoveL pBlock3\_10,v300,fine,tMyGripper\WObj:=Wobj\_Block3;

grip\_block3;

MoveC Appr\_pBlock3,Appr\_pBlock1place\_2\_2,v300,fine,tMyGripper\WObj:=Wobj\_Block3;

MoveJ pBlock1\_10place\_2\_2,v300,fine,tMyGripper\WObj:=Wobj\_Block3;

ungrip\_block3;

MoveC Appr\_pBlock1place\_2\_2,Appr\_pBlock1\_3\_2,v300,fine,tMyGripper\WObj:=Wobj\_Block3;

ENDPROC

PROC Path\_Block1\_3()

ungrip\_block1;

MoveJ Appr\_pBlock1\_3,v300,fine,tMyGripper\WObj:=Wobj\_Block1\_3;

MoveL pBlock1\_3,v300,fine,tMyGripper\WObj:=Wobj\_Block1\_3;

grip\_block1;

MoveC Appr\_pBlock1\_3,Appr\_pBlock1\_3place,v300,fine,tMyGripper\WObj:=Wobj\_Block1\_3;

MoveJ pBlock1\_3place,v300,fine,tMyGripper\WObj:=Wobj\_Block1\_3;

ungrip\_block1;

MoveC Appr\_pBlock1\_3place,Appr\_pBlock2\_2\_2,v300,fine,tMyGripper\WObj:=Wobj\_Block1\_3;

ENDPROC

PROC Path\_Block2\_2()

ungrip\_block2;

MoveJ Appr\_pBlock2\_2,v300,fine,tMyGripper\WObj:=Wobj\_Block2\_2;

MoveL pBlock2\_20,v300,fine,tMyGripper\WObj:=Wobj\_Block2\_2;

grip\_block2;

MoveC Appr\_pBlock2\_2,Appr\_pBlock2\_2place,v300,fine,tMyGripper\WObj:=Wobj\_Block2\_2;

MoveJ pBlock2\_20place,v300,fine,tMyGripper\WObj:=Wobj\_Block2\_2;

ungrip\_block2;

MoveC Appr\_pBlock1\_4place\_2,Appr\_pBlock1\_4\_2,v300,fine,tMyGripper\WObj:=Wobj\_Block2\_2;

ENDPROC

PROC Path\_Block1\_4()

ungrip\_block1;

MoveL Appr\_pBlock1\_4,v300,fine,tMyGripper\WObj:=Wobj\_Block1\_4;

MoveJ pBlock1\_40,v300,fine,tMyGripper\WObj:=Wobj\_Block1\_4;

grip\_block1;

MoveC Appr\_pBlock1\_4,Appr\_pBlock1\_4place,v300,fine,tMyGripper\WObj:=Wobj\_Block1\_4;

MoveJ pBlock1\_40place,v300,fine,tMyGripper\WObj:=Wobj\_Block1\_4;

ungrip\_block1;

ENDPROC

ENDMODULE

Group 7 -Robotic Laboratory: Hanoi Tower ( Industrial Robot)

# 1.7 DISCUSSION

## 1.7.1 Problem faced

In order to use MoveC, 2 positions need to be set and the robot arm will move to the desired position based on the 2 points and its original position. However, it is difficult to find the 2 positions for the shortest path as compared to MoveL and MoveJ. This is due to the fact that the position of the circle point cannot be too close to the starting point and the destination point. Sometimes the shortest path that has been planned is rejected due to the limitation of MoveC instruction. Most of the time, a major arc of a circle is formed instead of the desired minor arc. The major arc makes the path sway out before settling at the target position. While the end goal is achieved, we had to painstakingly vary the end points to form a clean minor arc. But some paths will have a slight major arc due to the limitation of range of the end points.

Duplicating a station may yield an error of the same virtual controllers being used for two stations. Doing so will yield the entire station inaccessible and thus, the loss of your hard work. Hence, it is advised to duplicate a solution and not a station.

## 1.7.2 Discussion on the result

From the result we obtained, we can conclude that method 2 ( circular path ) is faster compared to method 1 ( linear path ), in which the time taken to solve the Hanoi Tower problem is 22 second for method 2 and 33 second for method 1. MoveL is used in method 1 and MoveJ and

MoveC are used in the trajectory path for method 2.

For Method 1, only MoveL was used. As MoveL involves only the linear movement of the robot arm, thus more instructions are required to perform the desired path and avoid the collision between the blocks and the robot arm. Hence, method 1 is less efficient but we can easily account for obstacle avoidance in our trajectory path as the linear path is deterministic and predictable.

For Method 2, MoveJ and MoveC are used. MoveJ is the instruction that enables the faster movement of blocks from one location to another as it was set to take the shortest path from the current position to the desired position, whereas MoveC is used to solve the weaknesses of MoveJ instruction. For example, for some movements MoveJ needs 2 steps to complete one movement but MoveC can take 1 step to complete one movement. The weakness of MoveC is that it is difficult to account for obstacle avoidance in the circular path. We have to beware of setting the coordinates while testing the paths, so that we can avoid any obstacle in the trajectory path.

Group 7 -Robotic Laboratory: Hanoi Tower ( Industrial Robot)

# 1.8 CONCLUSION

From this laboratory, we have learned how to perform the simulation of the robot arm using ABB Robot Studio. We know the way to set and program the robot arm using RAPID programming. We successfully found the shortest path to solve the Hanoi Tower which is by using circular motion where include MoveJ and MoveC instruction. The time taken for the robot arm using circular motion to solve the Hanoi Tower is 22s.

# 1.9 REFERENCE

Dr. Mohd Ridzuan Ahmad ,Dr. Mohamad Hafis Izran Ishak ,En. Ahmad Ridhwan Wahab and

Dr.Mohamad Amir Shamsudin Additional\_Materials\_PBLab\_Robotik\_IndustrialRobots\_

StudentPack\_ABBpick&placeExercises, *ROBOTICS LAB ADDITIONAL*

*MATERIALSRobotStudio Pick & Place Exercise , 2-27 , 2020*

Wikipedia, the free encyclopedia, Tower of Hanoi from <https://en.wikipedia.org/wiki/Tower_of_Hanoi>

[Paul Miller](https://www.theverge.com/users/futurepaul), Calm your troubled soul by watching this robot arm solve Tower of Hanoi from [https://www.theverge.com/circuitbreaker/2016/6/29/12061860/robot-arm-solves-tower-of -hanoi](https://www.theverge.com/circuitbreaker/2016/6/29/12061860/robot-arm-solves-tower-of-hanoi)