RAS 545 Final Project:

Smart and Accurate Path Planning with the Mycobot Pro 600: Solving Mazes

Instructor: Prof. Mostafa Yourdkhani

Lab-incharge: Rajdeep Adak

Date: 10th April 2025

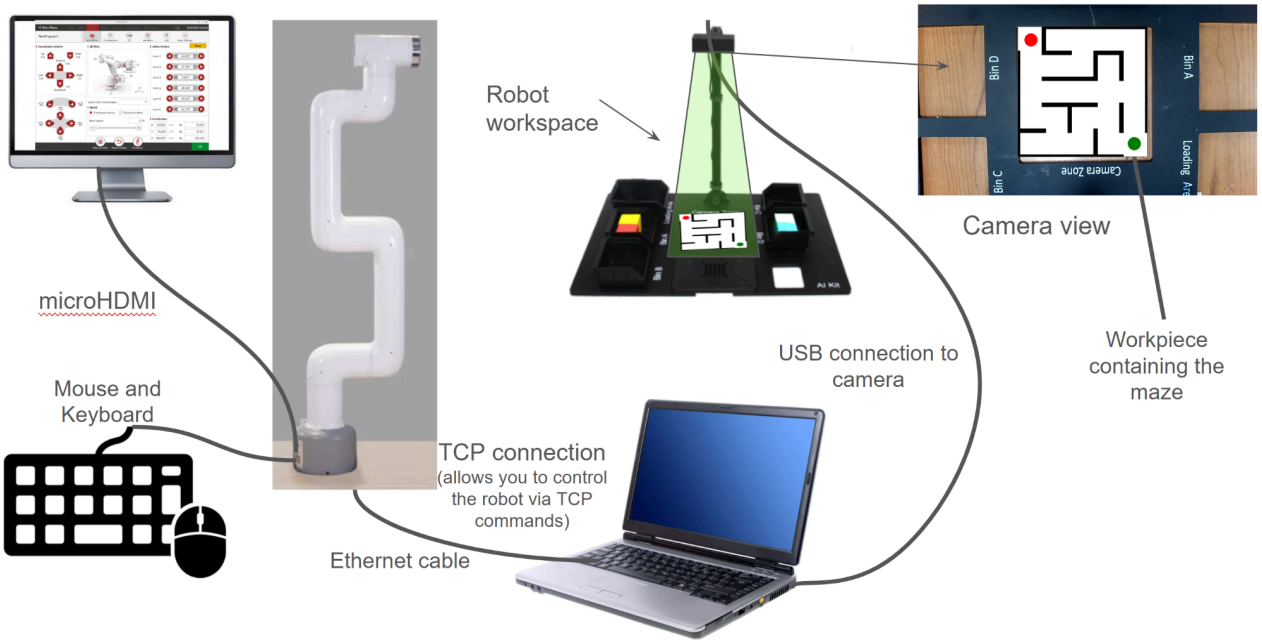
Introduction

So far you have been tasked with the development of a digital twin and simple path planning based on visual feedback from a camera. Now you will perform smart and accurate path planning with the cobot. This project builds on the concepts you have previously learned, therefore, we will reuse the system you have created so far. This project is more algorithm heavy and in that you must now think of an efficient way to accurately traverse a path

Laboratory Task

This lab task is to make the robot solve a maze in the ***shortest*** way possible. You will be using a camera to capture the image of a Maze. The maze will have a starting point (**red circle**) and an end point (**green dot**). A maze can have several solutions. From the image of the maze, you should be able to determine the physical coordinates of the shortest path from start to end. Use your digital twin to obtain the real joint angle solution for the 6 joints, and then perform FK iteratively by sending TCP commands to move to the robot. The result should be the robot's end effector following the maze solution path.

The following diagram shows the setup of the robot in the lab. The hardware is already in place. Note that *the starting point (red circle) of the maze will always be away from the camera zone and the end point will be close to the camera zone as shown in the image below*



General Advice

This lab activity tests your learnings from the decisions you made in the previous labs. Whether good or bad, you must be able to improve. Secondly, industrial path planning tasks are often complex, requiring the robot to determine efficient ways to reach a destination. You have already learnt how to incorporate visual feedback from a camera, but now you will work more on the "brain" of the robot. Making the robot smarter really depends on how smart you are (ha ha). This lab will test your ability to deliver accuracy. Therefore there is very little scope of errors and tolerances. Aside from that, this lab will require problem solving skills, critical thinking, engineering ingenuity, time and effort management skills and general proficiency with robotics software. You must do teamwork to collectively determine a plan and the underlying steps to achieve the goal. Lastly note that in industry and academia, successful professionals are always very good decision makers.

Technical Advice

1. You need not use a new cobot digital twin.
2. Get accustomed to using the cobot beforehand.
3. Time management is of key importance. This lab activity requires you to obtain a mathematical understanding of the robot's workspace. Come up with a smart approach to make your robot go to a physical location by looking at its image captured by the camera. Tip: ***There exists a smart way to obtain the mathematical understanding of the robot's workspace by mapping the pixel coordinates to physical coordinates. Figure this out on your own.***
4. Use the same platform to perform image processing and inverse kinematics. It saves a lot of time and effort. For example: If you are using MATLAB to perform inverse kinematics, use MATLAB's image processing toolbox to perform path detection.
5. Work in a team. Produce one really good solution that satisfies the expectations of the lab.
6. You might want to build a small 15cm x 15cm workpiece of your own. Use the maze generator code provided in the ***Maze Generator Code*** *section to print mazes. During the lab session, we will provide you with several samples that are generated from the same code. The solutionless maze will be printed on the workpiece samples.*
7. You can use any process to solve the maze that does NOT involve human intervention.
8. Interpolation between any two straight sections of the solution path is NOT required.

Note that it is not mandatory to follow the above advice. You are free to make your own decisions.

Advice on Software/tools to use

You can choose any software of your preference to build the digital twin. But we provide the following suggestions:
1. Python or MATLAB for doing mathematical calculations.
2. Simulink is a good option for designing robot workflows. Explore the robotic systems toolbox from the learning resources.
3. Python IKPy is a good package to do inverse kinematics.
4. ROS: Robotic Operating Systems has several tools to deploy and simulate robot URDFs such as Gazebo and Rviz.
5. MATLAB has various inverse kinematics solvers as well.
6. For viewing the CAD, you can use any software. We suggest SolidWorks, AutoCAD, Fusion 360, Blender etc.
7. CAD files often require conversions. You can use the tool on 3DPEA to convert CAD files.
8. If you are ambitious, try using MuJoCo

9. For image processing, we suggest using OpenCV, Matlab's Image Processing Toolbox or any other image processing tool that suits your architecture.

Additional Learning Resources:

The learning resources provide a general direction towards various relevant topics/subtopics. It is advised that you go through these resources and develop your own approach. By now you must have already gained the knowledge needed to attempt the project. However, some important ones are mentioned below:

1. The Homogeneous Transformation for the cobot pro 600 has already been posted on the canvas. Use them to validate your FK and IK solution.
2. You must already be aware that the mycobot pro 600 is easily programmable via TCP commands.The TCP command list can be found **here**. You can send TCP commands via any programming language.
3. Maze solving is a general process. There are several path finding algorithms to solve mazes. Some are based on searching feasible routes, brute forcing, or using machine learning. Feel free to choose any approach of your choice. Here's a video of how to write your own maze solver from scratch: **link.** This lab does NOT expect you to build your own maze solver from scratch.

Note that it is not necessary to only use the above resources. You are free to refer to any available online resources. But make sure to include references in your report.

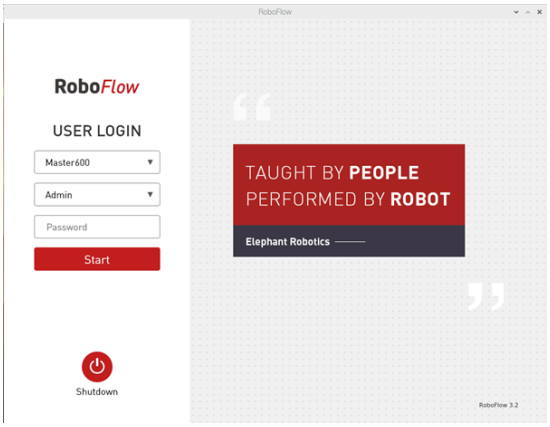
The general workflow of the expected solution

1. Connect the camera to the USB port on your computer. Take an image of the workpiece that has the maze on it.
2. Use any programming environment of your choice to perform a maze solution from the image of the workpiece taken by the camera.
3. Determine the pixel coordinates of the path that forms the maze solution.
4. Figure out a way to convert the pixel coordinates to physical x, y, z coordinates to which the robot will move.
5. Pass the x, y, z coordinates of the solution path one at a time to an IK solver. Obtain the realistic joint angles for the given x, y, z coordinates. The IK solver can be implemented in any approach of your choice.
6. ***Important****: Before sending any commands allocate a home position of the cobot near the camera but outside its view. This prevents any possible collisions with the camera setup. Do not use the origin of the robot (upright position) as the home position.*
7. *Arrange the robot in such a way that the end-effector is pointing downwards throughout the movement. Below is an image of such an arrangement of the robot.*

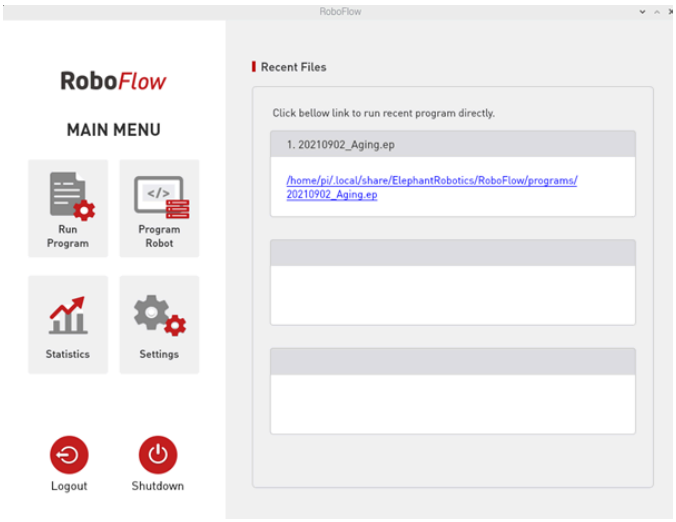
8. Use the `set_angles(joint1_angle, joint2_angle, joint3_angle, joint4_angle, joint5_angle, joint6_angle, speed)` TCP command to send the joint angles to the robot via ethernet connection. Refer to the TCP command list. An example code in Python and MATLAB is provided here. See the "**Process of connecting my cobot pro 600 to your computer**" section to know how to connect your robot to your computer.
9. The maze solution path will have several turns. Create a program to loop through these turning points one at a time. The result should be the maze solution path.
10. Between each command, there should be a wait time for the robot to complete the movement. Figure out the best possible time to wait to ensure a smooth navigation. **Tip: Use the wait_command_done() function from the API list.**
11. Remember that the starting point is the red circle and the end point is the green circle.

Process of connecting my cobot pro 600 to your computer.

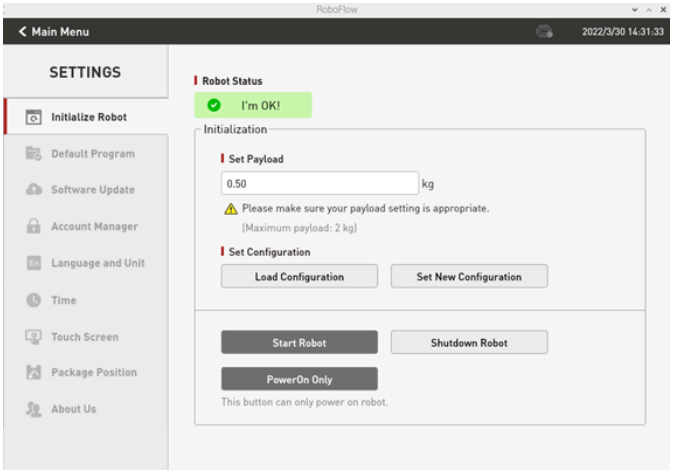
1. Power on the robot.
2. Connect the robot to your computer via the ethernet cable.
3. Connect the robot to a monitor, keyboard and mouse. Most robots already have them.
4. Log into roboflow OS using the password: **elephant**. Username: **admin**.



5. After logging in the main menu is displayed

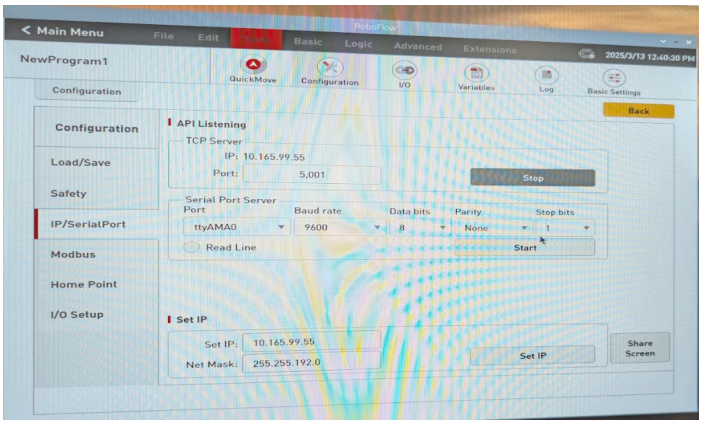


6. Click on the “**Settings**” icon and on the side bar select “**Initialize Robot**”. In this section, click “**Start Robot**”. This will turn-on the robot. Wait for the robot to activate. You will hear a small click sound when the robot motors are activated. Once the robot is turned-on, the menu will display the message “**I’m OK**”.



7. Return to the Main Menu and click “**Program Robot**”, and then click “**Empty Program**”.
8. The “**Quick Move**” panel will appear.

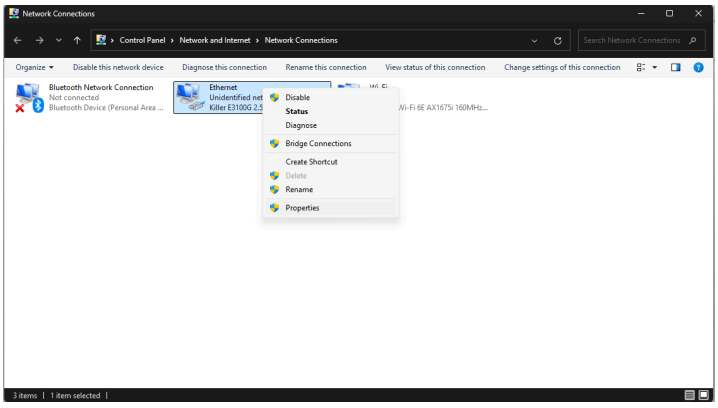
9. Click on “**Configuration**” on the topbar and then click on “**IP/Serial Port**”. The robot broadcasts TCP commands on this IP address. The broadcasting can be started/stopped using the gray button. If the broadcast is stopped then a computer will not be able to communicate with the robot.



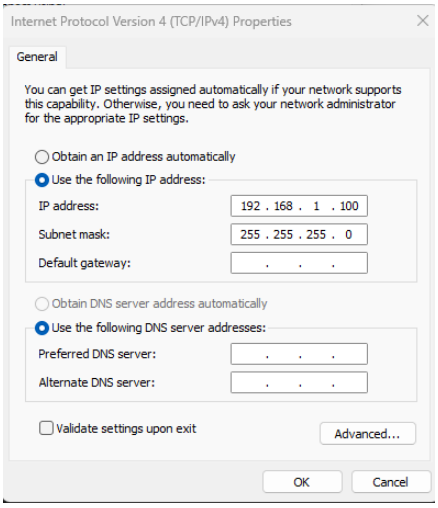
10. Connect your computer to the robot via an RJ45 ethernet cable.
11. On your computer Open Network settings and set your ethernet port’s IPv4 address to the same domain as the robot’s IP address. This allows your device to be in the same network as your robot.

    For example: If the robot’s IP address is **192.168.0.100**, then your ethernet port’s IP address should have the same first 3 values i.e. **192.168.0.XXX**. **XXX** can be any number between 1 and 255 except 100.

    On windows this can be done by searching “**View Network Connections**” in the Windows Taskbar Search. Right click on the **Ethernet “Unidentified Network”** and then click **properties.**

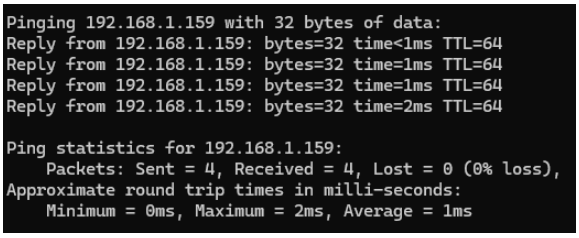


12. In the properties window. Scroll to “**Internet Protocol Version 4 (TCP/IPv4)**” and set the IP address as per the instructions mentioned in point 11. The subnet mask should be the same as that of the robot.



13. Confirm by pinging the robot via the terminal. Ping the robot using the ping command in the terminal to verify the connection.

ping < IP address of the robot >



14. You are now ready to program the robot via TCP commands. Run the example code given in the link.

CAD Files of the mycobot pro 600

You will require CAD files for building the digital twin. Please download them from the link provided below.

Link to mycobot pro 600 CAD File

Maze Generator Code

The mazes can been generated using the python code at the link below. Download the program and run it in your system to view randomly generated mazes each time you run the code. The maze is saved in a png file named “unsolved_maze.png”. You can generate your own samples on a 15 cm x 15 cm paper/cardboard to test your code.

You will require the pillow library: pip install pillow

Link to maze generator code

Grading Policy

Total: **6 + 4 = 10 points**

Demonstration (2 + 2 + 1 + 1 = 6 points)

All teams should demonstrate their approach on building the robot digital twin and verifying Inverse Kinematics solutions to the Lab incharge. The maze workpiece will be kept on a horizontal plane within the view of the camera. You must demonstrate a general solution to the robot following the maze solution path. This must be completed **before the due date**, and can only be done in-person. The demonstration should meet the following expectations:

1. (2 points) **Digital Twin and IK**: This digital twin model should be able to take the 6 joint angles as an input and perform robot movements. This demonstrates the forward kinematic capability. The digital twin model should also be able to perform inverse kinematics and produce a realistic solution of the 6 joint angles that yields the given end-effector state.

2. (2 points) **Accuracy**: A TCP tracer will be used to determine where the robot's end-effector is pointing. You will be judged on the ability of the robot to follow the solution path and NOT collide with the walls of the maze.

3. (1 points) **Speed**: Total time it takes to solve the maze. The entire maze (start to end) should be traversed by the robot within **1 minute**.

4. (1 points) **Efficiency**: A maze can have multiple paths. Your solution must provide the shortest path.

Submission (1 + 2 + 1 = 4 points)

All teams must upload their submission on canvas **by the due date**. Your submission should meet the submission requirements (see below).

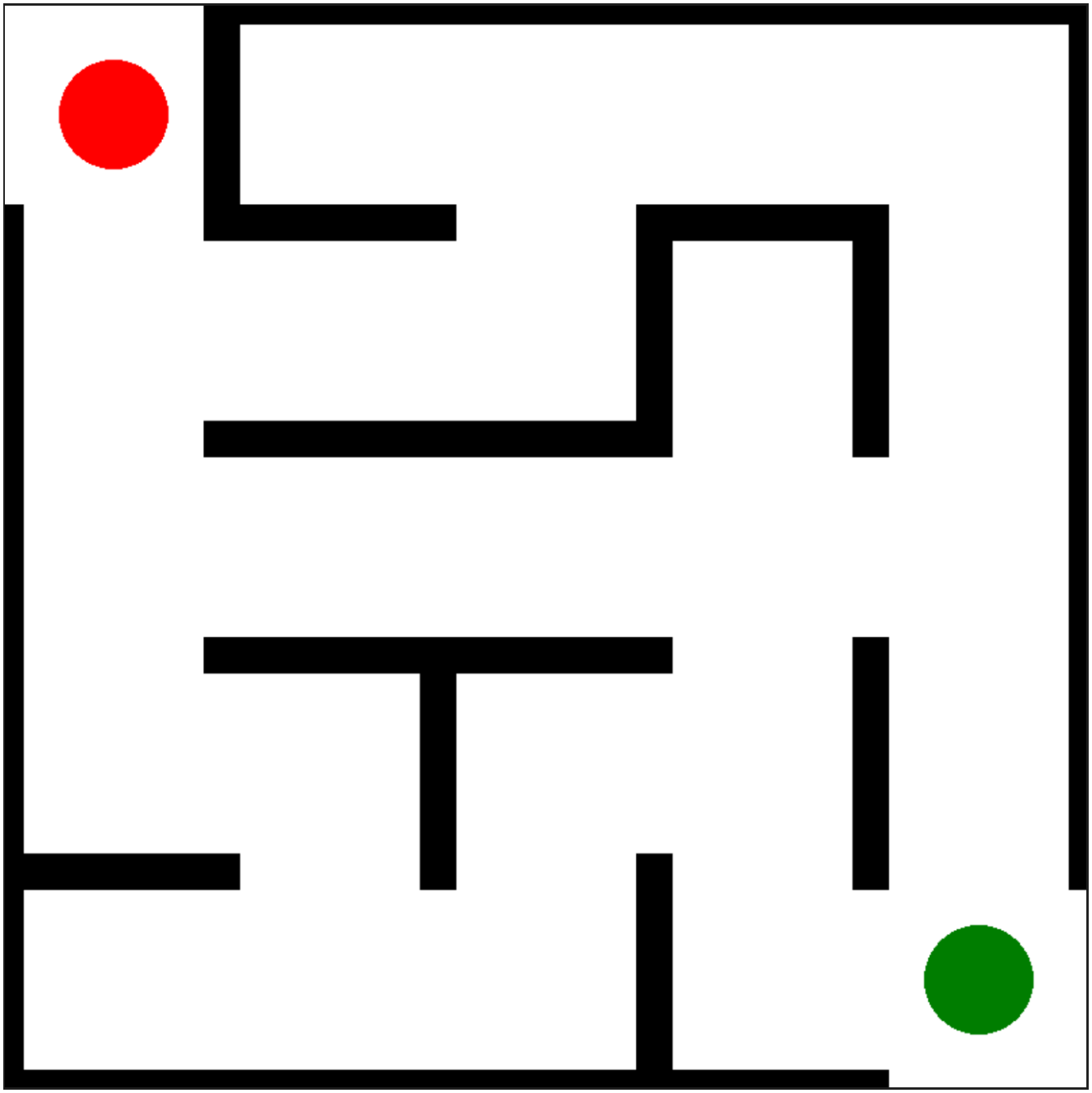
1. (1 points) **Report**: A report that describes your approach, methodology, mathematical calculations, conclusion and references. *Please make sure to follow the report format*.
2. (2 points) **Code**: All relevant code and program files used to build the digital twin and the implement the inverse kinematics and path planning.
3. (1 points) **Video**: A video of the digital twin's performance given an end-effector state and cross validation of the performance with the physical robot on the given path.

Scroll further to see the sample image of the maze.

Deadlines

Due date: **1st May 2025 11:59 PM** (To demonstrate and submit on canvas)
Last Available date: **2nd May 11:59 PM** (Submissions close)

A sample image of the workpiece is provided for your reference. This image is not to scale.



The solution of the sample image is also provided below. Notice that there are multiple solution paths. But among them there is a shortest path (green). It may be possible that there are two paths that are the shortest i.e. the green path produces two plausible solutions. You can chose either one.