

Project Deliverable H: Prototype III and Customer Feedback

Ani Freedom
Christy Lau
Paul Shedden
Claire Durand

GNG1103 Project Group 6
March 27, 2022

Abstract

On Halifax-class frigates, the Department of National Defence has a need for a robotic arm that uses inverse kinematics to paint surfaces. The robot must also scan and clean areas to identify and remove defects. To design the robot, a design process with several steps will be followed. Thus far, two separate prototypes have been made using feedback from the client and the professor. At the last stage in the design process, we have finished our final prototype and have documented the prototyping process in detail.

Table of Contents

1.0 Introduction	4
2.0 Detailed Design Drawing and Client Feedback	4
3.0 First Prototypes	8
3.1 Test Results	11
3.1.1 Open/Close End-Effector	11
3.1.2 Horizontal/Vertical Positions of a Clipboard Holding a Paintbrush	11
3.1.3 Solution to Two-Dimensional Inverse Kinematics Problem in Python	11
3.1.4 Usability of User Interface	13
3.1.5 Structural Strength of the End-Effector	14
3.1.6 Compressive Strength of the End-Effector	14
3.2 Analysis of Results	15
3.2.1 Open/Close End-Effector	15
3.2.2 Horizontal/Vertical Positions of a Clipboard Holding a Paintbrush	15
3.2.3 Solution to Three-Dimensional Inverse Kinematics Problem in Python	16
3.2.4 Usability of User Interface	16
3.2.5 Structural Strength of the End-Effector	16
3.2.6 Compressive Strength of the End-Effector	16
4.0 Current Bill of Materials	21
5.0 Conclusion	21

1.0 Introduction

The Department of National Defence expressed the need for a robotic arm with three degrees of freedom to paint areas on Halifax-class frigates. A third prototype testing plan was created based on the conceptual designs and feedback from the client. Using this third test plan, the final prototype was created and its progress was recorded.

2.0 Detailed Design Drawing and Client Feedback

Prototype designs for each subsystem were created in deliverable F. These prototypes were shown to the client and no feedback was given. Physical and more detailed prototypes were then created based off of previous designs. Figure 1 shows a detailed design diagram of the three stepper motors for controlling the degrees of freedom of the robotic arm, the ultrasonic sensor to detect possible individuals or other objects in the area, and the kill switch to turn off the arm in case of an emergency.

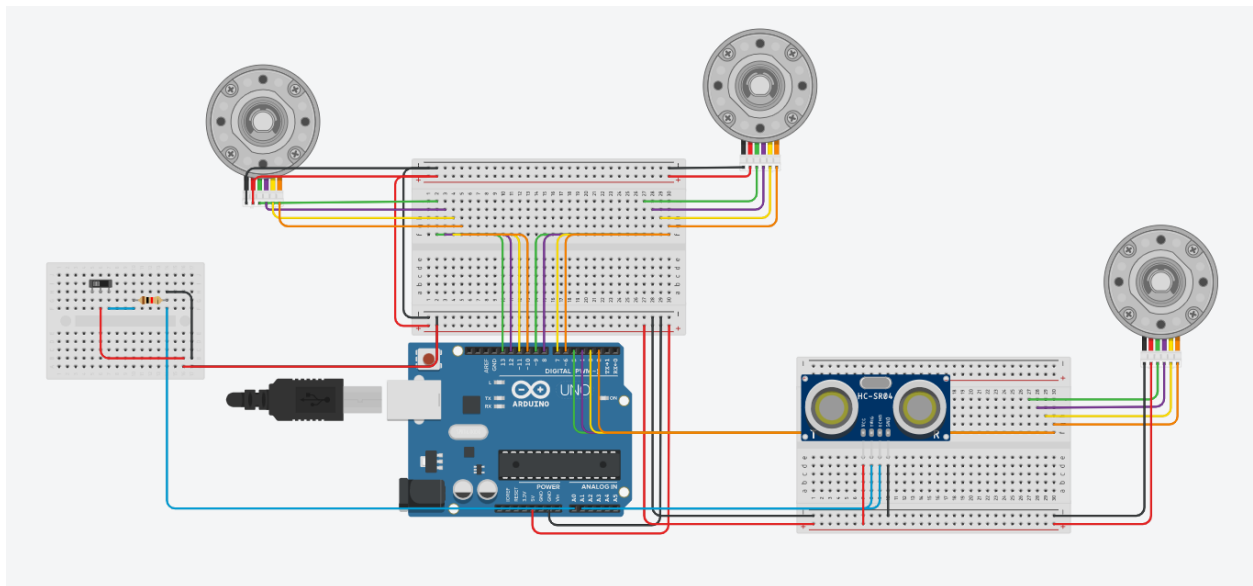


Figure 1. Detailed design diagram of Circuit for Motors and Sensors.

Figure 2 shows the full arm, with the attached 3D printed end effector prototype holding a pencil.

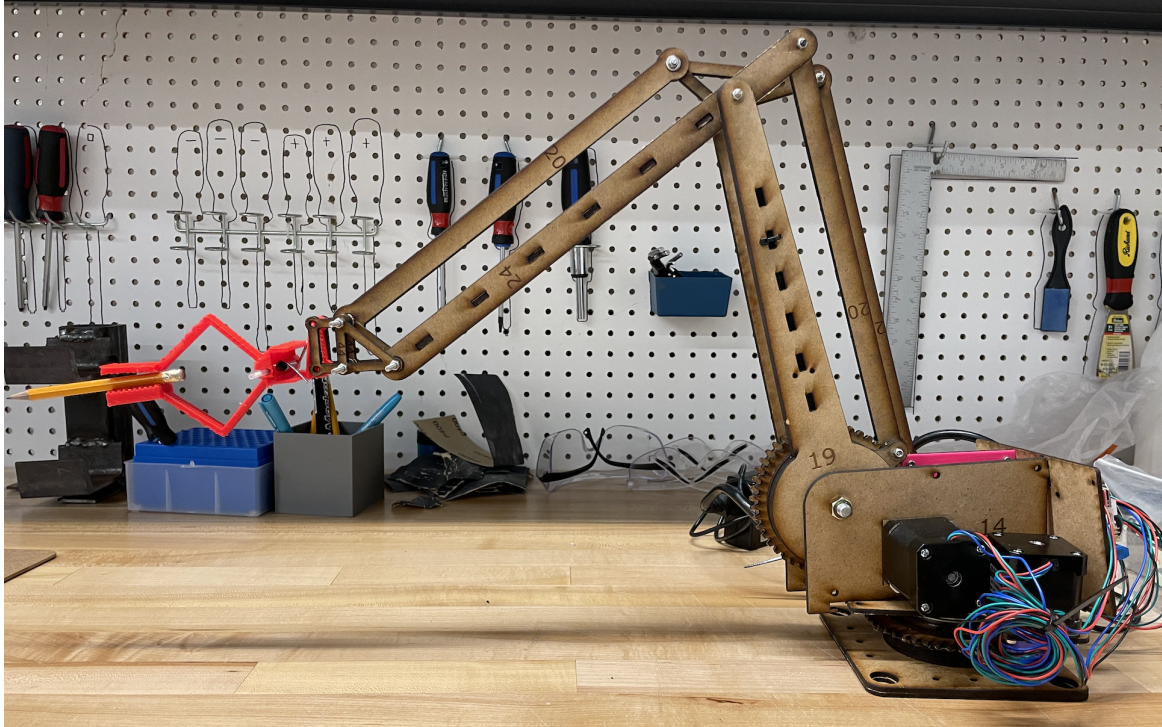


Figure 2. Robotic arm with a prototype end effector.

Figure 3 shows a close up of the full end effector prototype holding a pencil, including the spring mechanics.

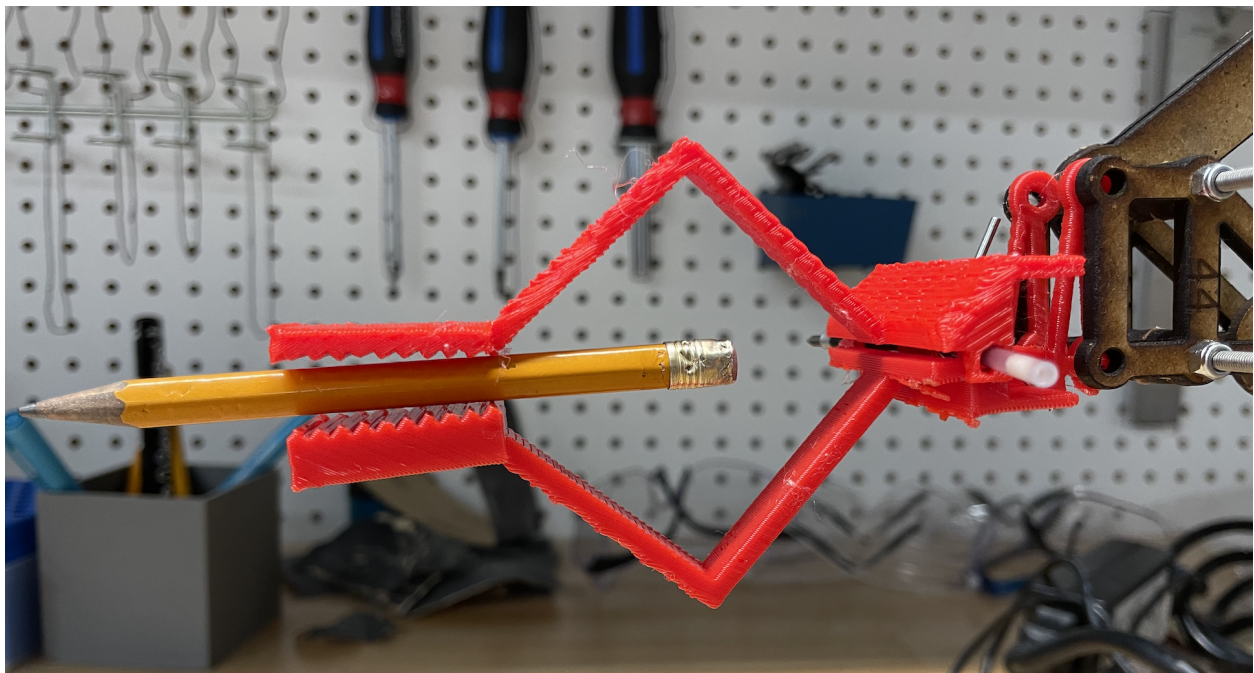


Figure 3. Close up of the end effector

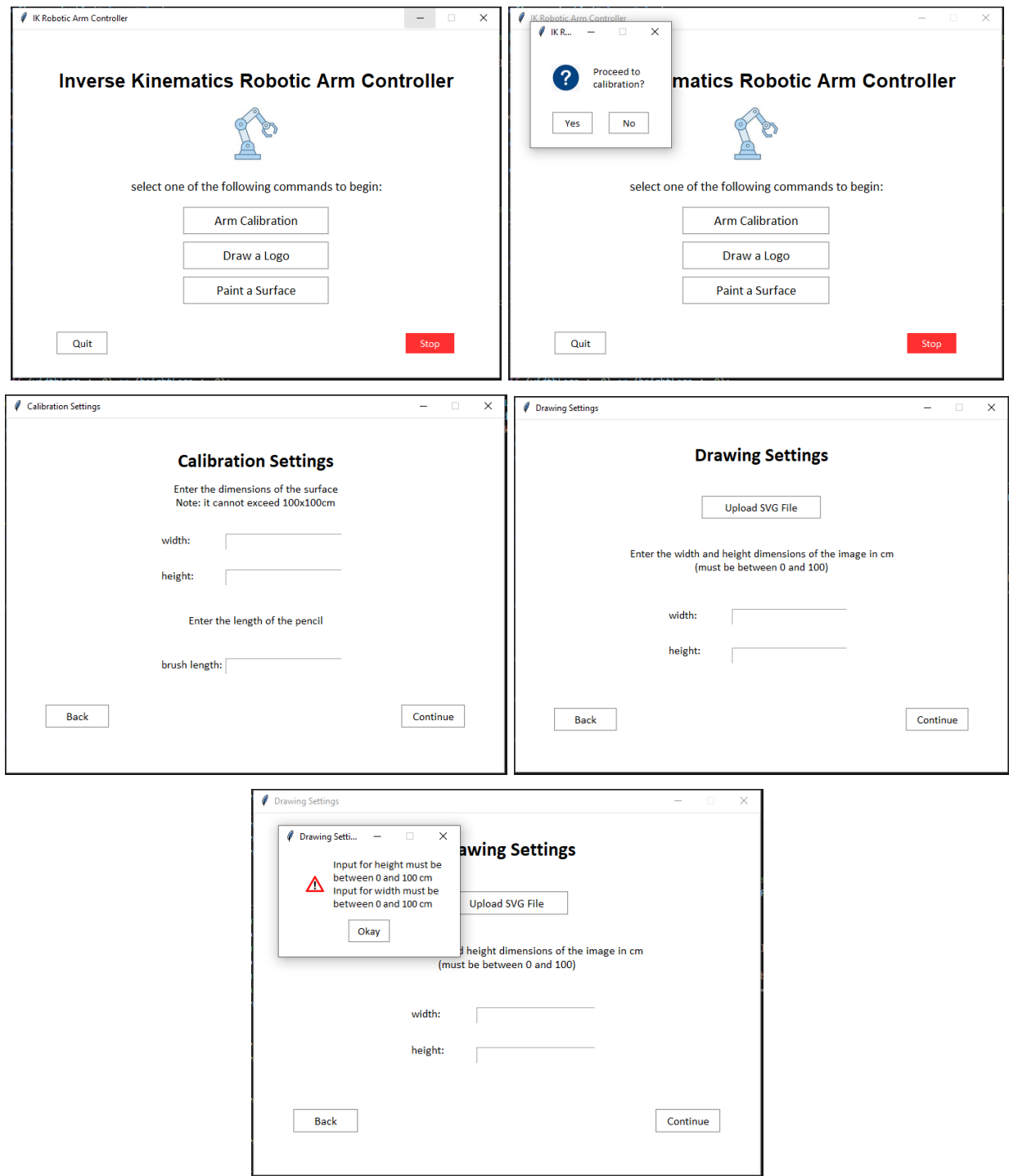


Figure 4. Detailed Images of User Interface

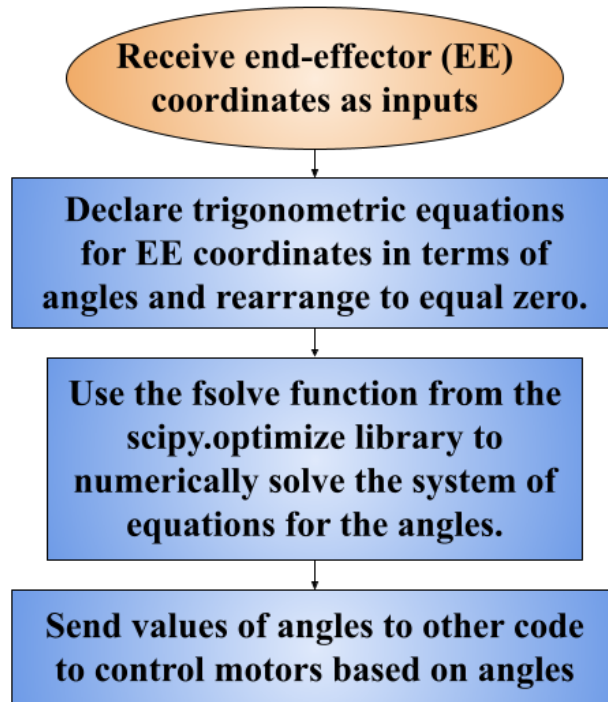


Figure 5. Flowchart of code for the inverse kinematics calculations in Python.

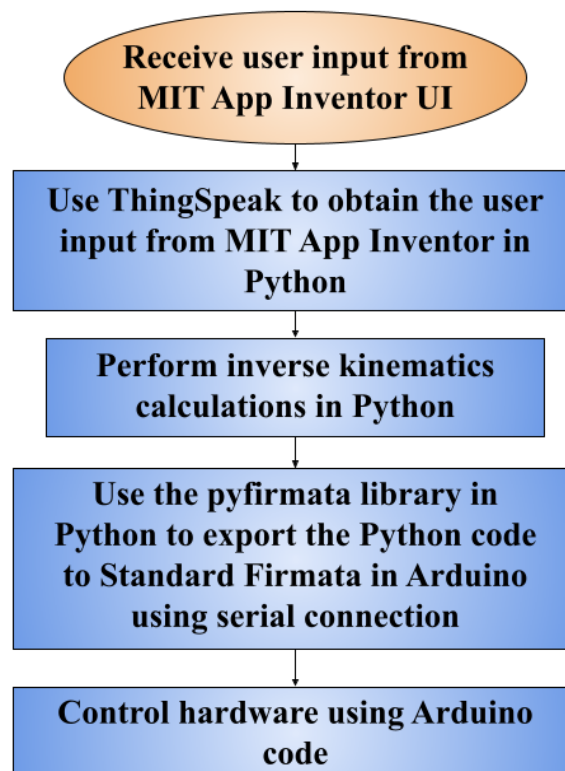


Figure 6. Flowchart of overall software subsystem from the user interface to the Arduino.

3.0 First Prototypes

This section provides background information regarding all the prototype tests completed to date. The details of each test are presented in Table 1.

Table 1. Overview Prototype Test Descriptions

Test ID	Test Objective	Description of Prototype used and of Basic Test Method	Summary of Results	Date Conducted
S1	Compute correct values for angles	Python code numerically computed a solution of a 2D inverse kinematics problem of which the analytical solution was known. Angles corresponding to a single point were solved.	<ul style="list-style-type: none">• Correct solution was obtained with fast convergence using Newton's method• The method diverged when given a bad initial guess	
S2	Compute correct values for angles	Python code numerically computed a solution of a 3D inverse kinematics problem of which the analytical solution was known. Angles corresponding to a single point were solved.	<ul style="list-style-type: none">• Correct solution was obtained with fast convergence using Newton's method• The method diverged when given a bad initial guess	
E1	Check for interference in end effector design	Top and bottom parts of the end effector were mated together in onshape using the cylindrical mate. The angles were restricted by assuming what would be realistic on a physical model.	<ul style="list-style-type: none">• The top part of the end effector could be successfully opened without hitting/overlapping the bottom end effector	2022-03-05

E2	Test the spring strength in a clipboard	A paintbrush was clamped in a clipboard and held at various angles and positions relative to the ground.	<ul style="list-style-type: none"> The clipboard successfully held the paintbrush in place without any slippage or other movements 	2022-03-06
E4	Ensure that the end-effector has enough compressive force structural integrity to lift objects such as a paint brush, nozzle, and camera.	A 3D printed end effector held various objects while being attached to the end of the arm.	<ul style="list-style-type: none"> The end-effector was able to hold the mass of a pencil and paintbrush however slippage did occur 	2022-03-11
E5	Test whether the end-effector spring has enough compressive force to hold a paint brush and a cell phone.	Someone's hand will open the end-effector and place a paint brush and a cell phone in the clamp and the spring force will hold the objects in place once the person lets go of the end-effector.	<ul style="list-style-type: none"> The end-effector was able to hold the mass of a pencil and cell phone. Since some slippage did occur, there will be something like anti slip pads for furniture to increase friction and reduce slippage. 	2022-03-11
E6	-Test whether the end-effector spring has enough compressive force to hold objects -Test if the robot arm can	Use a 3D printed prototype of the end-effector to connect to the robot and pick up an object.	<ul style="list-style-type: none"> This test was successful, the arm did not bend or break under the weight of the end effector and attached object. The end effector can attach to the arm with no 	2022-03-11

	support the end-effector and the object and that the end-effector can connect to the robot arm without any problems		issues.	
E7	Test for slippage on new end effector design	Use the new 3D printed end effector to hold and draw with a paintbrush	<ul style="list-style-type: none"> This test was partially successful, when the end effector held in it's intended rigid position no slippage occurred 	2022-03-23
U4	Test if UI is able to receive and store user input	A number was inputted into the height, width and length sections of the calibration interface to ensure that the numbers could be recorded in variables and printed to verify	<ul style="list-style-type: none"> The UI was able to successfully store and print the numbers inputted by the user to the terminal 	2022-03-11
U5	Test UI for exception handling when values are outside of range	Multiple tests were conducted to verify the exception handling of the UI: 1. A number outside of the range was inputted into the UI 2. Letters were inputted into the UI 3. No input was also tested	<ul style="list-style-type: none"> The error window successfully popped up for the three tests completed 	2022-03-11
U6	Test file upload button on drawing interface	4 SVG files were stored in different locations and uploaded to the UI and the file location was printed to	<ul style="list-style-type: none"> The files were accepted by the UI and python was able to print the file paths 	2022-03-12

		the terminal		
U7	Test UI's ability to convert SVG files to coordinates and draw an image correctly	An SVG file was uploaded into the UI and ran to show the image being drawn using turtle graphics	<ul style="list-style-type: none"> The file was accepted by the UI and the image was successfully drawn compared to the original image 	2022-03-16
U8	Test the safety button on the UI to ensure movement stops	Test U7 was repeated and the stop button on the UI was pressed	<ul style="list-style-type: none"> The turtle graphics were able to stop movement within 0.1 seconds 	2022-03-16
H3	Find the maximum radius of the robot's workspace.	To find the maximum radius, motors will be removed from the robot and the arm will be extended manually and measured using a measuring tape.	<ul style="list-style-type: none"> The maximum radius of the arm with the end effector is around 76.2cm. 	2022-03-11
F1	Test if the python code can instruct the motors to move	Using the robotic arm with motors connected to a CNC shield and Arudino's Firmata, and python's Pyfirmata libraries, the arm was instructed to move in the x-direction a total of 20 steps	<ul style="list-style-type: none"> The robot was able to move in the x-direction 	2022-03-23
F2	Test if the robot can move in a straight line	Using the robotic arm with motors connected to a CNC shield and Arudino's Firmata, and python's Pyfirmata libraries, the arm was instructed to move from one point to another	<ul style="list-style-type: none"> All parts of the arm were able to move to form the motion of drawing a straight line 	2022-03-23

3.1 Test Results

This section outlines some background information for each test and presents the results from each prototype test.

3.2.1 Open/Close End-Effector

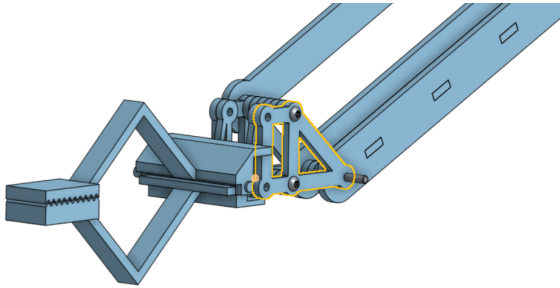


Figure 7. End effector in closed position

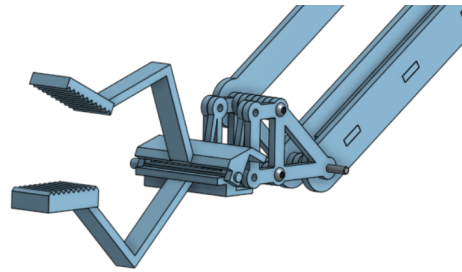


Figure 8. End effector in opened position

The purpose of this test is to check for any irregularities or overlaps when the end effector is in its open and closed positions.

3.2.2 Horizontal/Vertical Positions of a Clipboard Holding a Paintbrush



Figure 9. Clipboard and paint brush horizontal to the ground

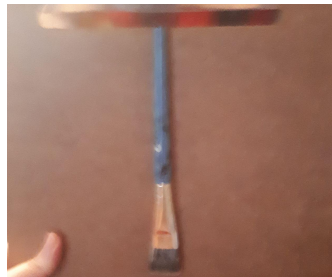


Figure 10. Clipboard and paint brush vertical to the ground

The purpose of this test is to check if a clipboard spring has the compressive strength to hold a paintbrush.

3.1.3 Conversion of Angles to Discrete Steps

A crucial step in the implementation of the inverse kinematics solution is to ensure that the results from the solver can be used to control the arm. The angles on the robot are controlled by stepper motors, which can only rotate in 1.8° steps. Because of the discrepancy between the

continuous angles solved by Newton's method and the discrete steps the motors can take, a function to convert the angles to steps was necessary to implement.

A function named *convert* was implemented in the Python code. The function takes in a list of lists of angles and returns a list of lists of steps for the motors. The sign of the steps indicates the direction to turn the motors and the magnitude specifies the number of steps a given motor should take. To find the magnitude, the function simply multiplies each angle in radians by the conversion factor, and the sign of the angle is already specified, so no treatment is required regarding the angle.

$$N_{steps} = \Delta \theta \cdot \left(\frac{100 \text{ steps}}{\pi} \right)$$

To test this function, a tiny simulation was created where the end-effector must move in a horizontal line and then return to its starting position by tracing the line in reverse. The results from this test showed that the motor at the base took six steps in one direction and only five steps when returning to the starting position.

Analysis of this test is discussed in Section 3.2.3 and explains that the initial implementation of the *convert* function induces significant rounding error. The analysis section discusses the logical basis of the solution to the rounding error that was implemented to correct this problem, which is governed by the following equation.

$$\theta_{i+1} = \theta_i + \frac{\pi}{100} \cdot N_{steps}$$

To test the corrected *convert* function, coordinates for a star were passed into Newton's method to solve for angles, and the angles from Newton's method were passed into the *convert* function to solve for the number of steps required for the end-effector to move between each point. The test was designed so that the end-effector must return to the starting point because if the rounding error continues to add instead of being corrected, the robot would not return to the starting point.

The results indicated that the test of the function was successful and the rounding errors are minimised. Analysis of the results are presented in Section 3.2.3.

3.1.4 Functionality of the UI

Test U7: [Extracting coordinates from the SVG files and Drawing Graphics](#)

Using the SVG library on Arduino, the points of each individual segment were extracted and the image was split up into 1000 separate points. Using the turtle library, a window was created and

the turtle was instructed to draw segments from the end of one point to the beginning of the next point, using the first point as the starting coordinates for the turtle.

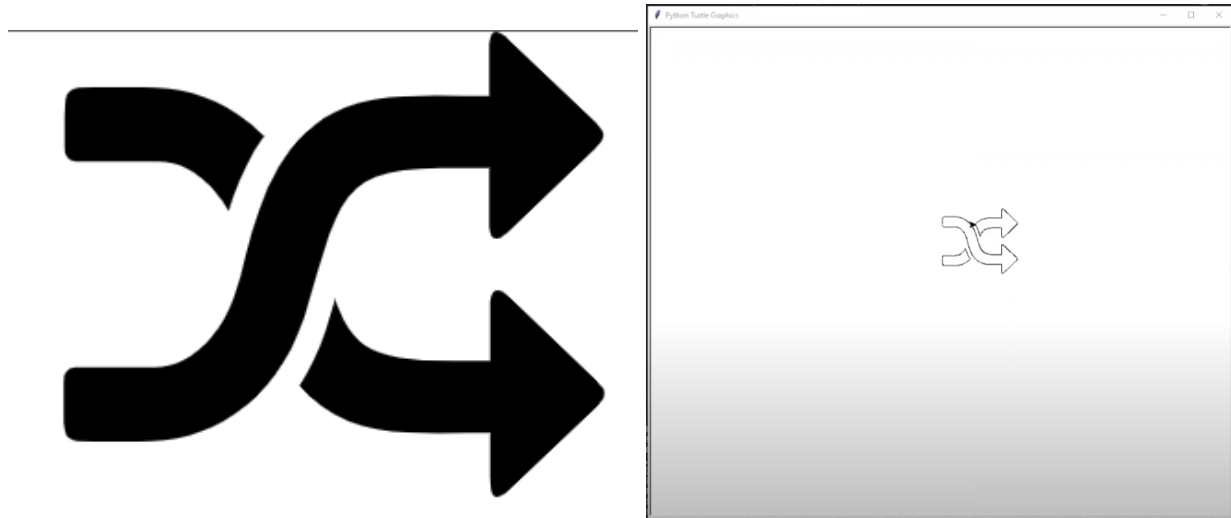


Figure 11. Comparison of Image of Graphic from the Original SVG File (left) and the Image Drawn Using Turtle Graphics Using the Coordinates of the SVG File (right)

Test U8: [Testing the Stop Button on the User Interface](#)

In Test U8, test U7 was repeated using the same SVG file. Around the midpoint of the drawing, the stop button on the UI was pressed to ensure that the turtle was able to stop moving, given commands from the UI, which can later be translated into commands to the robot.

3.1.5 Structural Strength of the End-Effector

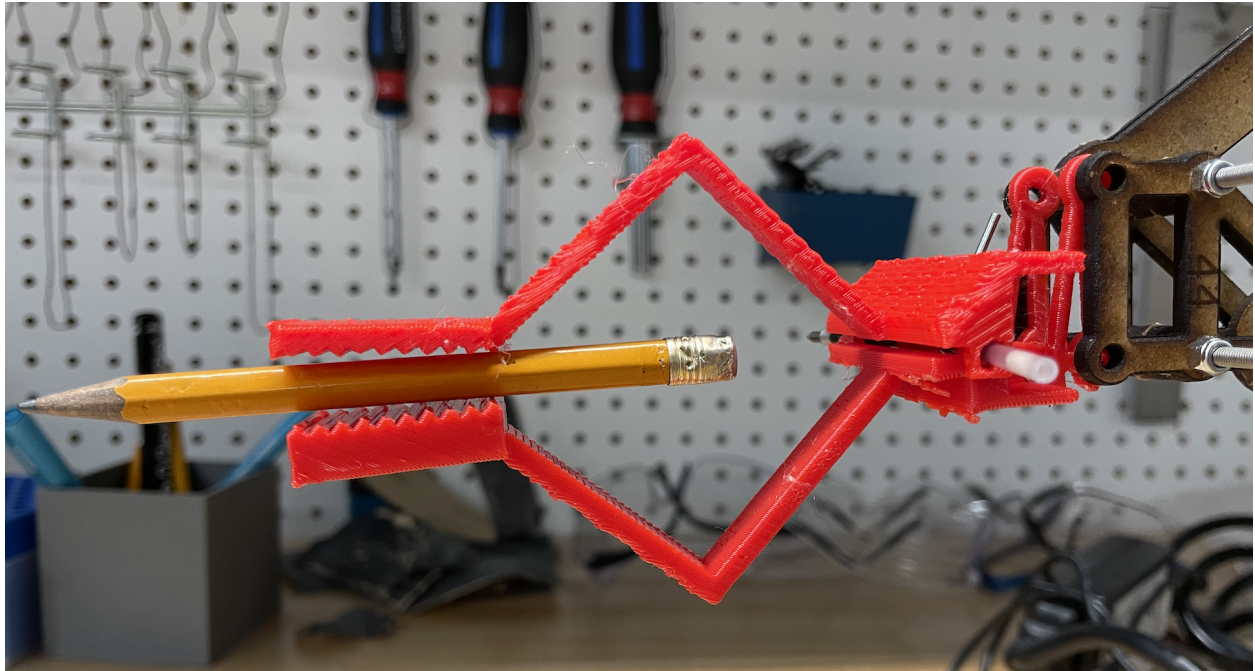


Figure 13. The end-effector holding a pencil

This test was to determine if the end-effector has the structural integrity to hold objects in place without having damage occur to the object or end-effector. A maximum safe mass was then going to be chosen based on the results of the test.

3.1.6 Compressive Strength of the End-Effector

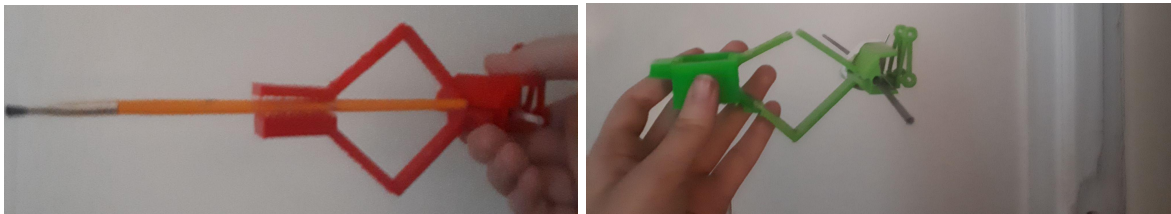


Figure 14. The end-effector holding a paintbrush and broken EE2

The purpose of this test was to determine whether the compressive strength of the spring was strong enough to hold an object in the end effector without slippage occurring.

3.1.7 Robotic Arm Supporting End-Effector and Object



Figure 15. The end-effector holding a paintbrush while being attached to the arm
The purpose of this test was to determine if the arm had the structural strength to hold the combined masses of the end effector and another object.

3.1.8 Functionality of the Overall Arm

Test F1: [Controlling the movement of the motors](#)

The objective of Test F1 was to test if the arm could be controlled using Python code instead of Arduino code. In test F1, the standard Firmata library was uploaded to the Arduino, and using the Pyfirmata library in Python, commands shown in Figure 16 were given to the Arduino board to move the motor at the base of the arm a total of 20 steps in the clockwise direction.


```

1  from pyfirmata import Arduino, util
2  from time import sleep
3  import newtonsMethod3D
4
5  board = Arduino("COM4") #assigns the port connected to the Arudino board
6
7  #enables pin on Arduino as output pins
8  stepX = 2
9  stepY = 3
10 stepZ = 4
11 dirX = 5
12 dirY = 6
13 dirZ = 7
14 enable = 8
15
16 board.digital[enable].write(1) #set the enable pin to high
17
18 board.digital[dirX].write(0)
19     for x in range(20):
20         board.digital[stepX].write(1)
21         sleep(0.001)
22         board.digital[stepX].write(0)
23         sleep(0.001)
24

```

Figure 16. Sample code for Movement of Motor in the x-direction a total of 20 steps.

The success of the test was measured qualitatively by visually determining whether the arm rotated clockwise at the base. The results of this test were essentially binary: the code would work, or not work. If the code worked, then more complex tests with higher fidelity would be executed. If the test failed, debugging would take place to try to solve the error. The results of this test are shown in the video at the top of this section and the analysis of this result will be discussed in Section 3.2.9.

Test F2: [Moving the arm in a horizontal line](#)

In Test F2, the standard Firmata library was uploaded to the Arduino. Using the Newton's Method function, which was previously written, the function was given the start and end coordinates of (0.1, 0.01, 0.1) and (0.1, 0.1, 0.1), respectively, and returned a set of angles that the robot needed to be moved to in every direction. Using the convert function that was previously written, the angles were converted into steps that the robot needed to take with the positive and negative signs representing the counterclockwise or clockwise directions of the motors. In this case, the given steps for each direction were (218, -1, -30). These steps were given to the arduinoTest.py file that instructed the angles to move using the Pyfirmata library and the robot was able to move in a diagonal line.

This test (F2) has higher fidelity than Test F1 because as opposed to indicating exactly how many steps each motor should take, this test aims to draw a line from one point to another, where the points are specified as coordinates, passed through Newton's method and the convert function, and then to the pyfirmata code. This test was partially successful because as seen in the video, the movement of the arm corresponds to the number of steps indicated (218, -1, -30), however, there was a problem with the coordinates, which will be analysed in Section 3.2.10.

3.2 Analysis of Results

3.2.3 Conversion of Angles to Discrete Steps

The results from the initial test where the end-effector had to move in a horizontal line and return to its starting position showed that the motor at the base took six steps in one direction and only five steps when returning to the starting position. If the function were correct, when returning, the motor should have taken six steps to return to its initial position. This incorrect result is certainly caused by rounding error.

If an angle is between 0° and 1.8° , such as 0.9° , the code will have to round the value of the angle to the nearest step. This rounding error may not be problematic when applied to one angle, but with hundreds of angles, the error of a given error will be the sum of errors of all prior angles.

To prevent rounding errors from carrying over to other angles, the solution to this rounding error makes use of the fact that lists in python are mutable. The function always calculates the difference between two consecutive angles in steps, given by the following equation.

$$N_{steps} = \Delta \theta \cdot \left(\frac{100 \text{ steps}}{\pi} \right)$$

The solution to the rounding error is that after the number of steps between two angles is calculated, the value of the second angle is corrected based on the number of steps calculated, shown by this correction equation.

$$\theta_{i+1} = \theta_i + \frac{\pi}{100} \cdot N_{steps}$$

This correction ensures that even if the position of the end-effector is incorrect due to rounding error, the next calculation considers the actual position of the end-effector, as opposed to considering the position at which the end-effector should be. This correction is governed by the following equation.

The results of the star test (described in Section 3.1.3) were analysed by computing the sum of all the steps taken by one motor. If the sum of the steps taken by that motor is zero, then that motor has returned to its initial state. If all three motors have returned to their initial states, then the end-effector has returned to its initial position and the correction implemented in the convert function is successful.

The test for the star was successful, so other tests where the end-effector must move in arbitrary geometrical patterns and return to its starting position were generated to further test the functionality. The results from all other tests indicated that the end-effector returned to its initial position, so the prototype test was concluded to be successful.

3.2.4 Functionality of User Interface

Test U7 was conducted on the user interface to ensure that it was able to take an SVG file and extract the correct coordinates to draw the image using turtle graphics. From the results shown in section 3.1.4 of this report, the UI was able to draw the image using turtle graphics to a high degree of accuracy, as shown in the side by side comparison of the image from the SVG file and the image from the turtle graphics, shown in Figure 11.

3.2.5 Safety Features on User Interface

Test U8 was conducted to determine if the safety features on the user interface were able to quickly and safely stop the movement of the robot. The test results for Test U8 shown in section 3.1.4 of this report show that the stop button on the UI was able to successfully stop the movement of the turtle graphics, demonstrating the feasibility of the UI.

3.2.6 Structural Strength of the End-Effector

The end-effector was able to hold the mass of a paintbrush and pencil even after suffering structural damage. This leads the group to assume that similar masses should be acceptable with a structurally intact end-effector. However, during the tests slippage did occur while holding a paintbrush and pencil. Less slippage occurred with a cell phone. These results may be caused by the different centres of mass of the various objects as well as the shape contrast.

3.2.7 Compressive Strength of the End-Effector

The end-effector was able to hold a pencil and paintbrush in place, however there was slippage that occurred. The end effector will be modified to minimise possible slippage. A cell phone was also held by an end-effector. This test will not be conducted while attached to the arm due to the mass of the average cell phone in the group exceeding the maximum load capacity of the arm (without including the mass of the end effector).

The second end effector prototype successfully held a paintbrush without slipping when it was in

it's intended formation, however some breakage occurred while removing the supports from the print, this was partially fixed with duct tape and will soon be with crazy glue to ensure a stronger more rigid body. The take away from these tests is that the time should've been taken to print with a smaller nozzle size to ensure better quality and finer prints.

3.2.8 Robotic Arm Supporting End-Effector and Object

The robotic arm was able to be successfully connected to the end-effector. While attached the end-effector was easily able to open and close to accommodate for the placement or removal of an object. The robotic arm was able to withstand the weight of the end-effector in addition to the object the end-effector was holding.

3.2.9 Controlling the Movement of the Motors

The motors of the arm were able to be controlled using the Python and the Pyfirmata library, as shown in section 3.1.8, however, the initial movements of the arm were slow and choppy and unsuitable for drawing images. To fix the issue, the delay between the pulses given to the motor were shortened to make the robot move as smoothly as possible.

3.2.10 Moving the Arm in a Straight Line

As seen in the video in Section 3.1.8, the arm rotates mostly in at the base and at the top-most point. This qualitative observation matches the steps specified by the code: (218, -1, -30). The match indicates that the connection between the Python code and the arduino using the pyfirmata library works well. However, an inaccuracy in this test is that the coordinates given to Newton's method were not in the robot's workspace, so the inverse kinematics solution diverged and found an incorrect solution.

From this test, several learning points were obtained. One was confirmation that the connection between Python and Arduino works, but to test more rigorously in future tests, coordinates that have known solutions should be tested, as well as any other parameters that are used so that the test is as isolated as possible. Once all components of the system are known to be working, then a comprehensive test can be executed. Given the result, this comprehensive test was clearly premature. Another learning point was that the inverse kinematics solution should be updated to include error handling, so that invalid coordinates cannot be passed into Newton's method.

After analysing the results from this test, future steps to improve the code are to make the rates at which the motors turn more general to any case. To draw a diagonal line, the motors have to appear to be moving simultaneously at the correct rates. To achieve this result by rotating the motors using serial commands, an algorithm to determine how often each motor should step must be developed.

Once the rates of rotation have been generalised, the next prototype test would be to pass an SVG file into the GUI and have the robot draw on a surface using the end effector and a marker. This future test will be the most comprehensive so far.

4.0 Current Bill of Materials

Some small changes have been made to the bill of materials bringing the total down to \$1.75, which is under the \$50 limit. At the time it was created, an Arduino Uno and shield were included; they have since been taken out after discovering that they are given in class. See [spreadsheet](#) for details.

5.0 Conclusion

The Department of National Defence has a need for a robotic arm that uses inverse kinematics to paint surfaces. In this deliverable, the last prototype was finalized using feedback from the user and the prototype test plan was completed. The next steps of this project are to complete a detailed user manual.