GNG 1103

Design Project User and Product Manual

Inverse Kinematics Robotic Arm

Submitted by:

GNG1103 C06

ANI PREEDOM, 300253240

CLAIRE DURAND, 300246672

CHRISTY LAU, 300172192

PAUL SHEDDEN, 300195267

April 9, 2022

University of Ottawa

List of Figures

List of Tables

List of Acronyms and Glossary

Table 1. Acronyms

Acronym	Definition
EE	End Effector
IKRA	Inverse Kinematics Robotic Arm
UI	User interface
UPM	User and Product Manual
3DOF	Three degrees of freedom

1 Introduction

This User and Product Manual (UPM) provides the information necessary for users with high school level education to effectively use the Inverse Kinematics Robotic Arm (IKRA) and for prototype documentation. This product has been designed for the Department of National Defense; they had a need for a safe, light, and durable robotic arm that uses inverse kinematics with 3 degrees of freedom (3DOF) to paint logos on ships. The robot must also be easy to use and maintain by a single user.

This document has been designed for users of the robotic arm and anyone who wished to recreate it. It starts with an overview where a brief outline of the product is given. Including a full image of the arm and the major functions of the product. The next section will give a general walkthrough of the system from setup to exit. Pictures will be included for better understanding. This section will also contain configuration considerations and system organization. From there it will explain how to use the different functions of the robot such as drawing and painting. Next is a section on troubleshooting containing information on error messages, special considerations, maintenance, and support. This will have details in case anything goes wrong in the construction or use of the arm. The final and arguably biggest section will document the progress of the arm including the separate prototypes that were made and the testing that was completed. The purpose of the UPM is to inform the user of all the necessary information and give a complete overview of the product so reproducibility is possible.

2 Overview

The Department of National Defense has a need for a robotic arm with 3 DOF that uses inverse kinematics to scan areas for corrosion and paint logos on Halifax-class frigates. Surveying and painting by hand takes time and costs money. With this arm, they can train an individual to start the program, this person can then leave and check on it periodically. Since it will not have to be monitored constantly this will save time and money. The main focus of this project was the inverse kinematics problem, this was needed for easy movement of the arm and so the robot could use pictures uploaded into the user interface to draw logos. Safety is extremely important with this product as well, meaning the arm can not be too heavy and must be easily turned off if need be. It must be reproducible using 3D printing and conveniently used by one person with a high school level education.

The EE designed in this product manual is different from other products because it is under 30g, can be used for several different needs and is user friendly. Since the EE only uses 3D printing and a spring from a clipboard it is also inexpensive. The code described in this document can also be scaled to more degrees of freedom. The robot given only has 3DOF, however if need be it can be easily changed to have 6 or more. This gives a lot more liberty with the code, since it does not necessarily only have to be used with this robotic arm. It would only have to be calibrated with the measurements of the new arm.

In this document the product that has been developed over the past couple of months will be described in detail. Including how it was developed, tests that were done and how to work the software itself.

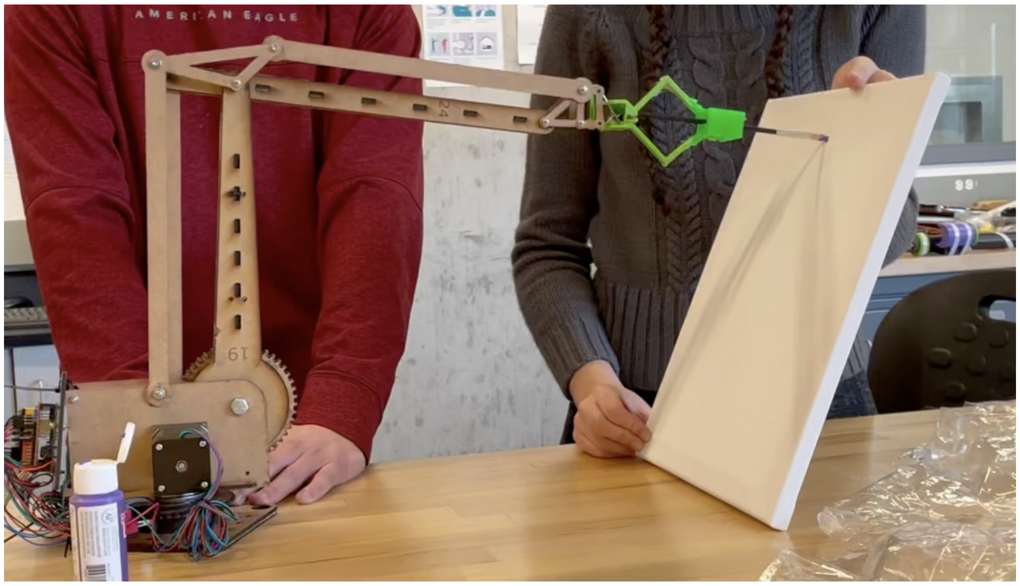


Figure 1. Final Prototype with end effector holding a paint brush

The goal of this project was to make an end effector that could be used in multiple different ways, this was achieved by creating a versatile design on Onshape which was then 3D printed. The design is a type of clamp containing a spring that can hold different objects such as a paint brush. A user interface was created that can be controlled from a computer to start and stop the program and enter images for the robot to paint. This flowchart demonstrates the basic functionality of the UI.

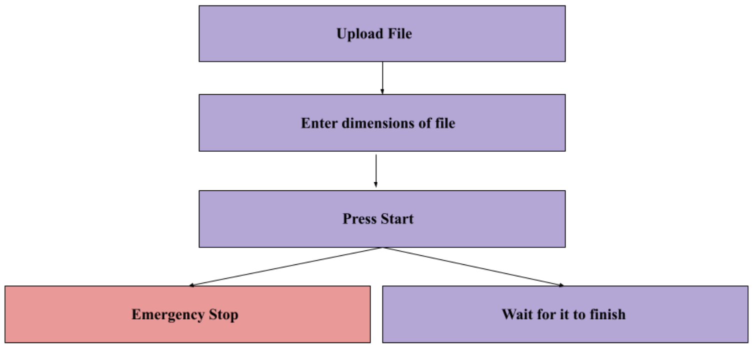


Figure 2. Basic flowchart of UI

3 Getting started

3.1 Configuration Considerations

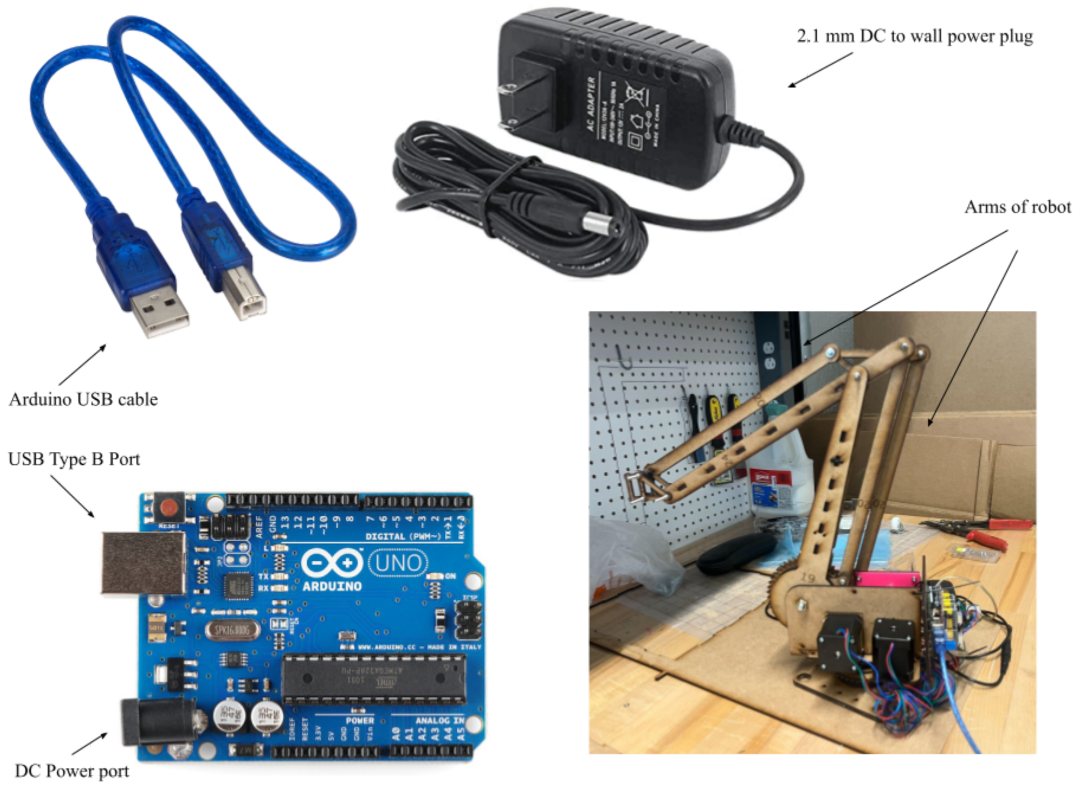


Figure 3. Parts necessary for installation of robot

In order for the user to control the robot, the motors of the robotic arm must be connected to the Arduino controller, which is connected to the computer through the Arduino USB cable and a power source through the DC power port. To properly run the program, the user must also download and install Python as outlined in section 6.3.1 “Installing the Software”

3.2 User Access Considerations

This product has been designed to be used with any user that has a high school level education. This does not provide many restrictions to the use of the arm, however the reach of the robot is 32.5cm, this places restrictions on what the arm is capable of. It has been designed for employees to paint logos, this gives the arm freedom to do lots of different tasks in many different projects. Any business may use this arm, the only restrictions would be the size of the arm.

3.3 Accessing/Setting up the System

1. Set the robot down on its base at the desired position and ensure that both arms of the robot are positioned as shown in the image below

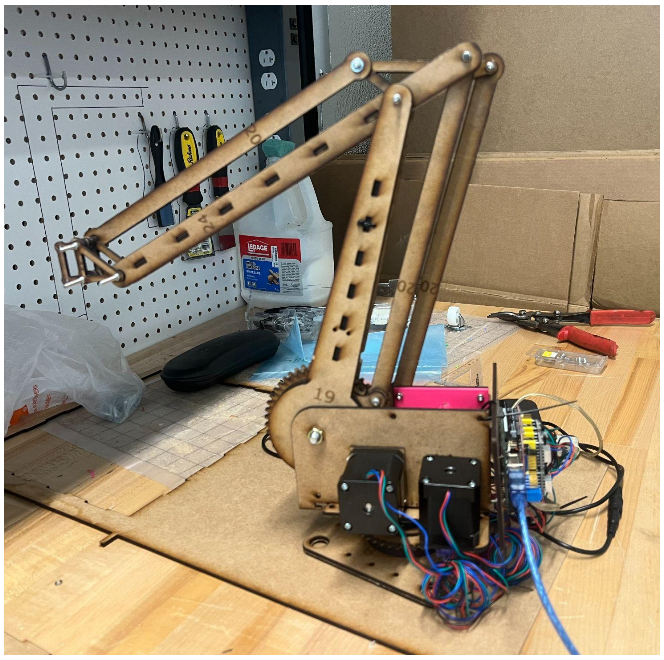


Figure 4. Robotic arm with both arms in the start position.

2. Attach the desired brush into the end effector by pressing down on points A and B and inserting the brush into the claw side of the end effector, ensuring that the tip of the brush is facing outwards

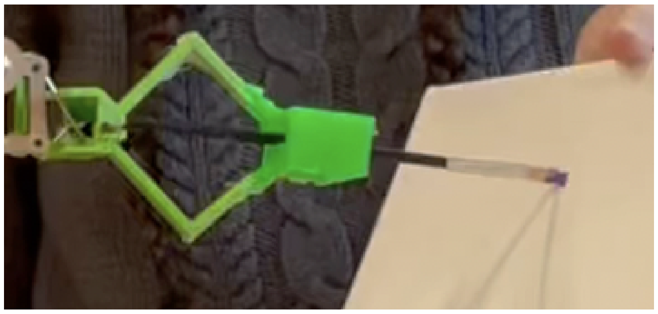
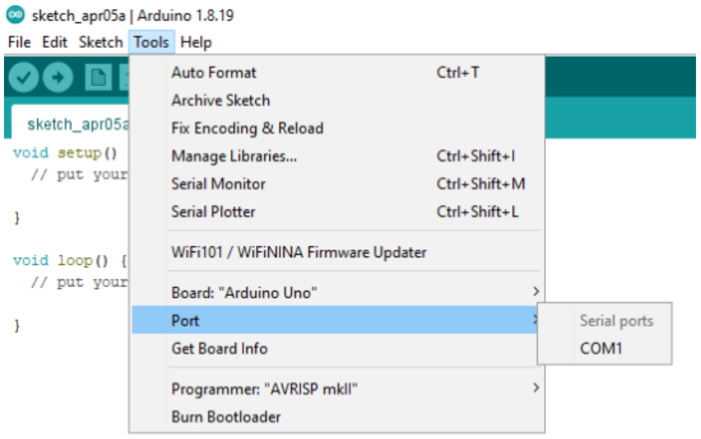


Figure 5. Example of EE holding a paintbrush properly

3. Plug the Arduino USB cable into the USB Type B port on the board and plug the other end of the cable into the USB port on your computer
4. Plug the round end of the 2.1 mm DC to wall plug into the power port on the Arduino board and plug the other end into a wall outlet

5. Open the Arduino application on your computer
6. On the top menu bar, navigate to Tools > Port and record the name of the available port

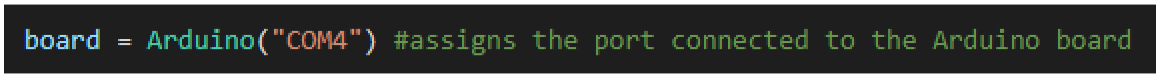
   For a Windows operating system:


sketch_apr05a | Arduino 1.8.19
File Edit Sketch Tools Help
sketch_apr05a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM1

```
void setup()
  // put your

}

void loop() {
  // put your

}
```

Figure 6. Windows Operating System Port

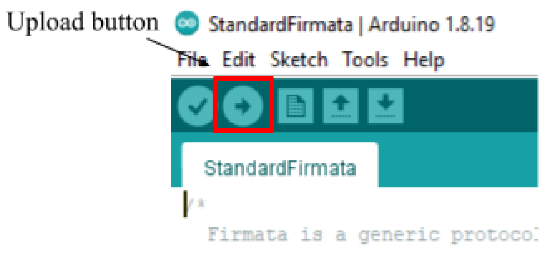
7. Open the file labelled arduinoControl.py and locate line 5:



```
board = Arduino("COM4") #assigns the port connected to the Arduino board
```

Figure 7. Line 5 of Arduino Code

8. Change COM4 to the correct name for your system (For Windows OS, it should be labelled COM followed by a number, for MacOS, it should be labelled /dev/cu. usbmodem…
9. Connect the arduino board to a wall plug using the power port connector
10. On the Arduino application, select File > Examples > Firmata > StandardFirmata
11. Click on the upload button located in the top corner of the Arduino Application



```
/*
  Firmata is a generic protoco
```

Figure 8. Upload button on Arduino

12. Wait until the file is finished uploading to the Arduino board
13. Exit out of the Arduino Application by clicking on the "X" in the top right corner of the screen
14. Open the file labelled mainTest.py. The following application should open

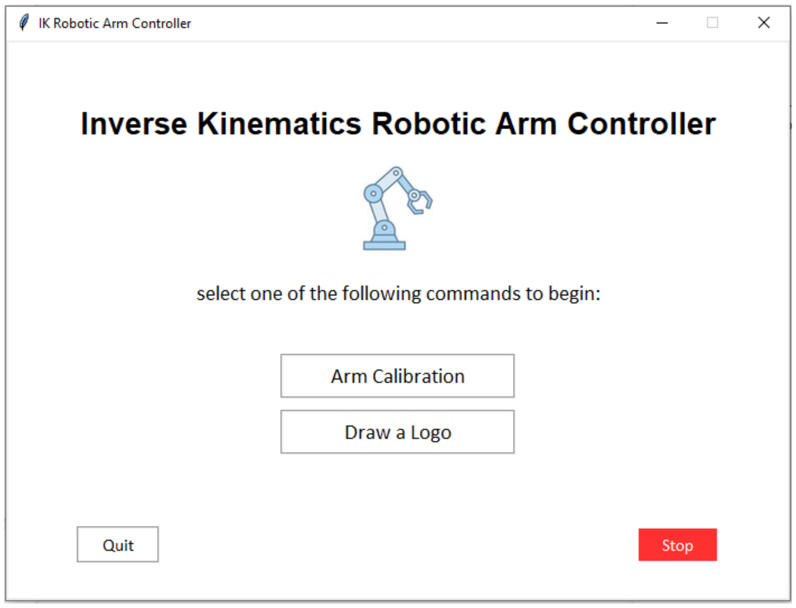

IK Robotic Arm Controller
Inverse Kinematics Robotic Arm Controller
select one of the following commands to begin:
Arm Calibration
Draw a Logo
Quit
Stop

Figure 9. Application that will appear when file mainTest.py is opened

3.4 System Organization & Navigation

3.4.1 Home Page

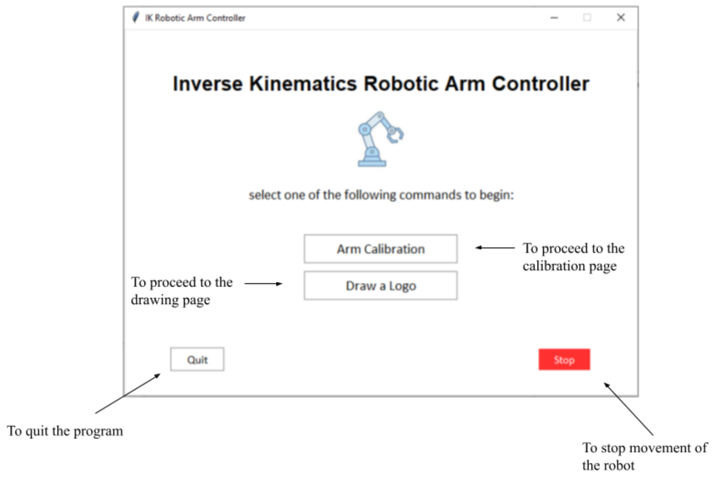
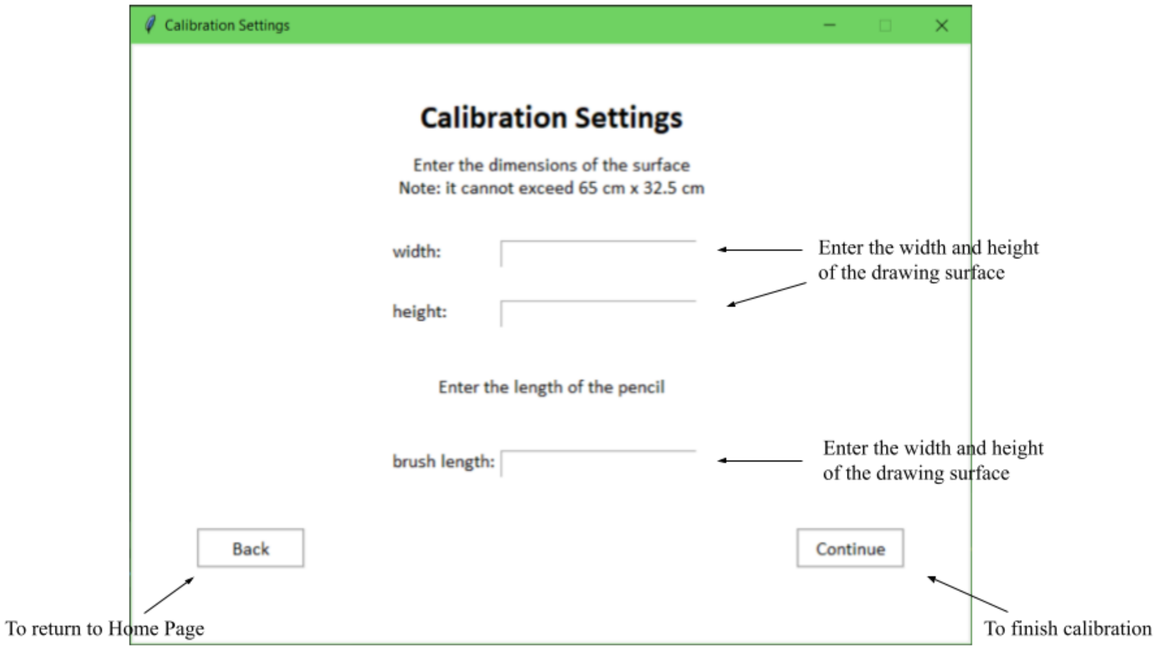

IK Robotic Arm Controller
Inverse Kinematics Robotic Arm Controller
select one of the following commands to begin:
Arm Calibration
To proceed to the calibration page
To proceed to the drawing page
Draw a Logo
Quit
Stop
To quit the program
To stop movement of the robot

Figure 10. Home Page Layout of System/User Interface

From the home page of the system, the user may navigate to different sections and discover different functions of the robotic arm. The user can choose to either calibrate the arm or draw using the arm. The user can also exit the program by selecting the quit button of stop the robot in motion by selecting the stop button

3.4.2 Arm Calibration


Calibration Settings

Calibration Settings

Enter the dimensions of the surface
Note: it cannot exceed 65 cm x 32.5 cm

width:

height:

Enter the length of the pencil

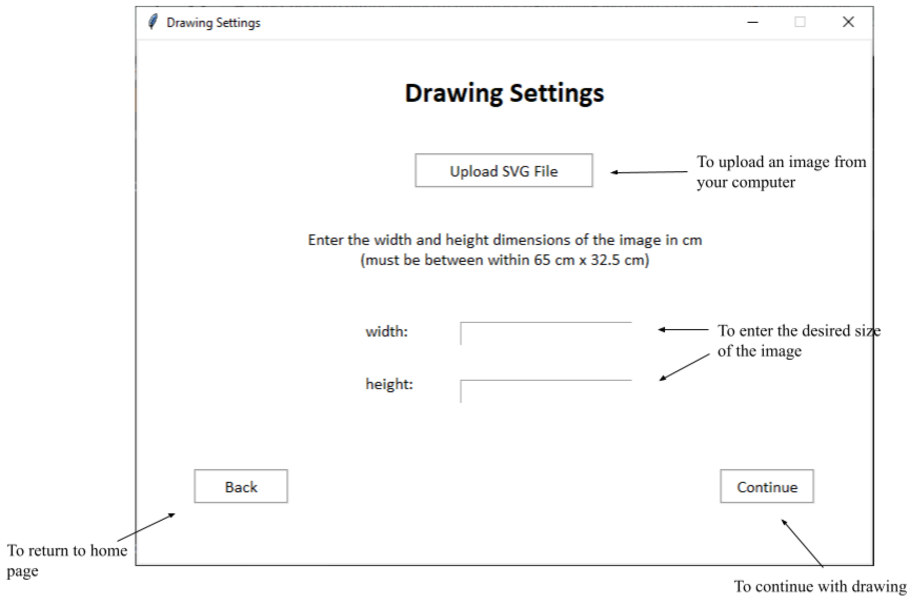
brush length:

Enter the width and height of the drawing surface
Enter the width and height of the drawing surface
Back
Continue
To return to Home Page
To finish calibration

Figure 11. Layout of Calibration Page of System/User Interface

The calibration page allows the user to calibrate the product. The user may change the width and height dimensions of the drawing canvas, within a range of 0-65 cm in width and 0-32.5 cm in height. The user may also enter the length of the drawing instrument to ensure that the arm is able to draw on the desired surface with the desired brush. From the calibration page, the user may return to the home page by selecting the “back” button or continue with the calibration by selecting the “continue” button

3.4.3 Drawing a Logo


Drawing Settings

Enter the width and height dimensions of the image in cm
(must be between within 65 cm x 32.5 cm)

Figure 12. Layout of Drawing Page of System/User Interface

From the drawing page, you may upload an SVG file with your desired image. Selecting the “Upload SVG File” button will take you directly to the file explorer on your device to upload your desired image. The width and height of the image can also be added to scale the image to your desired size. From the drawing page you may choose to return to the home page by selecting the “back” button or continue to draw by selecting the “continue” button.

3.5 Exiting the System

To Exit the System:

1. From the home page, click the ‘Stop’ button to ensure the robot is not in motion
2. Click the ‘Quit’ button to exit the program
3. Unplug the robot from the power source
4. Unplug the USB from the computer and the Arduino board

5. Reset both arms of the robot to the following position

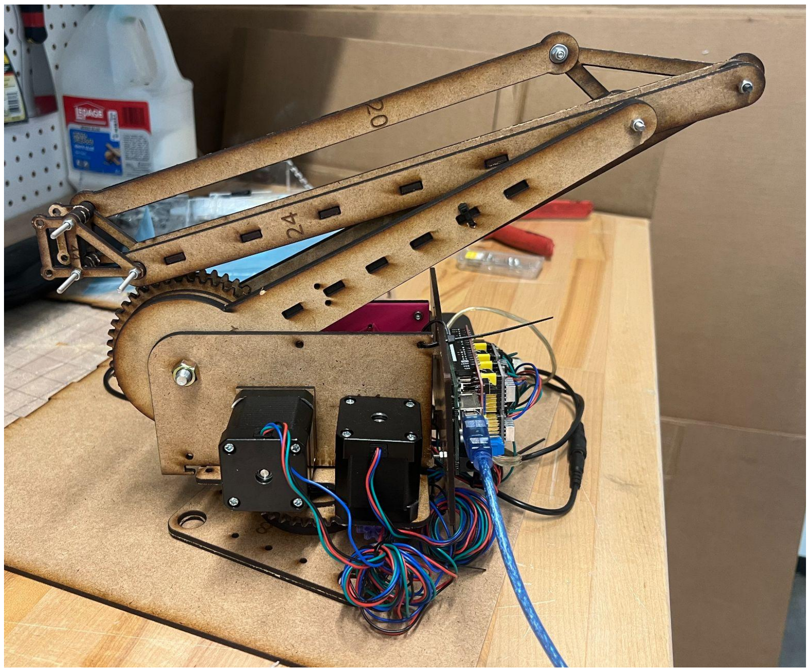


Figure 13. Robotic arm with both arms set down in position for storage

The robot is now ready for movement and storage

4 Using the System

The following subsections provide detailed, step-by-step instructions on how to use the various functions or features of the Inverse Kinematics Robotic Arm.

4.1 Calibrating the Arm

To calibrate the arm to a specific surface, the following steps need to be taken:

1. Measure the area of the surface that you wish to draw or paint on
2. Measure the distance from the end of the end effector attached to the arm to the tip of the brush

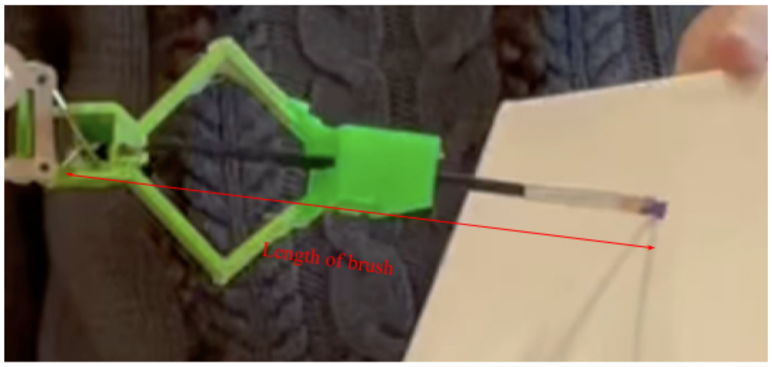


Figure 14. Example of how to measure the length of the end effector

3. Navigate to the Home Page of the User Interface
4. Select the “Arm Calibration” setting and select “yes” to continue
5. Enter the measurements for the area of the surface into the user interface in centimetres (cm)
6. Enter the measurements for the length of the brush (measured in step 2) into the user interface
7. Select “Continue” to proceed with the calibration

4.2 Using the Arm to Draw/Paint

To draw or paint using the arm, follow the upcoming steps to proceed to the drawing settings:

1. Follow the instructions in section 4.1 to calibrate the arm to the desired surface and brush
2. Return to the homepage of the user interface
3. Select the “Draw a Logo” setting and select “yes” to continue

4.2.1 Drawing a Logo

To draw a logo, the following steps must be completed:

1. Turn the desired logo into an scalable vectors graphic (SVG)
2. Select the “Upload SVG File” button on the drawing page
3. From the file explorer on your system, select the desired SVG file
4. Enter the size of the logo that you wish to draw
5. Select “Continue” to proceed with the drawing

At this point in the process, the robotic arm should begin drawing and the following page will appear.

If you wish to stop the arm for any reason, select the "Stop" button on this page.

4.2.2 Painting a Surface a Solid Colour

If you wish to paint a solid colour on a surface instead of an image:

1. Create an SVG file with lines spaced no more than half the width of your paint brush.
   For example, if your brush is 3 cm width, then the lines should be spaced no more than 1.5 cm in the SVG file. Figure 15 resembles an SVG file with the correct layout

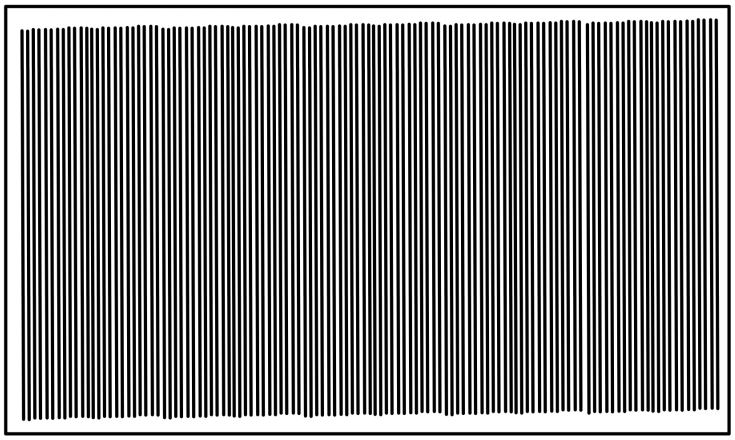


Figure 15. Example of SVG file for painting a surface a solid colour.

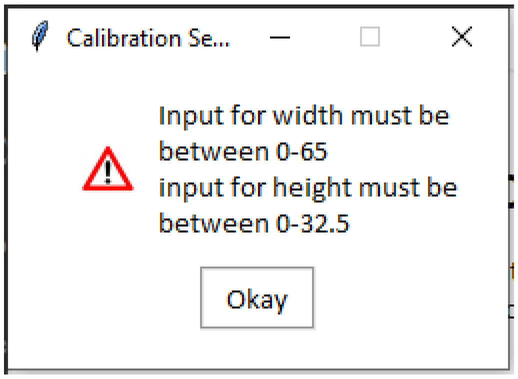
2. Follow the steps listed in section 4.2.1, entering the dimensions of the surface as the width and height of the logo and using the SVG created in step 1 as the logo file

5 Troubleshooting & Support

5.1 Error Messages or Behaviours

5.1.1 Common Calibration Errors

If improper dimensions or inputs are given for the width and height of the object, the following error message can be obtained:

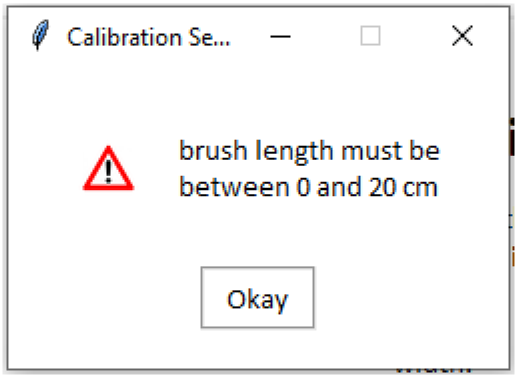


Input for width must be between 0-65
input for height must be between 0-32.5

Figure 16. Error message for improper dimensions for width and height

Ensure that there are numerical values entered in the entry boxes for the width and height and the numbers fit within the parameters for the width (in this case 0 to 65 cm) and the parameters for the height (in this case 0 to 32.5 cm)

If an improper brush length is given, the following error will occur:



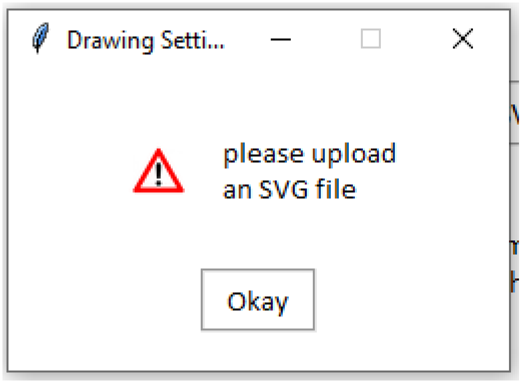
brush length must be between 0 and 20 cm

Figure 17. Error message for improper brush length

Ensure that a value is entered into the entry box for the brush length and the value is a numerical value that fits within the parameters for the brush length (from 0 to 20 cm)

5.1.2 Common Drawing Errors

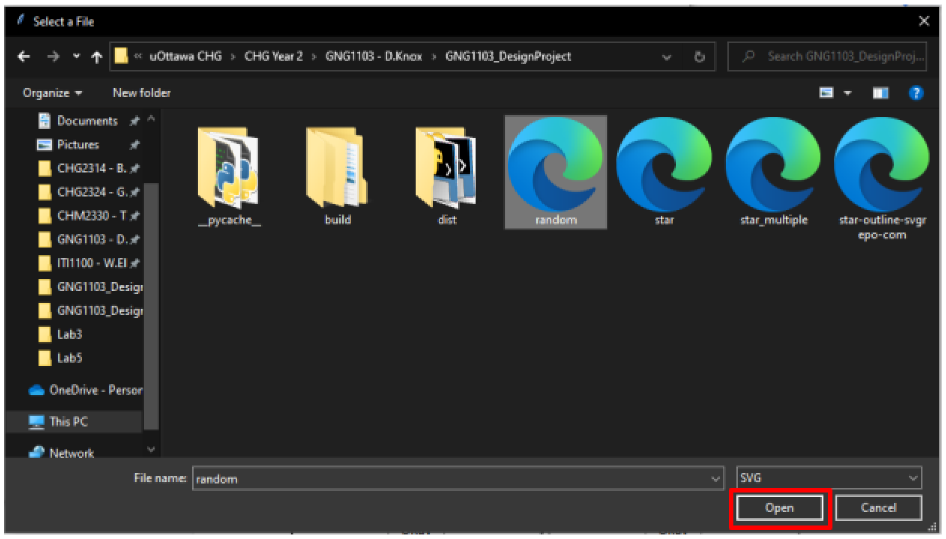
If an SVG file is not uploaded correctly, the following error will occur:


Drawing Setti...

please upload an SVG file

Figure 18. Error Message if a SVG file is not uploaded

When uploading a file to the system, ensure that the file is an SVG file and click the “Open” button located in the bottom right hand corner of your local file explorer



File name: random

SVG

Figure 19. Example of how to upload a SVG file

If the robot is observed to not be moving as instructed, it is possible that there is a problem with the inverse kinematics solver. The user will not be able to identify such a problem with the code, in which case it is better to contact a member of the support team. Section 5.3 outlines the steps required to receive support. A member of the support team can work with the user to identify and correct a problem with the code.

5.2 Maintenance

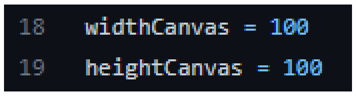
5.2.1 End Effector Maintenance

1. Small breaks in the EE can be fixed with hot glue as needed. Critical failure may result in the EE needing to be reprinted.

5.2.2 Graphical User Interface Maintenance

To adjust dimensions for a robotic arm with different dimensions:

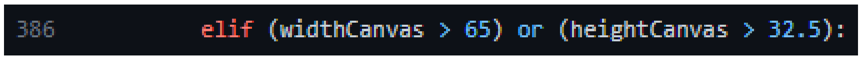
1. Locate line 18 and 19 in main.py file and change the number "100" to the desired horizontal reach of the robot in cm for the variable "widthCanvas" and change the number "100" to the desired vertical reach of the robot in cm for the variable "heightCanvas"



```
18   widthCanvas = 100
19   heightCanvas = 100
```

Figure 20. Line 18 and 19 of code to adjust for robot with different dimensions

2. Locate line 386 in main.py file and change the number "65" to the number set for "widthCanvas" and change the number "32.5" to the number set for "heightCanvas"



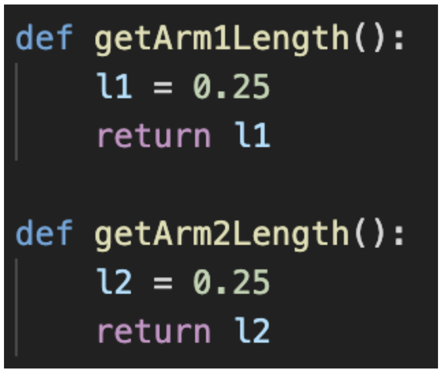
```
386           elif (widthCanvas > 65) or (heightCanvas > 32.5):
```

Figure 21. Line 386 of the code to change the reach of the robot

5.2.3 Inverse Kinematics Maintenance

To adjust dimensions for a robotic arm with different dimensions:

1. Find the *getArm1Length* and *getArm2Length* functions in the newtonsMethod3D.py file and set *l1* and *l2* to the new lengths of the arms, where *l1* is the arm closest to the base of the robot and *l2* is the arm connected to *l2*, farther from the base of the robot.

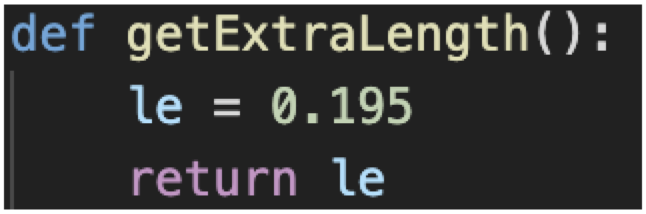


```
def getArm1Length():
    l1 = 0.25
    return l1

def getArm2Length():
    l2 = 0.25
    return l2
```

Figure 22. Adjusting dimensions of the arm

   a. If the robot is upgraded to have more degrees of freedom and more arms, copy and paste the code for the *getArm2Length* function, paste it below the *getArm2Length* function and change the name to "getArm3Length". Also change *l2* to "l3" and enter the length of the arm in metres. Repeat this process for more than three arms.

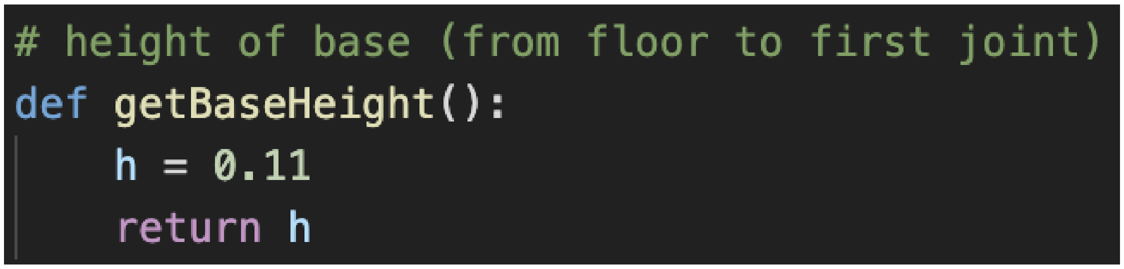
2. If any modification is made to the end-effector, the distance between the tip of a device held by the end-effector, such as a paint brush, and the joint where the end-effector connects to the arm should be measured with a measuring tape. This new distance should be converted to metres and substituted where the "0.195" is in the following diagram in the newtonsMethod3D.py file.:



```
def getExtraLength():
    le = 0.195
    return le
```

Figure 23. Substitution of the length of the end effector

3. If the robot is placed on a surface which adds height to the robot or if the base of the robot is modified, the distance between the centre of the large gear and the floor should be measured. This measurement should be converted to metres and substituted where the "0.11" is in the following diagram in the newtonsMethod3D.py file.



```python
# height of base (from floor to first joint)
def getBaseHeight():
    h = 0.11
    return h
```

Figure 24. Substitution of the distance between the centre of large gear and floor

5.3 Support

In the case where emergency assistance is needed, contact our support team at:
Email: ikgroup06@gmail.com
Phone: 613-555-5555

To maximise the quality of the response and minimise communication delays, in the email, please provide a detailed description of the problem with as many pictures as possible. If the problem is physical, physical pictures are appropriate and if the problem is related to software, provide screenshots of error messages from the graphical user interface. For the best possible support when troubleshooting virtually, it is recommended to have a Zoom account. If possible, video calls on Zoom can be used as a second step to solve a problem because verbal communication is faster than email exchange, and the problem can be shown to the support staff visually, which better equips them to offer support.

6 Product Documentation

6.1 Product Information

6.1.1 BOM (Bill of Materials)

The bill of materials is attached here. For this project, not a lot was purchased since it was mostly supplied. However, everything that is needed is listed in the spreadsheet below.
Bill of Materials

6.1.2 Equipment list

Listed here is all of the equipment necessary to build this product. Including all software, mechanical components, and applications.

Equipment List

6.2 End Effector (EE)

6.2.1 Instructions

Printing:

1. Sign in to Onshape.
2. Right click on the name of the EE piece and go to “Export..”.

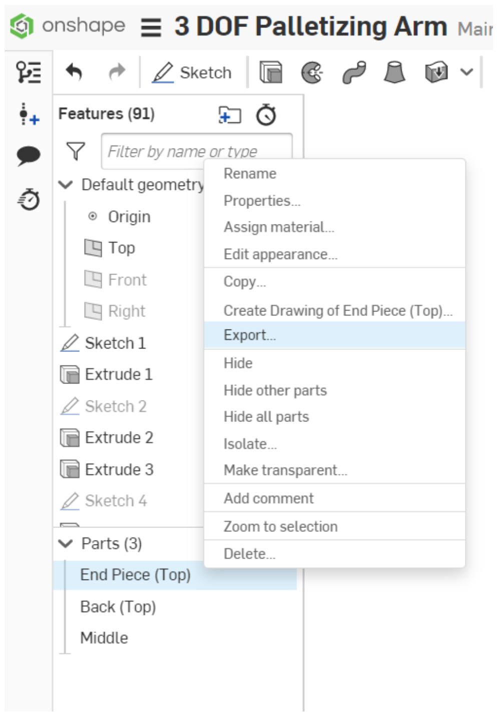
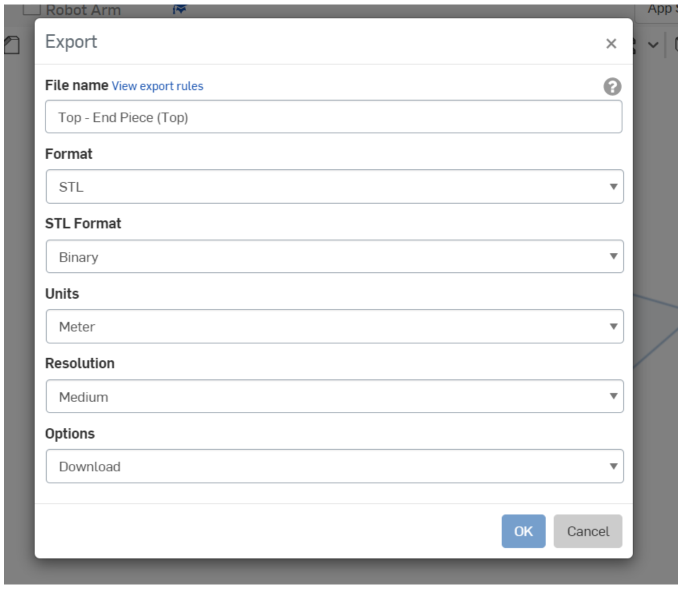


Figure 25. Selecting a CAD to export from Onshape

3. Use the drop down menus to select your desired settings. However, keep the format as STL and STL format as “binary”.

*****You may need to change “Units” to centimetres


Robot Arm

Export

File name View export rules

Top - End Piece (Top)

Format: STL

STL Format: Binary

Units: Meter

Resolution: Medium

Options: Download

Figure 26. Export settings for Onshape

4. Repeat steps 2 through 3 for remaining EE pieces.
5. Open Cura or a different slicing software.
6. A 0.4mm nozzle head or smaller is recommended for these prints.
7. These are the recommended printing positions for the EE. (Depending on the size of the printer's bed it is possible to print more than one piece at once.)

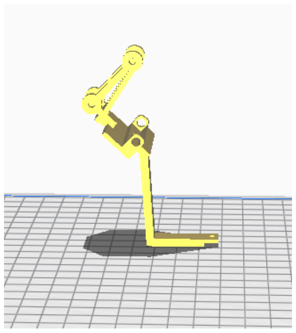


Figure 27. Print position for "Back (Bottom)"

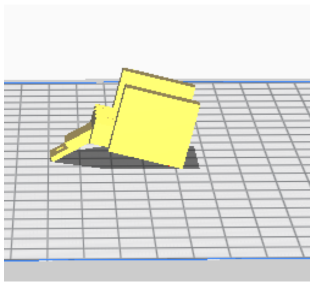
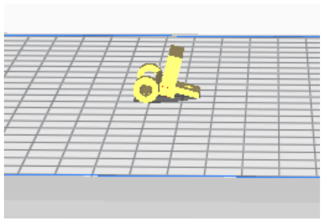


Figure 28. Print Position for "End Piece (Bottom)"



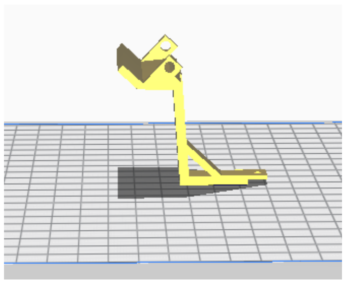


Figure 29. Print Position for “Middle” Figure 30. Print Position for “Back (Bottom)”

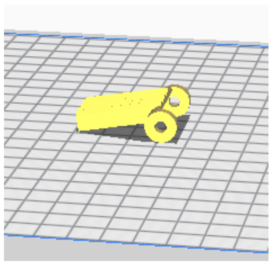


Figure 31. Print Position for “End Piece (Top)”

8. The recommended print settings for all prints:
    a. Profiles: default
    b. Infill: 20%
    c. Wall Line Count: 4
    d. Top Layers: 4
    e. Bottom Layers: 3
    f. Infill Density: 20%
    g. Enable Z hop when retracted (if applicable)
    h. Generate Support: on
    i. Support Placement: everywhere
    j. Support Overhang Angle: 45°
    k. Build Plate Adhesion: brim (or raft)
9. Click “Slice” and “Save to Disc”.
10. Save as a GCODE file.
11. Copy to an SD card for the printer.
12. Begin printing setup as normal.
13. It is recommended that the first layer or two of the print is observed to ensure proper bed adhesion.

Assembly:

1. Remove all supports and clear all holes in the printed pieces.
2. “Back (Bottom)” and “End Piece (Bottom)” can be held together using M3 screws, alternatively the pieces can be glued together.
3. Repeat step 2 with “Back (Top)” and “Middle”.
4. Fit the “End Piece (Top)” over the top part of the EE until the holes are aligned.

5. Insert the spring so it is cradled between the two circles.
6. Slide the shorter (4 cm) rod through the holes of the EE pieces and the springs to create a pivot.
7. Fit the top EE piece over the bottom EE piece.

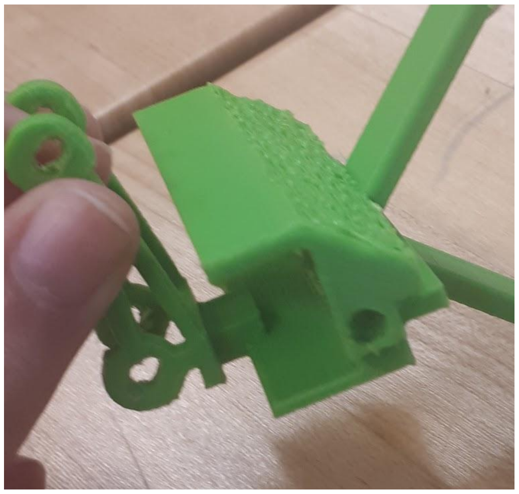


Figure 32. Top EE piece fitted over bottom EE piece

8. Fit the spring between the top and bottom EE pieces.

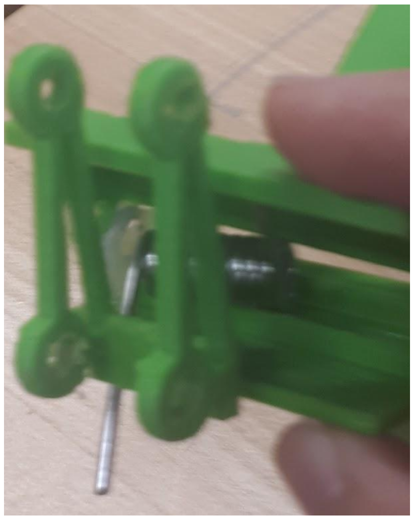


Figure 33. Placing the spring in the EE

9. Thread the longer rod (6 cm) between the holes to create another pivot.

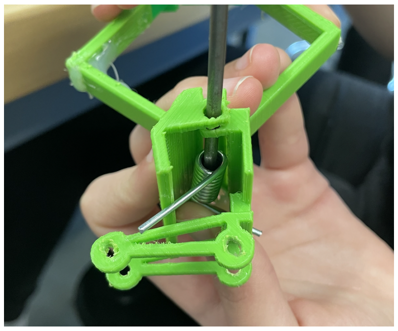


Figure 34. Threading the rod through the EE holes

6.3 Graphical User Interface

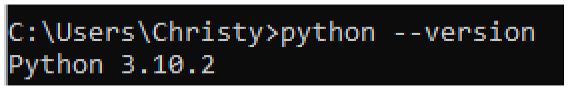
6.3.1 Instructions

Installing the Software

1. Navigate to https://www.python.org/downloads/
2. Download the latest version of python (3.10) for the appropriate system
3. Navigate to the location of the downloaded file and run the Python Installer
4. Ensure that the “Install launch for all users” and “Add Python 3.10 to path” boxes are checked
5. Select “Install Now” with the recommended options
6. Wait for the program to finish installing
7. Select “Disable path length limit” to allow Python to use longer longer path names

Verifying the Installation

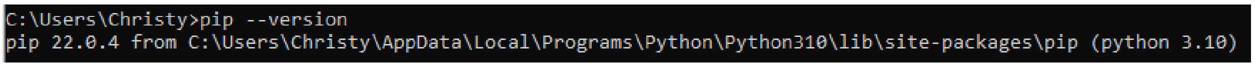
1. Open the Command Prompt on your system
2. Type “python --version” into your Command Prompt
3. The system should return the version information



```
C:\Users\Christy>python --version
Python 3.10.2
```

Figure 35. Example once you have typed python–version into Command Prompt

4. Type “pip --version” to ensure that pip was successfully installed
5. The system should return the version information



```
C:\Users\Christy>pip --version
pip 22.0.4 from C:\Users\Christy\AppData\Local\Programs\Python\Python310\lib\site-packages\pip (python 3.10)
```

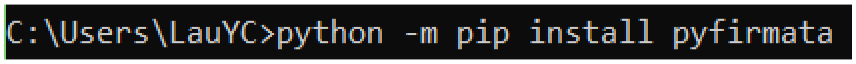
Figure 36. Example once you have typed pip--version into Command Prompt

6. If an error message occurs, follow the troubleshooting the instructions listed in “5.1.1 Pip Command Not Found”

Installing the Appropriate Packages

To ensure proper usage and operation of the graphical user interface (GUI), the following packages must be installed using pip:

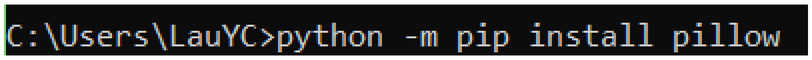
1. Navigate to the Command Prompt on your system
2. Install the Pyfirmata library using the “python -m pip install pyfirmata” command



```
C:\Users\LauYC>python -m pip install pyfirmata
```

Figure 37. Installation of the Pyfirmata library

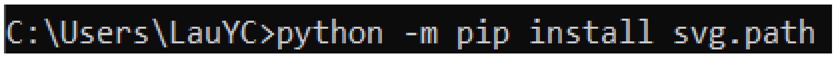
3. Install the PIL library using the “python -m pip install pillow” command



```
C:\Users\LauYC>python -m pip install pillow
```

Figure 38. Installation of the PIL library

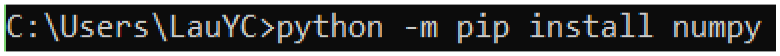
4. Install the SVG library using the “python -m pip install svg.path” command



```
C:\Users\LauYC>python -m pip install svg.path
```

Figure 39. Installation of the SVG library

5. Install the NumPy library using the “python -m pip install numpy” command



```
C:\Users\LauYC>python -m pip install numpy
```

Figure 40. Installation of the NumPy library

6.4 Inverse Kinematics

6.4.1 Instructions

For ease of use, all code for this product is written in Python to minimise the amount of downloading required from the user. If Python and the "numpy" library have not already been installed, refer to Section 6.3.1. And complete "Installing the Software", "Verifying the Installation", and the fifth step from "Installing Appropriate Packages" as preliminary steps before using the inverse kinematics code.

The decision to use Newton's method to solve the inverse kinematics problem was based on several features.

1. According to Chapra and Canale (2015, pp. 153-154) , Newton's method is quadratically convergent, so the method is very efficient. Quadratic convergence means that the error decreases quadratically each iteration. For example, if the error were 1/100 of a radian from the actual answer, with only one more iteration, the error could be reduced to 1/10000. An efficient algorithm is important because although computers can perform a single computation quickly, when thousands or orders of magnitudes more calculations are required to run a process, computational time can become undesirably long.
2. Using a function from a library, such as fsolve from scipy.optimize could have also solved the inverse kinematics problem numerically. However, numerical methods to solve nonlinear systems of equations usually require computation of derivatives to use search techniques using Jacobian matrices, gradients, or Hessian matrices to name a few examples. A disadvantage of a function from a library, such as fsolve, is that it is designed to solve any system a user can provide. For the solver to achieve such generality, derivatives would have to be computed numerically, which requires a numerical differentiation method, such as finite differencing, which can require many computations and increase program run time, or lead to significant truncation error if not done well. The advantage of implementing Newton's method to solve the inverse kinematics for this robot specifically is that all the partial derivatives for the jacobian matrix were found analytically, so no computational power was allocated to approximating derivatives and there is no truncation error

The inverse kinematics solution is integrated into the graphical user interface so that it can accept coordinates from an SVG file as inputs and solve for the angles of the joints. The angles are then converted to steps to be used by the stepper motors. This process is shown in the following flowchart.

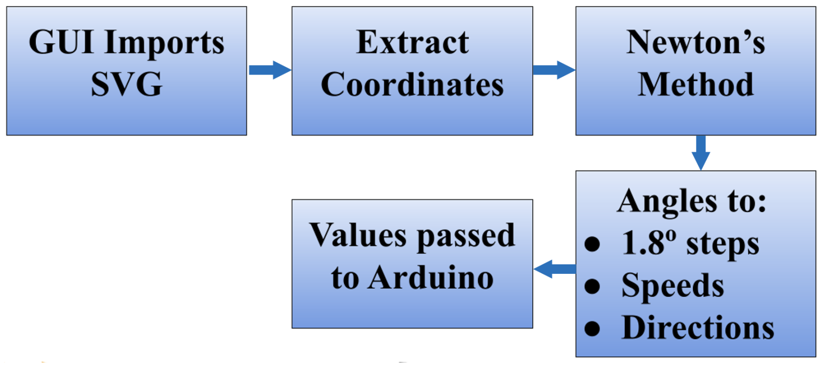


Figure 41. IK solution flowchart

6.5 Testing & Validation

To test that the code for the inverse kinematics solution worked, realistic test cases were generated and used to test the code. The test cases were generated using forward kinematics to ensure that the inverse kinematics could be validated using a solution from a different method.

To solve the inverse kinematics, Newton's method for systems of equations took an initial guess of the solution (a set of angles) as input, then iterated to find the correction solution using a jacobian matrix. The previous prototype proved to perform well when the initial guess of the solution passed into Newton's method was relatively close to the actual solution. The method diverged when unreasonable initial guesses were passed into the function, due to the sinusoidal nature of the functions.

The chosen solution for this divergence problem was to analyse the spatial characteristics of the situation to determine a good initial guess for Newton's method. The most logical idea for a good initial guess was the solution for the angles when the robot is at the average position. This position was defined to be in the middle of the rectangular surface on which the arm paints.

Using single variable optimization, the y_{max} and z_{max} were calculated to maximise the surface area of the rectangle. The x_{max} is the distance between the surface being painted and the base of the first arm when the robot is facing the wall perpendicularly. The average point lies at (x_{max}, $y_{max}/2$, $z_{max}/2$), where y_{max} is the maximum height on the painted surface and z_{max} is the width of the painted surface, where $z = 0$ is the left-hand side of the constrained area. To use the angles of the joints at the centroid as an initial guess, they first need to be solved. The angles were solved using Newton's method and the initial guesses used were based on visually inspecting the Onshape CAD file. The angles were validated by passing the angles into the forward kinematic equations and obtaining the correct coordinates.

To test the accuracy of Newton's method, ranges for each angle were generated. The ranges of these angles were divided evenly into 100 angles and using three "for loops" in the Python code, coordinates for each point corresponding to each set of angles were calculated using forward kinematic equations. Then, the coordinates were passed as inputs into Newton's method and using the same initial guess (x_{max}, $y_{max}/2$, 0), the angles corresponding to each set of coordinates were calculated numerically. To test the accuracy of the solutions, the actual angles generated were compared to the numerically calculated angles using an error function, where the absolute values of the differences between each angle are calculated. If the absolute values of the differences between all three angles were less than 0.01 radians, then the solution was considered correct. A function called *testAccuracy* was created and if the numerical solution passed the error test, then a variable called *accuracy* was increased by one. To check the percentage of test cases that were correct, the final value of *accuracy* was divided by the total number of test cases. After running 1000000 test cases covering the entire rectangular surface evenly, the *testAccuracy* function returned 1.0, indicating that 100% percent of the test cases were solved correctly. With this result, the code for the inverse kinematics solution was considered to have been validated.

7 Conclusions and Recommendations for Future Work

Throughout working on the final product many things were learned but the main thing was not to rush the final product. Three months were given to finish this product, this made it difficult to fully complete the design process and create a final prototype. Along the way it was made clear that a fully polished prototype was not needed. This gave more time to focus on making the things it did have (like the end effector) better. If someone were to recreate the work done to make this product, they would have more time and would therefore be able to make improvements to the original plans.
An improvement that was considered was adding a switch at the base of the actual robotic arm. This would act like the kill switch on the UI and would stop all movement of the arm. Adding this switch/button would add another safety element to the arm and improve the overall security of the product. Another safety precaution that could be taken is adding ultrasonic sensors to detect if people walk in front of it, the sensors would then tell the robot to shut off. Like the button, the robot would stop all movement.
Some other improvements that could be made would be to add the holes in the EE so that an arduino camera could be added. This would make the robot even more versatile. The EE is currently divided into five pieces, but it could be combined into three pieces so that it would be

easier to print. This would make it harder to break and would help with the overall aesthetics of the EE.
Another improvement would be adding a bluetooth module, this would eliminate the need to plug it in directly into a computer. Adding this would make the whole system more accessible and the person would not have to be right next to the robot to be able to start the software.
There are other things that would help improve the product that will come up as it is developed. These are the main things that our group thought of as we created our prototype, they should improve the overall functionality of the product.
A possible improvement to the inverse kinematics solver is removing the use of the linear algebra library from numpy. Once the elements of the Jacobian matrix are computed, the problem becomes a system of linear equations, as opposed to a system of nonlinear equations, so a linear solver can be used. Investigation of preconditioning techniques is strongly recommended as they can increase the efficiency of the solver by taking advantage of known characteristics of the specific case of this robotic arm.
In addition to preconditioning, relaxation techniques should be researched. Over-relaxation techniques can greatly improve the efficiency of Newton's method by essentially multiplying the convergence rate linearly to approach the solution in fewer iterations. A disadvantage of over-relaxation is that methods can lose stability and be more prone to divergence. However, a preconditioning function for Newton's method can be easily designed to generate a better initial guess of the solution, which would significantly reduce the possibility of divergence. Currently one single point is used as the initial guess for each use of Newton's method, which should be the primary step to improve Newton's method.

8 Bibliography

Chapra, S., C., Canale, R., P., (2015). Open Methods. *Numerical Methods for Engineers* (7th ed., pp. 153–172). McGraw-Hill Education.

APPENDICES

9 APPENDIX I: Design Files

Table 3. Referenced Documents

Document Name	Document Location and/or URL	Issuance Date
Github	https://github.com/PaulShedden/3DOF_Robot_Arm	April 6, 2022
OnShape model	https://cad.onshape.com/documents/17f82529141ccfc1b2c39232/w/c12cd73e320a79169e051575/e/dd09b81558dee06a0c4f0eab?renderMode=0&uiState=624f7f1c5f345e32e0bc8659	April 6, 2022
MakerRepo	https://makerepo.com/cdura/1139.ik-robotic-arm-gng1103c06	April 6, 2022