

GNG1103

Design Project User and Product Manual

Inverse Kinematics Solver

Submitted by :

GNG 1103 Group C03

Benoit Tremblay (300236751)

Rebeca Poulin (300236741)

Jess Beardshaw (300227707)

Isabelle Barrette (300228564)

Sarah Alkadri (300245117)

April 8th, 2022

University of Ottawa

Table of Contents

Table of Contents	2
List of Acronyms and Glossary	5
Table 1. Acronyms	5
Table 2. Glossary	5
1. Introduction	6
2. Overview	7
Figure 1. Drawing End-Effector	8
Figure 2. Camera End-Effector	9
Figure 3. Mounted Camera End-Effector	9
Figure 4. Sensor System	10
Figure 5. User Interface Return Page	10
Figure 6. User Interface Drawing Page	11
Figure 7. Corrosion Detection Block Diagram	11
2.1 Cautions & Warnings	11
3. Getting Started	12
3.1 Inverse Kinematics	12
3.1.1 Compatibilities	12
3.1.2 Mathematical Concepts	12
Figure 8. Inverse Kinematics	13
3.1.3 Coding	13
3.2 Corrosion Detection Code	13
Figure 9. Corrosion Detection Setup	13
3.3 User Interface	14
Figure 10. User Interface Main Page	14
3.4 End Effectors	14
3.5 Camera	15
3.6 Safety	15
3.7 User Access Considerations	15
4. Using the System	17
4.1 Corrosion Detection	17
4.2 Inverse Kinematics	17
Figure 11. Inverse Kinematics Variable Settings	17
Figure 12. Inverse Kinematics Pins	18
Figure 13. Inverse Kinematics Compilation	18

4.3	User Interface	18
	Figure 14. User Interface Camera Page	19
4.4	End Effectors	19
4.5	Camera	19
	4.5.1 ArduImageCapture Instructions	19
	4.5.2 Compiling in Arduino IDE	19
	4.5.3 OV7670 Hardware Connections	19
4.6	Safety	20
	4.6.1 Hardware Setup	20
5.	Troubleshooting & Support	21
5.1	Error Messages or Behaviors	21
	5.1.1 Corrosion Detection AI.	21
	Figure 15. String of commands in Command Prompt to reach desired folder directory	21
	Figure 16. Error message received because of scikit-image problems	21
	Figure 17. Error message from imageTools.py about cv2 (OpenCV)	22
	Figure 18. Images from the test folder being processed within the program.	22
	5.1.2 User Interface	23
5.2	Special Considerations	23
5.3	Maintenance	23
5.4	Support	23
6.	Product Documentation	25
6.1	Prototype III	25
	6.1.1 BOM (Bill of Materials)	25
	Table 1. Original bill of materials with total product cost estimate	25
	Table 2. Bill of materials with total product cost estimate	26
	6.1.2 Equipment list	27
	Table 3. Equipment list and resource description	27
	6.1.3 Instructions	27
	6.1.3.4 End Effectors	28
	Figure 19. End-Effector Models	29
	Figure 20. Drawing End-Effector	29
	Figure 21. Camera End-Effector	30
	6.1.3.5 Safety	30
6.2	Testing & Validation	31
	Table 4. Testing and planning	34
7.	Conclusions and Recommendations for Future Work	35
	Bibliography	36

Appendices	39
Appendix I: Design Files	39
Table 5. Referenced Documents	39

List of Acronyms and Glossary

Table 1. Acronyms

Acronym	Definition
BOM	Bill of Materials
IK	Inverse Kinematics
GUI	Graphical user interface

Table 2. Glossary

Term	Acronyms	Definition
Bill of Materials	BOM	List of materials we used for the project along with their prices.

1. Introduction

John Faurbo and the Royal Canadian Navy need a robotic arm that has the ability to detect, remove and paint over corroded areas to ensure proper maintenance on the Halifax class warships. Cost, ease of use, proportions and safety of product are key to creating the best product for the client's needs.

To solve his problem, we have put together a list of prioritized needs, target specifications, and similar existing products for benchmarking.

We have accumulated a list of the users interpreted needs, including function and non-functional requirements, as well as metric constraints. Our target specifications include the ideal and marginal values given to us by John Faurbo. In addition to the design criteria, we have begun to benchmark and research other solutions that satisfy our interpreted needs. We will use these to begin our conceptual design in the future.

We explored the different concepts belonging to 5 subsystems that will collectively join to form a final functional solution. We have divided our functional solution into five different subsystems including the user interface, the three end effectors, and the overall coding and programming of the robot. Each team member is responsible for generating ideas for each subsystem, discussing boundaries and relationships between the subsystems so the concepts are interchangeable in the final solution. From our individual ideas, our team created a selection matrix by discussing the requirements, similarities, and drawbacks and created three fully functional solutions and compared the global concepts with respect to our design criteria. The best solution is chosen, from which we will present to the client for feedback.

The end effectors are designed with respect to each task needed to be completed according to the design criteria. It is ideal for the arm to be able to translate efficiently and effectively from each position. Each position is then associated depending on the specific end-effector. To ensure portability and mobility, we have included "resting pose", where the arm will return in a compressed manner to help with transportability. These poses are achieved through inverse kinematics in a code later described in this deliverable. The inverse kinematics schemes are done in a 3D plane to achieve the three degrees of freedom in the x, y, z plane. Three rotational matrices can be calculated to define the precision in space associated with each end-effector pose.

We have compared different products with respect to the cost, time constraints, and projected usability to determine how we should proceed. Other concerns discussed are technical and resourceful risks surrounding circumstances not under our control, such as team member changes and receiving fault products.

Discussing all the risks before beginning the next design process will prepare us for all unfortunate scenarios that could set us back, preventing us from producing the best design possible. Our team is committed to the quality of the final solution, displaying the commitment we have put into the project.

This User and Product Manual (UPM) provides the information necessary for users to effectively use the Inverse Kinematics Robotic Arm (IKRA) and for prototype documentation.

2. Overview

John Faurbo worked as a Naval Combat Systems Engineer with the Royal Canadian Navy. Now working with Theo Eastmond at Naval Material Technology Management for the National Defense, they intend to deploy robotic arms aboard the Halifax Class Vessel to aid in the ship's maintenance, mainly with corrosion along steel surfaces.

Implementing robotic arms to perform simple tasks will free up crew members and sailors as well as cut the cost of hiring more crew for simple jobs. The robotic arm and its end-effectors must scan and record the geometry of the corroded regions. The surfaces can vary from simple and flat to curved planes and intricately designed piping in compact areas. Once scanned, the robot can apply corrosion removing spray, clean the surface from the coating, and spray a final layer of paint.

The previous attempt to solve this problem was met with difficulties as the arm is not programmed with inverse kinematics (such as the Thor model), which increases the accuracy of the motion to a certain position. GCODE is used to move each arm individually, but it is less effective without inverse kinematics. In addition, the sensors bought by the Navy did not fit the model of the arm used.

Our client needs a cheaper way to detect, remove, and paint over corrosion on the Halifax Class Vessel. The ideal case for the cost of this project would have it staying within the predetermined 100\$ budget, but it is possible to negotiate a slightly higher budget. The robot arm is required to fit into small spaces to scan, analyze, and paint in all areas of the ship. The general use involves:

- Setting the arm in proximity to the problem area,
- Activating the camera to scan and identify problem areas in a point coordinate system, and
- Painting the identified targeted areas.

The robot must be usable in low oxygen areas and have lights or night vision to detect imperfect areas anywhere on the ship in all lighting conditions. It must be deployable to paint the ship's outside on cables or a barge moving around the exterior.

The robot arm should be able to hold up to 1kg of weight (camera, nozzle and paint) and withstand approximately 140-180psi of pressure from the water hose. The robot arm must be powered by 120-volt outlets, the vessel's only energy source. The robot must be sustainable and only require minor repairs every two to three months and significant repairs after six months.

The 3D printed robotic arm will need 3 degrees of freedom to perform actions with precision. Accomplishing this requires inverse kinematic equations to determine the desired coordinate positions of the robot's movement. For efficiency, the robotic arm must reach a minimum speed of 1m² per hour.

The software used for the robotic arm design is open-source codes, G CODE and GRBL, and a well-known programming language such as C or C++ or Python to program the robot's tasks. A useful asset for the robot would be to capture a visual with the camera and send it over to Royal Canadian Navy offices.

The robot's controls must be simple and easily accessible by any crew member. A handle or carrying system is needed to ensure the user can maintain three contact points when climbing ladders or stairs on the vessel. The ability to switch between different endpoints, catered to clients' demands, quickly and efficiently is essential. If appropriate safety measures are implemented, crew members will be more comfortable working with the arm, such as an automatic pause or shut down the system when a person approaches the robot. Ideally, the robot is small or compactable into smaller pieces that do not exceed 20lbs (including its end-effector, which weighs less than 750g).

Our solution is a multi functional product that solves different aspects with inverse kinematics, a corrosion detection software and sensors.



Figure 1. Drawing End-Effector

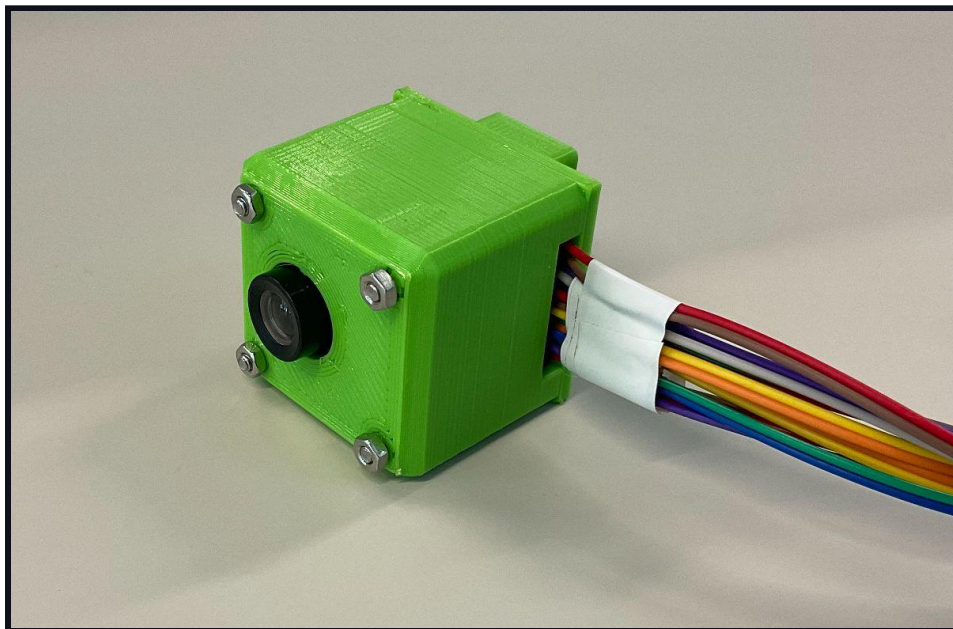


Figure 2. Camera End-Effector

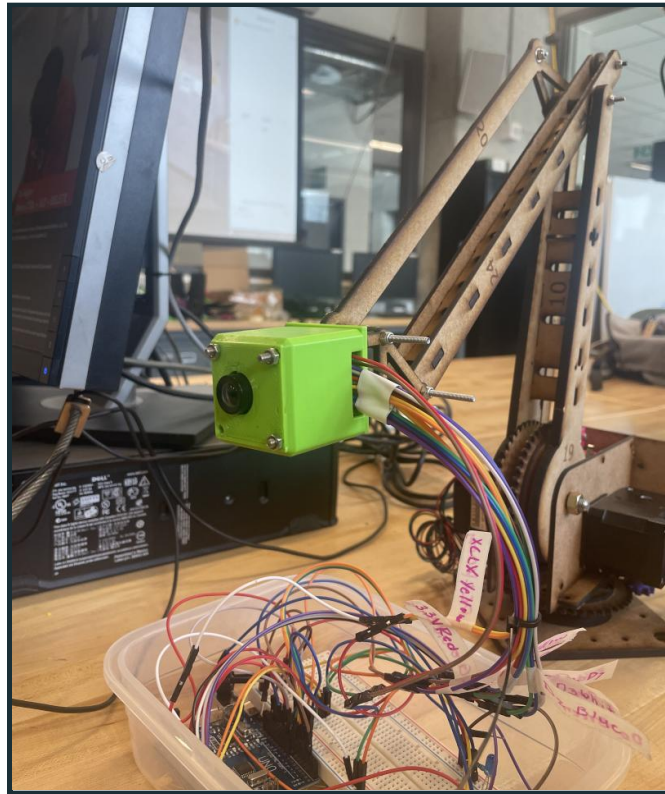


Figure 3. Mounted Camera End-Effector

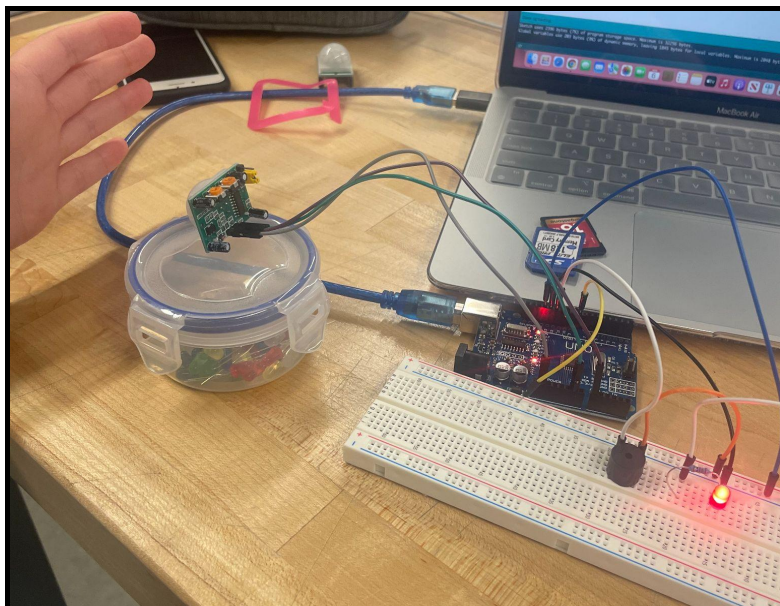


Figure 4. Sensor System

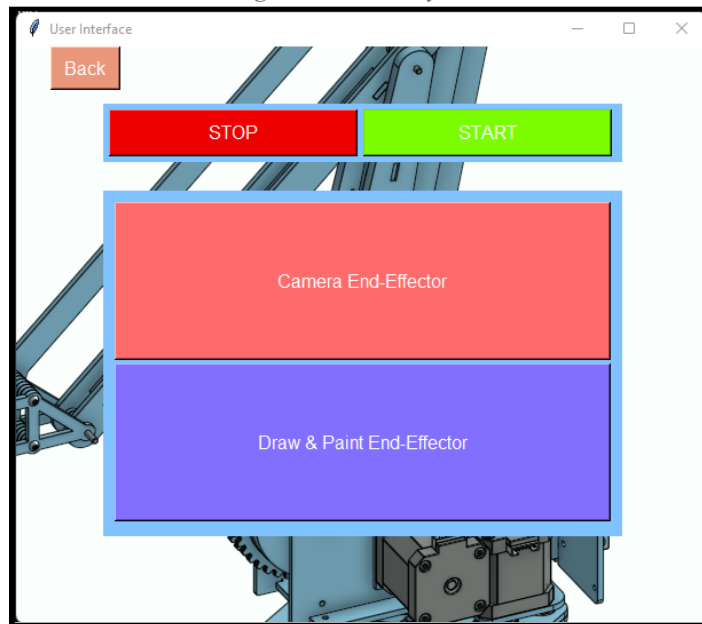


Figure 5. User Interface Return Page

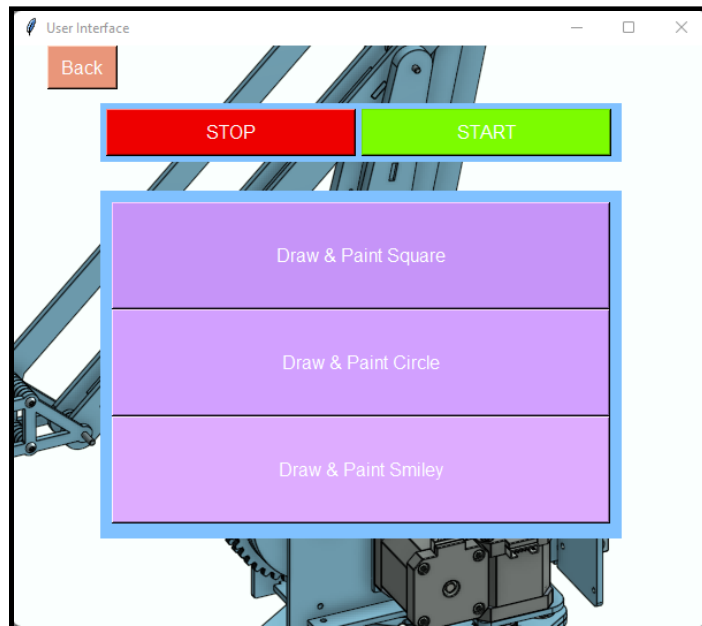


Figure 6. User Interface Drawing Page

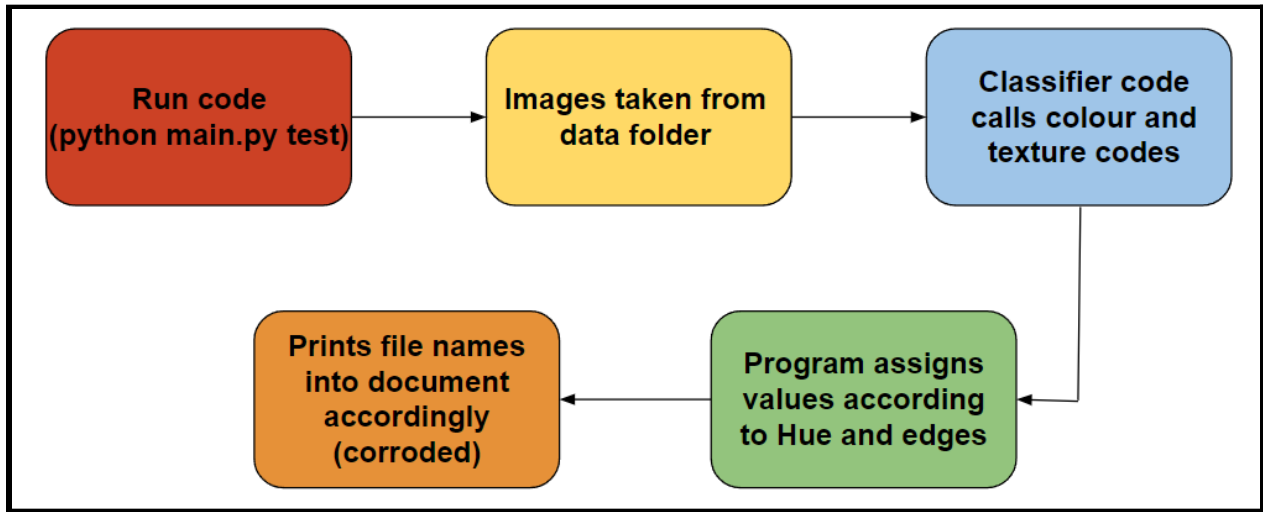


Figure 7. Corrosion Detection Block Diagram

The key features include our corrosion detection code, our camera end effector and its sensor system as well as the many features such as the drawing patterns and stop function available for the use of the robot arm through the inverse kinematics and the GUI.

2.1 Cautions & Warnings

The programs and separate components are still more in a prototype state therefore they are not fully safety tested. They may very well malfunction, however there should not be any major concerns of safety since we did implement measures to deal with such. Some occasions where it may be best to be cautious would be avoiding walking in front of the arm or on its sides while in motion and paying checking up on it every now and then so that it does not overextend itself.

3. Getting Started

Before starting, the user should download the Arduino IDE and the python compiler. Instructions can be found on the Arduino and Python official websites:

- Arduino Download: <https://www.arduino.cc/en/software> (Arduino IDE version 1.8.18 is used in this product)
- Python Download Instruction Guide: <https://wiki.python.org/moin/BeginnersGuide/Download> (Python version 3.14 is used in this product)
- NOTE: When downloading Python make sure that the "Add to PATH" box is checked before completing the download

3.1 Inverse Kinematics

Begin by downloading the Arduino files for the Inverse Kinematic code.

- Github Link for Inverse Kinematic Code: <https://github.com/becapoulin/inverse-kinematics>

This is a C++ open source code that calculates the required steps to be done by stepper motors in order to move the robot arm to desired coordinates. This code is compatible and tested on a 3 DOF robotic arm. To use the inverse kinematics code, you need a robot with 3 stepper motors, one for the base joint, one for the shoulder joint and one for the elbow joint. The file entitled *IK_CODE_palletizing_stepper.ino* is the main inverse kinematics solver code.

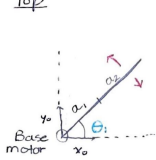
3.1.1 Compatibilities

The code is functional with a robot arm using a CNC shield, DRV8825 motor drivers, Nema 17 stepper motors to move the base, shoulder and elbow of the arm. To make this compatible, we needed to understand how the CNC shield works. In our case, we only need the x, y and z connections. For the robot arm used to test the code, the x coordinate is connected to the elbow joint, the y coordinate is connected to the shoulder joint and the z coordinate is connected to the base joint.

3.1.2 Mathematical Concepts

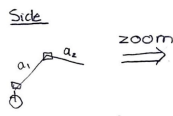
Pythagorean theorem is used for the right angle triangles. *SOHCAHTOA* is used to evaluate the sine, cosine and tangent angle relationships. The law of cosines is used when there is no right angle triangle. These calculations show the math concept behind finding the angles needed to reach an (x,y,z) coordinate if modifications are required:

Top



input $\rightarrow x_0, y_0$
 Base rotation = $\theta_1 = \tan^{-1}(y_0/x_0)$
 if theta > 0, direction = HIGH else, direction = LOW

Side



zoom

a_1 = shoulder length
 a_2 = elbow length
 B = Hypotenuse

① $Hypot = \sqrt{x_0^2 + z_0^2}$

$\alpha_b = \arccos((B^2 + a_1^2 - a_2^2)/2) \cdot B \cdot a_1$
 $\alpha_i = \alpha_b + \arctan(z_0/x_0)$
 $\alpha = 90 - \alpha_i$
 $\beta_b = \arccos((a_2^2 + a_1^2 - B^2)/2) \cdot a_1 \cdot a_2$
 $\beta = 180 - (\beta_b + (90 - \alpha_i))$

Figure 8. Inverse Kinematics

3.1.3 Coding

The code is simplified to fit a palletizing robot arm, where the angle of the Elbow servo motor stays constant and partially independent to the rotation of the Shoulder servo motor. A `float` variable is used for all variable types for maximum precision for inverse cosines involving significant decimal places.

The code begins with setting the dir and step pin of each connection to a joint. The code requires all the motor drivers to output and set their direction to either clockwise or counter clockwise, depending on whether the angle is positive or negative. Calling the calculation function is used to obtain the angles of the arm placement. For loops are used to increment the arm's position by one unit until it reaches the amount of steps required to reach the desired coordinate.

It is required to divide the value of the angle by 1.8 for the declaration to function with the 200 revolution stepper motors.

3.2 Corrosion Detection Code

In order to get the corrosion detection code running we need to first download to the corrosion detection folder from the Github link here: <https://github.com/benoittremblay/Corrosion-Detection-AI>

The next step is to then download Python 3.10.4, the most important part is to select “Add to PATH” during the download process. This is the single most important step. Next download the following Python libraries using the pip installer from the command prompt. The libraries are Numpy, Scikit-image, Scipy, Matplotlib, OpenCV, by wavelets, and Jupiter.

After downloading the libraries, set up the directory to run from python. Using the `cd` command we use it repeatedly until we reach the `src` folder which is where all of the python files are located. This allows us to be able to run all of the python files simultaneously. The picture below is an example of the directory being set up.

```
C:\Users\btrem>cd\  
  
C:\>cd python  
  
C:\Python>cd Corrosion detection2  
  
C:\Python\Corrosion detection2>cd corrosion-detection-master  
  
C:\Python\Corrosion detection2\corrosion-detection-master>cd src
```

Figure 9. Corrosion Detection Setup

Finally, to run the code type “python main.py test 0.18 0.2” into the command prompt. Python calls for the python program to be used. Main.py is the controlling code that calls the other python files to be used. Finally “test 0.18 0.2” is the string to execute the code, the numbers following the test are the hue saturation and edge threshold values when detecting the pixels.

3.3 User Interface

To prepare the GUI and its functions the python libraries must be checked to see if they are installed through the command prompt which is explained in the corrosion detection section. The necessary libraries are only the pillow or PIL libraries which may come already installed within the python software that is downloaded onto the computer. Once this is done and confirmed, the code file and relative images must be prepared and their respective paths in the computer's folders must be adjusted in the spot in green shown in the image below.

```
background_image = ImageTk.PhotoImage(Image.open("C:\\Users\\isasb\\Pictures\\arm.png"))
background_label = tk.Label(root, image=background_image)
background_label.place(relwidth=1, relheight=1)
```

Once these have been adjusted to the environment, it is now time to run the code. This can be done in the IDLE python interface by pressing run at the top or it can be pasted into the python interface and run automatically. Once it has run the following image should pop up as a window to interact with all of its functions which are explained later on in the manual.

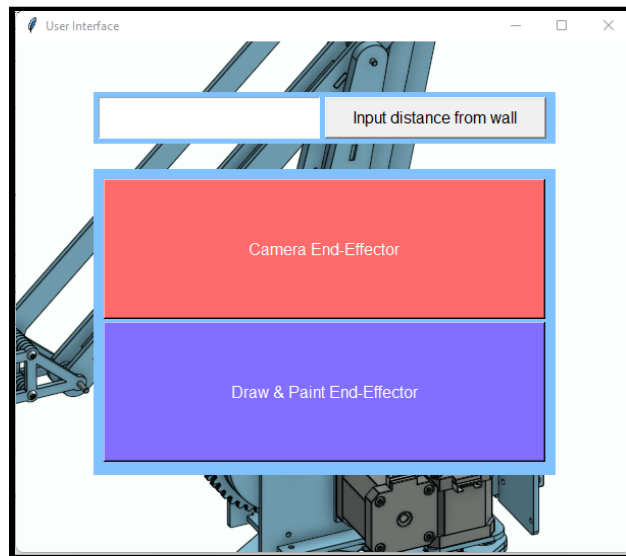


Figure 10. User Interface Main Page

3.4 End Effectors

Both end effectors must be 3D printed with 0.8mm material thickness and 0.30mm filing. A higher precision thread can also be used, but be cautious of printing times. Printing instructions vary depending on the 3D printer used.

- [Onshape link to download STL Files](#)

To attach the end effectors to the robot, use 2 M3 bolts (minimum of 35mm in length) and a corresponding nut fastener. The robot's finger has two holes where the user may insert the end effector, ensuring the holes are lined up and the end effector is oriented properly. Each bolt must be placed in opposition (for example, a bolt head facing the left direction then the other must be placed facing the right).

Ensure the sharpie is firmly held in place in the end effector by pushing on the sharpie with slight force.

3.5 Camera

The OV7670 Arduino Compatible camera is used in conjunction with the corrosion detection software to capture and save pictures to the user's computer. The code is a modification of [Indrek Luuk's guide to using the OV7670 over USB import](#). Download the code using the Github link below:

Github link to Ov7670 Camera Code: <https://github.com/jessbeardshaw/OV7670Camera-with-PIRsensors>
Keep in mind the "LiveOV7670" library required and is included in the installation files.

To run the camera code, install and download the Arduino IDE (see section 3 "Getting Started"). Follow the instructions in section 4.5 "Camera" under section 4 "Getting Started" for instructions on downloading the *ArdulImageCapture* image interface.

The following components are required for the wiring of the camera:

- A variety of male-female, male-male, and female-female wires (wire count depends on personal setup). Minimum required 36x male-female wires.
- 1x Breadboard
- 1x Arduino Uno
- 1x OV7670 Camera, no preferably Fifo (see section 4. whatever for with fifo instructions)
- 2x 10k ohm resistors
- A pair of resistors to create a 5V to 3.3V voltage divider (650 ohm and 1k ohm, 1k ohm and 2k ohm, etc...)

3.6 Safety

The motion detection sensors are implemented and tested to detect infrared presence within 7 meters. They alert any passerby of the robots movement by triggering a buzzer and blue LED on the breadboard. Once the sensors are triggered, they send a signal to the LED and buzzer and an interrupt function on the arduino will pause the movement of the arm for the timed duration. Assuming the sensors are not re-triggered, the arm will resume the performing function.

Begin by downloading the code from

The following components are required for the wiring of the camera:

- A variety of male-female, male-male, and female-female wires (wire count depends on personal setup). Minimum required 8x male-female wires.
- 1x LED
- 1x Buzzer
- Minimum x1 PIR sensor
- 1x 270 ohm resistor

The motion detection sensors do not require any additional set-up once the hardware is connected. They function independently from the user control to prevent injuries.

3.7 User Access Considerations

The different types of groups that will be using this product are the high school educated employees working on the Halifax Class Destroyer first and foremost. They do not need a technical background so the product must be able to be used and controlled by anyone with any type of experience.

Another user for this product is a technical worker, possibly engineer, working for the Canadian Navy. It is assumed that they would use the product for its intended purpose if necessary but will be performing more technical tasks such as fixing software if ever there is the need for it. For example if there is an error message in the code of any kind this is who would be fixing it.

4. Using the System

4.1 Corrosion Detection

Once the “python main.py test 0.18 0.2” is executed, the code takes the images from the data folder and are inputted into the various python files. The pictures are run through the colour, texture, image and imagetools python files. These files use the hue saturation and edge threshold values to look for rust coloured pixels on the images as well as the areas around those pixels to create a texture and compare to other rust images to determine if the considered area in the picture is rust. Once the picture is considered corroded or not it assigns a value accordingly. Finally, the pictures that are given a value of corroded, their file names are outputted onto a results notepad so the user knows which pictures/areas are corroded.

4.2 Inverse Kinematics

To run the code, replace the length variables with the lengths for the shoulder and elbow links. You can then enter the target coordinates at the top of the code.

```
float BaseLength = 3.229;
float ShoulderLength = 9.555;
float ElbowLength = 9.859;
float Hypot;
float x0 = 5; //Target end point TEST
float y0 = 8; //Target end point TEST
float z0 = 4; //Target end point TEST
```

Figure 11. Inverse Kinematics Variable Settings

The code is compatible with a CNC shield using the X, Y and Z step and dir pins. Depending on the attachments they can easily be changed.

```
const int StepX = 2; //elbow joint
const int DirX = 5;
const int StepY = 3; //shoulder joint
const int DirY = 6;
const int StepZ = 4; //base joint
const int DirZ = 7;
```

Figure 12. Inverse Kinematics Pins

Before running the code, verify and compile the code. Then connect the device running the code to the arduino via USB connection, select the correct port, and upload the code.

```
Output  Serial Monitor
Sketch uses 4588 bytes (14%) of program storage space. Maximum is 32256 bytes.
Global variables use 280 bytes (13%) of dynamic memory, leaving 1768 bytes for local variables. Maximum is 2048 bytes.

-----
Compilation complete.
```

Figure 13. Inverse Kinematics Compilation

4.3 User Interface

The code begins by importing the necessary libraries to execute the code downloaded and installed beforehand. Once everything is set up, the code begins reading the defined variable, which in this case is "emergency_stop," which has the pressing of the stop button interrupt the code, "startcamcode" which runs it when its associated button is pressed. There is also "distance_wall" which is the button pressed to print the entered distance from the wall in meters into the inverse kinematics, as well as "takepicture" which saves the images to a folder in case of needed extra inspection. The last button is the "camerafeed" which when pressed will run the camera feed code as well as the corrosion detection software attached to it.

The main code contains a mixture of labels, frames, buttons and an entry box, which, combined with all their conditions, create the user interface that you see when the code is running. When run, the code will pop up the window shown in the "Getting Started" section, which contains the buttons and entry box of the home page. Once the measurement is input in the entry box and the button beside it is pressed it will input this distance in meters into the inverse kinematics code to ensure the drawings and scanning are done effectively. This means it is now time to pick an end effector function, and if going in order it would start with the camera one which will look like so.

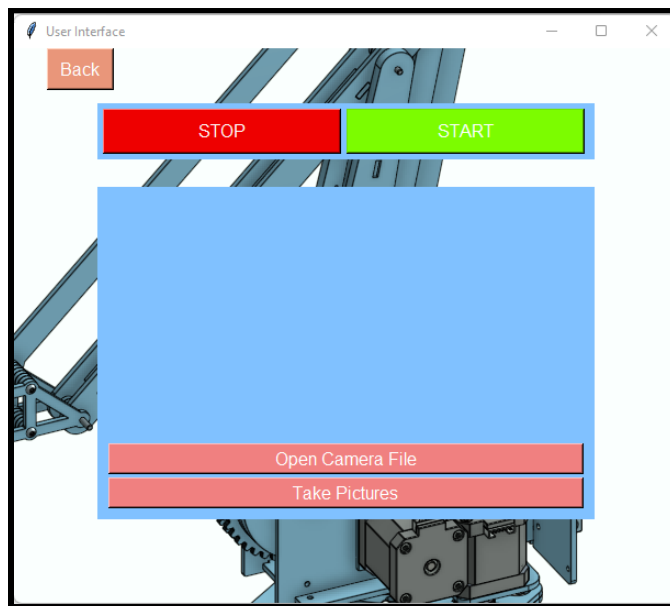


Figure 14. User Interface Camera Page

A back button also appears in the top left corner of the screen which, when pressed, replaces the “Open Camera File” and “Take Pictures” buttons with the end-effector selection buttons to allow the user to access the other control interface without closing the GUI. Once the camera code has successfully run, press the back button previously explained to return to a possible selection of the Paint end effector and its code options to then draw the desired design. When selected, the shape will be drawn on the surface if the proper end effector has been installed.

4.4 Paint End Effectors

Always ensure the end effectors are firmly attached to the robot arm using the appropriate bolt and nut fastener. No additional setup information is required.

4.5 Camera End Effector

To run the camera code, open the Arduino IDE from the camera files. Keep in mind the file contains the safety sensor code in the main void loop as well.

4.5.1 ArduImageCapture Instructions

Download the Github files in section 3.5 “Camera” under section 3 “Getting Started” and extract the zip file and copy the “ArduImageCapture” folder into the Arduino “tools” folder next to the Arduino “libraries” folder. Create a “tools” folder if it does not already exist.

- Example (on Windows): C:\Users\jessb\Documents\Arduino\tools

Creating the folder is essential in running the plug-in through the Arduino IDE. It can now be run through the IDE by clicking /tools/ArduImageCapture.

4.5.2 Compiling in Arduino IDE

Copy the LiveOV7670Library to the Arduino “libraries” folder. Open the Camera_Code.ino in the Arduino IDE. Make sure to select the appropriate board and COM pin (Select Tools/Board/Arduino Uno).

4.5.3 OV7670 Hardware Connections

See below the Arduino and OV7670 connections. The XLC input pin MUST BE LEVEL SHIFTED from 5V to 3.3V. See the page below on information about creating voltage dividers:

- Ohms Law Calculator: <https://ohmslawcalculator.com/voltage-divider-calculator>

OV7670 connections:

VSYNC - D PIN2

XCL - D PIN3 (must be level shifted from 5V -> 3.3V)

PCL - PIN12

SIOD - A4 (I2C data) - 10K resistor to 3.3V

SIOC - A5 (I2C clock) - 10K resistor to 3.3V

3.3V - 3.3V

RESET - 3.3V

D0, D1, D2, D3 - A0, A1, A2, A3 (pixel data bits 0, 1, 2, 3)

D4, D5, D6, D7 - PIN4, PIN5, PIN6, PIN7 (pixel data bits 4, 5, 6, 7)

GND - GND

PWDN - GND

4.6 Safety

Ensure to download the OV7670 Arduino IDE by referring to section 3.5 “Camera” under section 3 “Getting Started”. Once the Arduino IDE is downloaded and open, uploading and running the camera code will activate the safety sensors.

4.6.1 Hardware Setup

Some important remarks include the 270 ohm resistor needed from the PIR sensor to the LED. The PIR sensors require a 5V connection. A supply this large will short circuit the LED and damage the breadboard.

Follow the setup instructions below:

GND Arduino - GND

5V Arduino - POWER

POS Buzzer - PIN 9

NEG Buzzer - GND

POS LED - PIN 11 (270 ohm resistor)

NEG LED - GND

POWER PIR - POWER

GND PIR - GND

OUT PIR - PIN 9

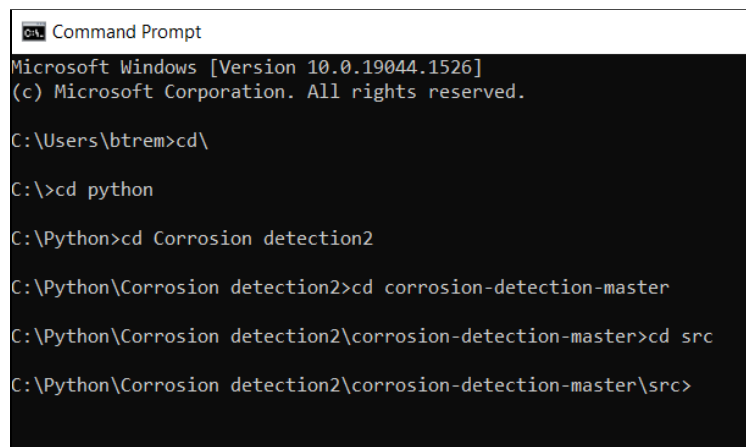
5. Troubleshooting & Support

5.1 Error Messages or Behaviors

5.1.1 Corrosion Detection AI.

Prototyping for the corrosion detection AI consisted mostly of fixing the numerous errors found within the code. This project is now 6 years old so many libraries have updated since and many strings have changed or been outright deleted which prevents the code from working properly. Below are the notes taken during the entire troubleshooting process and the various ways to fix the sections of the code that displayed an error message.

Firstly, when looking at the ReadMe file given by the code creator, it said to run a python command from the src directory (the file containing all the python files) to run them all at once. As a very inexperienced programmer, this was a big challenge. Python needed to be set up with a command prompt to run codes from the directory. To find the specific directory needed, use "cd NameOfFolder" several times to set up the directory to the src folder.



```
Command Prompt
Microsoft Windows [Version 10.0.19044.1526]
(c) Microsoft Corporation. All rights reserved.

C:\Users\btrem>cd\

C:\>cd python

C:\Python>cd Corrosion detection2

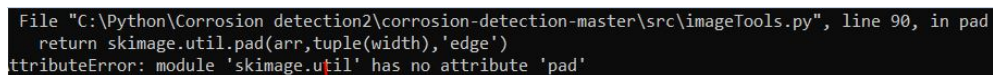
C:\Python\Corrosion detection2>cd corrosion-detection-master

C:\Python\Corrosion detection2\corrosion-detection-master>cd src

C:\Python\Corrosion detection2\corrosion-detection-master\src>
```

Figure 15. String of commands in Command Prompt to reach desired folder directory

The following error encountered was due to old code. The error that occurred had to do with the scikit-image library. Essentially the function `skimage.util.pad` gave an error message saying it does not have the `pad` attribute. The thread `skimage.util.pad` was removed from scikit-image, the patch notes stated that `numpy.pad` can be used for the same effect. After pip downloaded the NumPy library, the error message was fixed.



```
File "C:\Python\Corrosion detection2\corrosion-detection-master\src\imageTools.py", line 90, in pad
    return skimage.util.pad(arr,tuple(width),'edge')
AttributeError: module 'skimage.util' has no attribute 'pad'
```

Figure 16. Error message received because of scikit-image problems

The third encountered issue is found within the `imageTools.py` file, and it says `Assertion Failed`. To fix this issue change the directory to suit the files directory in the laptop.

```
Command Prompt
C:\Python>cd Corrosion detection2
C:\Python\Corrosion detection2>cd corrosion-detection-master
C:\Python\Corrosion detection2\corrosion-detection-master>cd src
C:\Python\Corrosion detection2\corrosion-detection-master\src>python main.py test
Initializing classifier...
rust.10.jpg
Traceback (most recent call last):
  File "C:\Python\Corrosion detection2\corrosion-detection-master\src\main.py", line 50, in <module>
    clf = Classifier(directories)
  File "C:\Python\Corrosion detection2\corrosion-detection-master\src\classifier.py", line 59, in __init__
    self.pos = get_imgs(directories["pos"], "pos", maxsize, pos)
  File "C:\Python\Corrosion detection2\corrosion-detection-master\src\classifier.py", line 48, in get_imgs
    return load_imgs(directory, group, maxsize=maxsize)
  File "C:\Python\Corrosion detection2\corrosion-detection-master\src\classifier.py", line 21, in load_imgs
    files.append(Image(fname, directory, group, maxsize=maxsize))
  File "C:\Python\Corrosion detection2\corrosion-detection-master\src\image.py", line 29, in __init__
    self.rgb = get_img(name, direc, maxsize, img)
  File "C:\Python\Corrosion detection2\corrosion-detection-master\src\image.py", line 19, in get_img
    return load_scaled(name, directory=directory, maxsize=maxsize)
  File "C:\Python\Corrosion detection2\corrosion-detection-master\src\imageTools.py", line 41, in load_scaled
    file = load(name, directory)
  File "C:\Python\Corrosion detection2\corrosion-detection-master\src\imageTools.py", line 32, in load
    return cv2.cvtColor(cv2.imread("../data/"+directory+"/"+name), cv2.COLOR_BGR2RGB)
cv2.error: OpenCV(4.5.5) D:\a\opencv-python\opencv-python\opencv\modules\imgproc\src\color.cpp:182: error: (-215:Assertion failed) !_src.empty() in function 'cv::cvtColor'
```

Figure 17. Error message from imageTools.py about cv2 (OpenCV)

To fix this change the directory to the first image in the file it is trying to read, so:

C:\Python\Corrosion detection2\corrosion-detection-master\data\test(name of first image) (ex:picture.jpg)

This fixed the issue, and the program finally imported and classified the images, which brought us to the following error. The images in the files are being shown as scanned and imported into the command prompt.

```
C:\Python\Corrosion detection2\corrosion-detection-master\src>python main.py test
Initializing classifier...
rust.10.jpg
rust.10.xml
rust.11.jpg
rust.11.xml
rust.13.jpg
rust.13.xml
rust.15.jpg
rust.15.xml
rust.16.jpg
rust.16.xml
rust.17.jpg
rust.17.xml
rust.18.jpg
rust.18.xml
rust.19.jpg
rust.19.xml
rust.2.jpg
rust.2.xml
rust.20.jpg
rust.20.xml
rust.23.jpg
rust.23.xml
rust.24.jpg
rust.24.xml
rust.25.jpg
rust.25.xml
rust.28.jpg
rust.28.xml
rust.29.jpg
rust.29.xml
rust.30.jpg
rust.30.xml
rust.31.jpg
rust.31.xml
rust.32.jpg
rust.32.xml
rust.35.jpg
rust.35.xml
rust.36.jpg
rust.36.xml
rust.4.jpg
rust.4.xml
rust.45.jpg
rust.45.xml
rust.46.jpg
rust.46.xml
rust.48.jpg
```

Figure 18. Images from the test folder being processed within the program.

In the Image.py file, there is an issue on line 99 and 126. The error message is: if self.hsHist != None and hist!= None:ValueError: The truth value of an array with more than one element is ambiguous. Use a.any() or a.all(). Add .any() after each hist in lines 99 and 126.

The error string below should be the one shown in the paragraph below. This is a relatively simple fix. On line 23 of the main.py change arg to argv. There will also be the same arg string a couple lines down, change that to argv as well.

File "C:\Python\Corrosion detection2\corrosion-detection-master\src\main.py", line 23, in getInput

```
edgeThr = sys.arg[3]
```

AttributeError: module 'sys' has no attribute 'arg'. Did you mean: 'argv'?

The before last error was once again an issue with scikit-images changelogs. When using the .remove_small_holes function in the program, the operation min_size was no longer a function that existed. Using area_threshold instead fixed the critical issues and brought us to the final result.

Here is a video showing the program being executed and its end results: <https://youtu.be/xPfUXLRoVfo>

5.1.2 User Interface

There are not many troubleshooting issues for the code, however this section will cover the few common possible errors. First off is in the setup section of the manual, where it is mentioned to put in the proper path for the image's file. This is sometimes a problem since once the path is copied and pasted from the files area of the computer, its format is a bit different than it should be to work. The backslashes must all be doubled and it must be confirmed that the whole path is wrapped with quotation marks and parentheses, if not, the image will not be found and the background will be blank. This error should not cause any major issues but does take away from the look of the interface. There are also paths to open the proper other codes and apps needed to run everything smoothly and they will have to be double checked for the same formatting issues that would cause malfunctions.

The other already previously mentioned troubleshooting error is not having the proper libraries imported and installed. There is a Pillow library and filedialog used in the code which must properly be installed in the command prompt and imported into the code with the proper lines at the very beginning.

Sometimes the formatting of the placement of widgets is for some reason disliked by the python runner and so it is possible that everything including and after the ".place" or ".pack" have to be moved down into a new line below with the name of the widget to be placed or packed in front.

5.2 Special Considerations

For the IK code, make sure to not put any negative values for the x coordinate to make sure that the arm stays intact.

When downloading Python check the "Add to PATH" box before completing the download. The code will not run without this setting being enabled.

5.3 Maintenance

The majority of this project does not require maintenance, the only thing is the marker needs to be replaced when the ink is depleted.

5.4 Support

For the corrosion code using the changelogs for the various python libraries will help in updating the code in the future. In addition, if there is confusion, fixing certain issues using StackOverflow is a

very important tool. StackOverflow is a website where users can create threads discussing certain Python issues that other users encountered. The link to the website is the following: <https://stackoverflow.com/>

Any other coding issues also have troubleshooting explanations available on the same website <https://stackoverflow.com/>

6. Product Documentation

6.1 Prototype III

6.1.1 BOM (Bill of Materials)

The first bill of materials was created with only an idea of the kinds of parts that were needed for this project. In addition, the budget restraints were uncertain as they were still shifting and being finalized. It was decided that the majority of our money would be spent on the camera end effector. The programming/software portion was assumed to be free so that the money could be spent on the physical components. The client had mentioned that the most crucial parts of this project were the inverse kinematics, GUI (user interface), safety, and finally the camera end effector. This worked out perfectly since the corrosion detection program we found could work alongside the camera end effector. There are PIR motion sensors in the bill of materials because they will be used as the main tool for the safety part of the arms project. One of the main issues with the cost estimate is the shipping times. With only about 2-3 months to build, code and design the end effectors and arm the shipping time of various materials had to be taken into account because they were sourced from websites like amazon. This occasionally led to more expensive parts being chosen to still have time to test after being delivered.

Table 1. Original bill of materials with total product cost estimate

Item Name and Link	Quantity	Cost (\$)	Justification
Camera Arducam with adapter board	1	23.68	The camera chosen needs to be compatible with Arduino software in order to access the data (live video feed) and send it to other devices.
PIR motion sensors PIR motion sensors	1	10.59	These sensors will be added to different end-effectors to ensure safety while operation. (soldered)
3D printing materials		0.00	Since most of our end effector components will be 3D printed, we will be using the machines and materials provided in the Maker Lab.
Arduino kit and wires	1	0.00 (Free at Maker Lab)	The Arduino will be useful for the spray guns in order to connect the sensors and triggers to a specific output in our software. This kit includes a breadboard and some resistors in order

Soldering kit	1	0.00 (Free at Maker Lab)	Used to mend our wires together and solder them to our things like our camera and sensors to ensure that they will not be easy to break off or to fall apart simply by moving.
Sharpie Marker	1	0.00 (Free from home)	
Total product cost (without taxes or shipping)		38.67	
Total product cost(including taxes and shipping)		46.61	

The final bill of materials received important changes that were made to stay under budget. Firstly, the camera module picked would take a long time to ship as well, and once we got our bill of materials approved the price jumped up to 37\$ which was a major issue with the budget only being 50\$. It was decided that the OV7670 camera would be used because it was cheap, could arrive rapidly and there were multiple tutorials found on how to program/use the camera. The tutorials help with cutting down on the time needed to troubleshoot that portion of the project. It was also decided to change the paint remover end effector which no longer required adjustable clamps further lowering the budget. Finally, the old PIR sensors were swapped with a more expensive kind but with the budget cuts, a product that had higher reliability according to amazon reviews was available to purchase. Many components remained the same, the Arduino board, 3D printing materials, bolts, and nuts. These were because they are essential parts of the composition of the product. For example, the Arduino board allowed the testing of the camera, sensors, and even the inverse kinematics code without the arm being complete yet.

Table 2. Bill of materials with total product cost estimate

Item Name and Link	Quantity	Cost (\$)	Justification
Camera OV7670 VGA CMOS Camera	1	23.68	The camera chosen needs to be compatible with Arduino software in order to access the data (live video feed) and send it to other devices.
PIR motion sensors PIR motion sensors	1	14.99	These sensors will be added to different end-effectors to ensure safety while operation. (soldered)
3D printing materials		0.00	Since most of our end effector components will be 3D printed, we will be using the machines and materials provided in the Maker Lab.

Arduino kit and wires	1	0.00 (Free at Maker Lab)	The Arduino will be useful for the spray guns in order to connect the sensors and triggers to a specific output in our software. This kit includes a breadboard and some resistors in order
Bolts and Nuts	14	0.00 (Free at MakerLab)	Used to attach end effector parts together and attach the end effectors to the robot arm.
Sharpie	1	0.00	The Sharpie is used for the paint/corrosion remover end effector to demonstrate its functionality to the client on design day.
Total product cost (w/o taxes or shipping)		38.67	
Total product cost (including taxes and shipping)		46.61	

6.1.2 Equipment list

Table 3. Equipment list and resource description

Item Name	Description	Type	Prototype #	Source
CAD software (Onshape)	This will be used to create a computer-aided design of different end effectors.	Analytical (Software)	1	https://www.onshape.com/en/
Arduino Studio (Tinkercad)	To test circuits.	Temporary software	2	https://www.tinkercad.com
3D printer	To 3D print all end effectors and attachment pieces.	Equipment.	3	MarkerSpace
Coding Software(Python, Arduino IDE)	To implement code.	Software	4	Personal device

6.1.3 Instructions

Run the GUI code and a window will pop up.

The functions of the buttons that show up are explained in more detail in the “Using the System” section, but they will be briefly explained again here. The distance of the robot arm from the wall should be input in meters in the white box at the top of the screen and input into the inverse kinematics code by pressing the button next to it.

Next, select the button with the end effector that you have installed, usually the camera one first. Run the camera and corrosion detection codes using the “Open Camera Files” button as well as the star button and any emergency can be interrupted by the pressing of the stop button at the top.

Next, press the back button to return to the main page once the camera code has been completely executed and press the next end effector button, which should be the paint one. This will bring the user to the final interface page, which presents three new options in the center of the screen, three different shapes that can be drawn by the arm. Once one of them is selected the code will begin and can be stopped or resumed by the stop and start buttons at the top of the screen.

Once the job is complete, simply close the pop up window GUI and it will stop running the code altogether.

Link to code : [FinalGUI_code.py](#)

6.1.3.4 End Effectors

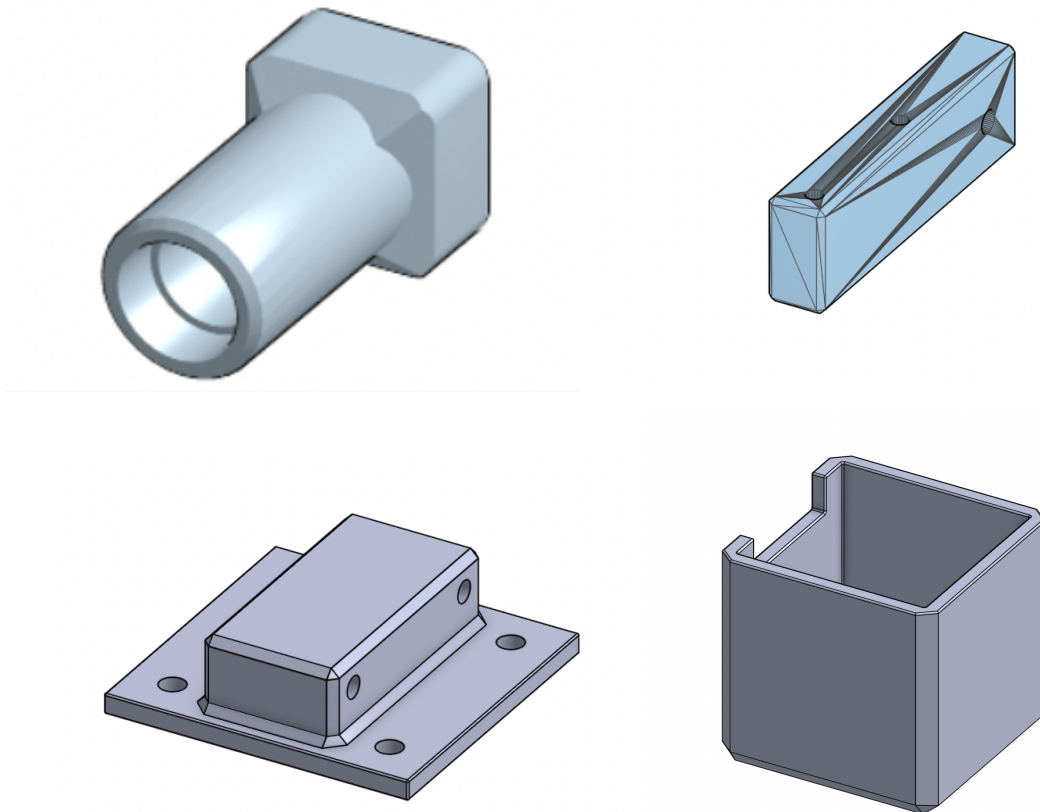


Figure 19. End-Effector Models

Attach the sharpie holder and the rectangular block together using a bolt and nut and place a sharpie firmly into the sharpie holder. The final product should look like the image below.



Figure 20. Drawing End-Effector

Now, place the camera into the 3D printed box and place the lid on top. Secure the lid using four M3 bolts and nuts. The final product should look like the image below.

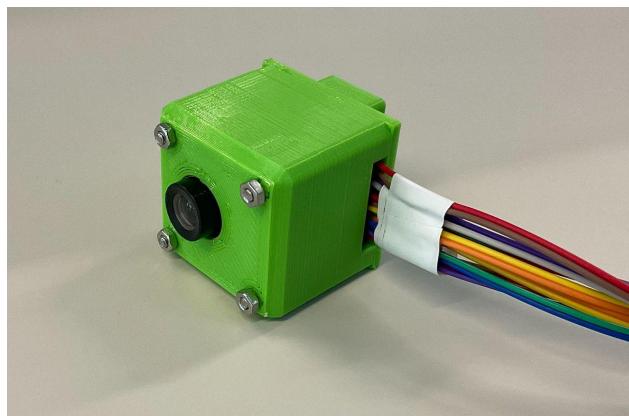


Figure 21. Camera End-Effector

6.1.3.5 Safety

With the same breadboard used and arduino used for the camera, follow the instructions below to wire the sensors, LED and buzzer together.

Using the same code as that of the camera test and uploading that code from your computer to the Arduino, test the functionality of the sensors by passing your hand in front of the sensors

6.2 Testing & Validation

Test #	Objective	Description and Test Method	Test Duration and Date	Results
1	Mathematical code concept	To have a logical and functional mathematical approach of the functionality and movement of the arm.	1 or 2 days Reading week	The mathematical concept has been achieved and implemented into code.
2	Analysis of materials	Lots of materials are used in this design, such as different 3D printable materials, cameras, sensors and Arduino components such as the wires and diodes. These will have to be tested for their effectiveness and researched extensively	2 or 3 days Reading week	Through research we have found sensors, cameras and more that will be compatible with our Arduino as well as with the end-effectors that are to be 3D printed
3	Engineering drawing of end-effectors	Detailed engineering drawing on paper of our design and the orthographic projections to show all sides and important components	2 days Reading week	Hand drawn engineering drawings have been made and are able to be transformed into Solidworks 3D models with a few necessary modifications but have proven very useful to the transformation into 3D models
4	Basic code for arm movement	Once the mathematical concept is achieved and the inverse kinematics equation is understood, the equations can be translated to code for future testing	2 or 3 days While the drawings are being made	The code compilation is complete with no errors and using only 29% of storage. Code has been tested on tinkercad for servo motor response.
5	3D modelling on Onshape or Solidworks	3D drawing or model on a 3D modelling site to determine our "final" design with more precision and to better our understanding of our design and ensure our understanding of it.	2 days As soon as engineering drawing is done	Successfully created our end-effectors using Solidworks and transferring them to onshape for any last minute modifications
6	Camera and corrosion detection code	If all goes well, the corrosion code we have found may be accessible to us and may be able to be translated, and that translation to a language that we	4 days While drawings and models are being made	

		understand would be this step.		
7	Create user interface and test with what we have	Attempt different user inputs and see how these are processed and outputted compared to the expected outcome.	1 day Before the first session with robot	Buttons do not function properly, give errors when they are coded with a purpose but the look that we want and need is achieved
8	Test materials with what we currently have	Materials have been analyzed, and the best ones are chosen and must be put to the test to see if they are good for our product. They will be tested in durability and compatibility with the arm and the code.	2 days First session with robot arm	Parts to be used on robot and arduino have been acquired and seem to be sized properly and not weigh too much that they are able to be used in our 3D printed end effectors that are also in a material that works and is not too heavy
9	Test arm movement code on arm	The algorithm and code for the inverse kinematics movement of the arm should be completed, and it will be tested on the arm as soon as the opportunity presents itself so that any issues are discovered and it can be modified accordingly quickly	1 day First session with robot arm	Will be tested once the robot arm is completed March 14th, 2022
10	Paper or cardboard quick prototype	Quickly make a 2D and/or 3D tangible model of end-effectors as a size comparison to the actual robot and objects that will be used with them to be sure of our dimensions	> 1 day First session with robot arm	Will be tested once the robot arm is completed March 14th, 2022
11	Retouch engineering drawing and 3D modeling of end-effectors	Any miscalculations or wrong dimensions are discovered through the previous tests and now the drawings and models can be readjusted to accommodate our new discoveries	1 day After first session with robot	Will be completed one day after the first session with the robot, March 15th, 2022.
12	3D printed model of what we have designed so far	The 3D model is adjusted and can now have the pieces printed and assembled for testing on the robot. If the previous analysis and prototyping were effective, this should be done once or twice to minimize the number	1 or 2 days Second session with robot arm	Will be completed during second session with the robot arm, March 17th, 2022

		of materials used and the overall cost		
13	Test code and user interface with newly 3D printed pieces and arm	Pieces are printed and the end-effectors are assembled, everything can be wired and plugged into the Arduino in its respective place, and the code can be tested on the arm and the user interface. If any errors occur, the code and user interface will have to be modified accordingly	Consistent with prototype testing. Five consecutive test trials with no errors.	The code is functioning with the newly printed end effectors. The user interface has yet to be implemented with the code.
14	Make sure attachments and necessary scenarios are compatible with end-effectors and code	The final test will entail putting all pieces together for one last test, running multiple scenarios with the user interface, arm and all the end effectors to simulate the users' experience and ensure that it is possible, simple and easy to understand for the high school students who will most likely be running the interface and interchanging the end effectors	Consistent prototype testing. Five consecutive test trials with no errors.	Corrosion remover attachment works with the inverse kinematics code, it draws a square which will be the coordinates for the search pattern.
15	Adjust all necessary things and create the final versions of end-effectors, code and user interface	Once the group and client have settled on a final version and has been through the tests previously mentioned, it is time to bring it to life and create the final version of everything necessary, test it on the robot arm and if all goes well, there will no longer be any need for prototyping	Either run out of time or be satisfied with the final product before design day	Final end effectors, inverse kinematics code, safety features and corrosion detection have been implemented and tested.
16	Test Corrosion detection program and fix bugs to properly detect corrosion	Currently the code struggles with assigning values according to whether there is corrosion or not in the picture. With the TA and other people we will try and fix the current issue	Either run out of time or until the error is fixed within the code	Final version of the corrosion detection code has been fixed and debugged.
17	Camera pictures running through	Now that the camera is complete we can save the pictures it takes into the input file for the corrosion code so it scans those pictures taken	Code needs to work properly to detect properly but we can test	The images taken with the camera now successfully run through the corrosion detection

	Corrosion Program		the input method now that the camera works and the code runs	software.
--	----------------------	--	--	-----------

Table 4. Testing and planning

7. Conclusions and Recommendations for Future Work

For a software heavy project this turned out to be a very big challenge for a group of students with little to no past programming experience. We had to learn programming of all types, C++ for the inverse kinematics, Python for the GUI and Corrosion Detection AI and Arduino for the camera, sensors and inverse kinematics. Learning all these different types of programming proved to be a challenge spending countless hours watching youtube videos, reading online websites and threads to understand how the different subsystems work and how to implement them. Overall we are very proud of the work we have done along with our final prototype that was presented for design day. The following paragraphs are different lessons learnt and a plan for if we had more time to work on this project.

In order for our team to develop a strong solution to the problem initialized by John Faurbo, a clear set of interpreted needs and design criteria are essential. Functional and non-functional needs, as well as constraints and target specifications must be thoughtfully planned in order to create the best solution to the problem.

Upon completing sketches to demonstrate our ideas, we discussed and analyzed possible consequences for each conceptual design. This allowed us to idealize and combine the best of our collective ideas into five functional subsystems to our solution. We were able to successfully stay within our constraints and produce descriptive sketches of all of our ideas to eventually bring them to life.

It is critical to plan our prototypes accordingly as design day is quickly approaching. Our team is dedicated to producing the best product possible, covering all our clients' needs. Planning responsibly according to delays, changes, and unexpected, unforeseen circumstances is essential. After the client meeting, the team took the feedback and prepared our bill of materials, prototype test plan, and final design drawings.

The prototyping portion of this product did prove to be the most challenging portion, as expected. We have had our fair share of difficulties with this prototype from our many failed attempts at end-effectors and coding troubles. However, our plan has slowly but surely come together, and we will continue the improvement of our prototype to produce the best product possible for our client. We had to often work together to solve the issues or ask for help from outside sources such as the TAs and other students/friends. This helped us to learn teamwork as well as further understanding our different skills, strengths and weaknesses. Our team has made substantial progress both individually and as a team.

If we had a couple more months to work on this project we would have focused our time and energy on making all of the different components work together more smoothly so that the user is able to easily control the robot and do its different tasks. We found how to simultaneously make all the components work together but due to time constraints we were unable to. In addition, we would like to work on improving the accuracy of the corrosion detection code.

Finally, our group was very productive so in the end all of the subsystems we intended to implement ended up being in the final product. Our original plan was to have the camera end effector, paint end effector, inverse kinematics, corrosion detection AI, safety feature and GUI.

Bibliography

- Agustian, I. (2021, March 25). *3DOF inverse kinematics for arm/leg of robot using Arduino*. Electrical Engineering Dept. University Of Bengkulu. Retrieved March 4, 2022, from <http://te.unib.ac.id/lecturer/indraagustian/2014/05/3dof-inverse-kinematic-for-armleg-of-ro-bot-use-arduino/>
- Arducam. (2016, December 19). *Arducam 2 megapixels MT9D111 Auto Focus Lens Camera Flex module with Adapter Board*. Amazon.ca: Electronics. Retrieved February 12, 2022, from <https://www.amazon.ca/Arducam-Megapixels-MT9D111-Camera-Adapter/dp/B013O8QB8O>
- Arducam. (n.d.). *Arducam Mini Module Camera Shield with OV2640 2 megapixels lens for Arduino Uno MEGA2560 Board & Raspberry Pi Pico*. ucronics. Retrieved February 12, 2022, from <https://www.ucronics.com/arducam-mini-module-camera-shield-w-2-mp-ov2640-for-arduino-uno-mega2560-board.html>
- Arduino Engineer. (2015, March 18). *Autonomous paintball sentry gun using Arduino*. Use Arduino for Projects. Retrieved February 14, 2022, from <https://duino4projects.com/autonomous-paintball-sentry-gun-using-arduino/>
- Arduino. (2022). *Servo*. Servo - Arduino Reference. Retrieved March 4, 2022, from <https://www.arduino.cc/reference/en/libraries/servo/>
- anirbankonar123. (2019, May 7). *ANIRBANKONAR123/corrosiondetector: Corrosion detection from images*. GitHub. Retrieved March 6, 2022, from <https://github.com/anirbankonar123/CorrosionDetector>
- Bzager. (2017, August 24). *Bzager/corrosion-detection: Image processing for automated detection of Steel Corrosion*. GitHub. Retrieved March 6, 2022, from <https://github.com/bzager/corrosion-detection>
- Galli, K., 2019. *How to Program a GUI Application (with Python Tkinter)!*. [video] Available at: <<https://www.youtube.com/watch?v=D8-snVfekto>> [Accessed 13 March 2022].
- Generic. "RedTagCanada OV7670 VGA CMOS Camera Image Sensor Module for Arduino Supports VGA CIF 640X480 Compatible I2C Interface." *Amazon.ca: Electronics*, 2021, https://www.amazon.ca/gp/product/B09JYZX32L/ref=ppx_yo_dt_b_asin_image_o00_s00?ie=UTF8&psc=1.

- GHEDIRI, A. (2017, June). *Design and Construction of Robotic Palletizer*. Faculty of Sciences and Applied Sciences, Department of Electrical Engineering. Retrieved March 3, 2022, from [Design and Construction of RoboticPalletizer](#)
- Gperco. (2013, December 20). *Robot arm: Reaching for the stars*. Robot Arm: Reaching for the Stars. Retrieved March 4, 2022, from <http://www.gperco.com/2013/12/robot-arm-reaching-for-stars.html>
- HiLetgo. "Amazon.com : 3pcs HC-SR501 PIR Infrared Sensor Human Body ..." *Amazon*, 2018, <https://www.amazon.com/HC-SR501-Infrared-Sensor-Arduino-Raspberry/dp/B085M7WSMY>.
- Konar, A. (2019, August 16). *Using tensorflow object detection API for corrosion detection and localization*. FloydHub Blog. Retrieved February 11, 2022, from <https://blog.floydhub.com/localize-and-detect-corrosion-with-tensorflow-object-detection-api/>
- Kumar, V. (n.d.). *Robot geometry and Kinematics - University of Pennsylvania*. Robot Geometry and Kinematics. Retrieved February 10, 2022, from <https://www.seas.upenn.edu/~meam520/notes02/IntroRobotKinematics5.pdf>
- K&RProject. (2018, January 22). *Arduino with PIR motion sensor, led and Buzzer (code)*. Arduino with PIR motion Sensor, LED and buzzer (Code). Retrieved March 1, 2022, from <https://kandrproject.blogspot.com/2018/01/arduino-with-pir-motion-sensor-led-and.html>
- Lazzari, E. (2022). *Downloading Python*. BeginnersGuide/Download - Python Wiki. Retrieved March 5, 2022, from <https://wiki.python.org/moin/BeginnersGuide/Download>
- Luuk, I. (2021). *Simplified! how to use OV7670 camera with Arduino*. SIMPLIFIED! How to Use OV7670 Camera with Arduino. - Circuit Journal. Retrieved February 7, 2022, from <https://circuitjournal.com/arduino-OV7670-to-pc>
- Mon, S. (2017, July 11). *Arduino using inverse kinematics (ik) - youtube*. Youtube. Retrieved March 4, 2022, from <https://www.youtube.com/watch?v=Y8ueTjqCcAg>
- Naing, W. W., Aung, K. Z., & Thike, A. (2018). *Position control of 3-DOF articulated robot arm using PID*. International Journal of Science and Engineering Applications. Retrieved February 12, 2022, from <https://ijsea.com/archive/volume7/issue9/IJSEA07091001.pdf>

Ohms Law Calculator. (2022). *Voltage divider calculator*. Ohm's Law Calculator. Retrieved March 1, 2022, from <https://ohmslawcalculator.com/voltage-divider-calculator>

Onyehn. (2018, September 27). *Onyehn ir pyroelectric infrared PIR motion sensor detector modules(pack of 2pcs), motion detectors - Amazon Canada.* , Motion Detectors - Amazon Canada. Retrieved February 12, 2022, from https://www.amazon.ca/Onyehn-Pyroelectric-Infrared-Detector-Modules/dp/B07GJDJV63/?ref=sr_1_11?crd=3A3YR2R3MK01B&keywords=arduino%2Bpir%2Bsensor&qid=1644689563&sprefix=arduino%2Bpir%2Bsensor%2Caps%2C71&sr=8-11#customerReviews

Pagurek, D. (2017, March 12). *Simple Inverse Kinematics*. Simple Inverse Kinematics - Dave Pagurek. Retrieved February 11, 2022, from <https://www.davepagurek.com/blog/inverse-kinematics/>

The Arduino Team. (2022). *Downloads*. Arduino. Retrieved February 7, 2022, from <https://www.arduino.cc/en/software>

Waveshare-Module. (2014, September 24). *WaveShare OV9655 Camera Board CMOS SXGA 1.3 megapixel camerachip module development kit*. Amazon.ca: Electronics. Retrieved February 12, 2022, from <https://www.amazon.ca/OV9655-Camera-Board-CameraChip-Development/dp/B00KM6WYTM>

Xingyiheng. (2019, June 4). *Xingyiheng 5Pcs HC-SR505 Micro Body Sensing Module pir motion detector switch module high power high efficiency digital measurement for electronic practice DIY, motion detectors - Amazon Canada.*, Motion Detectors - Amazon Canada. Retrieved February 12, 2022, from https://www.amazon.ca/XLX-HC-SR505-Efficiency-Measurement-Electronic/dp/B07QY7GPWT/ref=sr_1_3_sspa?crd=1SK6R56RF007E&keywords=pir%2Barduino&qid=1644689638&srefix=pir%2Barduino%2Caps%2C62&sr=8-3-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUExOzU0NURZR0MxV0s1JmVuY3J5cHRlZElkPUEwMDAzNjQ3MIJROFhSRVFQUSTRZWSZlbmNyeXB0ZWRBZEIkPUEwNTA2OTI4MjlWM0pBWjFXRU9XWSZ3aWRnZXROYWw1IPXNwX2F0ZiZhY3RpbnY2xpY2tSZWRpcmVjdCZkb05vdExvZ0NsaWNrPXRYdWU

Appendices

Appendix I: Design Files

Document Name	Document Location and URL	Issuance Data
Corrosion Detection Folder	https://github.com/bzager/corrosion-detection	This is the folder that the original corrosion detection AI was found from.

Table 5. Referenced Documents

MakerRepo link: <https://makerepo.com/Becap/1113.gng1103c1group-3armageddon>