Project Deliverable F - Prototype I and Customer Feedback

GNG 1103 Group C03

Benoit Tremblay (300236751)
Rebeca Poulin (300236741)
Jess Beardshaw (300227707)
Isabelle Barrette (300228564)
Sarah Alkadri (300245117)

University of Ottawa
March 6th, 2022

Wrike Link :
https://www.wrike.com/workspace.htm?acc=4975842#path=folder&id=824968763&c=list&vid=64521612&a=4975842&t=848163524&so=10&bso=10&sd=0&f=&st=nt-1

Introduction

Design day is approaching faster than we think, and now is the time to work on all aspects of the project in a tangible way to begin testing what we will inevitably be presenting for the milestone that is design day. This means beginning prototyping all the different parts of this project such as the end-effectors, the user interface, the inverse kinematics function of the robot, the corrosion detection program (which is still a maybe since we may not be able to get the open-source codes found to work) and the safety aspect of the robot and its end effectors. All of these tasks have been dispersed between the group in mostly even loads of work depending on people's strengths (whoever has done less work on this prototype will have more work on the next and vice-versa) to be able to achieve as much as possible within the limited timeframe that we have.

1. Client Feedback

During our interview with the client on February 17th 2022, we received positive and constructive feedback. He shared his enthusiasm towards our projected idea of the corrosion detection code. Our client agreed with our light detection mechanism idea and said that it would be efficient and help the robot arm complete its tasks in time.

Theodore reminded us to keep our budget constraints in mind while creating our prototypes. He expressed his concerns for the safety of others during the use of the robot arm. We described how we will in fact be incorporating safety measures in our final product. Our client outlined the importance of the robot’s ability to work in low oxygen spaces to clean pumps with solution.

Overall, our client is mostly concerned about the kinematics of the robot. This will help us prioritize this as we move on with prototypes I through III.

From the answers provided to us by our client, we learned that the user will most likely know that there is corrosion ahead of time. This helped us shift our priorities regarding the corrosion detection software. He also shared that he is not so concerned about autonomy. However, it would be an asset to have the robot articulate and survey an area to then flag users of corrosion so that they can get rid of it.

2. Prototype I

For our first prototype, we have included a combination of physical and analytical designs to present to our client. Our plan with the first prototype is to rough draft the sketches for each end effector, as we do not have any physical products (i.e. the camera, what the hose looks like, etc…) to reference. We plan to compose our main conceptual idea, and focus on exact sizing and measurements once we have received our camera and know more about the corrosion removing and painting devices.

2.1 Camera End Effector

We developed our first prototype of the camera end effector, completing objective three and five in our test plan. We 3D printed CAD models from the sketches designed during reading week and discovered what we need to change for our next prototype.

Our first conceptual design is a semi-circle shape, where the camera will be inserted in the middle, and a hole in the back for connection wires to come out. We included clips that attach to the posterior side and clip to the anterior semi-circle to access the camera if needed. Everything except for the bolts and nuts will be 3D printed.

The rear of the end effector is attached to the finger of the robot using phillips M3 - 0.5 35mm bolts and fastened with spacers in the empty space and bolts on either end.

2.1.1 Hand Sketches

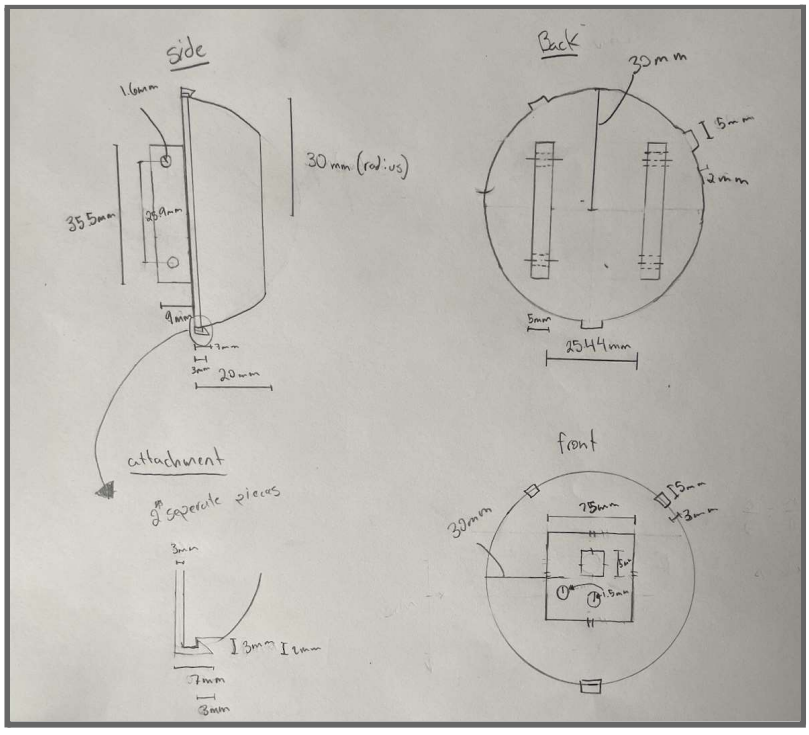


Figure 1. First engineering drawing of camera end effector with clip attachment

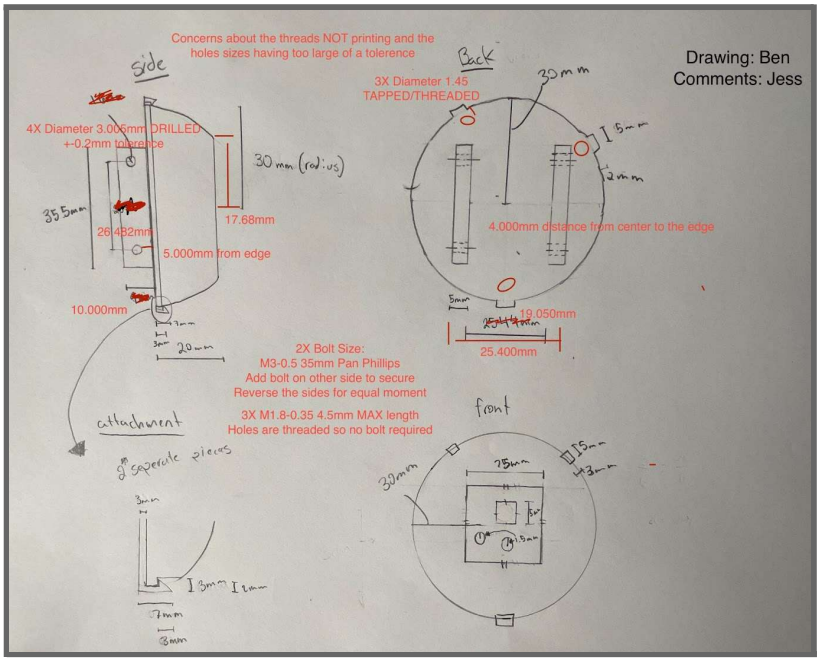


Figure 2. First engineering drawing of camera end effector with comments during CAD modelling

2.1.2 CAD Model

The CAD model is designed on Solidworks, and imported to On Shape where all team members can access the 3D printable files and assemblies.

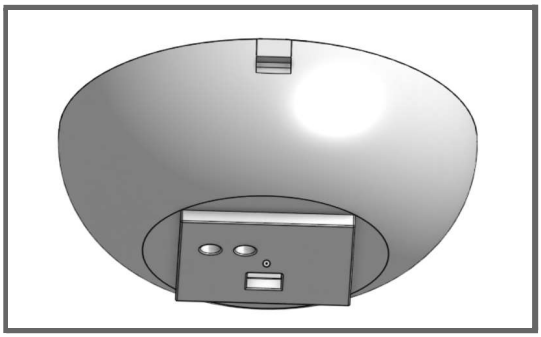


Figure 3. Anterior view of camera end effector

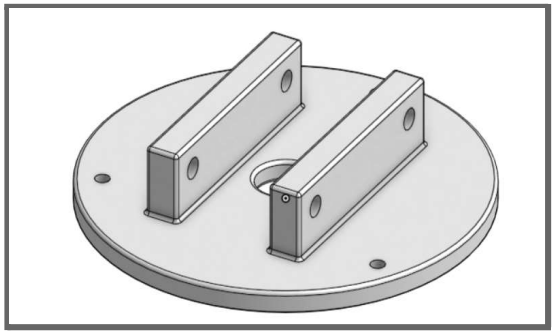


Figure 4. Posterior view of camera end effector

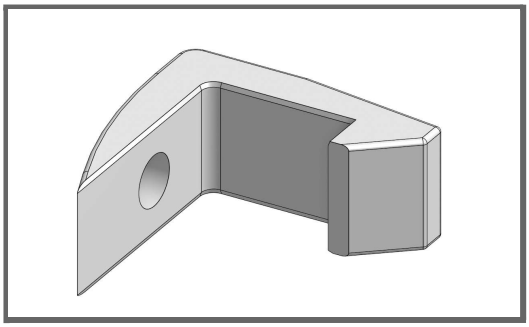


Figure 5. Orthogonal view of camera end effector clips

2.1.3 CAD Drawings and Measurements

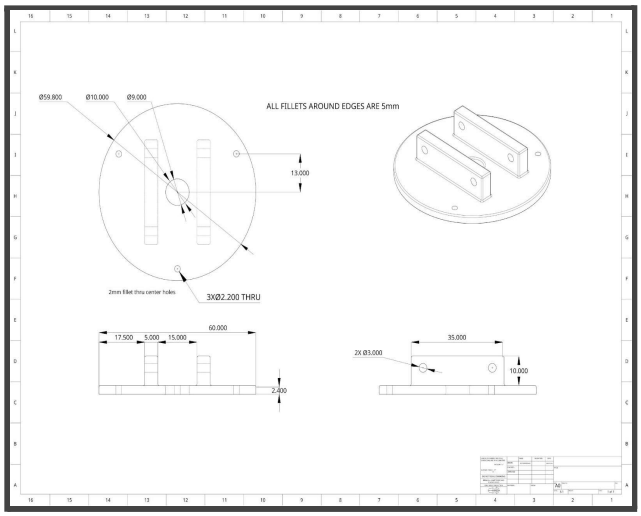


Figure 6. Drawing of posterior view of camera end effector

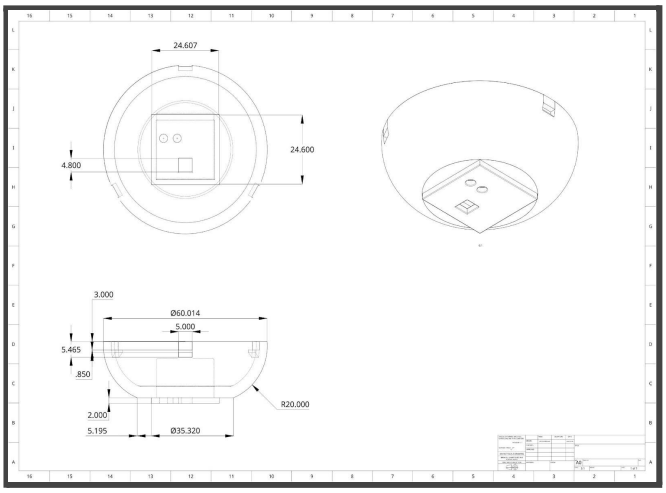


Figure 7. Drawing of anterior view of camera end effector

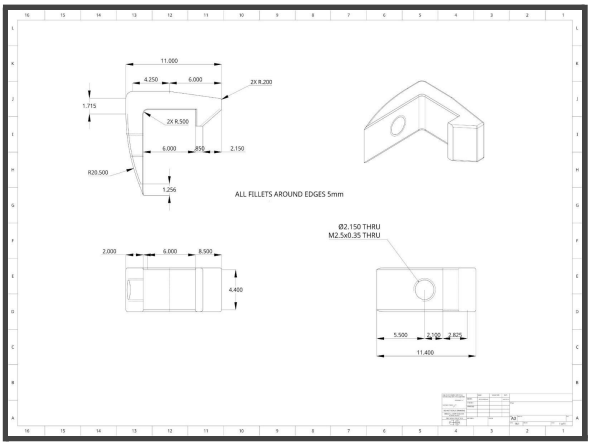


Figure 8. Drawing of camera end effector clips

2.1.4 3D Printed Product

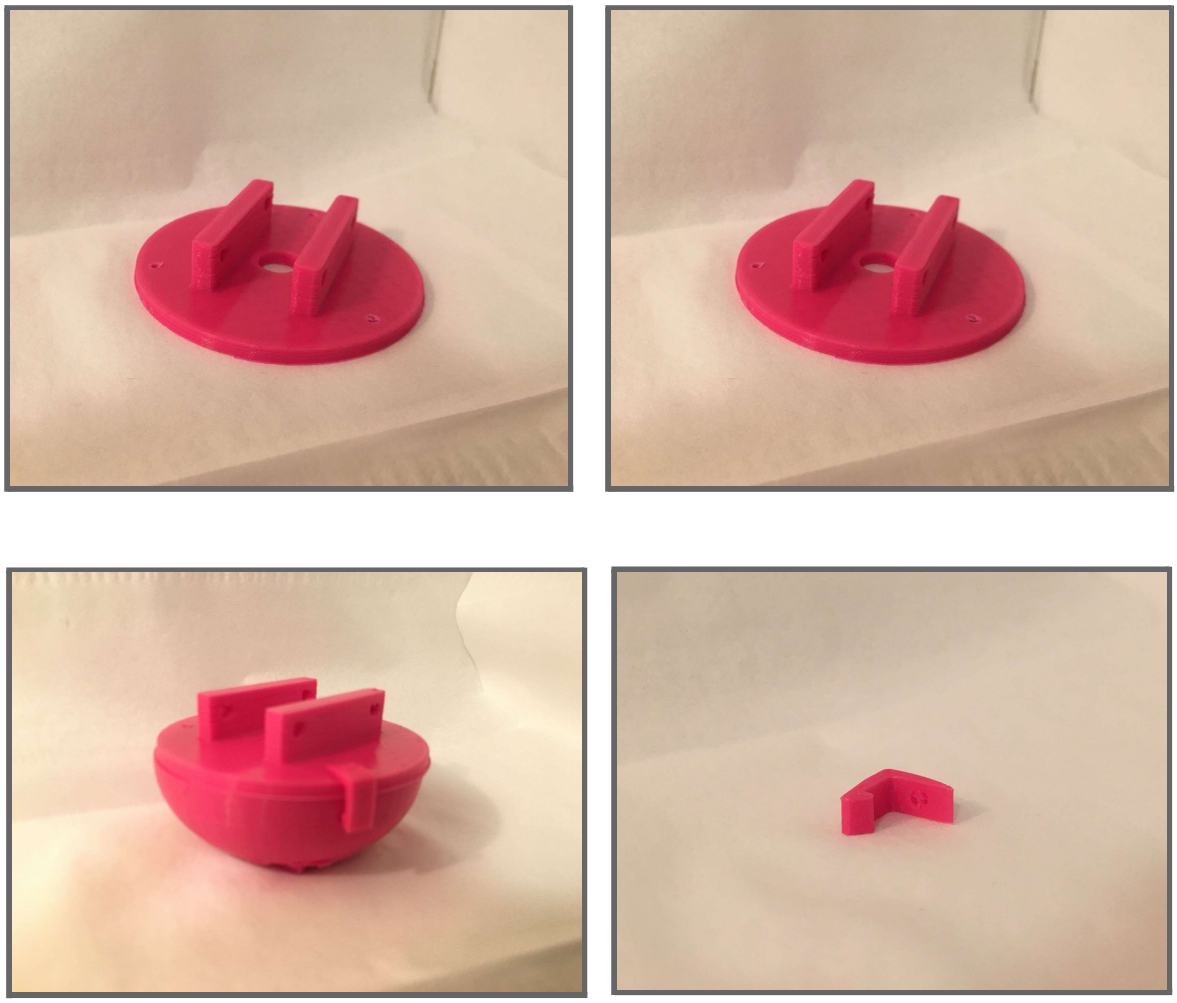


Figure 9. First physical prototype of camera end effector

The posterior and anterior parts were printed using a 0.8mm thread to decrease printing time. Our focus is having a base design product in order to decide what went good, and what went wrong. Our initial concerns are the inserts for the clips on the posterior view, not knowing if they would print properly and accurately. Since these clips were holding up the mass of the camera, end effector, and the torque of the moment of the arm, we decided to scrap the clip design for the next prototype. The functional components required are being able to support the mass of the camera, connect to the fingers of the robot arm, and be able to connect the wires of the camera to the arduino and breadboard from the outside.

We also discussed if it is necessary to be able to remove and access the camera in the end effector, and decided it is not necessary and proposed an obstacle in designing a product with the critical criteria mentioned above.

We printed the clips with 0.8mm thread and decided to print another set with a 0.4mm thread due to imperfections and the bolt hole uncertainties being too large. We also increased the size of the hole to fit a M2.5 0.35mm bolt, instead of the original M2 0.4mm to decrease the chances of large uncertainties.

2.2 Corrosion Remover and Painting End Effector

We developed our first prototype of the corrosion remover and painting end effector, completing objective three and five of our test plan.

Our first conceptual design is a clamp style with adjustable notches depending on the size of the object being held. Our intention is to have one end effector for the two functions for a low complexity design and to minimize user interaction. The end effector is adjustable for various object widths using a single bolt and nut for each adjustment setting. The adjustable legs of the top piece insert into the slot of the main body.

The rear of the end effector is attached to the finger of the robot using phillips M3 - 0.5 35mm bolts and fastened with spacers in the empty space and bolts on either end.

2.2.1 Hand Sketches

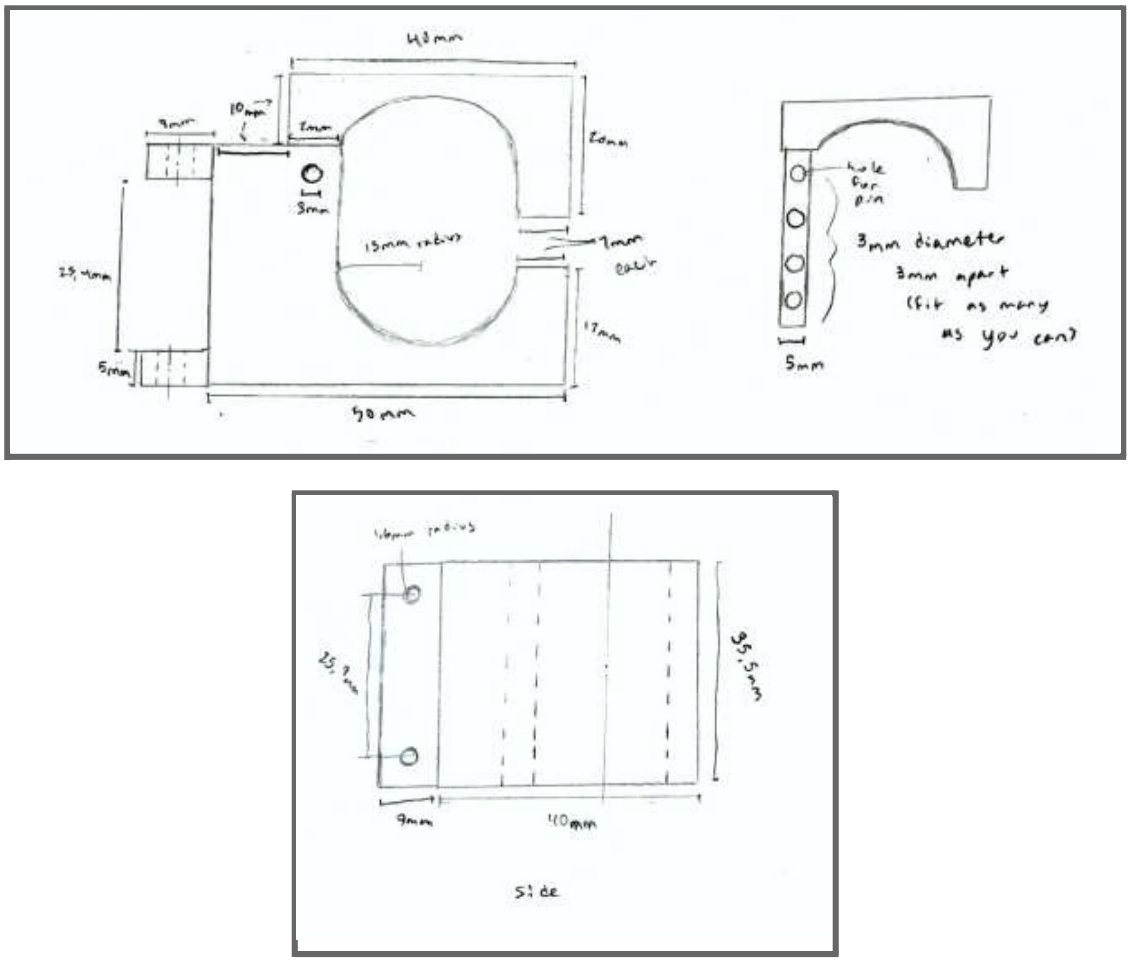


Figure 10. First engineering drawing of corrosion remover and painting end effector

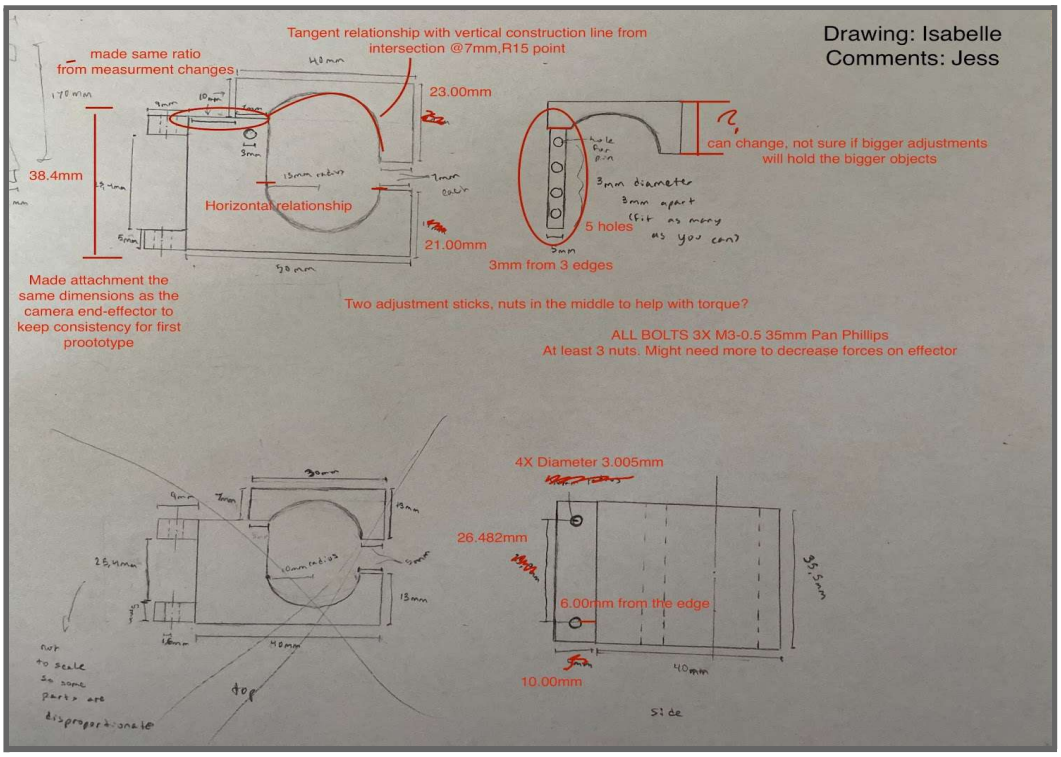


Figure 11. First engineering drawing of corrosion remover and painting end effector with comments during CAD modelling

2.2.2 CAD Model

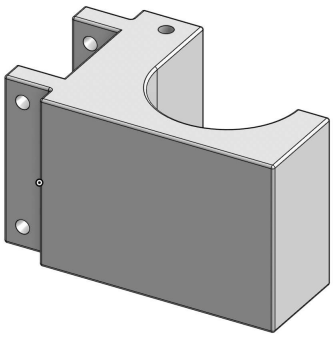


Figure 12. Isometric view of base of water/paint end effector

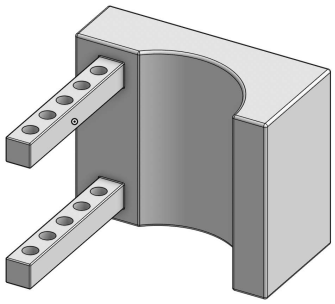


Figure 13. Isometric view of adjustable part of water/paint end effector

2.2.3 CAD Drawings and Measurements

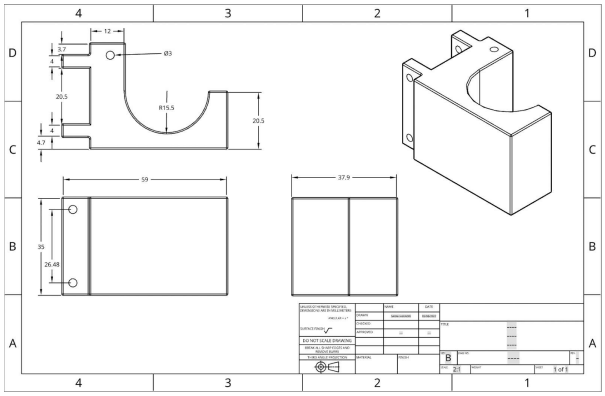


Figure 14. Orthographic projection of base water/paint end effector

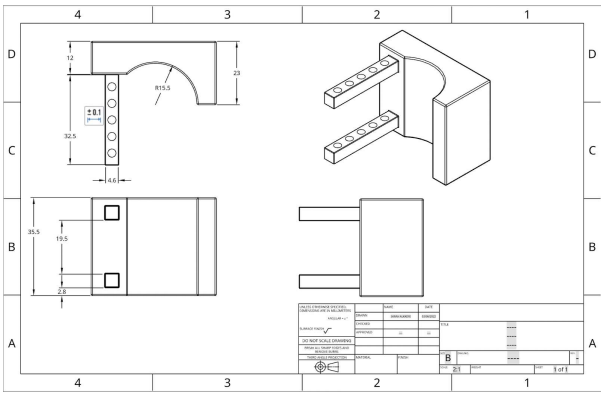
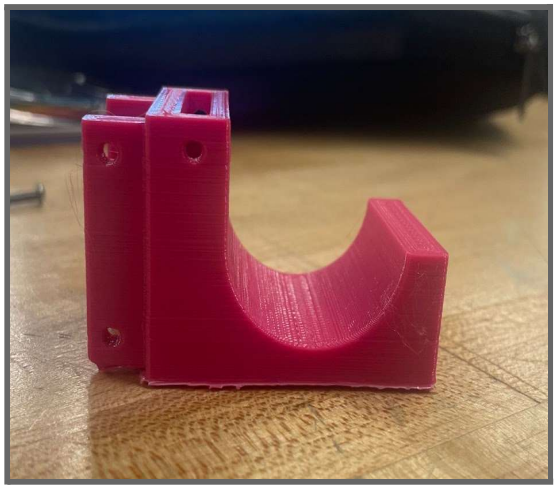
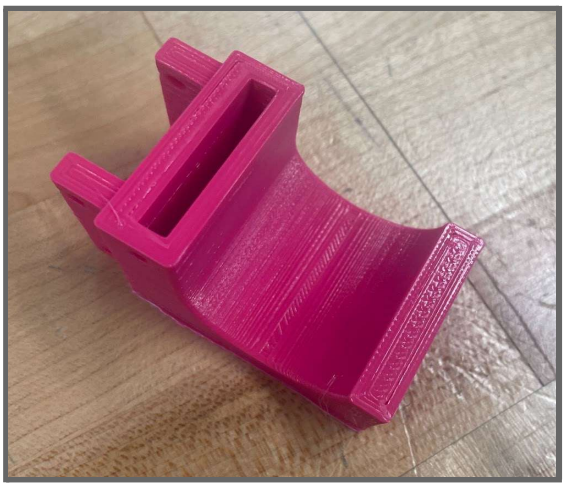


Figure 15. Orthographic projection of adjustable part of water/paint end effector

2.2.4 3D Printed product





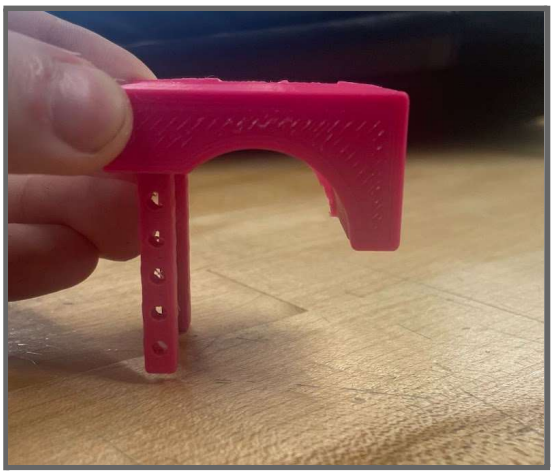


Figure 16. First prototype of adjustable corrosion removing and painting end effector

2.3 Coding and Kinematics

2.3.1 User Interface Prototype

We have been working on creating a simple user interface to facilitate the users experience with this arm and code. The client mentioned that the user will be an individual with highschool level education, therefore they will need an easy to use interface with no coding experience required to complete their task at hand. The GUI is coded in Python and imported into Tkinter, and will not be tied to the other components in this prototype. The goal achieved with the prototype is to complete a base visual interface and the functional code of said interface. Completed do date is the following code in Python as well as the produced interface.

Our expected result of this prototype is to produce a code mainly for a visual representation of the different functions of the robot and will be improved once implementing other functions closer to design day.

```
import tkinter as tk
import tkinter.font as font
window = tk.Tk()
window.configure(bg="white")
window.title("Robot Arm User Interface")
window.geometry("800x600")
frame1 = tk.Frame(master=window, width=800, height=60, bg="white")
frame1.pack()
cameraef = tk.Button(
text="Camera/Sensor",
width=20,
height=5,
bg="blue",
fg="white"
)
cameraef.pack()
frame2 = tk.Frame(master=window, width=800, height=60, bg="white")
frame2.pack()
waterpaint = tk.Button(
text="Water/Paint Sprayer",
width=20,
height=5,
bg="red",
fg="white"
)
waterpaint.pack()
window.mainloop()
```

Figure 17. First fully functional code prototype for general user interface

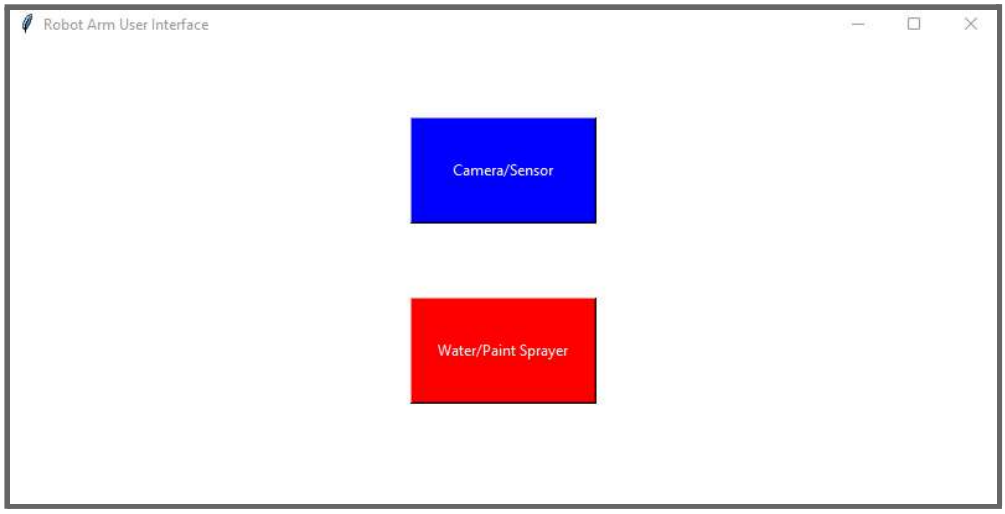


Figure 18. Results of first fully functional user interface prototype (pop-up window with buttons)

The two buttons on the pop-up window achieved by the code are pressable, they just do not have anything happening to them once pressed yet, since that part of the code is proving trickier than others to make function. They have been coloured, scaled and separated using multiple functions to achieve this look, as you can see in the code shown above. We have a few versions of different options for the outcome buttons pressing but they all seem to lead to many errors that are difficult to fix since the code itself is pieced together from examples found on websites and we are not familiar with this programming language whatsoever. Overall, this prototype is the first step towards our final goal with the user interface and the second prototype should have buttons functions which is what we are currently working on.

2.3.2 Kinematics Program Prototype

In order to achieve inverse kinematics, we gathered important information about the robot's mechanical and electrical attributes. The robot that will be used for prototyping is a 3DOF palletizing arm meaning that the end-effector is always parallel to the ground due to geometry. It can be controlled by an Arduino with GRBL CNC shield and by DRV8825 motor drivers.

2.3.2.1 Algebraic and Trigonometric Concept IK

Having the 3 lengths (links), we need to determine to set the servo motors at. With this in mind, when reaching a desired point in space, we can cup up or down to it. For this project we are going to be cupping down to it. (positive angle). Essentially, IK takes a point in space and it goes backward from motor to motor calculating the angles necessary to get the arms to bend to that location.

First off, our known variables are the length of stepper arms, the target locations (acquired from forward kinematics) and the starting location. The goal is to form triangles to then use trigonometric concepts to calculate the angles. To set this up, let`s use simpler values such as a 6 inch link for the base, a 4 inch link for the upper arm and a 4 inch link for the forearm. Let's say that the target coordinate is (10,7), the origin (0,0), where x and y are (x,y). To simplify, I'll use the parameters v for x of origin, w for y of origin, h for x of target and k for y of target. Thus, v=0, w=0, h=10 and k=7 will be our parameters to solve this.Drawing it out in a graph (as shown below), we want to find the slope y=mx+b. To do this, m=(k-w)/(h-y) = (7-0)/(10-0)=7/10. We can then find b by replacing the target coordinates. Our slope is then equal to y=mx+b=7x/10.The concept of math that is largely involved when solving inverse kinematics is the radius of circles. This is what represents the full functionality of 3 degrees of freedom. Essentially each joint could potentially complete a circle of a specific radius, That radius is the length between that joint and the following. In this scenario, the respective radius in the drawing are 6, 4 and 4 (representing the three links). The origin and the target are centers of their respective circles.

Now, we want to find the point j where the line intersects the circle. We used basic trigonometric equations to achieve this. The pythagorean theory was applied during this process. Afterwards, we can put the coefficients of our equation into the quadratic equation to find the x coordinate. Putting that into the slope, we find that the y coordinate. Thus, we now have the coordinates of said point j which is the intersection of the line entering the far right circle in the drawing below.

Moving on, we can start to think about the desired angles and how we will go about achieving this. Let's say that *K* is the length between point j and the origin. We can now find this length using the square root of the x and y coordinates of j to get one step closer to solving the IK problem.The method used to find angles is called the *side side side* calculation which involves the cosine law. Consider length B to be K, C to be the radius of origin joint and A to be the link between the end of C and j. We can determine the 3 desired angles. We can now start to visualize the schematics of the robotic pose to reach the target. To calculate the full angle of A (from the x axis), we can use the target x as the length of one side of the triangle and the target point y as the second length of the triangle. Using pythagorean theorem to calculate the third side length of the triangle, we can get angle *theta*. Adding *theta* to prior A, we get the full angle A. Lastly, we need to determine the new angle C for the last link. We just need to subtract the prior C from 180 degrees. This allows us to have all required angles to reach desired coordinate from origin by cupping up (safer method due to weight of end-effector and stability factors). Here below are the schematics to aid the explanation.

3DOF IK

Known variables: length of stepper arms, target locations, starting location

→ form triangles to calculate required angles

If { Base = 6 in, Upper arm = 4 in, Fore arm = 4 in

origin: (0,0) x=0 y=0

target: (10,7) x=10 y=7

Parameters

V=0 W=0

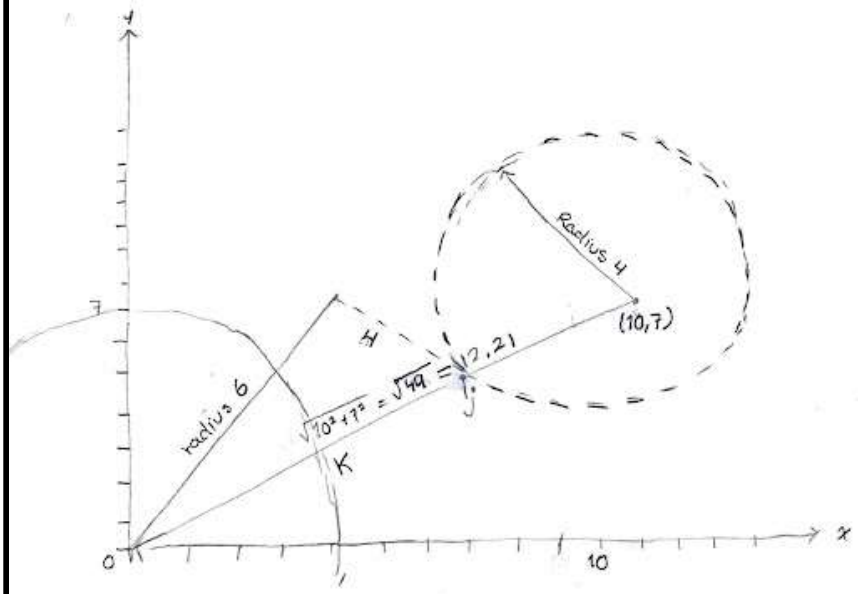
H=0 K=7

Slope

$y = mx+b$

$m = (K-W)/(H-V) = (7-0)/(10-0) = \frac{7}{10}$ $\quad 7 = 7+b \rightarrow b=0$

slope: $y = \frac{7}{10}x$



origin and targ are centers of two circles

→ we find the point (j) where lin intersects circle

$(x-h)^2 + (y-K)^2 = (\text{radius})^2$ radius = 4

$(x-10)^2 + (y-7)^2 = 16$

1: $x^2 - 20x + 100$ $\quad$ 2: $y^2 - 14y + 49$

sub slope for y

$x^2 - 20x + 100 + \frac{49}{100}x^2 - \frac{49}{5}x + 49 = 16$

A = 1.49 B = -29.8 C = 133

(quadratic equation)

$y_1 = 13.13$ $\quad x_2 = 6.7$

→ sub into slope $y = \frac{7}{10}(6.7) = 4.7$

j : (6.7 ; 4.7)

→ we can calculate K

$|K| = \sqrt{x^2+y^2} = 8.18$ in

Figure 19. Inverse kinematics calculations for robot arm

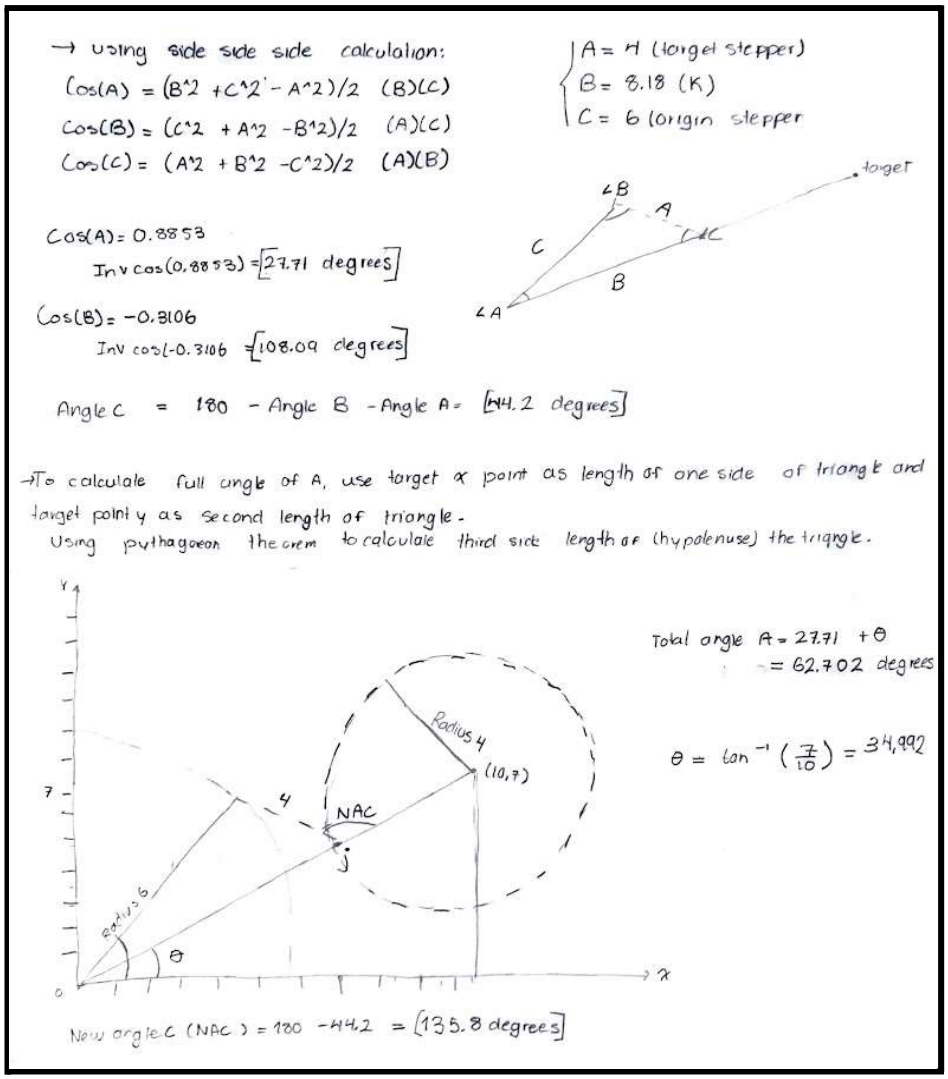


Figure 20. Part 2 of the Inverse Kinematics calculations

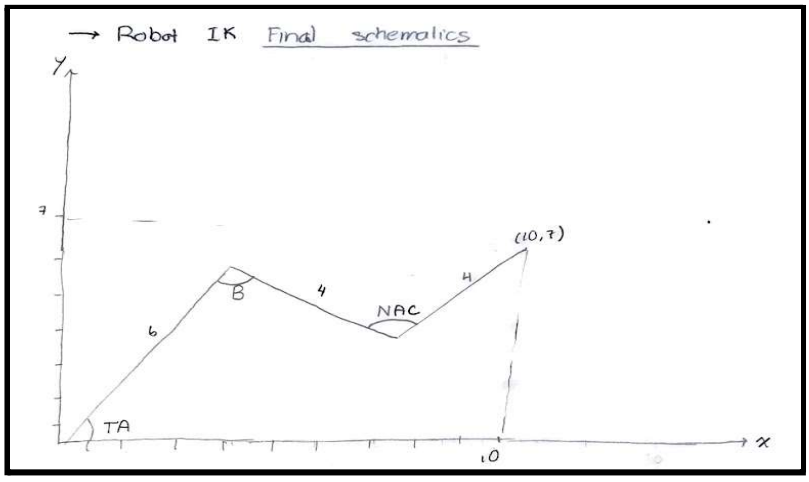


Figure 21. Final section of inverse kinematics calculation (Drawing)

2.3.2.2 Numerical methods and code implementation IK

The robot arm has 3 essential *links*; the *base* that connects **a0** to **a1** with a length of 8.202 cm, the *upper arm* that connects **a1** to **a2** with a length of 24.269 cm and the *forearm* that connects **a2** to **a3** with a length of 25.041 cm. Unlike most robotic arms, this arm does not have a hand since the end effector is inserted directly into **a3**. Each joint corresponds to each servomotor of the arm as described in section

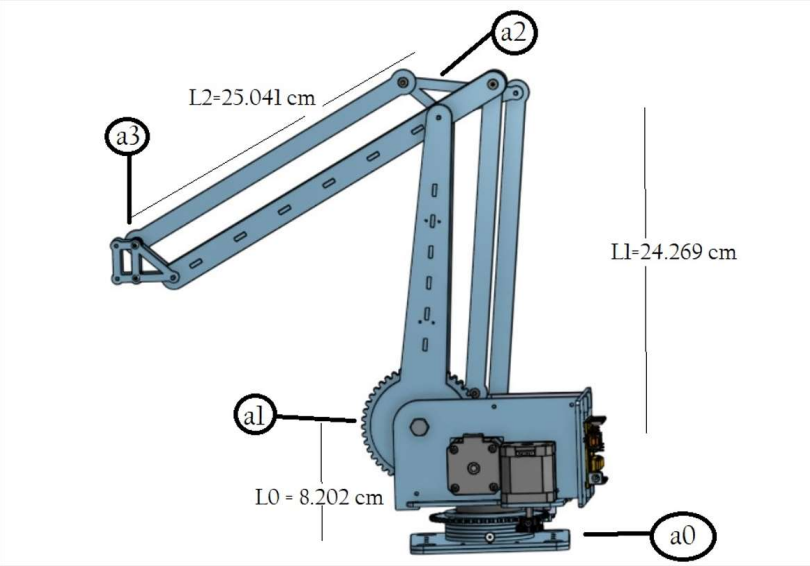


Figure 22. 3DOF robot arm designed at UOttawa with certain dimensions

During the coding process, we have decided to use `float` for all variable types to allow utmost precision (especially with inverse cosines involving a lot of decimals).

As mentioned in section 2.3.2.5, we use the Servo.h library included with function `#include <Servo.h>`. We can then create servo objects for each joint/motor and assign values to important parameters discussed in section 2.3.2.1. The x1 and y2 target values are just for testing purposes. After the ***FK*** program described in section 2.3.2.3, we will add a function to input the coordinates acquired from that software and use them to determine the inverse kinematics sketches.

```
#define PI 3.14159265

float BaseLength = 3.229;
float ShoulderLength = 9.555;
float ElbowLength = 9.859;
float Hypot, Slope, CirclePointX, CirclePointY;
float x1=7;//Target end point
float y1=10;//Target end point
float x2 = 0.0;//origin start point.
float y2 = 0.0;//origin start point.
float A, B, C,
      PosXAnswer,   //positive X portion of Quadratic Equation.
      PosYAnswer,   //positive Y portion of Qudratic Equation.
      NegXAnswer,   //negative X portion of Quadratic Equation.
      NegYAnswer,   //negative Y portion of Quadratic Equation.
      Angle_A,      //Angle located at origin (0,0).
      Angle_A_Temp, //Second 1/2 of Angle located at origin (0,0).Shoulder servo setting.
      Angle_B,      //Elbow servo setting.
      Angle_C;      //Wrist servo setting subtracted from 180.
```

Figure 24. Coordinate input portion of the Inverse Kinematics code

After initializing and/or declaring variables, we use a setup function to set pin modes to `OUTPUT,` we then proceed to attach the servos to the servo objects previously created and then we `ResetServos();`.

```
void setup()
{

  Serial.begin(9600);

  pinMode( 9, OUTPUT); //Base pin.
  pinMode( 10, OUTPUT); //Shoulder pin.
  pinMode( 11, OUTPUT); //Elbow pin.
  pinMode( 8, OUTPUT); //Wrist pin.


  BasePan.attach(9);  // attaches the servo on pin 9 to the servo object
  ShoulderTilt.attach(10);  // attaches the servo on pin 10 to the servo object
  ElbowTilt.attach(11);  // attaches the servo on pin 11 to the servo object. Can not use Pi
  WristTilt.attach(8);// attaches the servo on pin 8 to the servo object

ResetServos();
```

Figure 25. Pin setup section that assigns values to variables

We then begin a loop function to start the IK algorithm determined in section 2.3.2.1. We set search limits based on the maximum reach of the arm which is 22 inches. From there, there are two possible calculation functions : `Special_Calc_point` or `Calc_point`. An if else statement is used to make sure that if the x and y target points are shorter than the shoulder lengths, we need to use `Special_Calc_Point`. These two functions essentially follow the mathematical concepts of section 2.3.2.1.

```
for ( x1 = 3; x1 < 22 ; x1 = x1 + 1) {

for (y1 = 1; y1 < 22 ; y1 = y1 + 1) {


    Serial.println("******************");
    Serial.println();
    Serial.print("Point: (");
    Serial.print(x1);
    Serial.print(",");
    Serial.print(y1);
    Serial.println(")");
    if ((x1 < ShoulderLength) && (y1 < ShoulderLength)) {
      Special_Calc_Point();

    }//end if x1 & y1 < ShoulderLength.
    else {

      Calc_Point();
```

Figure 26. Final section of current code

2.3.2.3 Kinematics program for Prototype II **FK**

The goal for using forward kinematics is to set predetermined angles that will help map out the scan/search area. Using these sets of angles, we want to find the final coordinates that the robot arm will move to in order to scan the area for corrosion.

2.3.2.4 Arduino motor functions

As mentioned, there are 4 servo motors, one for each joint. As displayed in the image below, the 4 wires should connect to the 4 digital pins 8, 9, 10, 11. Here below is an example of how the servo responsible for the wrist movement will be wired to an arduino Uno.

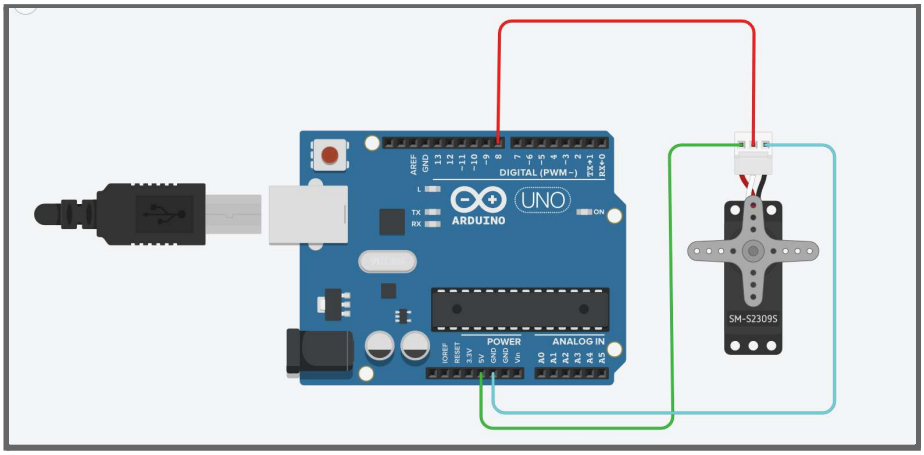


Figure 27. Arduino with code controlling stepper motor: Tinkercad

2.3.2.5 Library (motors)

To achieve motor function through inverse kinematics, we will be using the Servo.h arduino library. It allows us to control servo motors because they are more practical than stepper motors for this project. Hyperlink : https://www.arduino.cc/reference/en/libraries/servo/.

2.3.2.6 Code File (onedrive link)

IK : IK CODE.ino

2.3.3 Corrosion Detection Prototype

Upon conducting more research on the Corrosion AI, we discovered that the program used, FloydHub, has shut down. Without FloydHub, we have reached a significant speed bump in terms of progression towards a working AI for our project. The original blog post posted by the AI creator, Anirban Konar, says it is possible to run the program with your virtual environment using Python, Tensorflow, Keras and Anaconda. Now the task at hand is to read and understand the usage of all the files in the downloaded files pack, https://github.com/anirbankonar123/CorrosionDetector, and understand the files with the end goal of being able to run the corrosion AI. Without the proper software, this will prove to be very difficult. Still, if the program works in the end, it will be very beneficial to the capabilities of our camera end effector.

If the previously mentioned program doesn't work due to FloydHub being inactive, we have found another program that could yield identical/similar results. https://github.com/bzager/corrosion-detection is a python program that works in the same way as the previous code, learning through training images of rust/no rust pictures to detect if a picture does indeed have corrosion or not. Both of these are concepts that we would like to implement in our code, automated or not; they would be beneficial to the effectiveness of our robot arm program. This program seems to have lower accuracy and efficacy than the previous FloydHub AI, but some of the files in that folder seem to be FloydHub files that cannot be opened anymore. We have contacted the original creator of the AI, and he has responded, permitting us to use the code and offering support in making the code work. This seems like a promising alternative to the original AI we had in mind.

2.3.4 Safety Motion Sensors

To ensure the safety of all users of our product, it is absolutely necessary to implement a safety system and some safety features to avoid any serious injuries and being at fault for those unfortunate mishaps. Therefore, we will need to use sensors and more to create an environment that is as safe and secure as possible for these people to use our product. This system is a combination of motion sensors scanning the area around the robot, an automatic stop function, a buzzer sound when the sensor is triggered and an emergency stop button either on the arduino or part of the user interface. The

2.3.4.1 Hardware

Using a PIR sensor, the signal of detected movement is sent back to the arduino uno through a wire. The PIR sensors use a pair of polyelectric sensors to detect heat energy in its surroundings, hence its name *pyroelectric infrared sensor*. A 220 ohm resistor is used witht the LED
The full setup can be found here [Arduino with PIR motion Sensor, LED and buzzer]

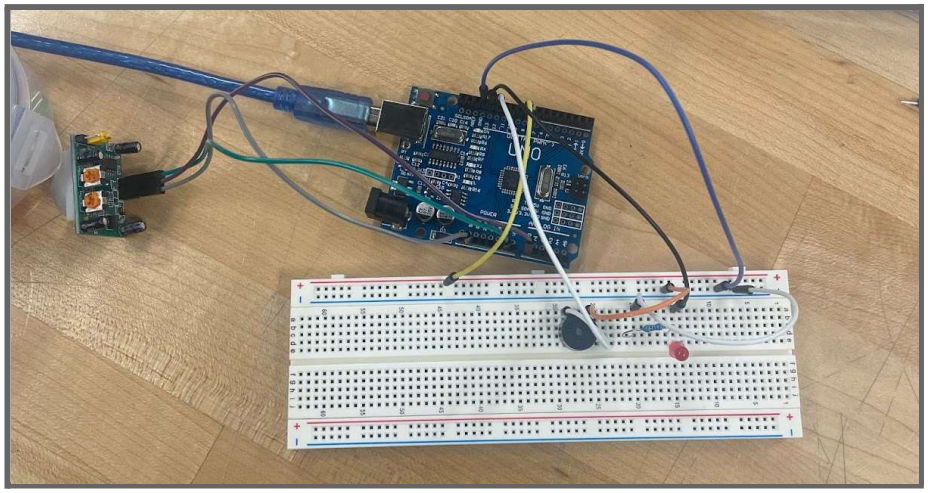


Figure 28. Hardware setup of motion sensors

2.3.4.2 Functionality and Code

The code is a simple void loop with an if else statement. Each constant variable is set corresponding to the pin connections on the arduino.

The code is sent to the arduino and is constantly scanning for surrounding movement while the robot is performing a task. The code and hardware represents the basic functionality with no interrupt functions or delays implemented yet. The plan is to attach two sensors, one on the left and one on the right side for maximum area coverage.

At the beginning of the code, we first initialize the sensor, light and buzzer to their respective digital or analog pins. This allows the code to have access to the light and buzzer if motion is detected.

When motion is detected, this triggers a motion sense value to be greater than 200 due to infrared in the surroundings. This triggers the “if “ statement where the LED light is set to a HIGH output through

a digitalWrite command. The buzz pin is also set at a value of 100. Delays have not been implemented to allow the buzzer to keep buzzing as long as there is IR in front of the sensor.

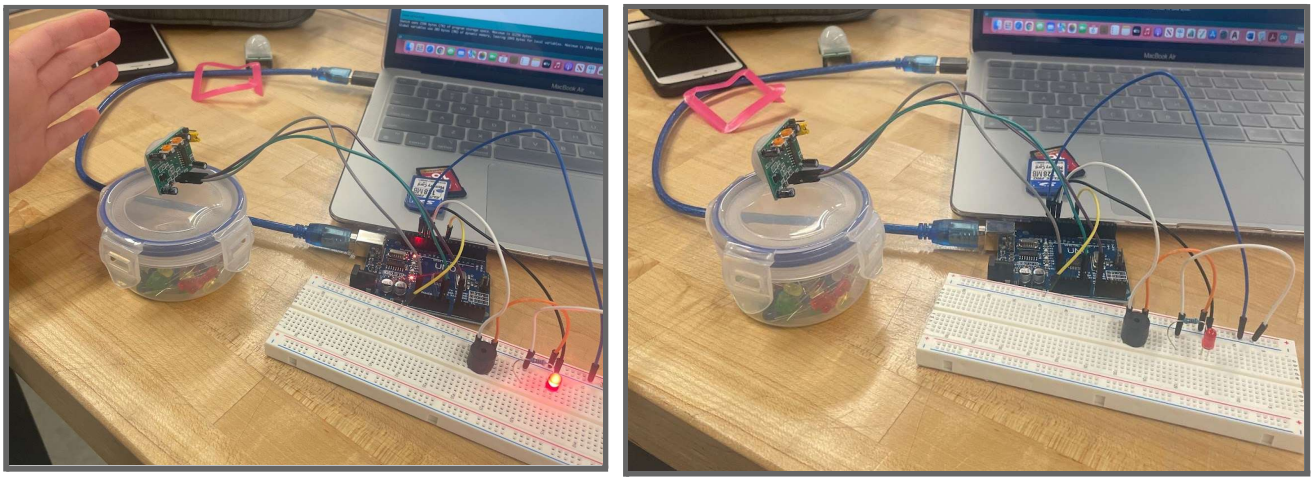


Figure 29. Working PIR motion sensor test with arduino, buzzer, and LED light trigger.

```
const int motionpin=A0;
const int ledpin=13;
const int buzzpin=12; // ledpin,motionpin and buzpin are not changed throughout the process
int motionsensvalue=0;
void setup() {
// put your setup code here, to run once:
Serial.begin(9600);
pinMode(ledpin, OUTPUT);
pinMode(motionpin,INPUT);
pinMode(buzzpin,OUTPUT);
}
void loop() {
// put your main code here, to run repeatedly:
motionsensvalue=analogRead(motionpin); // reads analog data from motion sensor
if (motionsensvalue>=200){
digitalWrite(ledpin,HIGH);
tone(buzzpin,100); //turns on led and buzzer
}
else {
digitalWrite(ledpin,LOW); //turns led off led and buzzer
noTone(buzzpin);
}
}
```

Figure 26. Arduino code for PIR sensor with buzzer noise

3. Prototype Test Plans

3.1 Prototype I Test Plan

Table 1. Prototype I test plan with expected results

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	If all goes well, the corrosion code we have found may be accessible to us and may be able to be translated, and	4 days While drawings and models are	

	code	that translation to a language that we understand would be this step.	being made	
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

3.2 Prototype II Test Plan

Table 2. Prototype II test plan with expected results

Test #	Objective	Description and Test Method	Test Duration and Date	Expected Results
1	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	The robot successfully performs the inputted function
2	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	Production is successful
3	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	1 day After first session with robot	Successfully implement changes for second improved prototype designs.

		accommodate our new discoveries		
4	Second Paper or cardboard prototype	Another comparison with a quick and easy prototype and the arm with the new calculations and retouched dimensions to see if it is correct, if not repeat steps 5 and 6 until the prototype works	1 to 5 days For next session with robot arm	Successfully implement changes for second improved prototype designs.
5	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 of materials used and the overall cost	1 or 2 days Second session with robot arm	Successful printing process according to the measurements of the designs.

4. Updated Bill of Materials

The changes to our bill of materials consists of the change of camera we plan on using. The last camera – the Arducam 2 megapixels MT9D111 – was increasing in price and becoming more difficult to get ahold of the product. While doing research for our first prototype, we came across an issue where the Arducam 2 required an Arducam Parallel Camera Adapter Board for the USB3 camera shield, which we do not have. The new camera we chose has more reliable information and instructions on the internet making our lives easier when learning how to code with it. Although the camera quality capacity is significantly diminished (0.3 Megapixels compared to the Arducam 2 at 2 Megapixels), there is significantly more information available for the new camera, it is smaller in size, cheaper, and does not need additional parts. The weight and the total shipping cost cannot yet be determined.

Table 3. 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	7.89	These sensors will be added to different end-effectors to ensure safety while operation. (soldered)
Adjustable clamp Stainless steel adjustable	1	8.99	This clamp is to provide support to both the water gun and paint gun effectors. It can remain attached to the thor arm for both processes since it is compatible with different sizes.
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.
Total product cost (w/o taxes or shipping)		43.26	
Total product cost (including taxes and shipping)			

5. Target Specifications

Robot Mechanics	
Degrees of Freedom	3
Weight supported by end effector (kg)	1.00
Weight of robot (kg)	9.07
Camera end-effector	
Back and Main piece diameter (mm)	60.00
Back depth (mm)	2.90
Main piece depth (mm)	22.00
Clips length	10.00
Weight (kg)	0.035
Camera	
Weight (kg)	0.016
Camera Resolution	VGA 640 x 480 at 30 fps
Working Power	60mW/15fps
Pixel coverage	3.6um x 3.6um
Painter/corrosion remover end-effector	
Main piece dimensions (mm)	(49.00 x 37.40 x 35) + 2(9.0 x 4.0 x 4.0)
Adjustable piece dimensions (mm)	(44.198 m x 22.0 x 34.5) + 2(32.5 x 4.8 x 4.8)
Weight (kg)	N/A

Conclusion

Having completed our first prototype, we are ready to pitch our project concept/prototype in an organized manner at our client meet on Tuesday March 8th. In the process of creating our various prototypes, we have furthered our understanding on the problem and how our direct solution will come about. We branched when turning our concepts into prototypes to ensure that we could cover as much as the client's needs as possible.

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-robot-use-arduino/

anirbankonar123. (2019, May 7). *ANIRBANKONAR123/corrosiondetector: Corrosion detection from images*. GitHub. Retrieved March 6, 2022, from https://github.com/anirbankonar123/CorrosionDetector

Arduino. (2022). *Servo*. Servo - Arduino Reference. Retrieved March 4, 2022, from https://www.arduino.cc/reference/en/libraries/servo/

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

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

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

Mon, S. (2017, July 11). *Arduino using inverse kinematics (ik) - youtube*. Youtube. Retrieved March 4, 2022, from https://www.youtube.com/watch?v=Y8ueTjqCcAg