Project Deliverable H - Prototype III 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 20th, 2022

Introduction **4**

1. Client Feedback **5**

2. Prototype III **6**

2.1 Camera End-Effector 6

Figure 1. Third 3D printed prototype for camera end effector with posterior and anterior views 7

2.1.1 CAD Model 8

Figure 2. CAD model of prototype III of the camera end effector 8

2.2 Corrosion Remover and Painting End Effector 9

Figure 3. Prototype II of the corrosion and paint end effector 9

2.2.1 CAD Model 9

Figure 4. CAD model of prototype II of the corrosion and paint end effector 9

2.3 Coding and Inverse Kinematics 10

Figure 5. Base, Shoulder and Elbow connections to CNC shield 10

Figure 7. Calculation function 11

Figure 8. Setup function 11

Figure 9. Setup function (part 2) 12

2.4 User Interface 13

Figure 10. Main GUI Page 13

Figure 11. Open Camera File Code Snippet 14

Figure 12. Back Button Code Snippet 14

Figure 13. Camera End-Effector Page 15

Figure 14. Back Button Results 15

Figure 13. Draw & Paint End-Effector Page 16

2.5 Corrosion Detection Code 17

2.6 Safety 18

2.5.1 Emergency Stop 18

Figure 7. Emergency stop function within the user interface 18

2.5.2 Motion Detection Sensors 18

Figure 8. Motion detection sensors functioning 19

3. Prototype Test plans **20**

3.2 Prototype III Test Plan 20

Table 1. Prototype III Test Plan with expected results 20

4. Updated Bill of Materials **22**

Table 2. Bill of materials with total product cost estimate 22

5. Target Specifications **23**

Table 3. Target Specification of all aspects 23

Conclusion **24**

Wrike Link :

https://www.wrike.com/workspace.htm?acc=4975842&wr=20#path=folder&id=824968763&c=list&vid=64521612&a=4975842&t=830081719&so=5&bso=10&sd=0&f=&st=nt-1

Introduction

With only one and a half weeks until design day, this is a very important week as we are doing the final tweaks and fixes to the different sections of the project so that they are as improved as possible for the final pitch presentation to the judges. This week mainly focuses on improvements to certain aspects of the project, physical and software, as well as merging the different parts of the project together so they can all be run at once. Since the previous week, we have been able to finally test our prototypes on the robotic arm since it is now functional.

The first section is the new camera end effector and corrosion remover/paint end effector which has been tweaked to better fit the arm. The inverse kinematics code has been tested on the robotic arm. The GUI user interface has made great progress while the other sections were being completed to be compatible with the interface. The corrosion detection code which has been completed and will be further improved in terms of its accuracy, and finally the safety aspect of the project which is fully functional and awaits implementation onto the actual robot arm itself in the situation that it is used on the Halifax destroyer.

1. Client Feedback

At the previous client meeting, the client did not give our group feedback regarding the development of the end effectors, user interface or inverse kinematics software. However, the client expressed his interest in our corrosion detection software, and he encouraged our solution's continued development.

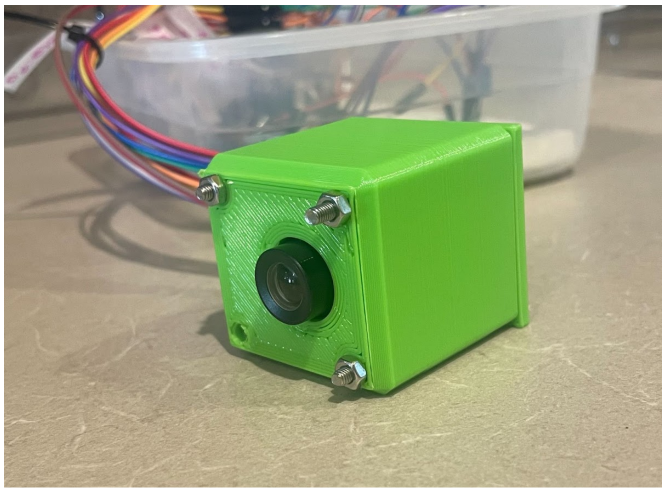
At the next client meeting, our group is planning to further our prototypes by testing our inverse kinematics code and end effectors on the robot arm and the user interface and the corrosion detection software.

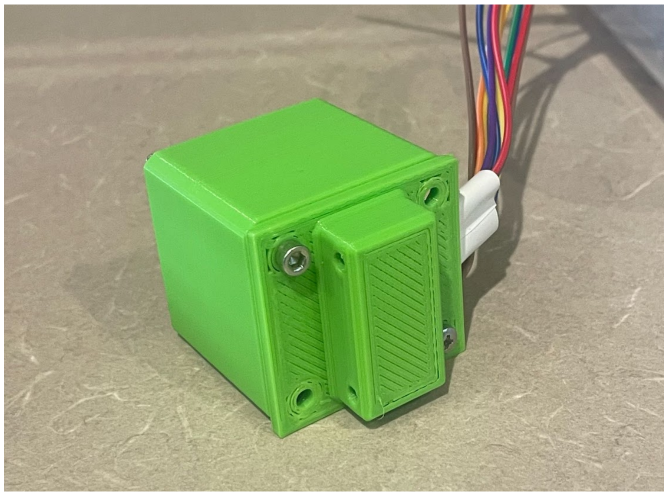
2. Prototype III

As we continue testing, we will have to update and change each subsystem significantly as we get new feedback, better ideas, and encounter technical problems. As our knowledge expands, we are constantly changing our game plan, referring to the target specifications and solving the client's needs most efficiently.

We decided to remove the clips feature and instead create a simple box shape where the camera slides inside. The back piece will still be bolted to the finger using spacers and nuts but will also be bolted directly to the anterior piece of the end effector. This eliminates our concern of the clips malfunctioning, not fastening correctly, or falling off when the arm is moving. Following that, the semi-circle shape is replaced with a box shape, following the camera's shape. The camera lens protrudes out the front and is secured interiorly using four bolts securing each corner of the camera. The bolts extend through the posterior side and are fastened with nuts.

2.1 Camera End-Effector





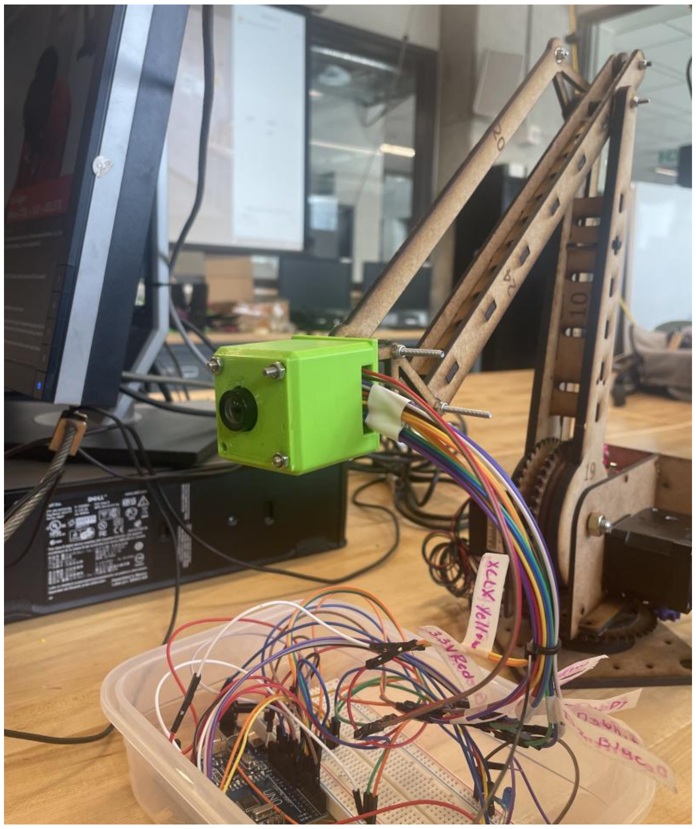


Figure 1. Third 3D printed prototype for camera end effector with posterior and anterior views

2.1.1 CAD Model

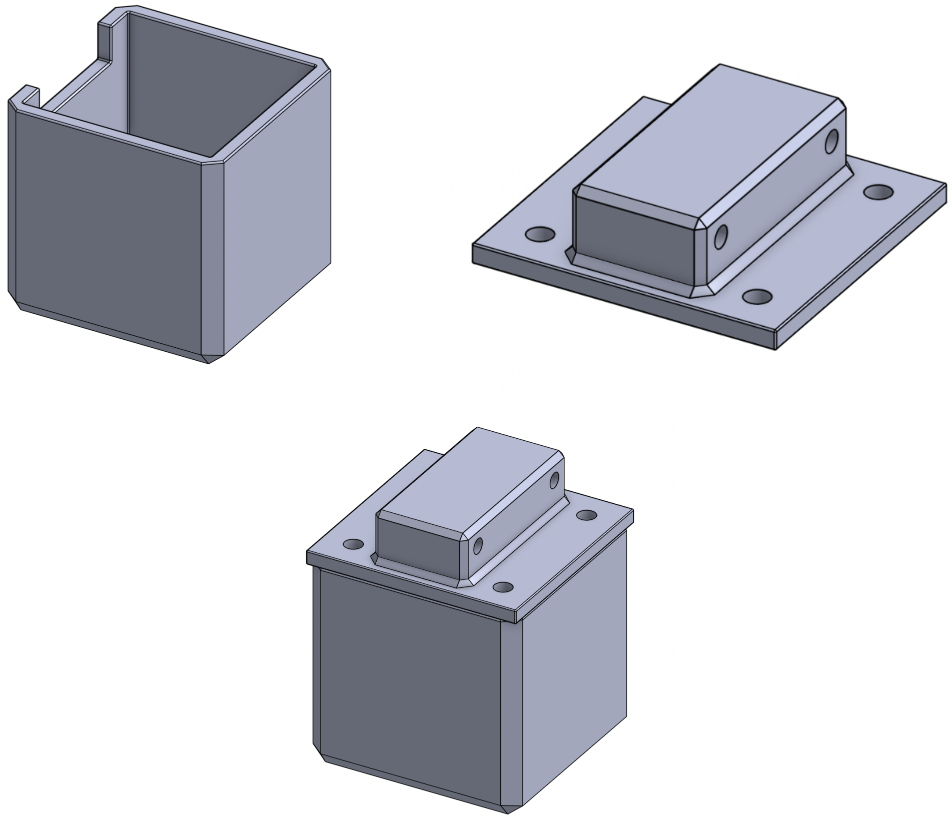


Figure 2. CAD model of prototype III of the camera end effector

2.2 Corrosion Remover and Painting End Effector

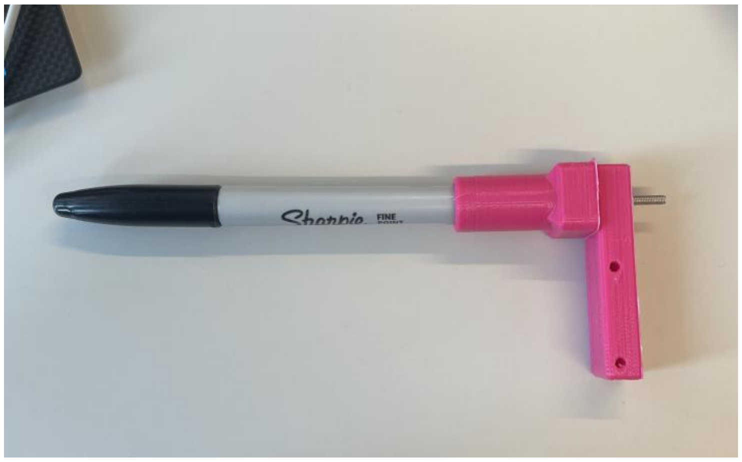


Figure 3. Prototype II of the corrosion and paint end effector

2.2.1 CAD Model

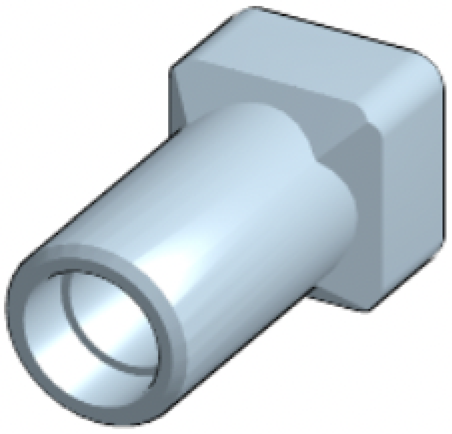


Figure 4. CAD model of prototype II of the corrosion and paint end effector

2.3 Coding and Inverse Kinematics

For this prototype, we have made the code 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.

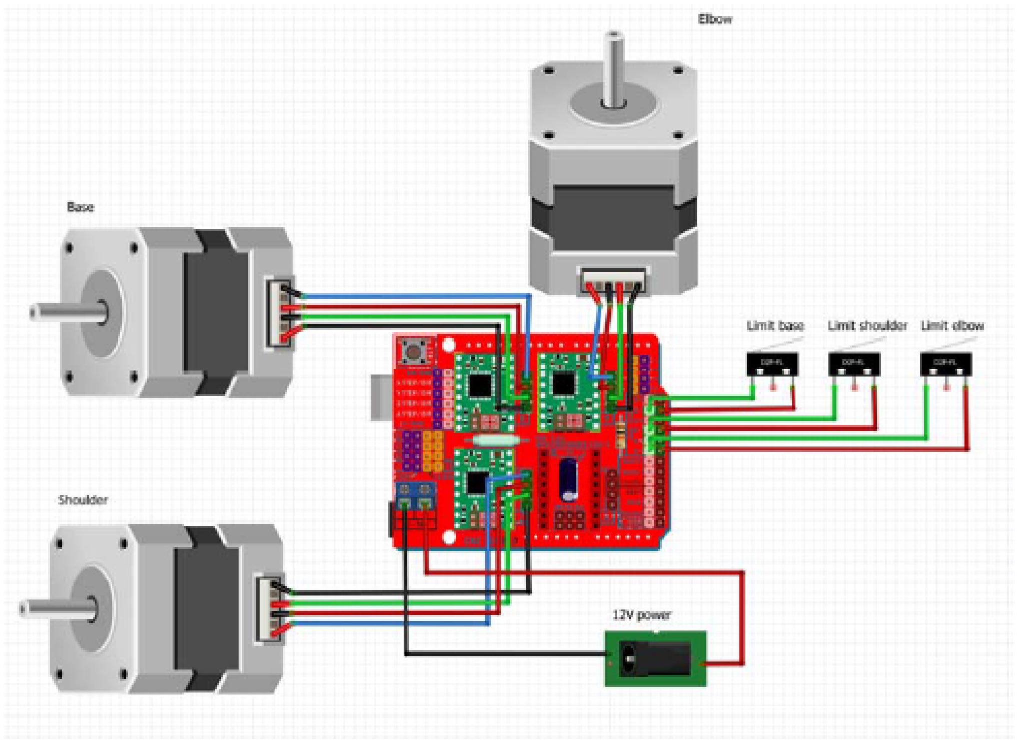
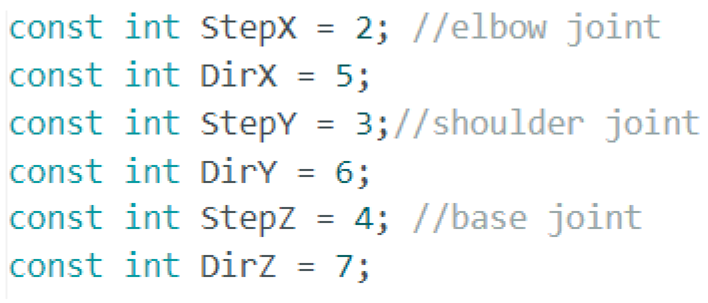


Figure 5. Base, Shoulder and Elbow connections to CNC shield

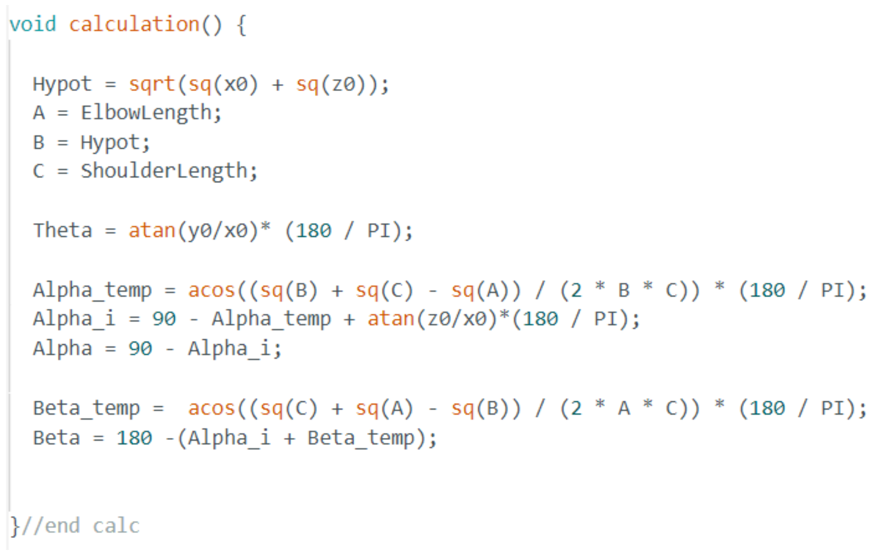
In the code, we start by setting the dir and step pin of each connection to a joint:



```
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;
```

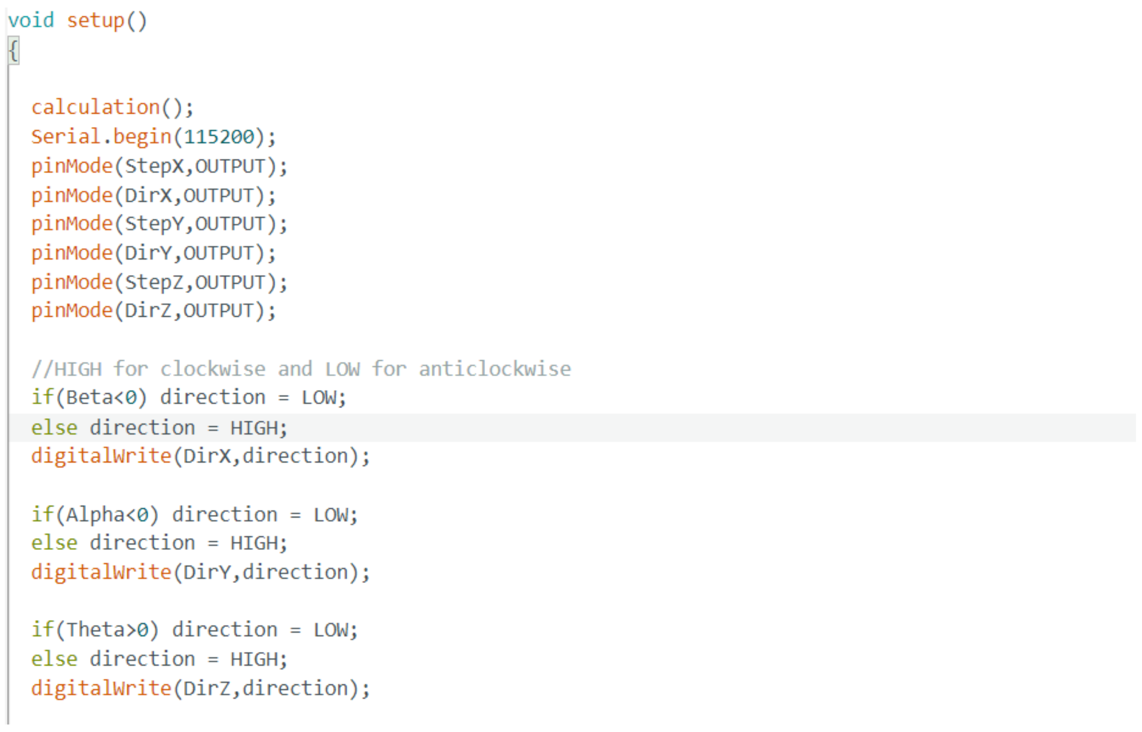
Figure 6. Initialization of Dir and Step pins

In our setup function, we set 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. We also call the calculation function to obtain the angles used for the arm movement. The calculation is the same as that of the second prototype.



```
void calculation() {

  Hypot = sqrt(sq(x0) + sq(z0));
  A = ElbowLength;
  B = Hypot;
  C = ShoulderLength;

  Theta = atan(y0/x0)* (180 / PI);

  Alpha_temp = acos((sq(B) + sq(C) - sq(A)) / (2 * B * C)) * (180 / PI);
  Alpha_i = 90 - Alpha_temp + atan(z0/x0)*(180 / PI);
  Alpha = 90 - Alpha_i;

  Beta_temp =  acos((sq(C) + sq(A) - sq(B)) / (2 * A * C)) * (180 / PI);
  Beta = 180 -(Alpha_i + Beta_temp);


}//end calc
```

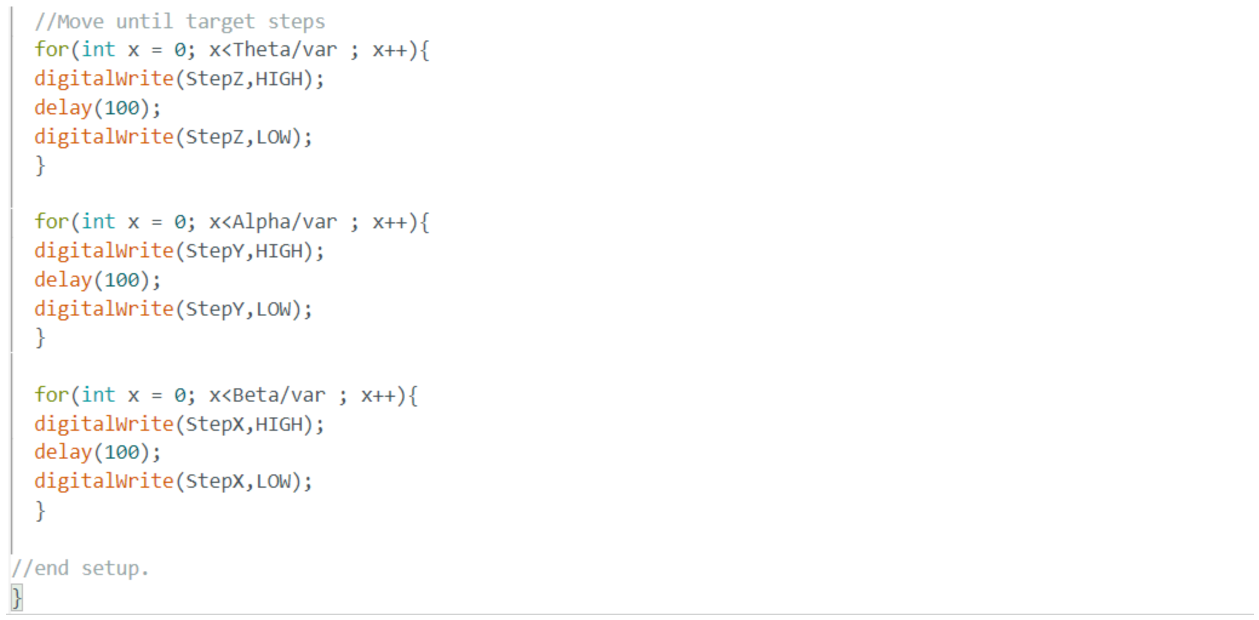
Figure 7. Calculation function



```
void setup()
{

  calculation();
  Serial.begin(115200);
  pinMode(StepX,OUTPUT);
  pinMode(DirX,OUTPUT);
  pinMode(StepY,OUTPUT);
  pinMode(DirY,OUTPUT);
  pinMode(StepZ,OUTPUT);
  pinMode(DirZ,OUTPUT);

  //HIGH for clockwise and LOW for anticlockwise
  if(Beta<0) direction = LOW;
  else direction = HIGH;
  digitalWrite(DirX,direction);

  if(Alpha<0) direction = LOW;
  else direction = HIGH;
  digitalWrite(DirY,direction);

  if(Theta>0) direction = LOW;
  else direction = HIGH;
  digitalWrite(DirZ,direction);
```

Figure 8. Setup function

This is then followed up by for loops that increment the arm steps by one until it reaches the desired amount of steps. We divided the angle value by 1.8 for this declaration since the motors used are 200 step revolution motors.



```
  //Move until target steps
  for(int x = 0; x<Theta/var ; x++){
  digitalWrite(StepZ,HIGH);
  delay(100);
  digitalWrite(StepZ,LOW);
  }

  for(int x = 0; x<Alpha/var ; x++){
  digitalWrite(StepY,HIGH);
  delay(100);
  digitalWrite(StepY,LOW);
  }

  for(int x = 0; x<Beta/var ; x++){
  digitalWrite(StepX,HIGH);
  delay(100);
  digitalWrite(StepX,LOW);
  }

//end setup.
}
```

Figure 9. Setup function (part 2)

2.4 User Interface

After a long break in communication in regards to our GUI to figure out how the rest of the components were to work, we have finally achieved the minimum of our capabilities that satisfies our needs for the presentation and the maximum of what we are capable of in the allotted time. The GUI that we have developed over time is not at its absolute best but it does what is needed and shows all the visuals necessary to succeed. Building off of what we were able to create last prototype, we have added new buttons with new functions that will be shown below and these new buttons will be helping "connect" all of our components into this user interface.

Firstly, we have the good old main user interface page with the distance box as well as the two main buttons (the water/paint is now dubbed draw & paint) that were all explained last deliverable so we will not repeat it but they both clear the current widgets and replace them with the appropriate new buttons which will be explained next.

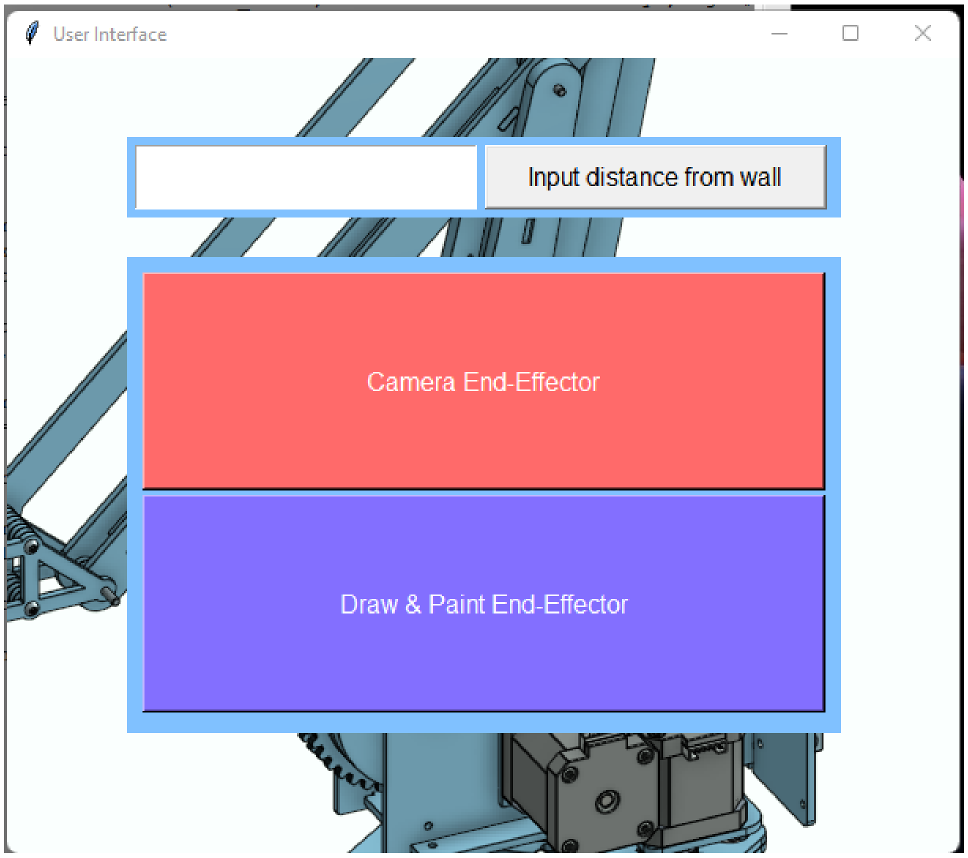
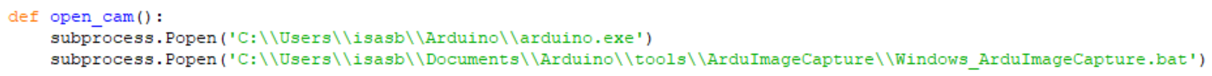


Figure 10. Main GUI Page

Next, we have the new and improved Camera End-Effector interface. This page has gone through the addition of a new button named "Open Camera File" which is not quite what it should be named, but, in simple terms, its function is to open the arduino code file that will then run the live camera feed interface connected to it. The code is still slightly a work in progress, but as it stands now it is along the lines of the following image:

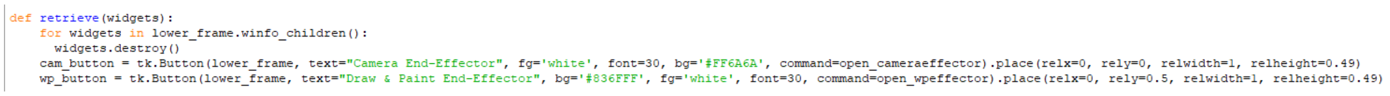


```
def open_cam():
    subprocess.Popen('C:\\Users\\isasb\\Arduino\\arduino.exe')
    subprocess.Popen('C:\\Users\\isasb\\Documents\\Arduino\\tools\\ArduImageCapture\\Windows_ArduImageCapture.bat')
```

Figure 11. Open Camera File Code Snippet

This image shows the small command used to "run" the Arduino Camera Code. This function simply opens whatever applications directory is mentioned and opens it for you to then use yourself. This was the simplest and quickest way we could accomplish what was needed since we did not have very much time to work on the user interface once the other components were functional.

The next new addition to this interface is the "Back" button that now appears on both interfaces as a way to move from one end-effectors interface to the other. This button uses the "widget.destroy()" function to get rid of and place the end effectors and then simply contains the original buttons with their commands and proper placement to return. However, the start and stop buttons do not disappear since halfway through the detecting and painting of an area, you would not want to have to remodify the inverse kinematics distance, and if you do, then sorry, the program can simply be restarted and rerun. This button was a lot simpler than it seemed and a lot simpler than the time that was taken to realize how to 'properly' do it. It took a very long time but the promise of a back button mentioned in the last deliverable is now a reality and works very well. The snippet of code can be found below as well as the image of the new and improved camera page and what happens when the back button is pressed. There is still an empty spot which will maybe have an image show up in it, maybe just have resized buttons but it is just like that for now.



```
def retrieve(widgets):
    for widgets in lower_frame.winfo_children():
      widgets.destroy()
    cam_button = tk.Button(lower_frame, text="Camera End-Effector", fg='white', font=30, bg='#FF6A6A', command=open_cameraeffector).place(relx=0, rely=0, relwidth=1, relheight=0.49)
    wp_button = tk.Button(lower_frame, text="Draw & Paint End-Effector", bg='#836FFF', fg='white', font=30, command=open_wpeffector).place(relx=0, rely=0.5, relwidth=1, relheight=0.49)
```

Figure 12. Back Button Code Snippet

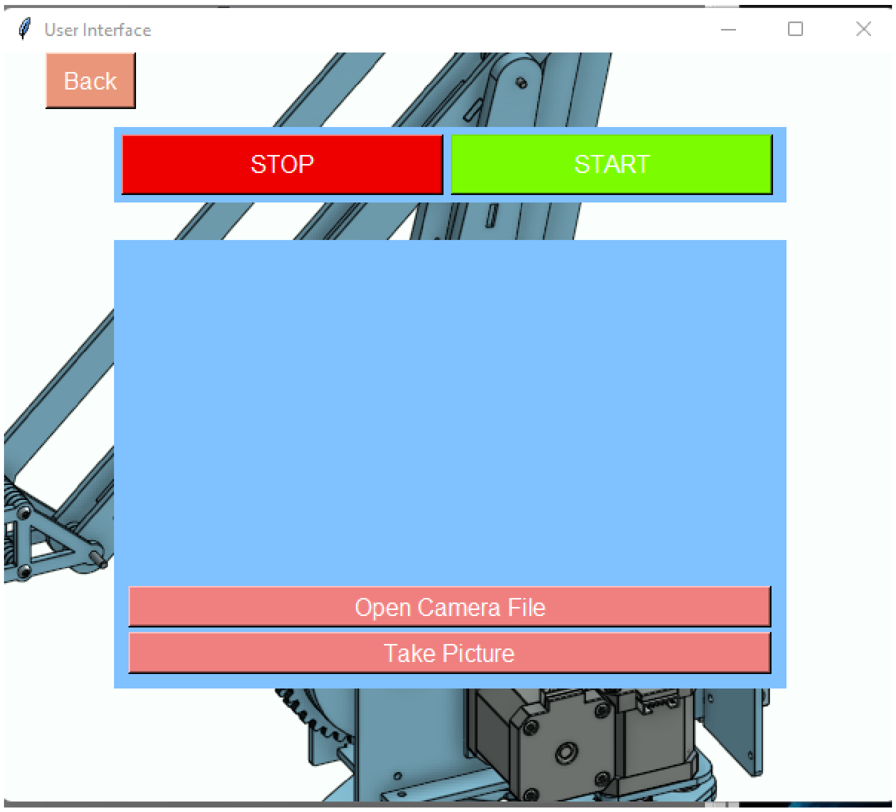


Figure 13. Camera End-Effector Page

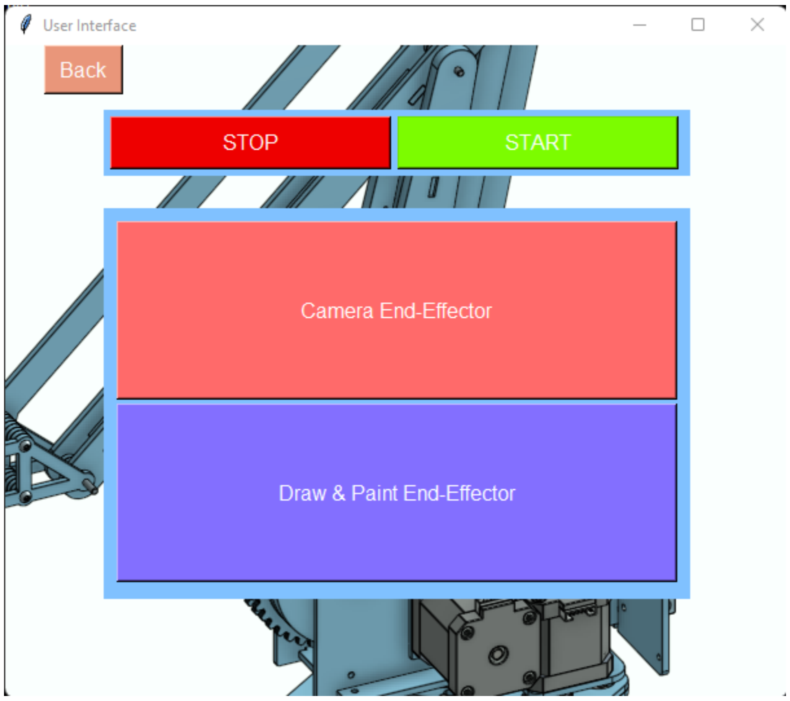


Figure 14. Back Button Results

Finally, we have added a few new buttons to the now Draw & Paint interface page to fill in that empty gap we had last time and give the user options of what they would like to have our robot arm output. These buttons currently do not have any physical outputs as we have not and will not have the time to properly interconnect the inverse kinematics code with the user interfaces code but the idea is that the user would choose one of the three options and then the inverse kinematics code would then follow an if statement that would become true for whichever one was chosen and proceed to upload and run that code on the arm's arduino.

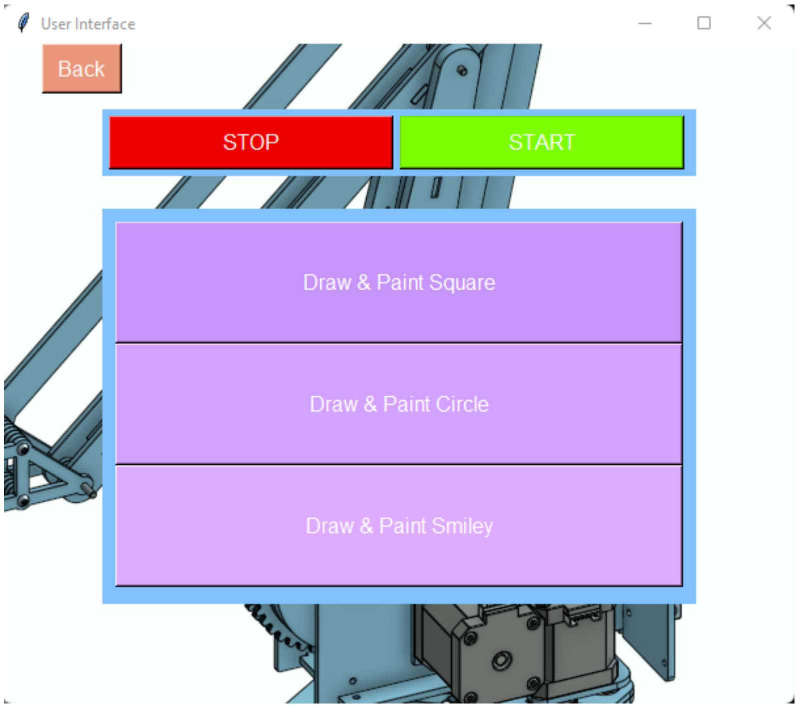


Figure 13. Draw & Paint End-Effector Page

That sums up the changes for this third and final prototype of the user interface. The final product will resemble this interface a lot and most likely will only undergo a few corrections before we present our final product to the judges on Design Day.

2.5 Corrosion Detection Code

Currently, we are at a standstill with the corrosion detection code. There are no longer errors in the code and it fully runs when the string “python main.py test 0.18 0.2” is typed into the command prompt in the proper location. The directory for the code on Benoit’s laptop is the following “C:\Python\Corrosion detection2\corrosion-detection-master\src” using the cd command we can search for a specific file to be selected in the command prompt. We use this several times until we reach the src folder (folder containing the Python files.

Although the code runs completely it still doesn’t output the desired result. Using 5 rust images and 1 blue picture we would expect the results notepad to show the names of just the 5 rust images but the blue picture still shows up in the notepad. Most likely this is an issue with assigning values to the images. The current goal for this deliverable is to be able to run the code and get the correct output. More updates will appear below once we get help from the TA’s and fellow peers.

The issue is believed to lie within this section of the main.py python file.

```
def writeResults(clf):
    res = open("../results/results_"+testDir+".txt","w+")
    for im in clf.test:
        if im.label == 1:
            res.write(im.name+"\n")
    res.close()
```

This function opens the results_test notepad and writes the name of all the files that have a value of 1 (containing corrosion), but we believe that somewhere in the files the program assigns a value of 1 to all of the images.

With the help of Pankaj we found a solution to this problem. The video below shows the code working properly. Thank you Pankaj.
https://youtu.be/g0TKss9Bc-A

2.6 Safety

2.5.1 Emergency Stop

Our team will be implementing an emergency stop function within the user interface that will stop all robot operations in light of a malfunction of the robot. This function will be implemented within the user interface to ease use for those operating the robot. Considering that those operating the robot will have limited technical knowledge, the design of the function will be in the corner of each page with large lettering to be visible to the user at all times.

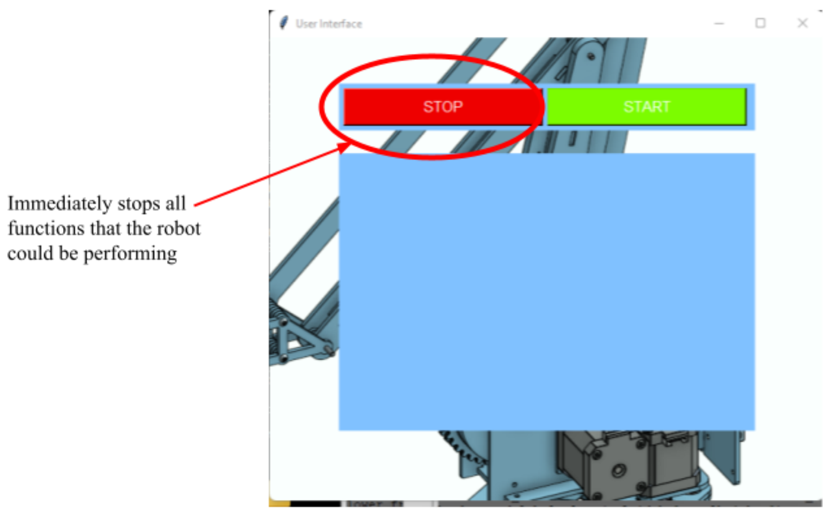

User Interface
STOP
START
Immediately stops all functions that the robot could be performing

Figure 7. Emergency stop function within the user interface

2.5.2 Motion Detection Sensors

The motion detection sensors have been implemented and tested to detect human motion presence within 7 meters of the sensors. This will alert the passerby that the robot functions with a buzzer sound. The red led light on the breadboard also indicates when the sensors have detected motion. Once the sensors are triggered, an interrupt function on the arduino will pause the movement of the arm for a timed duration. Then, assuming the sensors are not re-triggered, the arm will resume the performing function.

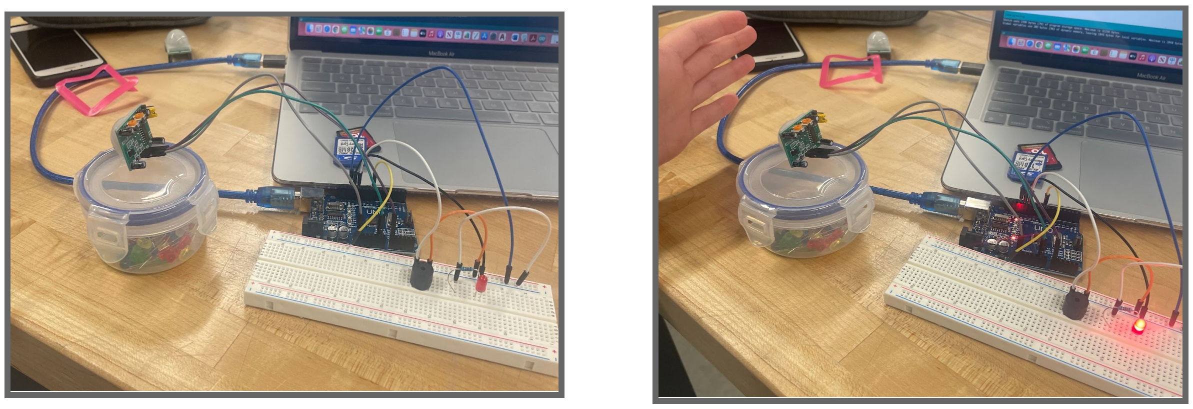


Figure 8. Motion detection sensors functioning

3. Prototype Test plans

3.2 Prototype III Test Plan

Table 1. Prototype III Test Plan with expected results

Test #	Objective	Description and Test Method	Test Duration and Date	Results
1	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.
2	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.
3	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.
4	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.

5	Camera pictures running through Corrosion Program	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 input method now that the camera works and the code runs	The images taken with the camera now successfully run through the corrosion detection software.

4. Updated Bill of Materials

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)		41.16	

5. Target Specifications

Table 3. Target Specification of all aspects

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)	0.052
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	
Weight (kg)	0.028

Conclusion

In summary, we have completed the development of our end effectors, corrosion detection software, inverse kinematics solution and safety components. 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.

Bibliography