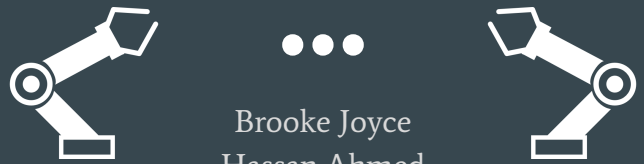
Engineering Design

Robotic Arm



Brooke Joyce
Hassan Ahmed
Christopher Webster
Ancelin Avenido
Jacky Zhu

*Note to TA

We don’t present until Friday and are waiting on some feedback and prototypes until then and that’s why we have some missing information:)

01. **About the Project**
Defining our client's problem

02. **Benchmarking**
Our design criteria and research

03. **Ideate**
Rough ideas and our initial sketches for the final project

04. **Our Prototypes**
Models of our design

05. **Functionality**
Implementing the inverse kinematic calculations in the code

06. **Conclusion**

01. About the Project

Empathize

Who?

- Theodore Eastmond (P. engineer)
- National Defense of Canada
- Royal Canadian Navy

What?

- Design a 3DOF Robotic arm that can scans, paints and water blast
- After the first client meeting: We know what the arm needs
- After the second client meeting: We only have to design the end effector

Where?

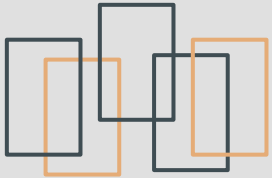
- Halifax Class Ships
- Open area with a chance of water
- Workers may be present nearby

How?

- Use inverse kinematics and code
- Simple UI feature

Define

Our Problem Statement



The robotic arm must hold a reasonably sized pen, camera, and water blaster that can withstand high pressures, all while ensuring it is cost and energy-efficient. The easily accessible robotic arm should safely operate to complete its everyday tasks.

02. Benchmarking

Evaluation Criteria:

- DOF
- The Use of Code
- Overall Volume
- 3D Printable ?
- Safety Feature
- Technical Requirements

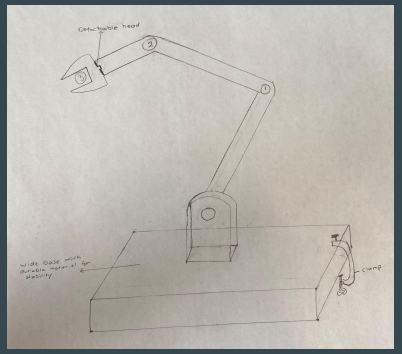
With :

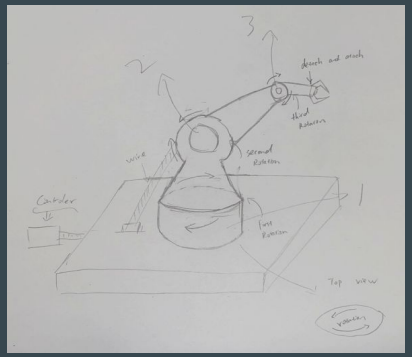
- MicroBot
- Universal UR3
- BCN3D

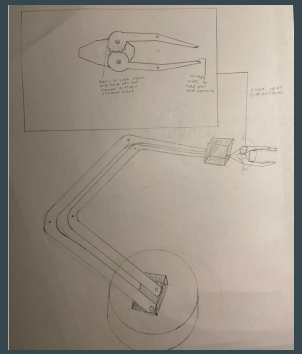
03. Ideate

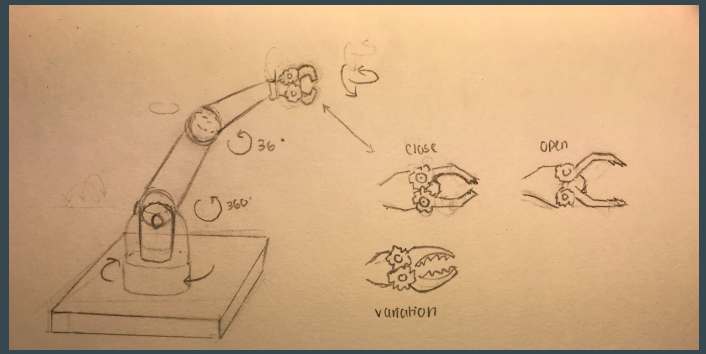
Rough Sketches

- Various ideas for the arm were drawn by each member
- Each sketch was later combined for the best design
- Benchmarking influenced idea for our final design





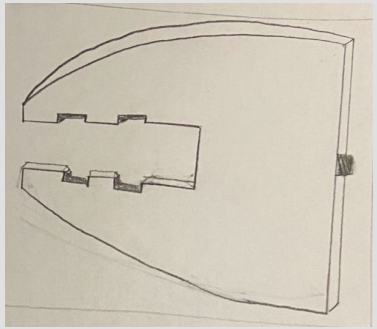




End Effector Design Options

Option 1

- More functional and flexible
- Able to grab different sized cameras



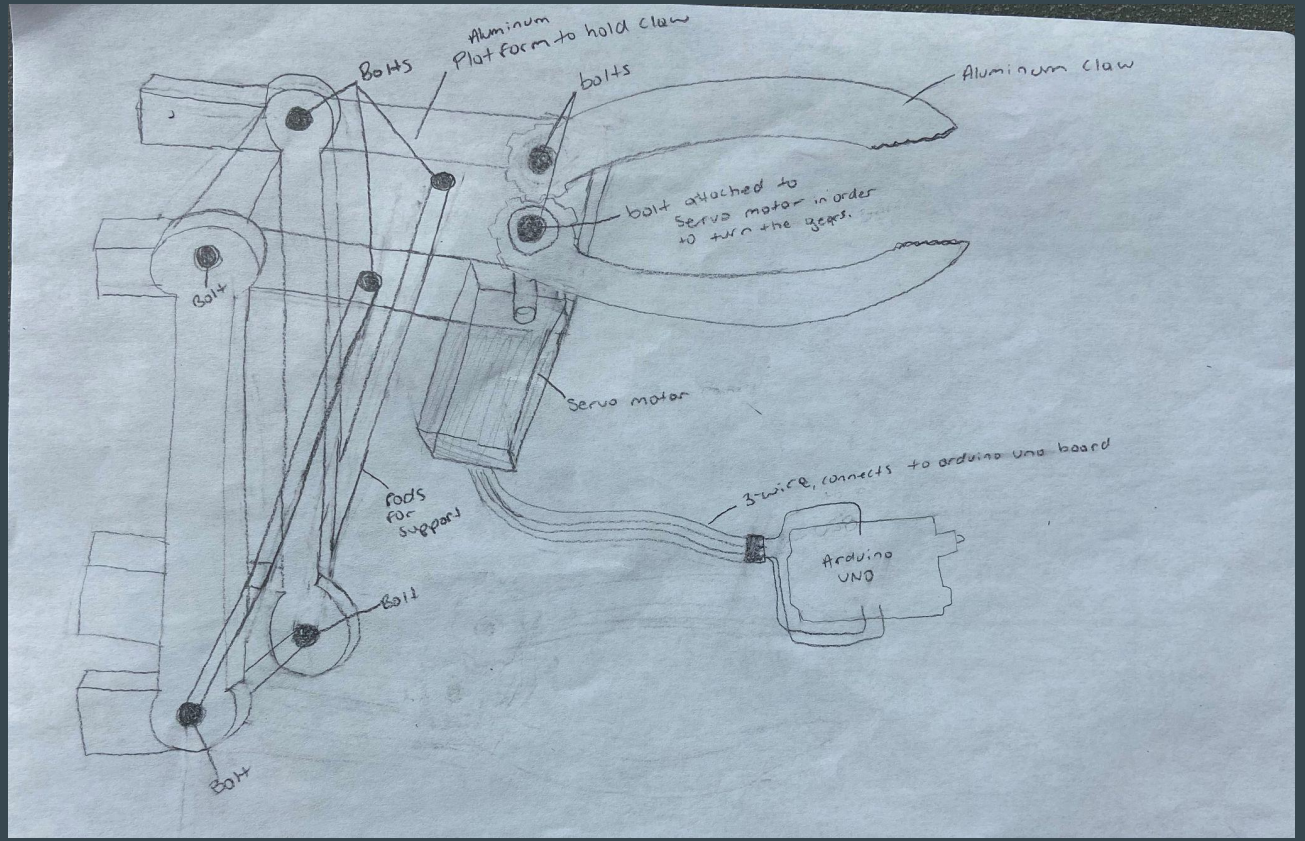
Option 2

- Poor design
- Too bulky to function
- Movement is limited

EZ-Robot Servo Gripper

A benchmarked product our final end effector design is based off of

A Detailed Sketch



04. Our Prototypes

Prototype 1

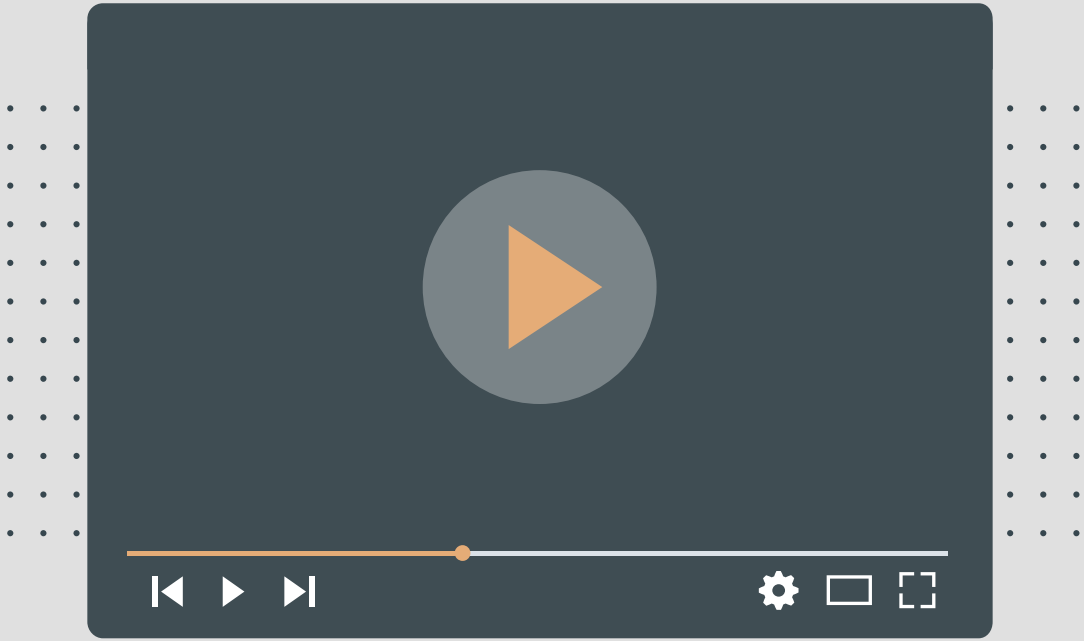
Side View

Front View

Top View

Paperclip acts as the servo motor, enabling the arm to open and close through rotating the servo horn that's attached to the claw.

A short demonstration

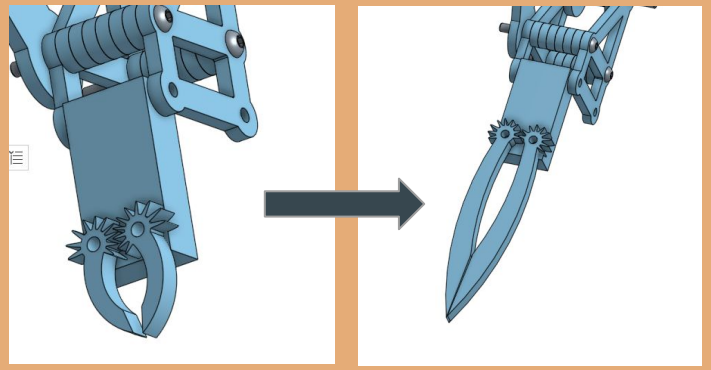


^Insert your multimedia content here

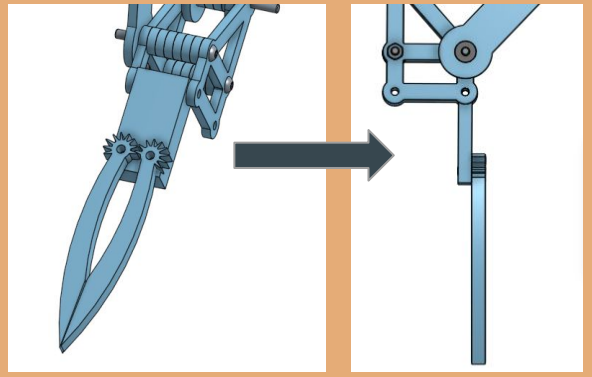
Insert video of first prototype*

Our Prototypes

Prototype 2



- Claw dimensions such as its length, were increased to firmly hold a camera



- Made platform thinner to save materials and costs

Second Prototype - Test

Insert Photo once received

What we tested...

- Functionality to see if claws are big enough
- If it fits
- Correct bolt sizes
- If servo motor fits and works with the gear claws
- If gears can turn
- How much weight it can hold

What we will test next...

- If it can hold a camera
- *Re-test anything that didn't work
- Code
- Inverse Kinematics
- Arduino Keyboard
- Emergency stop

Lessons Learned from both prototypes

First Prototype

- Don’t need the rods
- Functionality works
- Servo motor and horn concept is feasible

Second Prototype

-

05. Functionality

Inverse Kinematics

Insert diagram once confirmed it's correct

Inverse Kinematics

Mathematical approach, using matrices and equations

$$\begin{pmatrix} cos(x) & 0 & sin(x) & 0 \\ sin(x) & 0 & -cos(x) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} cos(y) & -sin(y) & 0 & Jcos(y) \\ sin(y) & cos(y) & 0 & Jsin(y) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} cos(z) & -sin(z) & 0 & Lcos(z) \\ sin(z) & cos(z) & 0 & Lsin(z) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Joint	Z		X	
	Alpha	d	theta	a
1	90	0	x	0
2	0	0	y	J
3	0	0	z	L

Inverse Kinematics

Multiplying the 3 matrices gives the x, y, and z coordinates of...

$$\begin{pmatrix} l\cos(x)\cos(y)\cos(z) + j\cos(x)\cos(y) - l\cos(x)\sin(y)\sin(z) \\ l\sin(x)\cos(y)\cos(z) + j\sin(x)\cos(y) - l\sin(x)\sin(y)\sin(z) \\ l\sin(y)\cos(z) + l\cos(y)\sin(z) + j\sin(y) \end{pmatrix}$$

Use these equations to solve for theta 1, 2, and 3...

Solve for theta 1, 2, and 3 using a mathematical approach through substitution, trig identities, and isolating

Theta 1 = (arcsin)*((z-J*cos(y))/(L*cos(z)+J))

Theta 2 = (arcsin)*(z/(L*cos(z))

Theta 3 = (arcsin)*((x-J*cos(x)*cos(y))/(-L*cos(x)*sin(x))

CHANGE IF INCORRECT***

Our Approach to Inverse Kinematics

PROS

- Free of cost ($)
- Prior knowledge required is gained from labs
- Solves whole system at once

CONS

- Chance of human error through mathematical mistakes
- Chances of drawing the diagram wrong

Code - C++

```ino
#define NUM_MOTORS 3

struct Motor {
  int pin;
  int angle;
  int limit_switch;
};

Motor motors[NUM_MOTORS] = {
  // TODO: fill in pin assignments
  Motor {0, 0, 0},
  Motor {0, 0, 0},
  Motor {0, 0, 0},
};

// the setup function runs once when you press reset or
void setup() {
  Serial.begin(9600);
  home();
}

// the loop function runs over and over again forever
void loop() {
  read_command();
}

int read_int() {
  char c;
  int i = 0;
  while ((c = Serial.read()) != ' ') {
    if ('0' <= c && c <= '9') {
      i = i*10 + (int)(c - '0');
    }
  }

  return i;
}
```

```ino
  return i;
}

void read_command() {
  static char buf[50];

  int x = read_int();
  int y = read_int();
  int z = read_int();

  sprintf(buf, "\nMoving to x: %d, y: %d, z: %d\n", x, y, z);
  Serial.print(buf);

  move_to(x, y, z);
}

void step_motor(int motor, int angle) {
  // TODO: step motor
  motors[motor].angle += angle;
}

void home() {
  // drive motors backwards until limit switches are pressed
  for (int motor = 0; motor < NUM_MOTORS; motor++) {
    while (!digitalRead(motors[motor].limit_switch)) {
      step_motor(motor, -1);
    }
    motors[motor].angle = 0;
  }
}

void move_to(int x, int y, int z) {
  // Get current position
  int cx, cy, cz;
  forward_kinematics(&cx, &cy, &cz);

  const int NUM_STEPS = 255;
  for (int i = 0; i < NUM_STEPS; i++) {
    float t = (float)i / (float)NUM_STEPS;
```

```ino
}

void move_to(int x, int y, int z) {
  // Get current position
  int cx, cy, cz;
  forward_kinematics(&cx, &cy, &cz);

  const int NUM_STEPS = 255;
  for (int i = 0; i < NUM_STEPS; i++) {
    float t = (float)i / (float)NUM_STEPS;
    int angles[NUM_MOTORS];
    inverse_kinematics(
      lerp(cx, x, t),
      lerp(cy, y, t),
      lerp(cz, z, t),
      angles
    );

    for (int motor = 0; motor < NUM_MOTORS; motor++) {
      int dist = angles[motor] - motors[motor].angle;
      step_motor(motor, dist);
    }
  }
}

void forward_kinematics(int* x, int* y, int* z) {
  // TODO: calculate position from current motor angles
}

void inverse_kinematics(int x, int y, int z, int* angles) {
  // TODO: get motor angles
}

int lerp(int a, int b, int t) {
  return a + (b - a)*t;
}
... (3 lines left)
```

Source:

What did we learn?

- How to take advantage of our diverse skill sets.
- Communication is very important and weekly meetings are a great idea.
- Having a team manager that coordinates meetings, organizes tasks, and ensures everything is getting done makes things go by a lot faster and smoother.

Biggest lesson learned?

- Learned that even though having a team manager is great, it impacts the teams overall understanding of the project since each person gets assigned their own task and only the team manager gets to see final results and how everything works together. This restricts creativity and the flow of ideas.

Future work?

- Test our code and inverse kinematics by connecting it to our end-effector and seeing if it moves to the correct spots.

Thanks for listening!

Do you have any questions?