Widener University

School of Engineering

Touchless Mechatronics Laboratory
April 25, 2022

Senior Project Team #19

Team Members
Brennon Conner, Robotics Engineering, Team Leader
Jack Merhar, Robotics Engineering
Shane Mulcahy, Robotics Engineering
Keanu Williams, Robotics Engineering
Emily Wolfe, Robotics Engineering

Faculty Advisor
Prof. Xiaomeng Shi

Senior Projects Coordinator
Prof. Vicki Brown

Disclaimer

This report was generated by Senior Project Team #19, academic year 2021-2022, a group of engineering students at Widener University. It is primarily a record of an educational project conducted by these students as a part of the curriculum requirements for a Bachelor of Science degree in engineering. Widener University makes no representation that the material contained in this report is error free or complete in all respects. Furthermore, the University, its employees and students make no recommendation for the use of said material and take no responsibility for such usage. Thus, persons or organizations that choose to use said material do so at their own risk.

Table of Contents

List of Tables

List of Figures

Executive Summary

The goal of this project was to design and program a hands-free mechatronics laboratory system which is controlled by the user's hand motion and able to complete lab activities. For this academic year, the team's goal was to design and develop software to control the robotic arm system as well as create a gripper that can interface with the current arm. The goal and overall measure of success for the system was to be able to complete lab activities such as the grasping of various mechatronics components, item picking and placing, and soldering.

The project began in late August 2021 and lasted through mid-April 2022. Throughout this time, the first three months were dedicated to initial planning, safety testing, and research. The middle months were planned to be used for development and the final three months were dedicated to additional development, testing, and analysis of the system. The total cost of this project did not exceed $357.48, which included a 20% contingency.

The project required extensive programming, networking, simulation, drafting, and fabrication in order to achieve the original project goals. Overall, the project was mostly successful in completing the planned lab activities, including basic movement based on the user's hand motion, gripper control, component picking and placing, and two-hand component exchange.

1 Introduction

1.1 Background

The initial inspiration for the touchless mechatronics laboratory came about during the remote learning period brought upon by COVID-19 and posed the question, how can students have access to the advanced machinery present at university institutions without putting themselves at risk of contracting the virus? The main purpose of the project is its touchless nature, theoretically minimizing any surface-based spread of the virus.

The theorized answer to this question involved the integration of a telerobotic system with laboratory equipment. These types of robots are typically semi-autonomous and are tethered via wireless communication with a controller. The original plan for the project was to have two high degree of freedom robotic arms be controlled through the user's hand movement using a motion tracking camera, doing so touch-free. The arms would be able to complete sophisticated tasks including, but not limited to, inserting wires and small pins into breadboards, soldering, and grasping items from different surfaces. These arms will be available to students during in-person mechatronics laboratory sessions.

In addition, soldering is typically used to join electrical components with a semi-permanent connection, but it is a safety hazard. Soldering relies on a high-temperature iron and a low melting point metal, most commonly tin, to develop these electrical connections. Even with training, soldering is extremely dangerous and can easily result in burns from the iron or molten tin. The touchless mechatronics laboratory would allow students to solder their own projects while greatly reducing the risk of contact with hazardous materials.

1.2 Literature Review

As our project was inspired by minimizing surface-based spread of viruses, such as COVID-19, we decided to take the time to research how surface-based spread occurs, how long viruses can stay on surfaces, and how impactful surface-based spread is. In a study conducted by Van Doremalen et al. (2020), the lingering times of different surface materials were found, summarized in Table 1.

Table 1 SARS-CoV Surface Lingering

Surface Type	SARS-CoV-1 Time Present (Hours)	SARS-CoV-2 Time Present (Hours)
Plastic	72	72
Steel	48	48
Copper	8	4
Cardboard	8	24

As time on each surface increased, the concentration of the virus decreased exponentially until it was no longer perceivable.

We had also found a few articles which contradicted the original purpose of the project. One study, conducted by Zedtwitz-Liebenstein (2022), swabbed commonly used surfaces in public areas, such as handrails, buttons, doorknobs, and numerous restroom surfaces. After

1,200 swabs over the rough span of three months, zero tests came back positive for SARS-CoV-2. Another study, conducted by Hennessey et al. (2022), followed a similar pattern and yielded similar results. Different public surfaces were swabbed including, traffic light buttons, public playgrounds, and various restroom surfaces. After 210 swabs, three tests came back positive for SARS-CoV-2 viral RNA, of which, zero contained a live version of the virus. These studies did follow real life conditions; however, the age of these samples taken was unknown, and the results relied on guessing the correct surfaces at the right time. The results between these two studies appear to only differ due to randomness when sampling. Additionally, these studies differ from a lab environment where certain areas on different surfaces would receive a high amount of traffic in a short period of time.

Additionally, we were interested in researching similar, existing projects and examining their approaches. One study, conducted by Vysocký et al. (2020), deeply analyzed the Leap Motion infrared hand tracking camera, one of the options for the sensor used in our project. They went in depth on optimal conditions and practices when working with the sensor as well as testing the stability and precision of the camera.

A study conducted by Chen et al. (2017), cataloged the use of a Leap Motion to control a SCARA robotic arm, and the use of gesture control to allow the robot to perform different actions. The paper outlined the data collected, as well as some of the methodologies used to connect and utilize the Leap Motion.

Another study, by Jang et al. (2019), suggested using a virtual reality headset along with a Leap Motion. This allowed the user to have different viewpoints as well as to see their movements and actions in a virtual 3D space. The paper also outlined the ROS algorithms used for the project, along with the hand gesture control that was built into the algorithms. These algorithms and gesture controls were used as a reference for our own control systems.

The use of collision avoidance was also of interest to the group for the creation of motion tracking algorithms. Park et al. (2020) studied the use of trajectory planning and inverse kinematics to prevent robot arms from colliding with each other when in parallel use.

Finally, the group was also concerned with setting safety limits in the robot's workspace to prevent accidental injuries or damage to the environment. A paper written by Zacharaki et al. (2020), detailed the use of priority-focused algorithms and hardware safety features for human-robot interactions (HRIs). These features were studied and noted by the group as references for safety protocols.

2 Design Approach

2.1 Design Criteria

As previously discussed, the overall purpose of the project is to develop a hands-free mechatronics laboratory to reduce the spread of the virus. The success of the project is determined by the robot's ability to correctly interpret and mimic the movement of the user's arms and hands and the robot's ability to successfully pick and place objects of various shapes and sizes, such as stranded 22-gauge wire. Both tasks must be accomplished while ensuring the safety of both the operator and robot.

A. The first major goal of the project, concerning the interpretation and mimicking of user input, requires a motion control algorithm to interface between the robotic arm and hand tracking camera and is responsible for ensuring the accurate recreation of the user's hand and arm movement. Two motion control algorithm types were proposed at the start of the project to accomplish this task: full hand tracking and joystick control. Full hand tracking consists of the motion control algorithm calculating the absolute position, orientation, and angle of the user's hands and arms. Joystick control is a gesture-based system, utilizing the position and velocity of the user's hands and arms to determine the motion of the robot. For example, if the user raises their hand in the positive Y direction, the robot's end effector would do the same. These control algorithms will be discussed in further detail in the selected and alternative design control algorithms sections (2.2.4 and 2.3.4, respectively).
B. The second major goal of the project, concerning object picking and placing, requires research and 3D modeling to develop an end effector for the robot. As no preliminary design ideas were proposed at the start of the project, research is necessary to determine the optimal design to meet the goals of the project while following each constraint.
C. As for the technical constraints, we must consider safety protocols, the speed of our calculations, and the versatility of the gripper. The UR3e arms come equipped with force and speed limits, collision detection, and safety planes to allow for safe operation. The force and speed limits will ensure the robot will not exceed the preset thresholds, regardless of the user's hand movement or any errors in the calculation process. As well, the arms can detect any build-up in torque within its motors to determine if it has possibly collided with an object. Safety planes can be created, which when crossed, triggers either a reduced speed setting or stops the arm entirely. With these features combined, the arm will be able to move without fear of damaging the user or itself. The calculations used to move the arm according to sensor data must be performed and executed with as little delay as possible. This is required to ensure that the user is not distracted by any lagging movement, as this could be both difficult to work with and could potentially be a safety hazard. Finally, the gripper will need to be designed such that it can interact with every item specified through the scope. While it may be easier to simply design the gripper to be an interchangeable component, this would go against the touchless aspect of the project and is therefore not an option.
D. Engineering standards and safety rules that must be followed are discussed in section 2.4.
E. Other relative considerations in the design are discussed in section 2.5.

To effectively meet the proposed goals, the senior project group was split into two groups consisting of two students each, focused on each of the motion control algorithms. Both algorithms were developed simultaneously and analyzed later into the project timeline to determine the optimal control algorithm for the task. The final student was assigned to work on the gripper aspect of the project. The students met primarily online, once per week, for the beginning phases of the project, and transitioned to meeting in person once per week with longer hours towards the final month and a half of the project.

2.2 Selected Designs

2.2.1 Robotic Arms

The robot arm that was chosen for the project is the Universal Robots UR3e Collaborative Robot, shown Figure 1. The UR3e has 6 rotational joints, allowing for 6 degrees of freedom. It is a collaborative robot, or "cobot", because the robot's design includes safety features that allow it to work alongside humans without risk of harm to users or its environment. The robot will likely be operating near people during operation, and it may be handling potentially dangerous tools (e.g. soldering irons), so these safety features will be implemented to prevent any possible injuries during use. The UR3e was also chosen due to its high levels of versatility and accuracy. It has a modular end effector, which allows users to easily change the end effector to a variety of other tools that can be used to carry out tasks in the laboratory. The UR3e is also both consistent and accurate, with a 0.01-millimeter resolution and a pose repeatability of 0.03mm.

Figure 1: UR3e Robotic Arm

2.2.2 Hand Tracking

The Leap Motion camera was chosen for the hand tracking in this project. This decision was made after researching and testing machine vision alternatives. The Leap Motion camera utilizes two infrared cameras in each module, which are backlit by three infrared LEDs, seen in Figure 2. The backlight is used to achieve consistent results regardless of ambient lighting conditions. The infrared backlighting is invisible to humans, so it will not impede users while the

camera is in use. In addition, the Leap Motion module can easily be connected to a computer for data extraction and for use with the UR3e robot arm.

Figure 2: Leap Motion Camera Layout

Additionally, the Leap Motion has up-to-date API and documentation available. There are many example codes available in the C programming language that make developing code for the Leap Motion much more efficient. From the Leap Motion, a wide array of useful data can be extracted for use in controlling the UR3e robotic arms including hand position, hand orientation, hand velocity, thumb and index finger pinch position, and grab strength based on the user's fingers. These can be used to accurately control the robotic arms based on the user's hand movement.

The Leap Motion controller has a working area of 2 feet (609.6 mm) above and 2 feet on either side of the device, shown in Figure 3.

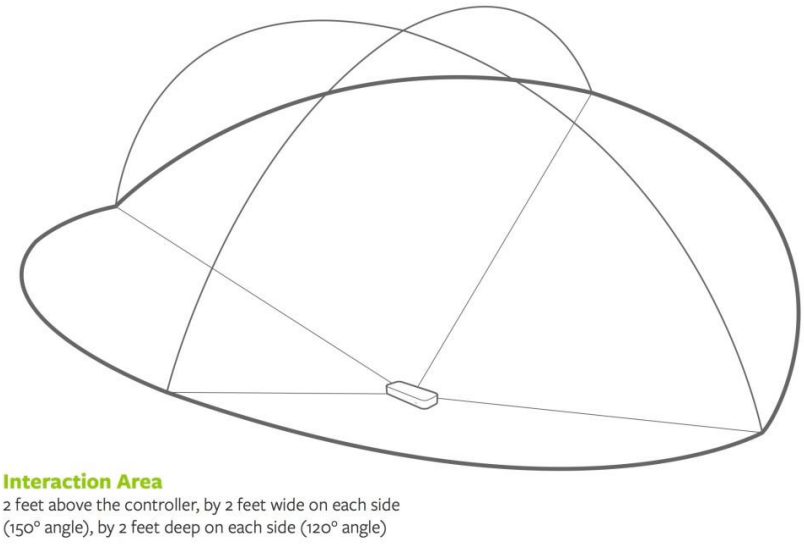


Figure 3: Leap Motion Tracking Area

With the maximum reach of the UR3e arm being 1.64 feet (450 mm) and a usable reach of 0.82 ft (250 mm) from the inner 0.65 ft (200 mm) singularity boundary, shown in Figure 4, this

enables the implementation of 1:1 positional tracking between both hands of the operator on one Leap Motion camera and two UR3e robotic arms.

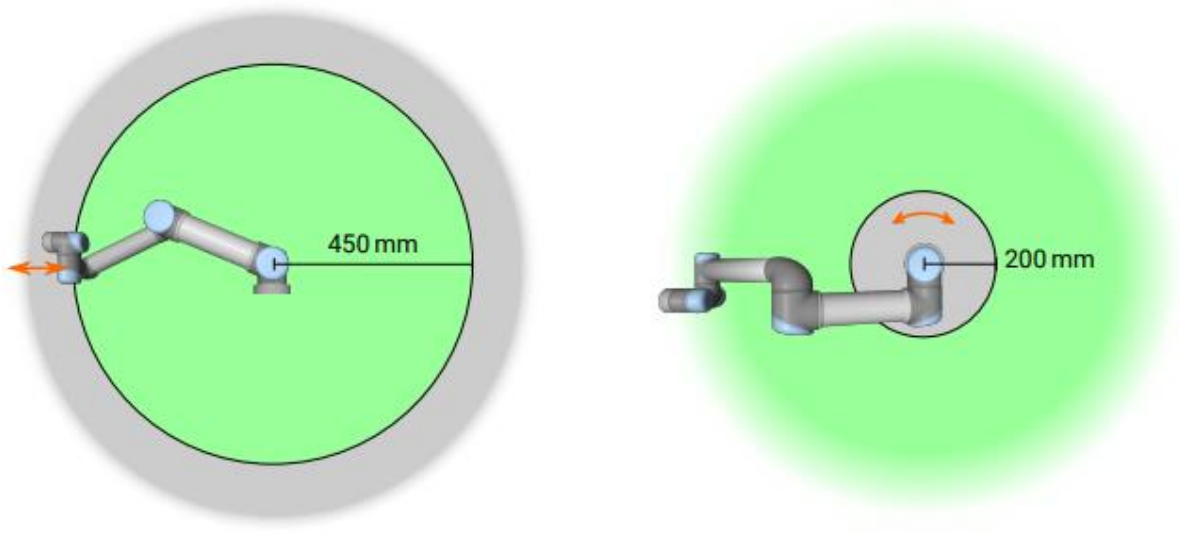


Figure 4: UR3e Workspace

To measure hand position, the onboard controller reads and adjusts the data measured by the infrared sensors before transferring the data to the Leap Motion hand tracking software. The software applies algorithms to the data to filter out background objects and account for ambient lighting. This filtered data is then processed to search for the user's hands within its field of view and determine their position in 3D space. The software constructs a representation of the user's hands with tracking models layered over the video feed from the camera module. This visualization can be seen in Figure 5.

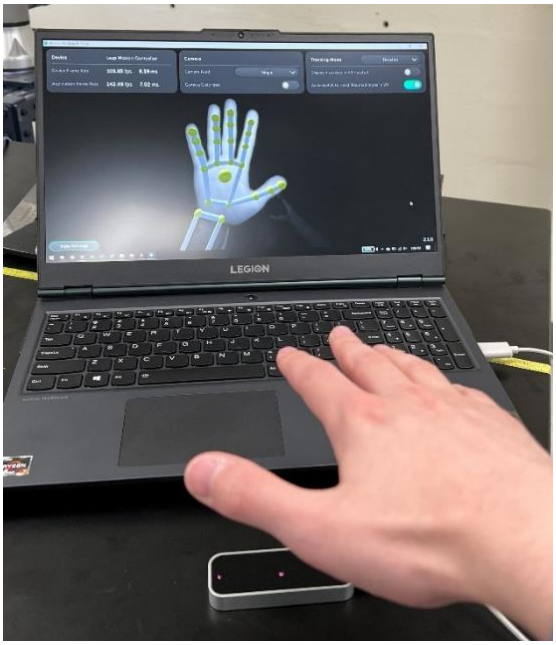


Figure 5: Leap Motion Hand Visualization

2.2.3 Gripper

For our gripper design, we decided on a claw gripper due to its simplicity and ability to be integrated with the UR3e. A design matrix was created, shown in Table 2, to compare the benefits and drawbacks of each of the proposed designs, displaying which was the most appropriate choice to follow based on the constraints and goals of the project. For reference, a score of five is the best and a score of zero is the worst.

Table 2: Gripper Type Design Matrix

Claw Styles	Mechanical Simplicity	Programming Simplicity	Mounting Simplicity	Cost	Research Required	Various Item Picking	Object Stability	Total
Humanoid Gripper	1	1	1	3	0	4	5	15
Suction Gripper	2	3	1	0	1	1	5	13
2-Prong Claw Gripper	5	5	5	5	5	3	3	31
3-Prong Claw Gripper	4	5	5	5	4	3	4	30

The criteria considered in the design matrix covers the technical constraint of variance along with the management constraints of schedule, in terms of complexity and research required, along with budget. One additional criterion considered, object stability, covers the project goal of successfully picking and placing objects.

The result of the design matrix supports the decision to use the claw style gripper, however it also ranks the two potential types of claw style grippers, two prong and three prong. A two-prong claw style gripper grasps the object with one prong on each side while a three-prong style gripper grasps the object with one prong on one side and two prongs on the other side. These designs are extremely similar; the notable differences being size, with the two-prong style being slightly smaller, and stability, with the three-prong style having three points of contact with the object as opposed to the two-prong only having two points of contact.

For our gripper material, we decided on using 3D printed filament over aluminum due to its simplicity and ability to be more rapidly prototyped. A design matrix was similarly made for this decision, shown in Table 3.

Table 3: Gripper Material Design Matrix

Claw Material	Fabrication Cost	Fabrication Time	Strength	Total
Aluminum	3	2	5	10
Plastic (PLA)	5	5	3	13

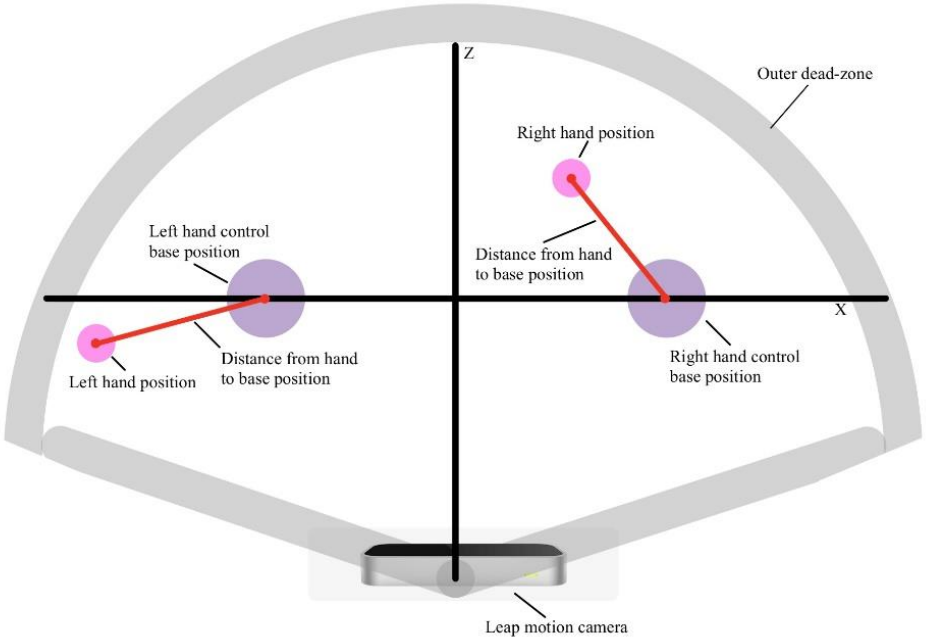
This design matrix was interpreted differently compared to gripper type, mainly regarding the strength. Although an aluminum part is far stronger than a 3D printed part, the project is mainly focused on working with smaller objects that will not put a high amount of strain on the gripper. Additionally, the 3D printed parts can be designed to be sturdier, either in Computer Aided Drafting (CAD) or by using stronger filament types and printed differently by increasing the infill of the part.

Our gripper decision allows for simple integration with the current configuration of the UR3e robots, without sacrificing any criteria that could jeopardize the success of the project. This allows prototyping and testing to begin sooner, resulting in more time for greater optimization of the robot.

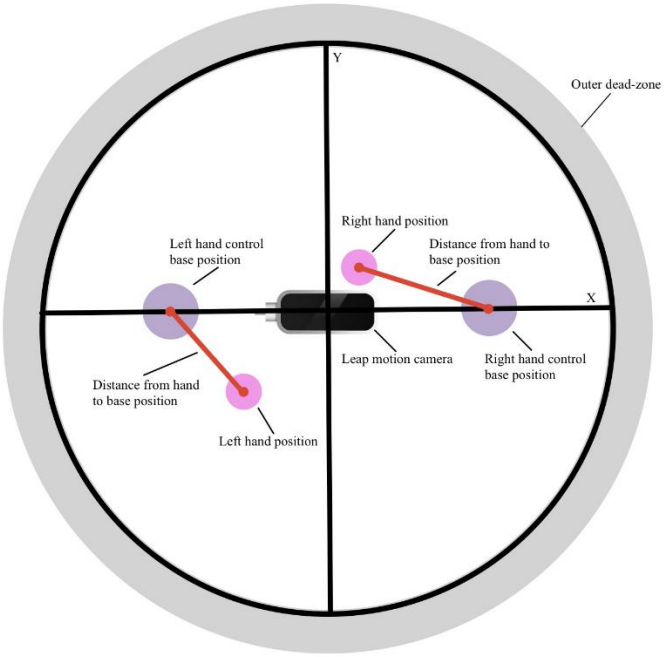
2.2.4 Control Algorithm

The chosen method of control was the full hand tracking mode. This control algorithm takes the position of the users' hands with respect to two points that serve as the center of each robot arm's workspace, shown in

Figure **6**. From the position of the user's hands, the algorithm uses inverse kinematics calculations to move the robot arms accordingly. The program has functions in which it can either control two arms at once or one at a time. When only one hand is found to be in the Leap Motion camera's tracking space, the program will default to controlling only the right arm, but if both hands are in the workspace, the program will control both arms. The program also has built-in gesture recognition that can stop the tracking of a hand or close the gripper. This method of control was chosen due to its intuitive control and ease of use for completing tasks. This method also requires less experience with the system than other tracking and control options.



a)



b)

Figure 6: Full Hand Tracking Conventions Shown in the a) XZ and b) XY Plane

2.3 Alternative Designs

2.3.1 Robotic Arms

There are other numerous robotic arms and systems that would be viable to complete the main goals of this project. Some of these robotic systems include the Lynxmotion AL5D articulated arm and the Rethink Robotics Baxter. The AL5D, shown in Figure 7, is a four degree of freedom articulated arm powered by an Arduino clone developed by Lynxmotion. Its four degrees of motion include a rotating base, shoulder, elbow, and wrist joints. This arm has a reach of approximately 410 mm, 190° around the center of the robot. The main downside to this robot is its accuracy, the AL5D utilizes potentiometers for position control which results in more error than systems utilizing encoders. The AL5D also has a small workspace and would only allow for smaller systems to be worked on.

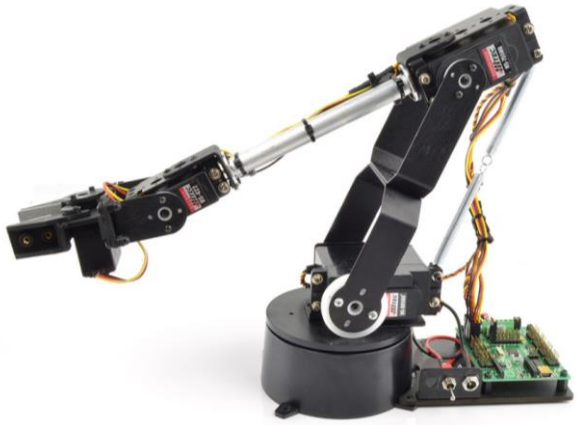


Figure 7: Lynxmotion AL5D

The Baxter, shown in Figure 8, is a dual seven degree of freedom humanoid robot. According to Williams (2017), the Baxter's seven degrees of motion include shoulder roll, shoulder pitch, elbow roll, elbow pitch, upper wrist roll, lower wrist pitch, and lower wrist roll. These arms each have a reach of 1.175 m at a 192° range. The main downsides to this robot are its lack of accuracy, as well as it only having mechanical safety features, instead of software integrated safety features.

Figure 8: Rethink Robotics Baxter

2.3.2 Hand Tracking

The alternative option for hand tracking that was considered was the use of an Intel RealSense D455 depth camera, shown in Figure 9. While the RealSense is not a dedicated hand tracking device, there are tracking algorithms available for use with the camera. The RealSense is compatible with both standard 64-bit computers and the ARM processor used in the Raspberry Pi. The RealSense would have been used to track the users' hands, with the Raspberry Pi acting as the communication and data processing device between the RealSense and the UR3e robotic arms.

Figure 9: Intel RealSense D455 Depth Camera

We decided against the use of the Intel RealSense D455 because, while it is compatible with a wider range of processors than the Leap Motion, the hand tracking algorithms are not as up-to-date or as well-documented as the Leap Motion. The older hand tracking algorithms of the RealSense do not perform as well as they are less accurate, and lack of documentation compared to the Leap Motion makes development less ideal. The decision matrix shown in Table 4 displays the justification for choosing the Leap Motion over the Intel RealSense D455.

Table 4: Hand Tracking Camera Design Matrix

Sensor	Dedicated Hand Tracking	Supports x64 Archtecture	Supports ARM Archtecture	Open Source APIs and Doctumentation	Total
LeapMotion	5	5	0	5	15
Realsense D435i	0	5	5	2	12

We decided against the use of the Raspberry Pi because the Leap Motion drivers do not support its ARM processor. While the RealSense does support this processor, because of our decision to use the Leap Motion instead, the use of the Raspberry Pi was no longer viable.

2.3.3 Gripper

Two main aspects of the gripper - type and material - allow for design freedom. The first alternative gripper design considered is a humanoid style hand. This style of hand, shown in Figure 10, is extremely complex, but it allows for high levels of dexterity which can be taken advantage of in software. This design utilizes individual servo motors to control the gripper's fingers independently. However, this would lead to a higher cost, increased programming difficulty, and introducing more potential points of failure in the robot.

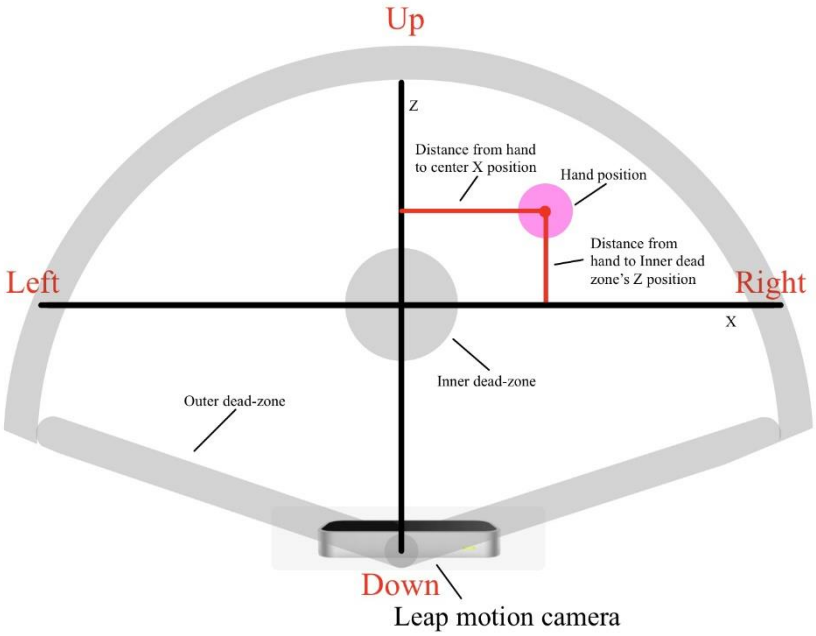
Figure 10: Hiwonder Robotic Hand

The second alternative gripper design considered is a suction style gripper, which is commonly seen on industrial pick-and-place robots. Similar to the humanoid gripper, this design is extremely complex and would not integrate well with the variance constraint of the project. Suction-based grippers are most effective when working with items consistently shaped items. The picking and placing of objects of various sizes, one of the goals of the project, could not be accomplished effectively by a suction style gripper.

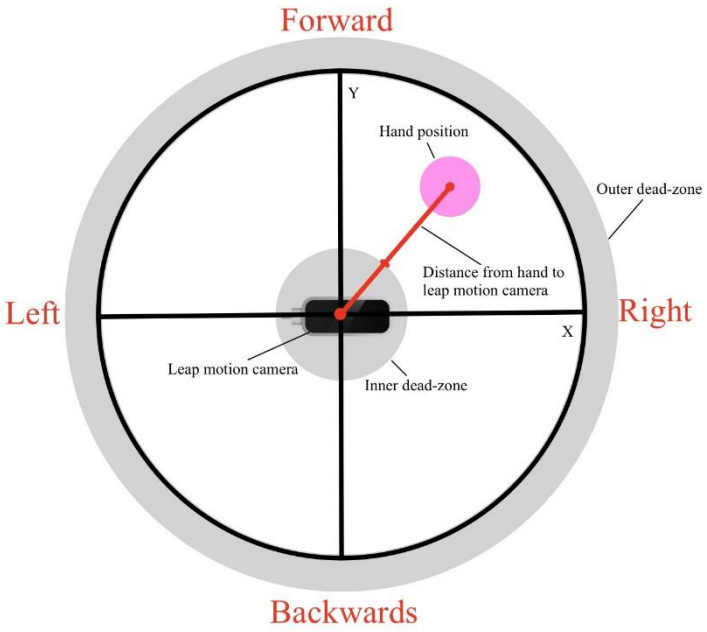
There were only two reasonable materials to fabricate the gripper from: aluminum and 3D printed filament (in this case, PLA). The default gripper on the robot is made of aluminum, which would be able to be machined in Widener's machine shop. Similarly, a 3D printed gripper would also be able to be designed and fabricated at Widener.

2.3.4 Control Algorithm

There were two main methods proposed for controlling the robotic arms, a joystick style of control and a full hand tracking option. The joystick control would set a deadzone in the center of the work area where users would leave their hand to stop any motion from the robot arm. Essentially, any hand motion within the deadzone would result in no motion from the robot arm. Once the hand leaves the inner deadzone, shown in Figure 11, the algorithm would take the position of the hand and calculate the distance from the outer edge of the inner dead zone to the hand. The robot arm would then move that direction at a speed relative to the distance. The hand's position is broken down into its X, Y, and Z components. It would then continue this movement until the user's hand reached the outer or inner deadzones or the robot arm has reached the limit of its workspace. While this means of control is a valid option, we found that it was less user-friendly and could easily cause confusion for the user when trying to control two arms accurately at once.



a)



b)

Figure 11: Joystick Mode Conventions Shown in the a) XZ and b) XY Plane

2.4 Engineering Standards and Safety Codes

Safety standards are not a priority for the project in its current state but would be introduced if the Touchless Mechatronics Laboratory was to be sold commercially. Beginning with the Universal Robots 3e arm, it meets global safety standards for industrial robots (ISO 10218-1, ISO 13849-1 & -2), global safety standards for industrial systems (ISO 10218-2), regional global acceptance standards (ISO 10218-1 & -2), 3rd party safety certifications, and key safety clauses within ISO 10218-1. The full extent and descriptions of these standards can be found in Appendix A. Additionally, the Leap Motion controller is certified compliant by the Federal Communications Commission (FCC) to safety and electrical regulatory standards. Despite the compliance of these two primary components of the project, when integrated with each other, they lose their safety certifications. Our project solution has no impact in the global, economic, environmental, or societal contexts in its current state.

2.5 Engineering Design Factors

Of the following engineering factors: public health, safety, welfare, global, cultural, social, environmental, and economic, our project considers and relates to public health and safety. The project focuses on the safety of students in both a public health and a general safety aspect. The original intent of the project is to focus on public health, reducing the potential surface-based transmission of COVID-19 among students. Additionally, the project was intended to be used to complete potentially dangerous tasks, such as soldering, in order to reduce the risk of related injuries. The project could also be used to safely handle hazardous tools or substances.

3 Procedures

3.1 System Composition

The overall system for our project consists of three main components, shown in Figure 12: the UR3e arms and their workspace, the Leap Motion camera, and the laptop running the software. The full robot workspace is kept clear to avoid any outside interference.

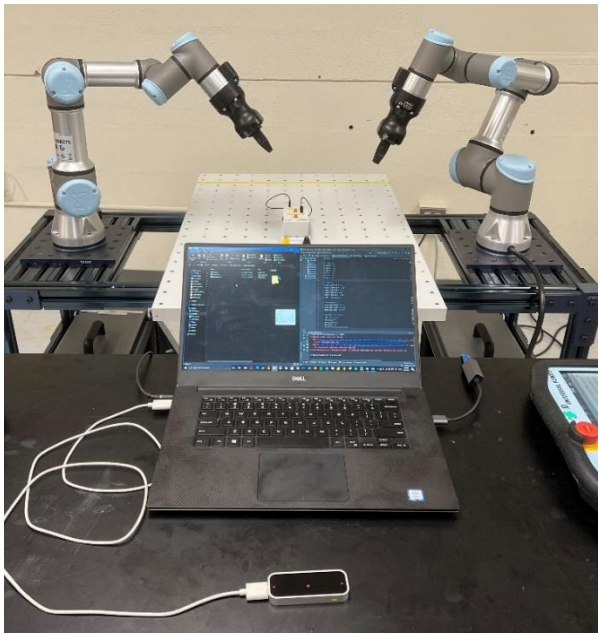


Figure 12: Full System Composition

3.2 Hardware

The hardware for the system is connected following the diagram shown in Figure 13.

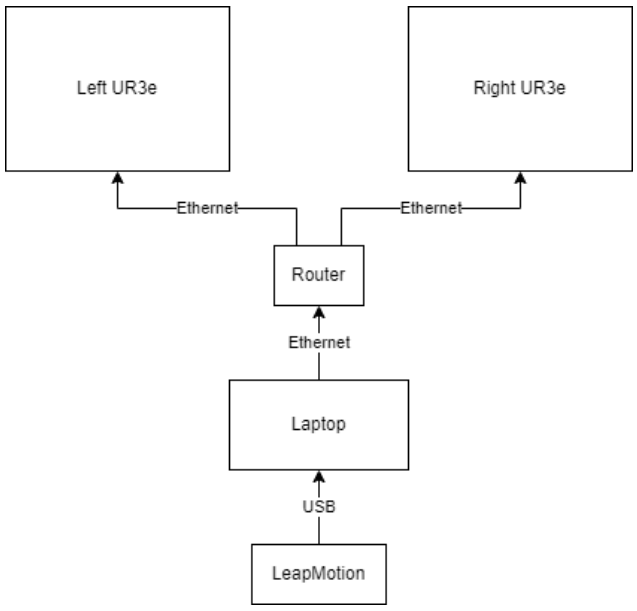


Figure 13: Hardware Connection Diagram

The Leap Motion's position and orientation data of the user's hands is transferred to the laptop running the control algorithm code using a USB connection. The resulting commands are then sent out to the corresponding UR3e arms via ethernet. A router is required for this connection, since the laptop used for lab tests only has one integrated ethernet port, and one ethernet connection is required for each arm.

As explained in the design approach, plastic, specifically PLA, was chosen to be the material for the gripper. The gripper was designed in Autodesk Inventor Professional 2022 and heavily based on the standard gripper design found on the Robotiq Hand-E gripper already used with the UR3es. Public CAD files found on Robotiq's website were utilized to ensure that the new 3D printed gripper prongs could be correctly mounted onto the Hand-E gripper.

The main difference between the existing Hand-E gripper prongs and the 3D printed gripper prongs, besides material, are overall length and shape. The stock Hand-E gripper prongs are slightly shorter - measuring 41.7mm from mounting point to tip - than the 3D printed gripper prongs, which measure 55mm from mounting point to tip. Additionally, the stock gripper prongs maintain a 21mm thickness throughout the length of the gripper prong until reaching the final filet, while the 3D printed gripper prongs taper from 21mm to 11mm. This gripper geometry reduces bulk where necessary, granting the robot the ability to reach into tighter spaces compared to the stock gripper. The new gripper geometry still retains an adequately sized grip area, 344.5mm^2 per gripper prong, allowing for grip objects to be held stably. The geometric differences between the stock and 3D printed gripper are shown in Figure 14.

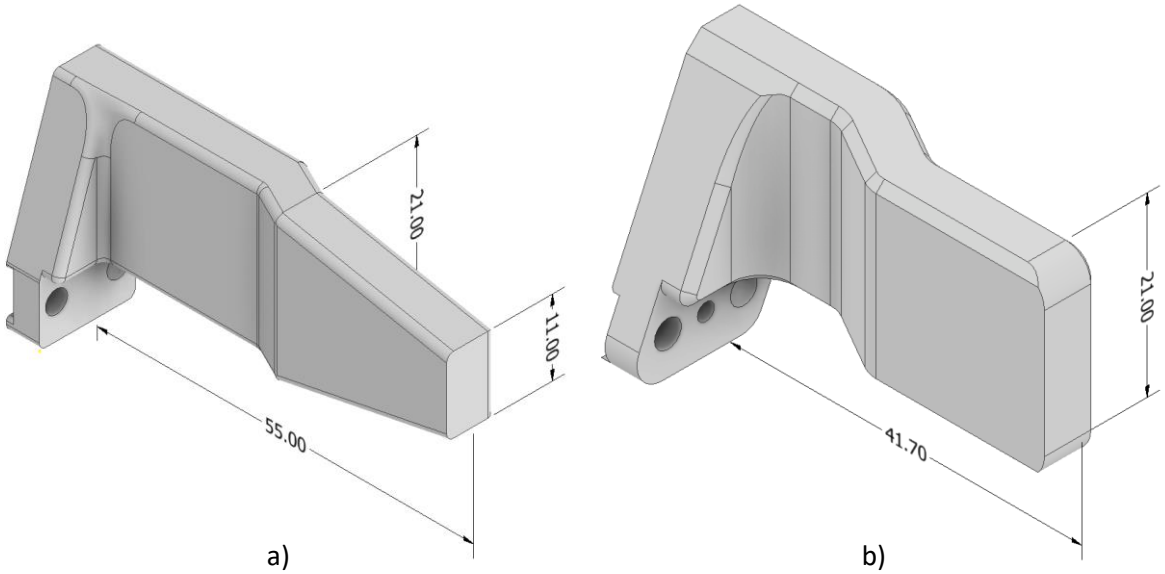


Figure 14: a) 3D Printed Gripper Geometry (mm), b) Stock Gripper Geometry (mm)

The CAD files for the entire Robotiq Hand-E were able to be imported into Inventor to test fit the gripper prongs before printing. The test fit can be seen in Figure 15.

Figure 15: Gripper Prongs on the Hand-E in Inventor

For the final set of grippers, they were printed out of PLA at a 0.1 mm layer height and 30% infill. These settings ensured the grippers were printed strong enough in order to grasp the components planned to be tested, without wasting filament.

3.3 Software

Prior to testing any of the control software written on the actual robot, the official Universal Robots E-Series URSim was used to simulate motion in a virtual environment. Testing all code in a simulator ensured the safety of the physical robot in case any major bugs were present. The only downside to the URSim software was its inability to integrate the aftermarket Robotiq Hand-E gripper; as a result, all gripper testing needing to be done on the physical robot. The only requirement to control the URSim version of the UR3e is to have the external control URCap enabled, which allows for the simulator to be controlled by a host device and monitored within the simulator as shown in Figure 16. URCaps are addons that enable additional support for more advanced 3rd party attachments.

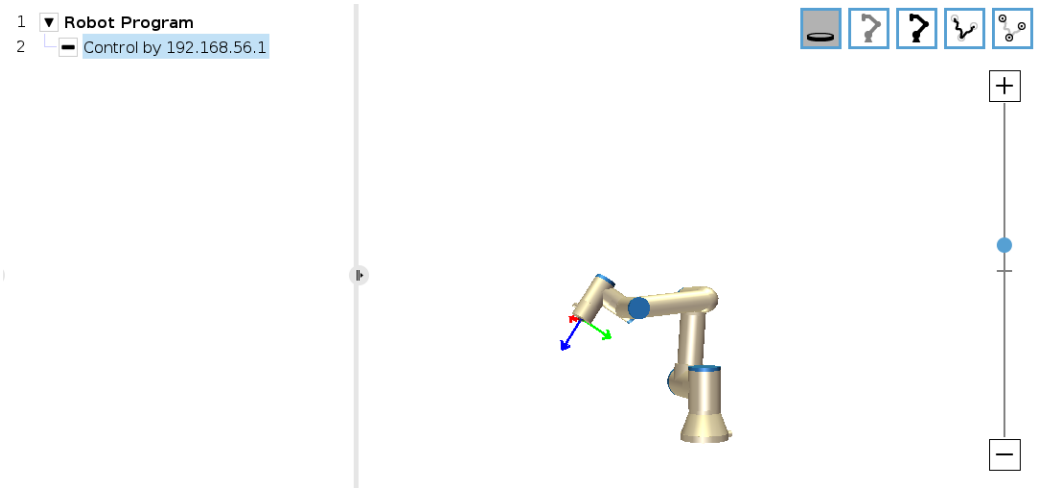


Figure 16: Simulated Control of the UR3e in URSim

Our arm control includes a PID controller which quickly and accurately moves the robot to a desired position. One of the issues that we encountered throughout the process of refining the motion control system, both in the simulator and on the physical robot, was overshoot. For the robot to operate in real time and not lag behind the user's movement, its joints needed to move at a relatively high speed. One of the main tradeoffs of PID controllers involves speed and overshoot. As seen in Figure 17, when the integral and derivative terms are held at a constant value and the proportional term is changed, there is a tradeoff between the signal reaching the desired value and not having overshoot. This is an exaggerated example; the PID terms in our project are tuned to maintain a quick response time and reduce overshoot. However, when working with a robot in a limited workspace, there were occasional instances where small overshoots occurred when the robot was being moved near its workspace limits which resulted in the robot attempting to travel to an impossible location and give an error.

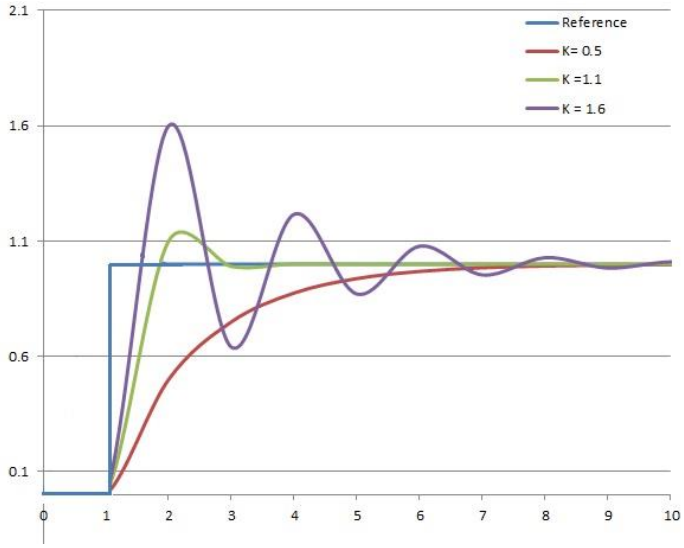
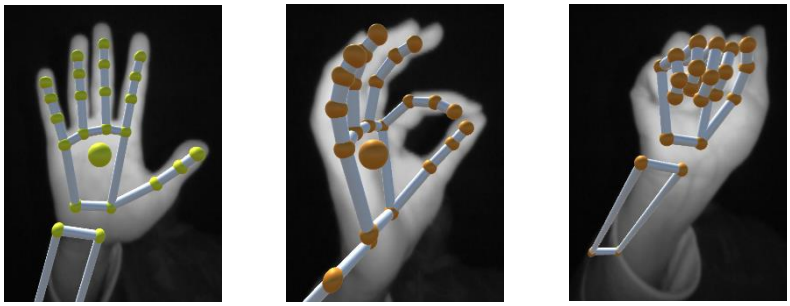


Figure 17: PID Control Tradeoff

One of the glaring issues we discovered from the preliminary hand tracking testing was that if the user's hand left the working area of the Leap Motion camera and returned in a different location, the arm would quickly jump to the new location at high speed. This would result in tripping the maximum velocity safety alarm within the robot's software, causing it to shut down and need to fully restart. This issue has not been fully resolved yet. However, we plan to implement a system where, if multiple identical frames are sent in succession to the UR3e, the robot will know that the user's hands have left the interaction area of the Leap Motion, and subsequently stop sending new information until the hands are placed back into the interaction area.

A second issue that was discovered from this preliminary testing phase was that there was no way to disconnect the user's hands from the Leap Motion tracking, in case the use was done, wanted to hold the robot still, or take a quick break. The Leap Motion would continue tracking the user's hand until it left its working area. This issue was solved by creating three

gestures for the Leap Motion to read for: open hand, closed hand, and pinched fingers. The open hand gesture is the default gesture used to control the robot, the Leap Motion will always track the user's hand if it is open and within the working area. The closed hand gesture is the gesture used to stop the arms at any point, or for any reason. The arms will remain stopped until the open gesture is seen. Finally, the pinched fingers gesture is used to control the arm's claws. When the user's index finger and thumb are brought together, the arm's grippers will pinch, and vice versa. Visualizations of these gestures in the Leap Motion tracking software are shown in Figure 18.



a) b) c)

Figure 18: a) Open Hand Gesture, b) Pinched Fingers, c) Closed Hand Gesture

The overall flowchart showing how data is converted and travels through the system is shown in Figure 19 and the full Python code can be seen in **Appendix D**.

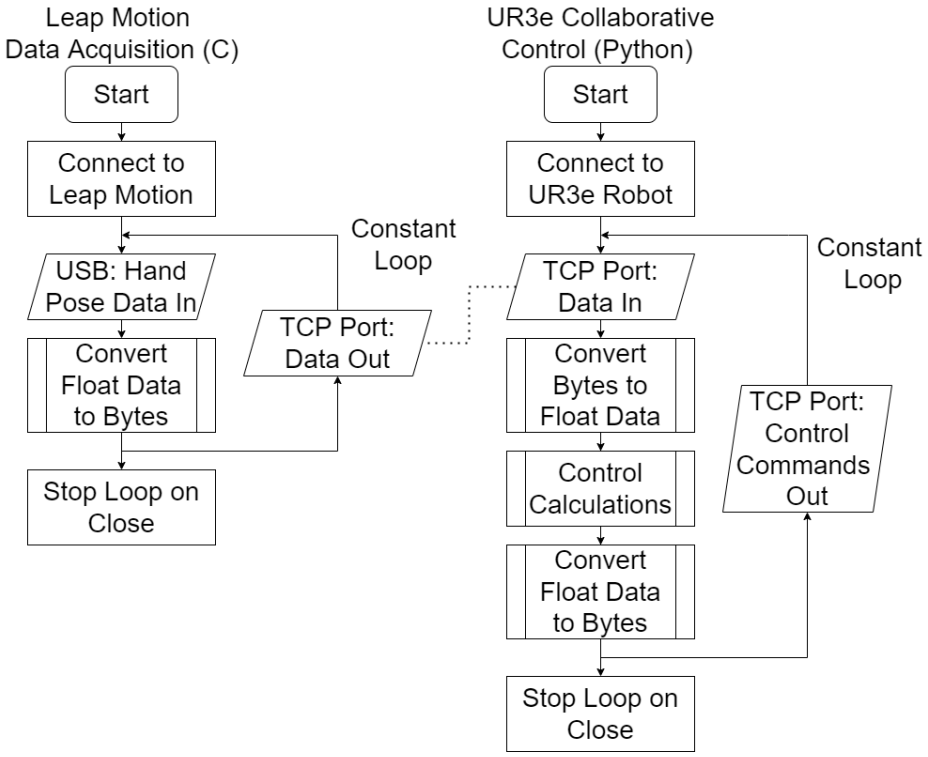


Figure 19: Leap Motion Input and Control Command Output

b)

3.4 System Tests

The result of our project was based on its ability to complete tasks outlined earlier as project goals. We devised numerous tests ranging from simple to more complex to determine the success of the system. The full testing criteria sheet used to quantify the success of the system is shown in Table 5.

Table 5: Robot Testing Criteria

	Tests	Success?
Movement	One-Handed	
	Two-Handed	
Joystick Movement	One-Handed	
	Two-Handed	
Gripper	One-Handed	
	Two-Handed	
Grasping	Push Button	
	22 Gauge Wire	
	14 Gauge Wire	
	Sponge	
	Screwdriver	
Picking & Placing	Picking 22 Gauge Wire	
	Placing 22 Gauge Wire	
	Picking 14 Gauge Wire	
	Placing 14 Gauge Wire	
Two-Hand Exchange	Push Button	
	22 Gauge Wire	
	14 Gauge Wire	
	Sponge	
Soldering	Holding Solder	
	Soldering	

The first two test groups cover movement utilizing the two proposed control algorithms, beginning with one-handed control, and followed by two-handed control.

The second test group, gripper, consists of testing the gripper controls of the robot, both one and two-handed.

The third test group, grasping, focuses on the system's ability to grasp and hold different items, starting with a pushbutton, different thickness of wires, picking up a sponge, and finally holding a screwdriver.

The fourth test group, picking and placing, builds off the previous, focusing on the system's ability to manipulate wires on a breadboard and other sockets. This testing includes both pulling both 22- and 14-gauge wires from a breadboard and reinserting them.

The fifth test group, two hand exchange, was not a goal from the beginning of the project, however, we realized it would be able to test the precision of the system when the user's hands were moving close together. Additionally, this skill could prove to be useful for lab activities if a lab material needs to be moved from one side of the workspace to the other. This testing included handing a pushbutton, different thickness of wires, and a sponge between the two arms.

The final test group, soldering, focuses on the system's ability to both grasp a soldering iron and solder to feed to the iron when soldering.

4 Results

The completed table used for testing the system can be seen in Table 6 and summarizes the system's overall performance.

Table 6: Robot Testing Results

	Tests	Success?
Hand Tracking Movement	One-Handed	Yes
	Two-Handed	Yes
Joystick Movement	One-Handed	No
	Two-Handed	No
Gripper	One-Handed	Yes
	Two-Handed	Yes
Grasping	Push Button	Sometimes
	22 Gauge Wire	Sometimes
	14 Gauge Wire	Yes
	Sponge	Yes
	Screwdriver	Sometimes
Picking & Placing	Picking 22 Gauge Wire	Yes
	Placing 22 Gauge Wire	No
	Picking 14 Gauge Wire	Yes
	Placing 14 Gauge Wire	Sometimes
Two-Hand Exchange	Push Button	No
	22 Gauge Wire	No
	14 Gauge Wire	No
	Sponge	Yes
Soldering	Holding Solder	No
	Soldering	No

For the movement test groups, the robot was successfully able to complete one and two-handed control when running the full hand tracking mode control algorithm. The robot was unsuccessful in tracking either hand when running the joystick control algorithm, which was expected as we had focused more on full hand tracking due to it being a more intuitive method of control.

For the gripper test group, the robot's gripper was successfully able to be controlled by pinching the index finger and thumb. The gripper control worked successfully on both hands and was able to be done simultaneously.

Moving into the grasping test group, the robot was successfully able to grasp most objects which measured between 1.5 and 19 mm. This was found to be the ideal working range for the printed gripper, with objects smaller not being able to be grasped properly and objects bigger beginning to bend and damage the plastic grippers. Objects like the 22-gauge wire were found to be slightly too thin to pick up while objects such as the screwdriver were found to be

too large. Repeated testing with the screwdriver specifically resulted in cracks beginning to form in the plastic grippers, shown in Figure 20.

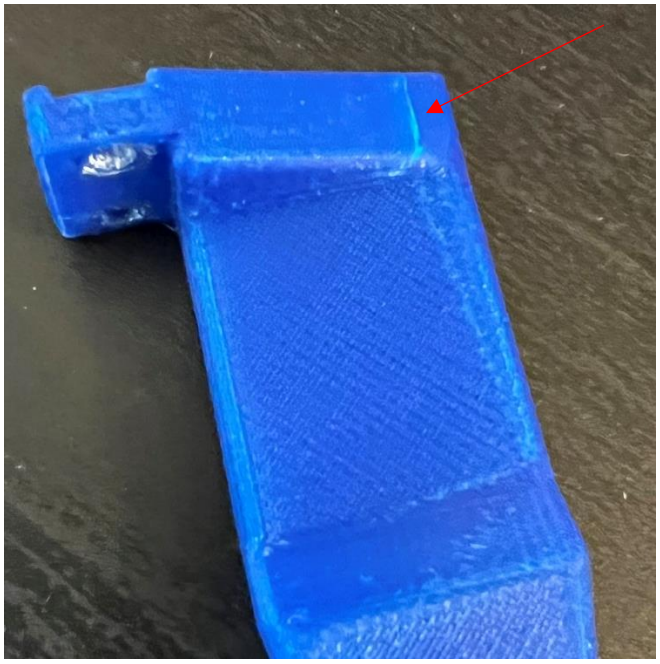


Figure 20: Torsion-Caused Crack

The 22-gauge wire, however, was able to be picked up if it was grabbed around the thicker, black plastic ends. There were also issues found with object shape. Original testing with the bare plastic grippers resulted in issues when picking up rounded objects such as the pushbutton. The lack of grip force control resulted in the rounded objects being pinched too hard and flying out of the gripper. To combat this, electrical tape was wrapped around the contact area on the grippers to increase grip friction. After this change, the gripper was able to consistently grasp the pushbutton. The gripper was able to successfully and consistently grasp the sponge due to its ideal thickness, larger size, and compressibility.

For the picking and placing test group, wires were pulled out of and reinserted into a breadboard or socket. For the 22-gauge wire, the system was only able to pull the wires out of the breadboard, reinserting them was an extremely precise process that resulted in numerous bent jumper wires. The system pulling the 22-gauge wire out of the breadboard is shown in Figure 21.

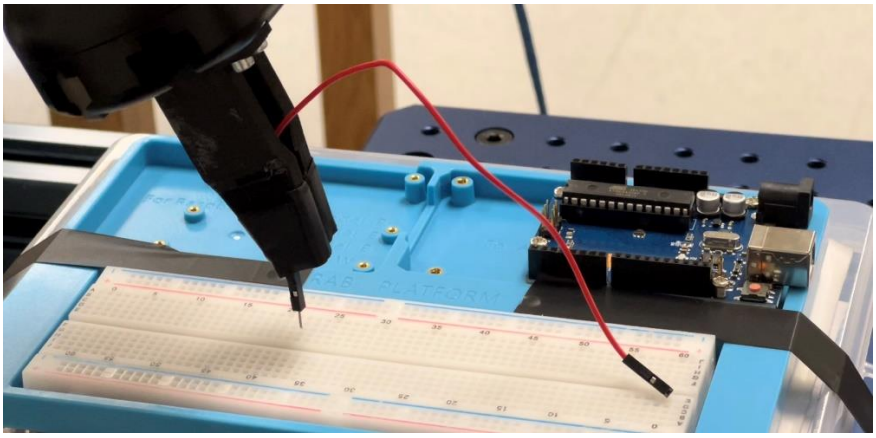
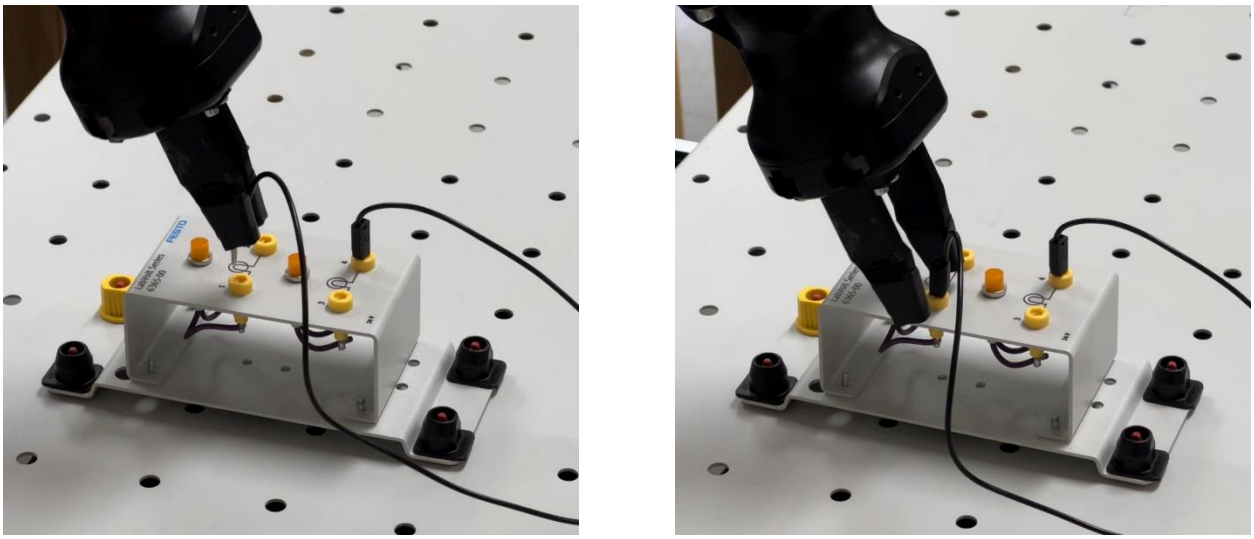


Figure 21: Removing 22-Gauge Wire from Breadboard

The system was able to remove the 14-gauge wire more easily. Additionally, it was able to repeatedly reinsert the wire back into the socket, albeit taking 20 seconds to line up. The system removing and reinserting the 14-gauge wire is shown in Figure 22.



a) b)

Figure 22: a) System Removing 14-Gauge Wire, b) System Reinserting 14-Gauge Wire

For the two-hand exchange test group, the system was successfully and consistently able to pass the sponge between the two arms, shown in Figure 23. The other objects: pushbutton, 22-gauge wire, and 14-gauge wire, were unable to passed from arm to arm. These objects were either too small to be able to interact with the two grippers at once or were too flimsy and fell out of the gripper.

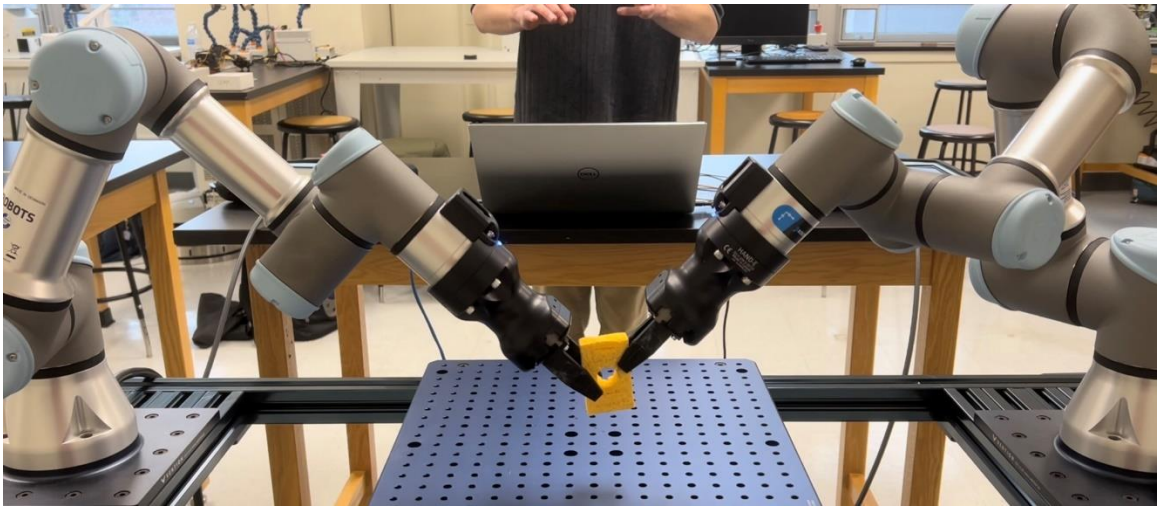


Figure 23: Sponge Two-Hand Exchange

The final test group consisted of soldering tests, which we determined the robot was not ready for, and did not attempt. To perform successful soldering, the system must be stable while holding both the soldering iron and solder. We determined the soldering iron and solder thicknesses fall outside of the gripper's ideal grasping range, 1.5-19 mm. Additionally, the current gripping solution is made of PLA plastic, which has a melting point of around 175°C and soldering irons are typically used at around 300-400°C. Although the iron itself does not get as hot, the gripper feeding the solder to the iron would be close to the extremely hot tip, and therefore in danger of melting.

Outside of the results documented in Table 6, there were some additional behaviors that were worth documenting. The first is the robot's tendency to jump when starting the code. Occasionally, when the code is first started and the user's hands are introduced into the Leap Motion's interaction area, the robot quickly jolts to the hand's location and trips the maximum velocity alarm, resulting in the system needing to be restarted. Also, there are some issues with the Leap Motion detecting the pinch gesture, mainly causing false closing and openings for the gripper. This results in parts being dropped or accidentally grasping objects too early.

5 Conclusions

The team was mostly successful in completing the original goals set out in the beginning of the project. The full system, consisting of the two UR3e six degree of freedom arms, Leap Motion camera, and laptop, was successfully designed and laid out to ensure the safety of the user and the arms. Real-time control software was successfully implemented with the system, allowing the user to control three degrees of freedom of the UR3e arms in order to manipulate objects. This control software consists of four parts to be able to successfully complete this task: (1) data acquisition for hand tracking, (2) motion control of the UR3e arms, (3) gesture interpretation for gripper control, and (4) maintaining the built-in safety features of the arms. This software was developed for the full hand tracking method of control, discussed previously. Finally, grippers were successfully designed and printed to allow for the robot to grasp the objects used in testing. The project was completed on time and under budget, spending in total $111.93, or 31.31%, of the $357.48 budget.

This project required extensive programming, networking, simulation, drafting, and fabrication to successfully complete these goals. The programming language was changed numerous times prior to settling on the final language of Python. Networking was required due to the presence of the router and understanding of IP addresses for controlling the two UR3e arms. Simulation was used heavily throughout the beginning two-thirds of the project, until the motion control algorithm was deemed to be optimized and safe enough for testing on the physical robot. Drafting in Autodesk Inventor was required for the development of the grippers. Finally, fabrication, by the means of 3D printing, was also required for the development of the grippers, however this task was offloaded to a more knowledgeable contact for the final printing.

Although this project was inspired by the COVID-19 pandemic and the sudden social changes it caused, we believe there are additional useful applications for this project both inside and beyond the lab.

The first is distance learning or training; since the system is hands-free, all the user needs is a Leap Motion camera to control the arms. This would require additional work to transfer the commands over the internet as opposed to just over ethernet cables, however it could become a valuable tool for training workers without them needing to be in person.

The second, more direct application of this project, although already employed in industry, would be for work in cleanrooms. Cleanrooms have requirements for the number of airborne particles present in the air, which results in human operators needing to be sanitized and decontaminated every time they enter. Employing a hands-free method would allow robotic arms to complete tasks within these cleanrooms, both lessening the amount of time taken sanitizing humans which, in turn, could potentially reduce the contaminants able to enter the room.

6 Recommendations

As with any project, there are a wide variety of functions and features that we would have wanted to include in this project but lacked the training and/or time to implement correctly. One of the major changes we would have made to this project, which would greatly improve its performance, is wrist control. Having the ability to rotate the robot's wrist joint would allow for much better recreation of the user's hand movement by adding an additional degree of freedom. Allowing the wrist to move independently from the elbow joint would also enable the robot to work in spaces that it is currently too big for. Additionally, since the UR3e wrist joint can rotate infinitely, wrist rotation would allow the robot to work with rotation-based tools such as screwdrivers.

A second major change we would like to make to this project would be to have finer force control for the gripper. Currently, the gripper only has two states, open and closed. However, there are 256 possible gripper force states available. Setting new gestures which corresponded to different gripper forces would allow the robot to have much better control over the objects it is grabbing. This would be a requirement for working with fragile electrical components and other delicate objects.

Another major change we would like to make for this project would be the implementation of a proper collision avoidance system between the two arms. In its current state, the two arms are able to hit each other, and although they would shut down prior to damaging themselves if this occurred, it would still greatly increase the safety of the system. Additionally, implementation of this safety feature would allow the user to move the arms faster, without having to focus on keeping them from colliding. Finally, this change would allow the arms to be opened up to their full ranges of motion, as they currently can only be used within a limited area to prevent them from moving too far into each other's workspace.

The next major change that we would like to make to this project is something discovered during our testing phase. Although the robot is able to accurately make precise movements, it is sometimes extremely difficult for the user to see what they are working on due to their depth perception and distance from the workspace. Our proposed solution to this would be to attach a camera to the wrist joint of each robot and stream its view to the user. This would allow the user to have an accurate view of what the robot sees, so they can make accurate movements confidently. A rough mockup of this system is shown in Figure 24.

a) b)

Figure 24: a) Example Camera Mounting Location, b) Example Camera Point of View

The next big change that we would have liked to implement to this project is improving the overall gripper design. Although our final design worked well for most gripping tasks, there were some which required extreme precision to properly grasp, such as very thin wires and rounded objects. Very small wires were able to easily slide through the gripper prongs while rounded objects, combined with our current lack of gripper force control, were occasionally turned into projectiles. Additionally, gripper strength needs to be reconsidered, the PLA gripper prongs can bend or crack when gripping objects. Again, this can be reduced through the proper implementation of gripper force control.

The final substantial change that we would have liked to fix about this project is the robot's tendency to quickly move to the user's hand position when the code is first started. This issue causes the robot to move too quickly and occasionally force a safety shutdown. This would mainly just be a quality-of-life improvement for the system.

Additionally, there are some changes that are still notable that focus on more niche applications and use cases for the robot. The first of these changes we would like to implement is various gripper designs, including both different shapes and materials. Not every task can be completed easily with just the default gripper prongs.

Another smaller change that we would have liked to implement would be a solder extruder that would attach to the arm. This solder extruder, which would look similar to a 3D printer's filament feeding system, would allow the robot to complete one handed soldering, reducing the risk of melting the gripper if two handed soldering was attempted.

These recommendations would allow the robot to be used confidently to work with densely populated breadboards, solder through hole joints on PCBs, and comfortably grip objects of different sizes and materials.

7 List of References

Chen, C., Chen, L., Zhou, X., & Yan, W. (2017, December). Controlling a robot using leap motion. In *2017 2nd International Conference on Robotics and Automation Engineering (ICRAE)* (pp. 48-51). IEEE.

Colgan, A. (2014, July 4). *Interaction Area* [Illustration]. Leap-Motion-Interaction-Area. https://blog.leapmotion.com/how-to-build-your-own-leap-motion-art-installation/leap-motion-interaction-area/

Hennessy, N. T., Toomey, S., Gautier, V., O'Reilly, S., de Barra, E., Hanrahan, E. O., & Hennessy, B. T. (2022). COVID-19 contamination of high-touch surfaces in the public domain. *Irish Journal of Medical Science (1971-)*, 1-2.

Hiwonder. (n.d.). *uHand: Hiwonder Robotic Hand Fingers Move Individually for Robot DIY* [Photograph]. Robotic Hand. https://www.hiwonder.hk/collections/robotic-hand/products/uhand-hiwonder-robotic-hand-fingers-move-individually-for-diy

IEEE. (2012). *Baxter* [Photograph]. Robots. https://robots.ieee.org/robots/baxter/

Intel. (n.d.). *Intel RealSense Depth Camera D455* [Photograph]. Intel RealSense. https://www.intelrealsense.com/depth-camera-d455/

Jang, I., Carrasco, J., Weightman, A., & Lennox, B. (2019, July). Intuitive bare-hand teleoperation of a robotic manipulator using virtual reality and leap motion. In *Annual Conference Towards Autonomous Robotic Systems* (pp. 283-294). Springer, Cham.

K, T. (n.d.). *Response of PV to step change of SP vs time* [Graph]. PID Controller. https://en.wikipedia.org/wiki/PID_controller#/media/File:PID_varyingP.jpg

Lynxmotion AL5D 4 Degrees of Freedom Robotic Arm Combo Kit. (n.d.). [Photograph]. RobotShop. https://www.robotshop.com/en/lynxmotion-al5d-robot-arm-combo-kit-4-dof.html?gclid=Cj0KCQjwgMqSBhDCARIsAIIVN1Vd8cKgv-r8wo8uCYGekctwnkAAERw8vn_mkJ4mt5uyoIfFdvKDG0QaAo7GEALw_wcB

Park, S. O., Lee, M. C., & Kim, J. (2020). Trajectory planning with collision avoidance for redundant robots using Jacobian and artificial potential field-based real-time inverse kinematics. *International Journal of Control, Automation and Systems*, *18*(8), 2095-2107.

Robotiq. (n.d.). *Hand-E Adaptive Gripper*. Retrieved from https://robotiq.com/products/hand-e-adaptive-robot-gripper?ref=nav_product_new_button

SFUPTOWNMAKER. (n.d.). *Top Down View* [Photograph]. Leap Motion Teardown. https://learn.sparkfun.com/tutorials/leap-motion-teardown/all

Universal Robots. (n.d.). *Universal Robots e-Series User Manual*. Version 5.0.2. Retrieved from https://s3-eu-west-1.amazonaws.com/ur-support-site/41169/UR3e_User_Manual_en_US.pdf

Universal Robots. (n.d.). *UR3e* [Photograph]. UR Shop. https://shop.universal-robots.com/ur3e/

Van Doremalen, N., Bushmaker, T., Morris, D. H., Holbrook, M. G., Gamble, A., Williamson, B. N., ... & Munster, V. J. (2020). Aerosol and surface stability of SARS-CoV-2 as compared with SARS-CoV-1. *New England Journal of Medicine*, *382*(16), 1564-1567.

Vysocký, A., Grushko, S., Oščádal, P., Kot, T., Babjak, J., Jánoš, R., Sukop, M., Bobovský, Z. (2020). Analysis of precision and stability of hand tracking with leap motion sensor. *Sensors*, 2020, 4088.

Williams, R. (2017). Baxter Humanoid Robot Kinematics. *Ohio University*. 1-68. Retrieved from https://www.ohio.edu/mechanical-faculty/williams/html/PDF/BaxterKinematics.pdf

Zacharaki, A., Kostavelis, I., Gasteratos, A., & Dokas, I. (2020). Safety bounds in human robot interaction: A survey. *Safety Science*, *127*, 104667.

Zedtwitz-Liebenstein, K. (2022). SARS-CoV-2: low virus load on surfaces in public areas. *Environmental Science and Pollution Research*, 1-4.

8 Acknowledgments

Team 19 would like to begin by thanking our faculty advisor, Dr. Xiaomeng Shi, for his guidance throughout the two semesters spent working on this project.

Secondly, the team would like to thank the Widener Robotics Department for their gracious allowance of their machinery and space throughout this project. Without their support, this project would not have been possible.

Finally, the team would like to thank Josh Kenna for 3D printing the final version of the arm's grippers on his personal printer.

Appendix

Appendix A: Full Standards and Safety Codes

Global Safety Standards for All Industrial Robots

ISO 10218-1 : Manufacturer of robots

ISO 13849-1 & -2 : Provides safety requirements and guidance on the principles for the design and integration of safety-related parts of control systems (SRP/CS), including safety software.

Global Safety Requirements for Robot Systems

ISO 10218-2 : Integrator of robot systems

ISO 13849-1 & -2 : Safety-related parts of control systems

ISO/TS 15066 : Technical Specification with additional guidance and recommendations for collaborative applications.

Global Acceptance of ISO 10218-1 and ISO 10218-2

Europe: Harmonized, shown as EN ISO 10218-1 & -2

USA: National adoption as ANSI/RIA R15.06

Canada: National adoption as CAN/CSA Z434

Japan: National adoption as JIS B 8433-1 & -2

Republic of Korea: National adoption as KS B ISO 10218-1 & -2

3rd Party Safety Certifications for All UR Robots

ISO 10218-1 : The global safety standard for industrial robots, including those with power and force limiting (PFL) capabilities for collaborative operation

Key Safety Clauses from ISO 10218-1

§ 5.10: Robots designed for collaborative operation shall comply with 1 or more of the requirements in § 5.10.2 through § 5.10.5

§ 5.10.2 safety-rated monitored stop

Safeguard Stop safety function fulfills § 5.10.2.

§ 5.10.5 power and force limiting by inherent design or control

UR robots: power & force limiting safety functions fulfills § 5.10.5.

§ 5.12.3 safety-rated soft axis and space limiting

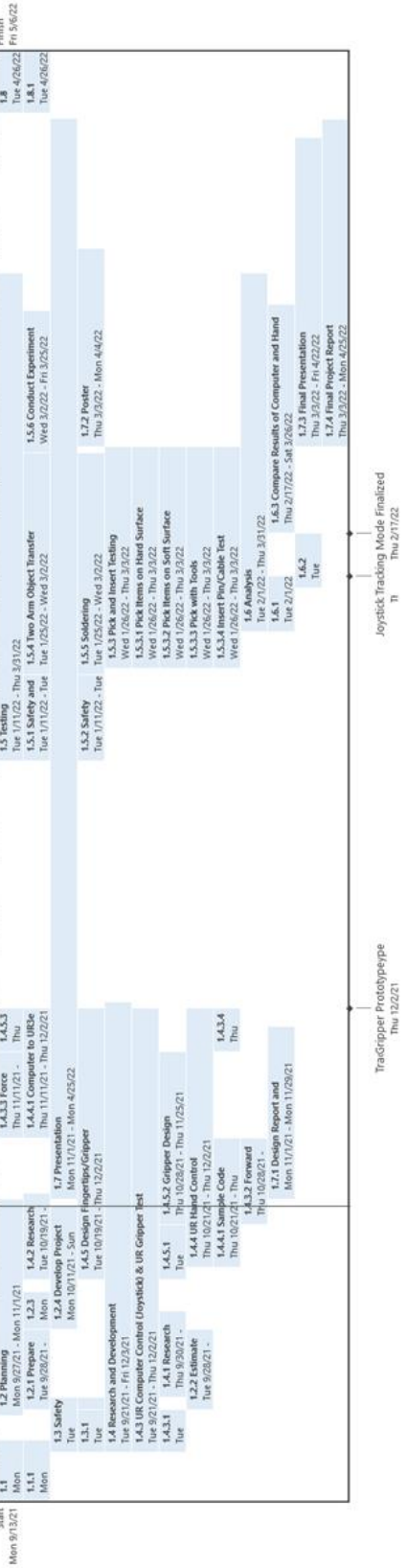
With UR robots, the following can be used for § 5.12.3:

- Safety Boundaries (Planes)
- Joint Position Limits
- Pose Limits for the tool flange and TCP.

Appendix B: Full Cost Breakdown

Item	Budget	Committed	Forecast	Comments
UR3e Arms	$0.00	$0.00	$0.00	Provided by the Robotics Department
3D Printer	$0.00	$0.00	$0.00	Provided by the Robotics Department
Filament	$0.00	$0.00	$0.00	Provided by the Robotics Department
Filament	$50.00	$0.00	$0.00	Additional filament (if necessary)
Raspberry Pi	$0.00	$0.00	$0.00	Provided by the Robotics Department
Intel Realsense Camera	$0.00	$0.00	$0.00	Provided by the Robotics Department
LeapMotion Camera (x2)	$177.90	$111.93	$111.93	1 provided by the Robotics Department, 1 purchased by team member
Gripper Hardware	$20.00	$0.00	$0.00	Screws, nuts, tools, etc
Other Materials	$50.00	$0.00	$0.00	User protection, cables, robot work surface,etc
Contingency (20%)	$59.58			
Total	$357.48	$111.93	$111.93	

Appendix C: Project Timeline



Appendix D: Full Python Code

```
#!/usr/bin/env python
import math
import socket
import numpy as np
import quaternion
import rtde_control
import rtde_receive

# Master UR Robot Enable [0, 1]
Right_Master_Robot_Enabled = 1
Left_Master_Robot_Enabled = 1

# UR Robot Gripper Enable [0, 1]
Right_Gripper_Enabled = 1
Left_Gripper_Enabled = 1

# UR Robot IP-Addresses
UR_IP_R = "192.168.1.5"
UR_IP_L = "192.168.1.6"
# 192.168.56.102
# 192.168.137.114

# RTDE Control/Receive
if Right_Master_Robot_Enabled == 1:
    rtde_c_r = rtde_control.RTDEControlInterface(UR_IP_R)
    rtde_r_r = rtde_receive.RTDEReceiveInterface(UR_IP_R)
if Left_Master_Robot_Enabled == 1:
    #rtde_c_l = rtde_control.RTDEControlInterface(UR_IP_L)
    #rtde_r_l = rtde_receive.RTDEReceiveInterface(UR_IP_L)

# Gripper Port
PORTGRIP = 63352

# PC IP-Address
HOST = "192.168.56.1"  # Standard loopback interface address (localhost)
PORT = 2048  # Port to listen on (non-privileged ports are > 1023)

# Control Type [Absolute, Relative]
Control = "Absolute"

# Position:
right_px = 0
right_py = 0
right_pz = 0
left_px = 0
left_py = 0
left_pz = 0

# Rotation:
right_u = 0
right_v = 0
right_w = 0
left_u = 0
left_v = 0
```

```
left_w = 0

# Velocity:
right_vx = 0
right_vy = 0
right_vz = 0
left_vx = 0
left_vy = 0
left_vz = 0

# Offsets:
right_x_offset = 0.5
right_y_offset = -0.15
right_z_offset = 0
left_x_offset = 0.5
left_y_offset = -0.15
left_z_offset = 0

# Workspace
Workspace_Outer = 0.5
Workspace_Inner = 0.25
Workspace_Base = 0.09

# Radius from Center
right_r_xy = 0
right_r_z = 0
left_r_xy = 0
left_r_z = 0

# Angle from Center
right_r_xy_angle = 0
right_r_z_angle = 0
left_r_xy_angle = 0
left_r_z_angle = 0

# PID Position Gain
PID_Follow_Strength = 100

# Disable/Enable Forward Kinematics [0, 1]
Kinematics = 0

# UR Robot Toggle [0, 1]
Right_Robot_Enabled = 0
Left_Robot_Enabled = 0

# Link Length Array
link = np.array([0.15185, -0.24355, -0.2132, 0.13105, 0.08535, 0.0921])

# Read in Bytes Function
def readnbyte(sock, n):
    buff = bytearray(n)
    pos = 0
    while pos < n:
        cr = sock.recv_into(memoryview(buff)[pos:])
        if cr == 0:
            raise EOFError
```

```
        pos += cr
    return buff

# Robot Absolute Control
def ControlCommand(Robot, Control, PID_Follow_Strength, Workspace_Outer, Workspace_Inner,
Workspace_Base, gripper_grab, px, py, pz, rot_x, rot_y, rot_z, r_xy, r_z, r_xy_angle,
r_z_angle):
    if (Control == "Absolute") and (gripper_grab < 255):
        if (r_z < Workspace_Outer) and (r_xy > Workspace_Inner) and (py >
Workspace_Base):
            Robot.servoL([px, pz, py, rot_x, rot_y, rot_z], 0.0, 0.0, 0.01, 0.2,
PID_Follow_Strength)
        if (r_z < Workspace_Outer) and (r_xy > Workspace_Inner) and (py <
Workspace_Base):
            Robot.servoL([px, pz, Workspace_Base, rot_x, rot_y, rot_z], 0.0, 0.0, 0.01,
0.2, PID_Follow_Strength)
        if (px > 0) and (pz < 0):
            if (py > Workspace_Base):
                if (r_z >= Workspace_Outer) and (r_xy > Workspace_Inner):
                    Robot.servoL([abs(Workspace_Outer * math.cos(r_z_angle) *
math.cos(r_xy_angle)), -abs((Workspace_Outer * math.cos(r_z_angle)) *
math.sin(r_xy_angle)), abs(Workspace_Outer * math.sin(r_z_angle)), rot_x, rot_y, rot_z],
0.0, 0.0, 0.01, 0.2, PID_Follow_Strength)
                if r_xy <= Workspace_Inner:
                    Robot.servoL([abs(Workspace_Inner * math.cos(r_xy_angle)), -
abs(Workspace_Inner * math.sin(r_xy_angle)), py, rot_x, rot_y, rot_z], 0.0, 0.0, 0.01,
0.2, PID_Follow_Strength)
            if (py < Workspace_Base):
                if (r_z >= Workspace_Outer) and (r_xy > Workspace_Inner):
                    Robot.servoL([abs(Workspace_Outer * math.cos(r_z_angle) *
math.cos(r_xy_angle)), -abs((Workspace_Outer * math.cos(r_z_angle)) *
math.sin(r_xy_angle)), Workspace_Base, rot_x, rot_y, rot_z], 0.0, 0.0, 0.01, 0.2,
PID_Follow_Strength)
                if r_xy <= Workspace_Inner:
                    Robot.servoL([abs(Workspace_Inner * math.cos(r_xy_angle)), -
abs(Workspace_Inner * math.sin(r_xy_angle)), Workspace_Base, rot_x, rot_y, rot_z], 0.0,
0.0, 0.01, 0.2, PID_Follow_Strength)
        if (px < 0) and (pz < 0):
            if (py > Workspace_Base):
                if (r_z >= Workspace_Outer) and (r_xy > Workspace_Inner):
                    Robot.servoL([-abs(Workspace_Outer * math.cos(r_z_angle) *
math.cos(r_xy_angle)), -abs((Workspace_Outer * math.cos(r_z_angle)) *
math.sin(r_xy_angle)), abs(Workspace_Outer * math.sin(r_z_angle)), rot_x, rot_y, rot_z],
0.0, 0.0, 0.01, 0.2, PID_Follow_Strength)
                if r_xy <= Workspace_Inner:
                    Robot.servoL([-abs(Workspace_Inner * math.cos(r_xy_angle)), -
abs(Workspace_Inner * math.sin(r_xy_angle)), py, rot_x, rot_y, rot_z], 0.0, 0.0, 0.01,
0.2, PID_Follow_Strength)
            if (py < Workspace_Base):
                if (r_z >= Workspace_Outer) and (r_xy > Workspace_Inner):
                    Robot.servoL([-abs(Workspace_Outer * math.cos(r_z_angle) *
math.cos(r_xy_angle)), -abs((Workspace_Outer * math.cos(r_z_angle)) *
math.sin(r_xy_angle)), Workspace_Base, rot_x, rot_y, rot_z], 0.0, 0.0, 0.01, 0.2,
PID_Follow_Strength)
                if r_xy <= Workspace_Inner:
                    Robot.servoL([-abs(Workspace_Inner * math.cos(r_xy_angle)), -
abs(Workspace_Inner * math.sin(r_xy_angle)), Workspace_Base, rot_x, rot_y, rot_z], 0.0,
0.0, 0.01, 0.2, PID_Follow_Strength)
```

```
        if (px > 0) and (pz > 0):
            if (py > Workspace_Base):
                if (r_z >= Workspace_Outer) and (r_xy > Workspace_Inner):
                    Robot.servoL([abs(Workspace_Outer * math.cos(r_z_angle) *
math.cos(r_xy_angle)), abs((Workspace_Outer * math.cos(r_z_angle)) *
math.sin(r_xy_angle)), abs(Workspace_Outer * math.sin(r_z_angle)), rot_x, rot_y, rot_z],
0.0, 0.0, 0.01, 0.2, PID_Follow_Strength)
                if r_xy <= Workspace_Inner:
                    Robot.servoL([abs(Workspace_Inner * math.cos(r_xy_angle)),
abs(Workspace_Inner * math.sin(r_xy_angle)), py, rot_x, rot_y, rot_z], 0.0, 0.0, 0.01,
0.2, PID_Follow_Strength)
            if (py < Workspace_Base):
                if (r_z >= Workspace_Outer) and (r_xy > Workspace_Inner):
                    Robot.servoL([abs(Workspace_Outer * math.cos(r_z_angle) *
math.cos(r_xy_angle)), abs((Workspace_Outer * math.cos(r_z_angle)) *
math.sin(r_xy_angle)), Workspace_Base, rot_x, rot_y, rot_z], 0.0, 0.0, 0.01, 0.2,
PID_Follow_Strength)
                if r_xy <= Workspace_Inner:
                    Robot.servoL([abs(Workspace_Inner * math.cos(r_xy_angle)),
abs(Workspace_Inner * math.sin(r_xy_angle)), Workspace_Base, rot_x, rot_y, rot_z], 0.0,
0.0, 0.01, 0.2, PID_Follow_Strength)
        if (px < 0) and (pz > 0):
            if (py > Workspace_Base):
                if (r_z >= Workspace_Outer) and (r_xy > Workspace_Inner):
                    Robot.servoL([-abs(Workspace_Outer * math.cos(r_z_angle) *
math.cos(r_xy_angle)), abs((Workspace_Outer * math.cos(r_z_angle)) *
math.sin(r_xy_angle)), abs(Workspace_Outer * math.sin(r_z_angle)), rot_x, rot_y, rot_z],
0.0, 0.0, 0.01, 0.2, PID_Follow_Strength)
                if r_xy <= Workspace_Inner:
                    Robot.servoL([-abs(Workspace_Inner * math.cos(r_xy_angle)),
abs(Workspace_Inner * math.sin(r_xy_angle)), py, rot_x, rot_y, rot_z], 0.0, 0.0, 0.01,
0.2, PID_Follow_Strength)
            if (py < Workspace_Base):
                if (r_z >= Workspace_Outer) and (r_xy > Workspace_Inner):
                    Robot.servoL([-abs(Workspace_Outer * math.cos(r_z_angle) *
math.cos(r_xy_angle)), abs((Workspace_Outer * math.cos(r_z_angle)) *
math.sin(r_xy_angle)), Workspace_Base, rot_x, rot_y, rot_z], 0.0, 0.0, 0.01, 0.2,
PID_Follow_Strength)
                if r_xy <= Workspace_Inner:
                    Robot.servoL([-abs(Workspace_Inner * math.cos(r_xy_angle)),
abs(Workspace_Inner * math.sin(r_xy_angle)), Workspace_Base, rot_x, rot_y, rot_z], 0.0,
0.0, 0.01, 0.2, PID_Follow_Strength)

# Socket Communication Initialization
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
if Right_Master_Robot_Enabled == 1:
    sgr = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
if Left_Master_Robot_Enabled == 1:
    sgl = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((HOST, PORT))
s.listen(1)
conn, addr = s.accept()

# Enable Gripper
if Right_Master_Robot_Enabled == 1:
    sgr.connect((UR_IP_R, PORTGRIP))
if Left_Master_Robot_Enabled == 1:
```

```
    sgl.connect((UR_IP_L, PORTGRIP))

# Main Program
with conn:

    print(f"Connected by {addr}")

    # Enable Gripper
    if Right_Master_Robot_Enabled == 1:
        sgr.sendall(b'SET ACT 1\n')
    if Left_Master_Robot_Enabled == 1:
        sgl.sendall(b'SET ACT 1\n')

    # Main Loop
    while True:

        # Enable Gripper Position
        if Right_Master_Robot_Enabled == 1:
            sgr.sendall(b'SET GTO 1\n')
        if Left_Master_Robot_Enabled == 1:
            sgl.sendall(b'SET GTO 1\n')

        # Capture Old Hand Positions
        right_x_old = right_px
        right_y_old = right_py
        right_z_old = right_pz
        left_x_old = left_px
        left_y_old = left_py
        left_z_old = left_pz

        # Read In Leap Motion Data
        data = readnbyte(conn, 84)

        # Leap Motion Data Collection
        if data[0] == 0:
            right_px = -((65026 * data[1]) + (256 * data[2]) + data[3]) / 10000 +
right_x_offset
        if data[0] == 1:
            right_px = ((65026 * data[1]) + (256 * data[2]) + data[3]) / 10000 +
right_x_offset

        if data[4] == 0:
            right_py = ((65026 * data[5]) + (256 * data[6]) + data[7]) / 10000 +
right_y_offset
        if data[4] == 1:
            right_py = -((65026 * data[5]) + (256 * data[6]) + data[7]) / 10000 +
right_y_offset

        if data[8] == 0:
            right_pz = ((65026 * data[9]) + (256 * data[10]) + data[11]) / 10000 +
right_z_offset
        if data[8] == 1:
            right_pz = -((65026 * data[9]) + (256 * data[10]) + data[11]) / 10000 +
right_z_offset
```

```
        if data[12] == 0:
            right_u = ((65026 * data[13]) + (256 * data[14]) + data[15]) / 1000
        if data[12] == 1:
            right_u = -((65026 * data[13]) + (256 * data[14]) + data[15]) / 1000

        if data[16] == 0:
            right_v = ((65026 * data[17]) + (256 * data[18]) + data[19]) / 1000
        if data[16] == 1:
            right_v = -((65026 * data[17]) + (256 * data[18]) + data[19]) / 1000

        if data[20] == 0:
            right_w = ((65026 * data[21]) + (256 * data[22]) + data[23]) / 1000
        if data[20] == 1:
            right_w = -((65026 * data[21]) + (256 * data[22]) + data[23]) / 1000

        if data[24] == 0:
            right_vx = ((65026 * data[25]) + (256 * data[26]) + data[27]) / 10
        if data[24] == 1:
            right_vx = -((65026 * data[25]) + (256 * data[26]) + data[27]) / 10

        if data[28] == 0:
            right_vy = ((65026 * data[29]) + (256 * data[30]) + data[31]) / 10
        if data[28] == 1:
            right_vy = -((65026 * data[29]) + (256 * data[30]) + data[31]) / 10

        if data[32] == 0:
            right_vz = ((65026 * data[33]) + (256 * data[34]) + data[35]) / 10
        if data[32] == 1:
            right_vz = -((65026 * data[33]) + (256 * data[34]) + data[35]) / 10

        if data[36] == 0:
            left_px = ((65026 * data[37]) + (256 * data[38]) + data[39]) / 10000 +
left_x_offset
        if data[36] == 1:
            left_px = -((65026 * data[37]) + (256 * data[38]) + data[39]) / 10000 +
left_x_offset

        if data[40] == 0:
            left_py = ((65026 * data[41]) + (256 * data[42]) + data[43]) / 10000 +
left_y_offset
        if data[40] == 1:
            left_py = -((65026 * data[41]) + (256 * data[42]) + data[43]) / 10000 +
left_y_offset

        if data[44] == 0:
            left_pz = -((65026 * data[45]) + (256 * data[46]) + data[47]) / 10000 +
left_z_offset
        if data[44] == 1:
            left_pz = ((65026 * data[45]) + (256 * data[46]) + data[47]) / 10000 +
left_z_offset

        if data[48] == 0:
            left_u = ((65026 * data[49]) + (256 * data[50]) + data[51]) / 1000
        if data[48] == 1:
            left_u = -((65026 * data[49]) + (256 * data[50]) + data[51]) / 1000
```

```
        if data[52] == 0:
            left_v = ((65026 * data[53]) + (256 * data[54]) + data[55]) / 1000
        if data[52] == 1:
            left_v = -((65026 * data[53]) + (256 * data[54]) + data[55]) / 1000

        if data[56] == 0:
            left_w = ((65026 * data[57]) + (256 * data[58]) + data[59]) / 1000
        if data[56] == 1:
            left_w = -((65026 * data[57]) + (256 * data[58]) + data[59]) / 1000

        if data[60] == 0:
            left_vx = ((65026 * data[61]) + (256 * data[62]) + data[63]) / 10
        if data[60] == 1:
            left_vx = -((65026 * data[61]) + (256 * data[62]) + data[63]) / 10

        if data[64] == 0:
            left_vy = ((65026 * data[65]) + (256 * data[66]) + data[67]) / 10
        if data[64] == 1:
            left_vy = -((65026 * data[65]) + (256 * data[66]) + data[67]) / 10

        if data[68] == 0:
            left_vz = ((65026 * data[69]) + (256 * data[70]) + data[71]) / 10
        if data[68] == 1:
            left_vz = -((65026 * data[69]) + (256 * data[70]) + data[71]) / 10


        if Right_Master_Robot_Enabled == 1:
            # Actual TCP Pos
            right_pos_actual = rtde_r_r.getActualTCPPose()
            # Actual Joint Pos
            right_q_actual = rtde_r_r.getActualQ()
        if Left_Master_Robot_Enabled == 1:
            # Actual TCP Pos
            left_pos_actual = rtde_r_l.getActualTCPPose()
            # Actual Joint Pos
            left_q_actual = rtde_r_l.getActualQ()

        # Right Hand Quaternion to Axis-Angle Rotations
        #right_quaternion = np.quaternion(1, 1, 0, 0) * np.quaternion(1, right_w,
right_u, right_v)
        #right_axis_angle = quaternion.as_rotation_vector(right_quaternion)
        #right_rot_x = right_axis_angle[0]
        #right_rot_y = right_axis_angle[1]
        #right_rot_z = right_axis_angle[2]
        right_rot_x = right_pos_actual[3]
        right_rot_y = right_pos_actual[4]
        right_rot_z = right_pos_actual[5]

        # Left Hand Quaternion to Axis-Angle Rotations
        #left_quaternion = np.quaternion(1, 1, 0, 0) * np.quaternion(1, -left_w, -left_u,
left_v)
        #left_axis_angle = quaternion.as_rotation_vector(left_quaternion)
        #left_rot_x = left_axis_angle[0]
        #left_rot_y = left_axis_angle[1]
        #left_rot_z = left_axis_angle[2]
        #left_rot_x = left_pos_actual[3]
```

```
        #left_rot_y = left_pos_actual[4]
        #left_rot_z = left_pos_actual[5]

        # Pinch and Grab
        right_pinch_strength = ((65026 * data[72]) + (256 * data[73]) + data[74]) / 1000

        right_grab_strength = ((65026 * data[75]) + (256 * data[76]) + data[77]) / 1000

        #left_pinch_strength = ((65026 * data[78]) + (256 * data[79]) + data[80]) / 1000

        #left_grab_strength = ((65026 * data[81]) + (256 * data[82]) + data[83]) / 1000

        # Gripper Position
        gripper_grab_r = round(right_grab_strength * 255)

        gripper_pinch_r = round(right_pinch_strength * 255)

        #gripper_grab_l = round(left_grab_strength * 255)

        #gripper_pinch_l = round(left_pinch_strength * 255)

        # Absolute Control Calculations
        if Right_Master_Robot_Enabled == 1:
            right_r_xy = math.sqrt(pow(right_px, 2) + pow(right_pz, 2))
            right_r_z = math.sqrt(pow(right_r_xy, 2) + pow(right_py, 2))
            right_ee_r_xy = math.sqrt(pow(right_px - right_pos_actual[0], 2) +
pow(right_pz - right_pos_actual[2], 2))
            right_ee_r_z = math.sqrt(pow(right_ee_r_xy, 2) + pow(right_py -
right_pos_actual[1], 2))
            # Right Hand Angle Quadrant
            if right_px != 0:
                right_r_xy_angle = math.atan(abs(right_pz) / abs(right_px))
            if right_px == 0:
                right_r_xy_angle = math.pi
            if right_r_xy != 0:
                right_r_z_angle = math.atan(abs(right_py) / abs(right_r_xy))
        if Right_Master_Robot_Enabled == 1:
            left_r_xy = math.sqrt(pow(left_px, 2) + pow(left_pz, 2))
            left_r_z = math.sqrt(pow(left_r_xy, 2) + pow(left_py, 2))
            left_ee_r_xy = math.sqrt(pow(left_px - left_pos_actual[0], 2) + pow(left_pz -
left_pos_actual[2], 2))
            left_ee_r_z = math.sqrt(pow(left_ee_r_xy, 2) + pow(left_py -
left_pos_actual[1], 2))
            # Left Hand Angle Quadrant
            if left_px != 0:
                left_r_xy_angle = math.atan(abs(left_pz) / abs(left_px))
            if left_px == 0:
                left_r_xy_angle = math.pi
            if left_r_xy != 0:
                left_r_z_angle = math.atan(abs(left_py) / abs(left_r_xy))

        if ((right_px == right_x_old) && (right_py == right_z_old) && (right_pz ==
right_z_old)):
            Right_Robot_Enabled = 0
```

```
        if ((left_px == left_x_old) && (left_py == left_z_old) && (left_pz ==
left_z_old)):
            Left_Robot_Enabled = 0

        if ((right_px != right_x_old) && (right_py != right_z_old) && (right_pz !=
right_z_old)):
            if right_ee_r_z < 0.05:
                Right_Robot_Enabled = 1

        if ((left_px != left_x_old) && (left_py != left_z_old) && (left_pz !=
left_z_old)):
            if left_ee_r_z < 0.05:
                left_Robot_Enabled = 1

        if Right_Robot_Enabled == 1:
            # Right Arm Absolute Control
            ControlCommand(rtde_c_r, Absolute, PID_Follow_Strength, Workspace_Outer,
Workspace_Inner, Workspace_Base, gripper_grab_r, right_px, right_py, right_pz,
right_rot_x, right_rot_y, right_rot_z, right_r_xy, right_r_z, right_r_xy_angle,
right_r_z_angle)

        if Left_Robot_Enabled == 1:
            # Left Arm Absolute Control
            ControlCommand(rtde_c_r, Absolute, PID_Follow_Strength, Workspace_Outer,
Workspace_Inner, Workspace_Base, gripper_grab_l, left_px, left_py, left_pz, left_rot_x,
left_rot_y, left_rot_z, left_r_xy, left_r_z, left_r_xy_angle, left_r_z_angle)

        # Control Gripper Position
        if Right_Gripper_Enabled == 1:
            # Closed Right Hand Gripper
            if gripper_pinch_r > 0:
                sgr.send(b'SET POS 255\n')
            # Open Right Hand Gripper
            if gripper_pinch_r == 0:
                sgr.send(b'SET POS 0\n')
        if Left_Gripper_Enabled == 1:
            # Closed Left Hand Gripper
            if gripper_pinch_l > 0:
                sgl.send(b'SET POS 255\n')
            # Open Left Hand Gripper
            if gripper_pinch_l == 0:
                sgl.send(b'SET POS 0\n')

        if not data:
            break
        conn.sendall(data)
```