MEC E 460: Group 18 Phase III Report

Robotic System for Automated Dataset Generation

Fabri Sciences Inc

University of Alberta
Faculty of Engineering
Edmonton, AB, Canada

Submission Date: April 9, 2025

This report is submitted as part of MEC E 460 course requirements.

Executive Summary

Fabri Sciences Inc. has advanced the development of a robotic system for automated dataset generation, as specified by Kondor Devices Inc. This 9-axis system is designed to systematically photograph aluminum slides sputtered with tantalum at controlled angles and distances—producing datasets used for AI training to improve wound analysis and patient care.

Tantalum is a specialized metal used to create slides embedded with microscopic circles lined with antigens. When blood or urine samples are introduced into these circles, certain antibodies will react with the antigen-lined surface. A microscopic amount of protein coagulation occurs at the surface of the slide, allowing for certain wavelengths of visible light to reflect off of the tantalum surface. As a result, the tantalum will reflect specific wavelengths of light depending on the presence of unique antibodies. Such technology is commonly used by labs for COVID-19 diagnostics. By precisely adjusting the distance and angle of a camera relative to the slides, we can capture unique light patterns that help indicate the presence of disease.

Traditionally, determining the ideal conditions (angles, distances, lighting) requires a lab-based setup with tight control over each parameter. This process is not automatable, as it involves physically repositioning components like cameras and lights to test each condition.

The team's 9-axis robotic system automates all these adjustments—angles, distances, and lighting—allowing us to rapidly test a virtually limitless range of configurations without manual intervention. Where previously only 2 slides could be processed per day, the team's system enables data collection from over 30,000 slides in the same time frame—vastly accelerating research and enabling more accurate diagnostics.

Over the course of all three phases, a total of 625 engineering hours have been allocated to this project, amounting to a manpower cost of $56,250. This exceeds the original estimate of 541 hours projected prior to Phase III, as additional time was required to address unforeseen challenges in both mechanical assembly and software integration.

The prototype was completed at a subtotal cost of $5,399.70—coming in more than $4,000 under the original budget of $10,000. Final validation will focus on achieving clinical-grade dataset reproducibility through rigorous performance testing.

The final design integrates iPhone cameras, a custom-designed gantry system, robotic arm, gripper assembly, and a dedicated ROS-iOS interface. Together, these components enable seamless, high-throughput data collection using tantalum slides—all in a cost-effective package that aligns with Kondor Devices' operational objectives.

Fabri Sciences is proud to present this working prototype as part of the Phase III deliverables, demonstrating both its technical capabilities and real-world potential.

Word Count: 372

Table of Contents

Report Body Word Count: 5176

List of Figures

List of Tables

1 Introduction

The healthcare industry continually seeks faster and more precise diagnostic tools for disease detection and patient care. This imperative has driven Kondor Devices Inc. to partner with Fabri Sciences Inc. in the development of a 9-axis robotic system for automated dataset generation. By capturing detailed images of tantalum-sputtered aluminum slides at controlled angles and distances, the system provides extensive datasets suitable for AI-driven analysis, as well as system accuracy validation for the expansion of Kondor's blood diagnostic technology onto mobile devices.

While tantalum slides offer significant promise for clinical diagnostics—particularly in detecting diseases such as COVID—current research and testing methods typically require a strictly controlled laboratory environment. Through the team's system, Fabri Sciences enables a shift from limited manual testing to high-throughput, fully automated image capture, thus reducing operational costs and expediting research timelines.

2 Project Background

2.1 Current Challenges in Laboratory Testing

In many research facilities, testing the subtle light diffraction and reflection properties of tantalum slides involves a labor-intensive process. The slides, outfitted with antigen-lined microcircles, must be observed under meticulously controlled angles, distances, and lighting conditions. Laboratories often deploy static rigs and rely on trained personnel to reposition slides and equipment, achieving only incremental data collection—commonly constrained to just a few slides per day. A render of Kondor's legacy camera apparatus is included below:

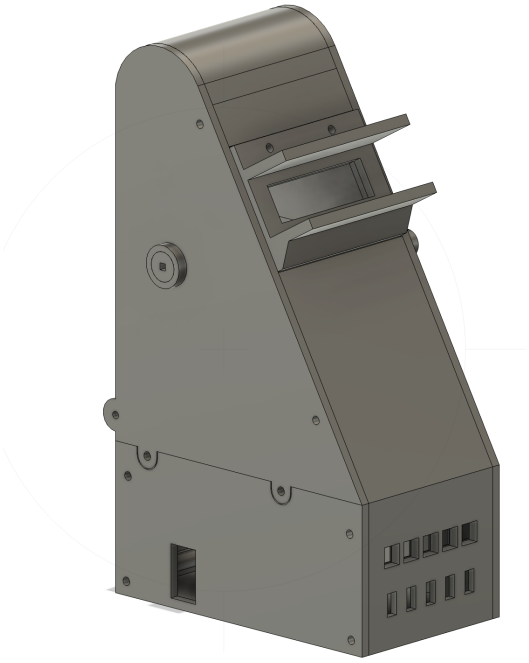


Figure 1. Legacy camera apparatus used in clinical lab settings.

This approach not only limits throughput but also restricts accessibility: only labs with the necessary specialized equipment and environmental controls can feasibly perform these tests. Consequently, the potential for broader clinical research and faster diagnostic validation remains underutilized.

2.2 Motivation for an Automated Solution

By integrating multi-axis movement with advanced imaging, the team's 9-axis robotic system overcomes these limitations. The robotic arm can adjust angles, distances, and lighting parameters on-the-fly, allowing the system to process tens of thousands of slides in a fraction of the time it would take under manual conditions. As such, the data required for AI model training can be gathered exponentially faster and more accurately, potentially transforming diagnostic capabilities across medical fields.

Moreover, the team's system's modular design ensures it can be replicated or adapted by any end-user with access to a compatible robotic arm—expanding beyond the traditional lab-centric model. This innovation moves the testing process toward a more flexible, decentralized setup, making high-throughput analysis of tantalum slides feasible in diverse clinical and research environments.

2.3 Tantalum Slides and Light Refraction

Tantalum is uniquely suited for this diagnostic process because of its high optical contrast and stability when sputtered onto aluminum. Each slide is embedded with an array of precisely patterned microcircles lined with biological antigens. When a blood or urine sample is introduced, these antigens bond with present proteins or pathogens, forming a thin biological film across the surface of each microcircle.

This thin film creates a 3D interference environment. Light refracted or reflected off these films behaves differently depending on the biological content of the sample. Subtle shifts in angle, intensity, and diffraction patterning can indicate the presence or absence of disease markers.

Figures 2 and 3 show both the physical appearance of the tantalum slides and a simplified diagram illustrating how light interacts with the thin film created by sample bonding. The ability to control lighting, camera position, and angle is critical to capturing consistent and useful data for diagnostic training.

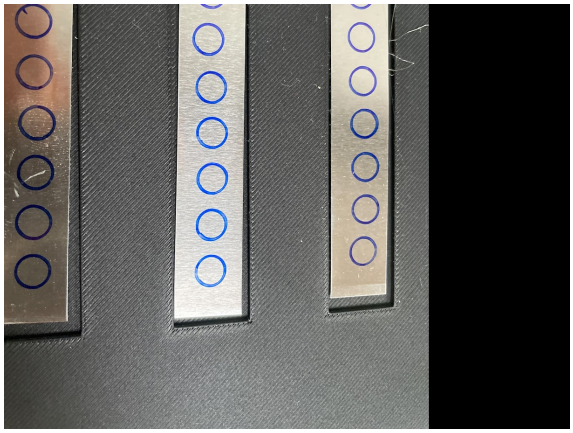


Figure 2. Tantalum-sputtered aluminum slides with micro-patterned antigen wells.

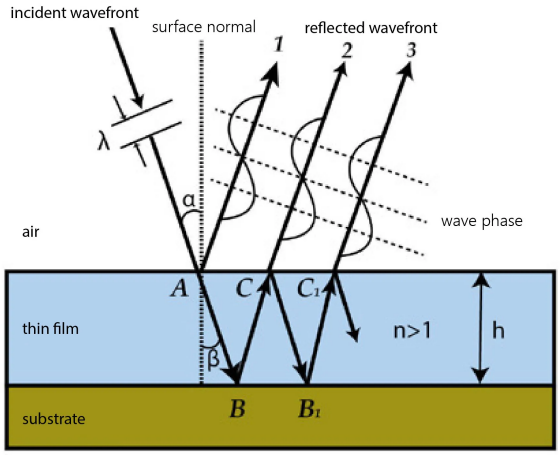


Figure 3. Diagram of light refraction through a thin biological film formed by antigen-sample bonding.

2.4 System Impact and Further Validation

In total, 625 engineering hours have been dedicated to developing this prototype—an investment reflecting the system's complexity and the unforeseen assembly and software integration challenges encountered during Phase III. These challenges have informed design improvements, ensuring that the final product is robust, extensible, and capable of delivering on its promise of faster, more comprehensive data acquisition.

Ultimately, the 9-axis robotic system aims to achieve clinical-grade dataset reproducibility. Ongoing and future validation efforts will assess how well the system's automated processes align with rigorous performance standards required for clinical diagnostics. By addressing the

limitations of current lab-based approaches and offering a scalable, user-friendly alternative, Fabri Sciences Inc.'s prototype stands to make a substantial impact on medical diagnostics and beyond.

3 Design Specification Revisions

Revisions to the original design specifications were made throughout the development process to reflect engineering constraints and prototype findings. Full details, including the updated compliance matrix and revised specification table, are provided in Section 8 and Appendix A, respectively.

4 Final Design Solution

The final prototype presented in this report builds upon the concepts developed during Phase II, where multiple design configurations were evaluated and refined. Phase III focused on translating these conceptual models into a fully functional and physically integrated system. The core subsystems finalized during this phase include the gripper design, robotic arm, gantry system, cloud server, and the combined Robotic Operating System (ROS) and computer vision interface.

To ensure fidelity between the design intent and the physical implementation, an extensive CAD model was created during Phase II. This digital model served as the foundation for mechanical integration, wiring layout, and subsystem alignment. Figure 4 shows the fully assembled CAD model used as a reference for manufacturing and system integration.

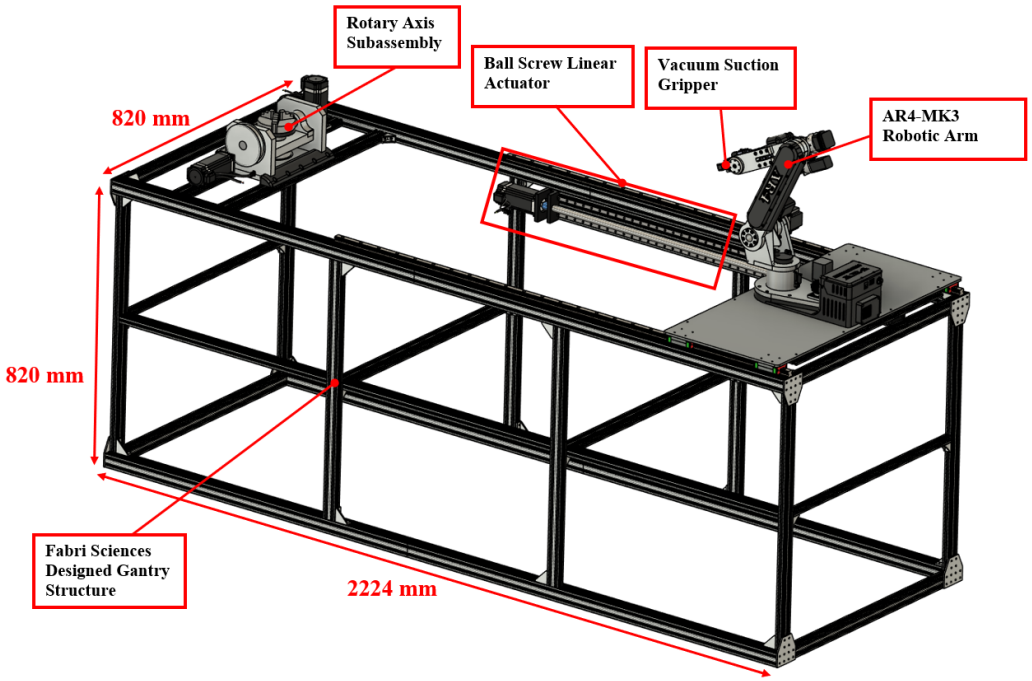


Figure 4. Fully assembled CAD model developed during Phase II conceptual design.

Compared to the deployed prototype, shown in Figure 5, the overall design remained highly consistent. The primary differences include the addition of tantalum slides and ArUco marker boards—both developed in Phase III to support computer vision and slide alignment within the ROS framework. It should be noted that ArUco marker boards are printed patterns consisting of unique square fiducial markers that can be easily detected and identified by computer vision algorithms, enabling precise localization and orientation of components within the system. Additionally, the vacuum pump and air tubes were physically integrated but not modeled in the original CAD due to their flexible nature and vendor variability.

Despite these minor deviations, the final system closely matches the original CAD model in form and function, validating the robustness of our Phase II conceptual design work and demonstrating the team's ability to execute a precise and well-integrated build process.

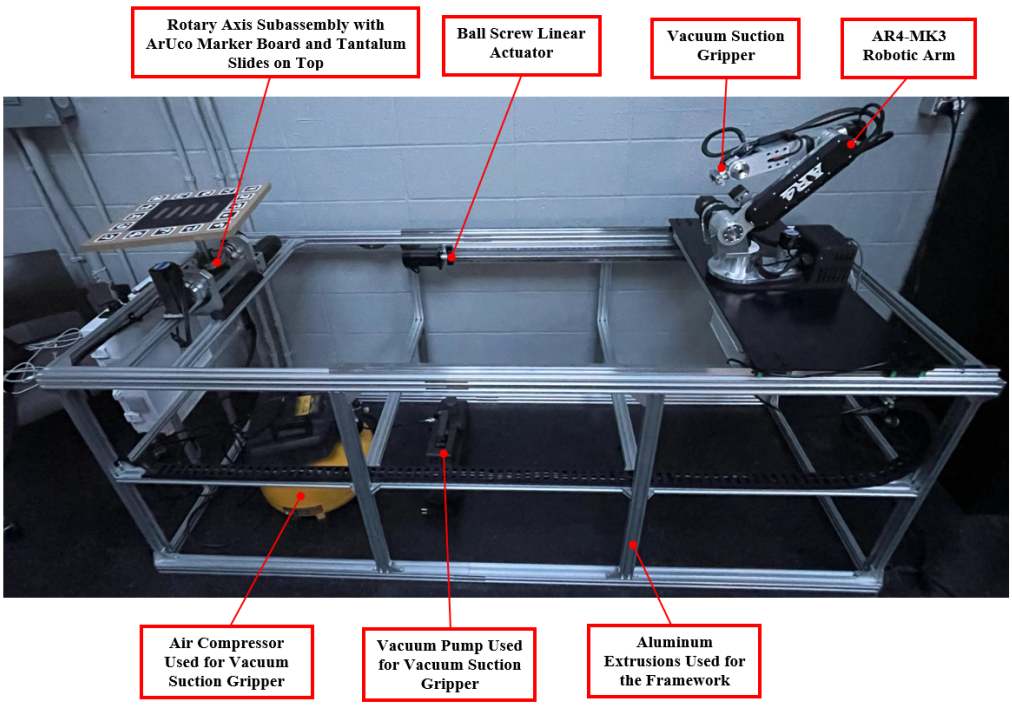


Figure 5. Deployed prototype system featuring tantalum slides, ArUco marker boards, and added pneumatic lines.

4.1 Gripper Design

The final concept selected for the gripper subsystem was a vacuum pad–based design. After evaluating several alternatives during Phase II, the vacuum gripper was chosen due to its simplicity, reliability, and compatibility with off-the-shelf components. The selected assembly was sourced from McMaster-Carr and features a threaded M6 attachment specifically suited for robotic arms. This commercial vacuum gripper is rated to lift objects up to 5 pounds—well beyond the force requirements for handling lightweight tantalum slides.

Figure 6 shows the exploded view of the vacuum assembly, which is composed of multiple modular parts for ease of replacement and maintenance. The full bill of materials for this subsystem is listed in Table 1.

To adapt the vacuum assembly for our robotic arm, a custom hub adapter was designed and 3D printed in-house. This part allowed for secure interfacing between the M6-threaded gripper and the robotic arm's native mounting plate. The adapter uses a 12 mm M4 screw to create a flush, rigid connection, ensuring repeatable accuracy during motion. To connect the adapter to the robotic arm, keyhold slots are utilized. This adapter is shown separately in Figure 7.

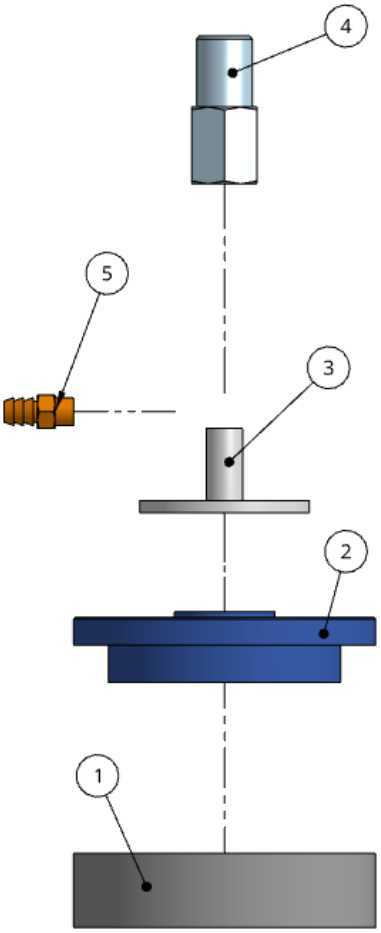


Figure 6. Exploded view of the vacuum pad gripper assembly used in the final design.

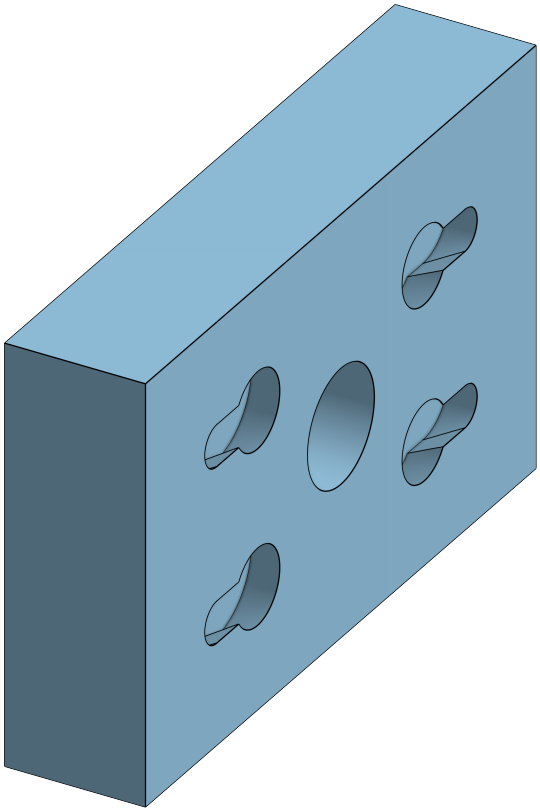


Figure 7. 3D printed adapter used to connect the vacuum assembly to the robotic arm via M6 and M4 interface.

Table 1: Simplified bill of materials for the suction-based gripper design.

Label #	Part #	Description	Qty
–	VAC-000	Complete Vacuum Assembly	1
1	VAC-001	Vacuum Pad	1
2	VAC-002	Vacuum Pad Attachment Piece	1
3	VAC-003	Base Plate for Attachment	1
4	VAC-004	M6 x 1mm Attachment Piece	1
5	VAC-005	6mm Barbed Side Air Connection	1
–	VAC-006	Air Tubes Set	1
–	VAC-007	Vacuum Pump	1
–	VAC-008	3D Printed Gripper Adapter	1

4.2 Robotic Arm

The final design concept for the robotic arm is the AR4-MK3, an open-source and DIY model that took around 100 hours to build for prototype integration. It supports ROS2 (Robotic Operating System 2), which is a critical requirement for the overall system to function. An annotated model of the AR4-MK3 is shown in Figure 8 along with the corresponding parts list in Table 2.

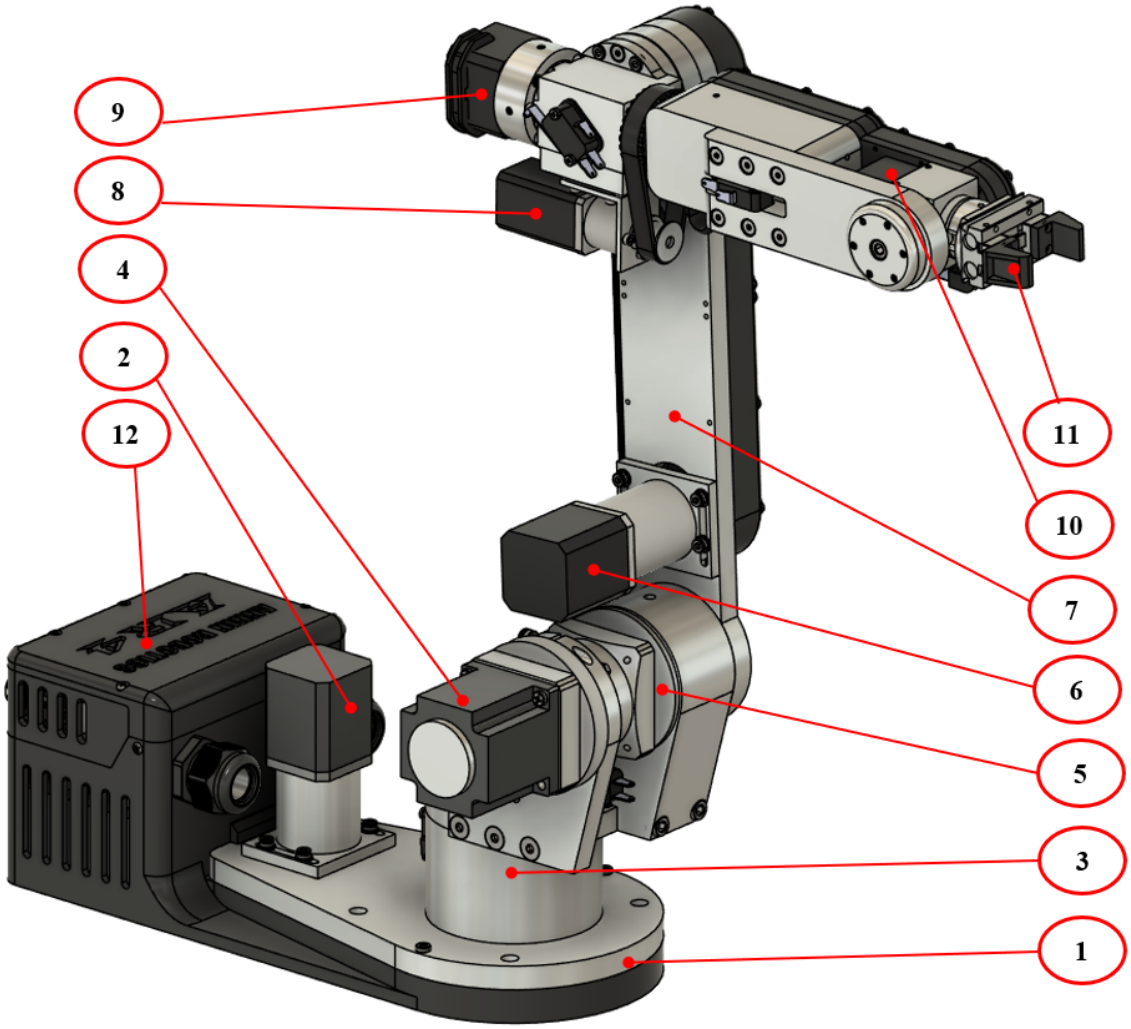


Figure 8. AR4-MK3 robotic arm where part classification is found in Table 2.

Table 2: Parts List for AR4-MK3 Robotic Arm

Label #	**Part Name**	**Qty**	**Part Description**
1	Base Plate	1	Provides stability and support for the arm
2	Base Rotation Motor	1	Controls the rotation of the entire arm around its vertical axis
3	Lower Arm Segment (Shoulder Link)	1	Connects the base to the mid-arm; responsible for major lifting movements
4	Shoulder Motor	1	Drives the shoulder joint, enabling vertical movement of the lower arm
5	Mid-Arm Segment (Elbow Link)	1	Connects the shoulder to the elbow, extending the arm's reach
6	Elbow Motor	1	Controls the bending motion at the elbow joint
7	Upper Arm Segment (Wrist Link)	1	Connects the elbow to the wrist, allowing controlled precision movements
8	Wrist Pitch Motor	1	Controls the up-and-down tilting motion of the wrist
9	Wrist Roll Motor	1	Rotates the wrist around its axis for better positioning
10	Wrist Yaw Motor	1	Adjusts the side-to-side movement of the wrist
11	End Effector (Gripper)	1	Allows the arm to grip, hold, or manipulate objects. It will be replaced by the gripper design
12	Control System	1	Includes the control board with 6 motor drivers, power supply, and all necessary wiring and connectors for signal and power distribution

Important aspects of the AR4-MK3 [1]:

- Payload capacity is 1.9 kg
- Mass is 12.25 kg
- DIY build
- Maximum reach is 62.9 cm

4.3 Gantry System

Fabri Sciences was able to develop a 3-axis robotic gantry system (a mechanical platform that moves a payload in linear directions) to precisely position a camera mounted on a robotic arm. The primary objective is to capture images at specific angles for dataset generation. It is important to note that the gantry subsystem is strictly for internal use and will not be consumer-facing; its purpose is to aid in dataset creation for the iOS app development that the client requested.

An annotated model of the robotic gantry's main structure is presented in Figure 9, with the corresponding parts list in Table 3.

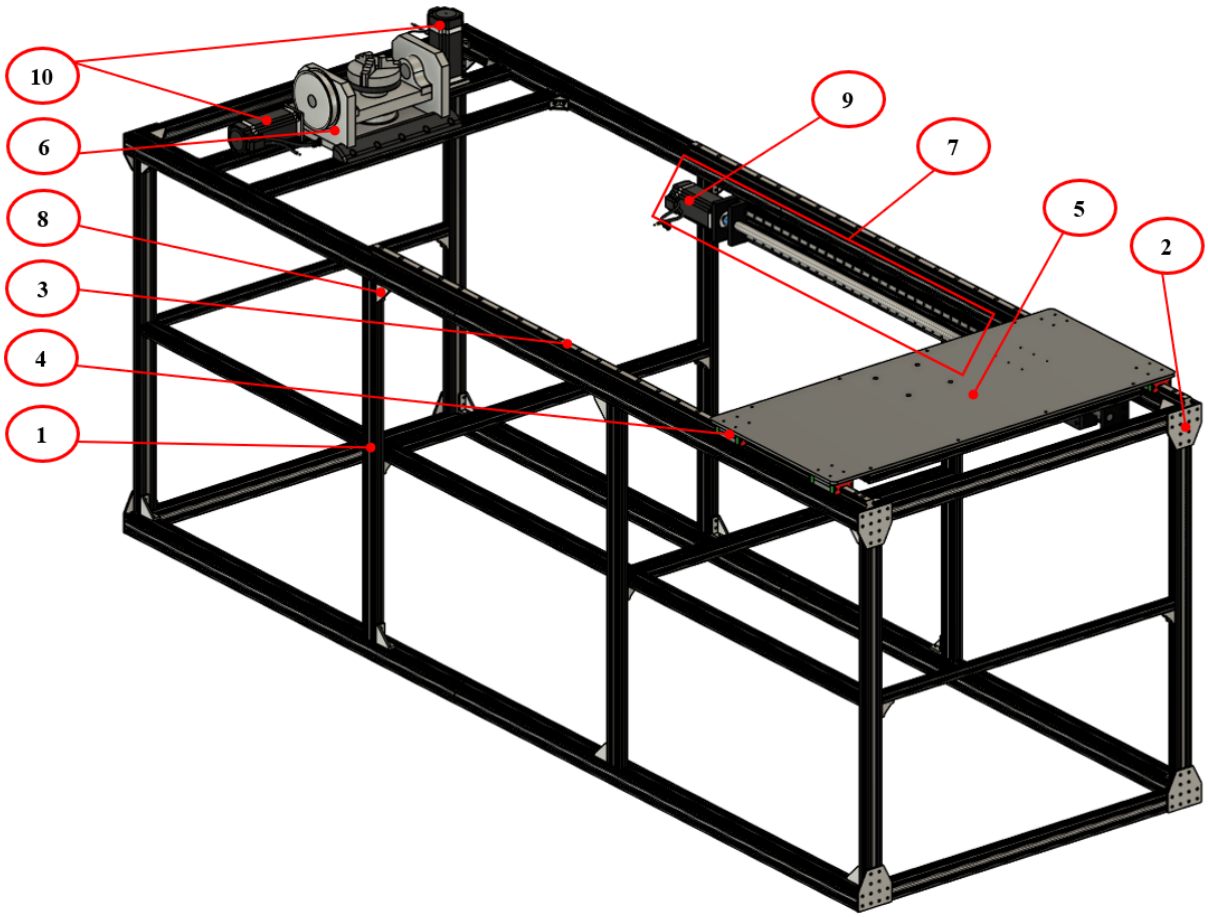


Figure 9. Annotated Gantry subassembly where part names are found in Table 3.

Table 3: Parts List for Gantry Assembly

Label #	**Part Name**	**Qty**	**Part Description**
1	Aluminum Extrusions	25 m	Includes different dimensioned 20 series extrusions for structural support and reinforcement
2	Aluminum Extrusion Fasteners	8 pcs	Used to ensure proper spacing and alignment of components
3	HGR20 Rail	2 pcs	Linear guide rails that provide smooth movement for the gantry system
4	HGR20CA	4 pcs	Linear rail blocks that slide along the HGR20 rails, allowing motion control
5	Robotic Arm Plate	1 pc	Mounting plate for securing the robotic arm to the gantry system
6	Rotary Axis Assembly	1 pc	Rotational mechanism allowing controlled movement of the robotic arm
7	Linear Axis Actuator	1 pc	Example concept to provide linear motion for moving the robotic arm along a single axis
8	Brackets	44 pcs	Aluminum corner brackets for securing and reinforcing extrusion connections
9	NEMA 23 Stepper Motor (Linear Axis)	1 pc	High-torque (3.0 Nm) stepper motor used to drive the linear motion system [2]
10	NEMA 23 Stepper Motor (Rotary Axis)	2 pcs	High-torque (3.0 Nm) stepper motors used for control of the rotary axis motion [2]

The final linear actuator design uses a ball screw-driven linear motion system, powered by an electric motor that transmits torque to the screw through a coupling. The ball screw provides gantry movement along the rail, with minimal friction due to ball bearings circulating within the helical grooves of the screw. Position sensors that are integrated within the NEMA 23 stepper motor (the team didn't have to design them) continuously monitor the system to ensure accurate gantry positioning and reliable control.

An annotated model of the ball screw linear actuator is shown in Figure 10, with the corresponding parts list in Table 4. The actuator's dimensions are provided in Figure 11, with the stroke length of 1100 mm highlighted.

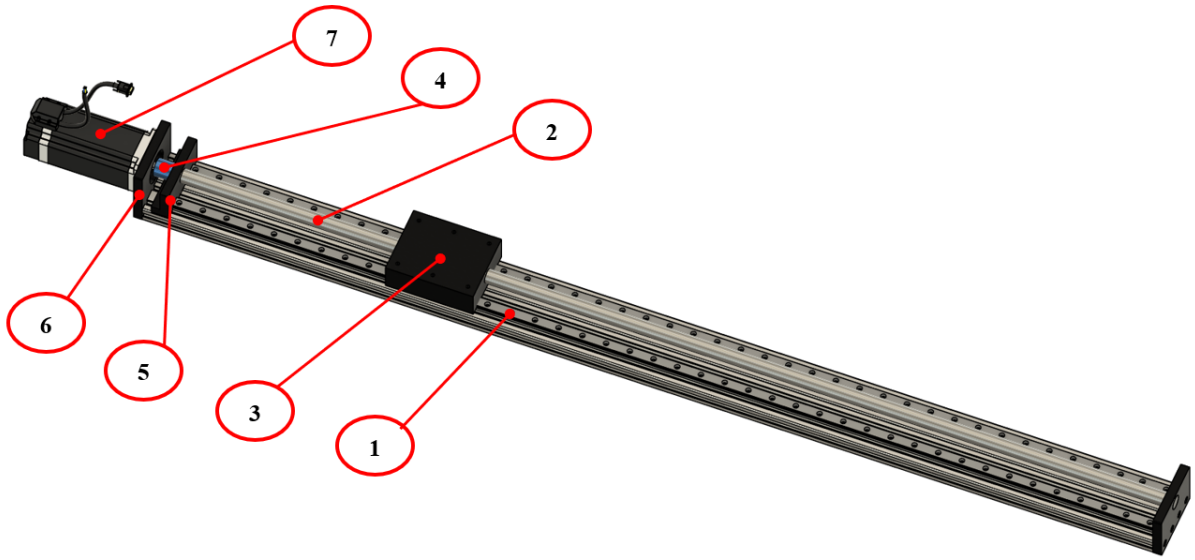


Figure 10. Ball Screw Linear Actuator where part classification is found in Table 4

Table 4: Parts List for Ball Screw Linear Actuator

Label #	**Part Name**	**Qty**	**Part Description**
1	Aluminum Profile Including Linear Rails	1	Structural frame of the actuator, providing mounting points and linear guidance for the carriage
2	Ball Screw Rod	1	Translates rotational motion into linear motion; 5 mm lead (pitch)
3	Carriage Plate	1	Plate that connects to the main robotic arm plate via a mounting bracket
4	Coupling	1	Connects the motor shaft to the ball screw, compensating for slight misalignments
5	Locking Support Seat	1	Provides rigid support for the ball screw, ensuring stability and reducing deflection
6	NEMA 23 Motor Bracket	1	Securely mounts the motor to the actuator frame
7	NEMA 23 Stepper Motor	1	Motor responsible for driving the ball screw mechanism

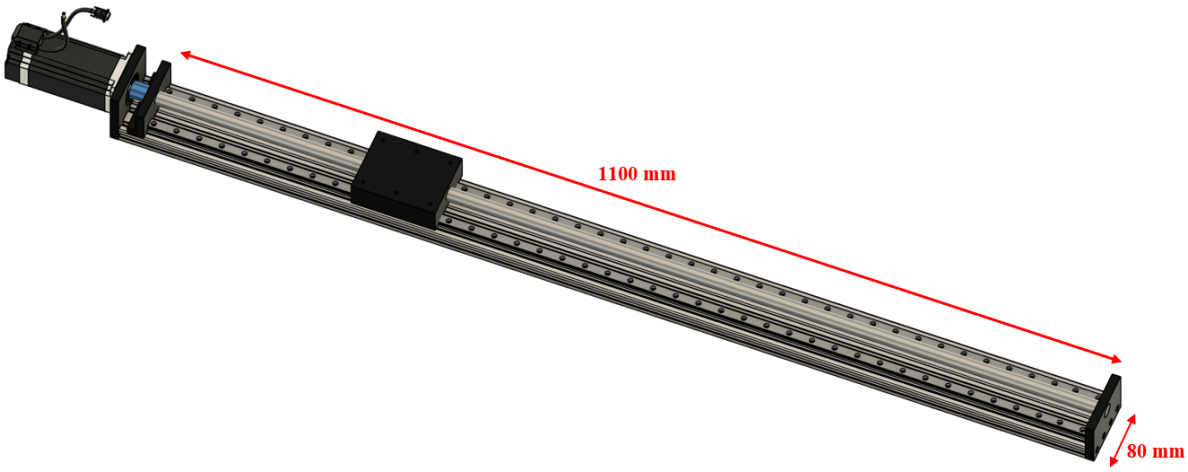


Figure 11. Dimensions of Ball Screw Linear Actuator.

Key aspects of the ball screw linear actuator [3]:

- Maximum linear speed is 0.3 m/s
- Backlash is 0.05 mm (Max backlash for C7 precision grade ball screw) [4]
- 5 mm lead (pitch)

Since the ball screw for this linear actuator (Screw Model: SFU2005) is single-started, the lead and pitch values are the same.

4.4 Cloud Server

The robotic dataset-generation system designed by Fabri Sciences integrates advanced subsystems to coordinate robotic motion, image capture, and backend data management. This architecture involves precise communication between the ROS2 robotic arm, an iPhone-based imaging application, and a backend server that manages state synchronization, high-bandwidth data transfers, and metadata storage. Specifically, the iOS application leverages Apple's AVFoundation for efficient RGB-depth image capture, coordinated in real-time with robotic arm movements using a custom capture queue.

Communication between subsystems employs WebSockets for real-time commands and RESTful APIs for reliable bulk data transfer. Additionally, the infrastructure is designed to scale using cloud services such as AWS EC2, S3, and RDS to support future multi-robot deployments. Comprehensive technical details regarding subsystem integration, image acquisition techniques, network architecture, and cloud scalability strategies are elaborated in Appendix D.

4.5 Robotic Operating System (ROS)

The prototype integrates the Robot Operating System (ROS) as the core communication layer between software and hardware, enabling precise, coordinated control of the robotic arm. A full simulation envi-

ronment—featuring MoveIt 2 for motion planning, Gazebo for physics, and Rviz for visualization—was used to validate motion sequences before hardware deployment, minimizing risk and development time. Once verified, the control stack was transitioned to the physical system with minimal adjustments. Further details on ROS integration, and kinematic modeling are available in Appendix E.

4.6 Computer Vision

The prototype features a custom computer vision pipeline that accurately determines the spatial relationship between the robotic arm and protein slides using ArUco fiducial markers and refined depth map generation. Initial tests with the iPhone's onboard depth camera yielded poor results, but the augmented system significantly improved spatial accuracy and consistency under varying conditions. By directly measuring translation and rotation vectors in the image frame, the system compensates for mechanical imperfections such as backlash or misalignment, enabling precise robotic operation. Additional technical details are available in Appendix F.

5 Critical Design Analysis

An overview of the critical design analysis is shown in Figure 12 with further explanation divided into the two high-level design categories.

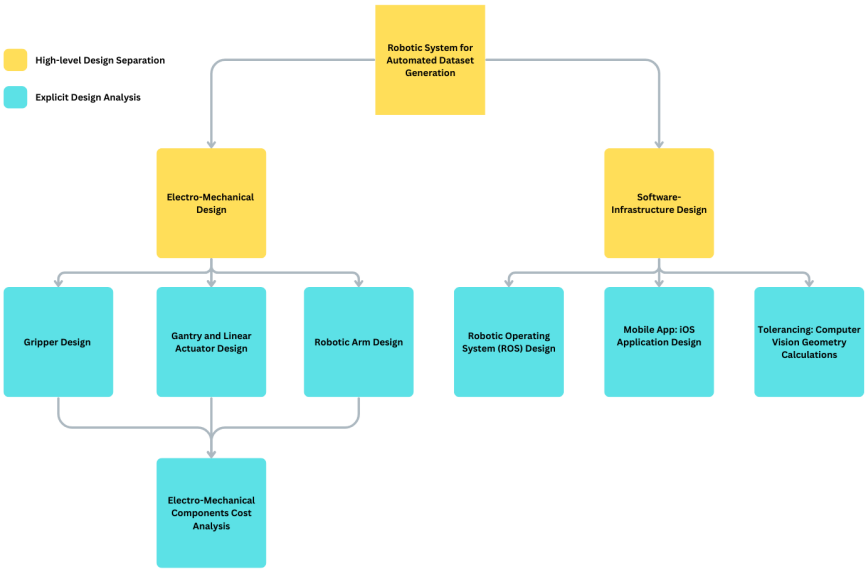


Figure 12. Critical Design Analysis Flowchart

5.1 Electro-Mechanical Design

This section presents the physical components of Fabri Sciences Inc.'s solution, with focused analyses of the gripper, gantry and linear actuator system, and robotic arm.

5.1.1 Gripper Design

The gripper utilizes a vacuum pad mechanism to securely hold items. Calculations were performed to determine the theoretical maximum holding force of the pad. Under ideal conditions, assuming a perfect seal and full surface contact, the gripper is capable of exerting approximately 57.45 N, equivalent to 12.92 lbs of holding force.

However, the manufacturer specifies a conservative working load of 2.82 lbs at the operating vacuum level of 24 in Hg, accounting for real-world factors such as:

- Minor leaks due to imperfect sealing,
- Material deformation under load,
- Safety factors and structural limitations of the vacuum pad.

While the vacuum pump is capable of achieving an ultimate vacuum of 2×10^{-2} Pa with a flow rate of 9.6 CFM, the practical performance is limited by these factors. For further details on the calculation process and assumptions, refer to Appendix B.1.

5.1.2 Gantry and Linear Actuator Design

The gantry system uses a ball screw linear actuator driven by a NEMA 23 stepper motor with a rated torque of 3.0 Nm. Calculations were performed to verify whether the motor provides sufficient torque to move the combined weight of the robotic arm and its aluminum mounting plate.

Under worst-case assumptions, where the full mass of both the robotic arm (12.25 kg) and the aluminum plate (5.36 kg) is applied as load, the total force due to gravity is approximately 172.79 N. Applying this to the torque formula for a ball screw actuator and accounting for system efficiency, the required torque is:

$$T \approx 0.155\,\text{Nm}$$

This is significantly lower than the motor's maximum output of 3.0 Nm, confirming that the selected stepper motor has ample capacity for the application. For detailed step-by-step calculations and assumptions, refer to Appendix B.3.

5.1.3 Robotic Arm Design

The robotic arm used in this design is the AR4-MK3. Two main analyses were conducted: forward kinematics and inverse kinematics, to ensure accurate positioning of the end-effector in three-dimensional space.

The forward kinematics analysis derived the transformation matrices for each joint using the Denavit-Hartenberg (D-H) method. Multiplying these matrices yielded the end-effector's position as:

$$(x, y, z) = (327.85,\ -52.83,\ -0.36)\ \text{mm}$$

and its orientation as:

$$(\phi,\ \theta,\ \psi) = (1.57,\ 0.0032,\ -1.572)\ \text{radians}$$

These results confirm the correct geometric configuration of the arm and its capability to reach the desired workspace.

The inverse kinematics analysis applied Newton's method to solve for the joint angles that achieve a target end-effector position of:

(200, 300, 500) mm

Starting from an initial guess, the iterative process successfully reduced the positional error, with the first update yielding a new joint configuration that significantly improves alignment with the target. Details of this iterative solution process are documented in Appendix B.2.

These analyses validate that the AR4-MK3 arm can achieve the required range of motion and positional accuracy for the application.

5.2 Software-Infrastructure Design

This section outlines the software infrastructure of Fabri Sciences Inc.'s solution, with a focus on ROS.

5.2.1 Robotic Operating System (ROS) Design

The software system is built on the Robot Operating System (ROS), which acts as the main communication hub between the user, microcontrollers, and robotic hardware. ROS takes user commands—like moving the robotic arm to a certain position—and turns them into clear instructions for the motors and sensors. These instructions are then carried out by the microcontrollers to control the physical robot. A full explanation of this setup is provided in Appendix E.

The software is organized into different parts, each with its own job:

- **`annin_ar4_description`:** Describes the robot's shape and components for computer simulations.
- **`annin_ar4_driver`:** Connects ROS to the real robot hardware.
- **`annin_ar4_firmware`:** Runs the basic code that controls the motors and sensors.
- **`annin_ar4_moveit_config`:** Plans robot movements and shows them in 3D models.
- **`annin_ar4_gazebo`:** Simulates how the robot moves in a virtual environment.

The development started in the simulation environment to test everything before using the real robot. A custom ROS program sends target positions to the robot, and the MoveIt 2 planner figures out how to move the robotic arm safely to these positions. After testing in simulation, this setup will be transferred to control the real hardware.

The simulation tools, MoveIt 2, Gazebo, and Rviz, help make sure the robot works as expected:

- **MoveIt 2** calculates how the robot should move its joints to reach specific points in space.
- **Gazebo** simulates real-world physics like gravity, friction, and obstacles.
- **Rviz** provides an easy-to-understand visual display of how the robot will move.

Unfortunately, establishing a reliable connection between ROS and the robotic hardware via serial communication proved to be problematic during testing. As a result, the robotic arm could not be directly controlled through the ROS interface as originally intended. To proceed with physical testing, manual control was instead carried out using the graphical user interface (GUI) provided by the manufacturer. This GUI works in conjunction with the default control software developed by Annin Robotics, which was flashed onto the robot to enable basic motion and functionality.

5.2.2 Mobile App IOS Application Design

The iOS mobile application plays a critical role in the system by capturing and transmitting high-quality images while the robotic arm moves to various positions. The app works alongside the ROS-based robotic system and backend server to ensure that image capture is synchronized with the robot's movements. A high-level overview of the system architecture is shown in Figure D.4, with technical details available in Appendix D.

The system is made up of four main parts:

- **Coordinator server:** Manages communication and keeps track of the state of the robot and mobile app.
- **iOS application:** Captures RGB images and depth maps, and sends them to the server.
- **ROS2 robotic system:** Controls the robot's movements and provides real-time position updates.
- **Database and storage:** Saves image files and keeps records of each capture session.

The iOS app uses Apple's AVFoundation framework to capture both high-resolution images and depth data. It operates a capture pipeline that processes images one at a time to manage memory efficiently. Using a custom queue system, the app ensures images are captured with the correct exposure, focus, and depth accuracy, without overloading the device's memory. Captured data is temporarily stored on the device and then transmitted to the backend server.

Data Transmission Image files can be large, so the app compresses them using JPEG format to balance quality and size. Depth maps are saved in a 32-bit floating-point format to maintain accuracy while reducing file size.

For communication, the app uses two methods:

- **WebSockets:** For fast, real-time commands and updates between the app, robot, and server.
- **REST API:** For uploading larger data like images and saving capture session details.

This approach ensures reliable communication while keeping the system fast and efficient. The WebSocket connection allows the robot and app to stay in sync, while the REST API handles the larger, slower data uploads.

System Integration The mobile app uses a WebSocket client to stay connected with the server and Apple's URLSession for uploading images. It is designed to handle network problems by retrying failed uploads and making sure no data is lost if there are interruptions.

On the robot's side, a ROS2 node uses WebSockets to receive movement commands and send updates, keeping the robot's movements aligned with the image capture process.

Infrastructure and Scalability As the system grows, it needs to handle more robots and larger amounts of image data. To manage this, the backend uses a combination of local computing and cloud services:

- **Amazon EC2:** Runs the WebSocket server and API.
- **Amazon S3:** Stores images and depth data.
- **Amazon RDS (PostgreSQL):** Keeps records of each scan.

For early testing, the server is run locally to avoid cloud costs. However, as the system scales, cloud-based solutions with load balancing and multiple server instances will be used to handle many robots at once.

5.2.3 Tolerancing Computer Vision Geometry Calculations

The computer vision system is designed to improve the accuracy of image capture by correcting for small mechanical errors in the robot's assembly and movement. Rather than relying only on perfectly built mechanical parts, the system uses advanced image processing to measure and correct the robot's position and orientation in real time. An example of the setup, showing the camera mount and visual markers, is shown in Figure 13.

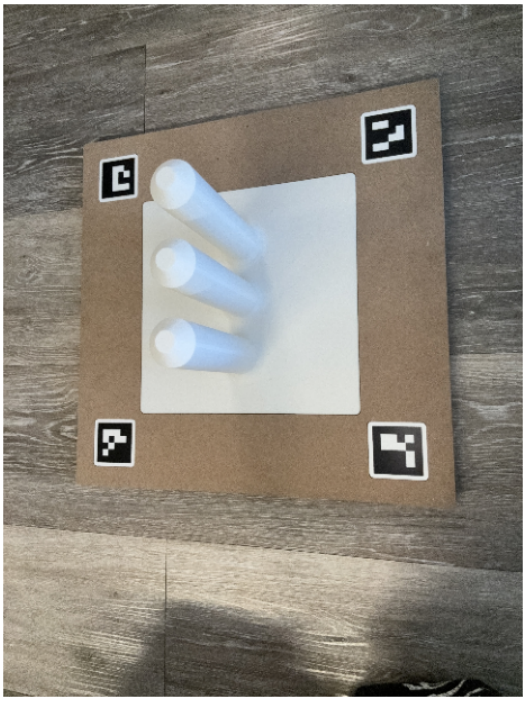


Figure 13. Image captured by the iPhone, showing the 3D-printed mount and ArUco marker placement.

Initial depth maps from the iPhone camera were not accurate enough for reliable use (Figure 14). To improve this, the team added ArUco markers — simple printed patterns that help the camera precisely understand its position in space. By processing images of these markers, the system generates much clearer and more accurate depth maps (Figure 15).

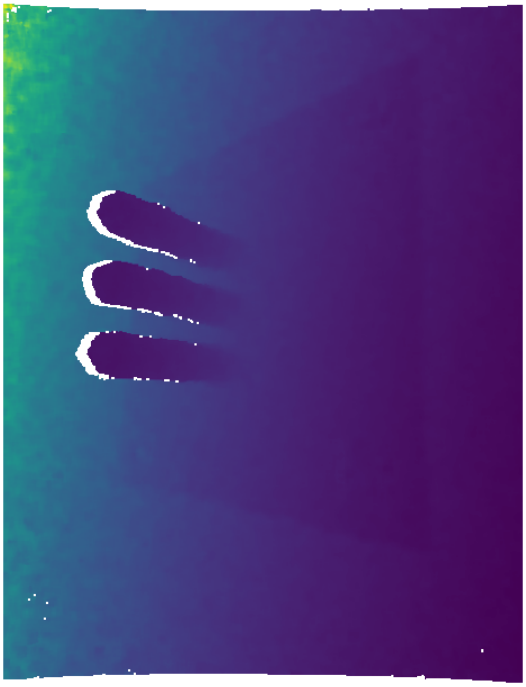


Figure 14. Initial depth map captured by the iPhone (insufficient quality).

Figure 15. Refined depth map generated via computer vision techniques.

The computer vision software works by taking pairs of RGB images and depth maps, cleaning up any noise, and reconstructing an accurate 3D model of the scene. This process allows the system to:

- Measure and correct any small shifts or rotations of the camera.
- Compensate for mechanical issues like gear backlash and assembly misalignment.
- Maintain precise control without needing expensive high-precision mechanical parts.

Instead of relying on expensive components, the team chose to use software to manage mechanical tolerances. The camera's position relative to the sample slide is calculated directly from the images, independent of mechanical imperfections. With over 12 million pixels in the iPhone camera, the system achieves very high accuracy — potentially at the micron level — making it much more efficient than manually tuning mechanical parts.

The full source code for the computer vision system is available on **GitHub**, and a detailed video walkthrough is provided on **Loom**. A full analysis of the computer vision system can be found in Appendix F.

6 Manufacturing and Cost Analysis

Gantry Frame and Rotary Axis Integration (30 hours)

- Cut stock aluminum extrusions (4040, 2040, 2020 profiles) to length using a miter saw provided by the client.
- Assemble the base frame and vertical supports with fasteners and brackets.
- Install the HGR20 linear guide rails with precise drilling/tapping for alignment.
- Install the mounting plate, provided by the client, onto the linear rail carriages to serve as the base for the robotic arm.
- Mount and align the ball screw linear actuator, and integrate the NEMA 23 stepper motor for carriage motion.
- Mount the rotary axis to the gantry.
- Install two NEMA 23 motors and couple them to the rotary axis.
- Wire and test the rotary motion for smooth rotation in two degrees of freedom.

Robotic Arm (100 hours)

- Follow official AR4-MK3 assembly instructions [1] using kits: Aluminum Parts, Hardware, Electrical, and Stepper Motor kits.
- Build each joint (base to wrist), install motors, route and connect wiring, and perform full mechanical and electrical integration.
- Calibrate joints, ensuring smooth operation across all axes.

Vacuum Gripper Assembly (2 hours)

- Mount the gripper bracket to the robot's wrist.
- Install the vacuum pump and route air tubing along the robot's structure.
- Wire the pump and solenoid valve to the control system, then test suction and release functionality.

Note that no manufacturing costs were determined as the process was completed using client-sponsored tools.

Table 5 shows the complete cost breakdown for the entire assembly including the robotic arm, gantry, and gripper subsystems.

Table 5: Detailed Cost Analysis for the Complete Assembly

Part #	Part Name	Model No.	Unit Price (CAD)	QTY	Total Cost (CAD)	Source
Gripper Design: Vacuum Suction						
VAC-000	Complete Vacuum Assembly	5083N145	77.31	1	77.31	Vacuum Assembly Link
VAC-001	Vacuum Pad	-	-	1	-	-
VAC-002	Vacuum Pad Attachment Piece	-	-	1	-	-
VAC-003	Base Plate for Attachment	-	-	1	-	-
VAC-004	M6 x 1mm Attachment Piece	-	-	1	-	-
VAC-005	6mm Barbed Side Air Connection	-	-	1	-	-
VAC-006	Air Tubes Set	B07T14L3CG	17.33	1	17.33	Air Tubes Link
VAC-007	Vaccuum Pump	B09KBJS36B	233.99	1	233.99	Vacuum Pump Link
				Subtotal	**328.63**	
Main Gantry Structure						
GS-001	4040 Aluminum Extrusion	ADV21124040	70.00 per 2 m	4	279.99	4040 Extrusion Link
GS-002	2040 Aluminum Extrusion	V2040	47.40 per 2 m	5	236.99	2040 Extrusion Link
GS-003	2040 Aluminum Extrusion	V2040	24.00 per 1 m	5	119.99	2040 Extrusion Link
GS-004	2020 Aluminum Extrusion	B09KL9NDRR	12.75 per 0.8 m	4	50.99	2020 Extrusion Link
GS-005	HGR20 Rail Kit	B09N6ZB7NS	480.69	1	480.69	Rail Kit Link
GS-006	Rotary Axis Assembly	4th 5th Axis	473.99	1	473.99	Rotary Axis Link
GS-007	Aluminum Brackets	B0CLXVFTWS	1.12	48	53.98	Bracket Link
GS-008	Aluminum Extrusion Fasteners	B0B289CSZH	2.69	8	21.59	Fastener Link

GS-009	NEMA 23 Stepper Motor	23HS45-4204D-E1000	36.02	3	108.06	NEMA Motor Link
				Subtotal	**1,826.27**	
Ball Screw Linear Actuator						
BS-001	Ball Screw Linear Actuator	SK-KGX150	684.16	1	684.16	Ball Screw Link
BS-002	NEMA 23 Stepper Motor	23HS45-4204D-E1000	36.02	1	36.02	NEMA Motor Link
				Subtotal	**720.18**	
AR4-MK3 Robotic Arm						
AR4-001	AR4 MK3 Aluminum Parts Kit	-	860.56	1	860.56	AR4 Aluminum Parts Kit Link
AR4-002	AR4 MK3 Hardware Kit	-	558.86	1	558.86	AR4 Hardware Kit Link
AR4-003	AR4 MK3 Primary Electrical Parts Kit	-	300.27	1	300.27	AR4 Electrical Kit Link
AR4-004	AR4 MK3 Stepper Motor Kit	AR4-MK3	1048.50	1	1048.50	AR4 Stepper Motor Kit Link
				Subtotal	**2,768.19**	
				Grand Total	**5,643.27**	

7 Product Feasibility Analysis

While this report in its entirety is unlikely to be used directly by the client, the assembly and operation instructions included remain highly valuable for future implementation and support. The design itself is fully automated and modular, requiring minimal manual intervention once assembled. This aligns directly with the project's original goal—to create a scalable, easy-to-operate system that reduces reliance on specialized labour. The result is a highly usable and client-ready product, even if the full report primarily serves documentation and evaluation purposes.

8 Design Compliance Matrix

The Design Compliance matrix shown below showcases all the major design considerations and decisions that are required for this project. The importance values for each specification are explained in Table A.1 in Appendix A. These values were determined based on several factors, with primary emphasis on the client's expectations, the team's resource allocation, and compatibility with design components. For the finalized design and prototype, all criteria with a value of 5 or 4 have been met in order for the prototype to be functional. In terms of the client, 5 to 4 were must haves and 3 to 1 were considered nice to have.

Specific safety and regulatory requirements are discussed in the compliance matrix along with compliance. Further description of these requirements can be found in Appendix I.

Throughout the design phase and, importantly, the prototype building phase, it became vitally important to distinguish these "Needs to have" from "Nice to haves." Thought and analysis had to be performed at client meetings and design meetings as to what would be critical to the project's main mission. After each discussion client approval also had to be obtained to ensure continuing satisfaction with the progress and design direction. Below in Table 6 is our client meeting checklist along with approval confirmation.

Table 6: Client Approval Checklist

Date	Client Approval	Description
2025/02/01	Dr. Todd McMullen	Phase 1 Overview and preliminary design objectives discussed.
2025/02/24	Dr. Todd McMullen	Prototype development update and design choices finalized and confirmed.
2025/03/05	Dr. Todd McMullen	Phase 2 Overview and prototype progress report.
2025/04/01	Dr. Todd McMullen	Phase 3 Overview, Prototype completion and showcase, minor client alterations.

Item	Specification	Design Control	Comments	Importance (1-5) 1-low 5-high	End of Phase 2	End of Phase 3	Design Compliance
Project Management							
1.1	Schedule	Dr. Tetsu	All deliverables submitted on time in accordance the course schedule. Deadline for Phase 3 Report is April. 9	5	5	5	Deadlines were all met
1.2	Manufacturability	Client	Manufacturing will be completed using various methods to achieve the lowest cost possible. Most likely will use various OEM components to make this possible.	4	4	4	All manufacturing and assembly were done in house including installation of outsourced components
1.3	Production Volume	Client	Single prototype is to be designed and manufactured as proof of concept	3	4	5	One functioning prototype was completed
Physical Dimension and Design							
2.1	Platform Size	Client	Platform that's stable and can hold the robotic arm and controller mechanisms. Max floor footprint of 250cm x 85cm and comfortable work height of 80cm	4	3	4	Final dimensions: LxWxH 2224mmx820mmx1230mm
2.2	Load limit	Client	A Load limits of 2 kg must be met for the chosen robotic arm system to ensure gripper mechanism and capture device are supported adequately.	4	5	5	Maximum Load limits calculated was : 1.9 kg It fell just short of our specification but still functions just with a slightly reduced factor of safety.
2.3	Material / Colour	Team	Materials will largely be comprised of stainless steel and aluminum for durability and ease of cleaning	3	4	3	Majority of materials was comprised of aluminum and stainless steel
2.4	User Interface	Team	User interface will be designed through an IOS/Android App for end user access for image capture and data transmission	3	4	4	App was completed and passed the testing phase.

Functional							
3.1	Control Mechanism	Client	Control of capture device will be facilitated though open source software in a local or cloud server. Via a microcontroller control of sensors is needed to execute desired functions.	4	4	4	Control of capture device was integrated into the IOS application and passed the testing phase.
3.2	Drive Mechanism	Client	Drive mechanisms must be powered and compatible with user controlled and control mechanism.	3	4	5	A ball screw linear actuator was selected along with stepper motors
3.3	Reachable Workspace	Client	Robotic arm with gripper should reach all points inside a 250cm x 85cm x 50 cm envelope.	3	3	3	Robotic arm assembly with the drive mechanism achieved 4 degrees of freedom allowing it to reach desired envelope.
3.4	Robotic Arm Integration	Team	Integration capability and simplicity with the chosen control software ROS2	5	5	5	ROS2 integrations was completed and functioning
3.5	Gripper Mechanism	Team	Needs to support and securely hold various phones and tablets max size: 200xmm250mm Min size: 70mm*130mm	4	4	5	Vacuum Gripper mechanism was designed, manufactured, and met or exceeded design requirements.
3.6	Gripper Inertia	Team	Gripper design must be constructed in a way to reduce or minimize the robotic arms inertia during operation. This is based on total mass of the gripper system.	4	3	3	Vacuum Gripper had the lowest weight total compared to all other considered gripper mechanisms
3.7	Computer vision implementation	Team	Computer vision tracking using ArUco squares and matrix calculation. Precision must be high enough to discern visible changes in standard blood/urine test slides.	5	4	5	Computer vision software and tracking systems are fully operations, integrated and performing as expected.
3.8	Operating conditions	Client	Intended operation range of 10-40 C	3	3	3	Operation was designed for 20 C
3.9	Life-cycle	Client	Components should have over a 10-year life cycle	3	3	3	Quality parts were sourced to ensure life-cycle requirements can be met.

3.10	Linear actuator load capacity	Client	Linear actuator system has to have adequate power to move loads along the gantry system.	4	4	5	Torque of 3.0 Nm was achieved with the selected system and passed our load testing. Maximum load requirement possible being 0.155 Nm under extreme operating conditions
3.11	Linear actuator Precision	Client	Linear actuator system must have precise movement control as to not interfere with photo capture calibration system.	5	4	4	Imprecisions in the movement control systems was fixed using post processing of the capture system.
3.12	Linear actuator Speed	Client	The linear actuator system must be capable of achieving a speed of at least 0.1 m/s to alter the position of the robotic arm during data generation.	2	2	2	Movement speed of 0.3m/s was achieved.
Safety							
4.1	Physical Guards	CSA	See section 9.2.2.4 in Appendix J	5	5	5	Physical Guards are being installed currently to prevent physical interaction with moving parts during operation.
4.2	Power Isolation	OHS	See section 212(1) in Appendix J	5	5	5	Power toggle switches and fuses are installed along with a emergency power cutoff button easily accessible.
4.3	Speed Limits	CSA	See section 9.2.2.6 in Appendix J	5	5	5	Speed limits of moving components have been limited using the onboard ROS software integration.
4.4	Secure Installation	OHS	See section 209.1(2) in Appendix J	5	5	5	Platform has been securely fastened to the floor via the gantry system using bolts and rubber feet.
MISC.							
5.1	Budget	Client	Allocations for components, assembly, software, and development hours. Total	4	4	5	Total project cost of 56,250$ which was under our project budget even with a longer than

			project development and prototype cost must be under 60,000$				expected coding and assembly time.
5.2	Testing	Team	Testing must be completed for both physical components and software/app controllers to ensure client goals are met.	4	4	5	Testing was successfully completed and protype is functioning as expected within our desired operating conditions.
5.3	Maintenance	Team	Components should be easily serviceable without the need for advanced tools or training.	1	2	2	All parts have been serviced, and a maintenance schedule has been created.
5.4	Environment	Team	Components should be sourced to have minimal impact to the environment	1	2	2	Majority of steel and aluminum components making up the majority of the prototype can be recycled or reused.

9 Project Management

Fabri Sciences Inc. has consistently tracked engineering labour hours throughout all phases of the robotic system development. Figures 16 and 17 provide comparisons between initial estimates and actual labour expenditures. Detailed scheduling information and updated Gantt charts for Phase III are presented in Appendix K.

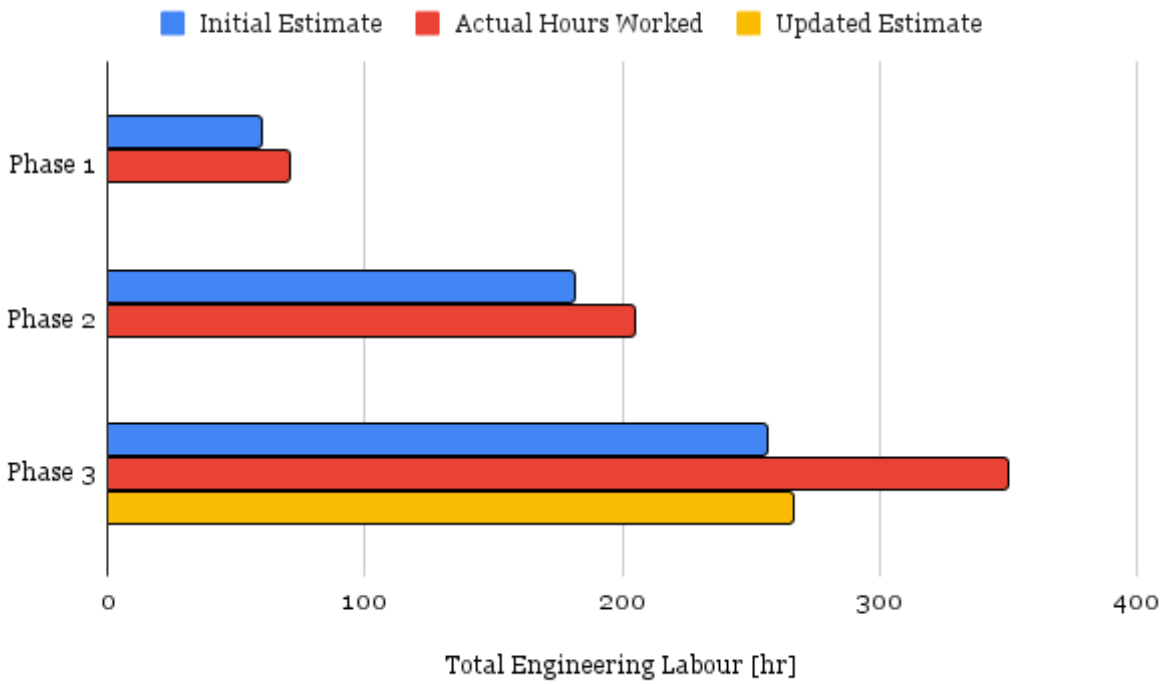


Figure 16. Breakdown of total engineering labour hours per phase.

The Phase III labour hours significantly exceeded the initial estimate of 541 hours, reaching a total of 625 hours, representing an 84-hour increase (approximately 16% variance). This increase was primarily due to unforeseen challenges in software development and robotic system assembly. The most substantial factor was the discovery of critical faults in the robotic arm source code provided by Annin Robotics, necessitating a fully custom software solution. Connor Poveledo led the extensive development and debugging of this new software, significantly contributing to the additional hours.

Furthermore, iterative adjustments and refinements during physical prototype assembly required additional time beyond initial expectations. Integration testing, mechanical troubleshooting, and design revisions also contributed notably to the expanded labour commitment.

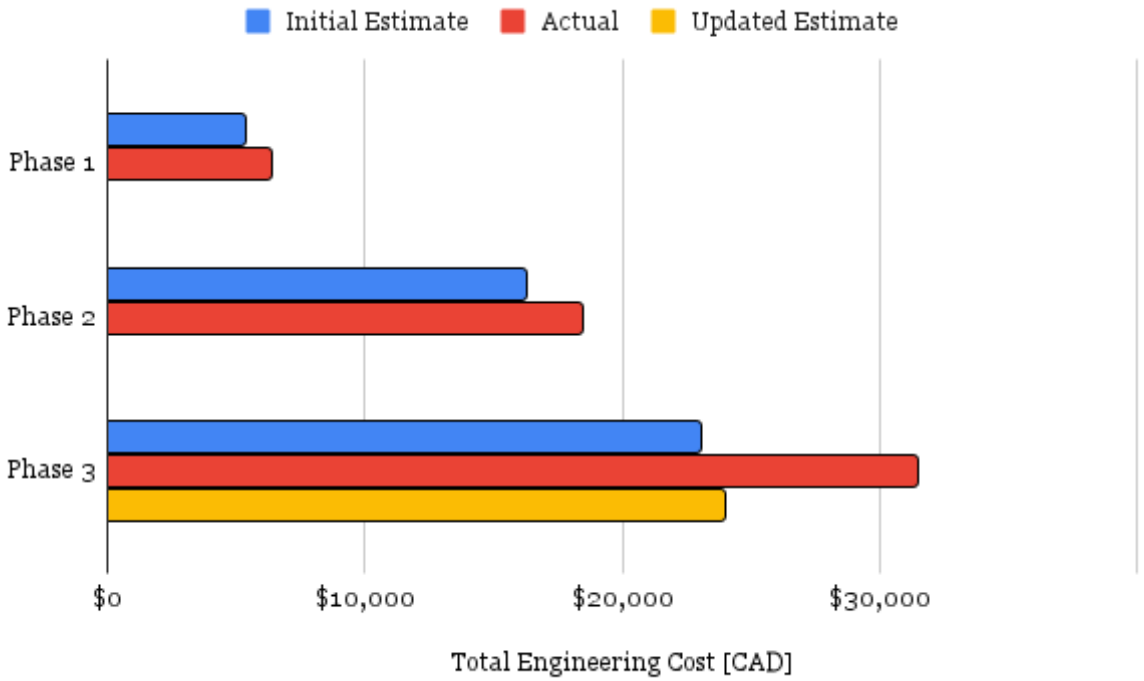


Figure 17. Breakdown of total engineering labour costs per phase.

As a direct consequence of these additional hours, the manpower costs also rose from the original Phase III estimate of $48,690 to a final total of $56,250. These adjustments accurately reflect the extensive efforts required to overcome the unforeseen challenges encountered in Phase III.

10 Conclusion

The robotic system developed by Fabri Sciences Inc. successfully meets and surpasses the client's specifications, demonstrating exceptional capability as a fully automated, high-precision robotic solution. Extensive testing of our prototype confirmed that the data produced by the system achieves accuracy within microns, ensuring clinical-grade dataset reproducibility essential for reliable medical diagnostics. Importantly, the total manufacturing cost was effectively managed, with the final prototype constructed for under $6,000 CAD, significantly below the original budget, and with inherent scalability to expand operations seamlessly.

In practical terms, this design has dramatically improved productivity by replacing manual processes—previously limited to two slides per day by a full-time worker—with automation capable of completing over a decade's worth of manual workload within half a day. This directly translates to eliminating annual labour expenses of approximately $75,000, generating long-term savings exceeding $700,000 and creating substantial economic and operational efficiencies. The integration of advanced computer vision, custom-developed robotic control software, and reliable ROS2-driven automation further ensures the system's precision, adaptability, and robustness. Ultimately, this innovative and scalable design positions Kondor Devices Inc. to substantially advance their diagnostic capabilities, providing new avenues for rapid, accurate, and cost-effective medical analysis.

To emphasize the success and feasibility of this system, Figure 18 showcases the completed real-life prototype in operation—highlighting that this is not just a theoretical concept, but a fully realized, functioning solution.

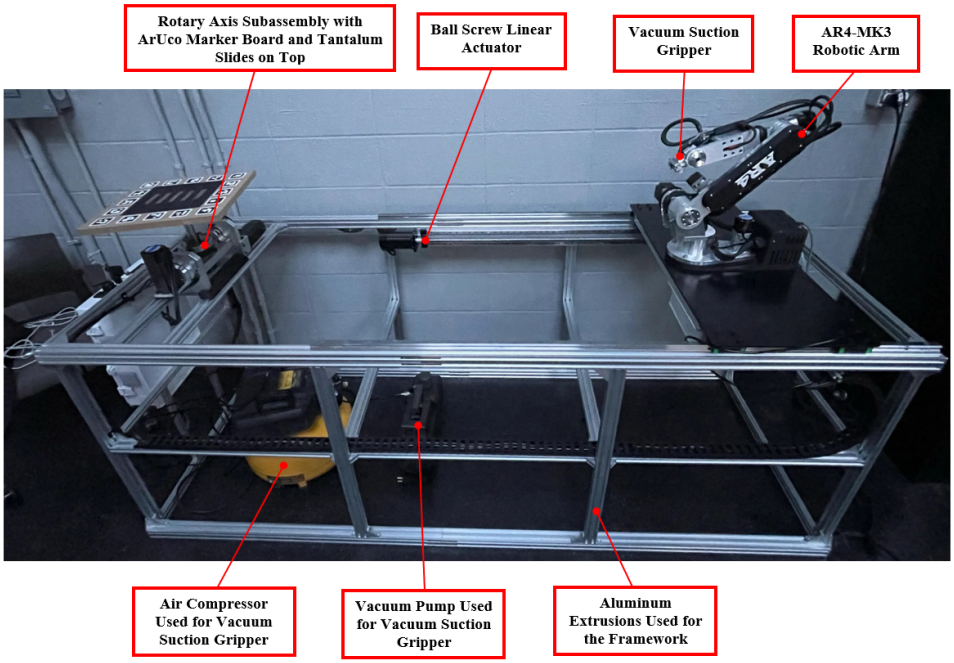


Figure 18. Final assembled prototype of the automated robotic dataset generation system.

References

[1] C. Annin, *AR4-MK3 Robot Manual FREE DESIGN 6 AXIS ROBOT Version 1.3*, 2024. [Online]. Available: https://orcaslicer.cn/upload/news/file/20241230/20241230151926_37001.pdf

[2] OMC-StepperOnline, "Nema 23 Closed-Loop Stepper Motor 3.0Nm 424oz.in Encoder 1000PPR 4000CPR (23HS45-4204D-E1000)," 2025. [Online]. Available: https://www.omc-stepperonline.com/nema-23-closed-loop-stepper-motor-3-0nm-424oz-in-encoder-1000ppr-4000cpr-23hs45-4204d-e1000

[3] AliExpress, "Ball screw linear rail module on aliexpress," 2025. [Online]. Available: https://www.aliexpress.com/item/1005008337723764.html

[4] H. Corporation, "Ballscrew catalog," n.d. [Online]. Available: https://www.hiwin.com/wp-content/uploads/Ballscrew-Catalog.pdf

[5] U. of Alberta, *University of Alberta Robotics Course Materials*, 2025, accessed: February 24, 2025. [Online]. Available: https://ugweb.cs.ualberta.ca/~vis/courses/robotics/

[6] Granta Automation. (n.d.) Electrical linear actuator calculations. [Online]. Available: https://www.granta-automation.co.uk/resources/technical-information/technical-resources-electrical-inear-actuator-calculations

Appendix A: Updated Design Specification Matrix

The specification matrix shown in Table B.1 showcases all the major design considerations and decisions that are required for this project. The importance values for each specification are explained in Table A.1. These values were determined based on several factors, with primary emphasis on the client's expectations, the team's resource allocation, and compatibility with design components. For the finalized design and prototype, all criteria with a value of 5 or 4 has to be met in order for the prototype to be functional. In terms of the client, 5 to 4 were must haves and 3 to 1 were considered nice to have.

Modifications to the matrix are displayed in the changes column, and the importance values have been updated from phase 2.

Table A.1: Design Specifications Criteria Chart

Importance Value	Criteria	Description
5	High Priority	Failure to follow renders the project infeasible
4	Moderate Priority	Significant impact on performance and is key to project success
3	Low Priority	Improves performance but is not vital for the core function
2	Optional	Minimal impact adding non-essential enhancements
1	Unnecessary	Negligible impact on project outcome

See updated matrix on the following pages.

Item	Specification	Design Control	Comments	Importance (1-5) 1-low 5-high	Changes
Project Management					
1.1	Schedule	Dr. Tetsu	All deliverables submitted on time in accordance the course schedule. Deadline for Phase 1 Report is Feb. 9	5	
1.2	Manufacturability	Client	Manufacturing will be completed using various methods to achieve the lowest cost possible. Most likely will use various OEM components to make this possible.	4	
1.3	Production Volume	Client	Single prototype is to be designed and manufactured as proof of concept	5	
Physical Dimension and Design					
2.1	Platform Size	Client	Platform that's stable and can hold the robotic arm and controller mechanisms. Max floor footprint of 250cm x 85cm and comfortable work height of 80cm	4	
2.2	Load limit	Client	A Load limits of 2 kg must be met for the chosen robotic arm system to ensure gripper mechanism and capture device are supported adequately.	5	Added clarity to design criteria.
2.3	Material / Colour	Team	Materials will largely be comprised of stainless steel and aluminum for durability and ease of cleaning	3	
2.4	User Interface	Team	User interface will be designed through an IOS/Android App for end user access for image capture and data transmission	4	.
Functional					
3.1	Control Mechanism	Client	Control of capture device will be facilitated though open source software in a local or cloud server. Via a microcontroller control of sensors is needed to execute desired functions.	4	

3.2	Drive Mechanism	Client	Drive mechanisms must be powered and compatible with user controlled and control mechanism.	5	
3.3	Reachable Workspace	Client	Robotic arm with gripper should reach all points inside a 250cm x 85cm x 50 cm envelope.	3	
3.4	Robotic Arm Integration	Team	Integration capability and simplicity with the chosen control software ROS2	5	
3.5	Gripper Mechanism	Team	Needs to support and securely hold various phones and tablets max size: 200xmm250mm Min size: 70mm*130mm	5	
3.6	Gripper Inertia	Team	Gripper design must meet functional requirements but also minimize total weight as much as possible to reduce the effect on the robotic arms inertia during operation.	3	Added clarity to design criteria.
3.7	Computer vision implementation	Team	Computer vision tracking using ArUco squares and matrix calculation. Precision must be high enough to discern visible changes in standard blood/urine test slides.	5	
3.8	Operating conditions	Client	Intended operation range of 10-40 C	3	
3.9	Life-cycle	Client	Components should have over a 10-year life cycle	3	
3.10	Linear actuator load capacity	Client	Linear actuator system has to have adequate power to move loads along the gantry system.	5	
3.11	Linear actuator Precision	Client	Linear actuator system must have precise movement control as to not interfere with photo capture calibration system.	4	
3.12	Linear actuator Speed	Client	The linear actuator system must be capable of achieving a speed of at least 0.1 m/s to alter the position of the robotic arm during data generation	2	
Safety					
4.1	Physical Guards	CSA	See section 9.2.2.4 of Table 1	5	
4.2	Power Isolation	OHS	See section 212(1) of Table 1	5	
4.3	Speed Limits	CSA	See section 9.2.2.6 of Tabe 1	5	
4.4	Secure Installation	OHS	See section 209.1(2) of Table 1	5	
MISC.					

5.1	Budget	Client	Allocations for components, assembly, software, and development hours. Total project development and prototype cost must be under 60,000$	4	
5.2	Testing	Team	Testing must be completed for both physical components and software/app controllers to ensure client goals are met.	4	
5.3	Maintenance	Team	Components should be easily serviceable without the need for advanced tools or training.	2	
5.4	Environment	Team	Components should be sourced to have minimal impact to the environment with sustainability in mind	2	Added clarity to design criteria.

Appendix B: Detailed Calculations and Coding

B.1 Calculation of Maximum Weight Holdable by the Vacuum Pad

Name and Date

Calculation done by Wesley Eze, on 2025-03-06.

Objective

To determine the maximum holding force of the vacuum pad using the pressure differential and the effective contact area.

Assumptions

- The calculation assumes ideal conditions with a perfect seal.
- The entire pad surface is in uniform contact with the load.
- Material deformation and other real-world inefficiencies are ignored in the theoretical computation.

Knowns

- Vacuum level: 24 in Hg.
- Conversion factor: $29.92\,\text{in Hg} = 101325\,\text{Pa}$.
- Pad diameter: 30 mm (therefore, radius $r = 15\,\text{mm} = 0.015\,\text{m}$).
- Vacuum pump specifications: Ultimate vacuum of 2×10^{-2} Pa and a maximum flow rate of 9.6 CFM.

Analysis

The holding force F is given by:

$$F = \Delta P \times A$$

where:

- ΔP is the pressure differential,
- A is the effective contact area of the pad.

Step 1: Pressure Differential The vacuum level is given as 24 in Hg. Converting this to Pascals:

$$\Delta P = 24\,\text{in Hg} \times \left(\frac{101325\,\text{Pa}}{29.92\,\text{in Hg}}\right) \approx 81273\,\text{Pa}$$

Step 2: Contact Area Calculation The pad is circular with a diameter $D = 30\,\text{mm}$. The radius is:

$$r = \frac{D}{2} = \frac{30\,\text{mm}}{2} = 15\,\text{mm} = 0.015\,\text{m}$$

Thus, the area is:

$$A = \pi r^2 = \pi(0.015\,\text{m})^2 \approx 7.07 \times 10^{-4}\,\text{m}^2$$

Step 3: Theoretical Maximum Force Using the computed values, the force due to the vacuum is:

$$F = \Delta P \times A \approx 81273\,\text{Pa} \times 7.07 \times 10^{-4}\,\text{m}^2 \approx 57.45\,\text{N}$$

Step 4: Conversion to Pounds Converting the force from Newtons to pounds (using $1\,\text{lb} \approx 4.44822\,\text{N}$):

$$F_{\text{lbs}} = \frac{57.45\,\text{N}}{4.44822} \approx 12.92\,\text{lbs}$$

Conclusion

The theoretical calculation shows that the vacuum pad could hold approximately 12.92 lbs under ideal conditions. However, the manufacturer's rating of 2.82 lbs at 24 in Hg accounts for several real-world factors:

- Imperfect sealing between the pad and the surface, which can lead to air leakage.
- Material deformation and inherent structural limitations of the pad reducing the effective holding force.
- Safety and design factors that necessitate a lower rated capacity.

Furthermore, even though the vacuum pump can generate an ultimate vacuum of 2×10^{-2} Pa and operates at a maximum flow rate of 9.6 CFM, the overall system performance is ultimately governed by the quality of the effective seal and the structural limitations of the pad itself.

B.2 Forward and Inverse Kinematics Calculations

The following appendix provides a detailed explanation of the forward and inverse kinematics calculations for the Annin AR4 robotic arm [5]. The Annin AR4 robotic arm can be modeled as a 6-degree-of-freedom (6-DOF) manipulator, consisting of six revolute joints arranged as follows:

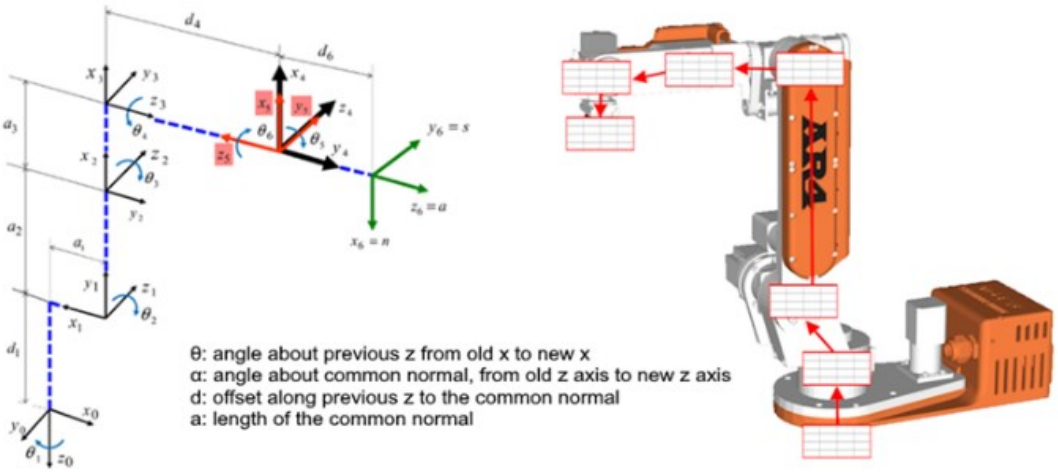


Figure B.1. Annin AR4-MK3 robotic arm with the coordinate frames

B.2.1 Forward Kinematics

Name and Date

Forward kinematics calculations performed by Quang Minh Luu on February 25, 2025.

Objective

To determine the forward kinematics of the Annin AR4 robotic arm using the Denavit-Hartenberg (D-H) method. This involves deriving the transformation matrices for each joint and multiplying them sequentially to obtain the final end-effector coordinate system, which provides both the position and orientation.

Assumptions

- The D-H parameter conventions are strictly followed.
- The coordinate frames are defined as per the standard D-H method.
- Joint movements are ideal and free from mechanical errors or slack.
- The provided D-H parameters and joint configurations accurately represent the Annin AR4-MK3 specifications.

Knowns

- **D-H Parameters:** As summarized in Table B.2:

Table B.2: Summary Table of Annin AR4-MK3 D-H Parameters

Joint	θ (rads)	α (rads)	Offset d (mm)	Linkage length a (mm)
1	0.00	-1.57	169.77	64.20
2	-1.57	0.00	0.00	305.00
3	3.14	1.57	0.00	0.00
4	0.00	-1.57	222.63	0.00
5	1.57	1.57	0.00	0.00
6	0.00	0.00	41.00	0.00

- **D-H Transformation Matrix:** The general D-H transformation matrix is illustrated in Figure B.2:

The Denavit-Hartenberg Matrix

$$\begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{(i-1)} \\ \sin\theta_i\cos\alpha_{(i-1)} & \cos\theta_i\cos\alpha_{(i-1)} & -\sin\alpha_{(i-1)} & -\sin\alpha_{(i-1)}d_i \\ \sin\theta_i\sin\alpha_{(i-1)} & \cos\theta_i\sin\alpha_{(i-1)} & \cos\alpha_{(i-1)} & \cos\alpha_{(i-1)}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure B.2. The Denavit-Hartenberg transformation matrix where i represents the joint number (0 to 6)

- **Transformation Matrices:** The individual transformation matrices derived from the joint configurations are provided.

Analysis

The forward kinematics is computed by sequentially multiplying the transformation matrices:

$$T_{\text{end-effector}} = T_1^0\, T_2^1\, T_3^2\, T_4^3\, T_5^4\, T_6^5$$

with the final transformation matrix expressed as:

$$T_{\text{end-effector}} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The individual transformation matrices are:

$$T_1^0 = \begin{bmatrix} \cos(0) & -\sin(0) & 0 & 64.20 \\ \sin(0)\cos(-\pi/2) & \cos(0)\cos(\pi/2) & -\sin(-\pi/2) & -\sin(-\pi/2) * -169.77 \\ \sin(0)\sin(-\pi/2) & \cos(0)\sin(-\pi/2) & \cos(-\pi/2) & \cos(-\pi/2) * 169.77 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_1^0 = \begin{bmatrix} 1 & 0 & 0 & 64.20 \\ 0 & 0 & 1 & -169.77 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Similarly, the transformation matrix of the other joint configurations can be found:

$$T_2^1 = \begin{bmatrix} 0 & 1 & 0 & 305 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^2 = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 22263 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_5^4 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_6^5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 41 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplying these matrices yields the end-effector transformation matrix:

$$T_{\text{end-effector}} = \begin{bmatrix} -0.0008 & 0.0008 & -1 & 327.85 \\ -1 & -0.0032 & 0.0008 & -52.83 \\ -0.0032 & 1 & 0.0008 & -0.36 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

From this, the Cartesian coordinates of the end-effector are:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 327.85 \\ -52.83 \\ -0.36 \end{bmatrix}$$

And its orientation, expressed in terms of Euler angles, is given by:

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \arctan 2(r_{32}, r_{33}) \\ \arctan 2(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}) \\ \arctan 2(r_{21}, r_{11}) \end{bmatrix} = \begin{bmatrix} \pi/2 \\ 0.0032 \\ -\pi/2 \end{bmatrix}$$

Conclusion

Using the Denavit-Hartenberg method, the forward kinematics of the Annin AR4 robotic arm have been successfully derived. The resulting end-effector transformation matrix indicates a position of $(x, y, z) = (327.85, -52.83, -0.36)$ and an orientation (in Euler angles) of $(\phi, \theta, \psi) = (1.57, 0.0032, -1.572)$. These outcomes provide the necessary mapping from the base frame to the end-effector frame for the given joint configurations.

B.2.2 Inverse Kinematics

Name and Date

Inverse Kinematics Calculations, done by Quang Minh Luu on February 25, 2025.

Objective

To solve the inverse kinematics of the Annin AR4 robotic arm using Newton's method. The aim is to iteratively adjust the joint angles so that the end-effector reaches a desired target position by minimizing the error between the computed forward kinematics and the target coordinates.

Assumptions

- The forward kinematics function $f(r)$ is continuously differentiable with respect to the joint angle vector r.
- The Jacobian matrix $J = \left[\frac{\partial f_i(r)}{\partial r_j}\right]$ can be accurately computed.
- The initial guess for the joint angles is sufficiently close to the true solution to ensure convergence of the iterative process.
- The transformation matrices and joint linkage dimensions are based on the actual Annin AR4-MK3 specifications.

Knowns

- **Forward Kinematics Relation:** $f(r)$ maps the joint angles r to the end-effector coordinates.
- **Iterative Update Equation:** Using Newton's method, the joint angles are updated by solving

$$J\,dr = W - f(r) \quad \text{and} \quad r = r + dr,$$

  where J is the Jacobian matrix, dr is the change in joint angles, and W is the target end-effector coordinate.

- **Target Position:**

$$W_{\text{target}} = \begin{bmatrix} 200 \\ 300 \\ 500 \end{bmatrix}.$$

- **Initial Guess for Joint Angles:**

$$r_0 = \begin{bmatrix} 0.5 \\ -0.5 \\ 0.5 \\ 0.5 \\ -0.5 \\ 0.5 \end{bmatrix} \text{ (radians).}$$

- **Result of Forward Kinematics on r_0:**

$$f(r_0) = \begin{bmatrix} 190 \\ 280 \\ 480 \end{bmatrix}.$$

- **Error between Target and Current Position:**

$$W - f(r_0) = \begin{bmatrix} 10 \\ 20 \\ 20 \end{bmatrix}.$$

- **Jacobian Matrix J:**

$$J = \begin{bmatrix} -100 & -140 & -60 & -10 & 5 & 0 \\ 140 & 110 & 80 & 10 & -5 & 0 \\ 50 & 80 & 100 & 20 & 0 & 5 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

- **Computed Correction:**

$$dr = \begin{bmatrix} 0.4935 \\ -0.2024 \\ 0.0130 \\ -0.3260 \\ -0.0771 \\ -0.1881 \end{bmatrix}.$$

- **Updated Joint Angles:**

$$r_{\text{new}} = r_0 + dr = \begin{bmatrix} 0.9935 \\ -0.7024 \\ 0.5130 \\ -0.1740 \\ -0.5771 \\ -0.3119 \end{bmatrix} \text{ (radians).}$$

Analysis

The inverse kinematics solution is obtained by iteratively applying Newton's method:

1. **Initial Guess:** Start with an initial guess r_0 for the joint angles.
2. **Compute Forward Kinematics:** Evaluate $f(r_0)$ to obtain the current end-effector coordinates.
3. **Error Calculation:** Compute the difference $W - f(r_0)$, which represents the positional error.
4. **Jacobian Evaluation:** Calculate the Jacobian matrix J from $f(r)$ using the specified joint linkage lengths of the ANNIN-AR4-MK3.
5. **Solve for dr:** Solve the linear system

$$J\,dr = W - f(r)$$

   to determine the correction dr for the joint angles.

6. **Update:** Set $r_{\text{new}} = r_0 + dr$. For the first iteration:

$$r_{\text{new}} = \begin{bmatrix} 0.5 \\ -0.5 \\ 0.5 \\ 0.5 \\ -0.5 \\ 0.5 \end{bmatrix} + \begin{bmatrix} 0.4935 \\ -0.2024 \\ 0.0130 \\ -0.3260 \\ -0.0771 \\ -0.1881 \end{bmatrix} = \begin{bmatrix} 0.9935 \\ -0.7024 \\ 0.5130 \\ -0.1740 \\ -0.5771 \\ -0.3119 \end{bmatrix} \text{ (radians).}$$

7. **Iteration:** Repeat the process using the updated joint angles until the error between the computed and target positions is within a predefined tolerance.

Conclusion

The inverse kinematics for the Annin AR4 robotic arm have been successfully approached using Newton's method. The iterative process begins with an initial guess and refines the joint angles by solving

$J\,dr = W - f(r)$ to reduce the positional error. In the first iteration, the computed adjustment dr leads to a new joint angle vector r_{new} that moves the end-effector closer to the target position $(200, 300, 500)$. This method, based on established techniques in robotic kinematics, confirms that the concept design is both feasible and realistic, aligning theoretical analysis with real-world joint linkage specifications.

B.3 Required Torque for Linear Actuator

Calculations done by Eric Petersen on March 10, 2025.

Objective: To confirm that the NEMA 23 stepper motor produces sufficient torque (3.0 Nm) to drive the ball screw linear actuator.

Assumptions:

1. Worst-case scenario: all the weight from the robotic arm and its plate is treated as the force the linear actuator must overcome.
2. In reality, the linear slide blocks (as shown in Figure B.3) will distribute the weight, reducing the required torque. However, since we only need to confirm that the worst-case torque is below 3.0 Nm, precise weight distribution is not critical.

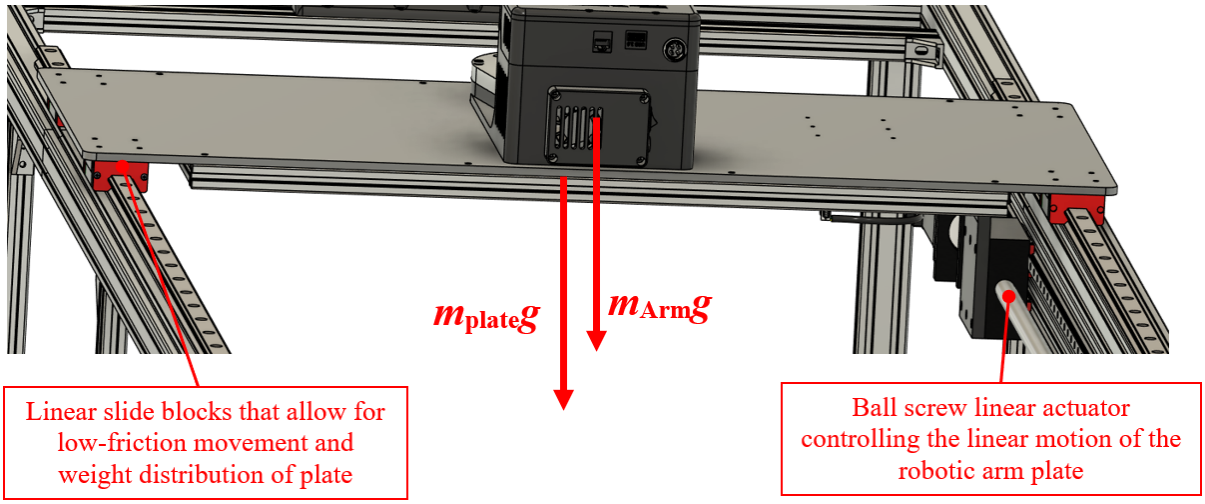


Figure B.3. Gantry system analysis for linear actuator torque calculation.

Known Values:

- Mass of AR4-MK3 robotic arm: 12.25 kg [1]
- System efficiency factor: 0.9 [6]
- Pitch of ball screw: 5 mm [3]
- Density of aluminum: 2700 kg/m^3
- Gravitational acceleration: 9.81 m/s^2

Analysis:

First, estimate the mass of the aluminum plate (822 mm × 380 mm × 6.35 mm):

$$V = l \times w \times h = 0.822 \text{ m} \times 0.38 \text{ m} \times 0.00635 \text{ m} = 0.001984 \text{ m}^3$$

$$m_{\text{plate}} = \rho \times V = 2700 \frac{\text{kg}}{\text{m}^3} \times 0.001984 \text{ m}^3 = 5.36 \text{ kg}$$

Next, combine the mass of the robotic arm and the aluminum plate:

$$m_{\text{combined}} = 12.25 \text{ kg} + 5.36 \text{ kg} = 17.61 \text{ kg}$$

Calculate the gravitational force:

$$F = m_{\text{combined}} \times g = 17.61 \text{ kg} \times 9.81 \frac{\text{m}}{\text{s}^2} = 172.79 \text{ N}$$

Apply the force to the torque formula for a ball screw linear actuator [6]:

$$T = \frac{F \times \text{Pitch}}{2000 \times \pi \times \text{Efficiency}}$$

$$T = \frac{172.79 \text{ N} \times 5 \text{ mm}}{2000 \times \pi \times 0.9}$$

$$T = \frac{863.95}{5654.87}$$

$$T \approx 0.155 \text{ Nm}$$

Conclusion: The torque required to move the load in the worst-case scenario is approximately 0.155 Nm, which is well below the available torque of the NEMA 23 stepper motor (3.0 Nm). Thus, the motor is more than capable of driving the ball screw linear actuator.

Appendix C: Camera Configuration Code

The following Swift code was developed by Fabri Sciences to utilize an iPhone's native camera SDK using AVFoundation. The AVCameraService object is an actor which initializes the iPhone's camera in a thread-safe manner. The service is tailored to capture high-quality data with depth information and camera properties. Next, the PhotoCaptureDelegate is a class responsible for extracting the image data from the camera sensors after the time of capture. Lastly, the data output stream is a class responsible for streaming RGBA image data and Float32 depth data from the camera sensors in real time. The class conforms to the AVCaptureOutputDataSynchronizer protocol, allowing it to receive paired (by timestamp) image frames. This class is especially useful because it allows our camera to camera up to 15 frames worth of data after the AVCapturePhoto is captured, allowing for retrospective temporal denoising and post-processing.

```
1
2  //
3  //  AVCameraService.swift
4  //  Cameras
5  //
6  //  Created by Jacob Damant on 2025-02-18.
7  //
8
9  /**
10  An actor that manages the capture pipeline, which includes the capture
        session, device inputs, and capture outputs.
11  The app defines it as an `actor` type to ensure that all camera
        operations happen off of the `@MainActor`.
12  */
13 actor AVCameraService {
14
15     /// The live video output.
16     let videoDataOutput: AVCaptureVideoDataOutput =
           AVCaptureVideoDataOutput()
17
18     /// The live depth sensor output.
19     let depthDataOutput: AVCaptureDepthDataOutput =
           AVCaptureDepthDataOutput()
20
21     /// An asynchronous queue that is strictly responsible for handling
           the synchronization of the video and depth frames.
22     private let dataOutputQueue = DispatchQueue(label:
           "av.output.data.synchronizer.queue", qos: .userInitiated,
           attributes: [], autoreleaseFrequency: .workItem)
23
24     /// The synchronizer responsible for syncing video and depth buffers
           by their timestamp.
```

```
25      var synchronizer: AVCaptureDataOutputSynchronizer?
26
27      /// A custom delegate responsible for handling and capturing live
            video and depth frames.
28      var delegate: OutputDataSynchronizerQueueDelegate?
29
30      /// The data receiver that will process the camera output
31      private var dataReceiver: AVDataReceiver?
32
33
34      /**
35       Establishes an object to record and synchronize the live video and
             depth feed.
36       */
37      func setup() {
38
39          // Initialize a synchronizer to handle the synchronization logic
40          self.synchronizer = AVCaptureDataOutputSynchronizer(dataOutputs:
                [videoDataOutput, depthDataOutput])
41
42          // Create a delegate and maintain a strong reference
43          let delegate =
                OutputDataSynchronizerQueueDelegate(videoDataOutput:
                videoDataOutput, depthDataOutput: depthDataOutput)
44          delegate.delegate = dataReceiver // Set the data receiver
45          self.delegate = delegate // Store strong reference
46
47          self.synchronizer?.setDelegate(delegate, queue: dataOutputQueue)
48
49      }
50
51      func captureSyncedBuffers(frameLimit: Int) async throws -> (images:
            [SendableCVPixelBuffer], depthMaps: [SendableCVPixelBuffer]) {
52
53          // Ensure the delegate has been initialed before trying to capture
                buffers
54          guard let delegate = self.delegate else {
55              throw CameraError.failedToCaptureScan
56          }
57
58          // Wrap the delegate-based capture API in a continuation to use it
                in an async context.
59          return try await withCheckedThrowingContinuation { (continuation:
                AVOutputDataContinuation) in
60
61              // Set a continuation object to transfer the captured frames
                    from the delegate
```

```
62              delegate.setContinuation(continuation)
63
64              // Begin capturing the live video and depth buffers
65              delegate.capture(frameLimit: frameLimit)
66          }
67      }
68
69      func setDataReceiver(_ receiver: AVDataReceiver) {
70          self.dataReceiver = receiver
71
72          // Update the output delegate if it exists
73          delegate?.delegate = receiver
74      }
75
76
77      /// A value that indicates whether the capture service is idle or
            capturing a photo.
78      @Published private(set) var captureActivity: CameraActivity = .idle
79
80      /// A Boolean value that indicates whether a higher priority event,
            like receiving a phone call, interrupts the app.
81      @Published private(set) var isInterrupted = false
82
83      /// A type that connects a preview destination with the capture
            session.
84      nonisolated let previewSource: AVPreviewSource
85
86      /// The app's capture session.
87      private let captureSession = AVCaptureSession()
88
89      /// An object that manages the camera's photo capture behavior.
90      private let photoCapture = PhotoCapture()
91
92      /// An object that manages the camera's live output data stream,
            including video data and depth data.
93      private let outputDataStream = AVOutputDataStream()
94
95     private let cameraPosition: CameraType.AVPosition
96
97      /// Provides a lookup object for any camera devices that meet the
            specified criteria.
98      private let videoDeviceDiscoverySession:
            AVCaptureDevice.DiscoverySession
99
100     /// The video input for the currently selected device camera.
101     private var activeVideoInput: AVCaptureDeviceInput?
102
```

```
103     /// A Boolean value that indicates whether the actor finished its
            required configuration.
104     private var isSetUp = false
105
106     // MARK: Init
107     init(cameraPosition: CameraType.AVPosition) {
108             self.cameraPosition = cameraPosition
109             self.videoDeviceDiscoverySession =
                    AVCaptureDevice.DiscoverySession(
110                  deviceTypes: [cameraPosition.deviceType],
111                  mediaType: .video,
112                  cameraPosition == .back ? .back : .front
113             )
114             previewSource = DefaultPreviewSource(session: captureSession)
115     }
116
117
118
119     // MARK: Stop
120     func stop() async {
121         // Stop the capture session if it's running
122         if captureSession.isRunning {
123             captureSession.stopRunning()
124         }
125
126         // Begin configuration
127         captureSession.beginConfiguration()
128
129         // Remove all inputs
130         for input in captureSession.inputs {
131             captureSession.removeInput(input)
132         }
133
134         // Remove all outputs
135         for output in captureSession.outputs {
136             captureSession.removeOutput(output)
137         }
138
139         // Clear the photo output settings
140         photoCapture.photoOutput.setPreparedPhotoSettingsArray([],
                completionHandler: nil)
141
142         // Clear the synchronizer and delegate
143         synchronizer?.setDelegate(nil, queue: nil)
144         synchronizer = nil
145         delegate = nil
146         dataReceiver = nil
```

```
147
148         // Clear the video input
149         activeVideoInput = nil
150
151         // Reset setup flag
152         isSetUp = false
153
154         // Commit configuration
155         captureSession.commitConfiguration()
156
157         // Cancel any ongoing tasks
158         subjectAreaChangeTask?.cancel()
159         subjectAreaChangeTask = nil
160     }
161
162
163
164     // MARK: - Authorization
165
166     /// A Boolean value that indicates whether a person authorizes this
        app to use device cameras.
167     /// If they haven't previously authorized the app, querying this
        property prompts them for authorization.
168     var isAuthorized: Bool {
169         get async {
170             let status = AVCaptureDevice.authorizationStatus(for: .video)
171             // Determine whether a person previously authorized camera
                access.
172             var isAuthorized = status == .authorized
173             // If the system hasn't determined their authorization status,
174             // explicitly prompt them for approval.
175             if status == .notDetermined {
176                 isAuthorized = await AVCaptureDevice.requestAccess(for:
                    .video)
177             }
178             return isAuthorized
179         }
180     }
181
182
183     // MARK: - Capture Session Life Cycle
184
185     /**
186      Sets up the capture session based on the selected AV camera.
187
188      - Throws: Throws an error if the capture session cannot be setup.
189      */
```

```
190     func start() async -> Result<Void, ErrorWrapper> {
191
192         // Exit early if not authorized or the session is already running.
193         guard await isAuthorized, !captureSession.isRunning else { return
                .success(()) }
194
195         // Configure the session and start it.
196         let setupResult = setUpSession()
197         switch setupResult {
198         case .success():
199             captureSession.startRunning()
200         case .failure(let error):
201             return .failure(error)
202         }
203
204         return .success(())
205     }
206
207
208     // MARK: - Capture Setup
209
210     /**
211      Performs the initial capture session configuration.
212
213      - Returns: Returns void on success and an error if the AV camera
             cannot be properly configured.
214      */
215     private func setUpSession() -> Result<Void, ErrorWrapper> {
216
217         // Return early if already set up.
218         guard !isSetUp else { return .success(()) }
219
220         // Observe internal state and notifications.
221         observeOutputServices()
222         observeNotifications()
223
224         do {
225             // Marks the beginning of a single atomic configuration that
                    can be applied to a running camera session
226             captureSession.beginConfiguration()
227
228             // Retrieve the default camera
229             let videoDevice: AVCaptureDevice =
                    self.videoDeviceDiscoverySession.devices.first!
230
231             // Add inputs for the default camera device
232             activeVideoInput = try addInput(for: videoDevice)
```

```
233
234             // Configure the session for photo capture by default
235             captureSession.sessionPreset = .photo
236
237             // Add the photo capture output as the default output type.
238             try addOutput(photoCapture.photoOutput)
239
240             // Ensure the photo output supports depth data
241             guard photoCapture.photoOutput.isDepthDataDeliverySupported
                    else {
242                 throw CameraError.failedToSetupCamera
243             }
244             photoCapture.photoOutput.isDepthDataDeliveryEnabled =
                    photoCapture.photoOutput.isDepthDataDeliverySupported
245
246             // Add the video data output
247             try addOutput(self.videoDataOutput)
248
249             // Configure the buffer format for the streamed video data
250             self.videoDataOutput.videoSettings = [
251                 kCVPixelBufferPixelFormatTypeKey as String:
                        Int(kCVPixelFormatType_32BGRA)
252             ]
253
254             // Add the depth data output
255             try addOutput(self.depthDataOutput)
256
257             // Disable Apple's filtering algorithms on the depth data
258             self.depthDataOutput.isFilteringEnabled = false
259
260             // Ensure the depth data is formatted as AVDepthData
261             guard let connection = self.depthDataOutput.connection(with:
                    .depthData) else {
262                 throw CameraError.failedToSetupCamera
263             }
264             connection.isEnabled = true
265
266             // Format the video device's depth data so that it can be
                    properly utilized by the metal shaders
267             let depthFormats =
                    videoDevice.activeFormat.supportedDepthDataFormats
268             let filtered = depthFormats.filter({
269                 CMFormatDescriptionGetMediaSubType($0.formatDescription)
                        == kCVPixelFormatType_DisparityFloat32
270             })
271             let selectedFormat = filtered.max(by: {
272                 first, second in
```

```
                CMVideoFormatDescriptionGetDimensions(first.formatDescription).width
                <
                CMVideoFormatDescriptionGetDimensions(second.formatDescription).width
            })

            // Apply the video format
            try videoDevice.lockForConfiguration()
            videoDevice.activeDepthDataFormat = selectedFormat
            videoDevice.unlockForConfiguration()

            self.setup()

            // Observe changes to the default camera's subject area.
            observeSubjectAreaChanges(of: videoDevice)

            isSetUp = true

            captureSession.commitConfiguration()

            captureSession.startRunning()
        } catch {
            return .failure(.init(CameraError.failedToSetupCamera))
        }

        return .success(())
    }

    /**
     Adds an input to the capture session to connect the specified capture
         device.

     - Parameters:
       - device: An AV camera device that can capture image or audio data.
     - Returns: The device input on success.
     - Throws: A .failedToSetupCamera error on failure.
     */
    @discardableResult
    private func addInput(for device: AVCaptureDevice) throws ->
        AVCaptureDeviceInput {

        // Try adding the input to the capture session
        guard let input = try? AVCaptureDeviceInput(device: device),
              captureSession.canAddInput(input)
        else {
            throw CameraError.failedToSetupCamera
        }
        captureSession.addInput(input)
```

```
318
319             return input
320         }
321
322         /**
323          Adds an output to the capture session to connect the specified
                 capture device, if allowed.
324
325          - Parameters:
326             - output: An output object that handles the data coming from the
                    camera sensors.
327          - Throws: Throws a .failedToSetupCamera error on failure.
328          */
329         private func addOutput(_ output: AVCaptureOutput) throws {
330             // Try adding the output to the capture session
331             guard captureSession.canAddOutput(output) else {
332                 throw CameraError.failedToSetupCamera
333             }
334             captureSession.addOutput(output)
335
336             // Configure the connection right after adding the output
337             if let connection = output.connection(with: .video) {
338                 connection.isVideoMirrored = false
339                 connection.automaticallyAdjustsVideoMirroring = false
340             }
341
342             // Handle depth data connection
343             if let connection = output.connection(with: .depthData) {
344                 connection.isVideoMirrored = false
345                 connection.automaticallyAdjustsVideoMirroring = false
346             }
347
348             return
349         }
350
351         /// The device for the active video input.
352         private var currentDevice: AVCaptureDevice {
353             guard let device = activeVideoInput?.device else {
354                 fatalError("No device found for current video input.")
355             }
356             return device
357         }
358
359         // MARK: - Preview Layer
360
361         private var videoPreviewLayer: AVCaptureVideoPreviewLayer {
362             // Access the capture session's connected preview layer.
```

```
363         guard let previewLayer = captureSession.connections.compactMap({
                $0.videoPreviewLayer }).first else {
364             fatalError("The app is misconfigured. The capture session
                    should have a connection to a preview layer.")
365         }
366         return previewLayer
367     }
368 
369     // MARK: - Automatic Focus and Exposure
370 
371     /// Performs a one-time automatic focus and expose operation.
372     ///
373     /// The app calls this method as the result of a person tapping on the
            preview area.
374     func focusAndExpose(at devicePoint: CGPoint) {
375         do {
376             try focusAndExpose(at: devicePoint, isUserInitiated: true)
377         } catch {
378             print("Unable to perform focus and exposure operation:
                    \(error)")
379         }
380     }
381 
382     func setExposure(value: Float) async {
383         guard let device = activeVideoInput?.device else { return }
384 
385         do {
386             try device.lockForConfiguration()
387 
388             // Keep auto-exposure on but adjust the target bias
389             if device.isExposurePointOfInterestSupported {
390                 device.exposureMode = .continuousAutoExposure
391 
392                 // Convert Float to Float64 for device API
393                 let targetBias = Float64(value)
394                 let minBias = Float64(device.minExposureTargetBias)
395                 let maxBias = Float64(device.maxExposureTargetBias)
396 
397                 // Clamp the bias value between min and max
398                 let clampedBias = max(minBias, min(targetBias, maxBias))
399 
400                 await device.setExposureTargetBias(Float(clampedBias))
401             }
402 
403             device.unlockForConfiguration()
404         } catch {
405             print("Error setting exposure: \(error)")
```

```
406         }
407     }
408
409     // Observe notifications of type 'subjectAreaDidChangeNotification'
            for the specified device.
410     private func observeSubjectAreaChanges(of device: AVCaptureDevice) {
411         // Cancel the previous observation task.
412         subjectAreaChangeTask?.cancel()
413         subjectAreaChangeTask = Task {
414             // Signal true when this notification occurs.
415             for await _ in NotificationCenter.default.notifications(named:
                    AVCaptureDevice.subjectAreaDidChangeNotification, object:
                    device).compactMap({ _ in true }) {
416                 // Perform a system-initiated focus and expose.
417                 try? focusAndExpose(at: CGPoint(x: 0.5, y: 0.5),
                        isUserInitiated: false)
418             }
419         }
420     }
421     private var subjectAreaChangeTask: Task<Void, Never>?
422
423     private func focusAndExpose(at devicePoint: CGPoint, isUserInitiated:
            Bool) throws {
424         // Configure the current device.
425         let device = currentDevice
426
427         // The following mode and point of interest configuration requires
                obtaining an exclusive lock on the device.
428         try device.lockForConfiguration()
429
430         let focusMode = isUserInitiated ?
                AVCaptureDevice.FocusMode.autoFocus : .continuousAutoFocus
431         if device.isFocusPointOfInterestSupported &&
                device.isFocusModeSupported(focusMode) {
432             device.focusPointOfInterest = devicePoint
433             device.focusMode = focusMode
434         }
435
436         let exposureMode = isUserInitiated ?
                AVCaptureDevice.ExposureMode.autoExpose :
                .continuousAutoExposure
437         if device.isExposurePointOfInterestSupported &&
                device.isExposureModeSupported(exposureMode) {
438             device.exposurePointOfInterest = devicePoint
439             device.exposureMode = exposureMode
440         }
```

```
441         // Enable subject-area change monitoring when performing a
                user-initiated automatic focus and exposure operation.
442         // If this method enables change monitoring, when the device's
                subject area changes, the app calls this method a
443         // second time and resets the device to continuous automatic focus
                and exposure.
444         device.isSubjectAreaChangeMonitoringEnabled = isUserInitiated
445
446         // Release the lock.
447         device.unlockForConfiguration()
448     }
449
450     // MARK: - Photo capture
451     func capturePhoto() async throws -> SendableAVCapturePhoto {
452         try await photoCapture.capturePhoto()
453     }
454
455     private var outputService: PhotoCapture { photoCapture }
456
457
458     // MARK: - Capture mode selection
459
460     /// Changes the mode of capture, which can be `photo` or `video`.
461     ///
462     /// - Parameter `captureMode`: The capture mode to enable.
463     func setCaptureMode() throws {
464         // Update the internal capture mode value before performing the
                session configuration.
465
466         // Change the configuration atomically.
467         captureSession.beginConfiguration()
468         defer { captureSession.commitConfiguration() }
469
470         // The app needs to remove the movie capture output to perform
                Live Photo capture.
471         captureSession.sessionPreset = .photo
472     }
473
474
475     // MARK: - Internal state management
476
477     /// Merge the `captureActivity` values of the photo and movie capture
            services,
478     /// and assign the value to the actor's property.`
479     private func observeOutputServices() {
480         Publishers.Merge(photoCapture.$captureActivity,
                movieCapture.$captureActivity)
```

```
481             .assign(to: &$captureActivity)
482     }
483
484     /// Observe capture-related notifications.
485     private func observeNotifications() {
486         Task {
487             for await reason in
                    NotificationCenter.default.notifications(named:
                    AVCaptureSession.wasInterruptedNotification)
488                 .compactMap({
                        $0.userInfo?[AVCaptureSessionInterruptionReasonKey] as
                        AnyObject? })
489                 .compactMap({
                        AVCaptureSession.InterruptionReason(rawValue:
                        $0.integerValue) }) {
490                 /// Set the `isInterrupted` state as appropriate.
491                 isInterrupted = [.audioDeviceInUseByAnotherClient,
                        .videoDeviceInUseByAnotherClient].contains(reason)
492             }
493         }
494
495         Task {
496             // Await notification of the end of an interruption.
497             for await _ in NotificationCenter.default.notifications(named:
                    AVCaptureSession.interruptionEndedNotification) {
498                 isInterrupted = false
499             }
500         }
501
502         Task {
503             for await error in
                    NotificationCenter.default.notifications(named:
                    AVCaptureSession.runtimeErrorNotification)
504                 .compactMap({ $0.userInfo?[AVCaptureSessionErrorKey] as?
                        AVError }) {
505                 // If the system resets media services, the capture
                        session stops running.
506                 if error.code == .mediaServicesWereReset {
507                     if !captureSession.isRunning {
508                         captureSession.startRunning()
509                     }
510                 }
511             }
512         }
513     }
514 }
515
```

```swift
//
//  PhotoCapture.swift
//  Cameras
//
//  Created by Jacob Damant on 2025-02-24.
//

@preconcurrency import AVFoundation
import CoreImage
import SwiftUI
import Utilities

typealias PhotoContinuation = CheckedContinuation<SendableAVCapturePhoto, Error>

/// An object that manages a photo capture output to perform take photographs.
final class PhotoCapture: Sendable {

    /// The capture output type for this service.
    let photoOutput = AVCapturePhotoOutput()

    // MARK: - Capture a photo.

    /**
     The app calls this method when the user taps the photo capture button.

     - Returns: An AVCapturePhoto encapsulating an RGB image, depth map, and camera properties.
     - Throws: Throws an error if the camera fails to generate an image.
     */
    func capturePhoto() async throws -> SendableAVCapturePhoto {

        // Wrap the delegate-based capture API in a continuation to use it in an async context.
        try await withCheckedThrowingContinuation { continuation in

            // Create a settings object to configure the photo capture.
            let photoSettings = createPhotoSettings()

            let delegate = PhotoCaptureDelegate(continuation: continuation)
            monitorProgress(of: delegate)

            // Capture a new photo with the specified settings.
            photoOutput.capturePhoto(with: photoSettings, delegate: delegate)
        }
```

```
558     }
559
560     // MARK: - Create a photo settings object.
561
562     /**
563      Create a photo settings object with a configuration that produces the
           best data for 3D scans.
564
565      - Returns: An object encapsulating the configured photo capture
           settings.
566      */
567     private func createPhotoSettings() -> AVCapturePhotoSettings {
568
569         // Create a new settings object to configure the photo capture.
570         let photoSettings = AVCapturePhotoSettings()
571
572
573         // Set the largest dimensions that the photo output supports
574         // ‘CaptureService‘ automatically updates the photo output's
               ‘maxPhotoDimensions‘ when the capture pipeline changes
575         photoSettings.maxPhotoDimensions = photoOutput.maxPhotoDimensions
576
577         // Set the flash mode to off
578         photoSettings.flashMode = .off
579
580         // Enable depth data delivery
581         photoSettings.isDepthDataDeliveryEnabled =
               photoOutput.isDepthDataDeliveryEnabled
582
583         photoSettings.isDepthDataFiltered = false
584
585         // Set the priority of the capture to quality
586         if let prioritization =
               AVCapturePhotoOutput.QualityPrioritization(rawValue:
               AVCapturePhotoOutput.QualityPrioritization.speed.rawValue) {
587             photoSettings.photoQualityPrioritization = prioritization
588         }
589
590         return photoSettings
591     }
592
593     /// Monitors the progress of a photo capture delegate.
594     ///
595     /// The ‘PhotoCaptureDelegate‘ produces an asynchronous stream of
           values that indicate its current activity.
596     /// The app propagates the activity values up to the view tier so the
           UI can update accordingly.
```

```
597     private func monitorProgress(of delegate: PhotoCaptureDelegate,
            isolation: isolated (any Actor)? = #isolation) {
598         Task {
599             _ = isolation
600             // Asynchronously monitor the activity of the delegate while
                    the system performs capture.
601             for await _ in delegate.activityStream {
602                 captureActivity = activity
603             }
604         }
605     }
606
607
608     // MARK: - Update the photo output configuration
609
610     /**
611      Reconfigures the photo output and updates the output service's
             capabilities accordingly.
612
613      The `CaptureService` calls this method whenever you change cameras.
614
615      - Parameters:
616         - device: The type of capture device used to take the photo.
617      */
618     func updateConfiguration(for device: AVCaptureDevice) {
619         // Enable all supported features.
620         photoOutput.isDepthDataDeliveryEnabled =
                photoOutput.isDepthDataDeliverySupported
621         photoOutput.maxPhotoDimensions =
                device.activeFormat.supportedMaxPhotoDimensions.last ??
                CMVideoDimensions()
622         photoOutput.isLivePhotoCaptureEnabled =
                photoOutput.isLivePhotoCaptureSupported
623         photoOutput.maxPhotoQualityPrioritization = .quality
624         photoOutput.isResponsiveCaptureEnabled =
                photoOutput.isResponsiveCaptureSupported
625         photoOutput.isFastCapturePrioritizationEnabled =
                photoOutput.isFastCapturePrioritizationSupported
626         photoOutput.isAutoDeferredPhotoDeliveryEnabled =
                photoOutput.isAutoDeferredPhotoDeliverySupported
627         photoOutput.isAutoDeferredPhotoDeliveryEnabled = false
628     }
629 }
630
631 // MARK: - A photo capture delegate to process the captured photo.
632
```

```
633 /// An object that adopts the ‘AVCapturePhotoCaptureDelegate‘ protocol to
        respond to photo capture life-cycle events.
634 ///
635 /// The delegate produces a stream of events that indicate its current
        state of processing.
636 private final class PhotoCaptureDelegate: NSObject,
        AVCapturePhotoCaptureDelegate, Sendable {
637
638     private let continuation: PhotoContinuation
639
640     /// A stream of capture activity values that indicate the current
            state of progress.
641     let activityStream: AsyncStream<CameraActivity>
642     private let activityContinuation:
            AsyncStream<CameraActivity>.Continuation
643
644     /// Creates a new delegate object with the checked continuation to
            call when processing is complete.
645     init(continuation: PhotoContinuation) {
646         self.continuation = continuation
647
648         let (activityStream, activityContinuation) =
                AsyncStream.makeStream(of: CameraActivity.self)
649         self.activityStream = activityStream
650         self.activityContinuation = activityContinuation
651     }
652
653     func photoOutput(_ output: AVCapturePhotoOutput, willCapturePhotoFor
            resolvedSettings: AVCaptureResolvedPhotoSettings) {
654         // Signal that a capture is beginning.
655         activityContinuation.yield(.photoCapture)
656     }
657
658     func photoOutput(_ output: AVCapturePhotoOutput,
            didFinishProcessingPhoto photo: AVCapturePhoto, error: Error?) {
659         if let error = error {
660             print("Error: Failed to retrieve the AVCapturePhoto from the
                    camera delegate: \(error.localizedDescription)")
661             continuation.resume(throwing: CameraError.failedToCaptureScan)
662             return
663         }
664
665         guard let sendablePhoto: SendableAVCapturePhoto = try?
                SendableAVCapturePhoto(photo, depthType: .TrueDepth) else {
666             print("Error: Failed to convert photo to sendable format")
667             continuation.resume(throwing: CameraError.failedToCaptureScan)
668             return
```

```
669             }
670
671             continuation.resume(returning: (sendablePhoto))
672         }
673     }
674
675
676     //
677     //  AVOutputDataStream.swift
678     //  Cameras
679     //
680     //  Created by Jacob Damant on 2025-02-20.
681     //
682
683     import AVFoundation
684     import Domain
685     import Foundation
686     import Utilities
687
688     /// Allows the delegate to return its result on success in an asyncronous
        function.
689     typealias AVOutputDataContinuation = CheckedContinuation<(images:
        [SendableCVPixelBuffer], depthMaps: [SendableCVPixelBuffer]), Error>
690
691     /**
692      An object that manages a photo capture output to perform take photographs.
693      */
694     final class AVOutputDataStream {
695
696         /// The live video output.
697         let videoDataOutput: AVCaptureVideoDataOutput =
            AVCaptureVideoDataOutput()
698
699         /// The live depth sensor output.
700         let depthDataOutput: AVCaptureDepthDataOutput =
            AVCaptureDepthDataOutput()
701
702         /// An asynchronous queue that is strictly responsible for handling
            the synchronization of the video and depth frames.
703         private let dataOutputQueue = DispatchQueue(label:
            "av.output.data.synchronizer.queue", qos: .userInitiated,
            attributes: [], autoreleaseFrequency: .workItem)
704
705         /// The synchronizer responsible for syncing video and depth buffers
            by their timestamp.
706         var synchronizer: AVCaptureDataOutputSynchronizer?
707
```

```
708     /// A custom delegate responsible for handling and capturing live
            video and depth frames.
709     var delegate: (any OutputDataSynchronizerQueueDelegateProtocol)?
710
711     /**
712      Establishes an object to record and synchronize the live video and
             depth feed.
713      */
714     func setup() {
715
716         // Initialize a synchronizer to handle the synchronization logic
                for the video and depth camera feeds
717         self.synchronizer = AVCaptureDataOutputSynchronizer(dataOutputs:
                [videoDataOutput, depthDataOutput])
718
719         // Create a delegate to handle the live video and depth buffers
                provided by the synchronizer
720         let delegate =
                OutputDataSynchronizerQueueDelegate(videoDataOutput:
                videoDataOutput, depthDataOutput: depthDataOutput)
721         self.delegate = delegate
722         self.synchronizer?.setDelegate(delegate, queue: dataOutputQueue)
723
724     }
725
726     /**
727      The app calls this method immediately after capturing a photo to
             store the synced video and depth frames.
728
729      - Parameters:
730         - frameLimit: The number of frames that should be captured by the
                data output synchronizer.
731      */
732     func captureSynchronizedBuffers(frameLimit: Int) async throws ->
            (images: [SendableCVPixelBuffer], depthMaps:
            [SendableCVPixelBuffer]) {
733
734         // Ensure the delegate has been initialed before trying to capture
                buffers
735         guard let delegate = self.delegate else {
736             throw CameraError.failedToCaptureScan
737         }
738
739         // Wrap the delegate-based capture API in a continuation to use it
                in an async context.
740         return try await withCheckedThrowingContinuation { (continuation:
                AVOutputDataContinuation) in
```

```
741
742             // Set a continuation object to transfer the captured frames
                    from the delegate
743             delegate.setContinuation(continuation)
744
745             // Begin capturing the live video and depth buffers
746             delegate.capture(frameLimit: frameLimit)
747         }
748     }
749 }
750
751 /**
752  Defines the custom method used to capture live video and depth buffers as
        they are streamed.
753  */
754 protocol OutputDataSynchronizerQueueDelegateProtocol:
    AVCaptureDataOutputSynchronizerDelegate {
755
756     /**
757      Adds a continuation object to the syncronizer delegate so that the
            captured frames can be fetched from the delegate after the capture.
758
759      - Parameters:
760         - continuation: A custom AVOutputData continuation that stores
                image and depth map buffers.
761      */
762     func setContinuation(_ continuation: AVOutputDataContinuation)
763
764     /**
765      Triggers the syncronizer delegate to being capturing its live video
            and depth buffers.
766
767      - Parameters:
768         - frameLimit: The maximum number of frames captured by the
                delegate.
769      */
770     func capture(frameLimit: Int)
771
772 }
773
774 final class OutputDataSynchronizerQueueDelegate: NSObject,
    OutputDataSynchronizerQueueDelegateProtocol {
775
776     /// A reference to the camera's live video feed.
777     let videoDataOutput: AVCaptureVideoDataOutput
778
779     /// A reference to the camera's live depth feed.
```

```
780     let depthDataOutput: AVCaptureDepthDataOutput
781
782     /// A continuation object that provides an outlet to transfer captured
            frames out of the delegate.
783     var continuation: AVOutputDataContinuation?
784
785     /// A queue that stores video and depth map frames until it reaches
            its specified limit.
786     var bufferQueue: BufferQueue?
787
788     /// Indicates if the buffer queue should enque the live frames .
789     var isQueueRunning: Bool = false
790
791     weak var delegate: AVDataReceiver?
792
793     init(videoDataOutput: AVCaptureVideoDataOutput, depthDataOutput:
            AVCaptureDepthDataOutput) {
794         self.videoDataOutput = videoDataOutput
795         self.depthDataOutput = depthDataOutput
796         super.init()
797     }
798
799     /**
800      Set a custom AVOutputData continuation object to transport video and
             depth frames from the syncronizer delegate.
801
802      - Parameters:
803        - continuation: The continuation object created by the capture
             function that calls the syncronizer delegate.
804      */
805     func setContinuation(_ continuation: AVOutputDataContinuation) {
806         self.continuation = continuation
807     }
808
809     /**
810      Captures the specified number of frames from the camera's live feed,
             including video and depth map buffers.
811
812      - Parameters:
813        - frameLimit: The maximum number of frames to capture from the
             delegate.
814      */
815     func capture(frameLimit: Int) {
816         self.bufferQueue = BufferQueue(maxSize: frameLimit)
817         self.isQueueRunning = true
818     }
819
```

```
820     func dataOutputSynchronizer(_ synchronizer:
            AVCaptureDataOutputSynchronizer, didOutput
            synchronizedDataCollection: AVCaptureSynchronizedDataCollection) {
821
822         // Unwrap the synchronizer data
823         guard let syncedVideoData: AVCaptureSynchronizedSampleBufferData =
                synchronizedDataCollection.synchronizedData(for:
                self.videoDataOutput) as? AVCaptureSynchronizedSampleBufferData,
824                 let syncedDepthData: AVCaptureSynchronizedDepthData =
                    synchronizedDataCollection.synchronizedData(for:
                    self.depthDataOutput) as? AVCaptureSynchronizedDepthData
825         else {
826             return
827         }
828
829         // Only proceed if the video and depth frames were properly
                delivered after the synchronization point
830         guard !syncedVideoData.sampleBufferWasDropped &&
                !syncedDepthData.depthDataWasDropped else {
831             return
832         }
833
834         guard let videoBuffer: CVPixelBuffer =
                CMSampleBufferGetImageBuffer(syncedVideoData.sampleBuffer) else
                {
835             return
836         }
837         let depthMapBuffer: CVPixelBuffer =
                syncedDepthData.depthData.converting(toDepthDataType:
                kCVPixelFormatType_DepthFloat32).depthDataMap
838
839         let temp: UIImage = UIImage(pixelBuffer: videoBuffer)!
840
841         delegate?.onNewAVData(image: videoBuffer, depthMap:
                depthMapBuffer, depthData: syncedDepthData.depthData)
842
843         // Convert the buffers into a sendable format to obey Swift
                concurrency rules
844         let sendableVideoBuffer: SendableCVPixelBuffer =
                SendableCVPixelBuffer(buffer: videoBuffer)
845         let sendableDepthMapBuffer: SendableCVPixelBuffer =
                SendableCVPixelBuffer(buffer: depthMapBuffer)
846
847         // Eneque the video and depth buffers if the camera is capturing
848         if isQueueRunning {
849             guard let bufferQueue = self.bufferQueue else {
```

```
850                     continuation?.resume(throwing:
                            CameraError.failedToCaptureScan)
851                     return
852                 }
853                 let isFull: Bool = bufferQueue.enqueue(imageBuffer:
                        sendableVideoBuffer, depthMapBuffer: sendableDepthMapBuffer)
854
855                 // Return the buffer queue once it is full
856                 if isFull {
857                     isQueueRunning = false
858                     let syncedBuffers = bufferQueue.fetchBuffers()
859                     continuation?.resume(returning:
                            (syncedBuffers.imageBuffers,
                            syncedBuffers.depthMapBuffers))
860                 }
861             }
862         }
863 }
```

Appendix D: Cloud System

D.1 System Capture, Coordination, & Compute

The solution proposed by Fabri Sciences involves precise coordination of multiple sub-systems, including the ROS2 robotic arm, the iPhone-based imaging application, and the backend server for real-time communication, data processing, and storage. The primary objective is to enable precise robotic movement for multi-angle image acquisition, where the iPhone captures RGB images and depth maps that are later processed for the client's research purposes. The system comprises four key subsystems:

- The coordinator server, which facilitates communication and state management between the subsystems.
- The iOS application, which is responsible for image capture and data transmission.
- The ROS2-based robotic system, which controls arm movement and provide live positioning feedback to the coordination server.
- The database and storage layer, which manages capture metadata and file storage.

Attached below is a high-level diagram outlining the relationships between the previously-listed software sub-systems.

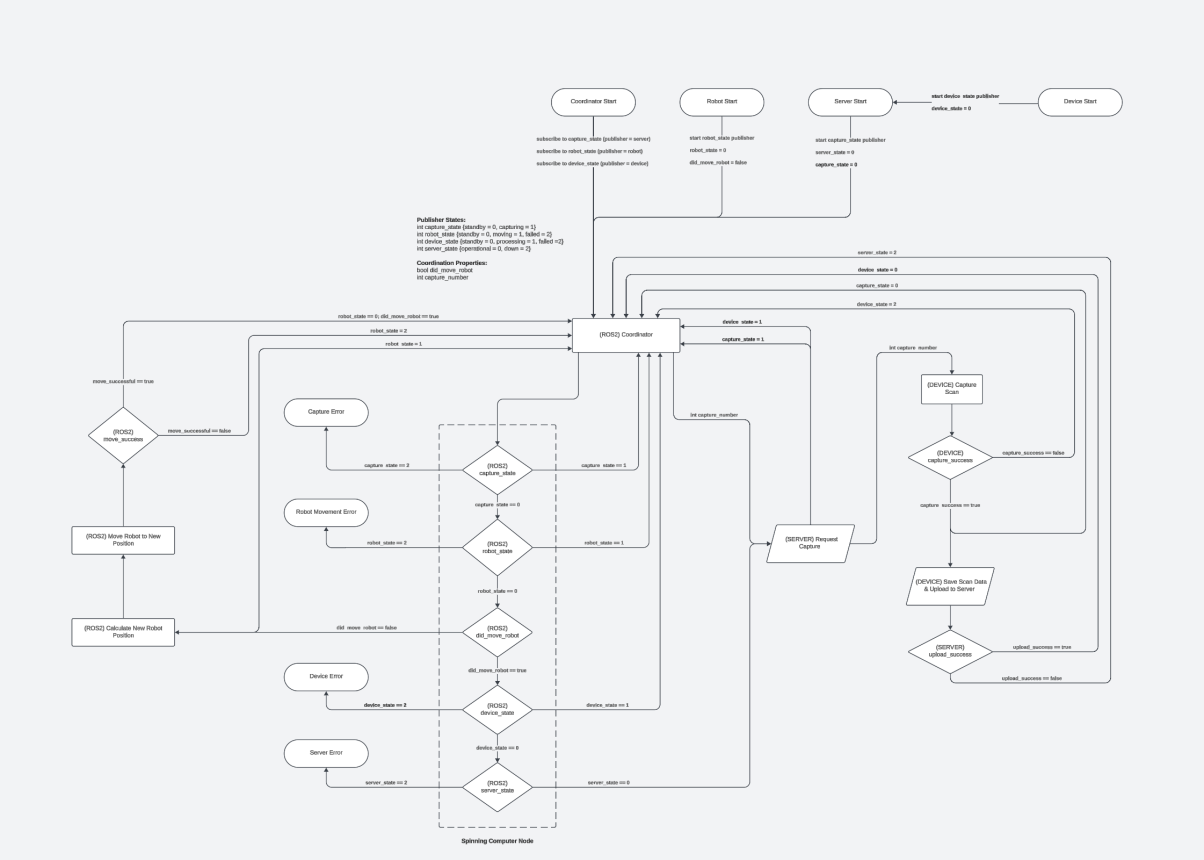


Figure D.4. System architecture for the ROS2 assembly, iOS capture application, and coordination server.

Some of the key technical challenges to overcome include real-time communication between subsystems, high-bandwidth data transfer, and ensuring synchronization between robotic movement and image capture. Efficiently managing large image payloads while keeping network latency low is crucial due to the unpredictable nature of wireless internet connections. Additionally, the architecture must be designed to scale for multi-robot environments while keeping infrastructure costs manageable.

D.2 Capture Configuration

The mobile application is responsible for capturing and transmitting high-quality image data as the ROS2 system moves it in 3D space. The iPhone application relies on AVFoundation, Apple's powerful media framework, to capture both RGB images and depth maps. Capturing complex image sets (in our case, 15 RGB-depth pairs with associated camera properties) requires an efficient, low-memory first-in-first-out (FIFO) capture queue system. Instead of capturing all frames at once, the app leverages AVCaptureSession with a custom capture delegate to ensure sequential image acquisition with proper exposure, focus, and depth accuracy. The AVCaptureSession actor, DataOutputStream class, and PhotoCaptureDelegate is included in Appendix C.

The AVCaptureSession is configured to use AVCapturePhotoOutput in parallel with the AVCapture-

DataOutputSynchronizer, supporting high-resolution image capture alongside depth data. It operates in a multi-format capture pipeline, where different streams (RGB, depth, and metadata) are processed in sync. The capture format balances performance and data accuracy by utilizing JPEG compression for the RGB images and Apple's disparity depth format for depth maps.

A capture delegate queue is utilized to process images efficiently, allowing the system to queue, prioritize, and process each frame before moving to the next. Images and depth maps are saved to the device's caches directory and then removed from memory. This ensures that image data is captured in a structured manner without overwhelming device memory. Capture delegates handle multi-threaded callbacks, ensuring images are stored in a way that minimizes lag between captures and synchronizes correctly with the robot's movement.

D.3 Communication & Data Transmission

Once images are captured on the iPhone application, they need to be efficiently transmitted to the backend server so they can be reviewed by the client's research team retrospectively. Image payloads can be large, with each capture session producing hundreds of megabytes of data. To optimize transfer speed, images are compressed using JPEG format with metadata embedding, reducing size while preserving quality. Additionally, depth maps are stored in efficient 32-bit floating-point representations to maintain precision while reducing storage requirements.

The system's network requirements include low-latency, high-bandwidth connections. A WebSocket-based communication channel is used for real-time control signals, ensuring minimal delay and bi-directional communication between sub-systems. Utilizing WebSocket communication allows for the systems to communicate with each other without wasting polling bandwidth by repeatedly pinging the remote server. Meanwhile, a REST API is employed for image uploads and database interactions, ensuring reliable and scalable data transmission since WebSocket channels are optimized for low-data payloads. The combination of WebSockets for commands and REST APIs for bulk data ensures that the system remains responsive while efficiently handling high data throughput.

D.4 System Integration

API design is central to the system's integration, ensuring that different components interact seamlessly. The backend WebSocket API handles real-time state synchronization (by opening a persistent, bi-directional channel that can transmit small amounts of data between a server and client), while a RESTful API (which is excellent for client-initialized communication and data transmission) manages image uploads, retrieval, and metadata storage. Endpoints are designed to be lightweight and efficient, with JSON-based communication and optional compression for large datasets.

The mobile application integration involves implementing a WebSocket client for live communication and using Apple's URLSession framework for REST-based image uploads. The app must handle network

failures gracefully by implementing retry mechanisms and ensuring transactions are atomic (i.e., partial uploads do not corrupt session data).

The ROS2 integration involves a WebSocket-based ROS2 node that communicates directly with the coordinator server. The ROS node listens for movement commands and publishes status updates to ensure robotic motion is synchronized with image capture.

D.5 Infrastructure Scalability

The system must handle significant computational and storage loads as Kondor Devices intends on scaling this system to multiple robotic arms in the future. The main sources of stress include high-frequency WebSocket updates, concurrent image uploads, and database queries. A combination of edge computing (processing data on the iPhone and ROS system) and cloud-based storage helps balance these loads. For cloud deployment, the required AWS services include:

- Amazon EC2: Hosts the WebSocket server and API.
- Amazon S3: Stores image data and depth maps.
- Amazon RDS (PostgreSQL): Manages metadata and scan logs.

For prototyping, cloud costs are minimized by utilizing local compute alternatives (running the WebSocket server on a dedicated PC connected to the same Wi-Fi network as the iPhone and ROS2 system). This eliminates cloud costs but reduces scalability.

To scale for multiple robots, the system must support multiple WebSocket connections, ensuring each robot operates independently while synchronizing data with the backend. A load-balanced architecture using multiple EC2 instances and containerized deployments will help distribute the workload.

Appendix E: ROS Integration and Simulation Framework

To ensure seamless integration between the software and hardware components, the Robot Operating System (ROS) serves as the central communication hub. ROS processes user commands—such as moving the robotic arm to specific positions or retrieving joint states—and translates these commands into actionable instructions for the microcontroller.

E.1 Simulation Environment

Given the high cost of the robotic arm, it is essential to first simulate its movements in a virtual environment to validate motion planning and detect potential collisions before execution on real hardware. In this setup, MoveIt 2 handles motion planning, while Gazebo provides a physics-based simulation that accurately models real-world interactions such as gravity, friction, and collisions. Simultaneously, Rviz is used for visualization, though it functions purely as an animation tool rather than a physics-based simulator like Gazebo. Two custom scripts were added. The first, motion-control-moveit-ik.cpp, allows users to input an array of target points for automatic motion execution. It also runs as a node that publishes the robot's state and position for use by other software. The second, add-collision.cpp, inserts virtual objects to represent the real environment, helping detect collisions before execution. If one is detected (highlighted red in Figure 6), the plan stops. MoveIt 2 simplifies inverse and forward kinematics using built-in solvers like getPositionIK() and getPositionFK(), avoiding manual calculations. A detailed explanation of these functions is provided in the Appendix B3 section. Once simulation is validated, the control stack can be transferred to the physical robot with minimal changes, ensuring reliable and safe deployment.

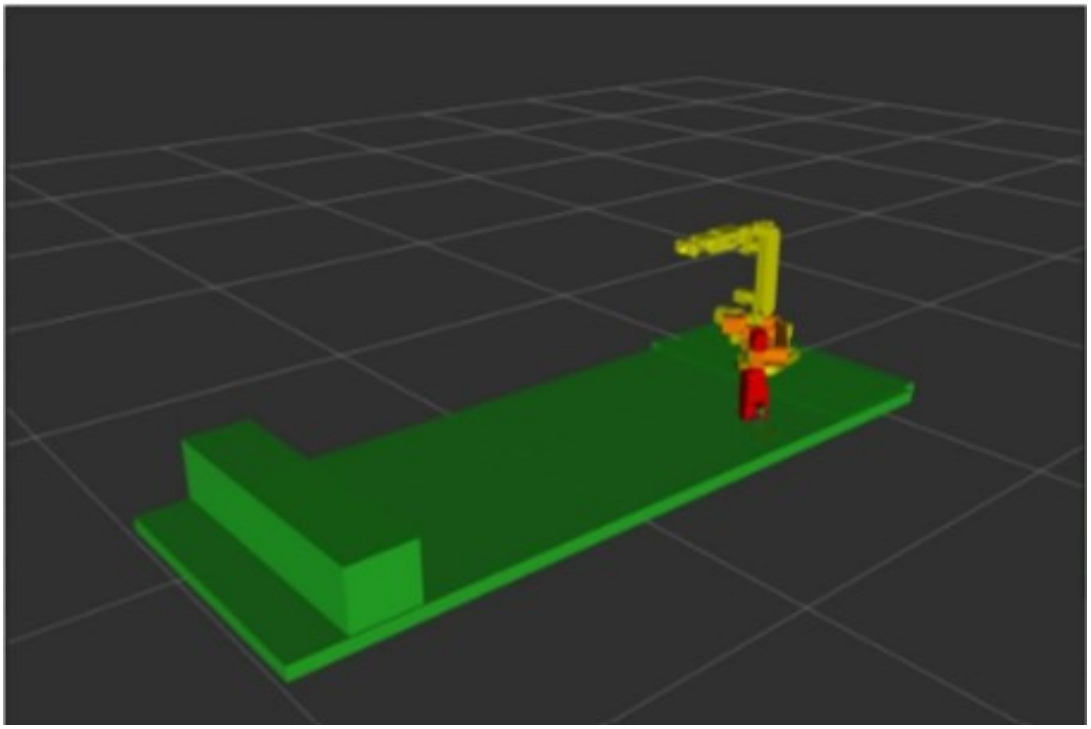


Figure E.5. Visualization of robotic arm control via ROS and MoveIt 2 simulation.

Appendix F: Computer Vision System

The first revision of the computer vision code, designed to determine the spatial relationship between protein slides containing biological samples and the robotic arm, has been completed. The primary objective of this software subsystem is to capture the environment accurately, thereby facilitating the analysis of light refraction at varying angles and distances as the robotic arm maneuvers the camera.

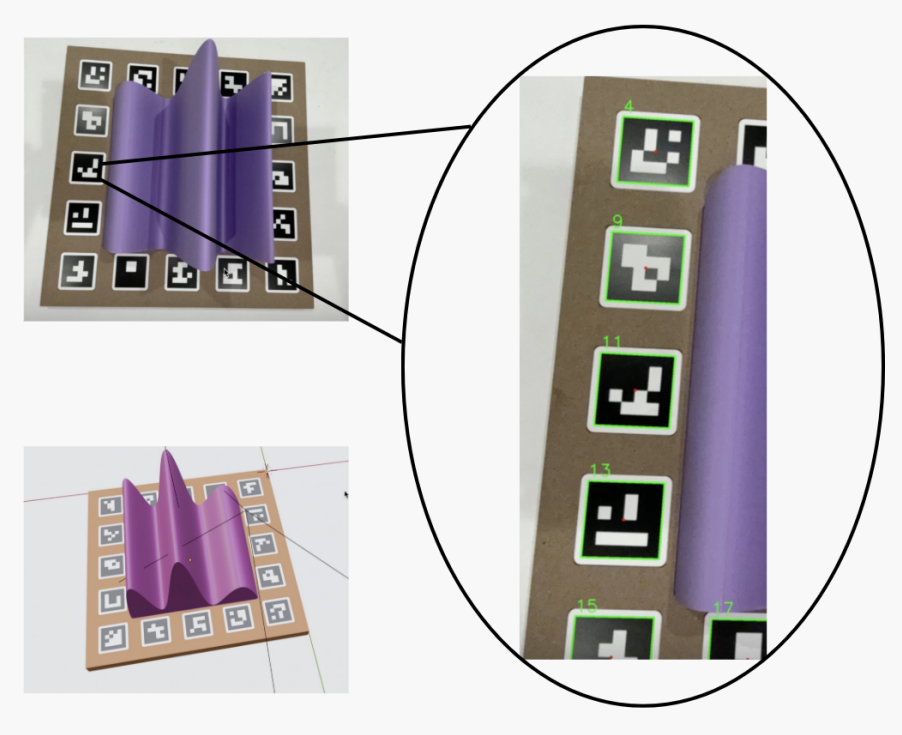


Figure F.6. Alignment algorithm ArUco detection and 3D scene reconstruction.

The initial depth maps generated directly by the iPhone's camera were inadequate for meaningful analysis, as illustrated in Figure F.7. However, through integration of advanced computer vision techniques leveraging ArUco fiducial markers, the system achieves significantly enhanced depth map accuracy (Figure F.8).

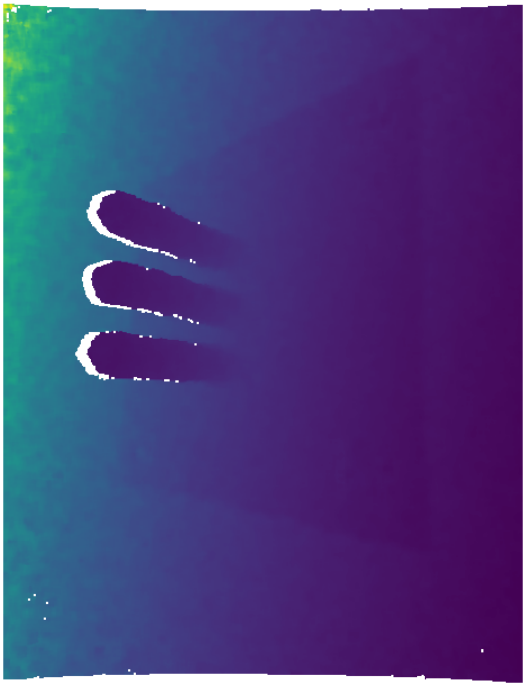


Figure F.7. Initial depth map captured by the iPhone (insufficient quality).

Figure F.8. Refined depth map generated via computer vision techniques.

Using paired depth-map sets, the implemented computer vision algorithms train the system to denoise and reconstruct depth data, substantially improving the accuracy and reliability of captured data. Crucially, the accuracy of the system is no longer mechanically constrained, as positional inaccuracies inherent to the robot's physical movements (such as translation and rotation) are explicitly measured and compensated for via image-based computer vision calculations.

The source code utilized for object detection, 3D reconstruction, and visualization is available on **GitHub**. Additionally, a comprehensive video overview of the software implementation can be found on **Loom**.

Key benefits of integrating the computer vision subsystem into the robotic arm prototype include:

- High-precision robotic movements without extensive mechanical tuning.
- Mitigation of mechanical issues such as gearbox backlash and accumulated errors common in open-loop control systems.
- Software-driven correction of depth and positional inaccuracies, removing dependency on precise physical mounting and alignment.

Appendix G: Operating Instructions

1. **Power-On Sequence**
   (a) Switch on 24 V DC supply (motors/pump) and 5V logic rail.
   (b) Verify Teensy heartbeat LED and ROS PC link-status LEDs are solid.
2. **Network & Software Initialization**
   (a) Connect the iPhone and ROS PC to the dedicated Wi-Fi SSID.
   (b) On the ROS PC, run: `ros2 launch annin_ar4_moveit_config bringup.launch.py`.
   (c) Start the Python coordination server: `python main.py`.
3. **iPhone Capture App**
   (a) Open the *SlideCapture* app, enter server IP, tap **Connect**.
   (b) Confirm "WebSocket Connected" status banner.
4. **System Homing & Calibration**
   (a) In RViz, click **Home All**. Ensure joints reach zero without alarms.
   (b) Run `ros2 service call /aruco_calibrate` to update camera–slide transform.
5. **Load Slides**
   (a) Place tantalum slides in the stage wells; orient fiducial side up.
   (b) Press the stage clamp lever until it locks.
6. **Start Capture Routine**
   (a) On the ROS PC, execute: `ros2 launch capture_routine.launch.py`.
   (b) Monitor vacuum gauge; ensure –60 kPa seal is achieved within 1s per pick.
   (c) Observe live RGB/depth preview in the iPhone app for focus/exposure.
7. **Data Verification**
   (a) Upon completion, open `/data/session_latest` and inspect a random RGB–depth pair.
   (b) Verify ArUco pose error in log¡0.5 mm RMS; re-calibrate if exceeded.
8. **Cloud Sync**
   (a) Trigger upload: `python upload_to_s3.py --session session_latest`.
   (b) Confirm AWS S3 console shows matching file count; check MD5 hash log.
9. **Shutdown Procedure**
   (a) Close the iPhone app (double-tap *Stop Capture* → *Disconnect*).

(b) In ROS, execute `ros2 service call /shutdown`.

(c) Turn off 24 V, then 5 V supplies; disconnect compressed-air line if servicing gripper.

10. **Daily Maintenance**

(a) Wipe slide stage with IPA; inspect vacuum pad for debris.

(b) Backup ROS bag and calibration YAML to NAS.

Appendix H: Assembly Instructions

1. **Inventory & Tools**
   (a) Verify all mechanical parts (gantry rails, fasteners, AR4 robotic arm, 3-D–printed gripper adapter, vacuum pad, barbed fittings, cable chain, iPhone mount, slide stage, ArUco boards).
   (b) Verify electrical items (Teensy 4.1, Arduino Nano, power supplies, stepper drivers, limit switches, wiring harness, RJ-45/USB cables).
   (c) Verify pneumatics (vacuum pump, 6 mm PU tubing, check valve).
   (d) Required tools: metric hex keys (M2–M6), Torx bits, #1 Phillips, torque wrench, side-cutters, zip-ties, Loctite 242, crimpers.

2. **Gantry Frame**
   (a) Fasten X-axis linear rails to the aluminium baseplate (use M5 socket-head screws + Loctite).
   (b) Install the X-axis stepper, pulley, and timing belt; tension belt to 2–3 mm deflection over 100 mm span.
   (c) Attach Y-axis carriage to X-axis slider; mount Y-axis rail and repeat belt installation.
   (d) Fit cable chain to the gantry spine; leave 100 mm slack at each end.

3. **Robotic Arm Mounting**
   (a) Secure the AR4 arm base to the Y-axis carriage using four M8 bolts (torque 25 N·m).
   (b) Verify arm verticality with a machinist's square; shim if deviation >0.25°.

4. **Vacuum Gripper Assembly**
   (a) Thread the M6 vacuum pad into the 3-D–printed adapter; apply a drop of Loctite 242.
   (b) Attach adapter to the arm end-effector using a 12 mm M4 cap screw; torque 2 N·m.
   (c) Push-fit 6 mm PU tubing onto the pad's barbed connector; secure with a zip-tie.

5. **Pneumatic Routing**
   (a) Route tubing through the cable chain to the vacuum pump; avoid ¿ 150 mm bend radius.
   (b) Insert an in-line check valve 150 mm upstream of the gripper.
   (c) Mount the pump beneath the baseplate with rubber grommets; connect to 24V DC supply.

6. **Slide Stage & Fiducials**
   (a) Fix the aluminium slide holder at the gantry datum using countersunk M4 screws.
   (b) Affix ArUco boards at the four corners of the stage (adhesive backing, ensure coplanarity).
   (c) Verify board spacing equals CAD model (±1 mm) for accurate pose estimation.

7. **iPhone Camera Mount**
   (a) Install the magnetic iPhone clamp to the gripper adapter (opposite the pad).
   (b) Place iPhone so rear lenses face the slide stage; tighten clamp to finger-tight plus 1/8 turn.
8. **Electrical Integration**
   (a) Terminate stepper motors to drivers; observe colour code per AR4 wiring diagram.
   (b) Connect limit switches to Teensy GPIO; test continuity.
   (c) Provide 24V @ 10A to motors/pump and 5V @ 3A to logic rails; common ground star-point.
9. **Firmware & ROS Setup**
   (a) Flash `annin_ar4_firmware` to Teensy and Nano (PlatformIO).
   (b) On the ROS 2 PC (Ubuntu 22.04), clone `annin_ar4` workspace, build with `colcon`.
   (c) Launch `moveit_config` demo; confirm joint-state feedback in RViz.
10. **Coordinator Server & iOS App**
   (a) Start the Python coordination server: `python main.py--port-8765`.
   (b) Install the capture app on the iPhone; set server IP and Wi-Fi credentials.
   (c) Verify WebSocket handshake appears in server log.
11. **Calibration & Homing**
   (a) Home each joint via ROS service call; ensure limit switches trigger correctly.
   (b) Run the custom calibration node to record transformation between ArUco origin and TCP.
   (c) Store calibration YAML in `/.ros/calibration/`.
12. **Dry-Run Validation**
   (a) Execute the `test_pick_place.launch.py` script without slides; observe gripper seal and release.
   (b) Check that vacuum reaches 60kPa within 0.8s; adjust pump regulator if necessary.
   (c) Review live depth frames; ensure ArUco pose error¡0.5mm RMS.
13. **Operational Test**
   (a) Load five tantalum slides into the stage; start the `capture_routine`.
   (b) Confirm sequential RGB–depth pairs are saved to `/data/session`.
   (c) Inspect first capture set for focus, exposure, and depth artifacts; iterate settings if required.
14. **Data Sync & Backup**
   (a) Verify S3 bucket upload via REST API completes with HTTP-200 response.

(b) Export ROS bag for the session; compress and archive to project NAS.

15. **Final Acceptance**

(a) Run the 30-slide automated protocol; target completion time ¡4h.

(b) Sign off checklist for mechanical, electrical, and software subsystems.

(c) Document calibration constants and firmware hashes in the project logbook.

Appendix I: Safety Requirements

The AR4-MK3 robotic arm, although smaller than industrial models, has enough torque to pose safety risks if the arm overextends. To mitigate hazards, operators must follow the safety standards outlined in Table I.3.

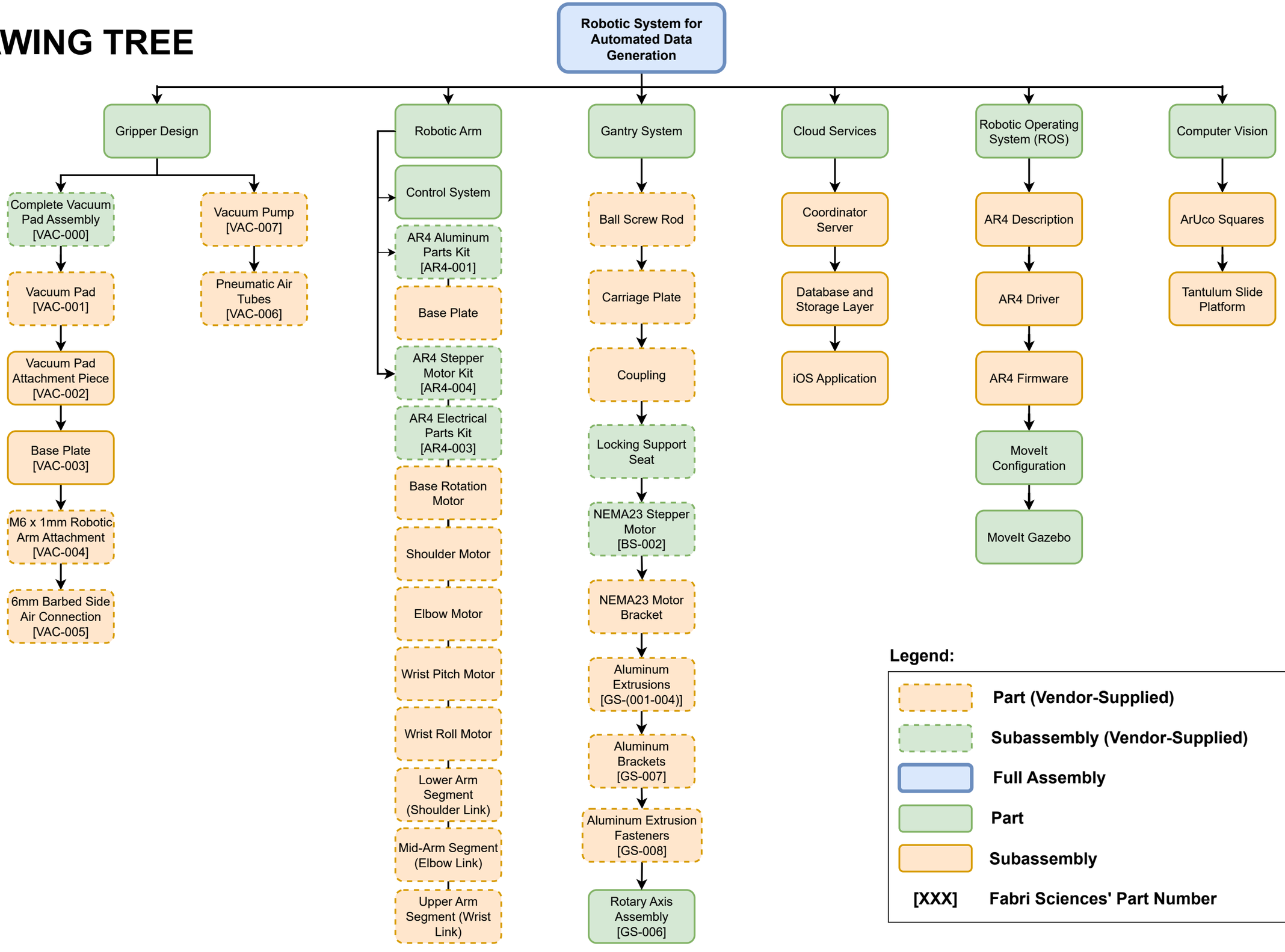
Table I.3: Safety standards for the robotic arm project.

Standard	Title	Application
Canadian Standards Association (CSA)	CSA 60601-1	Section 9.2.2.4: Install physical guards to prevent contact with moving parts. Section 9.2.2.6: Implement speed limits to prevent overextension and reduce collisions. Section 9.2.4: Provide easily accessible emergency stop buttons near operators.
Occupational Health and Safety (OHS)	OHS Code (2009)	Section 209.1(2): Ensure secure installation of the robot in its designated workspace. Section 212(1): Ensure complete power isolation before working on the robots. Section 317: Implement fail-safe mechanisms for unexpected power or system failures.
ISO/TS 15066	Collaborative Robot Safety	Section 5.3: Monitor force and speed of the robotic arm and gripper to prevent injuries. Section 5.4: Ensure emergency stop mechanisms meet collaborative robot standards. Section 6.2: Clearly mark collaborative zones to prevent unintended human-robot interactions.
ANSI/RIA R15.06	Industrial Robot Safety	Section 5.5: Perform risk assessments for custom hardware and software components. Section 6.1: Install safeguards on moving parts to prevent unintended arm over-extension.

Appendix J: Drawing Tree and Detailed Design Drawings

J.1 Drawing Tree

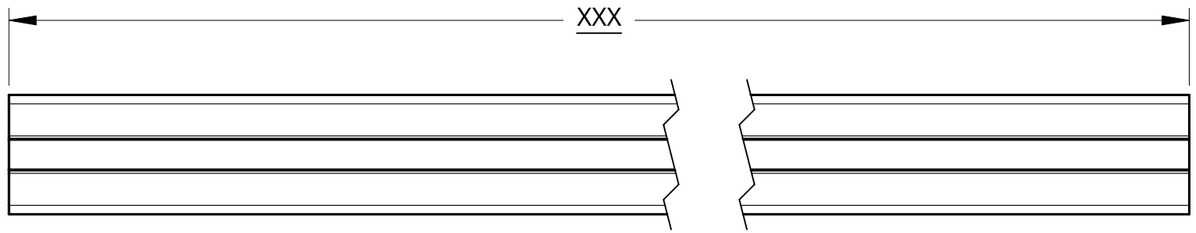
DRAWING TREE

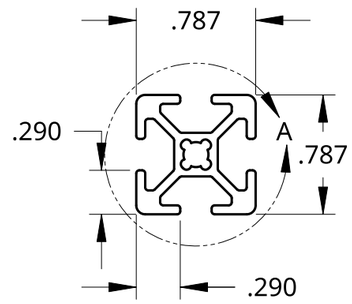


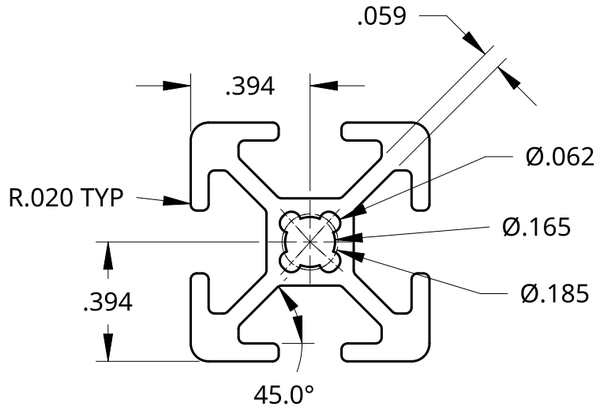
Legend:

	Part (Vendor-Supplied)
	Subassembly (Vendor-Supplied)
	Full Assembly
	Part
	Subassembly
[XXX]	**Fabri Sciences' Part Number**

J.2 Design Drawings







DETAIL A
SCALE 2:1

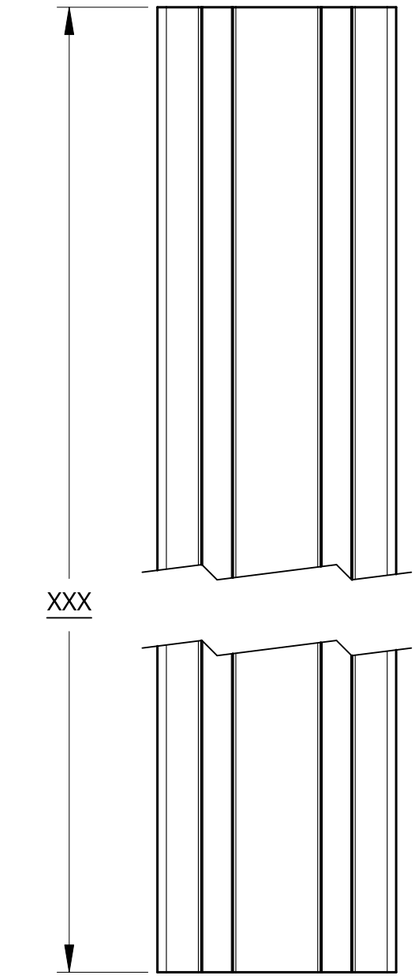
UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN INCHES .X = ±.1 .XX = ±.01 ANGULAR = ± 1° .XXX = ±.005 SURFACE FINISH ▲ 125		NAME	DATE
	DRAWN	WESLEY EZE	03/25/2025
	CHECKED		
	APPROVED		
DO NOT SCALE DRAWING	MANUFACTURED		
INSTRUCTOR: PROF DEJONG			
THIRD ANGLE PROJECTION	MATERIAL: ALUMINUM - 2020	FINISH	

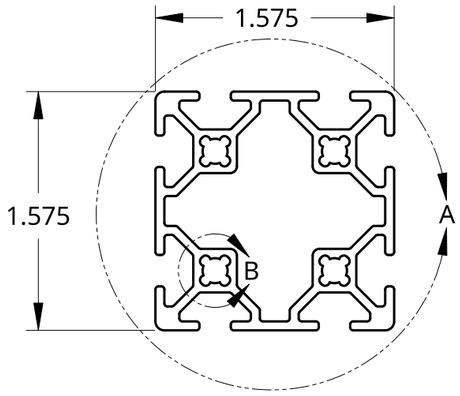
Mechanical Engineering — FABRI SCIENCES INC.

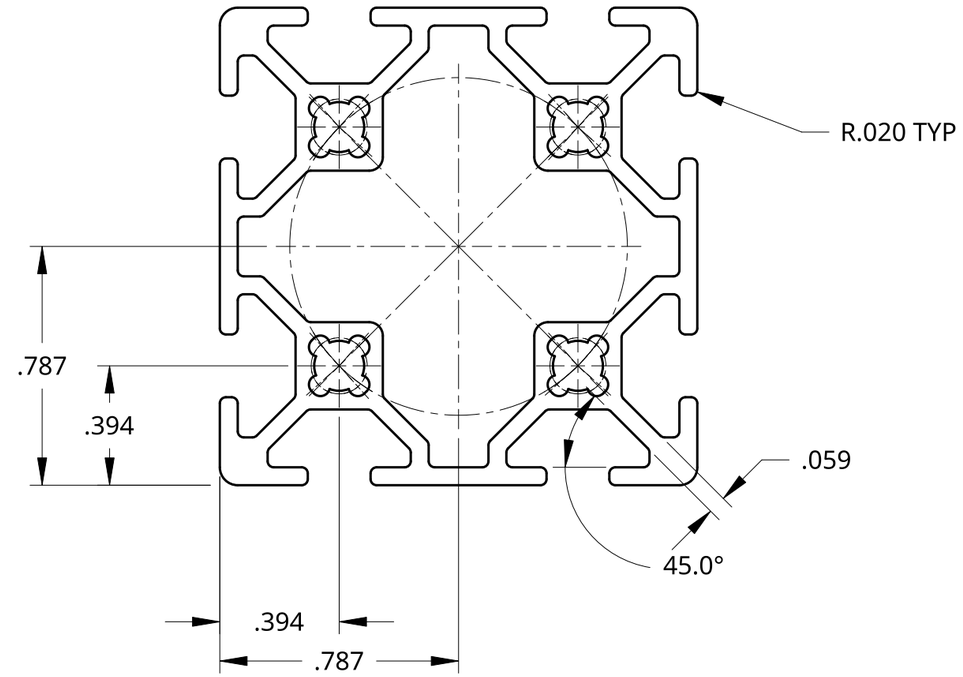
TITLE: 20 20XX - XXX MM

SIZE	REVISION	REV.
B		

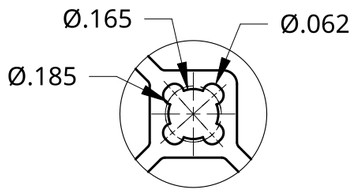
SCALE	WEIGHT	SHEET
1:1	3.307 LB	1 of 1







DETAIL A
SCALE 2:1



DETAIL B
SCALE 2:1

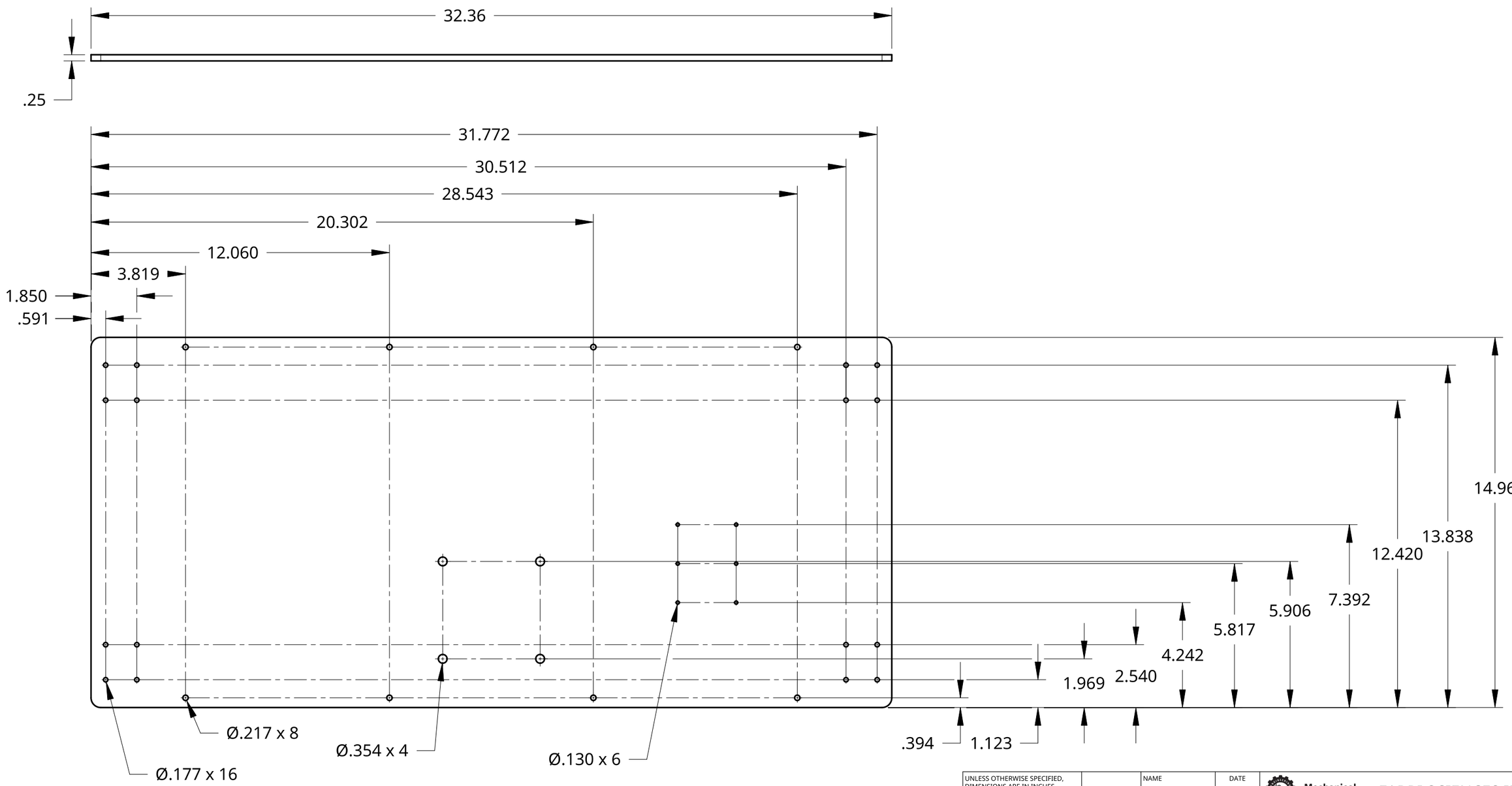
UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN INCHES .X = ±.1 .XX = ±.01 ANGULAR = ± 1° .XXX = ±.005 SURFACE FINISH ▲ 125		NAME	DATE
	DRAWN	WESLEY EZE	03/25/2025
	CHECKED		
	APPROVED		
DO NOT SCALE DRAWING	MANUFACTURED		
INSTRUCTOR: PROF DEJONG			
THIRD ANGLE PROJECTION	MATERIAL: ALUMINUM - 4040	FINISH	

Mechanical Engineering — FABRI SCIENCES INC.

TITLE: 20 40XX - XXX MM

SIZE	REVISION	REV.
B		

SCALE	WEIGHT	SHEET
1:1	25.818 LB	1 of 1



UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN INCHES .X = ±.1 .XX = ±.01 ANGULAR = ± 1° .XXX = ±.005 SURFACE FINISH ▲ 125		
DO NOT SCALE DRAWING		
INSTRUCTOR: PROF DEJONG		
THIRD ANGLE PROJECTION		

	NAME	DATE
DRAWN	WESLEY EZE	03/25/2025
CHECKED		
APPROVED		
MANUFACTURED		

MATERIAL	IRON
FINISH	

Mechanical Engineering

FABRI SCIENCES INC.

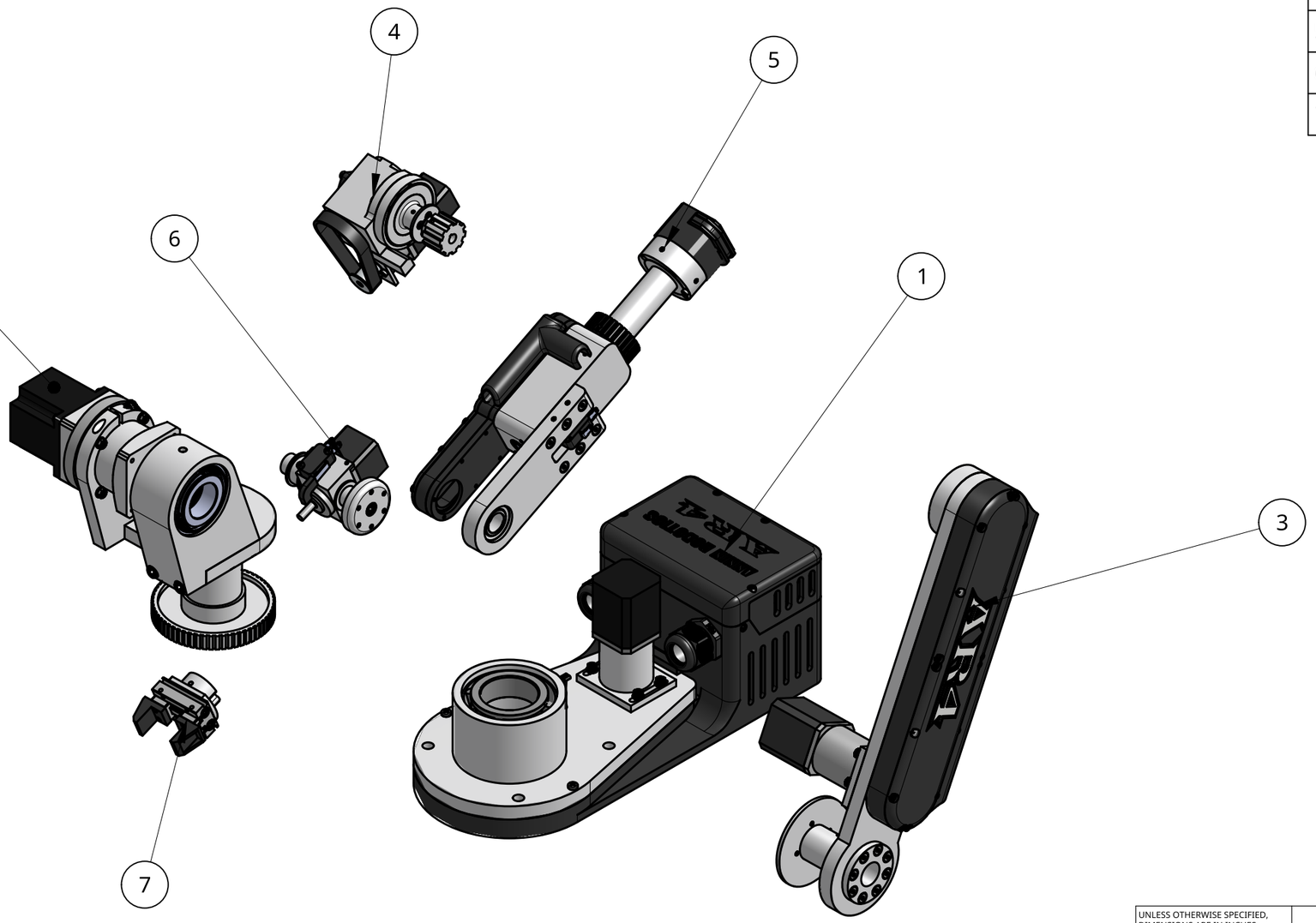
TITLE

AR4 PLATE V3 V1

SIZE	REVISION	REV.
B		

SCALE	WEIGHT	SHEET
1:4	34.235 LB	1 of 1

Item	Quantity	Name
1	1	Linear & Rotary Connection Plate
2	1	Link 1
3	1	Link 2
4	1	Link 3
5	1	Link 4
6	1	Link 5
7	1	Link 6



UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN INCHES
.X = ±.1
.XX = ±.01 ANGULAR = ± 1°
.XXX = ±.005
SURFACE FINISH ▲ 125

DO NOT SCALE DRAWING

INSTRUCTOR: PROF DEJONG

THIRD ANGLE PROJECTION

	NAME	DATE
DRAWN	WESLEY EZE	03/25/2025
CHECKED		
APPROVED		
MANUFACTURED		

MATERIAL: N/A

FINISH:

Mechanical Engineering

FABRI SCIENCES INC.

TITLE: AR4-MK3

SIZE: B

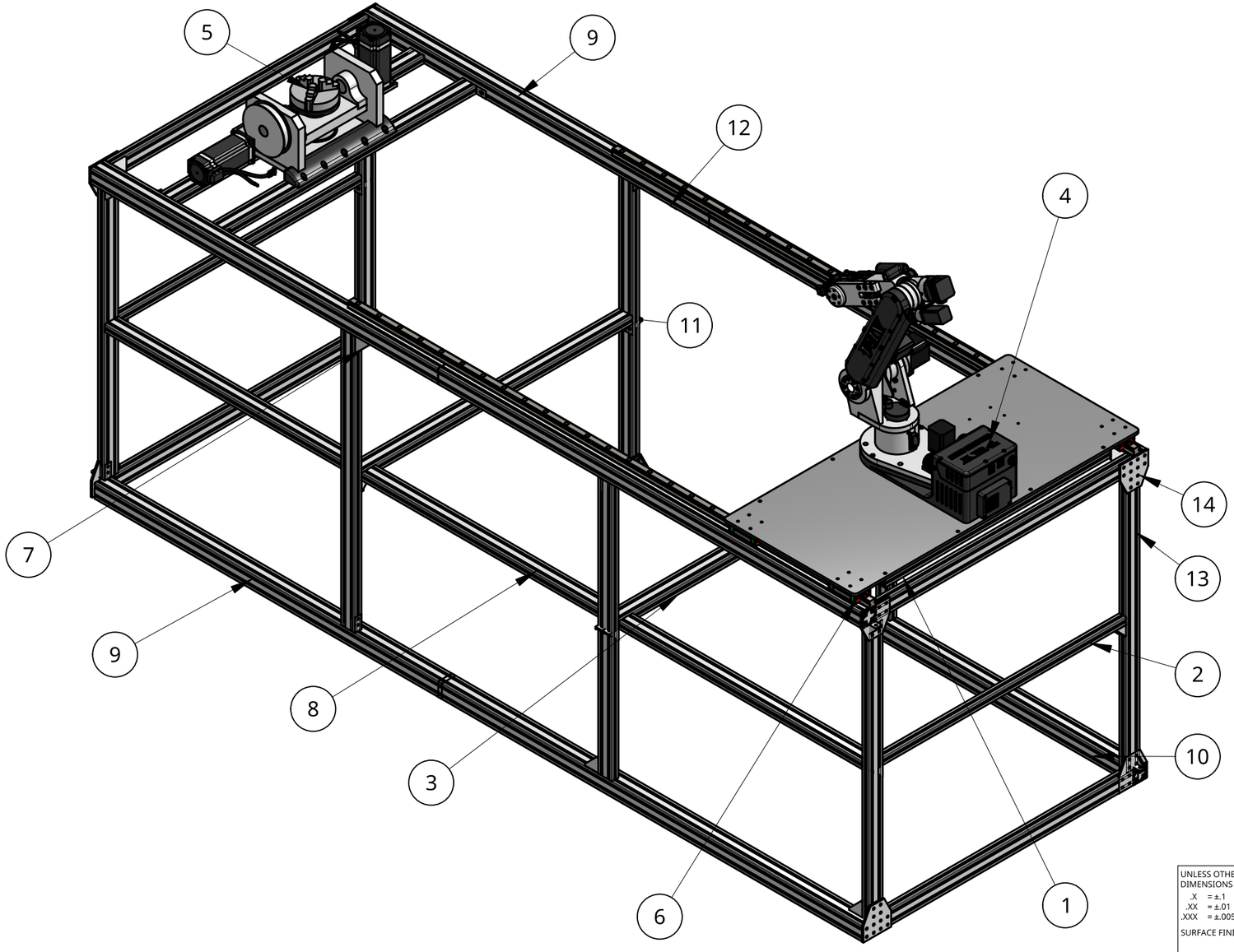
REVISION:

REV.:

SCALE: 1:4

WEIGHT:

SHEET: 1 of 1



Item	Quantity	Name	Material
1	2	20 2020 - 648 mm	Aluminum - 2020
2	2	20 2020 - 700 mm	Aluminum - 2020
3	2	20 2040 - 780 mm	Aluminum - 2040
4	1	AR4-MK3	N/A
5	1	Rotary Axis Assembly	N/A
6	1	Linear Rail Subassembly	N/A
7	20	Large Bracket	Aluminum - 2020
8	3	20 2040 - 700 mm	Aluminum - 2040
9	4	20 4040 - 1000 mm	Aluminum - 4040
10	4	20 4040 - 1220 mm	Aluminum - 4040
11	16	Medium Bracket	Aluminum - 2020
12	8	Small Bracket	Aluminum - 2020
13	14	20 2040 - 740 mm	Aluminum - 2040
14	8	Spacer Plate	Aluminum - 2020

UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN INCHES
.X = ±.1
.XX = ±.01 ANGULAR = ± 1°
.XXX = ±.005
SURFACE FINISH ▲ 125

DO NOT SCALE DRAWING

INSTRUCTOR: PROF DEJONG

THIRD ANGLE PROJECTION

	NAME	DATE
DRAWN	WESLEY EZE	03/25/2025
CHECKED		
APPROVED		
MANUFACTURED		

MATERIAL: N/A

FINISH:

Mechanical Engineering

FABRI SCIENCES INC.

TITLE: FULL ASSEMBLY

SIZE: B

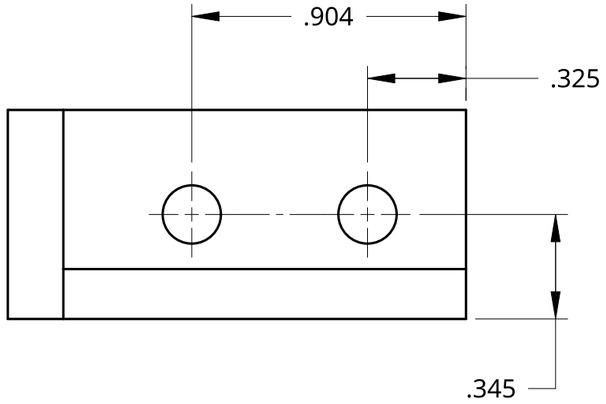
REVISION:

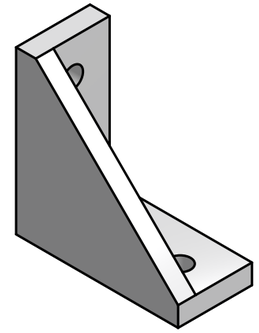
REV.:

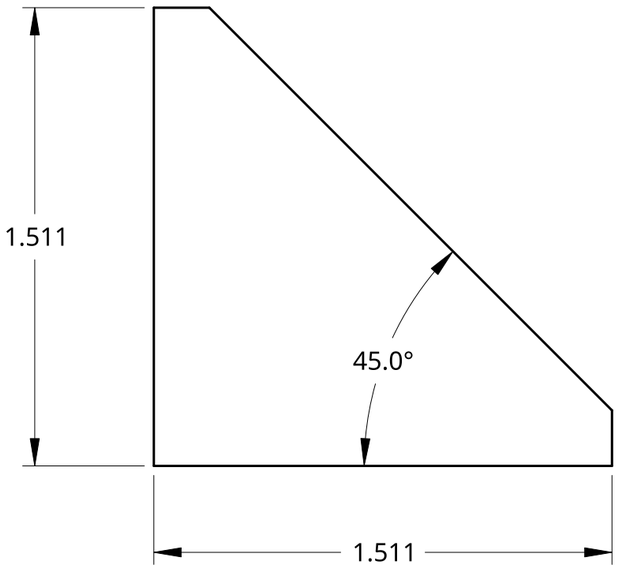
SCALE: 1:10

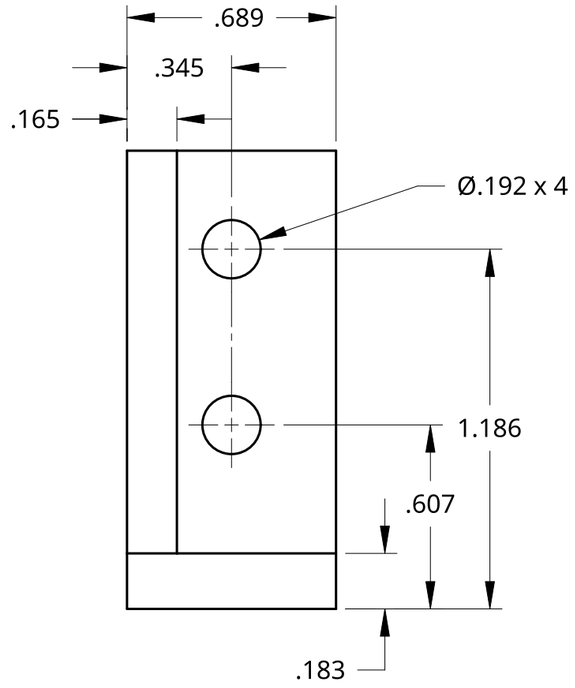
WEIGHT:

SHEET: 1 of 1









UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN INCHES .X = ±.1 .XX = ±.01 ANGULAR = ± 1° .XXX = ±.005 SURFACE FINISH ▲ 125		NAME	DATE
	DRAWN	WESLEY EZE	03/25/2025
	CHECKED		
	APPROVED		
	MANUFACTURED		
DO NOT SCALE DRAWING			
INSTRUCTOR: PROF DEJONG			
THIRD ANGLE PROJECTION	MATERIAL: ALUMINUM - 2020	FINISH:	

Mechanical Engineering

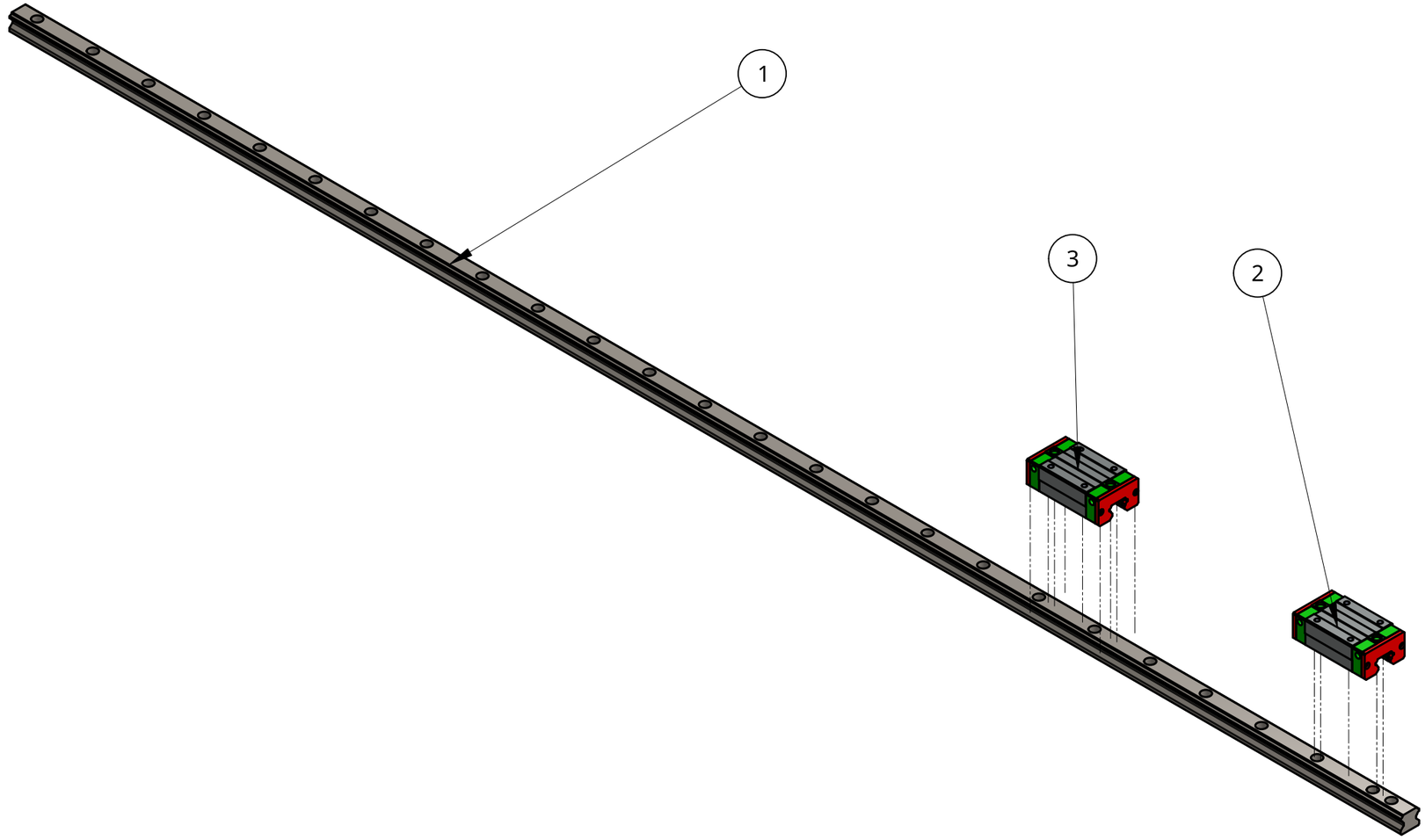
FABRI SCIENCES INC.

TITLE: LARGE BRACKET

SIZE	REVISION	REV.
B		

SCALE	WEIGHT	SHEET
2:1	0.482 LB	1 of 1

Item	Quantity	Name
1	1	HGR20CA Rail - 1500 mm
2	1	HGR20CA Track
3	1	HGR20CA Track



UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN INCHES
.X = ±.1
.XX = ±.01 ANGULAR = ± 1°
.XXX = ±.005
SURFACE FINISH ▲ 125

DO NOT SCALE DRAWING

INSTRUCTOR: PROF DEJONG

THIRD ANGLE PROJECTION

	NAME	DATE
DRAWN	WESLEY EZE	03/25/2025
CHECKED		
APPROVED		
MANUFACTURED		

MATERIAL	FINISH
N/A	

Mechanical Engineering

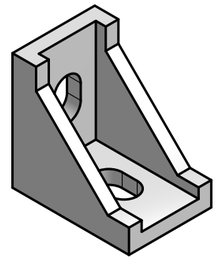
FABRI SCIENCES INC.

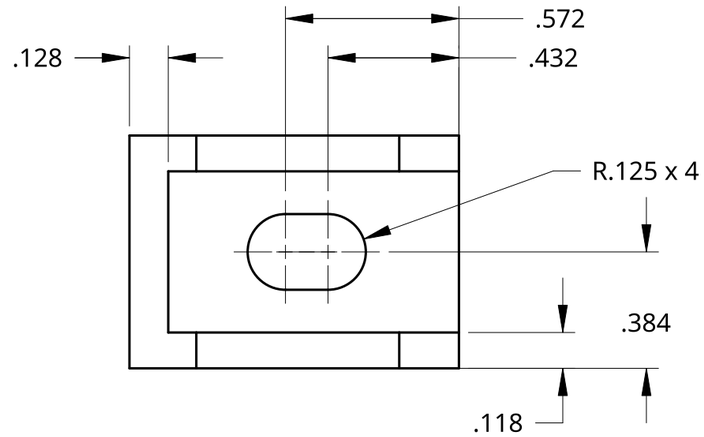
TITLE

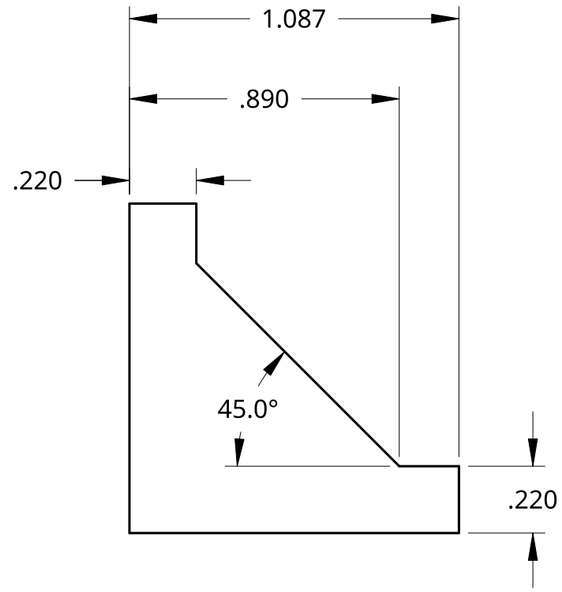
LINEAR RAIL SUBASSEMBLY

SIZE	REVISION	REV.
B		

SCALE	WEIGHT	SHEET
1:4		1 of 1







UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN INCHES
.X = ±.1
.XX = ±.01 ANGULAR = ± 1°
.XXX = ±.005
SURFACE FINISH ▲ 125

DO NOT SCALE DRAWING

INSTRUCTOR: PROF DEJONG

THIRD ANGLE PROJECTION

	NAME	DATE
DRAWN	WESLEY EZE	03/25/2025
CHECKED		
APPROVED		
MANUFACTURED		

MATERIAL: ALUMINUM - 2020

FINISH:

Mechanical Engineering

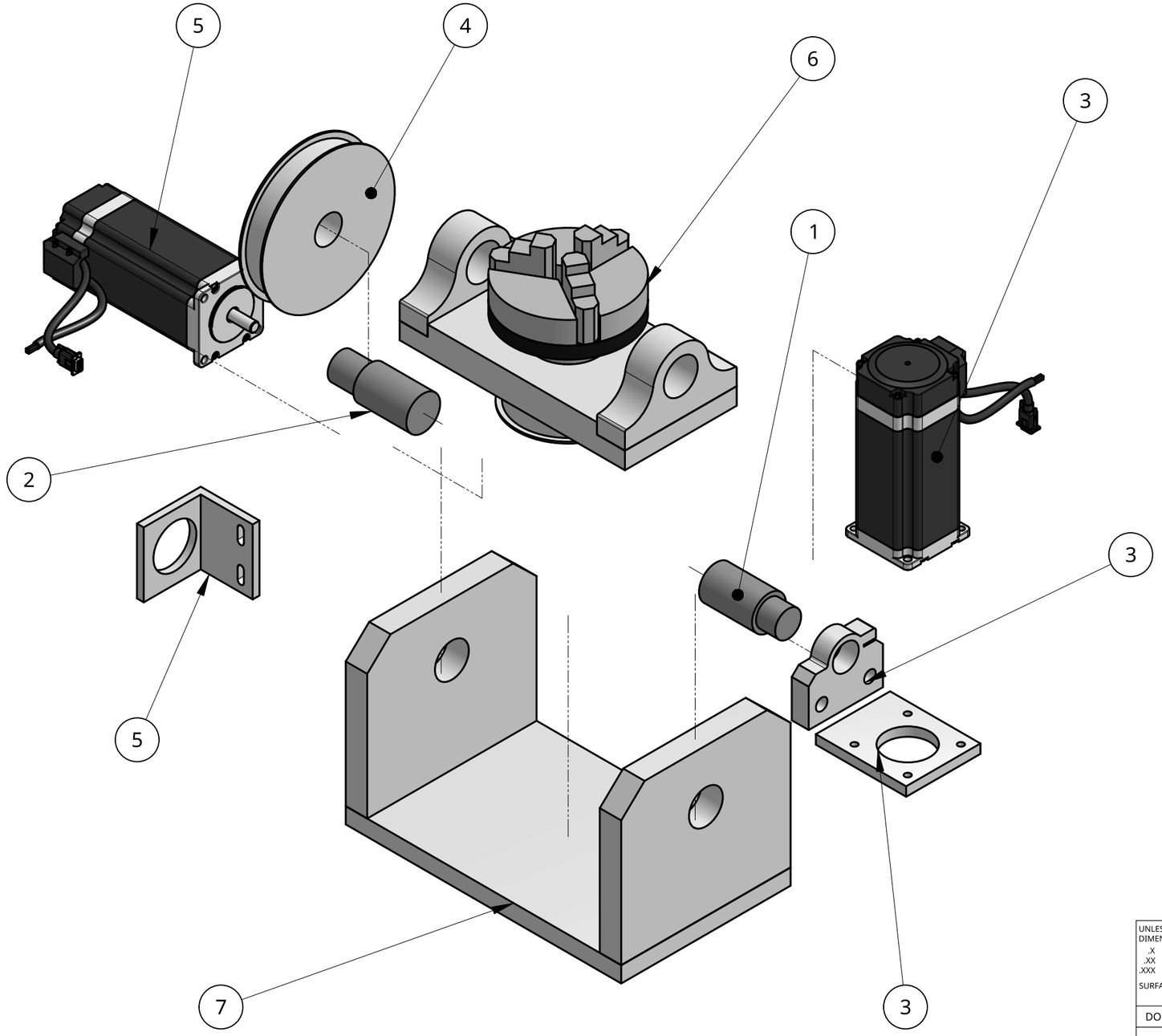
FABRI SCIENCES INC.

TITLE: MEDIUM BRACKET

SIZE: B | REVISION: | REV.:

SCALE: 2:1 | WEIGHT: 0.272 LB | SHEET: 1 of 1

Item	Quantity	Name
1	1	Right Shaft
2	1	Left Shaft
3	1	Mobile Motor v1
4	1	Large Gear
5	1	Fixed Motor Mount v1
6	1	Rotary Table Assembly
7	1	Table Base Plate



UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN INCHES
.X = ±.1
.XX = ±.01 ANGULAR = ± 1°
.XXX = ±.005
SURFACE FINISH ▲ 125

DO NOT SCALE DRAWING

INSTRUCTOR: PROF DEJONG

THIRD ANGLE PROJECTION

	NAME	DATE
DRAWN	WESLEY EZE	03/25/2025
CHECKED		
APPROVED		
MANUFACTURED		

MATERIAL: N/A

FINISH:

Mechanical Engineering

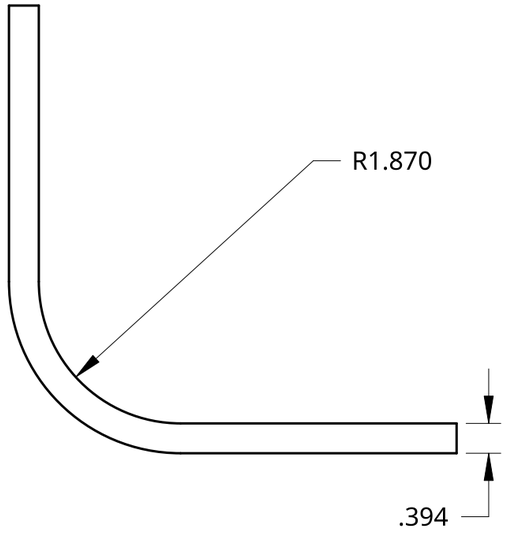
FABRI SCIENCES INC.

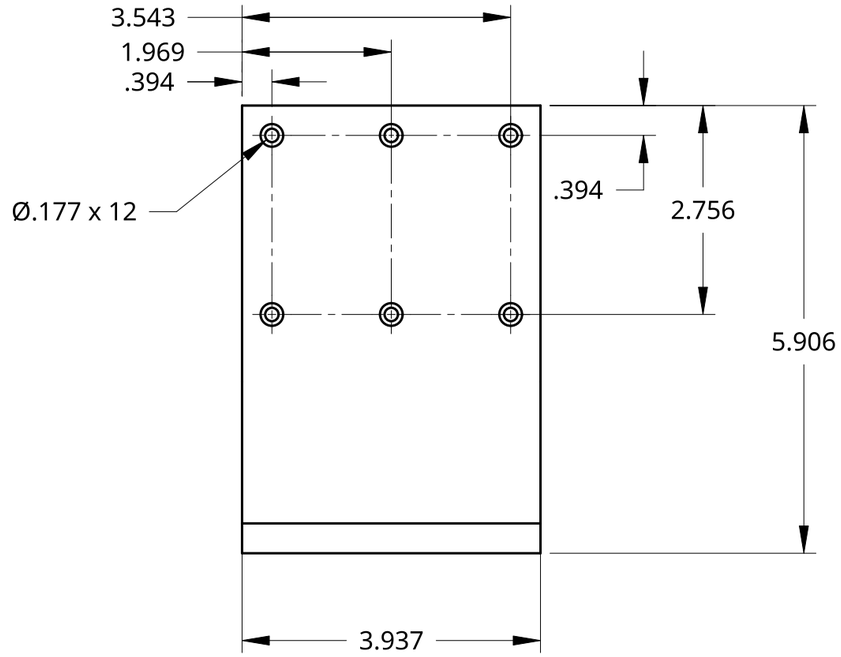
TITLE

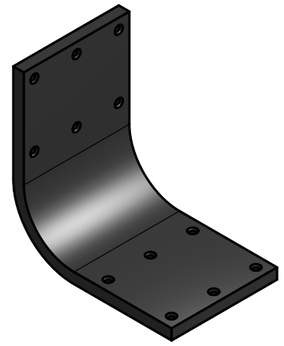
ROTARY AXIS ASSEMBLY

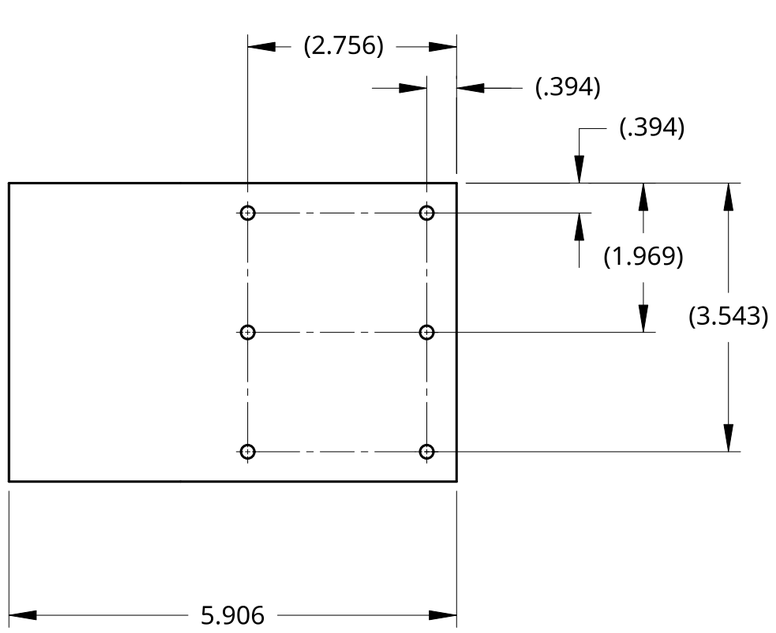
SIZE B | REVISION | REV.

SCALE 1:3 | WEIGHT | SHEET 1 of 1









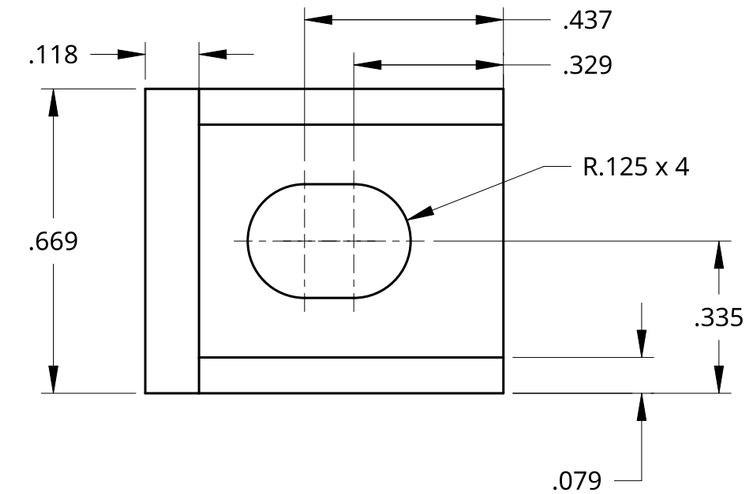
UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN INCHES .X = ±.1 .XX = ±.01 ANGULAR = ± 1° .XXX = ±.005 SURFACE FINISH ▲ 125		NAME	DATE
	DRAWN	WESLEY EZE	03/25/2025
	CHECKED		
	APPROVED		
DO NOT SCALE DRAWING	MANUFACTURED		
INSTRUCTOR: PROF DEJONG			
THIRD ANGLE PROJECTION	MATERIAL: ALUMINUM	FINISH:	

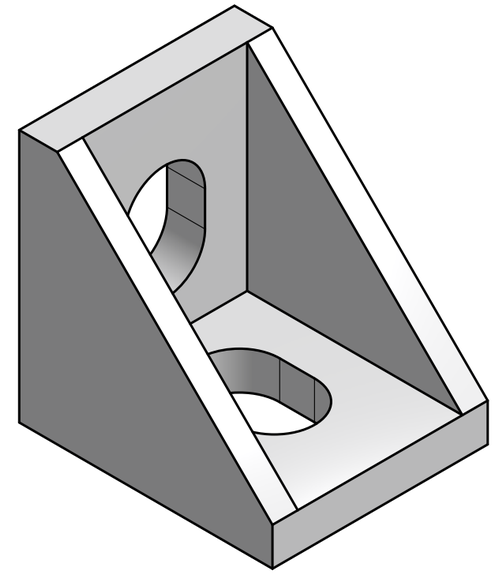
Mechanical Engineering — FABRI SCIENCES INC.

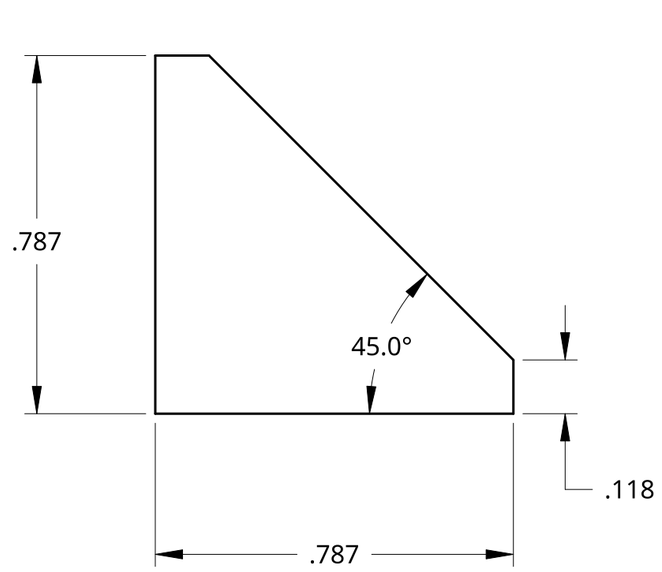
TITLE: ROUND BRACKET

SIZE	REVISION	REV.
B		

SCALE	WEIGHT	SHEET
1:2	1.579 LB	1 of 1







UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN INCHES .X = ±.1 .XX = ±.01 ANGULAR = ± 1° .XXX = ±.005 SURFACE FINISH ▲ 125		NAME	DATE
	DRAWN	WESLEY EZE	03/25/2025
	CHECKED		
	APPROVED		
DO NOT SCALE DRAWING	MANUFACTURED		
INSTRUCTOR: PROF DEJONG			
THIRD ANGLE PROJECTION	MATERIAL: ALUMINUM - 2020	FINISH	

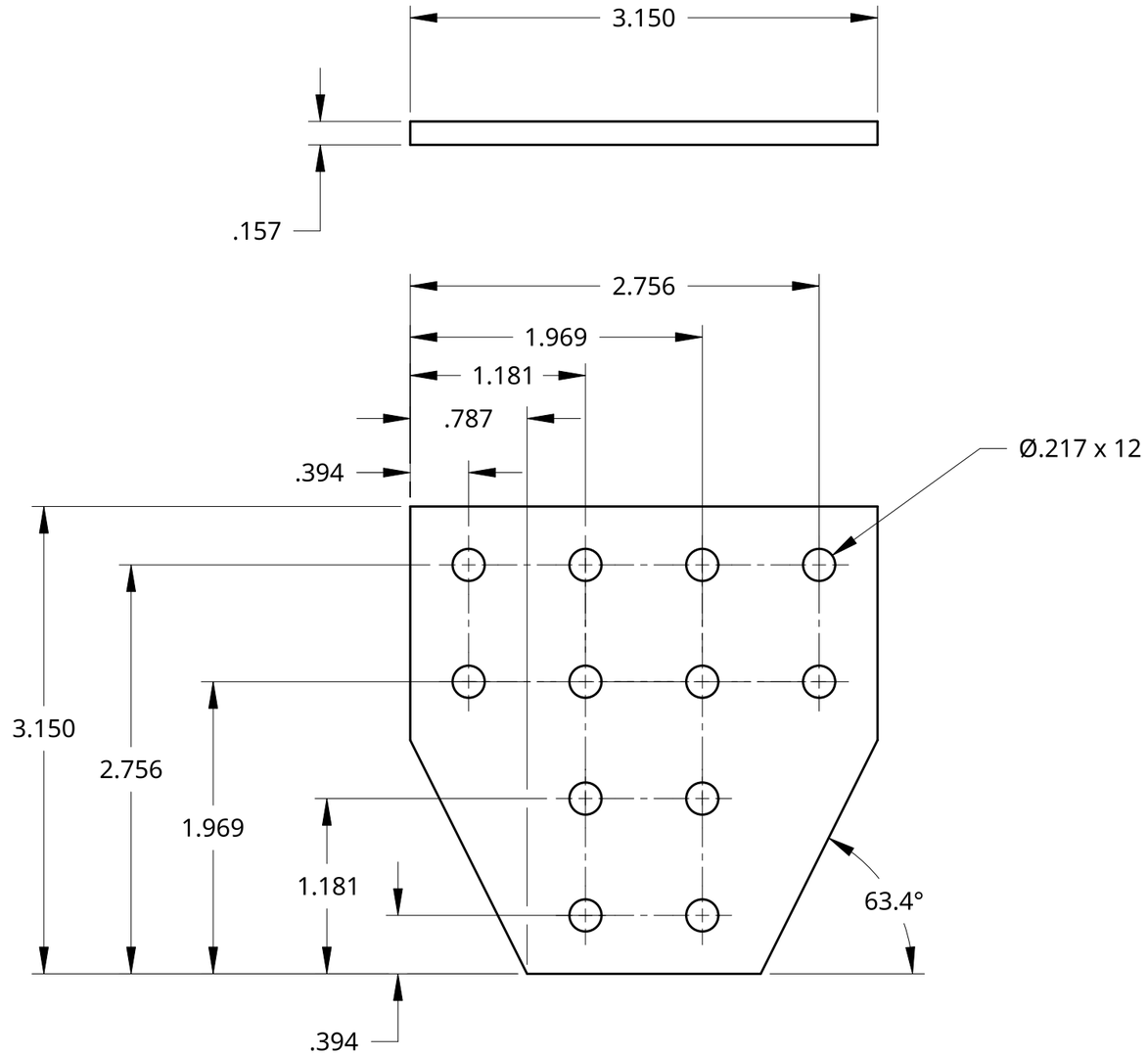
Mechanical Engineering

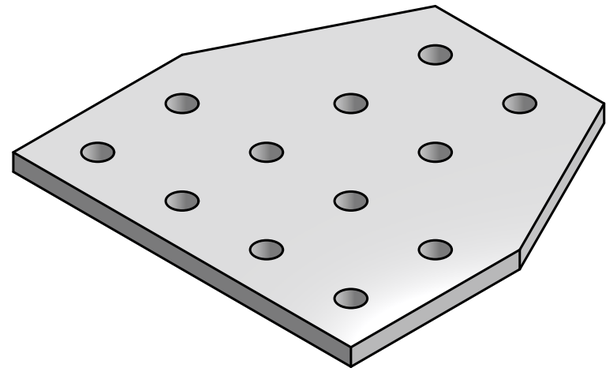
FABRI SCIENCES INC.

TITLE: SMALL BRACKET

SIZE	REVISION	REV.
B		

SCALE	WEIGHT	SHEET
3:1	0.132 LB	1 of 1





UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN INCHES
.X = ±.1
.XX = ±.01 ANGULAR = ± 1°
.XXX = ±.005
SURFACE FINISH ▲ 125

DO NOT SCALE DRAWING

INSTRUCTOR: PROF DEJONG

THIRD ANGLE PROJECTION

	NAME	DATE
DRAWN	WESLEY EZE	03/25/2025
CHECKED		
APPROVED		
MANUFACTURED		

MATERIAL: ALUMINUM - 2020

FINISH:

Mechanical Engineering

FABRI SCIENCES INC.

TITLE: SPACER PLATE

SIZE: B

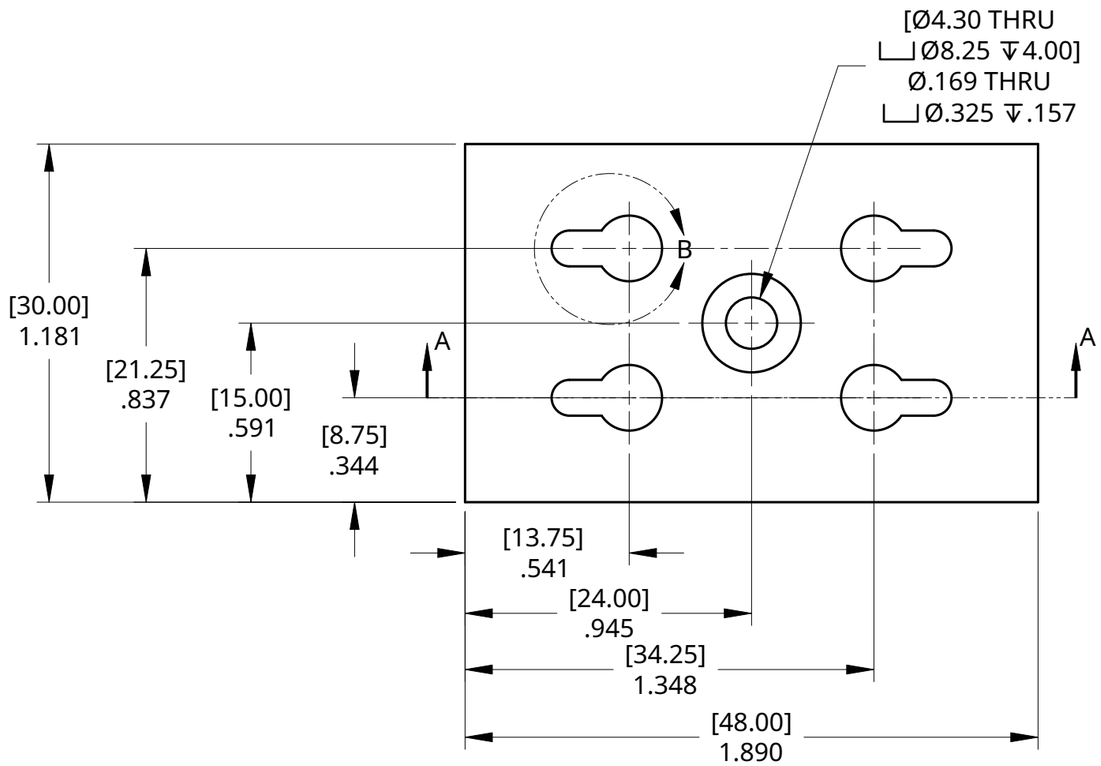
REVISION:

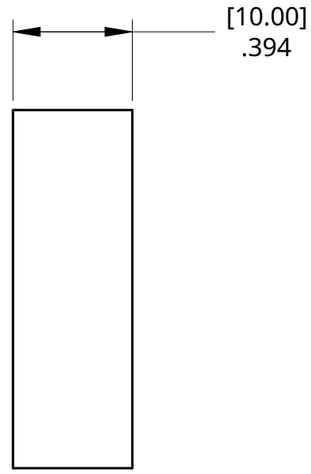
REV.:

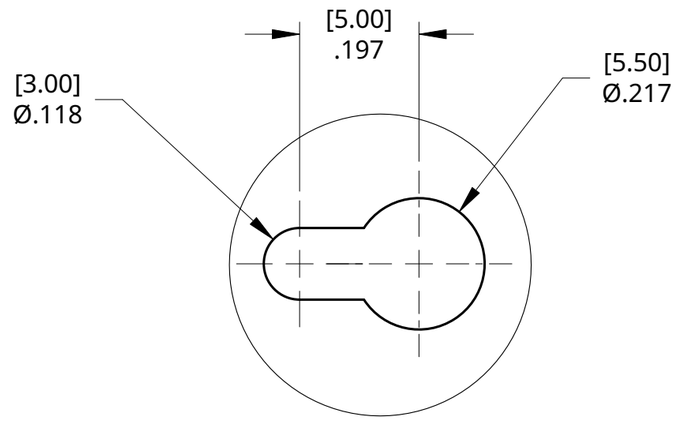
SCALE: 1:1

WEIGHT: 1.297 LB

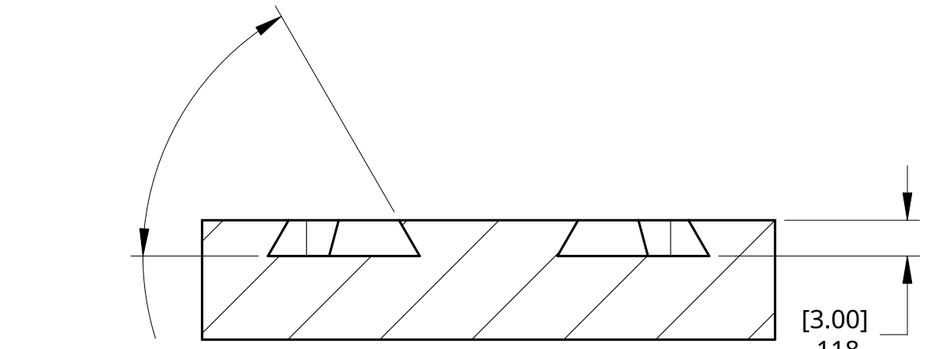
SHEET: 1 of 1







DETAIL B
SCALE 4:1



SECTION A - A

UNLESS OTHERWISE SPECIFIED, DIMENSIONS ARE IN INCHES .X = ±.1 .XX = ±.01 ANGULAR = ± 1° .XXX = ±.005 SURFACE FINISH ▲ 125		NAME	DATE
	DRAWN	WESLEY EZE	04/09/2025
	CHECKED		
	APPROVED		
DO NOT SCALE DRAWING	MANUFACTURED		
INSTRUCTOR: PROF DEJONG			
THIRD ANGLE PROJECTION	MATERIAL: PLA	FINISH	

Mechanical Engineering

FABRI SCIENCES INC.

TITLE: VACUUM CUP ATTACHMENT PIECE

SIZE	REVISION	REV.
B		

SCALE	WEIGHT	SHEET
2:1	0.037 LB	1 of 1

Appendix K: Updated Project Timeline

Fabri Sciences has elected to use Jira as its task management tool of choice. Jira offers a diverse set of project management services to organize deadlines, categorize work, assign tasks, and more. Jira is commonly used in technology-oriented companies as it allows for engineering teams to maintain momentum while working on extremely massive projects.

K.1 Hours Logged Per Person

Over the course of the project, a total of **625 engineering hours** were logged by the team. These hours were distributed across three development phases:

- **Phase I:** 70.5 hours
- **Phase II:** 204.5 hours
- **Phase III:** 350 hours

The following team members contributed to the project: Wesley Eze, Connor Poveledo, Jacob Damant, Minh Luu, Gerlof Bakker, and Eric Petersen. Time contributions varied based on the scope and complexity of each phase, as well as individual roles and responsibilities within the project.

The table below outlines the approximate hours logged by each team member throughout the duration of the project:

Team Member	Approx. Hours Logged
Wesley Eze	105
Connor Poveledo	125
Jacob Damant	120
Eric Petersen	97.5
Minh Luu	87.5
Gerlof Bakker	90
Total	**625**

Table K.4: Approximate engineering hours logged per team member.

This breakdown reflects each team member's contribution to the design, development, and testing phases of the project.

K.2 Project Timeline

	January	February	March	April
Sprints				
M4-52 Letter of Inent				
~~M4-66~~ Write introduction	DONE MINH LUU			
~~M4-83~~ Create project pr...	DONE WESLEY E...			
~~M4-84~~ Write conclusion	DONE GERLOF			
~~M4-85~~ Format report	DONE JACOB DA...			
M4-53 Gripper Protoype				
~~M4-67~~ Develop primitive de...	DONE GERLOF			
~~M4-80~~ Render primitive d...	DONE MINH LUU			
~~M4-81~~ Source competitive...	DONE MINH LUU			
~~M4-142~~ Develop CAD for gripper	DONE			
M4-54 ROS2 Architecture Develop...				
~~M4-68~~ Develop high-l...	DONE CONNORPO...			
~~M4-82~~ Software team fa...	DONE JACOB DA...			
M4-55 Phase 1 Report				
~~M4-69~~ Build cover page	DONE WESLEY E...			
~~M4-86~~ Client letter	DONE WESLEY E...			
~~M4-87~~ Table of contents	DONE WESLEY E...			
~~M4-88~~ Introduction	DONE ERIC PETER...			
~~M4-89~~ Market study	DONE JACOB DA...			
~~M4-90~~ Scope and desi...	DONE CONNORPO...			
~~M4-91~~ Technical requir...	DONE CONNORPO...			
~~M4-92~~ Manufacturing...	DONE CONNORPO...			
~~M4-93~~ Additional compon...	DONE MINH LUU			
~~M4-94~~ Safety requiremen...	DONE MINH LUU			
~~M4-96~~ Decision specificatio...	DONE GERLOF			
~~M4-98~~ Codes and sta...	DONE ERIC PETER...			
~~M4-99~~ Existing patents	DONE ERIC PETER...			
~~M4-100~~ Project manage...	DONE WESLEY E...			
~~M4-102~~ Conclusion	DONE WESLEY E...			
~~M4-104~~ Cost analysis	DONE MINH LUU			
~~M4-106~~ Calculations	DONE MINH LUU			
~~M4-111~~ Project schedule	DONE JACOB DA...			
~~M4-135~~ Phase 1 Peer R...	DONE WESLEY E...			
M4-56 Complete Electronics Assem...				
~~M4-70~~ Source and ord...	DONE CONNORPO...			
~~M4-114~~ Source and or...	DONE CONNORPO...			
~~M4-115~~ Assemble stepp...	DONE JACOB DA...			
~~M4-116~~ Install microcontro...	DONE MINH LUU			
~~M4-118~~ Test all circuits...	DONE CONNORPO...			
M4-57 Complete Mobile App Be...				
~~M4-71~~ Design UI/UX	DONE ERIC PETER...			
~~M4-119~~ Build reusable co...	DONE MINH LUU			
~~M4-120~~ Build injection e...	DONE JACOB DA...			
~~M4-121~~ Build camera m...	DONE JACOB DA...			
~~M4-122~~ Build dashboar...	DONE WESLEY E...			
~~M4-123~~ Build router	DONE GERLOF			
~~M4-124~~ Build dynamic s...	DONE JACOB DA...			
~~M4-137~~ QA / Testing	DONE JACOB DA...			
M4-58 Deploy Cloud Resources				
~~M4-72~~ Build endpoint protec...	DONE GERLOF			
~~M4-126~~ Build API	DONE JACOB DA...			
~~M4-128~~ Build all endpoi...	DONE JACOB DA...			
~~M4-129~~ Build repos iOS...	DONE JACOB DA...			
~~M4-130~~ Integrate endp...	DONE CONNORPO...			
~~M4-131~~ Integrate robot-fa...	DONE MINH LUU			
~~M4-136~~ Test all endpoin...	DONE WESLEY E...			
M4-59 Develop the Computer Vision Pipeline				
~~M4-73~~ Build slide-detectio...	DONE MINH LUU			
~~M4-132~~ Build rectificati...	DONE CONNORPO...			
~~M4-133~~ Build 3D positi...	DONE CONNORPO...			
~~M4-134~~ Build fiducial c...	DONE ERIC PETER...			
M4-60 Phase 2 Report				
~~M4-95~~ Finalized Cost Ana...	DONE MINH LUU			
~~M4-97~~ Design Drawings	DONE GERLOF			
~~M4-101~~ Design Model	DONE CONNORPO...			
~~M4-112~~ Phase 2 Peer R...	DONE WESLEY E...			
~~M4-144~~ Opening Letter	DONE WESLEY E...			
~~M4-145~~ Executive Sum...	DONE WESLEY E...			
~~M4-146~~ Introduction an...	DONE WESLEY E...			
~~M4-147~~ Gripper/Mounting D...	DONE GERLOF			
~~M4-148~~ Magnetic Design	DONE WESLEY E...			
~~M4-149~~ Pneumatic Arms an...	DONE GERLOF			
~~M4-150~~ Suction-Based...	DONE WESLEY E...			
~~M4-152~~ Linear Rail syste...	DONE MINH LUU			
~~M4-153~~ Lead Screw and...	DONE MINH LUU			
~~M4-154~~ Rack and Pinion...	DONE MINH LUU			
~~M4-156~~ AR4-MK3	DONE ERIC PETER...			
~~M4-157~~ MechArm 270	DONE ERIC PETER...			
~~M4-158~~ DFRobot's Ra...	DONE ERIC PETER...			
~~M4-159~~ Gripper/Mounting D...	DONE GERLOF			
~~M4-160~~ Size and Weigh...	DONE WESLEY E...			
~~M4-161~~ Concenpt 1: Ma...	DONE WESLEY E...			
~~M4-162~~ Concept 2: Pneuma...	DONE GERLOF			
~~M4-163~~ Cost Analysis	DONE GERLOF			
~~M4-164~~ Concept 3: Vac...	DONE WESLEY E...			
~~M4-165~~ Precision and...	DONE ERIC PETER...			
~~M4-166~~ Software and C...	DONE JACOB DA...			
~~M4-167~~ Cost Analysis	DONE ERIC PETER...			
~~M4-168~~ Motion Precision...	DONE MINH LUU			
~~M4-169~~ Integration with 6...	DONE MINH LUU			
~~M4-170~~ Cost Analysis	DONE MINH LUU			
~~M4-171~~ Cloud Server	DONE JACOB DA...			
~~M4-172~~ Communication...	DONE JACOB DA...			
~~M4-173~~ Integration and...	DONE JACOB DA...			
~~M4-174~~ Cost and Pricin...	DONE JACOB DA...			
~~M4-175~~ Performance an...	DONE JACOB DA...			
~~M4-176~~ ROS Architect...	DONE CONNORPO...			
~~M4-177~~ Driver Implem...	DONE CONNORPO...			
~~M4-178~~ Simulation Enviro...	DONE MINH LUU			
~~M4-179~~ Computer Visi...	DONE CONNORPO...			
~~M4-180~~ Design Evaluati...	DONE WESLEY E...			
~~M4-181~~ Recommended Des...	DONE GERLOF			
~~M4-182~~ Project Manage...	DONE WESLEY E...			
~~M4-183~~ Updated Design Sp...	DONE GERLOF			
~~M4-184~~ Detailed Calculations	DONE GERLOF			
~~M4-185~~ Strength of Mag...	DONE WESLEY E...			
~~M4-186~~ Theory and Eq...	DONE JACOB DA...			
~~M4-187~~ Strength of Pen...	DONE JACOB DA...			
~~M4-188~~ Updated Projec...	DONE JACOB DA...			
~~M4-189~~ BOM	DONE CONNORPO...			
M4-61 Complete Full Robotic Arm Assem...				
~~M4-75~~ Subsystem inte...	DONE CONNORPO...			
~~M4-138~~ Install gripper	DONE GERLOF			
~~M4-190~~ Write custom st...	DONE JACOB DA...			
M4-62 Design Compliance Mat...				
~~M4-76~~ Build compliance ma...	DONE GERLOF			
M4-63 QA & Testing				
~~M4-140~~ Edge-case pr...	DONE ERIC PETER...			
~~M4-139~~ End-to-end syst...	DONE WESLEY E...			
M4-65 Phase 3 Report				
~~M4-105~~ Cover Letter	DONE WESLEY E...			
~~M4-107~~ Executive Su...	DONE ERIC PETER...			
~~M4-108~~ Main Body	DONE JACOB DA...			
~~M4-109~~ Design Matrix	DONE GERLOF			
~~M4-110~~ Design Calculations	DONE MINH LUU			
~~M4-113~~ Phase 3 Peer...	DONE CONNORPO...			
M4-117 Design Conference				
~~M4-125~~ Product Design...	DONE JACOB DA...			
~~M4-127~~ Presentation	DONE CONNORPO...			

K.3 Project Deliverables

Phase	Key	Epic	Summary	Status	Assignee	Estimated Duration (hrs)	Actual Duration (hrs)
Phase 1	M4-66	Letter of Intent	Write introduction	Done	Minh Luu	0.5	0.5
	M4-83		Create project preference table	Done	Wesley Eze	0.5	0.5
	M4-84		Write conclusion	Done	Gerlof Bakker	0.5	0.5
	M4-85		Format report	Done	Jacob Damant	0.5	0.5
	M4-67	Gripper Prototype	Develop designs for prototype	Done	Connor Povoledo	2	3
	M4-80		Render primitive designs	Done	Gerlof Bakker	1	1.5
	M4-81		Source competitive open-source design alternatives	Done	Minh Luu	2	2
	M4-68	Plan ROS2 Architecture	Develop high-level node architecture	Done	Connor Povoledo	16	15.5
	M4-82		Software team familiarizes themselves with ROS	Done	Jacob Damant	8	10
	M4-69	Phase 1 Report	Build cover page	Done	Wesley Eze	1	0.5
	M4-86		Client letter	Done	Wesley Eze	1	1.5
	M4-87		Table of contents	Done	Wesley Eze	0.5	0.5
	M4-88		Introduction	Done	Eric Petersen	1	1.5
	M4-89		Market study	Done	Jacob Damant	4	5
	M4-90		Scope and design objectives	Done	Connor Povoledo	1	2
	M4-91		Technical requirements	Done	Connor Povoledo	1	1.5
	M4-92		Manufacturing and materials	Done	Connor Povoledo	2	2.5
	M4-93		Additional components to design	Done	Minh Luu	1	1
	M4-94		Safety requirements	Done	Minh Luu	1	1.5
	M4-96		Decision specification matrix	Done	Gerlof Bakker	1	2
	M4-98		Codes and standards	Done	Eric Petersen	1	1.5
	M4-99		Existing patents	Done	Eric Petersen	1	1
	M4-100		Project management	Done	Wesley Eze	1	1.5
	M4-102		Conclusion	Done	Wesley Eze	1	1
	M4-104		Cost analysis	Done	Minh Luu	2	2
	M4-106		Calculations	Done	Minh Luu	1	1
	M4-111		Project schedule	Done	Jacob Damant	2	2
	M4-135		Phase 1 Peer Review	Done	All	5	7
	NA	**TOTAL HOURS**	**Total hours for phase 1**	**NA**	**NA**	**59.5**	**70.5**

Phase 2	M4-70	Complete Electronics Assembly	Source and order steppers	Done	Connor Povoledo	4	5
	M4-114		Source and order cables, grommits, pneumatics, drivers	Done	Connor Povoledo	6	6
	M4-115		Assemble steppers and drivers	Done	Jacob Damant	4	4.5
	M4-116		Install microcontrollers	Done	Minh Luu	2	2.5
	M4-118		Test all circuits and electrical system interfaces	Done	Connor Povoledo	8	8
	M4-71	Complete Mobile App Beta	Design UI/UX	Done	Eric Petersen	3	2.5
	M4-119		Build reusable components	Done	Minh Luu	6	5
	M4-120		Build injection engine	Done	Jacob Damant	6	7
	M4-121		Build camera module	Done	Jacob Damant	16	15
	M4-122		Build dashboard screen	Done	Wesley Eze	6	7
	M4-123		Build router	Done	Gerlof Bakker	4	5
	M4-124		Build dynamic styles	Done	Jacob Damant	3	3.5
	M4-137		QA / Testing	Done	Jacob Damant	6	6
	M4-73	Develop Computer Vision Pipeline	Build slide-detection algorithm	Done	Minh Luu	2	2
	M4-132		Build rectification pipeline	Done	Connor Povoledo	3	3
	M4-133		Build 3D positioning engine	Done	Connor Povoledo	6	5.5
	M4-134		Build fiducial coordination scripts	Done	Eric Petersen	3	3
	M4-95	Phase 2 Report	Finalized Cost Analysis	Done	Minh Luu	3	3
	M4-97		Design Drawings	Done	Gerlof Bakker	8	9
	M4-101		Design Model	Done	Connor Povoledo	10	11
	M4-112		Phase 2 Peer Review	Done	All	6	6
	M4-112		Opening Letter	Done	Wesley Eze	1	1
	M4-144		Executive Summary	Done	Wesley Eze	1	1
	M4-146		Introduction and Project Background	Done	Wesley Eze	1.5	2.5
	M4-147		Gripper/Mounting Design	Done	Gerlof Bakker	2	2
	M4-148		Magnetic Design	Done	Wesley Eze	2	1.5
	M4-149		Pneumatic Arms and Clamps	Done	Gerlof Bakker	2	2
	M4-150		Suction-Based Attachment	Done	Wesley Eze	1.5	1.5
	M4-152		Linear Rail system with Belt Drive	Done	Minh Luu	2	2.5
	M4-153		Lead Screw and Ball Screw Actuated System	Done	Minh Luu	2	3
	M4-154		Rack and Pinion Gantry	Done	Minh Luu	2	2
	M4-156		AR4-MK3 Robotic Arm	Done	Eric Petersen	2.5	3
	M4-157		MechArm 270 Robotic Arm	Done	Eric Petersen	1.5	1.5
	M4-158		UFACTORY xArm 6 Robotic Arm	Done	Eric Petersen	1.5	2.5
	M4-159		Gripper/Mounting Design	Done	Gerlof Bakker	1.5	2
	M4-160		Size and Weight of Device Capabilitiers	Done	Wesley Eze	1.5	1.5
	M4-161		Concept 1: Magnetic Design	Done	Wesley Eze	2	2
	M4-162		Concept 2: Pneumatic Clamping System	Done	Gerlof Bakker	2	2
	M4-163		Cost Analysis	Done	Gerlof Bakker	1	1
	M4-164		Concept 3: Vacuum Mounting System	Done	Wesley Eze	1.5	2
	M4-165		Precision and Stability for Camera Mounting	Done	Eric Petersen	1	2
	M4-166		Software and Communication with Cloud System	Done	Jacob Damant	1.5	2
	M4-167		Cost Analysis	Done	Eric Petersen	1	2
	M4-168		Motion Precision and Load Capabity	Done	Minh Luu	1.5	1.5
	M4-169		Integration with 6 Axis Robot and Control System	Done	Minh Luu	2	2
	M4-170		Cost Analysis	Done	Minh Luu	1	1.5
	M4-171		Cloud Server	Done	Jacob Damant	0.5	1
	M4-190		Capture Configuration	Done	Jacob Damant	1.5	4
	M4-172		Communication and Data Transfer Requirements	Done	Jacob Damant	1	1
	M4-173		System Integration	Done	Jacob Damant	1.5	2.5
	M4-174		Infrastructure Scalability	Done	Jacob Damant	1.5	2
	M4-176		ROS Architecture	Done	Connor Povoledo	2	3
	M4-177		Driver Implementation	Done	Connor Povoledo	1.5	2
	M4-178		Simulation Environment	Done	Minh Luu	1	1
	M4-179		Computer Vision	Done	Connor Povoledo	1	1.5
	M4-180		Design Evaluation Matrix	Done	Wesley Eze	1	1
	M4-181		Recommended Design Concepts	Done	Gerlof Bakker	1	2
	M4-182		Project Management	Done	Wesley Eze	0.5	1
	M4-183		Updated Design Specification Matrix	Done	Gerlof Bakker	1	2
	M4-184		Detailed Calculations	Done	Gerlof Bakker	1	2
	M4-185		Strength of Magnetic Gripper Design	Done	Wesley Eze	2	1.5
	M4-186		Theory and Equations	Done	Wesley Eze	1.5	2
	M4-187		Strength of Penumatic Clamps	Done	Eric Petersen	2	2
	M4-188		Updated Project Timeline	Done	Jacob Damant	1.5	3
	M4-189		BOM	Done	Connor Povoledo	3	3
	NA	**TOTAL HOURS**	**Total hours for phase 2**	**NA**	**NA**	**181**	**204.5**

Phase 3	**M4-72**	Deploy Cloud Resources	Build endpoint protection	Done	Gerlof Bakker	8	12
	M4-126		Build API	Done	Jacob Damant	10	16
	M4-128		Build all endpoints	Done	Jacob Damant	20	26
	M4-129		Build repos iOS app	Done	Jacob Damant	24	30
	M4-130		Integrate endpoints in iOS app	Done	Connor Povoledo	6	9
	M4-131		Integrate robot-facing endpoints in ROS system	Done	Minh Luu	6	12
	M4-136		Test all endpoints	Done	Wesley Eze	6	8
	M4-75	Complete Robotic Arm Assembly	Subsystem integration testing	Done	Connor Povoledo	12	20
	M4-138		Install gripper	Done	Gerlof Bakker	3	6
	M4-139		Write custom stepper libraries	Done	Jacob Damant	8	12
	M4-140	QA / Testing	Edge-case prediction and testing	Done	Eric Petersen	3	4
	M4-139		End-to-end system testing	Done	Wesley Eze	4	4
	M4-105	Phase 3 Report	Cover Letter	Done	Eric Petersen	1	1
	M4-107		Executive Summary	Done	Jacob Damant	1	1
	M4-108		Main Body	Done	All	80	88
	M4-109		Design Matrix	Done	Gerlof Bakker	3	5
	M4-110		Design Calculations	Done	Minh Luu	3	6
	M4-113		Phase 3 Peer Review	Done	All	12	12
	M4-125	Design Conference	Product Design Poster	Done	All	16	15
	M4-128		Desing / render visual assets	Done	All	10	12
	M4-127		Presentation	Done	All	32	48
	NA	**TOTAL HOURS**	**Total hours for phase 3**	**NA**	**NA**	**268**	**347**

Appendix L: Updated Team Charter

The current team charter, detailing roles, responsibilities, and project management protocols, is included in full below. This document supersedes all previous versions and serves as the governing agreement for team operations.

Fabri Sciences Team Charter

Robotic System for Automated Dataset Generation

Prepared by: Wesley Eze
Revised: April 1, 2025

Fabri Sciences – Contact Information

Name	Phone	Email	Specialization
Wesley Eze	587-700-2630	weze@ualberta.ca	Project Management
Quang Minh Luu	587-501-2618	mqluu@ualberta.ca	Robotics Engineering
Jacob Damant	780-340-9738	damant@ualberta.ca	Software Development
Connor Povoledo	587-982-3742	povoledo@ualberta.ca	Systems Engineering
Gerlof Bakker	780-394-7441	gerlof@ualberta.ca	Hardware Design
Eric Petersen	825-419-3742	empeters@ualberta.ca	Computer Vision

About Our Relationship

Group Norms

We consider the following attitudes and behaviours to be important to our group and will strive to uphold these in our work as a group:

- **Entrepreneurship**
- **Accountability**
- **Attention to detail**
- **Innovate to improve other's lives (deontology)**
- **Going beyond expectations**
- **Clear communication**
- **Consequentialism**
- **Supporting one another**

Decision Making Process

Approach to Decision-Making:

- **Irreversible Decisions:** These decisions will be made collaboratively as a group, ensuring all members agree on the final outcome. This promotes alignment with the team's shared goals and ideals.
- **Reversible Decisions:** Decisions related to individual tasks that can be modified later will be made promptly by the responsible individual. This approach enables the team to maintain efficiency and focus on completing the project in a timely manner.

Conflict Resolution:

- **Client-Specific Disagreements:**
  - The team will always prioritize the client's best interests.
  - When proposing solutions, the team will present the most effective option, clearly communicating why specific decisions are being made.
  - If the client requests changes that may not align with optimal practices, the team will provide honest, direct feedback and explain the rationale behind any recommended adjustments.

- **Internal Team Disagreements:**
  - All team members are encouraged to raise concerns about internal issues or challenges with specific group members.
  - These concerns will be discussed in dedicated team meetings, with the goal of collaboratively identifying actionable solutions.
  - A portion of subsequent meetings will be allocated to reviewing and validating that the identified solutions have successfully resolved the issues.

Conflict Resolution Process

Timely Completion of Tasks: Group members are expected to complete assigned tasks promptly to ensure the team can utilize deliverables effectively. If a deliverable is submitted on the day before the deadline or later, and more than 50% of it requires revisions by another team member, the team will escalate the issue by submitting a formal letter to the professor.

Attendance Policy and Escalation Process: To maintain productive collaboration, all team members are expected to participate in group discussions and project meetings. The following escalation process will be followed for repeated absences or lack of participation:

- **First Missed Meeting/Office Hour:**
  - The team member will receive a verbal or written warning, emphasizing the importance of attendance.
  - The team will offer support to address any underlying challenges the member may be facing.
- **Second Missed Meeting/Office Hour:**
  - A second warning will be issued.
  - The member will meet with the team leader (or a designated representative) to discuss challenges and commit to a specific improvement plan.
- **Third Missed Meeting/Office Hour:**
  - The matter will be escalated to the professor or course instructor.
  - A formal letter will be submitted, detailing the team's efforts to resolve the issue and documenting the member's continued lack of participation.
  - The professor will determine any further actions as necessary.

Special Considerations: If a team member anticipates missing a meeting or office hour due to valid reasons (e.g., illness, emergencies), they must notify the team in advance and, if possible, contribute asynchronously to the project. The team will assess each case individually to ensure fairness and accommodate unforeseen circumstances, maintaining respect for all members' challenges.

Guidelines for Communication

Primary Communication Method: Team updates will occur at least every other day to ensure transparency, alignment, and efficient progress toward project goals. Updates may occur more frequently if required by project demands or critical deadlines. These updates will occur through the currently used Google Chat for the group.

Meeting Times and Locations: Work sessions and update meetings will primarily take place in person in the ETLC lecture hall. The group meets on Thursdays at 8 PM, and meetings with our advisor, Dr. Samira Dootsie, are held on Tuesdays at 2 PM. If a majority of team members cannot attend in person, meetings will be hosted online to maintain inclusivity and collaboration.

Additional Notes: Team members are expected to check communication channels regularly and respond promptly to messages or queries. The team will adapt communication methods as needed to address any challenges or preferences that arise during the project.

Project Tasks and Goals

Project Task Breakdown			
Summary	**Status**	**Assignee**	**Completed / Completed**
Epic	**Letter of Intent**	Due: 2025-01-17	Completed
Task	Write introduction	Minh Luu	Completed
Task	Create project preference table	Wesley Eze	Completed
Task	Write conclusion	Gerlof Bakker	Completed
Task	Format report	Jacob Damant	Completed
Epic	**Gripper Prototype**	Due: 2025-02-24	Completed
Task	Develop primitive designs for prototype	Gerlof Bakker	Completed
Task	Render primitive designs	Minh Luu	Completed
Task	Source competitive open-source design alternatives	Minh Luu	Completed
Task	Develop CAD for gripper	Wesley Eze	Completed
Task	Manufacture gripper	Gerlof Bakker	Completed
Epic	**ROS2 Architecture Development**	Due: 2025-02-24	Completed
Task	Develop high-level node architecture	Connor Povoledo	Completed
Task	Software team familiarizes themselves with ROS	Jacob Damant	Completed
Epic	**Phase 1 Report**	Due: 2025-02-09	Completed
Task	Build cover page	Wesley Eze	Completed
Task	Client letter	Wesley Eze	Completed
Task	Table of contents	Wesley Eze	Completed
Task	Introduction	Eric Petersen	Completed
Task	Market study	Jacob Damant	Completed
Task	Scope and design objectives	Connor Povoledo	Completed
Task	Technical requirements	Connor Povoledo	Completed
Task	Manufacturing and materials	Connor Povoledo	Completed
Task	Additional components to design	Minh Luu	Completed
Task	Safety requirements	Minh Luu	Completed
Task	Decision specification matrix	Gerlof Bakker	Completed
Task	Codes and standards	Eric Petersen	Completed
Task	Existing patents	Eric Petersen	Completed
Task	Project management	Wesley Eze	Completed

Task	Conclusion	Wesley Eze	Completed
Task	Cost analysis	Minh Luu	Completed
Task	Calculations	Minh Luu	Completed
Task	Project schedule	Jacob Damant	Completed
Task	Phase 1 Peer Review	Wesley Eze	Completed
Epic	**Complete Electronics Assembly**	Due: 2025-02-24	Completed
Task	Source and order steppers	Connor Povoledo	Completed
Task	Source and order cables, grommits, pneumatics, drivers	Connor Povoledo	Completed
Task	Assemble steppers and drivers	Jacob Damant	Completed
Task	Install microcontrollers	Minh Luu	Completed
Task	Test all circuits and electrical system interfaces	Connor Povoledo	Completed
Epic	**Complete Mobile App Beta**	Due: 2025-02-24	Completed
Task	Design UI/UX	Eric Petersen	Completed
Task	Build reusable components	Minh Luu	Completed
Task	Build injection engine	Jacob Damant	Completed
Task	Build camera module	Jacob Damant	Completed
Task	Build dashboard screen	Wesley Eze	Completed
Task	Build router	Gerlof Bakker	Completed
Task	Build dynamic styles	Jacob Damant	Completed
Task	QA / Testing	Jacob Damant	Completed
Epic	**Deploy Cloud Resources**	Due: 2025-03-02	Completed
Task	Build endpoint protection	Gerlof Bakker	Completed
Task	Build API	Jacob Damant	Completed
Task	Build all endpoints	Jacob Damant	Completed
Task	Build repos iOS app	Jacob Damant	Completed
Task	Integrate endpoints in iOS app	Connor Povoledo	Completed
Task	Integrate robot-facing endpoints in ROS system	Minh Luu	Completed
Task	Test all endpoints	Wesley Eze	Completed
Epic	**Develop the Computer Vision Pipeline**	Due: 2025-03-05	Completed
Task	Build slide-detection algorithm	Minh Luu	Completed
Task	Build rectification pipeline	Connor Povoledo	Completed
Task	Build 3D positioning engine	Connor Povoledo	Completed

Task	Build fiducial coordination scripts	Eric Petersen	Completed
Epic	**Phase 2 Report**	Due: 2025-03-09	Completed
Task	Finalized Cost Analysis	Minh Luu	Completed
Task	Design Drawings	Gerlof Bakker	Completed
Task	Design Model	Connor Povoledo	Completed
Task	Report Writing	Jacob Damant	Completed
Task	Phase 2 Peer Review	Wesley Eze	Completed
Task	Introduction	Eric Petersen	Completed
Epic	**Complete Full Robotic Arm Assembly**	Due: 2025-03-15	Completed
Task	Subsystem integration testing	Connor Povoledo	Completed
Task	Install gripper	Gerlof Bakker	Completed
Epic	**Design Compliance Matrix**	Due: 2025-03-17	Completed
Task	Build compliance matrix	Gerlof Bakker	Completed
Epic	**QA & Testing**	Due: 2025-03-20	Completed
Task	Edge-case prediction and testing	Eric Petersen	Completed
Task	End-to-end system testing	Wesley Eze	Completed
Epic	**Phase 3 Report**	Due: 2025-04-09	Completed
Task	Cover Letter	Wesley Eze	Completed
Task	Executive Summary	Eric Petersen	Completed
Task	Main Body	Jacob Damant	Completed
Task	Design Matrix	Gerlof Bakker	Completed
Task	Design Calculations	Minh Luu	Completed
Task	Phase 3 Peer Review	Connor Povoledo	Completed
Epic	**Design Conference**	Due: 2025-04-04	Completed
Task	Product Design Poster	Jacob Damant	Completed
Task	Presentation	Connor Povoledo	Completed

Signatures

These signatures along the bottom of this document signify that the group members of Fabri Sciences Inc. are committed to the completion of these tasks and to the agreed terms within this Charter.