The Third Hand - Remote Control Gantry System

Authors: Jack He, Leland Mersky, Zhenghao Jin
Affiliation: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*— In this paper, we propose a design for a teleoperated system intended for manipulating objects in a laboratory environment. This system offers the ability to manipulate objects remotely, without risk of injury for the user. This is achieved by a two-pronged approach: a novel force-sensing system to enable control of a gripper, and a computer vision-based system to enable high-precision control in position and attitude of the end-effector.**

***Index Terms*—Robotics, Force Sensing, Computer Vision, UI/UX**

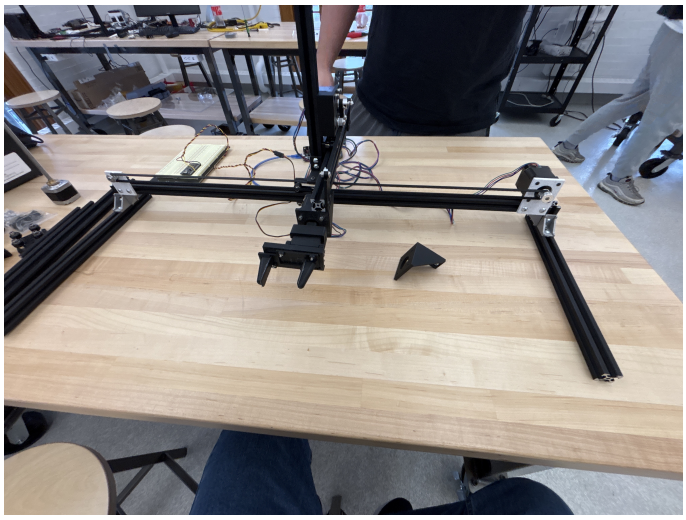

Figure 1: Final Robot System

1 INTRODUCTION

A chemistry lab is a dangerous environment to work in, and chemists take many precautions to avoid such dangers. Protective gear needs to be worn to protect human skin from burns and toxins. Proper ventilation needs to be installed so that no one accidentally exhales toxic fumes. What if we could eliminate the possibility of injury completely? That is the goal of our project. We aim to build a system that allows a chemist to perform tasks in the chemistry lab without needing to be in the lab.

Our teleoperated system involves a user holding and moving a remote spatially to control a robotic arm's position. The user squeezes the remote to open and close the gripper on the robotic arm. This allows the user to operate in a chemistry lab without the risk of actually needing to be in the room. With our setup, a chemist is be able to work at a lab bench while being on the other side of a protective glass panel. This system is important because it can reduce the risk of lab accidents that result in injury.

Robotics is already being used in lab environments for automation, like liquid handling and chemical synthesis. However, our system aims to be more general-purpose, such that it's applicable to many tasks instead of being optimized for one. It has to be able to do enough tasks; otherwise, there wouldn't be much use for it. Since we are making a general-purpose robotic arm, there can be many other use cases outside of the chemistry lab. Other labs and workshops could find this device useful. The goal of our system is to provide a teleoperated system to perform chemistry tasks suited for a human hand within a constrained space about the size of a desktop.

2 USE-CASE REQUIREMENTS

The robot needs to be proficient at performing tasks at the lab bench. These tasks consist of measuring and transferring liquids, mixing solutions, weighing chemicals, handling containers, etc. In general, the robot needs to be able to pick up objects, move them and in the specific case of liquid handling, be able to rotate the containers.

For the task of object handling, we specify that the robot shall be able to pick up and move a 300g payload. Due to the delicate nature of the use case, the robot must also be precise in its motion. On the other hand, speed is considered as well because we must make it easy for users to perform a task at or near the pace of a human operator. In order to maximize the usability of the system, we also set the use case requirement that the robot shall imitate the hand motion of the operator. See Design Requirements for how these requirements translate into quantitative metrics.

3 ARCHITECTURE AND/OR PRINCIPLE OF OPERATION

3.1 Motivation

In choosing the overall solution approach, we aimed to mimic the experience of physically picking up an object to the maximum practical extent. This means that instead of pushing buttons or using a joystick, the user moves their

hand as if they are manipulating the object themselves. Computer vision is used to capture the user's hand pose via an ARuCo tag on a handheld controller. The pose is then translated into commands for each motor on the robot.

3.2 Robotic Manipulation System

The robotic manipulation system consists of a gantry system and a gripper. The gantry system is responsible for moving the gripper across the workspace.

3.2.1 End-Effector

In the laboratory use case, objects such as tweezers, droppers, and microscope slides present challenges for the gripper. Fragile objects can break if there is too much force. Smooth objects can be dropped if there is not enough force. Depositing the correct amount of liquid from a dropper requires the force to be precisely controlled. The gripper is responsible for picking up an object, and must be able to do so while controlling the amount of force exerted with a high degree of precision.

The initial solution approach we came up with for the gripper consists of a DC motor connected to a torsional spring. As the gripper begins to close on an object and prior to making contact, the spring has no effect because there is no force in the system. After making contact with the object, initially there would be very little force. The torsional spring's purpose is to convert angular position into force: the further the DC motor turns after the gripper makes contact, the more the force exerted. This provides a relatively easy way to control both position and force with a single actuator.

In practice, sourcing the appropriate spring for the task proved to be difficult: torsional springs are not made in the range of force and size that our application demands. Therefore, we pivoted to using foam blocks attached to the gripper.

The principle of operation is similar to the spring approach, where force is modulated by the extent of compression. However, the mechanism is much simpler, with the added benefit of a larger contact patch on the object being gripped due to deformation of the foam conforming to the payload.

We also switched from using a DC motor to a servo. The initial mechanism required an ungeared motor to be able to wind up springs with enough rotations. The output shaft must then be attached to a gearbox to output high torque at relatively low speeds. This requirement for an ungeared motor led to the choice of a DC motor. After the pivot to foam pads for force control, this is no longer necessary. Servomotors come with built-in gear reduction, so the output torque is sufficient for our use case. Furthermore, positional feedback control is built into a servo motor. On the other hand, using a DC motor requires interfacing an encoder to detect position and writing a custom feedback controller.

3.2.2 Gantry System

The gantry system consists of linear slides in the XY plane, a lead screw to move the end effector along the Y-axis, and a direct-drive mechanism to rotate the gripper along the Y-axis. We have decided to not adopt a full 6DOF solution because we found that the added complexity does not pay off. The use case requirement of liquid pouring can be fully handled via the single rotation DOF.

All the DOFs in the gantry system are driven with stepper motors due to their high precision, low cost, and simple control setup.

3.3 Computer Vision Hand Tracking

In order to track the pose of the user's hand, we use OpenCV to detect ArUco tags on the user's handheld device.

The computer vision system consists of a stereo camera and a PC. The stereo camera is mounted above the user, who is sitting at a station with a table. Our software runs on the PC and reads the video stream from the stereo camera. It then extracts XYZ axis position as well as tilt from the tag using OpenCV. Finally, the pose information is serialized into Gcode commands and sent to the motor driver board that controls the gantry motors over serial communication.

3.4 Handheld Controller

The handheld controller is responsible for sensing the force input, as well as carrying the ArUco tag. The final iteration of the controller chassis consists of a 3D printed chassis with built-in flex to provide an elastic feel when the user grips it. Force sensors are embedded on the surface of the controller, where the user's fingers rest. When the user wishes to grip an object, they can squeeze the controller. The force readings are then sent to the gripper, which controls

3.5 Communication Between Subsystems

The interaction between subsystems is illustrated in figure 2

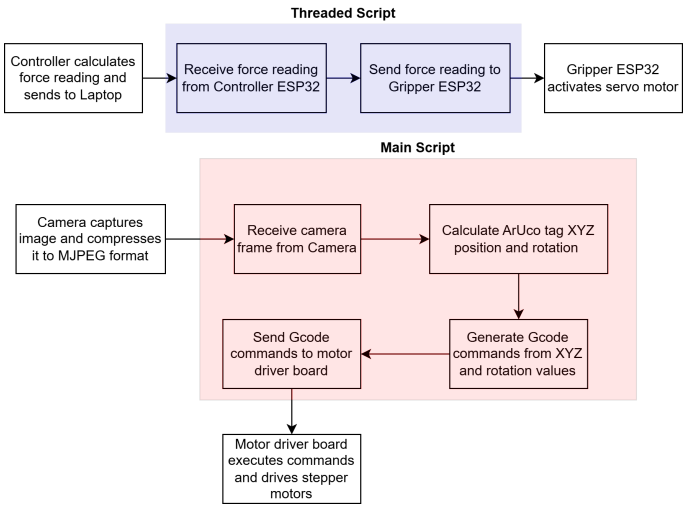

Figure 2: Flow of Information

The communication between the PC, the gantry, as well as the handheld controller all happen over USB. We used the pyserial library on the PC to read and write all the packets.

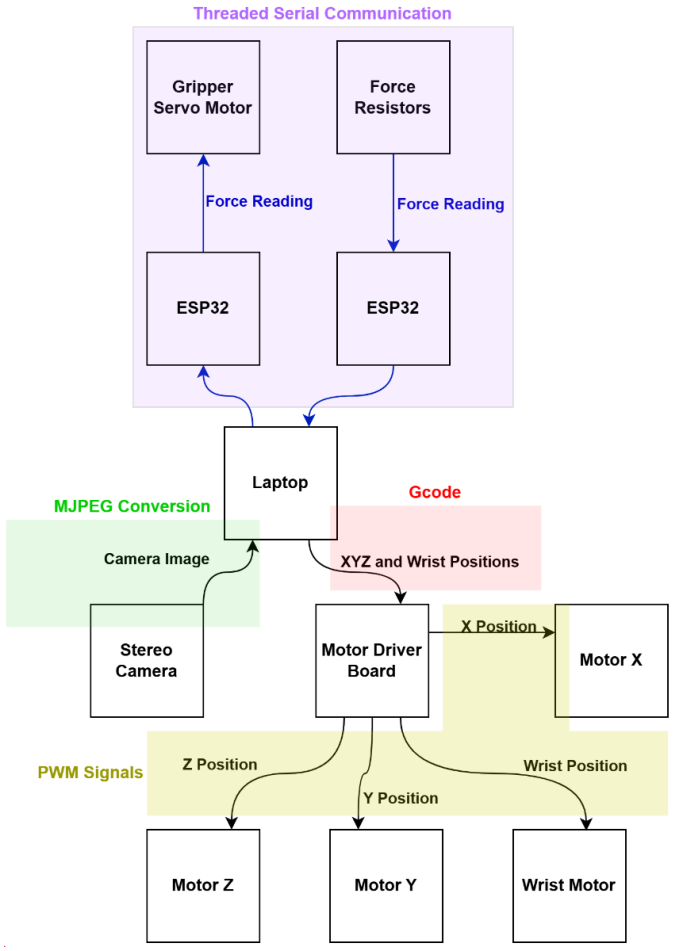

Figure 3: System Architecture

4 DESIGN REQUIREMENTS

In order to achieve the use case requirements, we have developed a series of quantitative engineering requirements. We arrived at these requirements from three sources: back-of-envelope calculations on how fast/accurate humans can perform manipulation tasks, existing research on the effect of latency in teleoperation, and typical payload weights in a chemistry lab setting. These requirements are summarized in table 1

In a study on the impact of latency on robotic surgery [2], it was found that latencies below 150ms do not have a significant impact on the Procedural Impact Score, a rating given by operators. Considering the similarity between our use case and this study, we concluded that a design requirement of 100ms latency should be acceptable.

Precision for an axis of movement is defined as the range of differences between a commanded movement and the actual movement, over a sample size of 10 moves.

5 DESIGN TRADE STUDIES

5.1 Robotic Manipulator

5.1.1 Selecting Geometry

We considered both a gantry-style geometry and a traditional robotic arm based on rotational joints. We ended up going with a gantry because it is difficult to attain good precision for the end effector in a traditional robot arm. This is because angular errors are amplified over a long lever arm (the lever arm has to be as long as 400mm in our use case). Furthermore, slack inside each rotational joint builds up on top of each other, resulting in even more amplified backlash. In addition to precision issues, a traditional robot arm also requires a high-torque actuator. This necessitates the use of gearboxes and high-current DC motors, which add cost and complexity. While a traditional robotic arm can be beneficial by more closely mimicing human anatomy, we found the engineering challenges involved tipped the balance toward a gantry.

5.1.2 Selecting Motor Type

For the gripper, we switched from using a DC motor to a servo. The initial mechanism required an ungeared motor to be able to wind up springs with enough rotations. The output shaft must then be attached to a gearbox to output high torque at relatively low speeds. This requirement for an ungeared motor led to the choice of a DC motor. After the pivot to foam pads for force control, this is no longer necessary. Servomotors come with built-in gear reduction, so the output torque is sufficient for our use case. Furthermore, positional feedback control is built into a servo motor. On the other hand, using a DC motor requires interfacing an encoder to detect position and writing a custom feedback controller.

Table 1: Design Requirements

Description	
speed (X/Y plane)	100mm/s
speed (Z-axis)	10mm/s
precision (XYZ)	2mm
grip force	9N
latency	100ms
workspace volume ($L \times W \times H$)	0.75m x 0.5m x 0.25m

For the gantry, we decided to use stepper motors to power the robot arm for the following reasons: high precision, open-loop controls and low cost. Alternatives we evaluated include hobby servo motors, DC motors, and stepper motors. The tradeoffs are summarized in Table 2.

Table 2: Motor Type Selection

	Stepper Motor	DC Motor	Servo
Gearbox Required	No	Yes	No
Encoders Required	No	Yes	No
Precision	High	High	Low

As shown in the table, stepper motor has high precision, and does not require gearing or encoders. The lack of a gearbox is a very important advantage: the linear degrees of freedom are actuated with a belt mechanism, which is zero-backlash provided the belt is tensioned. However, backlash is introduced with any gear mechanism, since gears cannot be made to mesh without any slack. Typical backlash in a DC motor gearbox ranges between 1-3 degrees.

In order to select the proper stepper motor, we started from the quantitative design requirements for each axis.

5.1.3 Z-axis Mechanism

For the Z-axis, the most important requirement is payload weight. In order to support a 300g payload, we estimated that the Z-axis motor must be able to lift 750g, accounting for the weight of the gripper mechanism. We chose a lead screw mechanism due to its large mechanical advantage. The threads are of the T8 type. This means the Z-platform travels 8mm for one rotation of the screw. The formula for torque required to move a load on a lead screw is given in [4]:

$$T = \frac{FL}{2ME} \tag{1}$$

Where T is torque(Nm), F is force(N), L is lead (m), the distance traveled per rotation, M is the coefficient of friction and E is efficiency. For coefficient of friction and efficiency, we selected typical values according to [9]. The values used are shown in Table 3.

Table 3: Lead Screw Equation Parameters

F(N)	L(m)	M	E
7.5	$8e^{-3}$	0.35	0.25

From (1), the required torque is 0.034Nm, or 34Ncm. Based on the required torque, we selected StepperOnline 17HE24-2104S. The torque curve is given in 4. The 17HE24-2104S has more than 60Ncm of torque at 0RPM, which is more than enough.

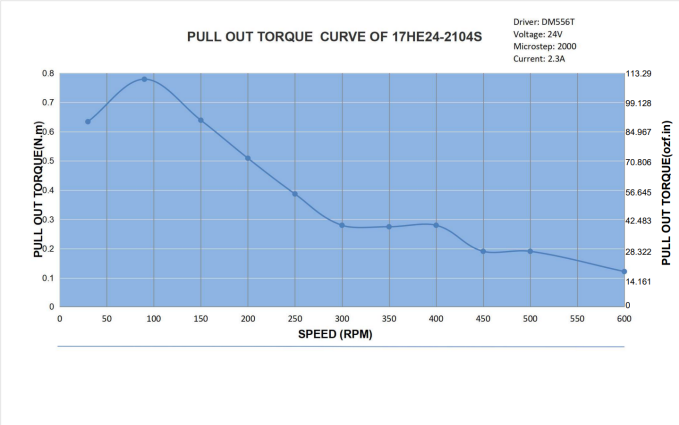

Figure 4: 17HE24-2104S Torque Curve

In addition to payload, the stepper motor also needs to meet speed and precision requirements. The 17HE24-2104S has a step angle of 1.8 degrees. This is 1/200th of a rotation. Therefore, one step of the motor translates to $\frac{8mm}{200}$, or $40\mu m$. This exceeds the requirement of 2mm precision.

In order to achieve the required 10mm/s speed, the motor must turn faster than $1.25RPS$, or $75RPM$ and be able to lift the payload at that speed. Referencing the torque curve of the 17HE24-2104S, 75 RPM is near the torque peak at about 80 Ncm. Therefore, the motor can lift vertical load at the required speed.

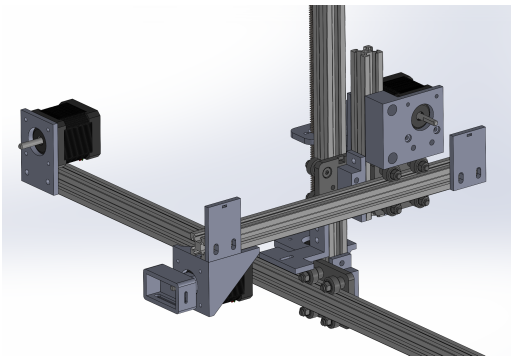

Figure 5: Gantry Assembly CAD design

5.1.4 XY-axis Mechanism

Unlike the Z-axis, the main design constraint is acceleration. If the acceleration is too slow, the robot would not be able to start from standstill and reach the maximum speed in time to satisfy the speed test. In order satisfy speed requirements, the tradeoff between torque and speed must be taken into consideration when choosing the drive mechanism and the motor.

We have chosen to use GT2 timing belts with 20 tooth pulleys to transmit linear motion to the robot arm assembly. Using timing belts provides the crucial advantage of zero-backlash: adjustable tensioners make sure that the belt is always tight. On the other hand, any kind of geared mechanism involves backlash that compromises precision.

To calculate the amount of torque required, we need to know the radius of the pulley. The 20 tooth GT2 pulleys that we selected have a pitch diameter of 12.73mm [6]. This means when the belt is wrapped around the pulley, measuring from the middle of the belt's thickness, the diameter is 12.73mm. The axis with the most moving mass is the x-axis, since the Y/Z mechanisms are all installed on top of the x-platform in our design. A back-of envelope combined weight was set at 1.5kg. Since the mechanism slides on ball bearings, it should be sufficient to ignore friction. This puts the required force at $g \cdot 1.5kg = 15N$. At a radius of $\frac{12.73mm}{2} = 6.4mm$, the torque required at the motor shaft is $15N \cdot 0.64cm = 9.6Ncm$.

Typical stepper motors rotate 1.8 degrees, or 1/200th of a rotation per step. With a 12.73mm diameter pulley, one rotation translates to approximately 40mm of translation motion. Therefore, one step is 40mm/200 = 0.2mm of movement. Therefore, this configuration satisfies our precision requirements. To achieve a speed of 100mm/s, the motor needs to turn (100mm/40mm) = 2.5 times per second. This is equivalent to 150 rpm.

Based on these requirements, we chose the Stepper-Online 17HE15-1504S motor. At the required speed of 150rpm, it can output 32Ncm of torque, which translates to apprximately 18N of linear force, more than enough to overcome friction.

5.2 Camera Selection

We chose a stereo camera, so that we could accurately measure depth. The exact model is the eYs3D Stereo Camera - EX8036. The other alternative considered was a monocular USB webcam, the Logitech StreamCam. The following subsections provide the logical reasoning that led to the stereo camera being chosen.

5.2.1 Resolution

The resolution of the camera determines how small of a change we can detect between pixels. Our stereo camera has a resolution of 1280 x 720 pixels. The USB webcam that was considered has a resolution of 1920 x 1080 pixels. The pixel size is calculated by

$$Size = \frac{Distance}{Pixels} \tag{2}$$

Size is the length and width of the pixel. Distance is the length along the axis. 0.75 m along the x-axis and 0.25 m for the y-axis. Pixels is the number of pixels along the specified axis. 1920 pixels for the x-axis, and 1080 pixels for the y-axis. Precision is measured in the worst case such that the ArUco tag is as far from the camera as possible leading to a larger error in positional calculations. The USB webcam would have a precision of approximately 0.391 mm/pixel and 0.231 mm/pixel for the x-axis and y-axis, respectively. In comparison, the stereo cam has a precision of about 0.586 mm/pixel and 0.347 mm/pixel. Both of these cameras satisfy our constraint of precision within 2 mm for the x and y axes.

The best way to measure depth resolution is to set up the camera up and measure depth error directly. Since that's not ideal, I researched and found that monocular depth estimation can often report errors ranging from 10% to 30% of the estimated depth according to [5]. The depth error could be $0.10 \cdot 50mm = 5mm$, assuming the camera is set at a height of about $50mm$. This does not meet our goal of 2mm.

The eYs3D Stereo Camera can output normal frames and a depth map simultaneously at 1280x720p at 60 fps. It has two lenses such that it can calculate depth more precisely within our $2mm$ range. This saves us time from having to do the depth calculations which decreases latency. Lastly, it was available in the 18-500 inventory, which saves us room in the budget.

Figure 6: An Illustration of the eYs3D camera

5.3 Force Sensor Selection

The force sensors are used to measure the applied force for gripping objects. The criteria for selection include accuracy, response time, load capacity, and cost. The following table provides a comparison of potential force sensor options.

Table 4: Force Sensor Trade Study

Sensor Model	Cost (per unit)	Load Capacity (kg)	Accuracy (N)
FSR-402	$6.00	10	0.5
Honeywell FSG	$45.00	50	0.1
Tekscan FlexiForce A201	$25.00	25	0.2
DF9-16	$4	20	0.2

- **FSR-402**: A cost-effective and readily available option, ideal for low-load applications. However, its accuracy is relatively low, making it suitable only for rough force measurements. - **Honeywell FSG**: A high-performance force sensor with excellent accuracy and response time. It is well-suited for precise control but comes at a significantly higher cost. - **Tekscan FlexiForce A201**: Offers a balance between accuracy and load capacity at a moderate price. It's a good option for more precise measurements without the high cost of Honeywell sensors. - **DF9-16**: A mid-range sensor with good accuracy and a moderate load capacity. It offers a balanced solution between performance and cost, making it suitable for applications requiring both precision and cost efficiency.

For applications requiring high accuracy and speed, the **DF9-16** is the best choice. It provides a good range of force input, and its price allows some flexibility for our design.

6 SYSTEM IMPLEMENTATION

6.1 Robot Manipulator Hardware

The robotic manipulator is controlled by an ESP32 microcontroller. The ESP32 receives force and position commands over serial from the PC. After receiving the command, it translates them into Gcode, a language used to specify machine movement commonly used in 3D printers and CNC machines. The translated Gcode is sent over serial to a SKR Mini E3 3D printer controller board shown in Figure 7. The printer control board then converts the Gcode into PWM signals and drives the stepper motors.

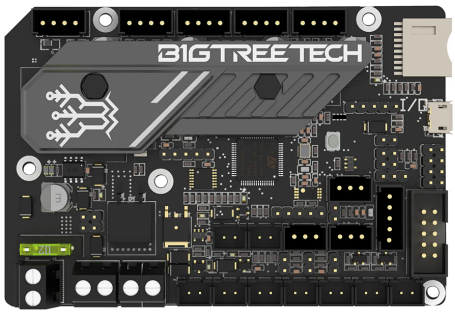

Figure 7: SKR Mini E3 3D Printer Control Board

While gantry motors are driven with a 3D printer control board, the gripper servo motor is controlled by the gantry-side ESP32 directly via PWM. Power is supplied to the system using a 12VDC, 15A module that plugs into a regular wall outlet. All gantry motors are supplied off this rail, while the ESP32 is powered over a USB connection to the PC, doing double duty as the serial communication link.

Force sensing resistors on the gripper are wired in a voltage divider setup, and we read the voltage off the ESP32's GPIO pin. See Figure 8 for details on the circuit diagram.

6.2 Robot Manipulator Software

Software for the robotic manipulator system is written in **C++**. The software consists of 4 components: force control, position control, force sensing, and mode control.

6.2.1 Position Control

The position control code generates Gcode commands from pose. More importantly, it must perform zeroing. A stepper motor's position is controlled by sending PWM signals to energize different coil windings alternatively. However, there is no way to know what position a stepper motor starts out at when the user turns on the system. With the 3D printer driver board, we use the built-in current sensing capabilities to create a software routine to drive all stepper motors until a mechanical endstop is reached. When the endstop is reached, our software sees a spike in current readings and stop the motors. The software is then able to keep track of all movement with the endstop position as the zero value, and translate the position commands accordingly.

6.2.2 Latency Reduction

In the course of implementation, we discovered a major issue with the SKR Mini E3 board. Due to its original use case of 3D printing, it is designed to prioritize the completion of an existing command before it executes a new one. When the user moves their hand quickly, the motors are not able to keep up. This is not a problem in and of itself. However, when the user wishes to stop or change direction after a quick movement, this second command is delayed, as the robot tries to finish the last movement.

The solution to this issue was to chop up motion commands going into the SKR board. Large movement commands are broken down on the software side into many small movements. As a result, it never takes too long for the robot to finish a move before it listens for the next. The slicing of movements is based on a maximum permitted move length, which is in turn calculated from speed of the motors. Using this approach, software-motor latency is kept under 20ms.

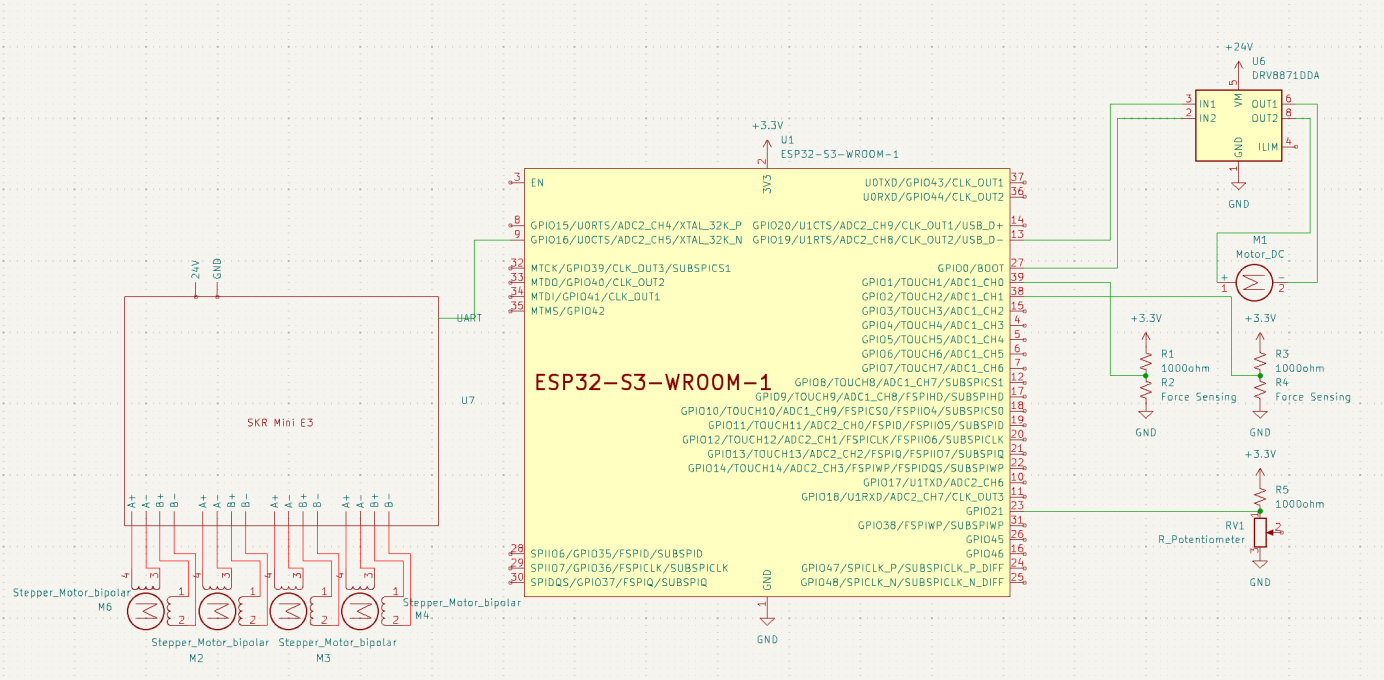

Figure 8: Robotic Manipulator Circuit Diagram

6.3 Remote Controller System

6.3.1 Remote Controller Hardware

The original plan was to use a section of PVC pipe as the chassis for the remote controller. Outside of the PVC pipe, there would be a layer of 3D-printed TPU soft material. Between the PVC pipe and the outer soft layer, we would embed force-sensing resistors. The ESP32 microcontroller, as well as the haptic motor, would be secured inside the PVC pipe. On the outside of the remote controller, there would be an ArUco tag, placed such that it is unobstructed by the user's hand.

During the implementation phase, we pivoted to a 3D printed design due to the ability to customize the shape of the controller. The design consists of a cylindrical handle with a long opening, such that the whole handle flexes when squeezed. We paste five force sensors on the controller based on human hand configuration, which will enable us to collect force data on all five of the fingers. The sensors are connected to an ESP32 chip for data processing.

6.3.2 Remote Controller Software

The software for the Controller is mostly on the Arduino. Instead of activating all 5 sensors at the same time, we decide to activate only one sensor at a time to avoid cross-interfering of sensor readings. We turn the five sensors on and off in an order very rapidly such that users can hardly feel the difference from always activating all the sensors. The P-MOSFET in Figure 10 serve as switches to turn the power supply for force sensors on and off.

The ESP32 was then connected to the controller PC through UART connection and data was read by a python program that matches the serial rate of the board.

6.3.3 Communication Module

The processed control signals are transmitted via serial to a PC, which relays the commands to the robotic hand. The communication link is designed to ensure low-latency transmission, providing real-time feedback to the operator.

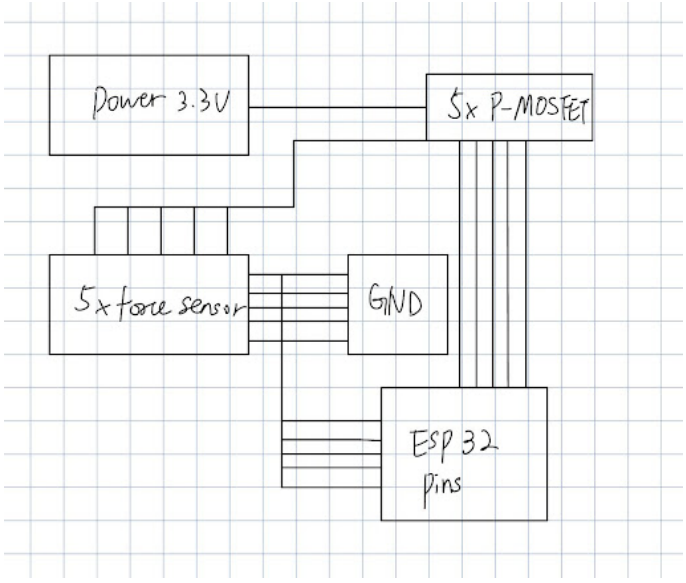

Figure 10: pull-up resistors built into the ESP32 GPIO pins complete the voltage divider setup

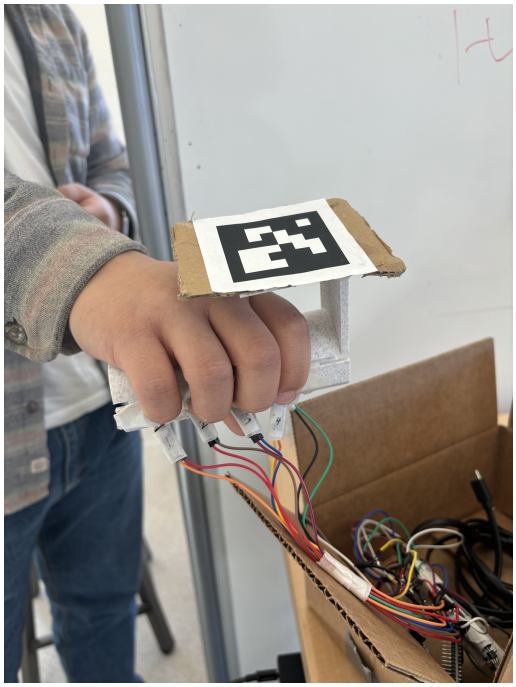

Figure 11: Handheld Controller

6.4 Computer Vision System

The camera being used is the eYs3D stereo camera, which was introduced in Section 5: Design Trade Studies. This setup includes the eYs3D camera, a camera stand, and an ArUco tag, providing all the necessary hardware. For computer vision software, we use OpenCV, which our team has successfully utilized before, making it ideal for real-time vision processing.

Due to the camera stand package being lost, we built a custom camera stand using leftover material from the gantry. The ArUco tag is attached to the controller such that the tag is above the user's hand. This arrangement allows the user to grip the controller while allowing the camera to see the ArUco tag overhead. We use this ArUco tag to approximate the position and pose of the user's hand.

6.4.1 OpenCV

The stereo camera can supply our laptop with 60 frames of normal image data and 60 frames of depth map data every second. For each pair of frames, we use OpenCV's ArUco library to detect and identify the ArUco tag. Once we locate and identify the tag, we need to convert the XYZ and tilt data into commands for the gantry system, which we send serially.

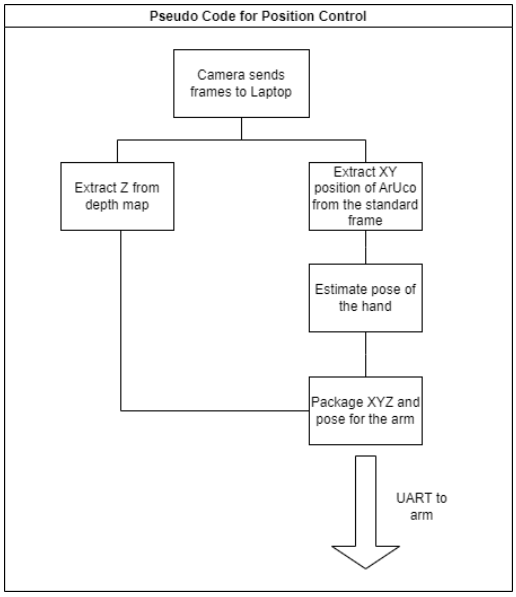

Figure 12: computer vision pipeline

6.4.2 Calibration

In most cameras, the output image experiences distortion near borders and corners. As a result, the same movement in different physical locations do not translate to equal distances on the image plane. This creates an issue for the user experience, where the control feels slightly different near the edge of the workspace. Furthermore, moves in a straight line translates to a curve due to distortion. In order to account for this effect, we implemented a calibration routine that identifies corners on a checkerboard pattern. The corner locations are then used to estimate the parameters of the distortion effect. The end result is a rectified image with distortion removed.

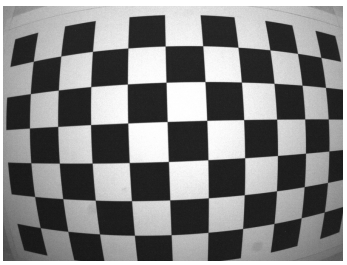

Figure 13: Checkerboard Pattern Used in Camera Calibration

6.4.3 Camera Latency Reduction

An unexpected source of latency is the camera. Due to bandwidth limitations of the USB interface, most cameras compress the frames before sending them. The camera uses H.264 codec to do so. This is a stateful codec that compresses information by only sending the difference between the current frame and previous frames. This approach maximizes throughput. However, latency suffers because a few frames have to be buffered before one can

be sent out. When we measured the latency of the system using video and a timer, we found that almost all of the measurable latency of 400ms came from the camera.

In order to mitigate this issue, we edited the camera configuration to compress using MJPEG. This approach compresses individual frames as JPEG images, and therefore does not require buffering. The tradeoff is that the frame rate is decreased from 60fps to 30fps due to decreased compression ratio. However, in context of the 400ms latency in H.264, the increase in interframe latency from $\frac{1}{60} = 17ms$ to $\frac{1}{30} = 33ms$ is an acceptable tradeoff.

6.4.4 Data Control

The laptop acts as the communicator between the remote and the robotic arm. To facilitate this communication, we can use the pySerial library to send and receive data from both the arm and the remote through serial. Each communication link uses an individual thread to eliminate IO waiting-related latency.

7 TEST & VALIDATION

7.1 Design Requirement Verification

7.1.1 Precision Tests

In order to test the precision of the system, we attached a pencil to the gripper and drew lines on a piece of paper by sending the gantry a position command from the PC. The distances drawn was then measured using a caliper.

Table 5: Precision Test

Axis	Precision (mm)
X	0.7
Y	0.6

7.1.2 Movement Tests

There are tests for movement in each direction. For the purpose of measuring speed, we send commands to the robot to move it as fast as possible. These are internal commands without the user in the loop, so as to eliminate the confounding variable of user skill. We do this separately for each axis of movement. Times for these tests are measured using a stopwatch. The time to pass figures below for the X and Y axes are based on moving at the required speed over the length of the axis, and adding 0.25s to account for accelerating from standstill.

Table 6: Movement Speed Tests

Axis	Distance (mm)	Time Required (s)	Time Measured (s)
X	100	1.25	0.98
Y	100	1.25	0.97

It should be noted that while the design requirements are achieved, the speed proved to make it difficult for users to control the robot precisely. Therefore, the actual speed of the gantry system was turned down to approximately 50mm/s.

7.1.3 Latency Testing

For latency measurement, what we originally wanted to do is to estimated using a round-trip time (RTT) system. A significant challenge in measuring latency stems from the lack of a common clock source between different subsystems. Therefore, it is difficult to measure one-way latency. We opt instead for the RTT method, where $T_{oneway} = \frac{RTT}{2}$. We are targeting a 100ms one way latency from the user to the robotic manipulator. In order to do so, we were going to embed timestamps and sequence numbers in command packets sent from the remote controller to the robot. The robot would have an echo mode that sends the packets back. For a given packet sequence number, we would check the the time it takes for the echo to arrive to calculate RTT.

However, we found that for some part of the system, we can not implement a time stamp into it. In addition, measuring latency only based on the software may overlook some of the hardware and mechanical latency. As a result, we decide to use active mode in smartphone camera to record a video with high FPS, and count the steps spend between one command from the controller and the system starts to move. This way, we include all the latency from controller to PC, camera to PC, and PC to gantry system.

As discussed in System Implementation, the main source of latency is the camera. After switching to MJPEG, the system latency decreased to 200ms. While this still falls short of the 100ms design requirement, the qualitative experience is that the latency is perceptible, but controlling the robot is still very feasible. Research into additional ways to decrease latency led to using industrial-grade cameras that are outside of our budget.

7.1.4 Range of Motion

Range of motion specified in the design requirements has been achieved for all but the Y-axis. The Y-axis is a cantilever mechanism and we found that extending it to the specified 50cm range caused a lot of mechanical strain to the extent of visible sagging. Therefore, the Y-axis sliding assembly was shortened to 25cm.

7.2 Use Case Validation

In order to test the use case requirement of 300g payload pick-up and place, we used a water bottled filled with 300mL of water. The user picks up the bottle and exercises the full range of motion of each movement axis. This is achieved for all movement axes. However, we decided to disable the Z-axis in the final demo due to the slower Z-axis motor causing latency in the system. Instead, pickup and place can be achieved by rotating the wrist, as opposed to lifting and dropping the gripper.

7.2.1 Transferring Liquids

The user transfers 100 mL of water from one container to another by gripping the container with the robot and rotating it.

In our testing, we use a water bottle with water to simulate the container that originally holding the liquid we intend to transfer. A plastic box is used as the container we transfer the water to. A successful approach should show that the water was clearly poured from the water bottle to the plastic box, with no water spilled on the table.

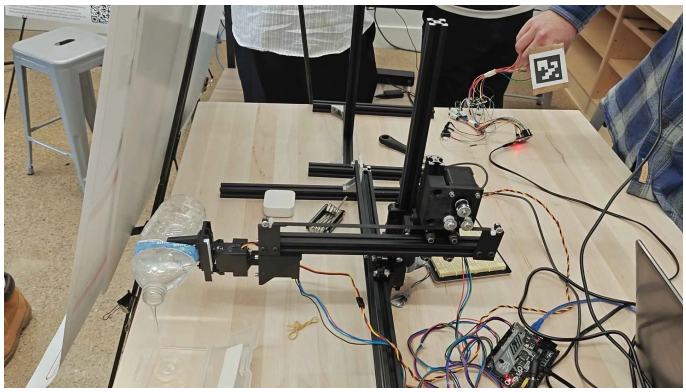

Figure 14: Use Case Testing: Liquid Transfer

8 PROJECT MANAGEMENT

8.1 Team Member Responsibilities

- Jack: Designing and building the robot manipulator system. This includes mechanical, electronics and software.
- Leland: Designing and implementing computer vision and PC-side software.
- Zhenghao: Designing and building the handheld remote controller. This includes mechanical, electronics and software.
- Shared: Designing shared components such as the communication packet protocol.

8.2 Risk Management

Effective risk management played a key role in the successful completion of our project. The team followed a flexible and adaptive approach, addressing challenges as they arose rather than relying on a rigid risk mitigation plan. While we initially developed a solution plan to guide our work, much of the risk management evolved organically as the project progressed.

At the start, we discussed potential risks such as the hardware designs being too complex to implement, software integration issues, and scheduling conflicts. However, many challenges were unpredictable, as we were learning about the components and their limitations through implementation. One significant challenge involved our motor driver board, which was designed for 3D printers and executed commands sequentially without interruption. This behavior conflicted with our need for rapid responsiveness to changing inputs. We overcame this by breaking down commands into smaller segments, storing them in a queue for execution, and refreshing the queue with new commands when needed. This solution allowed our system to adapt quickly in real time.

To address challenges throughout the semester, we held regular meetings to share updates, discuss problems, and plan next steps. These meetings helped us identify bottlenecks, reprioritize tasks, and create plans to meet deadlines. A notable turning point occurred after fall break, when the team returned with little progress made. Recognizing the urgency, we focused up and dedicated weeks of effort to ensure a successful interim demo, which became the highlight of the semester. This milestone reflected the team's ability to adapt, coordinate, and deliver under pressure.

Despite our successes, the final demo presented challenges that we were unable to fully resolve in time. While disappointing, the effort put into the project demonstrated the team's determination and willingness to tackle complex problems.

Though informal, our approach to risk management allowed us to address issues effectively and learn from challenges. Moving forward, adopting a more structured risk identification process early in a project could enhance our preparedness and efficiency.

9 RELATED WORK

In our brainstorming process, we consulted [1], a haptic feedback remote control system for rovers. We took inspiration from their approach, where the remote controller mimics the rover's movement. We also took inspiration from [8], a robotic surgery system for inserting catheters. In [8], the remote controller mimics the mechanical structure of the robotic end effector, and provides force feedback to the user via rotating parts of the remote control that correspond to actuators on the end effector. During our search for a force feedback mechanism, we considered a microfluidics approach similar to [7]. However, after examining their approach, we realized we did not have the capabilities to do something similar. Instead, we chose an approach similar to [3], a system that utilizes haptic feedback for rehab training using video games.

10 ETHICS AND COMMUNITY IMPACT

A significant ethical concern in our project was the possibility that user error causes a spill. The engineering solution we devised in response is to "freeze" the robot if no tags are detected by the computer vision system. This way,

Table 7: Bill of Material & Cost

Description	Count	Unit Cost ($)	Total Cost ($)
ESP-WROOM-32 pack of 3	1	15.99	15.99
2020v extrusion, 1500mm	1	50.02	50.02
linear slides pack of 4	1	26.97	26.97
lead screw kit	1	12.99	12.99
2020 extrusion corner bracket	1	13.99	13.99
2020 extrusion t slot screw/nuts	1	12.99	12.99
GT2 timing belt kit	1	9.99	9.99
3D printer driver	1	43.99	43.99
NEMA 17 motor x3	1	32.99	32.99
24V 15A power supply	1	20.98	20.98
Z-axis motor	1	8.97	8.97
Force sensing resistor pack of 2	5	8.95	44.75
Stereo Camera	1	0	0
Camera Stand	1	35.98	35.98
Vibration Motor pack of 5	1	6.99	6.99
Servo	1	9.99	9.99
10K potentiometer	1	0	0
perf board	2	0	0
pack of jumper cables	1	0	0
		Total:	311.60

if the user accidentally drops the controller or blocks the camera, the robot will hold on to the object and not move until the tag is reacquired.

Another important ethical issue concerns the possibility that decision makers such as lab PIs and university department heads will be hesitant to purchase the robot because they are less impacted by lab accidents than the front line researchers. We hope that the low cost of the system at approximately $400 (BOM cost, accounting for free items we obtained from ECE inventory), makes it accessible to all researchers. The low costs should hopefully mean that labs and researchers in developing nations, without access to ample funding, can also access our system.

11 SUMMARY

The "Third Hand" project presents a system designed for manipulating objects in hazardous laboratory environments. Our system combines force-sensing and computer vision for precise control, allowing users to operate a robotic arm from behind a fume hood, reducing the risk of injury. The robot utilizes a gantry system and gripper for handling delicate tasks such as transferring liquids and gripping fragile objects. The computer Tests confirm the system's ability to perform tasks with human-like accuracy and reliability, making it a safer alternative for lab operations. The expectation for the project is that users can utilize our system smoothly after very few times of training.

11.0.1 Future Work

Future areas of work we identified include stronger and more rigid mechanical structure, wireless communication between the handheld controller and the robot, implementing force feedback, and switching to a low-latency machine vision camera.

11.0.2 Lessons Learned

Lessons we learned include the importance of time management. There was a lot of work that could have been done better had we started early. We also suffered from making technical decisions early in the project, without fully realizing the implications. For example, a lot of work was put into fixing the latency issue on the 3D printer driver board. If we had understood how Gcode processing queues on these boards worked, we would not have gone with that solution approach.

Glossary of Acronyms

Include an alphabetized list of acronyms if you have lots of these included in your document. Otherwise define the acronyms inline.

- RTT: Round Trip Time
- NEMA: National Electrical Manufacturers Association
- ESP-IDF: Espressif IoT Development Framework
- TPU: Thermoplastic polyurethane

- ArUco: Augmented Reality University of Cordoba. A type of tags for computer vision based pose detection

References

[1] Kurihara et al. "Development of Haptic Feedback Control Stick for Remote Control between Different Structures". In: *2021 IEEE International Conference on Mechatronics (ICM)*. 2021, pp. 1–6. DOI: 10.1109/ICM46511.2021.9385652.

[2] Legeza P. et al. "Impact of network performance on remote robotic-assisted endovascular interventions in porcine model". In: *Journal of Robotic Surgery* 16.1 (2021), pp. 29–35.

[3] Guillermo Asín-Prieto et al. "Haptic Adaptive Feedback to Promote Motor Learning With a Robotic Ankle Exoskeleton Integrated With a Video Game". In: *Frontiers in Bioengineering and Biotechnology* 8 (2020), p. 113. DOI: 10.3389/fbioe.2020.00113. URL: https://pubmed.ncbi.nlm.nih.gov/32154239/ (visited on 01/06/2022).

[4] WM Berg Inc. *Calculating Lead Screw Maximum Load Lifting Torque*. URL: https://www.wmberg.com/resources/tech-and-application-support/lead-screw-load.

[5] Alexander Kern and Christine Willmann. *A New Perspective on Depth Estimation in Monocular Cameras*.

[6] LLC Pfeiffer Industries. *2mm PowerGrip GT Timing Belt Pulley PD & OD*. URL: https://www.pfeiferindustries.com/documents/(2mm%20PowerGrip%20GT)%20Timing%20Belt%20Pulley%20PD%20and%20OD.pdf.

[7] Ruben D. Ponce Wong, Jonathan D. Posner, and Veronica J. Santos. "Flexible microfluidic normal force sensor skin for tactile feedback". In: *Sensors and Actuators A: Physical* 179 (2012), pp. 62–69. ISSN: 0924-4247. DOI: https://doi.org/10.1016/j.sna.2012.03.023. URL: https://www.sciencedirect.com/science/article/pii/S0924424712001872.

[8] Cheng Yang, Shuxiang Guo, and Yangming Guo. "Development of a Novel Remote Controller for Interventional Surgical Robots". In: *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*. 2019, pp. 1964–1968. DOI: 10.1109/ICMA.2019.8816523.

[9] Daniel Zhang. *What's the difference between ball screw and lead screw?* URL: https://www.omc-stepperonline.com/support/whats-the-difference-between-ball-screw-and-lead-screw?srsltid=AfmBOortDUlMY4AzPD874_GudgJ-Xc5nmlqprsZERVHNuASs9TkAMvNX.

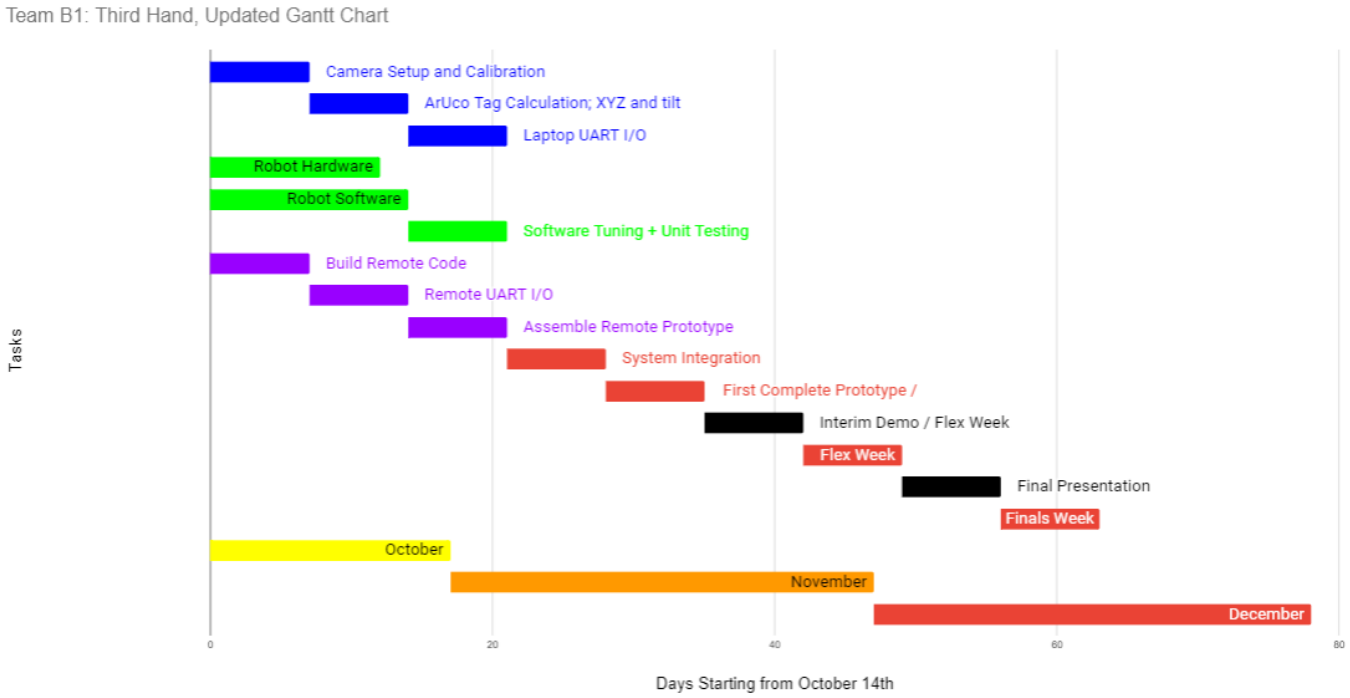

Figure 15: Gantt Chart