EE175AB Final Report

Robotics Arm Manipulator for Fruit Harvesting

EE 175AB Final Report

Department of Electrical Engineering, UC Riverside

Project Team Member(s)	Pamodya Peiris, Karen Gonzalez, Michael O'Dea, Priscila Huante Mendez
Date Submitted	3/14/2022
Section Professor	Merrick Campbell (TA), Dr. Roman Chomko (Professor)
Revision	1.5
URL of Project YouTube Videos, Wiki/Webpage	https://github.com/Pamodya98/Automatic-Fruit-Picking-Manipulator https://www.youtube.com/playlist?list=PLmYmW_H2cumjKVaWjVCRBdQTAdBVdFfo8
Permanent Emails of all team members	pamodya.peiris98@gmail.com, 1michaelodea@gmail.com, karengo1489@gmail.com, priscilahuante123@gmail.com

Summary

Our Senior Design Project is an Automatic Fruit Picking Manipulator that utilizes Computer Vision to identify the fruit and a Robotic Arm to pick it.

Revisions

Version	Description of Version	Author(s)	Date Completed	Approval
Version Number	Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being upgraded.	Full Name	00/00/00	
0.1	Report Template	Michael O'Dea	01/03/2022	Pamodya Peiris
0.2	Test Plan	Michael O'Dea	03/01/2022	Pamodya Peiris
0.3	Added information for Section 10	Michael O'Dea	03/01/2022	Pamodya Peiris
0.4	Added information for Section 2 and Section 3	Michael O'Dea	03/08/2022	Pamodya Peiris
0.5	Added information for Section 12 and Section 13	Michael O'Dea	03/11/2022	Pamodya Peiris
0.6	Added information for Section 8 and Section 5	Michael O'Dea	03/11/2022	Pamodya Peiris
1.1	Wrote and Finalized Section 1 and Section 14	Michael O'Dea	03/12/2022	Pamodya Peiris
1.2	Completed section 5 and 6	All	03/13/2022	
1.3	Completed Section 3-4	All	03/13/2022	
1.4	Completed Section 7-8	All	03/13/2022	
1.5	Completed Section 9	All	03/13/2022	
1.6	Completed Section 1 and 10	All	03/14/2022	
1.7	Completed Section 2, 11, and 12, 13	All	03/14/2022	
2.0	Final Report	All	03/13/2022	All

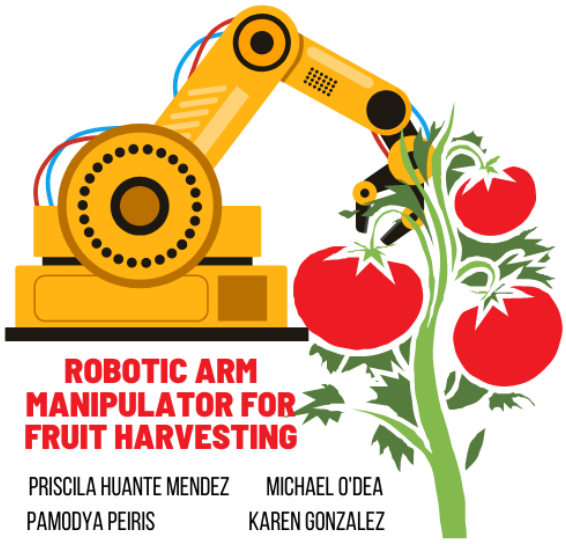


Figure 0.1 Logo

Figure 0.2 Robot Interface

Table of Contents

1 * Executive Summary

With the ever-increasing advancements in technology and its increasing accessibility around the world, this has led to the rise of technological innovations that aim to improve people's lives for low cost. These innovations also aim to allow people to be able to have more free time in their lives for personal or professional use. Our design project aims to accomplish both having our product be easily available for anyone in the world as well as also having a meaningful contribution for their lives. We propose a robotic arm manipulator fruit harvester with soft actuators for gripping fruit.

The goal of this project was for a robotic arm, using computer vision, be able to pick up red colored objects amongst a group of green colored objects. This required for the arm itself to be able to move within its configuration space, to the required object, lift it up, and place it down at a set of specific coordinates. In this design, by using two microcontrollers as well as an Intel RealSense Camera, the robotic arm is able to pick up the red colored object, which is a stimulated tomato, and then place it at a set of coordinates. When the main program is first initialized, whether from using a Raspberry Pi interface or using a desktop computer, there is a computer vision program which is started up and essentially runs until it finds the coordinates of a red object. Afterwards, the computer vision program sends that data to the inverse kinematics program, and then sends the data of what the angles of the servo motors should move to to the second microcontroller, the Arduino Mega. The Arduino Mega then uses the inputted numbers to move the servo motors accordingly as what was requested for the red object to be lifted and then dropped at a set place.

In this design project, the goal was for it to be useful while also being accessible and affordable for almost anyone and to also allow for easy integration into greenhouses and farms for tomato picking and harvesting. The design project also couldn’t be too complicated or overwhelming for potential customers in case they might be interested in purchasing the product.

The beginning of this project involved a substantial amount of research into the many different hardware and software components that could be implemented. Each device would have its pros and cons, however the ones that were chosen for this project were determined to be the best at an affordable price. The next challenge was implementing the design and then began running tests and experiments as obstacles arose. When the robotic arm was able to pick up a red object amongst green ones, after running two programs to determine its position and angles, the arm would move, pick up the ball, and then drop it in a predetermined area. The project can be considered as a success in achieving this primary goal as well as achieving its other objectives as well.

2 * Introduction

This space may be used to provide an introduction for the design and ties to other project materials.

2.1 * Design Objectives and System Overview

Our team has designed a robotic arm interface that, with computer vision, is able to pick tomatoes after being given a set of coordinates at which to pick up the tomatoes. We have two different softwares that can detect where a tomato is. The first software uses AI machine learning to detect a tomato while computer vision is used by using RGB in the intel camera.

This project utilizes computer vision and neural networking in order to identify ripe vs unripe tomatoes. It will then proceed to pick up the ripe colored one, which is red, and place it into a box, which is an already established set of coordinates. The robotic arm, combined with the two microcontrollers, would be able to pick tomatoes autonomously.

The design and integration of the hardware and software makes the project meaningful since it allows for autonomization of tomato harvesting. This project has the potential to save time and allow farmers to spend their time elsewhere without the need to put manual labor hours into tomato picking. Which also means that with this product, they will be able to see financial profit as this robot will be able to collect tomatoes without the need to be controlled.

This project is intended to be an investment of future growth for farmers. A farmer can implement this product in their greenhouse, and witness how the robot can collect tomatoes for them. With some simple instructions and computer programming, this robot can also be quite suited for a learning environment for farmers that would be interested in robotic technology for their farms or gardens. The product was developed with these intended applications in mind and to also allow for financial gain.

By using two different microcontrollers, neural networks, and computer vision, this project implements different topics and subjects of electrical engineering.

Our goal for the senior design project was to develop a robot arm manipulator for tomato harvesting. The system was supposed to identify the tomatoes using the RGB camera and the neural network. After detecting the tomatoes, the position of the tomatoes is taken using the depth camera where the geometric theory for inverse kinematics is used to calculate the angles for the robot. The angles will be sent to Arduino to pick the tomato and place it in a box. The gripper was to be made using soft actuators to pick up the tomato without crushing it. Due to the nature and complications of the project, the system was adjusted to perform better with lesser components.

The system detects red balls (tomatoes) using the OpenCV [p1] software and find the pixel points of the balls. The Intel RealSense API calculate the position of the red ball which then used to calculate the inverse kinematics for robot. The angles are inputted to Arduino to pick the closest red ball and place it in a box.

Responsibilities: clearly state which team member is responsible for which goals/objectives
Design Objectives:

- 90% successful retrieval - Pamodya
- 90% AI Model Precision - Priscila
- Voltage range less than or equal to 11.1V -Priscila

Name	Responsibilities:	
Pamodya Peiris	Computer Vision	Inverse Kinematics
Priscila Huante Mendez	Neural Network	Powering the robot
Michael O'Dea	Raspberry Pi	Serial Communication
Karen Gonzalez	Building Robot	Arm manipulation with Arduino

2.2 * Backgrounds and Prior Art

Since the design project's main idea was to program a robotic arm to autonomously pick tomatoes, there have already been quite a few designs and research developments already made, however, they each have advantages and disadvantages when it comes to harvesting. This area of engineering, at least for robotics, can still be considered revolutionary and due to that fact, there is yet to be a product actually released. However, there were plenty of similar designs that were found that established the foundation of this project. The first similar design that was found, is called the Certhon Harvest Robot, which is being designed and developed by DENSO and Certhon. This robot is designed to be able to also autonomously pick tomatoes while being on a track system. It essentially travels up and down rows of a greenhouse and is able to pick tomatoes from off of the plant. One of the advantages that our product has over this one, is that our arm is able to pick plants closer to ground level, and ours is also a lot smaller in size and weight, which allows for more tomato plants to be grown in close proximity. However a big disadvantage that our design has, is that it has a much smaller configuration space than the other design. Both of the designs are similar in respect to using computer vision to pick up tomatoes autonomously.

[https://pipanews.com/tomato-harvest-robot-insane-japanese-technology-robots-to-cut-tomatoes-if-you-look-your-eyes-should-be-distracted-japanese-tomato-harvest-robot-in-action-in-tomato-world-product-of-inoho/]

Another robot design that made us rethink how much better we could have designed our robot is the Metromotion Greenhouse Robotic Worker. It was much more efficient in terms of the type of end-effector they used. Instead of having any type of gripper, it had a rectangular shape that would cut off the entire stem that contained not just one but multiple tomatoes. It would target the stem and close which cuts but also holds onto the steam and then leaves it on the conveyor belt. Another design element that shows our shortcoming is that the robot was able to move around while our was stable. While we had a basket to hold our tomatoes they had a similar design concept but because their robot was larger, their tomatoes went into the robot itself using a conveyor belt. What was similar to our design was that they also used AI based software. Another difference this robot had that shows our shortcoming is that it was bi directional while ours only had a certain range it can move. Especially considering where our camera can reach where as the camera they used as a 3D camera. It also runs 24/7 whereas our robot right now has to recharge after it discharges just 2V. This helps us visualize what other designs can be used instead of something simple, thinking outside the box can make our robot much more efficient.

[https://www.metomotion.com/technology/]

2.3 * Development Environment and Tools

The software portion of this project is a combination of Anaconda Spyder with python 3.7.1, Intel RealSense SDK 2.0, Arduino IDE, and Debian OS. AS uses RS2 to get the camera stream from the IRC and get the xyz position of the red ball. The program uses python library skimage to identify the red balls, numpy to transform the camera frame to the robot frame, and math to calculate the IK.

Another part of the software portion that was made was the AI machine learning model. It was opened and tested via a virtualization docker that had to be through bios. In order to open the python wheel I used the Xailient virtualization that they provided. The code used to open and test our AI model was in python 3.7 using visual studio code. The AI model was created through the software app Xailient. OD HUB was used to annotate the pictures of both actual tomatoes and ping pong balls. We inserted every annotation which was fed to the Xalient software.

The Arduino IDE 1.8.13 was used to publish the data values to the robotic arm for it to move, after obtaining values from AS and RS2. Debian OS was used on the Raspberry Pi 4 to set up the proper environment and establish serial communication (not successful) between the two microcontrollers. The software was primarily run and developed on Windows and then tested on both Windows and Debian. Debian OS was accessed from the computer via SSH as well as using a VNC server and viewer for the RP4.

For the hardware portion, a voltage multimeter and Hewlett Packard power supply was often used for testing purposes to send power to each of the motors as well as the AM. The servos had to have enough power without overloading or underloading. The multimeter was originally used to see what voltage was needed before we officially bought a battery pack. And when we had a voltage that powered the entire robot it decided to fail days later so then we had to use the battery packs to test it with different amounts of servos. A soldering gun and wire were utilized in order to make the perf board that was used after testing the setup on basic breadboards and then were transferred over.

2.4 * Related Documents and Supporting Materials

1. Robotic Arm

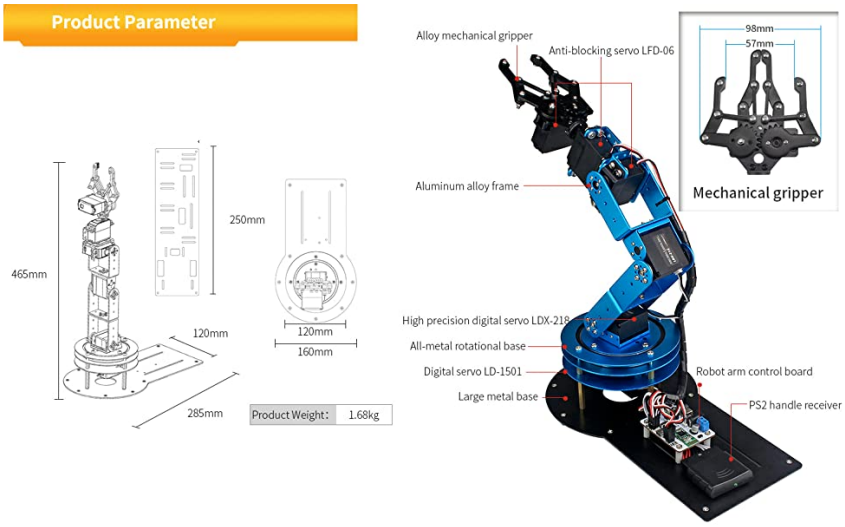


Figure 2.1 Robotic Arm parameters and servos used

2. Intel RealSense Camera D345i
    a. RGB frame rate 30 fps
    b. Depth frame rate up to 90fps
    c. Depth accuracy 2%
    d. Minimum depth distance 28 cm
    e. USB 3.0 serial connection

Figure 2.4.2 Intel RealSense D435i

3. Serial communication standards set by EIA

4. UART connection standards set by USB-IF

2.5 * Definitions and Acronyms

Acronyms	Definitions
RAMFH	Robotic Arm Manipulator for Fruit Harvesting
CV	Computer Vision
IK	Inverse Kinematics
IRC	Intel RealSense Camera
MC	Microcontroller
USB	Universal Serial Bus
OS	Operating System
IDE	Integrated Development Environment
RS2	Intel RealSense Software Development Kit, pysrealsense2 library
AS	Anaconda Spyder
SSH	Secure Shell
VNC	Virtual Network Computing
RP	Raspberry Pi 4
AM	Arduino Mega
SBD	System Block Diagram
HPPS	Hewlett Packard Power Supply
UCR	University of California Riverside

3 * Design Considerations

This section describes issues that need to be addressed or resolved prior to or while completing the design as well as issues that may influence the design process.

3.1 * Realistic Constraints

The first constraint that had an impact on the design of the system, was the voltage and current requirements that had to be met and tested for the different servo motors. The power consumption was alright considering we tested it about 200 times in six weeks however when it was connected constantly the voltage went down quickly so it wouldn't be as efficient for constant use. Therefore, our constraint was our power consumption along with our battery size.

Another constraint that has had a significant impact on our design is the style of the gripper. We had to consider a style of gripper that would be somewhat soft to the touch and would not damage the produce. We came upon a lot of research on pneumatic silicon grippers that would be a great addition to our robotic arm but could not get one that was accessible. We attempted to have a gripper mold 3D-printed to then create our gripper using silicone. In the end we could not obtain the mold needed due issues with the 3D-printers. For our final demonstration we used the gripper supplied with the arm which was only a 2 tipped gripper versus the 3 tipped gripper we were hoping for. In that case we also switched our target to pick up.

The gripper constraint led to another which was essentially moving from using actual tomatoes to balls of red and green colors to resemble our tomatoes. As well as the fact that it would be difficult to obtain a tomato plant at the time that would be suitable for testing. If we had chosen to go with actual tomatoes, we would have to constrain our size due to the fact that our arm could only hold up to 250 grams.

When it comes to DOF of our arm we wanted to have an arm that would have 6DOF. If we were to imagine our arm working and picking fruit, we want it to have as much movement capability as possible to reach them. We had to ensure the robot arm we selected could satisfy that requirement.

The next constraint that was heavily researched in order to implement into the design, was microcontroller processing speed and size. Both microcontrollers needed to have the fastest processing speed to allow for autonomization and to also allow for an increased frame rate stream from the LC. A faster speed would mean less lag and stall time.

An important constraint we had to consider was affordability and accessibility. We could only afford so much to create our project so we minimized the number of items and tools we may need and went for the ones with reasonable prices. In the beginning we did want most of our parts to be 3D printed so as to reduce the cost of larger parts like the arm and gripper. Still, there was some difficulty with attempting that route. There were technical issues with the printers themselves so it resulted in purchasing our arm at a reasonable price that would still provide us the movement we need.

3.2 * Industry Standards

The system uses Python 3 and C programming languages to code the program. The system uses Anaconda OS and Arduino IDE. The Raspberry Pi use Debian OS. The system uses USB/UART to send

the camera stream to the PC and the angles to Arduino. The program is written on Windows OS. The program uses the computer storage to save the image streaming data.

The servos need a voltage between 6-7 V with a current of 200mAs. If the current exceeds 600mAs, the servos over power and doesn't function. If all servos relate to a 12 V battery, then it doesn't function as well. The baud rate is 9600. This is used when the serial connection is established between the Arduino and the Python program.

3.3 * Knowledge and Skills

Michael O'Dea:

- Classes:
  - ENGR180W (Technical Communications)
  - CSEE120B (Embedded Systems)
  - EE100A (Electronic Circuits I)
  - EE144 (Introduction to Robotics)
- New Skills:
  - Raspberry Pi 4 Implementation and Integration
  - Learned how to use Serial Communication
  - Learned how to operate on Debian OS
  - Learned how to set up and compile libraries that needed workarounds based on the RP
  - Power System

Priscila Saray Huante Mendez

- Classes:
  - ENGR180W (Technical Communications)
  - CSEE120B (Embedded Systems)
  - EE100A (Electronic Circuits I)
  - EE144 (Introduction to Robotics)
  - CS170 (Intro to AI)
- New Skills:
  - Virtualization Via BIOS
  - Learned how to use a Docker
  - Learned how to use an Arduino
  - Learned how to open python wheels thru virtualization
  - Learned how to build Buk converter

Pamodya Peiris

- Classes:
  - ENGR180W (Technical Communications)
  - CSEE120B (Embedded Systems)
  - EE100A (Electronic Circuits I)
  - EE144 (Introduction to Robotics)

- CS171 (Intro to Machine Learning and Data Mining)
- EE146 (Computer Vision)

- New Skills:
  - How to use Intel RealSense SDK
  - How to interpret data collected from a camera stream
  - How to change reference frame from one frame to the other
  - How to calculate IK for a robotic arm using geometric and analytical methods
  - How to apply for grants
  - How to use OpenCV
  - How to use python libraries and functions

Karen Gonzalez

- Classes
  - ENGR180W (Technical Communications)
  - CSEE120B (Embedded Systems)
  - EE100A (Electronic Circuits I)
  - EE144 (Introduction to Robotics)
- New Skills:
  - Learned about possible pneumatic gripper method
  - Learned to manipulate the robot arm using the pre-programmed microcontroller
  - Learned to connect servos to Arduino
  - Learned how Arduino Mega functions
  - Learned to use Arduino to control servos

3.4 * Budget and Cost Analysis

With this project, there was an allocated amount of a $400 budget, and our total spent was $360.12. The design project consisted of various hardware components connected, which was heavily researched to get the best hardware available for the project while also making sure that the components that were needed were within our budget. Considering that there were a lot of variations of the components, ranging from the microcontrollers that were used, to the robot arm itself, the robot arm functionality and the microcontrollers functionality was limited by the budget.

Type	Name	Quantity	Price
MC	Arduino Mega 2560	1	$38
MC	Raspberry Pi 4	1	$35
Robotic Arm	Lewan Hiwonder Arm	1	$139
Battery	Ovonic 11.1V Battery	2	$30.99

Fabric	Black Fabric	1	$9.99
Base	Wood Base: 18” x 18”	1	$25.99
RGB-D Camera	Intel Realsense D435i	1	$329
Other			$81.15
UCR Undergraduate Education Mini-Grant			-$500
Total	——----------	——----------	$360.12

Some materials and items that were used that are not listed above, are ones that weren’t purchased but they were purchased for previous projects that were done and classes that were taken. A $500 mini grant was received for this project from the Undergraduate Education at UCR which allowed for the purchase of the Intel RealSense D435i camera that was implemented.

3.5 * Safety

One of the main safety considerations that was taken into account during project development was regarding the voltage and power levels that the robotic arm should receive. Depending on what the motors require for voltage, the risk of potentially miswiring the robotic arm could lead to a chance of an electrical fire occurring. This is due to the electrical wires not being able to handle the voltages running through them. This impacted our design to research a robot that would have motors to be easily powered by low voltages, but still able to achieve the end goal of the project.

The servos need a voltage between 6-7 V with a current of 200mAs. If the current exceeds 600mAs, the servos over power and don't function. If all servos relate to a 12 V battery, then it doesn’t function as well. After many experiments were done, this was the best result that would work on the design project. This can be a safety consideration

Another safety concern that was looked at, was into the safety of the robotic arm itself. Once the robot arm starts to move to a specific set of coordinates, it is not easily halted unless the power is disconnected. This leads to the potential of causing bodily harm if someone accidentally comes into contact with the arm while it is moving. In regard to this, the design was expanded in order for the arm to be put on a demonstration board for customers to see the potential, while also keeping a safe distance away from the robot while it is in motion.

3.6 * Documentation

User documentation was mainly kept on a shared google drive that was shared amongst the team. Any changes to the final report or to technical designs or even to individual progress reports, typically had comments to track history and changes that were being done. This also was implemented for the different versions of the final report which also contained history of what was added and changed on the document.

3.7 Risks and Volatile Areas

One of the main risks of this project, includes the utilization of the camera. Before the IRC was obtained, a computer vision program was constructed and trained for the camera that was almost purchased to be used with it. However, after receiving the IRC camera, the computer vision program was not needed anymore but with this new camera came new challenges. To simplify and increase the success rate of the camera, the tomatoes were switched to primary-colored balls. This allowed for the camera to easily recognize which balls were which, and which ones it should calculate the position of. Lighting was also an issue, since too much light can cause interference and errors with the camera attempting to register the color of an object in front of it. To fix this problem, black fabric was purchased to absorb light and prevent reflection off the surface that the camera was on.

The other risk that was challenging, was the utilization of remote virtualization for the robot to be autonomous. This was attempted to be solved by having the RP send data to the AM via serial communication and connection, which would allow for the robot to be controlled remotely as long as the RP was connected to the same Wi-Fi. The RP would run the program, which would pull data values from the camera, and then send the movement coordinates to the robotic arm. By SSH, the camera output and program would be displayed virtually. In case this risk area failed, the backup plan was to program the AM by having it be connected to the main computer and computer vision program by USB.

4 * Experiment Design and Feasibility Study

4.1 * Experiment Design

The first experiment that was conducted for this project, was concerning voltages and power for the servo motors. After researching what the servos would require, voltage level wise, the decision was made to purchase two LiPo batteries with 11.1 volts each. The setup of this experiment was designed to first test to see if one battery would be enough to power the robot. If it was too much then conductors would be used to lower the voltage, and if it was not enough then the second battery would be implemented into the circuit as well to provide power.

The setup involved the motors of the robotic arm, being connected to the AM for movement and then connected to the breadboard in order for power to be supplied to them. In order to determine what voltage and amps would be required for the servos, a Hewlett Packard Power Supply was implemented for this round of experimentation. If one alone was not enough, then we would use two power supplies.

The procedure followed around these steps:

1. Wire motors together getting power from the same breadboard, if repeating this step then separate the motors into groups.
2. Start at a low voltage power from the HPPS and test with different values inputted and published from the AM to the different motors to see if there is any movement.
3. If there was, or was not, write down what the voltage was and how many amps were being outputted.
4. If at any time the servos function improperly, or if the wiring is faulty, stop the procedure and write down what is happening and attempt to fix the problem.
5. Once the HPPS hits 12V or the servos start to stall, reset the experiment to 1V and attempt to use two HPPS and split the servos into two groups.
6. Increase the voltage by 1V, and repeat steps two and three, and if need be step four.
7. Once the servos start to move appropriately and there doesn't appear to be any errors with the functionality of the arm or issues with the electrical wiring, then consider the experiment a success and record what voltages are needed to power.

From the above experiment and testing of the servo motors, the expected result should be an output voltage within an appropriate range from the HPPS, and from there, batteries will be purchased accordingly. Priscila Huante Mendez and Michael O'Dea were primarily responsible for this part of the project.

4.2 * Experiment Results, Data Analysis and Feasibility

We first researched what the max voltage to power the servos was and found an 11.1V battery pack that would work for our project as it had 3 levels (3.1, 7.1, and 11.1). After setting up the servos to the Arduino we had to play with the right voltage that would power all the servos without overloading and underloading it. We used a voltage multimeter to power our servos and set different numbered voltages and stayed with whatever worked best. When it was first tested, we found that at 6V everything was powered. So, Priscila was in charge of making a buck converter that decreased the voltage of the 7.1V

mode to 6V. We ended up buying a 6V battery pack as well as back up but its discharge after one use, so it wasn't used at all. The buck convertor was able to reduce the voltage to 5.67V which still powered all the servos.

Although when first tested, 5.67V was enough to power the servos without overloading or underloading it. However, it suddenly failed to move servos 6 and 4. Priscila and Michael were testing each servo one at a time with different voltages to see what the optimal power for each servo was to function properly. It took a lot of testing to finally get servo 6 and 4 to move. Anything under 5.5V would not power the servos at all and as soon as it hit 7V the breadboard wires connected to the voltage multimeter started to smoke because it was overloading. You can also hear the servos rattling a lot because of how much power it was being given. All the servos were able to be powered with 6.85V and .2 amps of power.

Once again, the servos failed to fully power all the servos when we powered it on again. Only servo 4 wasn't moving this time. Priscila and Karen were trying to figure out why servo 4 was powering up and once again experimented with the number of voltages. The odd part about powering the servos is that 7.1 was a good number to power 2 sets of 3 servos. So servos 1, 2, and 3 were powered at 7.1V and servos 4, 5, 6 were also powered at 7.1V. This was odd because just giving it 7V to power everything was too much that it would start to smoke but it wouldn't do that when only connected to 3 servos. After powering it off after several days, there was no problem with any servos so having 7.1 for 3 servos worked at every given point without complications.

5 * Architecture and High Level Design

The architecture provides the top level design view of a system and provides a basis for more detailed design work. These are the top level components of the system you are building and their relationships.

5.1 * System Architecture and Design

In designing the SBD, the first main concept that was formulated was how the system would be powered. The system could either be powered by computer, a wall outlet, or purchasable batteries. If the system was powered by a computer, that would mean the design would have to be limited to what the computer's output would be for voltage. The same issues would occur had the design been based around being powered by a wall outlet. Different wall outlets and computers also let out different voltages, and since testing would be done in multiple locations, it would lead to errors with the motors and power. The last option was purchasable batteries, and for this reason, the design was based around what the motors would require for voltages.

Another portion of the project that the design was heavily based around, was communication. To combat this and attempt for our project to achieve autonomization, the design and project implemented two microcontrollers. The primary microcontroller would contain the CV program, and calculate the necessary values, and then send them to the second microcontroller via serial connection and communication. The second microcontroller would take those values, and then send them to the motors for them to move to the required coordinates. The two microcontrollers were needed instead of one, because one alone would not have been enough due to technical challenges and individual microcontroller limitations.

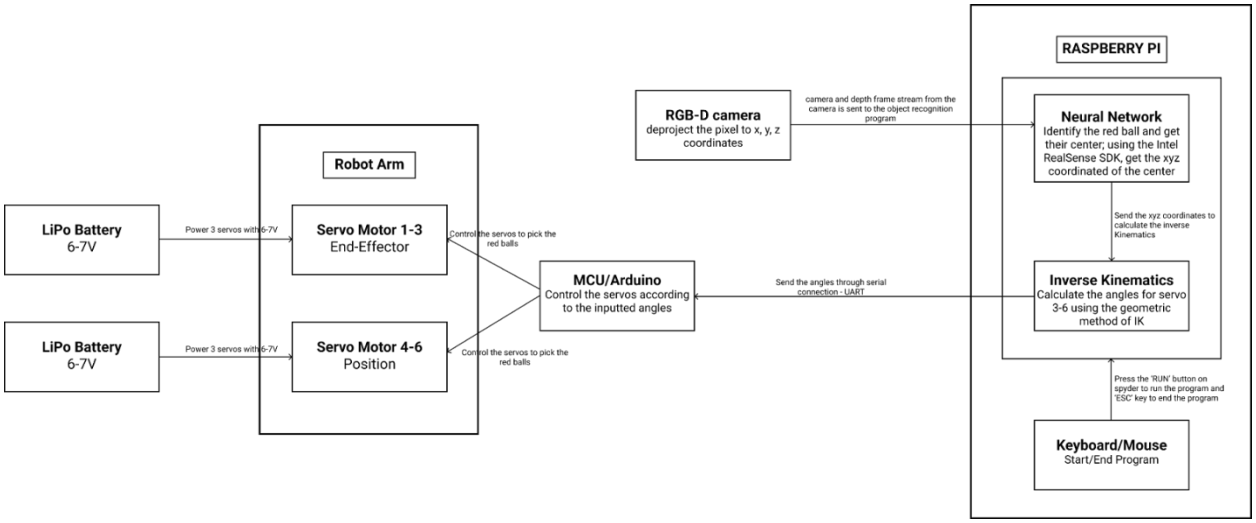


Figure 5.1.1: Visioned SBD

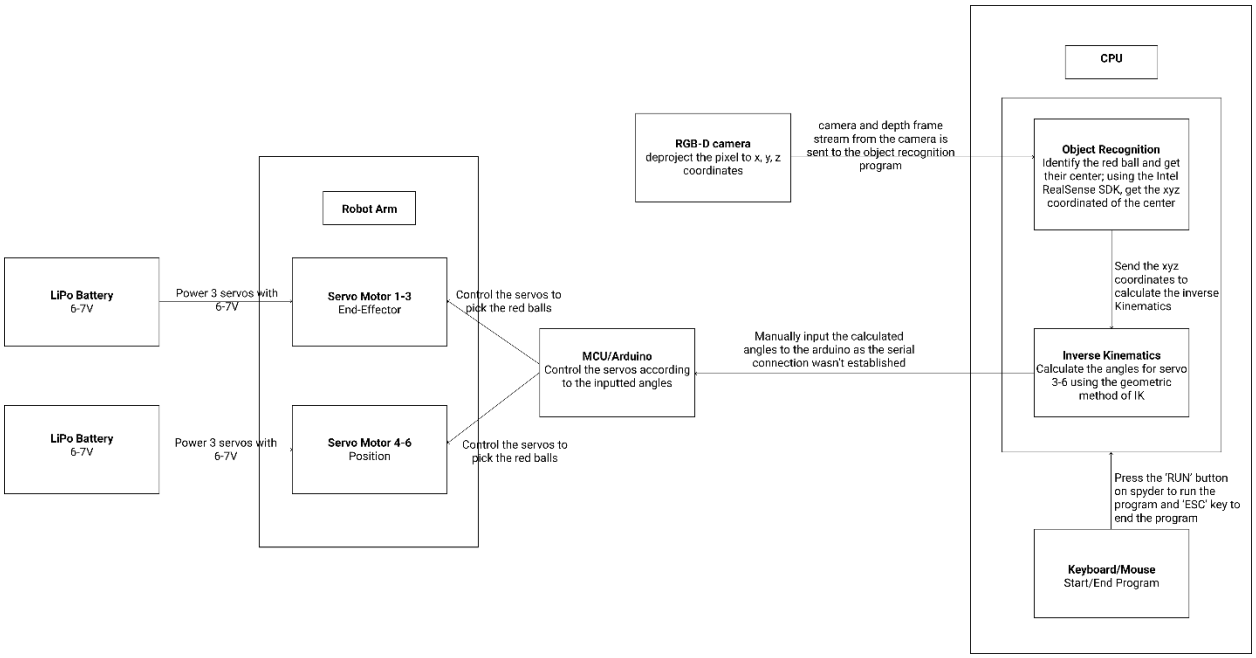


Figure 5.1.2: Final SBD (Appendix F)

Block	Responsible Person
Power System	Priscila
Arduino IDE Code	Karen
Inverse Kinematics	Pamodya
Object Recognition	Pamodya
Neural Network	Priscila
Raspberry Pi	Michael
Serial Connection	Michael
RGB-D camera	Pamodya

Table 5.1.1: Responsibilities for each block in the SBD

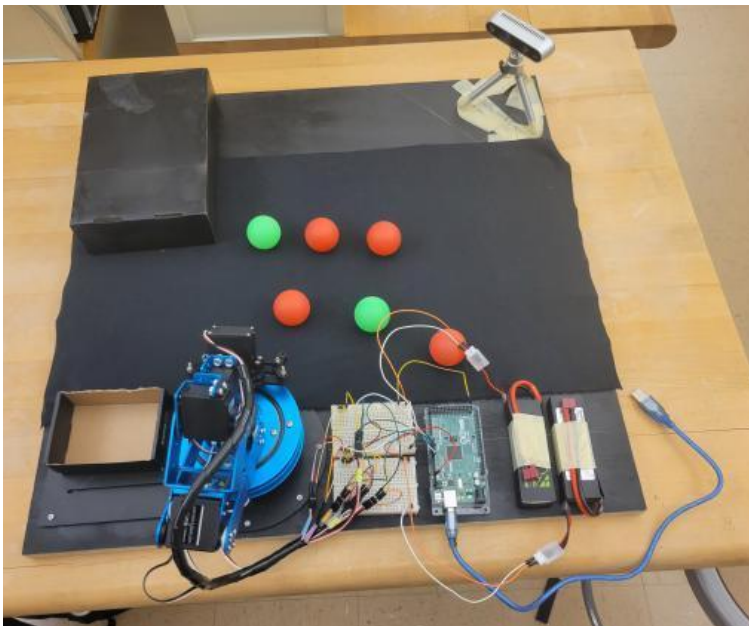


Figure 5.1.3 Connection without Raspberry Pi

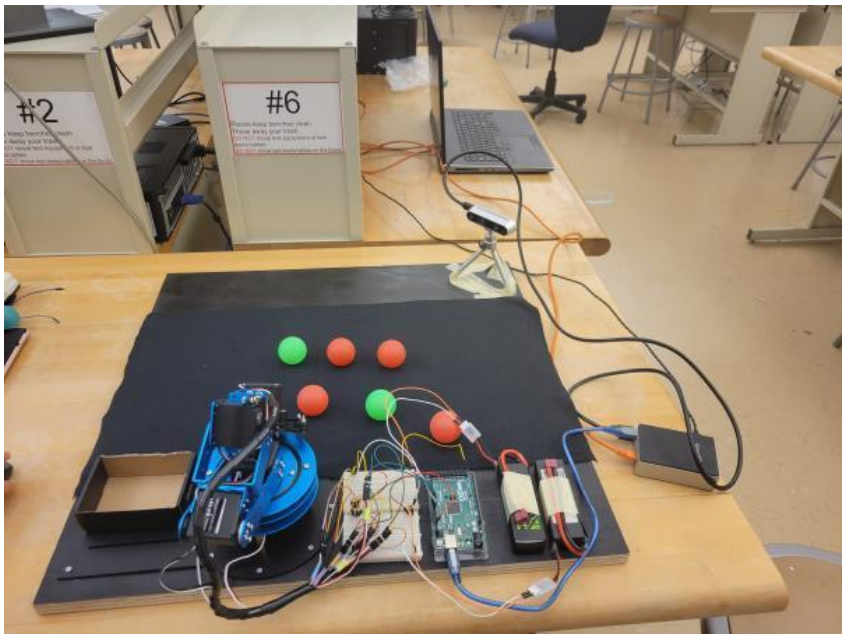


Figure 5.1.4 Connection with Raspberry Pi

Figure 5.1.1: This image is the high-level block diagram of the intended system of what this project attempted to achieve and accomplish. The CV program on the RP would run the RGB-D IRC and send the coordinates of where the object is to the IK program, which is also on the RP. The RP would be connected to the Wi-Fi via an Ethernet connection, and SSH would allow for remote access and control in

order to control and run the programs. After calculating the values, they would be sent via serial connection to the AM, which would register those values and move the motors accordingly. The servos themselves are split into two groups, which are each powered by a separate battery. This was the original design that was heavily implemented and tested. The implementation is shown in Figure 5.1.3.

Figure 5.1.2: This image displays the high-level block diagram of the finalized system that was implemented. The primary microcontroller, the RP, was removed due to serial connection issues, and was replaced with a desktop CPU. In this design, the desktop contains both programs and gets data values from the RGB-D IRC which is connected by USB. After manual input of the coordinates into the Arduino IDE program, the values would then be published to the AM for the robot arm to move to the required positions. Those are the only changes from the originally designed implementation. The implementation is shown in Figure 5.1.4.

Table 5.1.1: This table shows the responsibilities of each team member for each module or connection.

5.2 * Hardware Architecture

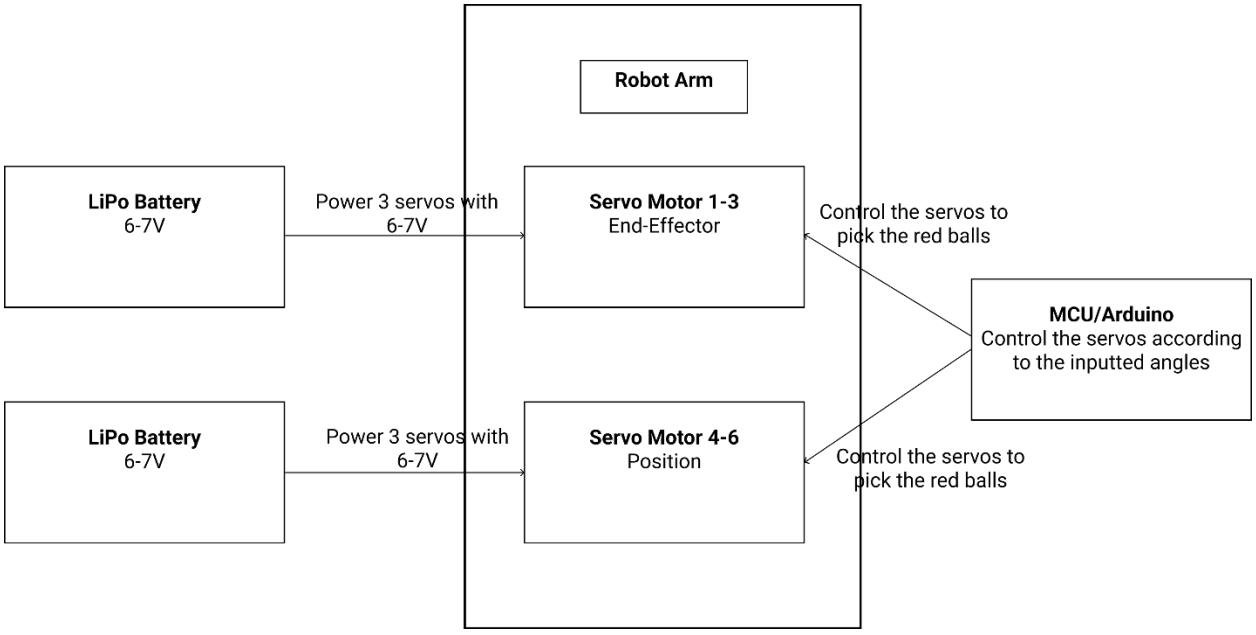


Figure 5.2.1 Hardware Architecture

Responsibilities	Name
Powering the robot using LiPo batteries	Priscila
Building the Robot Arm	Karen
Arduino Code	Karen

Table 5.2.1 Hardware architecture responsibilities

Table 5.2.1: This shows the hardware break down of the project. Each 3 servos are powered using a LiPo battery which ranges between 6-7 V. The servos cannot be powered together due to the increase in current. The Arduino code has three positions for the robot: 1. Home position 2. Pick the ball 3. Drop the ball into the box. Figure 5.2.2 shows the numbered servos.

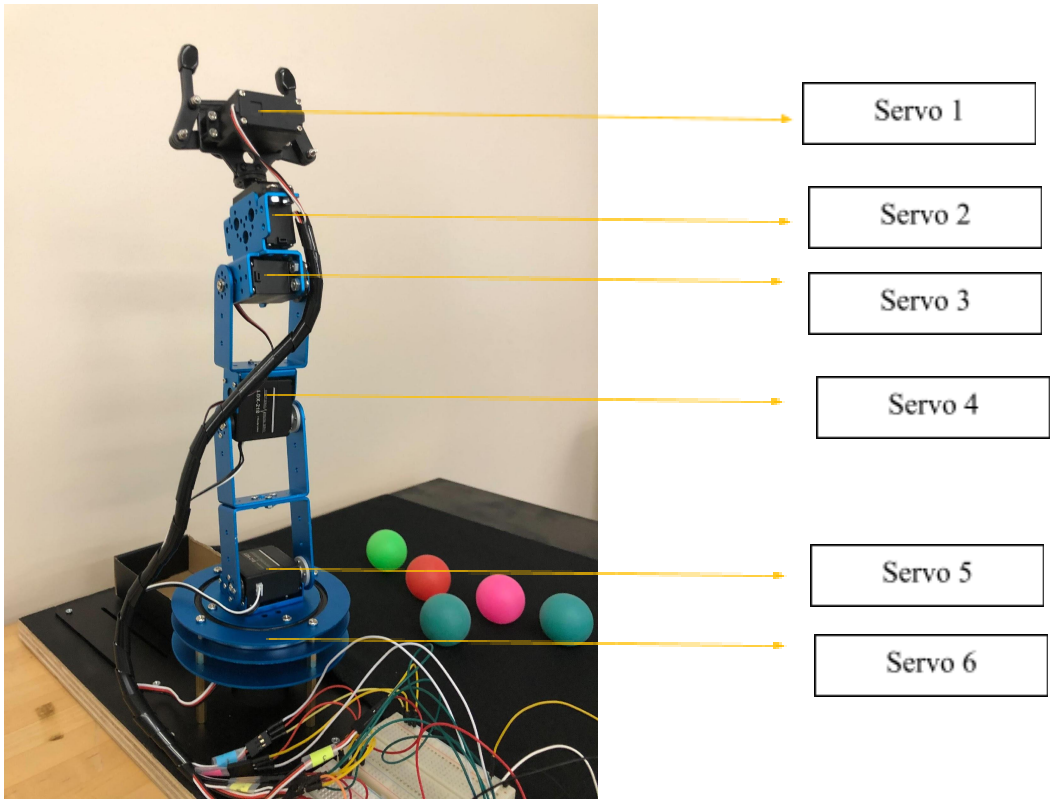


Figure 5.2.2 Servo break down

5.3 * Software Architecture (only required if your design includes software)

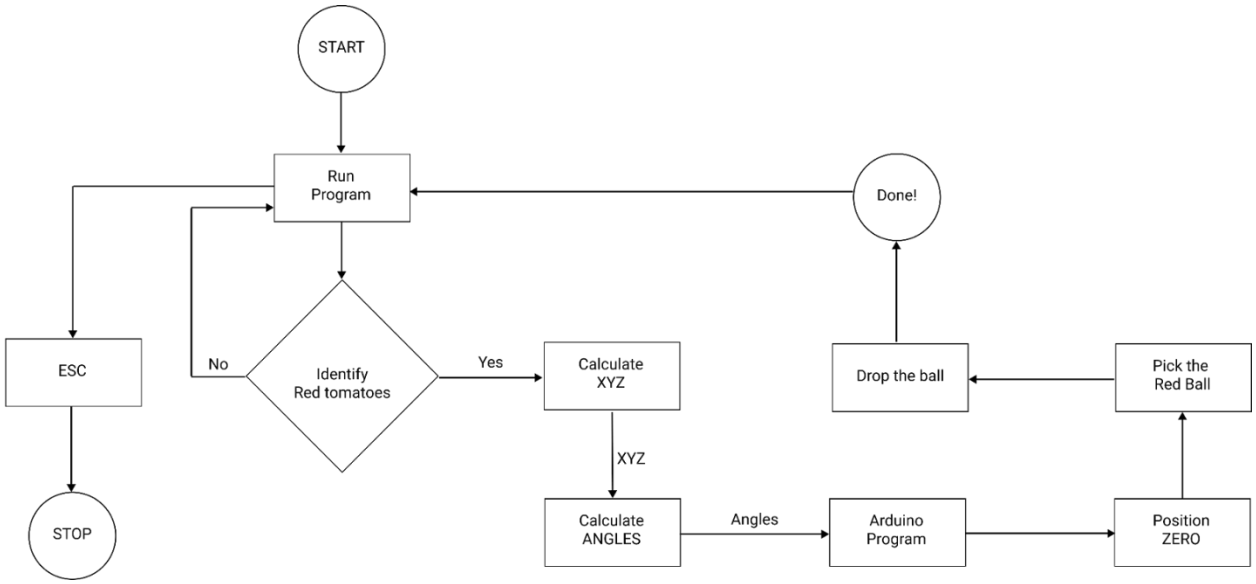


Figure 5.3.1: Software Block Diagram

The figure 5.3.1 displays the software block diagram that was designed for simplicity and integration with the robotic arm. After having the program start, it would first identify if there are red tomatoes in its view. If not, it will continue to run until it registers that there is a red tomato in front of it. Once it does, it will then proceed to calculate the x, y, and z coordinates of the tomato. From there, the IV program will calculate the angles and send them to the AM program via manual input. The robot arm would move from its zero position to then pick up the red ball and drop it to a specified place. The program will continue to run until it is manually closed.

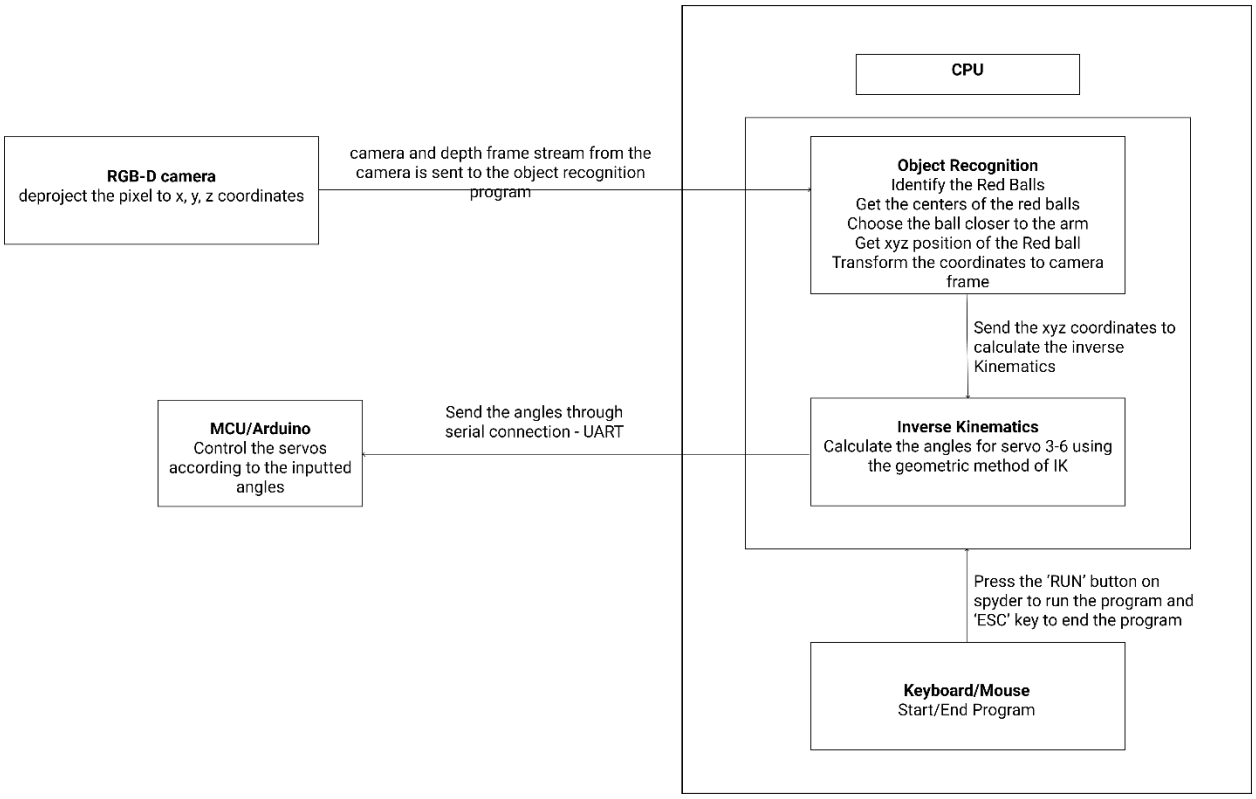


Figure 5.3.2 Software Architecture

Responsibilities	Name
Object Recognition	Pamodya
Inverse Kinematics	Pamodya
RGB-D camera	Pamodya
Serial Connection	Michael
Arduino	Karen

Table 5.3.1 Software Architecture responsibilities

Figure 5.3.2: This software architecture shows how the program recognize the red balls and calculate the angles for the servos of the robotics arm to pick up the ball. The IRC provide real-time image stream of color and depth images, and the program starts to run when the run button is pressed on the AS. Pressing the 'ESC' button stops the stream and end the program. The Object Recognition program identifies the red balls and extract the pixels of the center of the balls first. Then, the program gets the xyz coordinates of the pixel of the center ball that is closest to the robot (furthest from the camera). It transforms the coordinates from the camera frame to the robot frame which is sent to the IK program. The IK program calculates the angles for servo 3-6. Servo 1 and 2 is changes directly on Arduino as servo 1 is the amount

of grip it needs and servo 2 is always set to 90 degrees due to its orientation. The serial connection that is supposed to send the angles to Arduino was not successful.

The camera stream is a while loop which show the color and the depth frames with a frame rate of 30 fps (frames per secon4. The Object Recognition and IK functions are called inside the loop every 120 seconds.

5.4 * Rationale and Alternatives

The design goal was to make a robotic arm capable of picking tomatoes, after registering where the tomato was from CV and then using IK to calculate the proper movements of the arm to accomplish the needed task. The original plan was also designed for autonomization without the need for manual input. In order to achieve this goal, after heavily researching robotics and programming for CV, the decision was made to use two microcontrollers.

The batteries in the design were also decided upon after heavily testing how the motors should be powered and what the best voltage was for them to be able to achieve their best performance without overloading or underperforming. The layout regarding the two LiPo batteries was the result of the experiments.

During the time of the implementation, the camera stream has to be continuous without any disturbances to calculate the xyz coordinates. Due to that, the program runs the camera stream in the main function while calling the Object recognition and IK functions in the while loop every 120 seconds till the program is stopped.

6 * Low Level Design

This section provides low-level design descriptions that directly support construction of modules. Normally this section may be split into separate documents for different areas of the design. For each component we now need to break it down into its fundamental units or modules. Each module or block may be hardware or software or a subsystem implemented using hardware and software. Make sure to provide a well-commented schematic of all modules and/or system blocks described in the system block diagram (SB4.

Must include hardware circuit schematics and program flowchart for each module.

Example of circuit schematics: http://www.arduino.cc/en/uploads/Main/arduino-uno-schematic.pdf
Example of flowcharts of your programs:
http://www.mdpi.com/electronics/electronics-03-00636/article_deploy/html/images/electronics-03-00636-g011-1024.png

6.1 Powering up the Robot

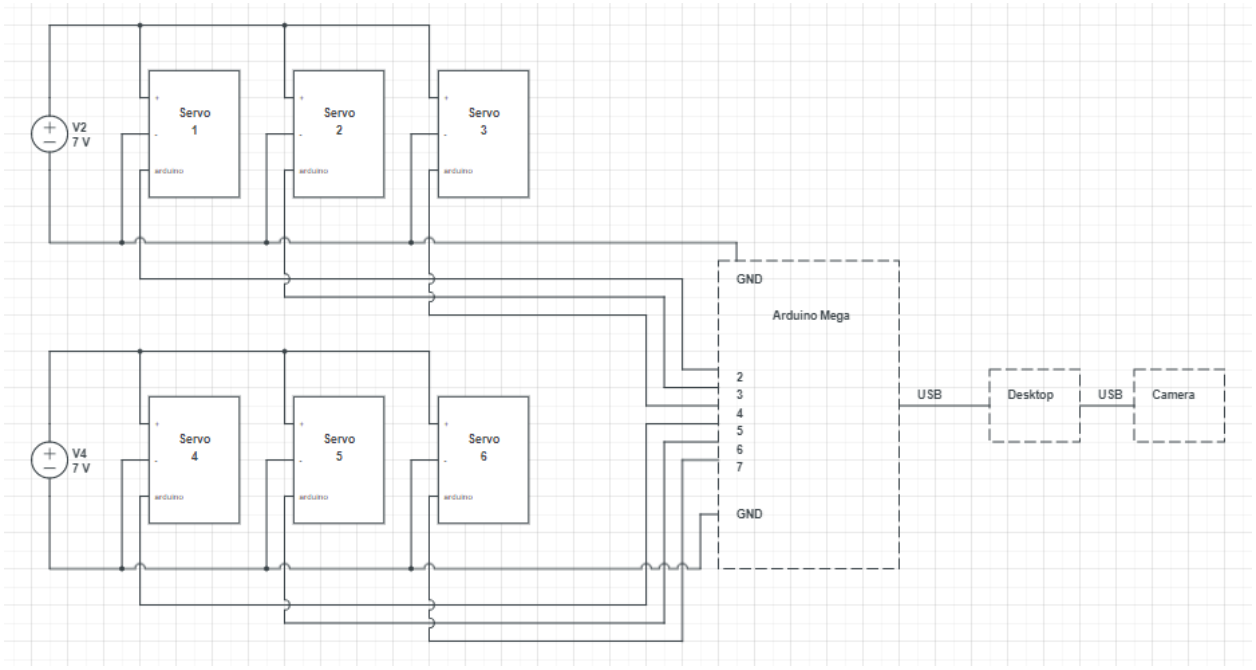


Figure 6.1.1 Powering up the robot with two 7V LiPo batteries and connecting them to Arduino

Responsible person: Priscila

6.1.1 Processing narrative powering up the robot

A processing narrative for module i is presented.

6.1.2 Powering up the robot interface description

A detailed description of the input and output interfaces of the module with other modules in the system, with other software or systems or module is presented.

6.2 RGB-D camera

Figure 6.2.1 Intel RealSense Camera D435i

API documentation: https://dev.intelrealsense.com/docs?_ga=2.56882863.427965197.1647212862-1040313709.1641767671

Github with recourse code: https://github.com/IntelRealSense/librealsense

Responsible person: Pamodya

6.2.1 Processing narrative for RGB-D camera

The camera provides visual information of the surface to identify the red balls and the position of them.

6.2.2 RGB-D camera interface

The stereo camera input two images using two cameras in the ends and the depth using an infra-red sensor. The camera output the color and depth images as a stream to the object recognition program. Both images are of size 640x480.

6.2.3 RGB-D camera processing details

This program uses the Intel RealSense API and the code provided by the Intel Corporation to start the camera stream for color and depth. It also uses their code to align the color and depth image. The alignment of these two images is necessary when getting the XYZ coordinates from the pixel coordinates.

The pixel coordinates are in the color image and the XYZ coordinates are in the depth image. To make that the take XYZ corresponds to the pixel point, alignment is necessary.

6.3 Object Recognition program

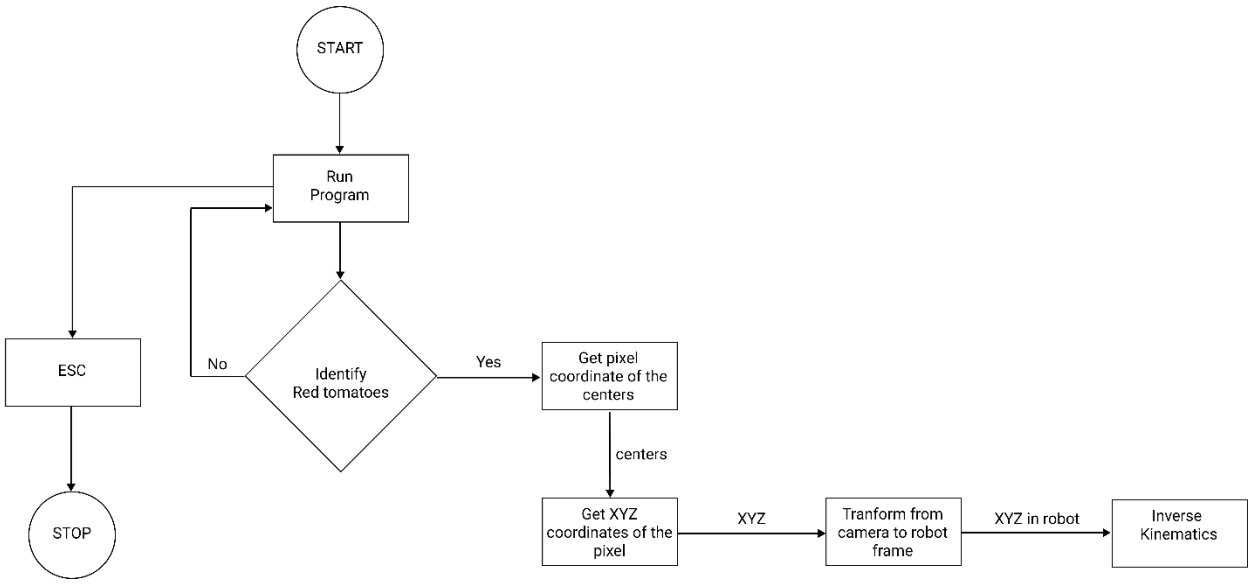


Figure 6.3.1 Flow chart on the Object Recognition program

Responsible Person: Pamodya

6.3.1 Processing narrative for Object Recognition Program

The program uses the color image to identify the red balls and find their centers. Then the RS2 is used to calculate the XYZ coordinates in meters of the red ball that is closest to the robot according to the camera frame. Then, transform the coordinates to robot frame as the IK program needs the coordinates from the robot frame to calculate the angles to pick the red ball.

6.3.2 Object Recognition Program interface

This program input both the color and depth frames from the camera to output the XYZ coordinates of the red ball closest to the robot from the robot frame.

6.3.3 Object Recognition Program processing

6.3.3.1 Identify the red balls and get their center pixel using OpenCV

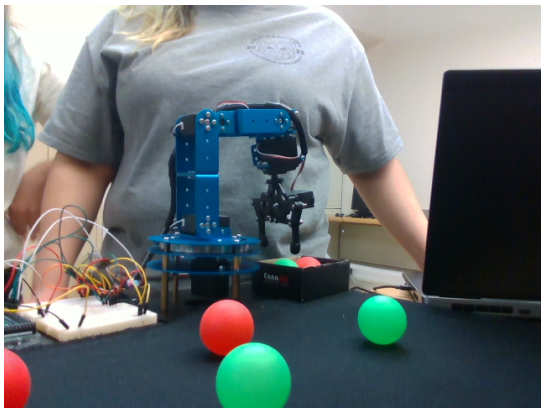


Figure 6.3.1 Original color image

Figure 6.3.2 Red image

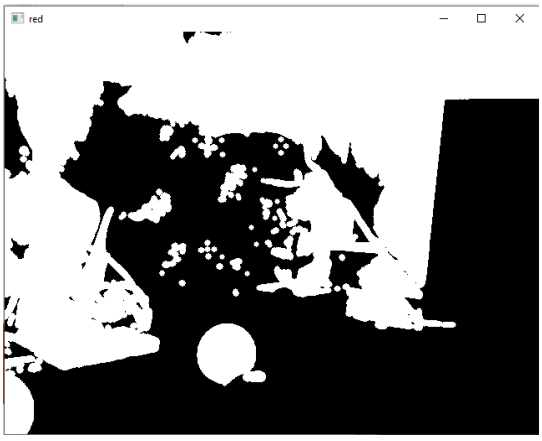
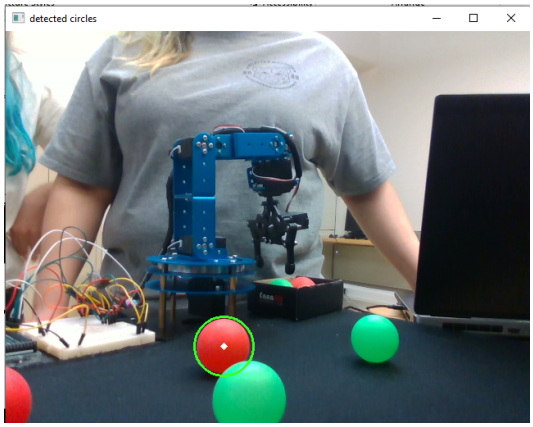


Figure 6.3.3 OTSU thresholder image



6.3.4 Hough Circles

After 30 seconds, the color frame at the 31st second is sent to identify the red balls at the start of the program. After the first loop, there's a 120 second wait and the next frame is sent to the program. Figure 6.3.1 shows the original color image seen by the camera. A color image has RGB channels which corresponds to Red, Green, and Blue. As the system needs to identify the red ball, the red (channel) image, Figure 6.3.2, is extracted from the three-dimensional color matrix. The intensity value of the pixels ranges from 0-255 in the red image. The values closer to 255 corresponds to the pixels that are closer to the color red. After calculating the intensity range of the center pixel of balls, it is identified that the pixel portion of the red ball range between 200-255. Therefore, Otsu thresholding algorithm is used to binarize the image into two: in Figure 6.3.3, black corresponds to pixels with intensity level below 200 and white corresponds to pixels with intensity level above 200.

After thresholding the image, the program performs a morphological operation named dilated to complete the circular shape of the ball. This use the python library sklearn. After completing the circular shape, the program uses the HoughTransform function in the OpenCV library to detect the circles and get the x, y pixel coordinated of the center. This allocates a minimum radius of 16 and a maximum radius of 50 to detect the red balls in the visible range of the camera. When the ball is closer to the camera, the radius is 50, and when the ball is far from the camera, the radius is 16. 50 and 16 are in mm. param2 in this function has to range between 0.87-0.89 to identify the accurate circles and also not give multiple circles to the same ball. To avoid getting multiple circles for the same ball, it also set that there shouldn't be centers of circles with in 5mm.

6.3.3.2 Get XYZ coordinates from the camera frame

The circles extracted from the HoughTransform is arranged in a decreasing order depending on their radius. Low radius means they are further away from the camera and High radius means they are closer to the camera. The x y points on the very top of the HoughTransform list is chosen to get the coordinates. The RS2 has a built-in function that calculates the XYZ coordinates that corresponds to the pixel coordinates of the color image using the saved camera information. This is the reason that the align function is necessary to make sure that the color and the depth images are aligned. Using this function, the program gets the XYZ coordinates of the center of the red ball that is furthest from the camera.

6.3.3.3 Transform the camera frame to robot frame

Since the coordinates are from the camera frame, they need to be transformed into the robot frame.

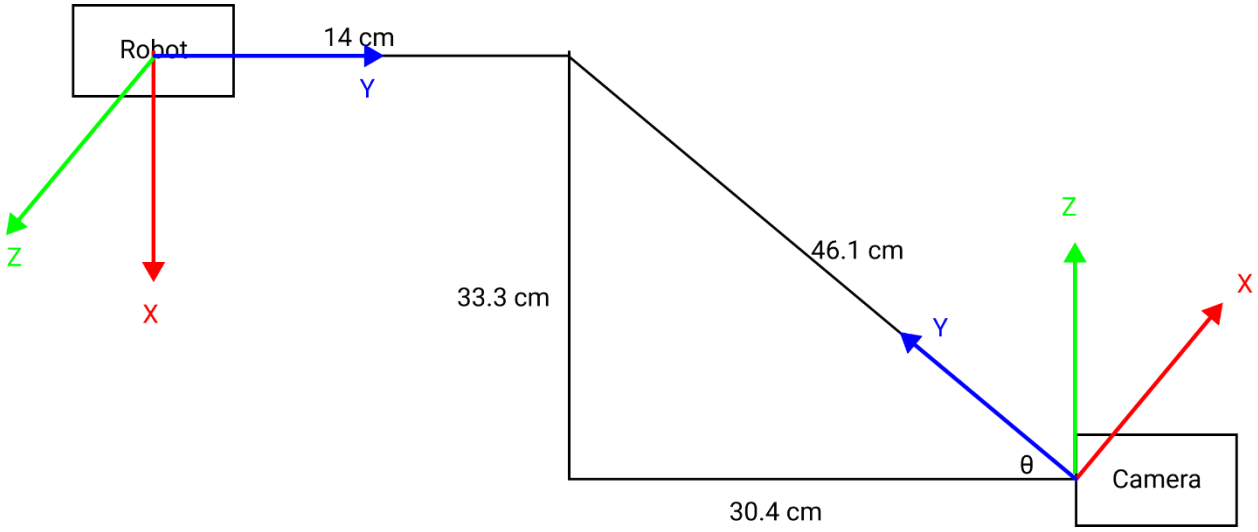


Figure 6.3.5 Parameters between the camera and the robot

To transform the camera coordinates to robot coordinates as seen in Figure 6.3.5, the function below is used.

$$Trans(Y, 46.1)Rot(Z, 90 + \theta)Trans(Y, -14)$$

Eq 6.4.1

$$\begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0.461 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\pi+\theta) & 0 & -\sin(\pi+\theta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\pi+\theta) & 0 & \cos(\pi+\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -0.14 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

Eq 6.4.2

The program uses this transformation to convert the XYZ coordinates from the camera frame to the robot frame to send to the Inverse Kinematics program.

6.4 Inverse Kinematics

Servo 3	Theta 4
Servo 4	Theta 3
Servo 5	Theta 2
Servo 6	Theta 1

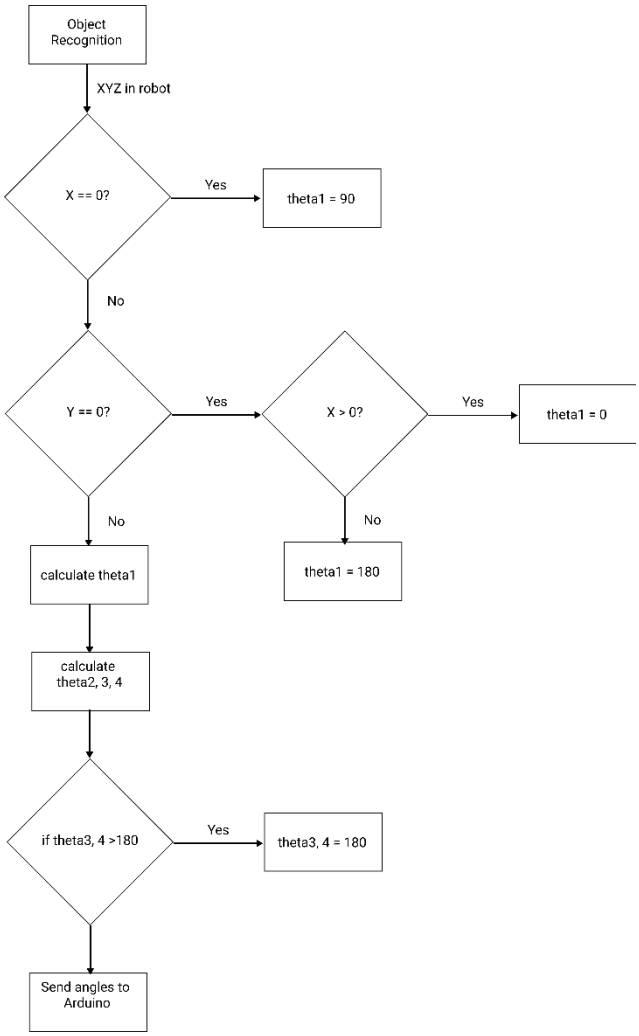


Figure 6.4.1 Inverse Kinematics flow

Responsible person: Pamodya

6.4.1 Processing narrative for Inverse Kinematics

This calculates the angles for servo 3-6. Due to the nature of the robot, servo 2 is always 90 degrees and servo 1 is the gripping strength changed accordingly using milliseconds.

6.4.2 Inverse Kinematics interface description

The inputs are the XYZ coordinates of the red ball closer to the robot (furthest from the camera) from the robot frame.

6.4.3 Inverse Kinematics processing details

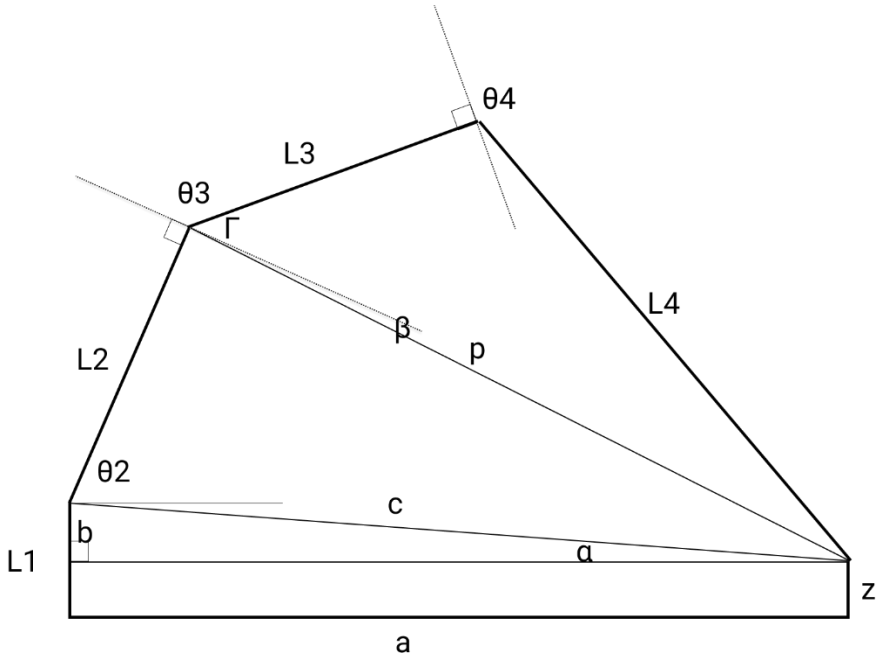


Figure 6.4.2 Geometric model to calculate IK

All the symbols used in the equations correspond to the symbols in Figure 6.4.2

L1 = 0.096 m, L2 = 0.105 m, L3 = 0.089 m, L4 = 0.135 m. The servos range from 0-180 degrees.

If x = 0, theta1 = 90. If y = 0 and x < 0 theta1 = 180. If y = 0 and x > 0, theta1 = 0. This is due servo 6 ranging between 0-180 degrees. Otherwise, theta1 is calculated using eq 6.4.1.

$$\theta_1 = \arctan(\frac{y}{x})$$ ----------Eq. 6.4.1

The robot arm can only reach balls within a range of 14 cm – 26cm. With that, a range is assigned to theta4 which is shown in Eq 6.4.2.

$$\theta_4 = [110 : 150] for (26cm - 14cm)$$ ----------Eq. 6.4.2

After finding the value for theta4, p in Figure 6.4.2 must be calculated to find theta 2 and theta3.

$$\alpha = \arctan(\frac{L1 - z}{a})$$ ----------Eq. 6.4.3

$$a = \sqrt{x^2 + y^2}$$ ----------Eq. 6.4.4

$$p = \sqrt{L_3^2 + L_4^2 - 2L_3L_4\cos(\frac{2\pi}{3} - \theta_4)}$$ ----------Eq. 6.4.5

Eq 6.4.5 is used to calculate p.

$$\theta_2 = \arccos\left(\frac{L_2^2 + c^2 - p^2}{2L_2c}\right) - \alpha$$ ----------Eq. 6.4.6

Eq 6.4.6 is used to calculate theta2.

$$\beta = \arccos\left(\frac{L_2^2 + p^2 - c^2}{2L_2p}\right) - \frac{\pi}{2}$$ ----------Eq. 6.4.7

$$\gamma = \arccos\left(\frac{L_3^2 + p^2 - L_4^2}{2L_3p}\right)$$ ----------Eq. 6.4.8

$$\theta_3 = \pi - \beta - \gamma$$ ----------Eq. 6.4.9

Eq 6.4.9 is used to calculate theta2.

Finally, make sure that all the angles range between 0-180 degrees to avoid errors.

These angle values are inputted to the Arduino to move the robotic arm.

6.5 Arduino Code

After getting the position of our target to pick up and calculating the angles we send the data to our robot arm to get it to move. In order for the angles to actually change the movement of our arm's servos we used arduino code to control them.

In our code we have our setup which declares and assigns our servos to pins directly connected to the Arduino Mega. From there we then move to our loop which we used to write to our servos to move as desired. We wrote to 5 out of 6 servos using degrees and the servo controlling the gripper in microseconds. We chose microseconds for the gripper because using degrees the gripper would not function properly. It would only stay open. Going through Arduino forums microseconds was the solution to getting the gripper to close as commanded. We used a set number of microseconds to control the gripper after testing it with the size of the balls we used.

Once a servo is written to, we added a 1 second delay in between servo writing so that the arm does not move sporadically as it did once during testing. We started our loop with a chunk that would set our arm to our zero position. We kept this mainly to ensure that our arm is starting and moving properly according to our calculations. It was also a helpful reference for calculating our inverse kinematics.

The next portion we had was that of the desired angles for our servos. Where we inputted our angles to the servos manually but did attempt to create a serial connection between the Arduino and Raspberry Pi. Then closely followed by the servos being instructed to then move the ball to our basket location. Finally afterwards we exited our loop and let our arm sleep. We did this to conserve some power and because we could not keep it running with it going to the same place. With serial connection and a stable power source we could keep it running and waiting until the camera sees the next target and continue to instruct the arm to move as expected.

Responsible person: Karen

6.5.1 Processing narrative for Arduino Code

Code the robotics arm to go to home position, pick up the red ball closer to the robot, and drop the ball in the box in a loop.

6.5.2 Arduino Code interface description

Input is the angles calculated from the IK and the output is the moving Arduino.

6.5.3 Arduino Code processing details

Explained in 6.5.

6.6 Neural Network

Responsible person: Priscila

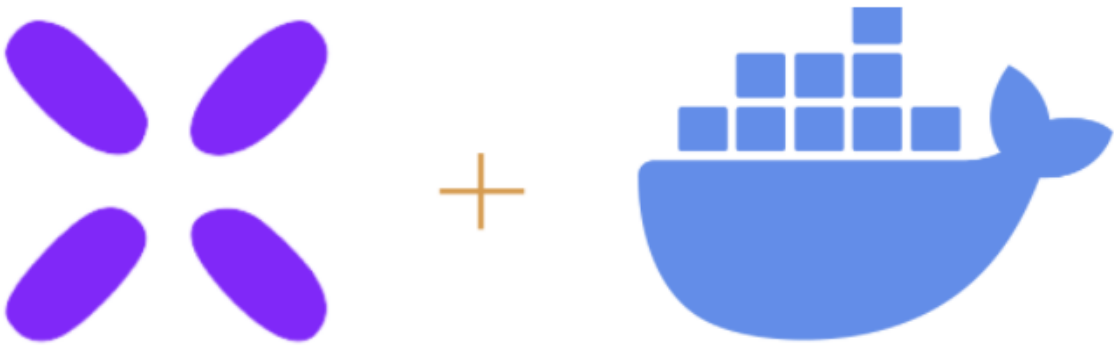


Figure: 6.6.1: Xailient connects with the Docker to run the the AI Model

6.6.1 Processing narrative for Neural Network

We used machine learning to find ripe tomatoes as well as red ping pong balls that signify ripe tomatoes.

6.6.2 Neural Network interface

Using our annotations of each red ping ball found in each picture, it was inputted into an excel sheet consisting of the image name, top left x coordinate, bottom right x coordinate, top left y coordinate and bottom right y coordinate. We then used the Xalient console to train our model. After the Xailient console trains the model, it outputs a python wheel with our trained AI Model. We open the python wheel using virtualization and a docker.

6.6.3 Neural Network processing details

After having our annotations in a .csv file, we input it into the Xalient console to train the data.

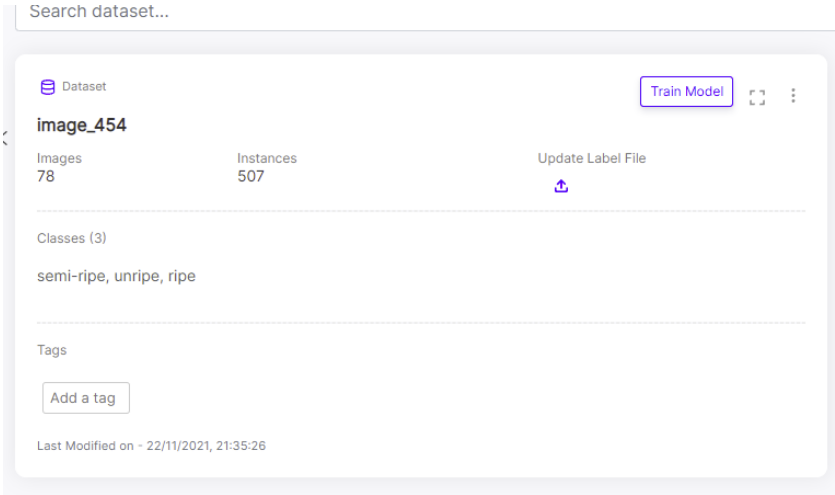


Figure 6.6.3: dataset of annotations

After it trains the data for about an hour, we then have our model.

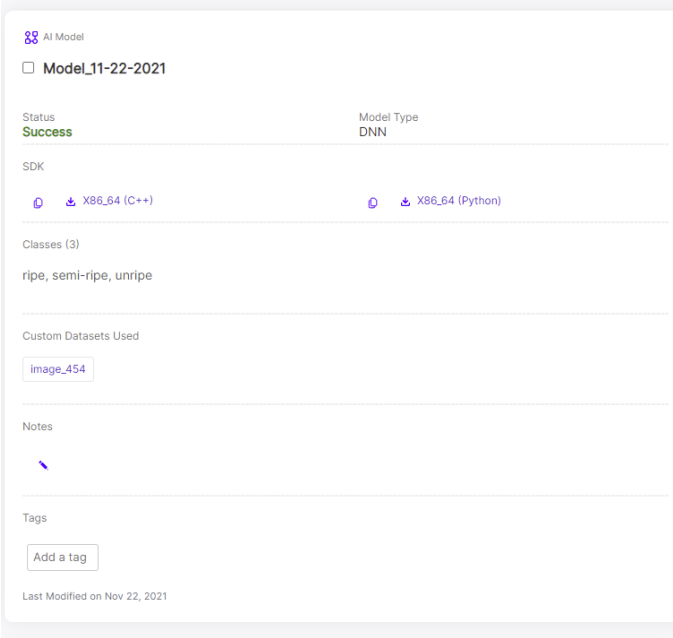
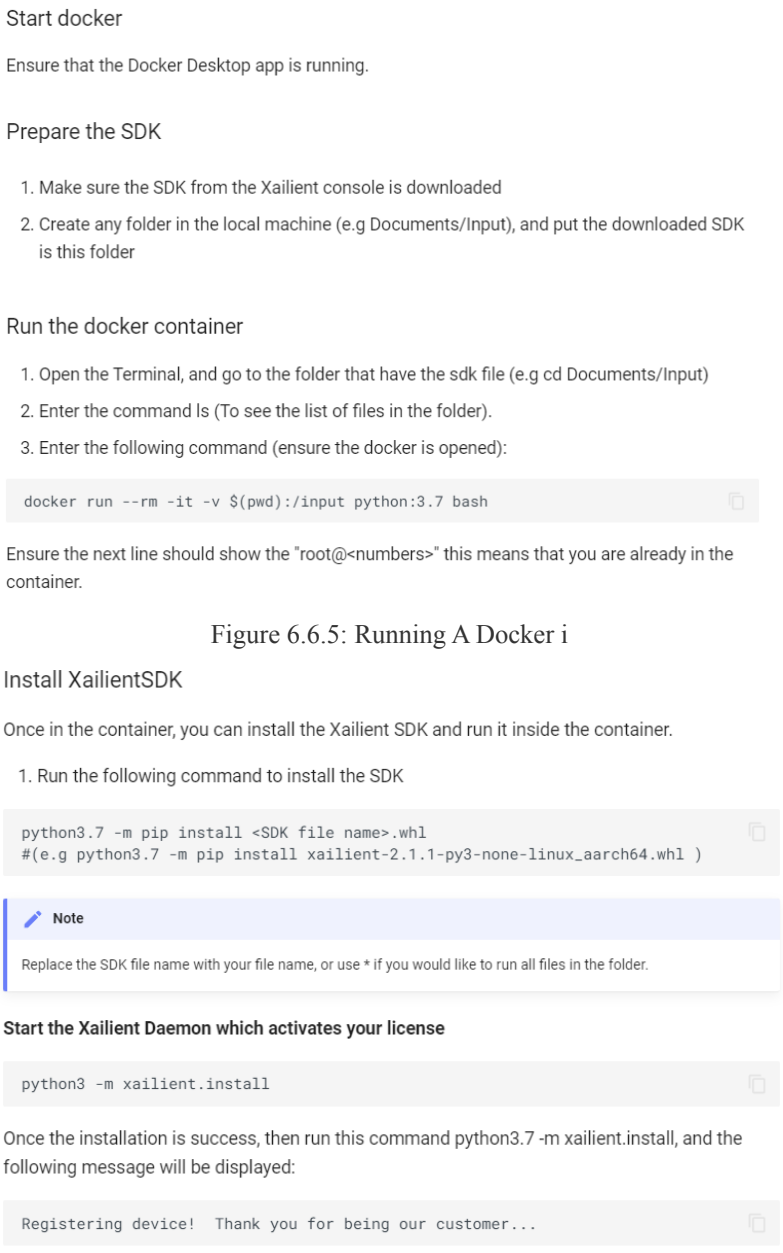


Figure 6.6.4: trained model

In order to open it we use a docker by enabling virtualization in our BIOS settings. The following pictures show how Docker was set up.



Start docker

Ensure that the Docker Desktop app is running.

Prepare the SDK

1. Make sure the SDK from the Xailient console is downloaded
2. Create any folder in the local machine (e.g Documents/Input), and put the downloaded SDK is this folder

Run the docker container

1. Open the Terminal, and go to the folder that have the sdk file (e.g cd Documents/Input)
2. Enter the command ls (To see the list of files in the folder).
3. Enter the following command (ensure the docker is opened):

```
docker run --rm -it -v $(pwd):/input python:3.7 bash
```

Ensure the next line should show the "root@<numbers>" this means that you are already in the container.

Figure 6.6.5: Running A Docker i

Install XailientSDK

Once in the container, you can install the Xailient SDK and run it inside the container.

1. Run the following command to install the SDK

```
python3.7 -m pip install <SDK file name>.whl
#(e.g python3.7 -m pip install xailient-2.1.1-py3-none-linux_aarch64.whl )
```

Note

Replace the SDK file name with your file name, or use * if you would like to run all files in the folder.

Start the Xailient Daemon which activates your license

```
python3 -m xailient.install
```

Once the installation is success, then run this command python3.7 -m xailient.install, and the following message will be displayed:

```
Registering device!  Thank you for being our customer...
```

Figure 6.6.6: Running A Docker ii

We could then connect the camera to the python wheel AI model by using the following code from Xailient.

```python
import cv2 as cv
from xailient import roi_bbox
import numpy as np

detectum = roi_bbox.ROIBBoxModel()
cap = cv.VideoCapture("sample.mp4")

while True:
    _, next_frame = cap.read() # Reads the next video frame into memory

    bboxes = detectum.process_image(next_frame)

    # Loop through list (if empty this will be skipped) and overlay green bboxes
    # Format of bboxes is: xmin, ymin (top left), xmax, ymax (bottom right)
    for bbox in bboxes:
        pt1 = (bbox.xmin, bbox.ymin)
        pt2 = (bbox.xmax, bbox.ymax)
        cv.rectangle(next_frame, pt1, pt2, (0, 255, 0))

    cv.imshow("Rendered Video", next_frame)

    key = cv.waitKey(50)
    if key == 27: # Hit ESC key to stop
        break
```

Figure 6.6.7: Camera to AI Model Connection i

```python
cap.release()
cv.destroyAllWindows()
```

Figure 6.6.8: Camera to AI Model Connection ii

6.7 Raspberry Pi

Responsible person: Michael

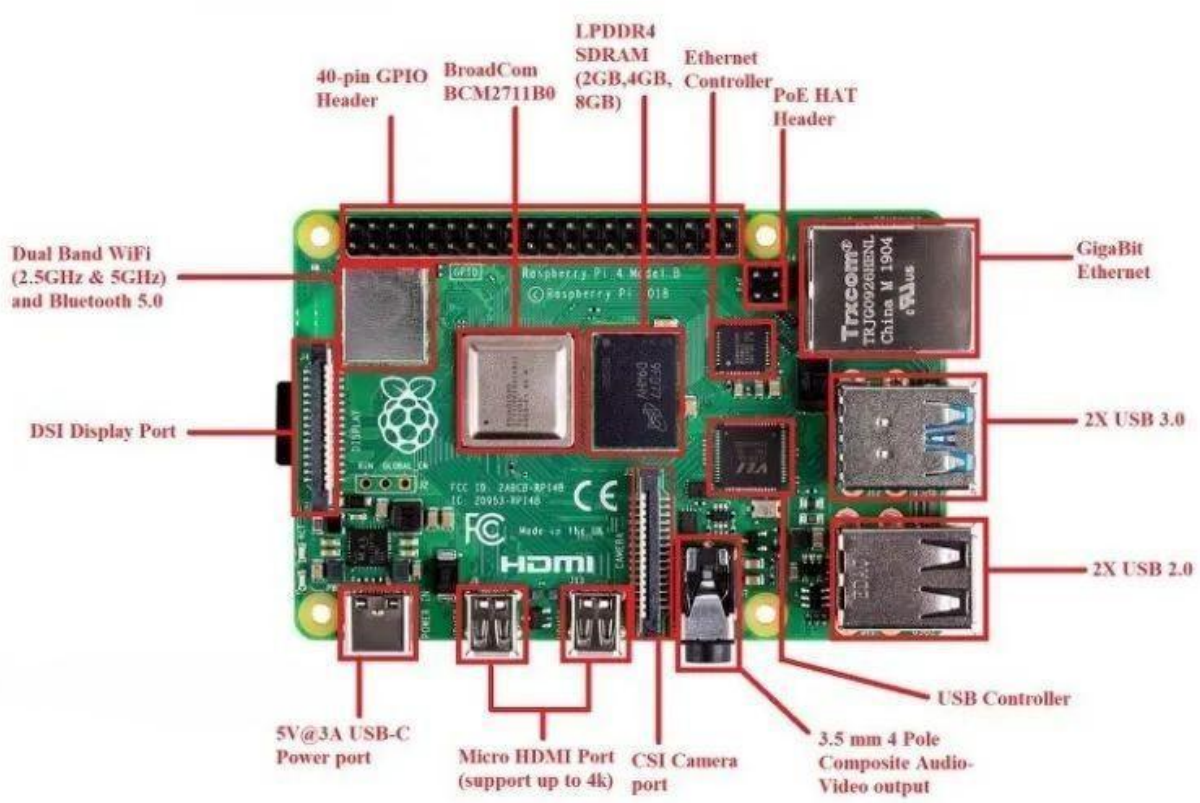


Figure 6.7.1 Raspberry Pi board

6.7.1 Processing narrative for Raspberry Pi

The above image, Figure 6.7.1, displays the Raspberry Pi 4 schematic. The Raspberry Pi contains the proper RAM and software necessary for the main programs to run virtually, only needing desktop integration to be able to work from a remote location.

6.7.2 Raspberry Pi interface description

The input interface was a software called Debian OS, which was the main software used when working with the RP. The output interface was with the Librealsense software, which had to be installed manually and compiled in order for the CV and IK programs to function and work properly. The interface also included a VNC Viewer in order to display output from the RP onto the computer. The RP would be connected to the computer via an Ethernet cable and connected to the AM and LC via USB 2.0 connection.

6.7.3 Raspberry Pi processing details

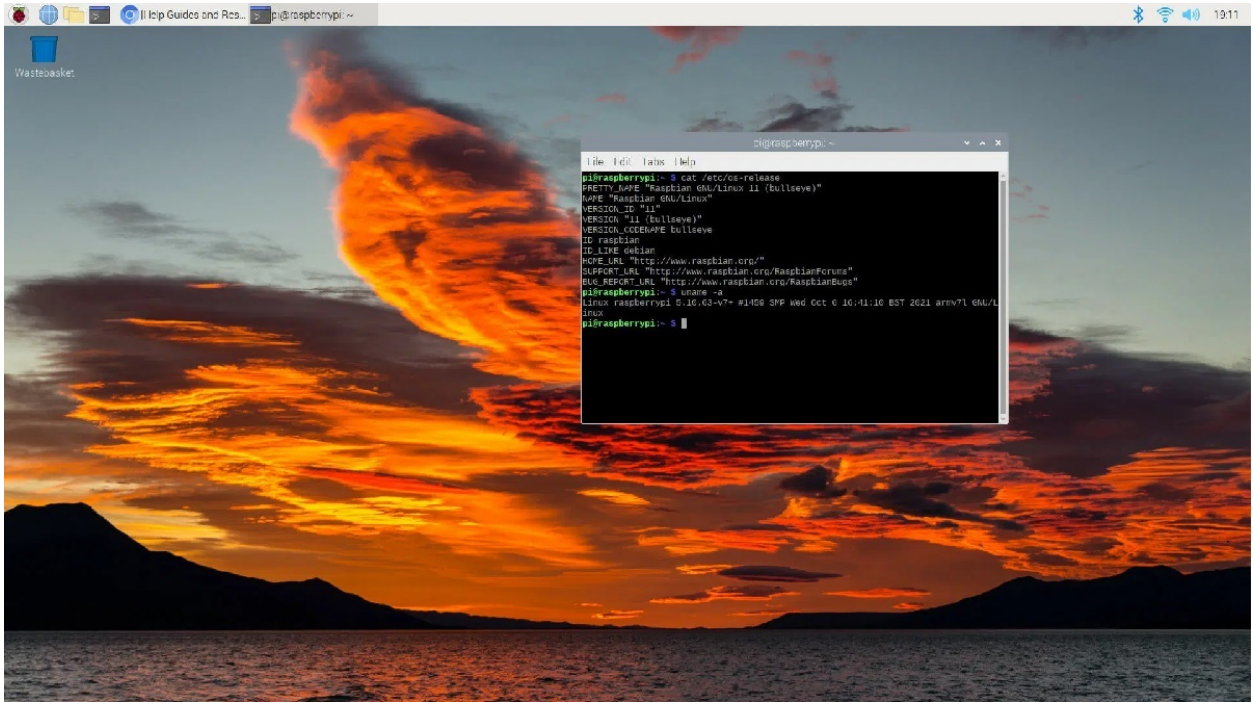


Figure 6.7.2: Raspberry Pi 4 with Debian OS

The figure above, figure 6.7.2, displays the Debian OS after having been SSH into via an Ethernet cable, outputting the terminal ready for input. The development and set up of libraries was done on this hardware, which contains the IK and CV programs.

6.8 Serial Communications

Responsible person: Michael

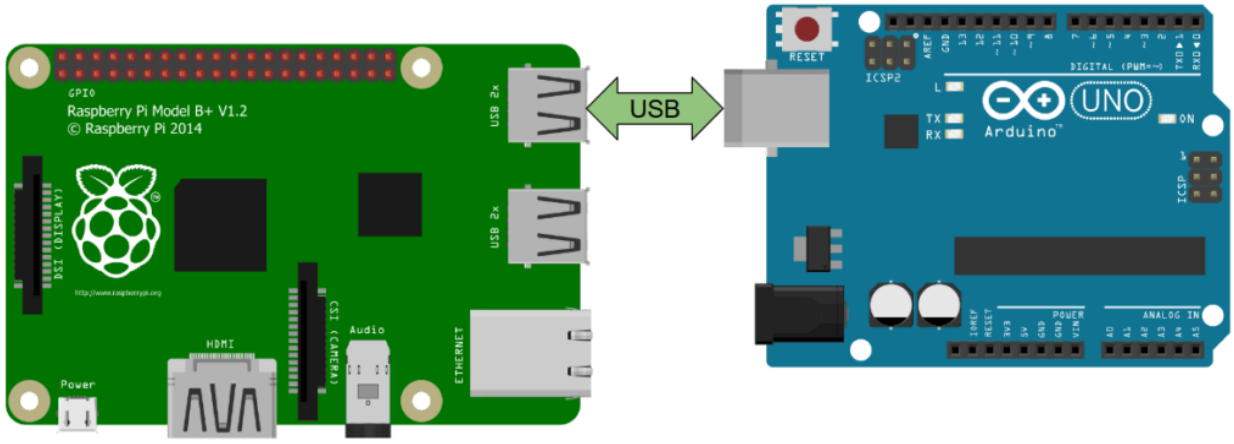


Figure 6.8.1: An example of Serial Communication between Arduino Uno and RP

The above figure, figure 6.8.1, displays an example of communication between an RP and an Arduino Uno, however in this design project, an Arduino Mega was used over the Uno.

6.8.1 Processing narrative for Serial Connection

The serial communication between the AM and the RP was done via USB connection.

6.8.2 Serial Connection interface description

The input would be values from the CK and IV programs, and the RP would output the angle data values to the AM for the motors to move accordingly.

6.8.3 Serial Connection processing details

To send the values to the RP, the RP needed to register that there was a connection between the two microcontrollers and then the AM needed to take the inputted values and move the motors to the required positions.

7 * Technical Problem Solving

7.1 * Serial Connection Problem

One of the main challenges of this project included using Serial Connection in order for there to be communication between the two microcontrollers. The communication was successful; however the primary problem was the conversion of data values from the RP to the AM. Michael O'Dea worked on this problem in order for data values to be sent between them.

7.2 * Solving the Serial Connection Problem

The main difficulty that this problem had was sending the proper data values from the RP to the AM. The AM can send data output easily to the RP, but it is much more difficult when sending data values from the RP to the AM. The AM has to be able to take the integers that are sent, and then proceed to move each motor accordingly. This is where the issue is. The AM doesn't know when to stop reading the data, so it first moves the robotic arm to the zero position, but then fails in moving each motor according to the values sent from the RP. The AM can read in bits and set them, but with a single stream of numbers, it lacks the capability of knowing when to stop and to set those numbers to integers instead of a string. There were multiple versions of code that were tried and attempted, but they each failed at getting the robotic arm to move. In the end, there was a serial connection established successfully with the RP and the AM, however the problem was having the numbers register on the AM from the RP.

In this process and system, working with serial connection, and with the two microcontrollers, was new as well as working with Debian OS in setting up and making necessary libraries. Working with the RP was new and experimental since it was never previously worked on by members of this team. Michael O'Dea was the main person working with the RP and solving the issue with serial connection.

The only way around this problem, which ended up being the one implemented in this design, was to connect the AM to the desktop by USB and manually inputting the angles given by the IK program after the CV program. This essentially removed the remote desktop virtualization and autonomization that was part of the original design.

7.3 * Neural Network Problem

The main problem when working on the Neural Network was training the data. Since we were using coordinates at first and not RBG it was hard to find anything on how to use my annotations to train our AI Model. After that, the final problem was getting the python wheel to open so we could test out the AI Model.

7.4 * Solving the Neural Network Problem

When I was trying to train my model, I used CoLab as they had an example. I was able to run everything without any errors however when it came to training 80 images with over 300 annotations, CoLab would continue to run fors hours and sometimes it would auto-kill itself. So then I started experimenting with the number of annotations and images. I decided to drop the number of images from

120 to 80. I kept the same number of annotations and the runtime would not pass 3 hours and stop the code altogether. So then I decided to change the many annotations, around 300, to about 100. Leaving at least 1 per image and some had two if they were really good annotations. Still not passing 3 hours. So then I went to 80 images, 80 annotations. One annotation per image. This set actually made it to 3 hours, but it completely lagged my PC, not computer. My PC has a good cpu and gpu so it's hard to get it to lag but this code did that. My screen was moving at 1 frame per 30 sec to 1 min. While it was lagging my audio driver crashed and I couldn't even move my mouse because it was lagging so bad. So after that happened, I let my PC rest for the remainder of the day because I didn't want to cause any damage. I remembered that when I ran their test code, it consisted of only 15 images and 15 annotations so I thought if I did 20 images, 20 annotations it would work. With their sample code, it took less than 2 min to complete the dataset. So I thought it should take the same amount of time, give or take a few minutes.

However, my guess was completely wrong, it actually passed the 3 hour mark that almost all the previous ones didn't pass. I thought it would take maybe 4-6 hours to complete, but this ran for almost 12 hours, going into 4am. And google collab makes it so that if it's running for a long time, it likes to periodically ask if I'm still there. If I don't answer within a set time period it stops my code. So when it was running my code, I periodically set my alarm for 30 min intervals so that I can wake up and move my mouse so that it knows I'm still there. I'm not entirely sure if the code just stopped like the rest or if google colab stopped it because of "inactivity". While I did make progress, it still took half a day to run. So I started thinking why it would not run as fast as the same code they provided. The only difference is that they used a .csv file while I used a .json file instead. My guess is that since it's a .csv file, it's a lot faster to get the data into a dataset for our json file, it has to iterate through each annotation bracket and add it to the dataset. The columns and rows make it easy to find our data. That's why I think it runs so much faster but I still don't understand why it's running for hours. I would like to change it to .csv to test that theory but for some reason when I entered my annotations as a .csv with the exact same format it didn't take it. I didn't want to go any lower than 15 because from there it's taking an 80-20 ratio to train and test. I was able to find a software that would train my data for me, Xailient Console. Now I had a trained AI Model in a python wheel format.

The other complication I had was opening the python wheel. When trying to open the python wheel, it keep giving me the error that the python wheel was not supported as shown in the following figure

```
xailient-2.1.1-py3-none-linux_armv7l.whl
root@fb21363fd0cd:/tomato# python3.10 -m pip install xailient-2.1.1-py3-none-linux_armv7l.whl
ERROR: xailient-2.1.1-py3-none-linux_armv7l.whl is not a supported wheel on this platform.
WARNING: You are using pip version 21.2.4; however, version 21.3.1 is available.
You should consider upgrading via the '/usr/local/bin/python3.10 -m pip install --upgrade pip' co
mmand.
```

Figure 7.4.1 Error Message

It turned out I had to open the x86_64 platform wheel instead of the linux_arm71 that is meant for the raspberry pi. I was able to open up the linux_arm71 by changing the "linux_arm71" to "any" in the file name. The issue with that is when I installed xalient with the docker, it needs to know what platform to use so it doesn't install it and thus we have a file that cannot be opened. I tried working around it many

times by trying different python versions to see if it was the version, I had that was incompatible. But nonetheless, neither version worked even if I ended up opening a virtual environment and trying to run the docker code there. So that's when I resorted to trying different platform files and it worked.

7.5 * The Inverse Kinematics Problem

It had n variables and n-1 equations to solve it.

7.6 * Solving the Inverse Kinematics Problem

Refer to 6.4.

7.7 * The Camera Transform Problem

When the color frame and the depth frame wasn't aligned, the XYZ coordinated gave incorrect measurements. After solving it, the XYZ coordinates weren't corresponding to the robot frame but the camera frame.

7.8 * Solving the Camera Transform Problem

The program uses functions of RS2 to align the color and depth frames to get the correct measurements. Refer to 6.3.3.3 to how it was solved on the transformation.

7.9 * The Object Recognition Problem

The Hough Transformation would detect multiple circles on a single ball.

7.10 * Solving the Object Recognition Problem

Refer to section 6.3.3.1.

8 User Interface Design

Responsible: Pamodya

8.1 Application Control

Once the run button is pressed, after 30 seconds, this window pops up and show the circles detected on the real time video shown in Figure 8.1. If no red balls are detected it says 'no circles found' in Figure 8.2. If red balls are detected, it output the XYZ coordinates according to the robot frame that is closer to the robot and the angles for servos 3-6 for that position as seen in Figure 8.2. The real time video is 30fps so it checks for circles in that rate which sometimes skip the frames due to the processing speed of the program.

8.2 User Interface Screens

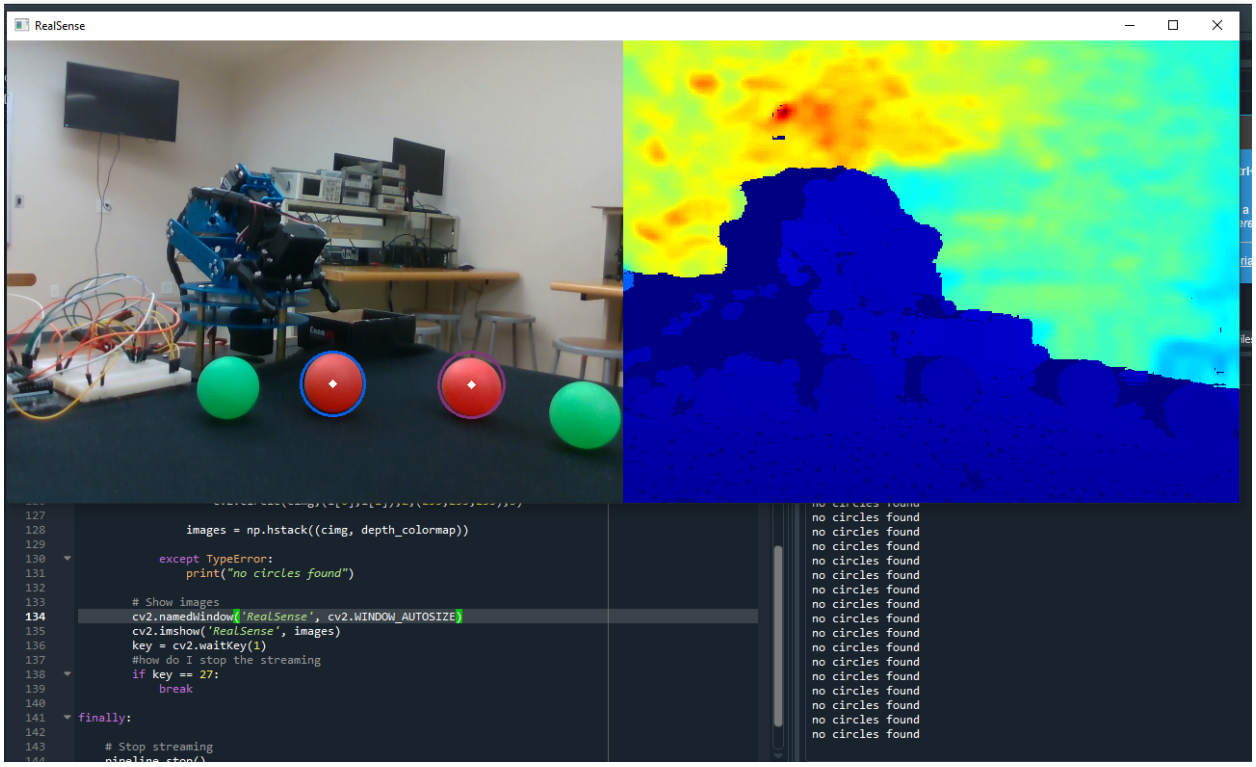


Figure 8.1 User Interface for the program

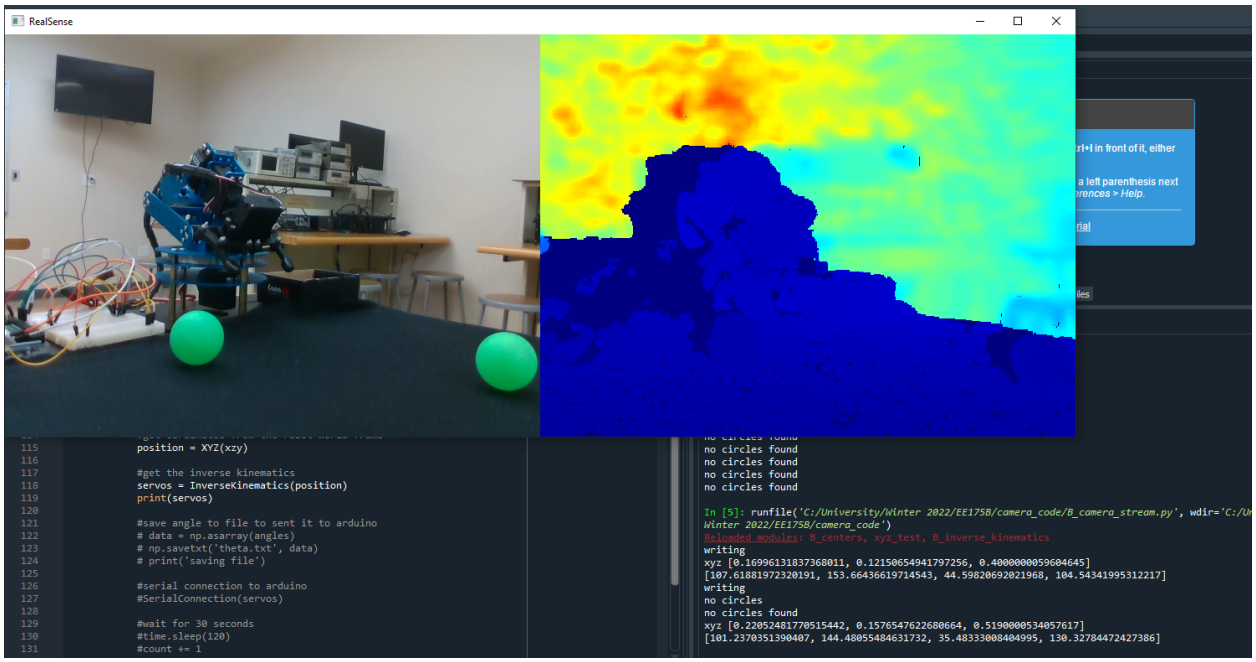


Figure 8.2 User Interface for the program with angles

9 * Test Plan

9.1 * Test Design

The tests below will measure the responses and output that is given from our hardware and our software. Our hardware tests will be focused on the robot performing the requested action, while our software tests focus on the computer vision algorithms and training the program itself to recognize the object for the robot to pick up.

Tests and expected responses

9.1.1: Keyboard Input, Case 1 (Pamodya Peiris)

1. The objective of this experiment is to properly test the computer vision detection program with keyboard input in order for the main program to start.
2. Our setup will consist of the program open and ready to run based on user input.
3. The program should be open and ready for operation.
4. We should expect the program to run based on the input, and if required, to stop if the user wants the program to.

9.1.2: Raspberry Pi, Case 1 (Michael O'Dea)

1. The objective of this experiment is to properly test the working raspberry pi within our hardware implementation. This will measure the appropriate response by having the raspberry pi acknowledge that there is a program within it sent from the computer.
2. The raspberry pi will have ethernet connection to the laptop to receive internet connection and be plugged into a stable power source to ensure it has enough power to ruin the given program.
3. The raspberry pi will signal that it has a program within it and is ready to run as it has been sent via bluetooth from the computer.
4. We should expect the raspberry pi to operate and function as intended, and this will make sure that the embedded computer is able to run with the program.

9.1.3: Raspberry Pi, Case 2 (Michael O'Dea)

1. The objective of this experiment is to properly test the raspberry pi microcontroller with connection to the computer via Bluetooth.
2. Our setup will consist of the raspberry pi powered by the battery and connected to the computer.
3. The raspberry pi will be connected to the computer.
4. We should expect the computer to display that it is the raspberry pi and that it is connected to them.

9.1.4: Battery, Case 1 (Karen Gonzalez, Priscila Huante Mendez)

1. The objective of this experiment is to properly test the batteries within our hardware implementation. The test will measure the appropriate response by having the battery providing power with proper connection to the servos.
2. This will be set up on a breadboard with the battery powered on and connected to the main microcontroller, which is the Arduino mega.
3. The battery will be tested by ensuring that it functions as intended and supplying enough power to the servos without overloading.
4. We should expect the batteries to operate, and this will make sure that the arm can have full range of motion with servos properly running connected to the batteries.

9.1.5: Robot Arm, Case 1 (Karen Gonzalez)

1. The objective of this experiment is to properly test the load that the robot arm is expected to make. The test will measure the appropriate response by having the battery powered on and wired up properly connected to the various hardware components.
2. This will be set up with the robot arm lifting a 0.5lb weight, which is the payload allowance.
3. The robot arm will be tested by lifting the weight at different configurations that are possible for the robot arm.
4. We should expect the robot arm to be able to lift the weight that is required with no errors.

9.1.6: RGB-D Camera, Case 1 (Pamodya Peiris)

1. The objective of this experiment is to properly test the working RGB-D Camera within our hardware implementation. The camera will measure the appropriate response by having the camera record the view in front of the robot arm and be able to capture video.
2. This will be set up with the camera powered on, placed in front of the robot arm.
3. The procedure for collecting this data is to accurately determine and confirm that the camera is working properly and as intended.
4. We should expect the camera to accurately capture and display what is being recorded liv

9.1.7: Object Recognition Program, Case 1 (Pamodya Peiris)

1. The objective of this experiment is to properly test the Object Recognition program to identify the red balls. The OpenCV function HoughTransform will change the different parameters to find the best values.
2. This will be set up with the camera and red and green balls on the board.

3. The procedure for collecting this data is to use the camera and run the python program while changing the number of red balls available on the board.
4. We should expect the program to identify that the function returns only one of lesser circles for each ball.

9.1.8: Object Recognition Program, Case 2 (Pamodya Peiris)

1. The objective of this experiment is to properly test the Object Recognition program to identify the red balls. The OpenCV function HoughTransform will change the different parameters to find the best values.
2. This will be set up with the camera and red and green balls on the board.
3. The procedure for collecting this data is to use the camera and run the python program while changing the number of red balls available on the board.
4. We should expect the program to identify that the function returns only one of lesser circles for each ball.

9.1.9: Object Recognition Program, Case 3 (Pamodya Peiris)

1. The objective of this experiment is to properly test the Object Recognition program to identify the red balls. The OpenCV function HoughTransform will change the different parameters to find the best values.
2. This will be set up with the camera and red and green balls on the board.
3. The procedure for collecting this data is to use the camera and run the python program while changing the number of red balls available on the board.
4. We should expect the program to identify that the function returns only one of lesser circles for each ball.

9.1.10: Object Recognition Program, Case 4 (Pamodya Peiris)

1. The objective of this experiment is to properly test the Object Recognition program to identify the red balls. The OpenCV function HoughTransform will change the different parameters to find the best values.
2. This will be set up with the camera and red and green balls on the board.
3. The procedure for collecting this data is to use the camera and run the python program while changing the number of red balls available on the board.
4. We should expect the program to identify that the function returns only one of lesser circles for each ball.

9.1.11: Inverse Kinematics, Case 1 (Pamodya Peiris, Priscila Huante Mendez and Karen Gonzalez)

1. The objective of this experiment is to test our inverse kinematics calculations using the coordinates provided by the camera.
2. We would have the camera connected to the Raspberry Pi to run the python and camera program. As well as the arm set up with its proper connections to the batteries and Arduino to test the calculated angles.
3. The camera's program will process the position of the ball/tomato and send the data to the python program that will then calculate the required angles.
4. We should expect the calculated angles provided to allow the arm to grab the ball/tomato with little to no error.

9.1.12: Inverse Kinematics, Case 2 (Pamodya Peiris, Priscila Huante Mendez and Karen Gonzalez)

1. The objective of this experiment is to test our inverse kinematics calculations using the coordinates provided by the camera.
2. We would have the camera connected to the Raspberry Pi to run the python and camera program. As well as the arm set up with its proper connections to the batteries and Arduino to test the calculated angles.
3. The camera's program will process the position of the ball/tomato and send the data to the python program that will then calculate the required angles.
4. We should expect the calculated angles provided to allow the arm to grab the ball/tomato with little to no error.

9.1.13: Inverse Kinematics, Case 3 (Pamodya Peiris, Priscila Huante Mendez and Karen Gonzalez)

1. The objective of this experiment is to test our inverse kinematics calculations using the coordinates provided by the camera.
2. We would have the camera connected to the Raspberry Pi to run the python and camera program. As well as the arm set up with its proper connections to the batteries and Arduino to test the calculated angles.
3. The camera's program will process the position of the ball/tomato and send the data to the python program that will then calculate the required angles.
4. We should expect the calculated angles provided to allow the arm to grab the ball/tomato with little to no error.

9.1.14: Neural Network, Case 1 (Priscila Huante Mendez)

1. The objective of this experiment is to properly test the computer vision detection program. The test will measure the appropriate response by having the program detect whether a plastic tomato is ripe or not.
2. The set up will be with the program running and accessing images of tomatoes and determining which are ripe.

3. The program should be able to distinguish which tomatoes are ripe or not.
4. We should expect the program to output how many tomatoes it has determined to be ripe or not based on each photo within the program's database.

9.1.15: Neural Network, Case 2 (Priscila Huante Mendez)

1. The objective for this experiment is to properly test our trained AI Model. The test will see if it can detect a red ball.
2. The setup is that it will be given a picture of a red ball that wasn't given to train the model.
3. The program will be able to detect a red ball as "ripe".
4. We should expect to see the output that the red ball is "ripe.

9.1.16: Neural Network, Case 3 (Priscila Huante Mendez)

1. The objective for this experiment is to properly test our trained AI Model. The test will see if it can detect a red ball.
2. The setup is that it will be given a picture of a red ball that wasn't given to train the model.
3. The program will be able to detect a red ball as "ripe".
4. We should expect to see the output that the red ball is "ripe.

9.2 * Bug Tracking

We will track bugs that we found within a document that we will use to measure and record our data as well as comment on any unnatural occurrences that happen from testing as well as comment on how the test goes overall. All defects that are found will be logged as record. Depending on whether the test was performed either as a hardware or software test, Karen Gonzalez/Michael O'Dea or Priscila Huante/Pamodya Peiris will investigate.

9.3 * Quality Control

To make sure that all cases were reviewed, the test cases will be run multiple times, and each run will be commented on to ensure that the test was run successfully. Depending on if the test ran successfully or not, we would mark it as either passed or failed. When a test case is failed, the date and time and name of who ran the test will be noted down, and when a failed case is marked as fixed, the date and time and name of who fixed it will also be recorded.

9.4 * Identification of critical components

Describe the components that are critical and demand particular attention during testing are identified.

Name	Why?
Arduino Mega 2560	Send angles from CV program or Raspberry Pi 4 to servos.
LiPo Battery (2x)	Too be able to power the servos.
Raspberry Pi 4	Send angles from the CV program on the interface to the arm.
Intel RealSense D435i	Identify the red balls and get their placements

9.5 * Items Not Tested by the Experiments

The neural networking with the camera was not tested, since it was already established that the camera would be able to send data to a computer via a USB cable as intended.

10 * Test Report

Carry out the experiments designed in the section above to test your modules and prototype and present the results. Present the results of the experiments and provide an analysis of the experimental test data.

State clearly who is responsible for which test case

Each test must be run multiple times. When you find deviation from the expected results, you must take action to debug your design, then test it again. Report each test iteration below

10.1 * Keyboard Input, Case 1

(Pamodya Peiris)

Test 1: run the program with the run button on Spyder

1. When there aren't any red balls, the program would exit unexpectedly
2. The expected result was to show the color and depth images in real time without any errors
3. The program is supposed to take care of boundary cases
4. Update the Object Recognition Program to smoothly run the program

Test 2: Close the program with the 'ESC' key

1. Test results show that the program close when 'ESC' is pressed
2. The expected result is to close the program for 'ESC'
3. The test had 100% accuracy
4. No corrective actions needed to be taken

10.2 * Raspberry Pi, Case 1

(Michael O'Dea)

1. The test resulted in the RP being able run to the CV program without any errors.
2. The expected result was output of the IRC from the CV program.
3. The microcontroller successfully displayed what the IRC was viewing.
4. No corrective actions were needed to be taken.

10.3 * Raspberry Pi, Case 2

(Michael O'Dea)

1. The computer was able to connect successfully to the RP after SSH and also making sure that the computers internet connection, with the WiFi provided by UCR-SECURE, was able to be shared to the RP via Ethernet.

2. This was the expected result after numerous attempts at other methods.
3. The microcontroller was successful in being able to allow the computer administrative access on it in order to work and program with it.
4. No corrective actions were needed.

10.4 * Raspberry Pi, Case 3

(Michael O'Dea)

1. The AM was successful in establishing a serial connection with the RP
2. This was the expected result after numerous attempts.
3. The RP was successful in being able to receive and send data from the AM
4. No corrective actions were needed.

10.5 * Battery, Case 1

(Karen Gonzalez and Priscila Huante Mendez)

Test 1: Using a power supply (HPPS) tests the amount of power necessary for the whole arm to be powered.

1. The power supply could power all the servos with home adjustment to voltage to ensure no overloading.
2. Expected to be within a range of 5-7 volts as that is the range for all the servos individually.
3. Around 6 volts of voltage was sufficient for powering all servos.
4. Test using the battery purchased that will be used in demonstration.

Test 2: Test using battery purchased.

1. A single battery though set to 6V did not send enough current to power all the servos at once.
2. Expected to only need one battery set to 6V to power all servos based on the power supply tested.
3. One 6V battery did not supply enough power to all 6 servos, but increasing the voltage overloaded the servos.
4. Check to see how many servos could be powered with a single battery

Test 3: Tested how many servos can work properly using the power from the single battery.

1. Found that the battery could supply enough power to 3 servos.
2. Expected to power all servos but could only send enough current to 3.
3. If one battery can send power to half of the servos we could add another to power the rest.
4. Used two separate 6V batteries to supply power to 2 sets of 3 servos.

Test 4: Test with two batteries.

1. Two separate batteries were needed to power all the servos to not overload the amount of amps.
2. Expected to power all servos with two batteries and was confirmed with a test.
3. Using two batteries provided the servos with necessary power needed to function properly.
4. No corrective actions needed.

10.6 * Robot Arm, Case 1

(Karen Gonzalez)

1. Arm was able to hold the weight of the possible ball options.
2. Arm was expected to hold the weight with possible staggering if it were not powered sufficiently.
3. Some staggering with heavier weight, but the arm was able to pick up the intended object.
4. Use the balls we set on as they are light and should not cause problems.

10.7 * RGB-D Camera, Case 1

(Pamodya Peiris)

Test 1: output the camera stream window

1. The test shows that both color and depth images are shown without any errors
2. Expected result is to output the color and depth images
3. This test has an 100% accuracy
4. No Corrective actions needed

Test 2: check the xyz coordinates with manual measurements

1. Test shows that the xyz taken from the program doesn't correlate with the actual measurements
2. Expected result is that xyz to be same as the measured values
3. only depth was accurate
4. Corrective actions was to align the color and the depth frames

Test 3: check the xyz coordinates with manual measurements after alignment

1. Test shows that the xyz taken from the program correlates with the actual measurements
2. Expected result is that xyz to be same as the measured values
3. 100% accuracy
4. no corrective actions taken

Test 4: Output Hough Circles in the color image in real time

1. Test results show circles on the color image in the 30 fps stream
2. Shows the circles detected in every frame
3. ~80% accuracy as the program takes time to process the circles more than the frame rate
4. None taken but output that no circles are found if the program takes time to process it

10.8 * Object Recognition Program, Case 1

(Pamodya Peiris)

Test 1: min radius

1. Test numbers ranging from 10mm-20mm till the circles are detected when the ball is furthest from the camera
2. At 16mm the circles are detected
3. At 16mm, the accuracy for the detected circles are ~80%
4. Change the value for minRadius on HoughTransform from 10mm-20mm to find the best value

Test 2: max radius

1. Test numbers ranging from 40mm-60mm till the circles are detected when the ball is closest from the camera
2. At 50mm the circles are detected
3. At 50mm, the accuracy for the detected circles are ~80%
4. Change the value for maxRadius on HoughTransform from 40mm-60mm to find the best value

10.9 * Object Recognition Program, Case 2

(Pamodya Peiris)

Test 1: change param2 in a range of 0.8

1. Test the HoughTransform with param2 = 0.8 get multiple circles
2. Detect only one circle
3. Multiple circles
4. increase the number of param2

Test 2: change param2 in a range of 0.85

1. Test the HoughTransform with param2 = 0.85 get multiple circles
2. Detect only one circle
3. Multiple circles
4. increase the number of param2

Test 3: change param2 in a range of 0.9

1. Test the HoughTransform with param2 = 0.9 get multiple circles

2. Detect only one circle
3. Multiple circles
4. decrease the number of param2

Test 4: change param2 in a range of 0.86

1. Test the HoughTransform with param2 = 0.86 get one circle
2. Detect only one circle
3. One circle accuracy ~ 80%
4. increase the number of param2

Test 5: change param2 in a range of 0.87

1. Test the HoughTransform with param2 = 0.87 get one circles
2. Detect only one circle
3. One circle accuracy ~ 80%
4. increase the number of param2

Test 6: change param2 in a range of 0.88

1. Test the HoughTransform with param2 = 0.88 get one circle
2. Detect only one circle
3. One circle accuracy ~ 85%
4. increase the number of param2

Test 7: change param2 in a range of 0.89

1. Test the HoughTransform with param2 = 0.89 get multiple circles
2. Detect only one circle
3. Multiple circles
4. none

10.10* Object Recognition Program, Case 3

(Pamodya Peiris)

Test 1: change the difference between 2 circles to 1mm

1. Test the HoughTransform with difference between 2 circles to 1 mm to avoid multiple circles and get multiple circles
2. Detect only one circle
3. Multiple circles

4. increase the difference between the centers of the circles to be 2 mm

Test 2: change the difference between 2 circles to 2mm

1. Test the HoughTransform with difference between 2 circles to 2 mm to avoid multiple circles and get multiple circles
2. Detect only one circle
3. Multiple circles
4. increase the difference between the centers of the circles to be 3 mm

Test 3: change the difference between 2 circles to 3mm

1. Test the HoughTransform with difference between 2 circles to 3 mm to avoid multiple circles and get multiple circles
2. Detect only one circle
3. Multiple circles
4. increase the difference between the centers of the circles to be 4 mm

Test 4: change the difference between 2 circles to 4mm

1. Test the HoughTransform with difference between 2 circles to 4 mm to avoid multiple circles and get multiple circles
2. Detect only one circle
3. Multiple circles
4. increase the difference between the centers of the circles to be 5 mm

Test 5: change the difference between 2 circles to 5mm

1. Test the HoughTransform with difference between 2 circles to 5 mm to avoid multiple circles and get multiple circles
2. Detect only one circle
3. One circles
4. none

10.11 * Inverse Kinematics, Case 1

(Pamodya Peiris, Priscila Huante Mendez and Karen Gonzalez)

Test 1: z value on XYZ

1. Did manage to work on getting the position at certain points.
2. Would receive the point of the ball but not the exact center.

3. The z coordinate would have to be adjusted as well as small error when it comes to placement on the ball from the camera's point of view.
4. add 0.01 m to z

Test 2: error when x is negative according to camera frame

1. Managed to get a better read when it comes to the negative field of view.
2. When in the positive view it the center of the ball would seem off to the left with respect to the camera.
3. Calculations would have to be adjusted based on placement.
4. check for when the ball is in positive of x according to the camera frame

Test 3: error when x is positive according to camera frame

1. Managed to get a better read when it comes to the negative field of view.
2. When in the positive view it the center of the ball would seem off to the left with respect to the camera.
3. Calculations would have to be adjusted based on placement.
4. check for when the ball is in negative of x according to the camera frame

10.12* Inverse Kinematics, Case 2

(Pamodya Peiris, Priscila Huante Mendez and Karen Gonzalez)

Test 1: test error in y axis

1. The arm was able to move to the right position but could not grab the ball as intended.
2. Expected the arm to be able to grasp the center of the ball/tomato given the calculated angles.
3. At times the arm would hit the ball when moving to its position or would try to grasp the ball too low or too high.
4. add error to y

Test 2: test error in z axis

1. The arm was able to move to the right position but could not grab the ball as intended.
2. Expected the arm to be able to grasp the center of the ball/tomato given the calculated angles.
3. At times the arm would hit the ball when moving to its position or would try to grasp the ball too low or too high.
4. add error to z

10.13* Inverse Kinematics, Case 3

(Pamodya Peiris, Priscila Huante Mendez and Karen Gonzalez)

Test 1: Checked movement of each servo.

1. Servos moved within range of 0-180 except for servo controlling the gripper.
2. Expected a range of 0-180 for all servos.
3. Servos moved within the expected range with small error. The gripper did not move as intended using degrees.
4. Find another way of adjusting the gripper's opening and closing mechanism.

Test 2: Test another method of writing to servos.

1. Gripper Servo moves using microseconds.
2. Expected another way to write to the server but was unsure of which method.
3. Servo moved as intended with microseconds being adjusted to close the gripper to the size needed.
4. No corrective action needed.

10.14* Neural Network, Case 1

Test 1:

1. Gave a picture of a green tomato

2. The result it gave us was a “unripe tomato”

3. Gave the correct analysis

4. No corrective actions taken

Test 2:

1. Gave a picture of a green tomato

2. The result it gave us was a “unripe tomato”

3. Gave the correct analysis

4. No corrective actions taken

Test 3:

1. Gave a picture of a green tomato

2. The result it gave us was a “unripe tomato”

3. Gave the correct analysis

4. No corrective actions taken

Test 4:

1. Gave a picture of a green tomato

2. The result it gave us was a “semi-ripe tomato”

3. Gave the incorrect analysis

4. Inserted more semi-ripe tomatoes picture to use to re-train our model

Test 5:

1. Gave a picture of a green tomato

2. The result it gave us was a “unripe tomato”

3. Gave the correct analysis

4. No corrective actions taken

Test 6:

1. Gave a picture of a green tomato

2. The result it gave us was a “semi-ripe tomato”

3. Gave the incorrect analysis

4. Inserted more semi-ripe tomatoes picture to use to re-train our model

Test 7:

1. Gave a picture of a green tomato

2. The result it gave us was a “unripe tomato”

3. Gave the correct analysis

4. No corrective actions taken

10.15* Neural Network, Case 2

Test 1:

1. Gave a picture of 2 red and 3 green balls

2. said there was 2 red balls and 3 green balls

3. Gave the correct analysis

4. No corrective actions taken

Test 2:

1. Gave a picture of 3 red and 3 green balls

2. said there was 3 red balls and 3 green balls

3. Gave the correct analysis

4. No corrective actions taken

Test 3:

1. Gave a picture of 0 red and 3 green balls

2. said there was 0 red balls and 3 green balls

3. Gave the correct analysis

4. No corrective actions taken

Test 4:

1. Gave a picture of 1 red and 3 green balls

2. said there was 3 green balls

3. Could not detect red ball but detected the correct amount of green balls

4. there was only 20% of the red ball showing so it did not detect it. I added more instances where the ball is cut off but not the picture we are testing as to not skew the results

Test 5:

1. Gave the same picture of 1 red and 3 green balls

2.said there was 3 green balls

3. Could not detect red ball but detected the correct amount of green balls

4. I added more instances where the ball is cut off but not the picture we are testing as to not skew the results

Test 6:

1. Gave the same picture of 1 red and 3 green balls

2.said there was 3 green balls

3. Could not detect red ball but detected the correct amount of green balls

4. I added more instances where the ball is cut off but not the picture we are testing as to not skew the results

Test 7:

1. Gave the same picture of 1 red and 3 green balls

2.said there was 3 green balls

3. Could not detect red ball but detected the correct amount of green balls

4. I added more instances where the ball is cut off but not the picture we are testing as to not skew the results

Test 8:

1. Gave the same picture of 1 red and 3 green balls

2.said there was 1 red ball and 3 green balls

3. detected 1 red ball and 3 green balls correctly

4. No corrective action taken

Test 9:

1. Gave a picture of 3 red and 0 green balls

2. said there was 3 red balls and 0 green balls

3. Gave the correct analysis

4. No corrective actions taken

Test 10:

1. Gave a picture of 2 red and 1 green balls

2. said there was 2 red balls and 1 green balls

3. Gave the correct analysis

4. No corrective actions taken

Test 11:

1. Gave a picture of 0 red and 0 green balls

2. said there was 0 red balls and 0a green balls

3. Gave the correct analysis

4. No corrective actions taken

11 * Conclusion and Future Work

11.1 * Conclusion

Responsibilities:

Michael O'Dea:

- RP
- Serial Communication
- Power Management

Priscila Saray Huante Mendez

- Neural Network (AI Program)
- Power Management

Karen Gonzalez

- Servo Communication
- Arduino
- Power Management
- Assembly of Robotic Arm

Pamodya Peiris

- Object Recognition using OpenCV
- XYZ coordinates using RS2
- Inverse Kinematics

Michael O'Dea:

This design project was challenging in all aspects, whether from researching the many various robotic arms that are being developed to even powering the motors. This was the first time that I worked with the RP and AM in establishing serial communication, and even working on the RP itself. After installing Debian OS on the RP, I essentially had to build all the required libraries in order for the CV and IK programs to work successfully on it. Apart from the technical knowledge that I learned from working with that hardware and software, this project heavily improved my ability to research electrical hardware and determine what would be the best for implementation. I learned how to solve problems as an engineer, both by myself and with my team, in order for this project to succeed. It is a valuable skill to be able to be confident and trusting in a given engineering team, and I was fortunate enough to be included for this design project.

Priscila Saray Huante Mendez:

I learned so much while doing this project. Initially I didn't see it as a big deal because all we had to do was make an AI model and then make the robot grab a tomato. I underestimated how hard machine learning can be. I had taken Intro to AI and it was easy for me to incorporate the features into training our data however things did not go my way at all. It took me a whole quarter and ¼ to get it to work. Although we didn't end up using my neural network, I did learn a lot especially as I had to install a

virtualization through my bios. I also learned how to use docker for the first time. Not only that but I was able to incorporate what we learned in robotics to this project. I learned how easy it is for things to fail and not go as expected, especially when we had high hopes for the gripper but we had to move on with what we had. Overall, it was a very fun project, even with all the struggles we went through.

Karen Gonzalez:

This project has allowed me the opportunity to learn different concepts that I have yet to learn in depth. I learned about using pneumatics for soft gripping actuators and the research behind some. I also got the chance to assemble a robotic arm and then see how the servos move upon command from the microcontroller provided. Afterwards I learned how to connect servos to an Arduino Mega and how to program them that way. As well as learning how to power the whole arm efficiently.

Pamodya Peiris:

I learned the fundamentals of Robotics and Computer Vision from this project. Using the Intel RealSense D435i camera was a huge learning curve. I had to study the API and how to use it for our project. I haven't properly coded on python before this from scratch. I had to make use of the proper libraries and write clear code. After learning that the camera frame and robot frame were different, properly transforming them was a challenge. The most challenging part was inverse kinematics. As I couldn't cross check my work using the analytical method and matrices, I used the geometric version which took about a week to understand. Object Recognition was programmed as I took the EE146 course and I would add something from the lab each week. I couldn't figure out how to remove noise from the image. The program would have been better if I took multiple images to calculate the centers and get the average. Professionally, I learned that opening up to the team during the early stages of the project is really important to avoid miscommunication and a team should have roles assigned to make it run smoothly. I was able to improve my presentation skills and public speaking skills. Personally, I learned that encouraging each other during the project is important.

11.2 Future Work

Some of the plans for future work on this project and improvements that could be implemented, include finalizing the plan of using the Raspberry Pi 4 microcontroller for remote desktop utilization, as well further improving the CV software that was used. This system has the potential to improve tremendously, since the robot could also be mounted onto a vehicle to allow it to move to a plant with a tomato on it instead of it being stationary.

A more impressive design that could be built upon this project, would be having the robotic arm be on top of a moving vehicle that would move to a plant, pick a tomato, and then drop it in a basket on the back of the vehicle. This sort of design would be extremely challenging, but one can use the foundation that was established in this project to build onto and expand further.

As a condition for receiving the mini grant, we are expected to present this work at the Undergraduate Research Symposium at UCR in May. The project will incorporate the neural network to

identify the red balls and enhance the Object Recognition program. Also, enhance IK to reduce the errors of picking the ball.

11.3 * Acknowledgement

Professor Roman Chomko:

For helping to simplify our design for demonstration purposes and make our end product possible.

Merrick Campbell:

For providing insight and helpful tips and techniques for the various challenges and obstacles that had to be overcome.

Dr. Amit K. Roy-Chowdhury:

For releasing a Intel RealSense D455 camera to Manglai for the use of the project.

Manglai Zhou

For allowing us to use the Intel RealSense D455 camera when needed.

Dr. Konstantinos Karydis

For providing a recommendation for the team to apply for the Mini-Grant by the Undergraduate Education at UCR and guidance on inverse kinematics.

University of California Riverside Undergraduate Education

For providing funding to purchase an Intel RealSense D435i for the project.

Members of the Autonomous Robots and Controls Systems Lab at UCR

For assisting on having the camera and guidance on the computer vision and the inverse kinematics portion.

12 * References

[1] "IntelRealSense/librealsense,"*GitHub*. https://github.com/IntelRealSense/librealsense/blob/master/doc/installation_raspbian.md.

[2] "The design of a force feedback soft gripper for tomato harvesting | Journal of Agricultural Engineering,"*www.agroengineering.org*. https://www.agroengineering.org/index.php/jae/article/view/1090/900.

[3] "Computer Vision | Xailient," *console.xailient.com*. https://console.xailient.com (accessed Nov. 21, 2021).

[4] "Installation — Anaconda documentation," *docs.anaconda.com*. https://docs.anaconda.com/anaconda/install/index.html.

[5] "XailientPublic/example_scripts," *GitHub*, Sep. 10, 2021. https://github.com/XailientPublic/example_scripts/blob/master/inference_on_video.py (accessed Nov. 14, 2021).

[6]"Docker Container - Xailient Docs," *xailient-docs.readthedocs.io*. https://xailient-docs.readthedocs.io/en/latest/container.html.

[7] "OpenCV: OpenCV modules," *docs.opencv.org*. https://docs.opencv.org/4.x/.

[8] "Intel® RealSense™ Depth Camera D435i," *store.intelrealsense.com*. https://store.intelrealsense.com/buy-intel-realsense-depth-camera-d435i.html.

[9] "Hiwonder," *www.hiwonder.com*. https://www.hiwonder.com/store/learn/2.html (accessed Noc. 14, 2021).

[10] "Intel® RealSense™ Cross Platform API: Class List," *intelrealsense.github.io*. https://intelrealsense.github.io/librealsense/doxygen/annotated.html (accessed Nov. 14, 2021).

[11] G. D. R. N. Denbsky, "Use of hydrogen peroxide as fuel and/or oxygen supplier in reciprocating piston engines." https://patents.google.com/patent/DE3545049A1/en (accessed Nov. 21, 2021).

[12] R. N. Jazar, *Theory of Applied Robotics*. Boston, MA: Springer US, 2010.

[13] "IntelRealSense/librealsense," *GitHub*. https://github.com/IntelRealSense/librealsense/blob/master/doc/installation_raspbian.md.

[14] "Modern Robotics - Northwestern Mechatronics Wiki," *Northwestern.edu*, 2017. http://hades.mech.northwestern.edu/index.php/Modern_Robotics (accessed Nov. 12, 2021).

[15] "tomatOD," *GitHub*, Aug. 25, 2021. https://github.com/up2metric/tomatOD (accessed Nov. 14, 2021).

[16] "Ovonic RC LiPo Batteries, Australia, Russia, Singapore, Japan,MY, etc," *Ovonicshop*. https://www.ovonicshop.com/ (accessed Nov. 14, 2022).

[17] "Tomato Harvest Robot: Insane Japanese technology .. Robots to cut tomatoes .. If you look, your eyes should be distracted .. | Japanese tomato harvest robot in action in Tomato World Product of Inoho Europe." https://pipanews.com/tomato-harvest-robot-insane-japanese-technology-robots-to-cut-tomatoes-if-you-look-your-eyes-should-be-distracted-japanese-tomato-harvest-robot-in-action-in-tomato-world-product-of-inoho/ (accessed Nov. 14, 2022).

[18] Grayrids, "Artificial Neural Network Solution (ANS)," *AI*. https://www.anscenter.com/ (accessed Nov. 14, 2021).

[19] Spyder Team, "Spyder Website," *Spyder-ide.org*, 2018. https://www.spyder-ide.org/.

[20] Thingiverse.com, "Mold for Soft actuator/gripper - Parametric - OpenSCAD by JB86," *www.thingiverse.com*. https://www.thingiverse.com/thing:4396187.

[21] J. Zhang, S. Lai, H. Yu, E. Wang, X. Wang, and Z. Zhu, "Fruit Classification Utilizing a Robotic Gripper with Integrated Sensors and Adaptive Grasping," *Mathematical Problems in Engineering*, vol. 2021, p. e7157763, Sep. 2021, doi: 10.1155/2021/7157763.

[22] scikit-learn, "scikit-learn: machine learning in Python — scikit-learn 0.20.3 documentation," *Scikit-learn.org*, 2019. https://scikit-learn.org/stable/.

13 * Appendices

*** Appendix A:** Parts List

- LewanSoul Hiwonder 6 DOF Robotic Arm
- Microcontroller: Arduino Mega
- Microcontroller: Raspberry Pi 4
- Breadboard
- Various electrical wires
- 2x Lipo Batteries

*** Appendix B:** Equipment List

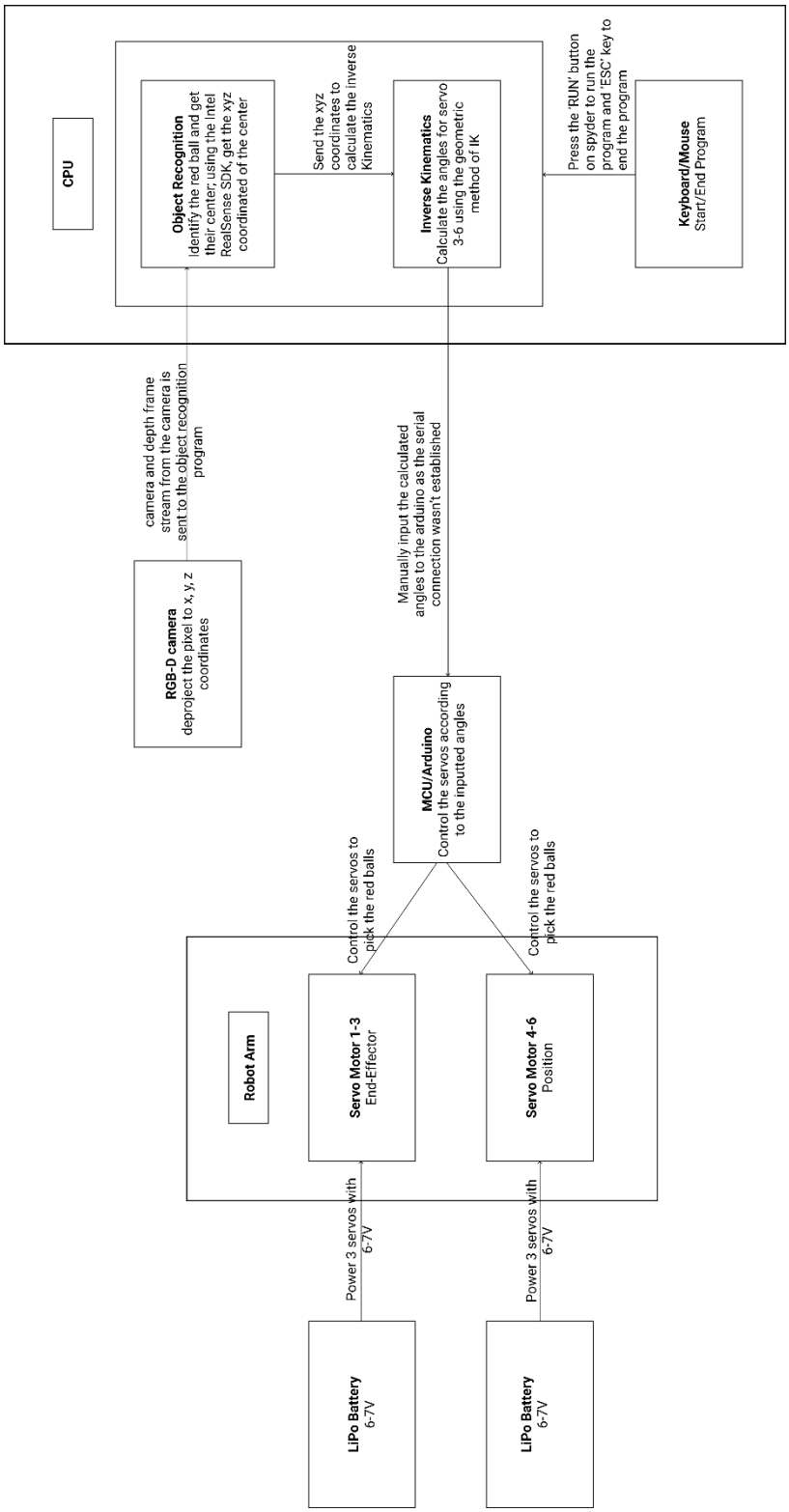
- 2x USB Cables: Connect camera to Raspberry Pi or desktop computer and Arduino Mega
- Laptop: Contained software related to the project and was uploaded to both microcontrollers.
- Multimeter: Used to check voltages during testing of our power supplies

*** Appendix C:** Software List (URL to online drive or SVN server, with sharing set to Public. Can omit this appendix if your project didn't involving writing a program)

- Arduino IDE: For programming on the Arduino Mega
- Debian OS: For programming on the Raspberry Pi 4
- Anaconda:

Appendix D: Special Resources
Appendix E: User's Manual - If your design requires instructions for future use, here is the place to put that information.

Appendix F: System Block Diagram


CPU
Object Recognition
Identify the red ball and get their center; using the Intel RealSense SDK, get the xyz coordinated of the center
Send the xyz coordinates to calculate the inverse Kinematics
Inverse Kinematics
Calculate the angles for servo 3-6 using the geometric method of IK
Press the 'RUN' button on spyder to run the program and 'ESC' key to end the program
Keyboard/Mouse
Start/End Program
camera and depth frame stream from the camera is sent to the object recognition program
RGB-D camera
deproject the pixel to x, y, z coordinates
Manually input the calculated angles to the arduino as the serial connection wasn't established
MCU/Arduino
Control the servos according to the inputted angles
Control the servos to pick the red balls
Control the servos to pick the red balls
Robot Arm
Servo Motor 1-3
End-Effector
Servo Motor 4-6
Position
Power 3 servos with 6-7V
Power 3 servos with 6-7V
LiPo Battery
6-7V
LiPo Battery
6-7V