TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

Major Project Final Report

On

Learning-based Model Predictive Controller for Drink Dispensing Robotic Arm Relying on Multimodal Inputs

Submitted By:

Joseph Thapa Magar (THA076BEI011)

Ritu Ram Ojha (THA076BEI024)

Rupak Mani Sharma (THA076BEI027)

Sujan Prasad Bhattarai (THA076BEI037)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

March 2024

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

Major Project Final Report

On

Learning-based Model Predictive Controller for Drink Dispensing Robotic Arm Relying on Multimodal Inputs

Submitted By:

Joseph Thapa Magar (THA076BEI011)

Ritu Ram Ojha (THA076BEI024)

Rupak Mani Sharma (THA076BEI027)

Sujan Prasad Bhattarai (THA076BEI037)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

In partial fulfillment for the award of the Bachelor's Degree in Electronics, Communication and Information Engineering.

Under the Supervision of

Er. Dinesh Baniya Kshatri

March 2024

DECLARATION

We hereby declare that the report of the project entitled "**Learning-based Model Predictive Controller for Drink Dispensing Robotic Arm Relying on Multimodal Inputs**" which is being submitted to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, in the partial fulfillment of the requirements for the award of the Degree of **Bachelor of Engineering in Electronics, Communication and Information Engineering**, is a bonafide report of the work carried out by us. The materials contained in this report have not been submitted to any University or Institution for the award of any degree and we are the only author of this complete work and no sources other than the listed here have been used in this work.

Joseph Thapa Magar (Class Roll No: 076/BEI/011) _____

Ritu Ram Ojha (Class Roll No: 076/BEI/024) _____

Rupak Mani Sharma (Class Roll No: 076/BEI/027) _____

Sujan Prasad Bhattarai (Class Roll No: 076/BEI/037) _____

Date: March, 2024

CERTIFICATE OF APPROVAL

The undersigned certify that they have read and recommended to **the Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, a major project work entitled "**Learning-based Model Predictive Controller for Drink Dispensing Robotic Arm Relying on Multimodal Inputs**" submitted by **Joseph Thapa Magar**, **Ritu Ram Ojha**, **Rupak Mani Sharma** and **Sujan Prasad Bhattarai** in partial fulfillment for the award of Bachelor's Degree in Electronics, Communication and Information Engineering. The Project was carried out under special supervision and within the time frame prescribed by the syllabus.

We found the students to be hardworking, skilled and ready to undertake any related work to their field of study and hence we recommend the award of partial fulfillment of Bachelor's degree of Electronics, Communication and Information Engineering.

_____

Project Supervisor
Er. Dinesh Baniya Kshatri
Department of Electronics and Computer Engineering, Thapathali Campus

_____

External Examiner

_____

Project Co-ordinator
Mr. Umesh Kanta Ghimire
Department of Electronics and Computer Engineering, Thapathali Campus

_____

Mr. Kiran Chandra Dahal
Head of the Department,
Department of Electronics and Computer Engineering, Thapathali Campus

March, 2024

COPYRIGHT

ACKNOWLEDGEMENT

We would like to express our gratitude to the head of Department of Electronics and Computer Engineering of Thapathali Campus and our supervisor, Er. Dinesh Baniya Kshatri, for offering this opportunity. We are deeply beholden to deputy head, Er. Umesh Kanta Ghimire, of Electronics and Computer respectively for your unwavering support, helpful feedback, beneficial suggestions, and positive criticism in research and study for the preparation of this proposal. This wouldn't have been possible without your direction, advice, instruction, and counseling. We would also like to acknowledge Real Time Solutions for granting hardware support to our project. Finally, we are also grateful to our classmates, colleagues of Thapathali Campus as well as schools (Mega, Texas International) for helping us in making our Audio Dataset.

Joseph Thapa Magar (THA076BEI011)

Ritu Ram Ojha (THA076BEI024)

Rupak Mani Sharma (THA076BEI027)

Sujan Prasad Bhattarai (THA076BEI037)

ABSTRACT

The project presents a voice-controlled four degrees of freedom robotic arm to automate the monotonous task of drink serving in a bartender domain. The system also consists of vision sensor to localize the glasses, drink dispenser and any obstacles within the robot's workspace. It incorporates a Squeezeformer architecture for voice command recognition where customers can request drinks using voice commands, You Only Look Once (YOLO) architecture to detect and localize the glasses, Model Predictive Control (MPC) for precise movement of robotic arm and drink dispenser to dispense the ordered drink by the customer. The whole system is realized in virtual as well as physical environment. CoppeliaSim is used to design the virtual environment where the robotic arm designed in Fusion360 is imported. The physical robot implements the cyclodial drive to increase the output torque of bipolar Nema17 motor and Gradient Descent method is used to perform the inverse kinematics calculations. The object detection model provides the mAP50 score of 0.989 for virtual environment and mAP50 score of 0.989 for physical environment. Also, the speech recognition model generated a Character Error Rate (CER) of 3.47% and Word Error Rate (WER) of 4.46% on training dataset.

Keywords: CoppeliaSim, Cyclodial, Gradient, MPC, Squeezeformer, YOLO

Table of Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interface
AST	Audio Spectrogram Transformer
C2f	Convolution to Fully Connect
CLAHE	Contrast Limited Adaptive Histogram Equalization
CLI	Command-Line Interface
CNN	Convolutional Neural Network
CSI	Camera Serial Interface
CV	Computer Vision
CVAT	Computer Vision Annotation Toolkit
DC	Direct Current
DCT	Discrete Cosine Transform
DOF	Degree of Freedom
FLOPs	Floating-Point Operations
IoU	Intersection over Union
LSTM	Long Short-Term Memory
mAP	Mean Average Precision
MFCC	Mel Frequency Cepstral Coefficient
MHA	Multi-Head Attention
MPC	Model Predictive Control
NLP	Natural Language Processing
NMS	Non-Maximum Suppression
PID	Proportional Integral Derivative
SPFF	Spatial Pyramid Pooling with Fuse
STFT	Short-Time Fourier Transform
VLF	Varifocal Loss
WER	Word Error Rate
YOLO	You Only Look Once

1. INTRODUCTION

1.1 Background

In recent years, robotics has advanced significantly, enabling sophisticated interactions with humans. The integration of speech recognition model and computer vision has revolutionized human-robot interaction. Speech recognition allows robots to interpret and respond to English language commands, while computer vision enhances their perception and manipulation abilities. This opens up practical applications in various industries, improving productivity and collaboration between humans and robots. Our focus on English language implementation aims to create an inclusive and user-friendly era of intelligent human-robot interactions.

The system incorporates a microphone for voice input, utilizing speech recognition algorithms to interpret user commands. The voice commands are processed using speech recognition algorithms to extract the intended actions or tasks. Simultaneously, computer vision techniques, including image processing and object recognition, are employed to capture visual input from an attached camera, enabling the arm to perceive and understand its surroundings. Leveraging control technique Model Predictive Control (MPC), the system plans and executes precise movements of the robotic arm, considering the current state, target positions, and physical constraints. Additionally, the system features a dispenser mechanism for drinks dispensing. After dispensing, the drinks will be served to customers and customers can order new drinks if they like.

1.2 Motivation

In the field of robotics, advancements in NLP and Computer Vision have enabled robots to understand and respond to human commands. Nowadays, many fancy restaurants and bars are using intelligent robots to serve drinks as a bartender. Traditional methods of manual drink serving can be time-consuming, monotonous, inefficient and prone to errors. The automation of drink serving through a robotic arm not only eliminates the need for manual intervention but reduces the chances of errors or spillage and by incorporating it into daily tasks, robots' technology will also aid human adaptation to it. By leveraging model predictive control mechanism in the stationary robotic arm, the system can accurately detect and grasp glasses, identify

drink containers, and precisely serve the desired beverages to the customers, thereby enhancing efficiency and consistency in the serving process.

1.3 Problem Definition

Many industrial applications of robotic system for drink serving mechanism include the control of robotic arm in the pre-determined trajectory and the location of drink dispenser as well as glasses are also in a fixed place. If the location of any of these things changes even within the reach of the robotic arm, the system may require additional tuning to work efficiently. Also, the human robot interaction in such systems is through the digital interface which may not always be easier for the customers. The system addresses this problem by incorporating the NLP techniques for voice commands interactions with the system, uses vision-based planning that accounts for the changes in ideal environment as well as customer's location and model predictive control mechanism for efficient and robust robotic arm movement enhancing efficiency and consistency in the serving process.

1.4 Objectives

- To model and simulate a drink dispensing robotic arm.
- To instantiate the robotic arm and perform performance comparison between simulation and reality.

1.5 Scope and Applications

The system has several impressive capabilities that contribute to its effectiveness in intelligent drink dispensing. Firstly, its language understanding capabilities enable users to easily communicate their drink preferences and customization options through natural language commands. This enhances user convenience and eliminates the need for complex input methods. Secondly, the integration of YOLO computer vision techniques empowers the system to accurately detect and recognize drink containers and related objects in real-time. This capability ensures precise object manipulation and allows for efficient and reliable drink dispensing. Finally, the learning-based model predictive control (MPC) framework enables the system to achieve precise and responsive control over the robotic arm. This results in accurate positioning and smooth

movements, enhancing the overall efficiency and effectiveness of the drink dispensing process.

Despite its impressive capabilities, the system does have certain limitations that should be acknowledged. Firstly, its language understanding capabilities may be limited to specific languages, or a predefined vocabulary related to drink preferences. This could potentially lead to difficulties in accurately interpreting complex or less common language constructs. Secondly, the system's object recognition capabilities, while robust with YOLO, may still encounter challenges in scenarios with difficult lighting conditions, occlusions, or objects with similar appearances. These limitations could impact the accuracy of object recognition and subsequently affect the system's performance. Additionally, like any robotic system, the system has physical constraints such as limited reach, weight-bearing capacity, and speed. These limitations need to be considered in terms of the size and weight of drink containers the system can handle, as well as its overall workspace limitations. Regular maintenance and robustness against external disturbances are also important factors to address for ensuring optimal system performance and reliability.

The developed robotic hand system for drink service using English language manipulation has a wide range of practical applications in various domains:

1.5.1 Medical and Healthcare

Robotic arms are employed in medical applications, including surgical procedures, rehabilitation, and diagnostics. Surgical robots assist surgeons in performing complex procedures with enhanced precision, minimal invasiveness, and improved patient outcomes. Robotic arms also aid in prosthetics, exoskeletons, and mobility assistance devices.

1.5.2 Aerospace and Defense

Robotic arms can be utilized in aerospace manufacturing for tasks like aircraft assembly, riveting, and inspection. They can also employed in space exploration for tasks such as satellite deployment and maintenance. In defense applications, robotic arms can handle hazardous materials, disarm explosive devices, and assist in

reconnaissance.

1.5.3 Manufacturing and Industrial Automation

Robotic arms can be extensively utilized in manufacturing processes for tasks such as assembly, welding, painting, pick and place operations, and packaging. They offer high-speed and high-precision capabilities, resulting in increased efficiency, reduced labor costs, and improved product quality

1.5.4 Hazardous Environments

Robotic arms can be utilized in environments that are dangerous or inaccessible to humans. Examples include nuclear power plants, deep-sea exploration, mining, and disaster response. They can handle hazardous materials, perform inspections, and assist in rescue operations.

1.6 Report Organization

The presented project report has been categorized into nine chapters. Chapter 1 introduces the project and its objectives, providing background information on the subject matter, including the motivation behind the project, a clear definition of the problem being addressed, and the specific objectives to be achieved. Chapter 2 is the literature review, this includes studies, articles, and publications related to model predictive control, computer vision, speech recognition, and other pertinent topics. Chapter 3 describes essential hardware and software components required for the project. It includes an explanation of sensors, controllers, and other hardware elements for the robotic arm, along with software tools like Fusion360 and CoppeliaSim. Chapter 4 discusses about system architecture and methodology employed in the project during simulation and real-world implementation. Chapter 5 delves into the practical implementation of the project, including a description of the virtual and physical environments used, the design procedures for the robotic arm and dispenser, the control mechanisms for motors and actuators, and the integration of vision and speech recognition modules. Chapter 6 presents the results obtained from the project implementation. This includes accuracy of speech recognition and object detection, performance of the model predictive controller, anomalies encountered during testing

and performance comparison between simulation and reality. Chapter 7 discusses potential improvements and developments that can be made to the project. Chapter 8 discusses the overall project in summary and gives brief conclusion about it. Chapter 9 contains supplementary materials, including project schedules, budget analysis, dataset collections, and any additional documentation. A comprehensive list of all sources cited in the report is included in the references section, ensuring the credibility and academic integrity of the project.

2. LITERATURE REVIEW

The authors of [1] proposed a Three-Degrees-of-Freedom robotic arm setup with monocular camera that can recognize the color and depth of the object. Then the object is localized and inverse kinematics is used to find the required rotating angles for each joints of the robotic arm. The color of the objects are identified based on HSV values and the inverse kinematics problem for the robotic arm is solved using geometric approach. It used Model Predictive Control (MPC) to optimize the input torques on each joints of the robotic arm, enabling it to move towards the object and grasp it precisely. [1] Chose the servo motor (DS3218) with torque value of 0.2kg-m. They have used software like MATLAB and SolidWorks for simulation purposes.

The authors of [2] investigates the Conformer model, widely adopted for various speech tasks due to its attention-convolution architecture capturing local and global features. However, after a careful analysis of design choices, the study reveals suboptimal performance. To address this, Squeezeformer is proposed, consistently outperforming state-of-the-art ASR models with the same training schemes. Macro-architecture enhancements include the Temporal U-Net structure, reducing computational cost on long sequences, and a simplified block structure replacing Macaron. Micro-architecture improvements involve simplified activations, reduced Layer Normalization, and an efficient depthwise down-sampling layer. Squeezeformer achieves outstanding results on LibriSpeech test-other, with 7.5%, 6.5%, and 6.0% WER, respectively, outperforming Conformer-CTC by 3.1%, 1.4%, and 0.6% without additional FLOPs. The review emphasizes Squeezeformer's contributions in enhancing performance and computational efficiency in ASR models, providing open-source code for wider accessibility.

The paper [3] introduces the Speech-Transformer, a novel sequence-to-sequence model for speech recognition that addresses the slow training speed issue faced by recurrent models due to internal recurrence limiting parallelization. The Speech-Transformer model relies entirely on attention mechanisms to learn positional dependencies, enabling faster and more efficient training. Additionally, the authors propose a 2D-Attention mechanism that jointly attends to the time and frequency axes of 2-dimensional speech inputs, enhancing the model's ability to capture expressive

representations. The model is evaluated on the Wall Street Journal (WSJ) speech recognition dataset, achieving a competitive word error rate (WER) of 10.9%. Impressively, the entire training process only requires 1.2 days on a single GPU, significantly faster than the published results of recurrent sequence-to-sequence models. The paper highlights the effectiveness of the Speech-Transformer in speech recognition tasks, showcasing its potential as an efficient alternative to recurrent models. The study contributes to the advancement of attention-based architectures in speech recognition.

The authors of [4]presents a fashion called LIFAS for automated language identification in voice recognition. LIFAS utilizes convolutional neural networks (CNNs) and achieves a high delicacy of 97 for double language bracket and 89 for multi-class bracket with six languages. The approach leverages deep literacy and spectrograms generated from audio signals, demonstrating the eventuality for accurate language identification in voice recognition systems.

The authors of [5]propose a deep literacy- grounded methodology that capitalizes on the use of spectrograms and Convolutional Neural Networks (CNNs). By using spectrograms as input and training a CNN model, the system autonomously learns and excerpts applicable verbal features from raw audio data. This eliminates the need for laborious homemade point engineering, which is frequently private and time-consuming. The data-driven nature of the proposed methodology expedites the development process and improves the accuracy and scalability of the language identification system. The proposed model exhibits remarkable performance, achieving an accuracy rate of 91.5 in relating to spoken languages. This accuracy surpasses the results attained by traditional language identification styles, suggesting that the integration of deep literacy ways significantly enhances the effectiveness of the system.

The paper [6] presents an on-wearable and non-intrusive result for fall discovery in the restroom terrain. By exercising ambient sound input recorded in homes, the proposed system achieves an accuracy of 0.8673. The approach eliminates the need for wearable devices. This has implicit operations in senior homes, hospitals, and recuperation installations, offering dependable fall discovery without constant monitoring.

The authors in [7]discussed about speech recognition using spectogram images and deep convolution neural network (CNN) of Uzbek spoken digits. Spectogram images from speech signal were generated and it were used for deep CNN training. There are 1,900 images of spectrograms with a size of 128 × 128 obtained from spoken digits from 0 to 9 digits in the Uzbek language in dataset. The speech signal were recorded in one channel with the frequency of 16 Khz. The classification accuracy of the spectogram image had achieved 99.7% in the test set and the validation set classification accuracy had achieved 96%. Authors of[7]concluded that the spectrogram images can be used in image recognition to extract different features in speech and is possible to achieve good results by classifying the sounds of Uzbek words.

The authors of [8]discussed that the CNNs have been widely used to learn a direct mapping from audio spectrograms to corresponding labels in the past decade. But to better capture long-range global context, a recent trend is to add a self-attention mechanism on top of the CNN, forming a CNN-attention hybrid model. The author of [8]introduced the Audio Spectrogram Transformer (AST), the first convolution- free, purely attention-based model for audio classification. The authors evaluate AST on various audio classification benchmarks, where it achieves new state-of-the-art results of 0.485 mAP on AudioSet, 95.6% accuracy on ESC-50, and 98.1% accuracy on Speech Commands V2. They also concluded that CNNs are not indispensable, and introduce the Audio Spectrogram Transformer (AST), a convolution-free, purely attention-based model for audio classification which features a simple architecture and superior performance.

This paper [9]proposed the process implemented in designing the motion of a bartender robot to be more engaging and visually attractive to grab the attention of a customer during the drink mixing and serving process. The system was designed to have a dual-arm manipulator, an overhead liquid dispenser system, a shaking station and an ice filing station. The industrial robot moves in a straight line which provides the shortest time trajectory in pick and place operation. But a magician or a bartender have their movement in a circular or an arc instead of a straight line. Two motion primitives: linear and circular trajectory, was defined is this paper. According to these motion primitives, the visual-attention of an observer was experimented. It is observed that the

circular motion primitives was more interesting to the customer but the linear motion was more natural for a robot.

This paper [10]describes about the design of a robotic arm having five degrees of freedom. It suggests that such robotic arm can be implemented to help the elderly for various basics tasks such as eating, picking up stuff which might be almost impossible for few as the motor skills dulls down as one grows older or one might suffer from medical conditions. The author uses a GUI interface and Arduino to interact with the robot. The robot uses stepper at the base and dc motor at other joints but the use of stepper motor at all joints could have been more preferable to get smooth motions which is crucial if the purpose of robot is to aid elderly. Similarly, hall sensor like AS5600 might be better as a positional encoder than a simple potentiometer

3. REQUIREMENT ANALYSIS

This section includes hardware and Software requirements of the project.

3.1 Hardware Requirements

The table provided enumerates all the necessary hardware components essential for project completion. Each entry includes a part number, detailed characteristics, purpose of use, and the respective quantity required.

Table 3-1: Hardware Requirements

Name	Part Number	Characteristics	Purpose	Quantity
Ball Bearings	6810	• 2zz double sided steel seal • 50x65x7mm	Gearbox	8
Ball Bearings	6804	• 2zz double sided steel seal • 20x32x7mm	Gearbox	16
Motor Driver	TB6600	• 4A current • 32 micro stepping	Stepper	4
Stepper Motor	Nema 17	• 45Ncm, • 60 Ncm • 1.7x1.7-inch cross section	Joints	4
Servo	MG90	• Metal Geared • 180 degrees of rotation • 20Ncm torque	Gripper	1
Power Supply	S-200-6	• 12V • 20A	Power	1
Webcam	Odroid	• 720P HD cam	Vison	1
Microphone	Fantech MCX01 Leviosa	• Condenser Microphone	Voice Dataset Collection	1
3D printer	Ender 3	• 220mmx220mmx 250mm • ABS support	Gearbox	1

3.2 Software Requirements

Below is a list of all the required software essential for building models, writing code, and other project purposes, along with their respective versions

Table 3-2: Software Requirements

Name	Version	Purpose
Pytorch	2.1.0+cu121	Creating and Training Speech Recognition and vision model
Python	3.10	Writing code for various purposes, including integration with other software modules.
Arduino IDE	2.3.2	Developing and writing code for Arduino microcontrollers, including robotic control.
Audacity	3.4.2	Manipulation of audio Datasets
CoppeliaSim	4.5.1	Simulating the robotic arm
Fusion 360	16.5.0	Designing the robotic arm
Cura	5.6	Slicing the mesh file
Play.ht	1.0	To generate voice clone

3.3 Feasibility Study

Technical Feasibility

Hardware Feasibility: The hardware feasibility of the proposed robotic arm project incorporating a 3D printed cycloidal gearbox, stepper motor, servo motor, and TB6600 driver suggests promising potential for successful implementation. Initial assessments indicate the durability and precision of the 3D printed gearbox components. The selected stepper motor meets torque and speed requirements, with satisfactory control feasibility using the TB6600 driver and Arduino microcontroller. Additionally, the MG90 servo motor exhibited promising results in the gripper, demonstrating sufficient torque to lift filled cups.

Software Feasibility: The feasibility of the project relies on the compatibility and functionality of key software tools, including PyTorch 2.1.0+cu121 for creating and training speech recognition and vision models, Arduino IDE 2.3.2 for developing code for Arduino microcontrollers and robotic control, and Audacity 3.4.2 for audio dataset manipulation. Their open-source nature encourages a collaborative development environment

Economic Feasibility: The economic analysis of the proposed robotic arm project primarily revolves around cost estimation, covering both initial investment and ongoing expenses. This includes assessing material costs for various components such as the 3D printed cycloidal gearbox, ball bearings, stepper motor, servo motor, TB6600 driver, and Raspberry Pi, alongside factoring in overhead expenses like equipment, facilities, and operational costs. Through the student's project grant provided by Real Time Solutions, the project becomes viable. Cost breakdown is available in Appendix Section

Legal Feasibility: All project actions were carried out with appropriate permissions in place. Formal authorization was obtained prior to dataset collection at various schools and colleges, with permission letters provided to the principals of each institution. Moreover, all software utilized adhered to legal requirements, ensuring that no illegal or pirated products were employed throughout the project's duration.

Timeframe Feasibility: The project is divided into two major objectives for each part. The first part of the project deals with realizing the whole functionality in virtual environment and the second part deals with making it in a physical environment. The project is completed as a two part system in a timeframe of 1 year. For more details, Gantt chart in Appendix Section can be referred.

4. SYSTEM ARCHITECTURE AND METHODOLOGY

This section includes overall architecture for speech recognition, vision, robot design and control mechanism of the complete system.

4.1 System Block Diagram

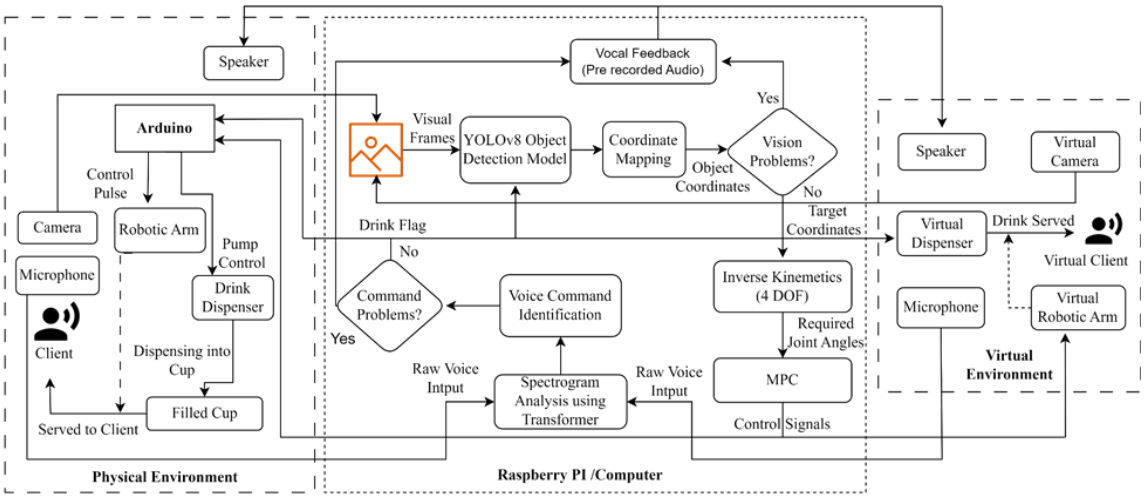


Figure 4-1: System Block Diagram

4.1.1 Microphone

The client places the order using the microphone and it is recorded for further processing.

4.1.2 Spectrogram Analysis using Transformer

This system accepts raw voice input and, through sophisticated analysis, generates the name of the drink as its output. This innovative approach showcases the power of deep learning techniques in processing and understanding voice commands for applications like automated drink ordering.

4.1.3 Command Problems

If the system doesn't recognize the command, then it provides vocal feedback requesting for new command. Otherwise, it provides control commands to the YOLOv8 object detection model as well as the dispenser control system.

4.1.4 Virtual Camera/Camera

The Vision Sensor captures frames of the project environment, encompassing various elements like dispensers, cups, robotic arms, and users. And outputs the captured frames to the YOLOv8 detection algorithm.

4.1.5 YOLOv8 Model

YOLOv8 Model takes the images of the environment provided by the vision sensor and detects the objects and estimate the corresponding position of the objects, which is further used as target coordinate for robotic arm.

4.1.6 Coordinate Mapping

The coordinates identified by the YOLOv8 model undergo a coordinate mapping technique to convert them into world coordinates. This mapping involves generating a transformation matrix that effectively converts the camera coordinates of the objects into corresponding world coordinates.

4.1.7 Vision Problems

The system conducts a thorough check for potential issues, such as the absence of cups or clients, or any obstructions in the path. If any problems are detected, the system delivers vocal error feedback through the speaker, alerting the users to address the issues accordingly.

4.1.8 Speaker

For any problem related to vision and command, there is vocal feedback to alert the user via speaker.

4.1.9 Inverse Kinematics via Gradient Descent

This method is used for solving inverse kinematics problem. It takes the target coordinates and orientation as input and provide the joints angle as output.

4.1.10 Model Predictive Controller

The MPC takes in the required joint angles as input to make a desired trajectory and generate the control signals that allows the robotic arm to follow that desired trajectory.

4.1.11 Arduino

It takes the controlled inputs from model predictive control block as well as desired drink information from the raspberry pi.

4.1.12 Virtual Robotic Arm

CoppeliaSim’s physics engine is responsible for the movement of robotic arm and any other 3D objects in the virtual environment. Angle to the arm is fed from MPC controller.

4.1.13 Robotic Arm

The controlled inputs from the model predictive control block is fed to each joints of the robotic arm via Arduino Uno board.

4.1.14 Drink Dispenser

The one of four DC motor pumps in dispenser run to dispense the desired drink ordered by the client according to the drink information sent from raspberry pi to Arduino Uno board.

4.1.15 Virtual Dispenser

In the virtual environment, a dispenser serves four different beverages. Rather than utilizing a pump to dispense water, the cup's color changes according to the ordered drink.

4.1 Robotic Arm Framework

The first thing done was the design of the 4-DOF robot in fusion 360. Along with the robot arm the design of the dispenser was also done using this software.

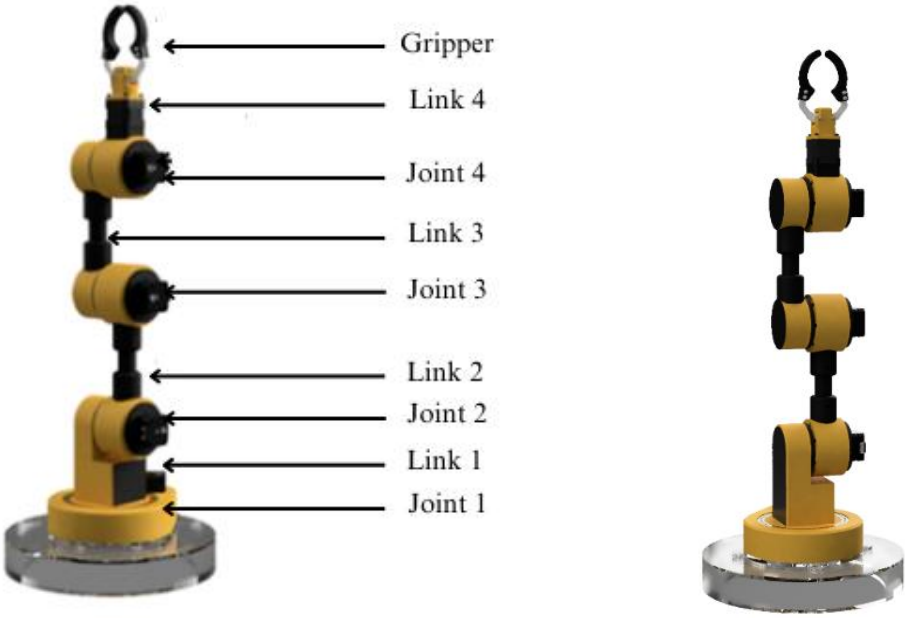


Figure 4-2: ¾ View and Front View

Figure 4-3: Side Views

4.1.1 Degrees of Freedom

The degrees of freedom of a robotic arm refer to the number of independent parameters

that define its motion. These parameters typically include rotations and translations along various axes. The number of DoF determines the complexity and versatility of the robot's movement. The minimum degrees of freedom required for a robotic arm to move freely in 3-D space is 6.

Translation along the x-axis: This allows movement in the horizontal direction.

Translation along the y-axis: This allows movement in the vertical direction.

Translation along the z-axis: This allows movement in the depth or forward-backward direction.

Rotation around the x-axis (roll): This enables rotation around the x-axis, often referred to as roll, which changes the orientation from side to side.

Rotation around the y-axis (pitch): This enables rotation around the y-axis, often referred to as pitch, which changes the orientation from up to down.

Rotation around the z-axis (yaw): This enables rotation around the z-axis, often referred to as yaw, which changes the orientation from left to right.

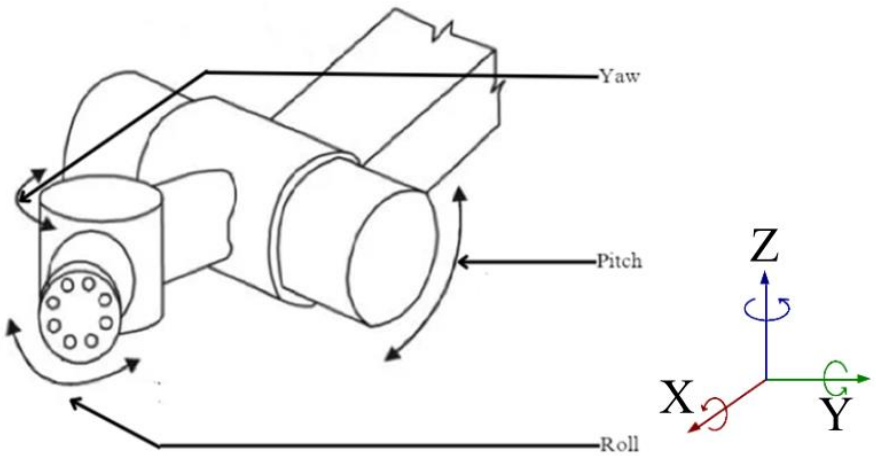


Figure 4-4: Roll, Pitch and Yaw

For the purpose of drink dispensing robotic arm, it must have the capability to translate in all axes but roll and yaw is redundant. Only pitch or rotation around the y-axis is

enough to move the cup with a drink without spilling. Hence 4 degree of freedom (3 translation and one rotation) is the appropriate choice.

4.1.2 Cycloidal Drive

A cycloidal drive is used to transmit torque or rotary motion between two shafts. It is characterized by its unique design that incorporates an eccentric disk with lobes that slides on to the ring pins to transfer the torque.

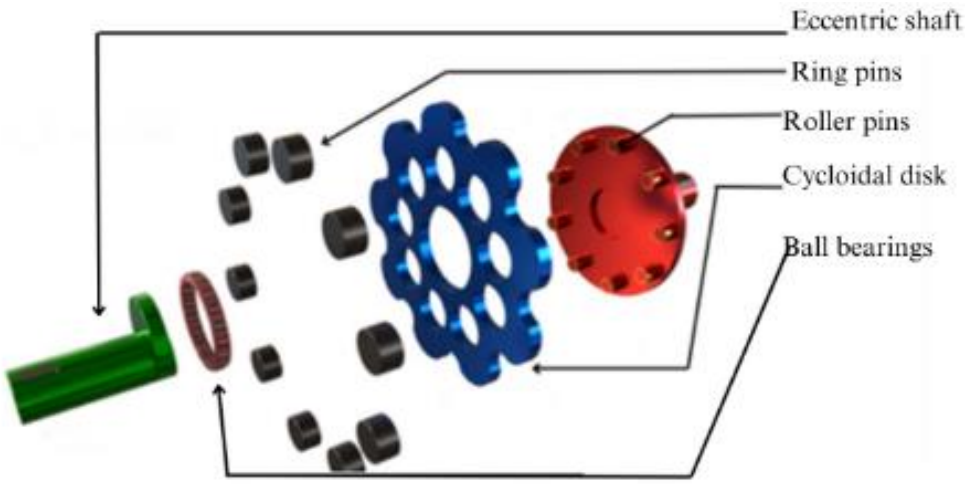


Figure 4-5: Cycloidal Drive Exploded View

Cycloidal Disk

A cycloidal disk is characterized by its shape, which typically has lobes or bumps arranged in a cycloidal pattern. It is mounted on the input shaft of the gearbox. As the input shaft rotates, the cycloidal disk creates a cycloidal motion to the output mechanism. This motion is achieved as the lobes slides on to the ring gear creating wobble like motion. When the point is inside the surface of the circle the curve formed becomes epitrochoid.

$$x(\emptyset) = (R + r)cos\emptyset - d\ cos\left(\frac{R+r}{r}\emptyset\right) \quad 4\text{-}1$$

$$y(\emptyset) = (R + r)sin\emptyset - d\ sin\left(\frac{R+r}{r}\emptyset\right) \quad 4\text{-}2$$

R: Radius of bigger circle

r: Radius of smaller circle

θ: Parameter

d: Distance of the point from the center of smaller circle

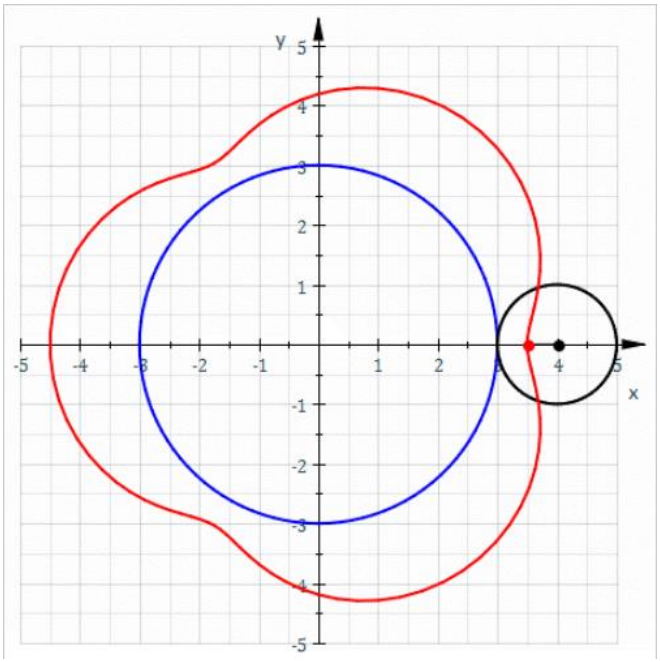


Figure 4-6: Epitrochoid Curve

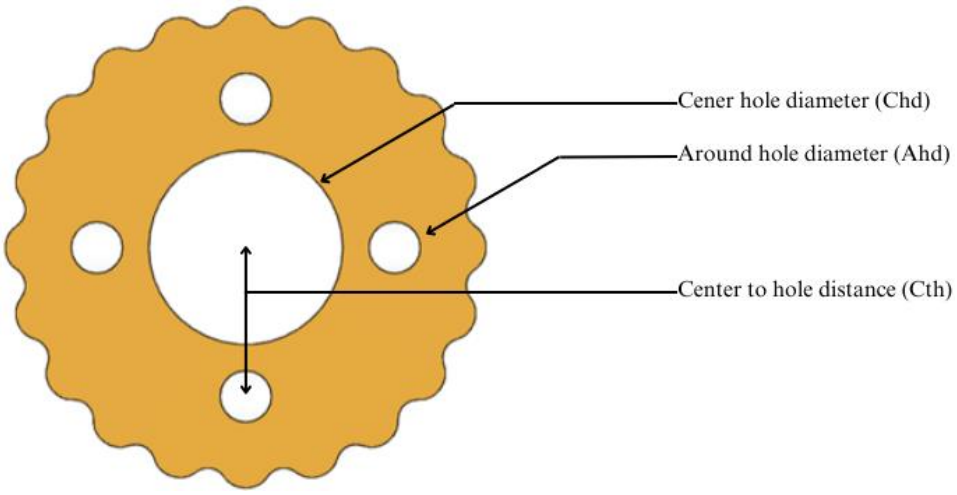


Figure 4-7: Cycloidal Disk

Housing/Case

It is the structure where the disk slides in between the rollers. It has one greater number of rollers compared to the number of lobes on the disk. In the following design the roller is permanently attached to the casing. Actual dimensions will be presented in the Implementation Details section.

Figure 4-8: Main Housing

Input Eccentric Shaft/Crank

The input eccentric shaft is attached to the input shaft which rotates the rotor on the roller. The point which is attached to the shaft of the input shaft is slightly eccentric which means the center of the input shaft is slightly offset from the center of cylinder which rotates the rotor or disk.

$$Eccentricity = 0.5 * ring\ pin\ radius \quad (4\text{-}3)$$

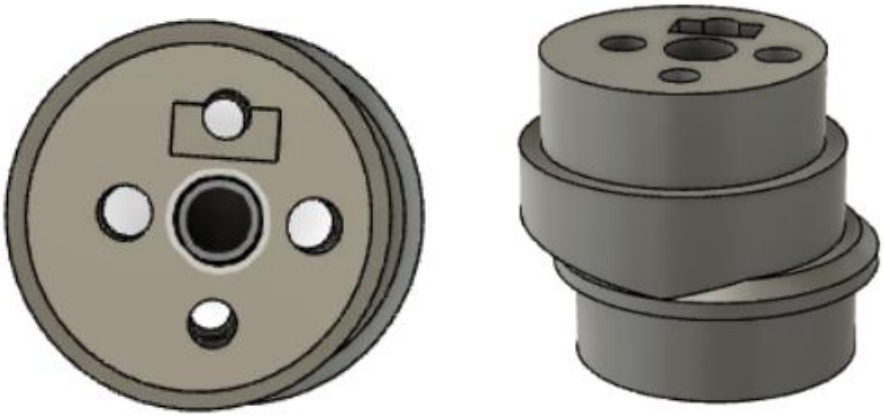


Figure 4-9: Eccentric Cam

Shaft Coupler

It is the body which is connected to the stepper motor. It has d cut slot so that the motor d shaft can firmly be secured to the body.

Figure 4-10: Stepper Shaft Coupler

Motor Mount

The stepper motor is secured to it via 3mm screws and finally the whole body is attached to the main housing.

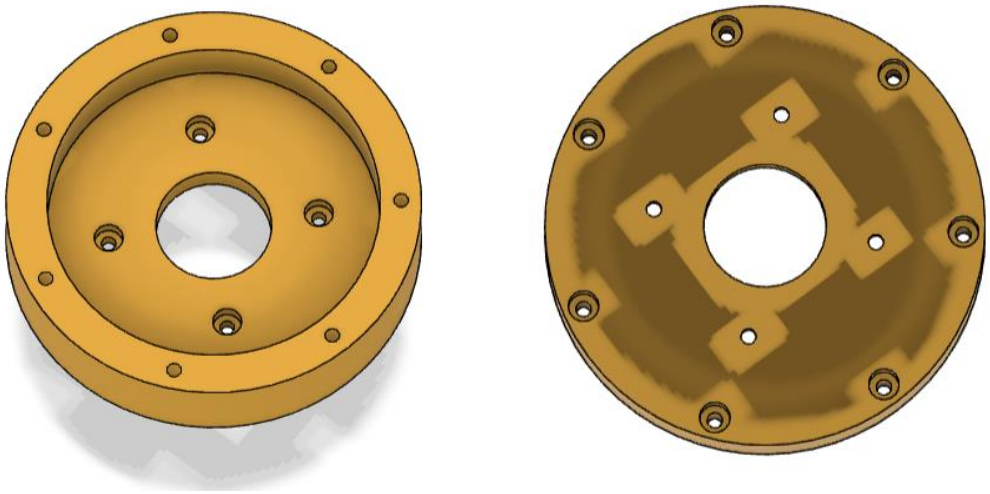


Figure 4-11: Stepper Motor Holder

Slow Speed Output Coupling

The rotational output is obtained through four steel rods coming out of the top which is inserted from the bottom body. These pair rotate in opposite direction of the input shaft direction of rotation.

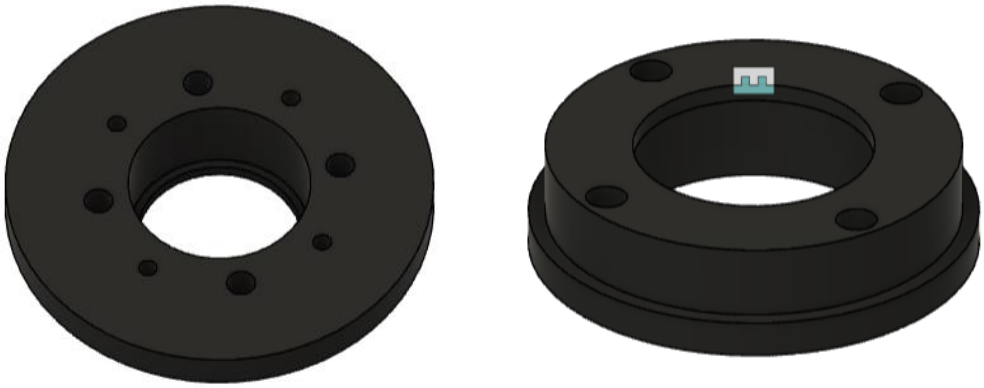


Figure 4-12: Output Coupling Pair

4.1.3 Cycloidal Disk Design Procedure

The reduction ratio (N) is selected according to the requirement, considering factors such as desired speed, torque, and efficiency.

Following the determination of the reduction ratio, the housing circle diameter is then

calculated to accommodate the required size. This dimensioning process ensures proper fit and structural integrity, essential for the smooth operation and durability of the system.

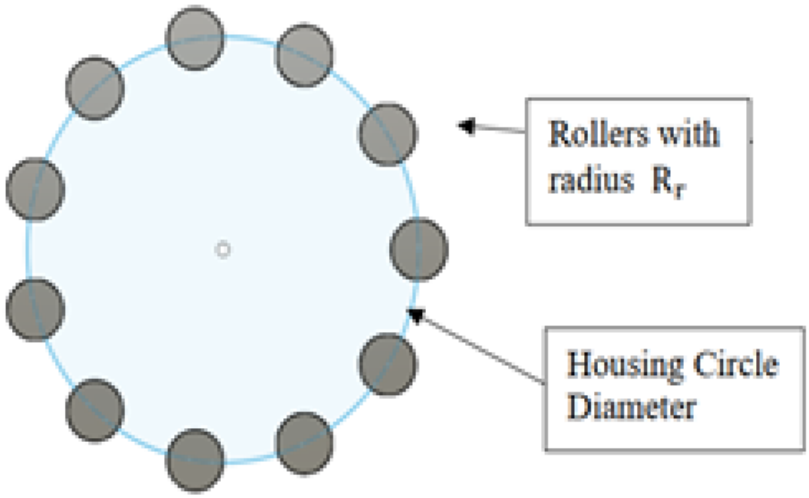


Figure 4-13: Sketch View

Afterwards the rolling pin (rollers) radius (Rr) is calculated to optimize contact area and load distribution.

$$Rr = \frac{Housing\ Circle\ Circumference}{2 * N} \quad 4\text{-}4$$

With the eccentricity determined as 0.5 times Rr, the motion and trajectory of the components are precisely defined, crucial for reliable performance.

$$\text{Eccentricity} = 0.5 * \text{Rr} \quad 4\text{-}5$$

Once these foundational elements are established, appropriate ball bearings size for the center hole of the cycloidal disk is selected. This choice is critical for supporting loads and ensuring smooth rotation, thus contributing to the longevity of the mechanism. Finally, consideration is given to determining the output holes number, diameter, and their distance from the center hole of the disk.

4.1.4 Joints

A revolute joint, also known as a rotary joint or hinge joint, is a type of mechanical joint that allows rotational motion around a single axis. It enables parts or segments of a mechanism to rotate relative to each other, mimicking the movement of a door hinge or the bending of a human arm at the elbow. The axis of rotation is fixed, and the motion is typically limited to a specified range. The cycloidal gearbox and stepper motor acts as a revolute joint in the robotic arm.

Table 4-1: Range of Motion of Joints

Joint	Range of Motion (Degrees)
Joint 1	0-360
Joint 2	-60-0, 0-60
Joint 3	-60-0, 0-60
Joint 4	-70-0, 0-70

4.1.5 Links

A rigid component that connects joints or other links together. These links typically form the structural framework of the robotic arm and determine its shape and range of motion. Each link is usually connected to one or more other links or joints, allowing the arm to move and manipulate objects.

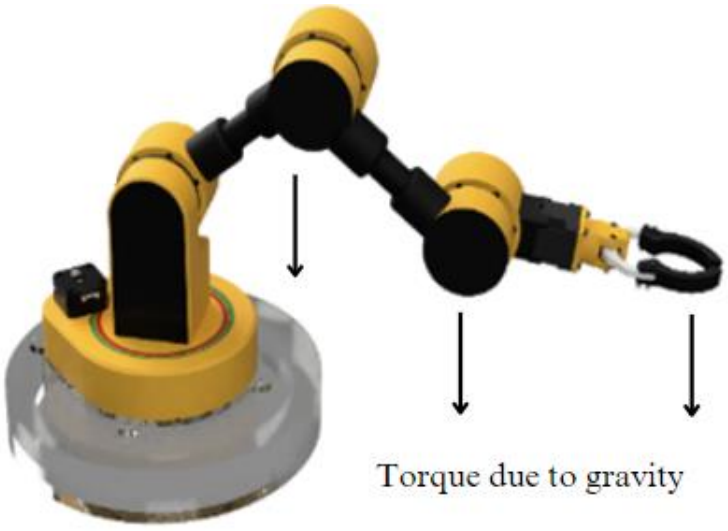


Figure 4-14: Various Sources of Torque

The maximum link length a robot can have depends upon the maximum torque a joint can produce. Longer link length increases torque as torque directly depends upon the distance from the axis of rotation.

Hence link length is determined according to the weight of the joints i.e., gearbox and stepper motors as well as the load it has to lift. Actual link lengths used for this project will be discussed in the Implementation Details section.

4.1.6 Gripper

The robotic arm is equipped with a specialized device called a gripper, which functions like its hand. This gripper is designed to grasp and hold objects securely, where the object is empty as well as filled cup in this scenario.

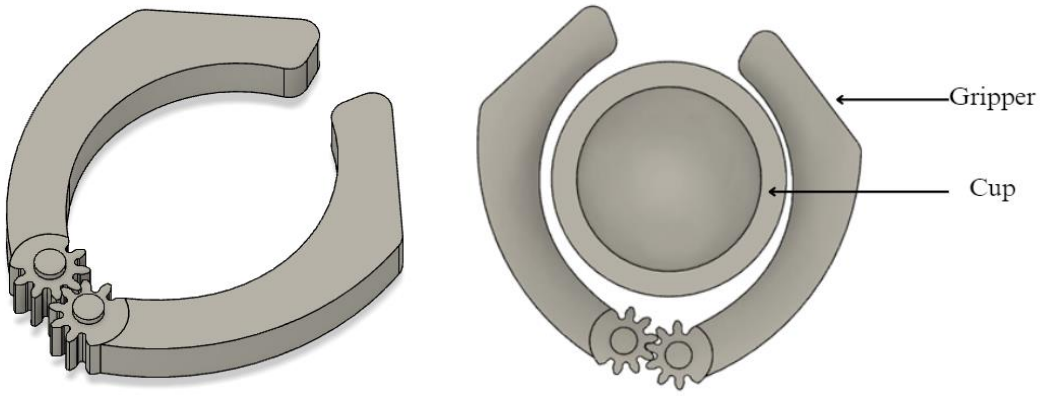


Figure 4-15: Gripper with Cup

The robotic arm is equipped with a claw-type gripper, resembling the claws of certain animals or machines used to grab objects. This gripper consists of two pincers that open and close to grip items securely. The design of the claw allows it to grab the cylindrical object securely.

4.1.7 Robotics Fixture Installation

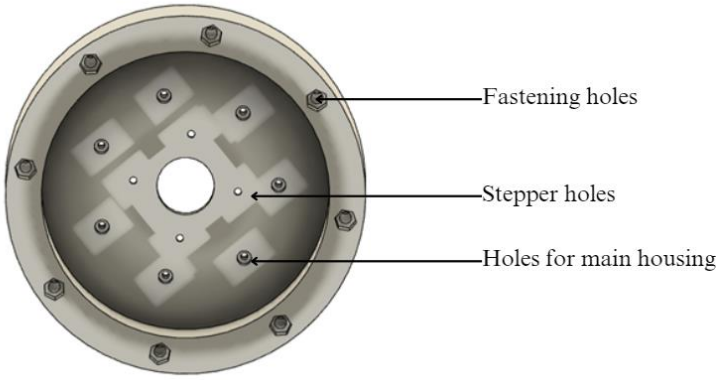


Figure 4-16: Fixing the Robotic Arm

The robot's base is securely attached to the table through a robust mechanism involving eight 4mm bolts. This process consists of carefully drilling corresponding holes into the table surface to accommodate the bolts. Once the holes are prepared, the bolts are inserted through the base of the robot and into the table. Subsequently, nuts are fastened onto the bolts, ensuring a tight and secure connection between the robot and the table. This process guarantees that the robot remains firmly fixed in its designated position, providing stability and reliability for its operations.

4.1.8 Stepper Motor

Bipolar Stepper Motor has two coils, and the direction of the current flow through the coils determines the direction of the motor's rotation. Having two coils we need Two H Bridge to drive each coil.

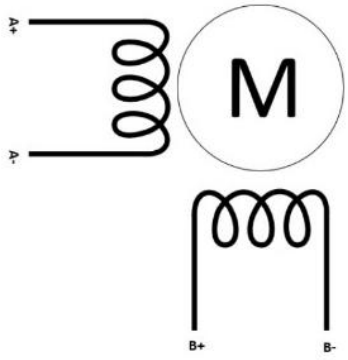


Figure 4-17: Nema17 Bipolar Stepper Wiring

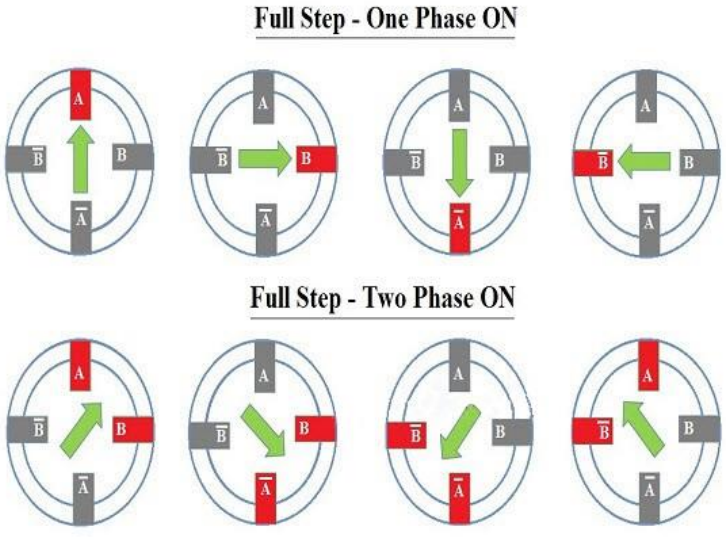
1-Phase Excitation Mode:

In this mode only one coil is excited at a time. Power consumption is less but the torque is also significantly low.

2-Phase Excitation Mode:

The motor can also be driven by 2-phase excitation method, both coils are energized simultaneously during each step. This means that current flows through both coils at the same time, producing a higher torque.

Both methods will require same number of steps to complete full revolution. For example, if the motor is of 200 steps per revolution both require 200 steps.



Step	Phase			
	A	B	$\overline{A}$	$\overline{B}$
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Step	Phase			
	A	B	$\overline{A}$	$\overline{B}$
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1

Figure 4-18: Excitation

4.1.9 Servo Motor

The gripper uses servo motor to grab the cup so that it can serve to the customer. A servo motor operates on the principle of feedback control to precisely control its position, speed, and torque. Inside the servo motor, there is a DC motor, gears, a potentiometer, and control electronics. Potentiometer is used to track the current position of the motor whereas the control electronics applies closed loop control system to minimize the error so that the servo can turn to the desired position.

Servos are controlled by sending an electrical pulse of variable width, or pulse width modulation (PWM), through the control wire. There is a minimum pulse, a maximum pulse, and a repetition rate. The PWM sent to the motor determines position of the shaft, and based on the duration of the pulse sent via the control wire; the rotor will turn to the desired position.

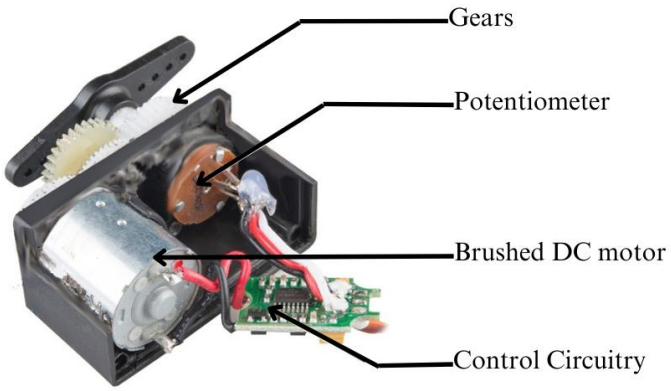


Figure 4-19: Servo Components

The servo motor expects to see a pulse every 20 milliseconds (ms) and the length of the pulse will determine how far the motor turns. For example, a 1.5ms pulse will make the motor turn to the 90° position. Shorter than 1.5ms moves it in the counter clockwise direction toward the 0° position, and any longer than 1.5ms will turn the servo in a clockwise direction toward the 180° position.

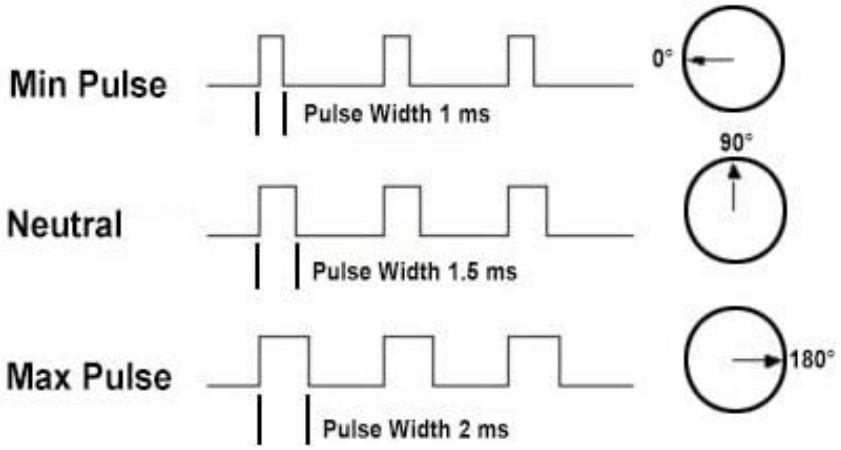


Figure 4-20: Servo Input Signals

4.2 Kinematic Model of Robotic Arm

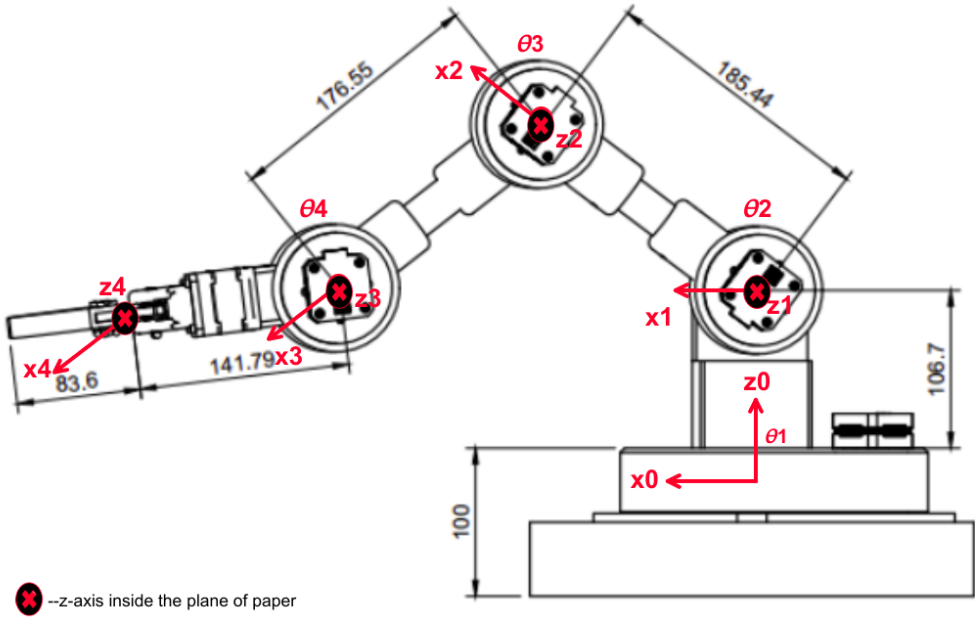


Figure 4-21: Frame Assignment on 4 DOF Robotic Arm (Measurements in mm)

It deals with the motion of robotic arm without considering forces and torques associated with it. It provides the position and orientation of end-effector based on robot joint's angular position. It ultimately provides the homogeneous transformation matrix that tells both the rotation and position of one frame n relative to another frame m. It is denoted by H_n^m.

Table 4-2: Nomenclature for Kinematic Model

Symbol	Remarks
θ_i	Angle of i^{th} joint
a_i	Length of i^{th} link
α	The amount of rotation of $frame_{i-1}$around axis x_i to get axis z_{i-1} to match z_i
r	The amount of displacement from $frame_{i-1}$to the $frame_i$ measured only in x_i direction
d	The amount of displacement from $frame_{i-1}$to the $frame_i$ measured only in z_{i-1} direction
c_i	$cos\theta_i$
s_i	$sin\theta_i$
s_{ij}	$\sin(\theta_i + \theta_j)$
c_{ij}	$cos(\theta_i + \theta_j)$
c_{ijk}	$cos(\theta_i + \theta_j + \theta_k)$
s_{ijk}	$sin(\theta_i + \theta_j + \theta_k)$
$\dot{x}, \dot{y}, \dot{z}$	Linear velocities of end effector in x,y,z direction
w_x, w_y, w_z	Angular velocities of end effector in x,y,z direction
$\dot{\theta}_i$	i^{th} joint velocities

Here, Denavit-Hartenberg method is used to find the homogeneous transformation matrix. In this method, Assignment of frames need to be done to each joints as well as the end effector according to DH frame assignment rules. Now, the DH parameters are to be found which are expressed in the table below:

Table 4-3: Denavit-Hartenberg Parameter Table

n (link)	Parameters			
	$\boldsymbol{\theta}$	$\boldsymbol{\alpha}$	**r**	**d**
1	θ_1	90°	0	a_1
2	θ_2	0	a_2	0
3	θ_3	0	a_3	0
4	θ_4	0	a_4	0

Once the parameter table is computed, Homogeneous transformation matrix is to be found from the frame n-1 to n by plugging in the parameter values from the table in the matrix given below:

$$H_n^{n-1} = \begin{bmatrix} c(\theta_n) & -s(\theta_n)c(\alpha_n) & s(\theta_n)c(\alpha_n) & r_n c(\theta_n) \\ s(\theta_n) & c(\theta_n)s(\alpha_n) & -c(\theta_n)s(\alpha_n) & r_n s(\theta_n) \\ 0 & s(\alpha_n) & c(\alpha_n) & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{4-6}$$

The transformation matrices calculated through DH parameters, for each links are given below:

$$H_1^0 = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & a\_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{4-7}$$

$$H_2^1 = \begin{bmatrix} c_2 & s_2 & 0 & a_2 c_2 \\ s_2 & c_2 & 0 & a_2 s_2 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{4-8}$$

$$H_3^2 = \begin{bmatrix} c_3 & -s_3 & 0 & a_3 c_3 \\ s_2 & c_3 & 0 & a_3 s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{4-9}$$

$$H_4^3 = \begin{bmatrix} c_4 & -s_4 & 0 & a_4 c_4 \\ s_4 & c_4 & 0 & a_4 s_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{4-10}$$

The overall transformation of end effector w.r.t base frame is computed as:

$$H_4^0 = H_1^0 H_2^1 H_3^2 H_4^3 = \begin{bmatrix} c_1 c_{234} & -c_1 s_{234} & s_1 & c_1(a_4 c_{234} + a_2 c_2 + a_3 c_{23}) \\ c_{234} s_1 & -s_1 s_{234} & -c_1 & s_1(a_4 c_{234} + a_2 c_2 + a_3 c_{23}) \\ s_{234} & c_{234} & 0 & a_4 s_{234} + a_2 c_2 + a_3 s_{23} + a_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{4-11}$$

To analyze the relationship between joint velocities and the resulting linear and angular velocities of the end effector, the Jacobian matrix is to be computed. It provides the

mapping between the joint space and the Cartesian space. In this case, it is a (6*4) matrix denoted by J and is represented as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ w_x \\ w_y \\ w_z \end{bmatrix} = J \begin{bmatrix} \dot{\theta_1} \\ \dot{\theta_2} \\ \dot{\theta_3} \end{bmatrix} \quad \text{4-12}$$

The Jacobian matrix corresponding to each link of our robot can be computed easily by the help of already found homogeneous transformation matrices. It is given by:

$$J_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad \text{4-13}$$

$$J_2 = \begin{bmatrix} -a_2 s_1 c_2 & -a_2 c_1 s_2 & 0 & 0 \\ a_2 c_1 c_2 & -a_2 a_1 s_2 & 0 & 0 \\ 0 & a_2 c_2 & 0 & 0 \\ 0 & s_1 & 0 & 0 \\ 0 & -c_1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad \text{4-14}$$

$$J_3 = \begin{bmatrix} -a_2 c_2 s_1 - a_3 c_{23} s_1 & -c_1(a_2 s_2 + a_3 s_{23}) & -a_3 c_1 s_{23} & 0 \\ a_2 c_1 c_2 + a_3 c_1 c_{23} & -s_1(a_2 s_2 + a_3 s_{23}) & -a_3 s_1 s_{23} & 0 \\ 0 & a_2 c_2 + a_3 c_{23} & a_3 c_{23} & 0 \\ 0 & s_1 & s_1 & 0 \\ 0 & -c_1 & -c_1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad \text{4-15}$$

$$J_4 = \begin{bmatrix} A & -c_1(a_4 s_{234} + a_2 s_2 + a_3 s_{23}) & -c_1(a_4 s_{234} + a_3 s_{23}) & -a_4 c_1 s_{234} \\ \mathrm{B} & -s_1(a_4 s_{234} + a_2 s_2 + a_3 s_{23}) & -s_1(a_4 s_{234} + a_3 s_{23}) & -a_4 s_1 s_{234} \\ 0 & a_2 c_2 + a_4 c_{234} + a_3 c_{23} & a_4 c_{234} + a_3 c_{23} & a_4 c_{234} \\ 0 & s_1 & s_1 & s_1 \\ 0 & -c_1 & -c_1 & -c_1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad \text{4-16}$$

Where,

A = $-s_1(a_4 c_{234} + a_2 c_2 + a_3 c_{23})$

B = $c_1(a_4 c_{234} + a_2 c_2 + a_3 c_{23})$

4.3 Dynamic Model of Robotic Arm

The dynamic model of the robotic arm gives information of torque and other forces resulting in the motion of the robot. The dynamic model derived here is formulated using Euler-Lagrange methods. This is the most commonly followed approach due to its simplicity and compact description.

Table 4-4: Nomenclature for Dynamic Model

Symbol	Remarks
m_i	Mass of link i
r_i	Distance of i^{th} link to its mass centre
M_i	Generalized mass matrices for the i^{th} link
$D(\theta)$	Manipulator inertia matrix
KE_i	Total kinetic energy related to each link
PE_i	Total potential energy related to each link
L	Lagrangian, difference of kinetic energy and potential energy
I_i	Inertia tensor of link i with reference to its frame
g	Acceleration of gravity
h_i	height for the mass center of the i^{th} link
τ	Vector of actuator torques
$g(\theta)$	vector of gravity forces
$C(\theta, \dot{\theta})$	Vector of Coriolis and Centrifugal forces

The Euler-Lagrange equation for the dynamic model of the robotic manipulator is given by:

$$\frac{d\frac{\partial L}{\partial \dot{\theta}}}{dt} - \frac{\partial L}{\partial \theta} = \tau \qquad 4\text{-}17$$

The kinetic energy of each link is added to find the kinetic energy of the robot which is given by:

$$KE = \sum_{i=1}^{n} KE_i(\theta, \dot{\theta}) = \frac{1}{2}\dot{\theta}^T D(\theta)\dot{\theta} \qquad 4\text{-}18$$

Where,

$$D(\theta) = \sum_{i=1}^{n} J_i^T M_i J_i \qquad 4\text{-}19$$

Now, the potential energy of each link of the robot is given by:

$$PE_i(\theta) = m_i g h_i(\theta) \qquad 4\text{-}20$$

Thus, the Lagrangian becomes:

$$L(\theta,\dot{\theta}) = \sum_{i=1}^{n}\left(KE_i(\theta,\dot{\theta}) - PE_i(\theta)\right) = \frac{1}{2}\dot{\theta^T}D(\theta)\,\dot{\theta} - PE(\theta) \qquad 4\text{-}21$$

For control purposes, it is common and more practical to rewrite the Euler–Lagrangian dynamic model of the robot in matrix form as follows:

$$D(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + g(\theta) = \tau \qquad 4\text{-}22$$

Where $C(\theta,\dot{\theta})$is given by:

$$C(\theta,\dot{\theta}) = \frac{1}{2}\left(\frac{\partial D_{kj}}{\partial \theta_i} + \frac{\partial D_{ki}}{\partial \theta_j} + \frac{\partial D_{ij}}{\partial \theta_k}\right) \qquad 4\text{-}23$$

The generalized mass matrix M_i simplifies to the following diagonal matrix:

$$M_i = \begin{bmatrix} m_i & 0 & 0 & 0 & 0 & 0 \\ 0 & m_i & 0 & 0 & 0 & 0 \\ 0 & 0 & m_i & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xi} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{yi} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{zi} \end{bmatrix} \qquad 4\text{-}24$$

Now, Expanding Inertia Matrix,

$$D(\theta) = J_1^T M_1 J_1 + J_2^T M_2 J_2 + J_3^T M_3 J_3 + J_4^T M_4 J_4 \qquad 4\text{-}25$$

Once the inertia matrix D is computed, the C matrix can be computed from the equation 4-23. Now, the components of $g(\theta)$ can be derived from:

$$g(\theta) = \frac{\partial PE}{\partial \theta_i} \qquad 4\text{-}26$$

Where the total potential energy is given by:

$$PE(\theta) = g\left(m_1 h_1(\theta) + m_2 h_2(\theta) + m_3 h_3(\theta) + m_4 h_4(\theta)\right) \qquad 4\text{-}27$$

Where,

$h_1(\theta) = r_1$

$h_2(\theta) = a_1 + r_2 s_2$

$h_3(\theta) = a_1 + a_2 s_2 + r_3 s_{23}$

$h_4(\theta) = a_1 + a_2 s_2 + a_3 s_{23} + r_4 s_{234}$

Finally,

$$g(\theta) = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} = \begin{bmatrix} \frac{\partial PE}{\partial \theta_1} \\ \frac{\partial PE}{\partial \theta_2} \\ \frac{\partial PE}{\partial \theta_3} \\ \frac{\partial PE}{\partial \theta_4} \end{bmatrix} = \begin{bmatrix} 0 \\ X \\ m_3 g r_3 c_{23} + m_4 g a_3 c_{23} + m_4 g r_4 c_{234} \\ m_4 g r_4 c_{234} \end{bmatrix} \quad 4\text{-}28$$

Where,

X= $m_2 g r_2 c_2 + m_3 g a_2 c_2 + m_3 g r_3 c_{23} + m_4 g a_2 c_2 + m_4 g a_3 c_{23} + m_4 g r_4 c_{234}$

In this way, all the matrices need to formulate the final dynamic equation for our robotic arm given by 4-22 is computed.

The equation 4-22 can be rewritten with a separate highest derivative as follows:

$$\ddot{\theta} = -D^{-1} C \dot{\theta} - D^{-1} g + D^{-1} \tau \quad 4\text{-}29$$

Now, this equation is suitable for further modification that enables us to obtain the standard linear-like state-space model for control design. It can be written as follows:

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0_{4x4} & I_{4x4} \\ 0_{4x4} & -D^{-1} C_{4x4} \end{bmatrix} \begin{bmatrix} \theta_{4x1} \\ \dot{\theta}_{4x1} \end{bmatrix} + \begin{bmatrix} 0_{4x4} \\ I_{4x4} \end{bmatrix} u_{4x1} \quad 4\text{-}30$$

$$[\theta_{4x1}] = [I_{4x4} \quad 0_{4x4}] \begin{bmatrix} \theta_{4x1} \\ \dot{\theta}_{4x1} \end{bmatrix} \quad 4\text{-}31$$

Which follows the form:

$$x_{k+1} = A x_k + B u_k \quad 4\text{-}32$$

$$z_k = C x_k \quad 4\text{-}33$$

Also, the calculation of joint torques τ from an auxiliary vector of control actions u is given by:

$$\tau = M u + g \quad 4\text{-}34$$

4.4 Model Predictive Controller

The project requires the precise and smooth manipulation of the robotic arm from one location to another for the seamless drink serving experience. For that the controller should account for the precise motion of each joint as well as multiple feedbacks and various constraints. So, MPC controller is used in the project.

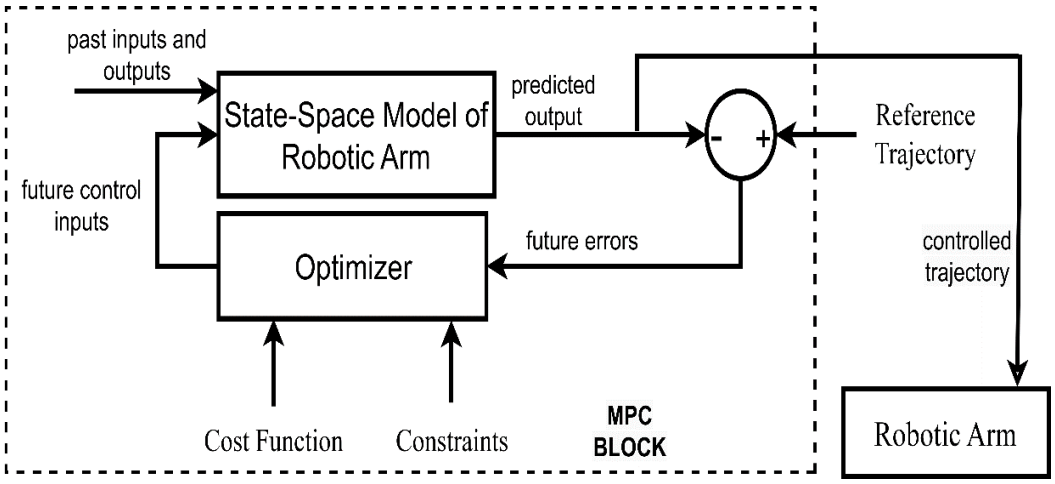


Figure 4-22: Basic Structure of MPC

MPC uses the model of the plant to make predictions about the future plant output behavior. It also uses the optimizer which ensures that the predicted output tracks the desired reference. The MPC controller needs to find the best predicted output so that it is closest to the reference. So, it simulates multiple future scenarios, also accounting for the given input/output constraints and the predicted output with the smallest cost function provides the optimal solution of the future inputs that is fed onto the model.

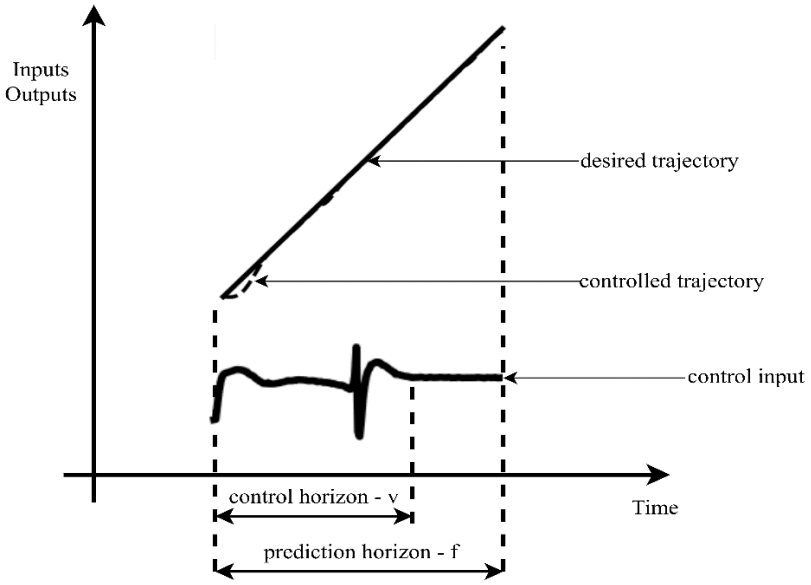


Figure 4-23: Working of MPC

Now, MPC algorithm applies only the first step of the predicted control inputs to the model at the current time stamp and based on that, the state of the model changes. Now, the same process repeats on the next time stamp over the prediction horizon.

4.4.1 Model Predictive Controller Design

Table 4-5: Nomenclature for Model Predictive Controller Design

Symbol	Remarks
x_k	State
u_k	Control Input
z_k	Output to Control
J_z	Cost function corresponding to tracking error
J_u	Cost function corresponding to change in inputs

Now, from the derived state-space model 6-31/6-32, x_k is the state, u_k is the control input, z_k is the output that we want to control, and A, B and C are the system matrices. The goal of the model predictive controller is to determine set of control inputs u that will make the output of the system z to follow the desired trajectory over the prediction horizon (f).

Let us assume that we are currently at time instant k and we want to make state and output predictions up to the time instant k+f. From 4-32, for time instant k+1,

$$x_{k+1} = Ax_k + Bu_k \quad 4\text{-}35$$

$$z_{k+1} = Cx_{k+1} = CAx_k + CBu_k \quad 4\text{-}36$$

For time instant k+2,

$$x_{k+2} = Ax_{k+1} + Bu_{k+1} = A^2x_k + ABu_k + Bu_{k+1} \quad 4\text{-}37$$

$$z_{k+2} = CA^2x_k + CABu_k + CBu_{k+1} \quad 4\text{-}38$$

By continuing this procedure, until the final step k+f, we obtain

$$z_k = Ox_k + Mu \quad 4\text{-}39$$

Where,

$$z = \begin{bmatrix} z_{k+1} \\ z_{k+2} \\ z_{k+3} \\ \dots \\ z_{k+f} \end{bmatrix}, u = \begin{bmatrix} u_k \\ u_{k+1} \\ u_{k+2} \\ \dots \\ u_{k+v-1} \end{bmatrix}, O = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \dots \\ CA^f \end{bmatrix}$$

$$M = \begin{bmatrix} CB & 0 & \dots & 0 \\ CAB & CB & \dots & 0 \\ CA^2B & CAB & \dots & 0 \\ \dots & \dots & \dots & \dots \\ CA^fB & CA^fB & \dots & C\overline{A}_{f,v}B \end{bmatrix}$$

4.4.2 MPC Problem Formulation

The goal is to track a desired output trajectory which is given by:

$$z^d = \begin{bmatrix} z^d_{k+1} \\ z^d_{k+2} \\ \dots \\ z^d_{k+f} \end{bmatrix} \quad 4\text{-}40$$

Now, the optimization problem is to determine the vector **u** minimizing the overall cost function given by:

$$\min_{u} (J_z + J_u) \qquad 4\text{-}41$$

Where,

$J_z = (z^d - z)^T W_4 (z^d - z)$

$J_u = (W_1 u)^T W_2 (W_1 u)$

User defined weight matrix to calculate input changes $(W_1) = \begin{bmatrix} I & 0 & 0 & ... & 0 \\ -I & I & 0 & ... & 0 \\ 0 & -I & I & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & I \end{bmatrix}$

User defined weight matrix to penalize tracking error $(W_4) = \begin{bmatrix} P_1 & 0 & ... & 0 \\ 0 & P_2 & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & 0 & P_f \end{bmatrix}$

User defined weight matrix to penalize input changes $(W_2) = \begin{bmatrix} Q_0 & 0 & ... & 0 \\ 0 & Q_1 & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & 0 & Q_{v-1} \end{bmatrix}$

Ultimately, to obtain the solution of model predictive control problem, The cost function described above should be minimized. It can be done that by using formulas for the derivative of scalar functions with respect to vectors. Let w be a vector and a be a constant vector. Also, let H be a constant symmetric matrix.

$$\frac{dw^T a}{dw} = \frac{da^T w}{dw} = a \qquad 4\text{-}42$$

$$\frac{dw^T H w}{dw} = 2Hw \qquad 4\text{-}43$$

Now, By using these two equations, the derivative of the cost function $(J_z + J_u)$ is defined as:

$$\frac{d(\mathrm{J_z} + J_u)}{du} = -2M^T W_4 s + 2M^T W_4 M u + 2W_3 u \quad \text{4-44}$$

To find the minimum of the cost function with respect to u:

$$\frac{d(J_z + J_u)}{du} = 0 \quad \text{4-45}$$

Finally, the solution of the model predictive control problem is obtained as:

$$\hat{u} = (M^T W_4 M + W_3)^{-1} M^T W_4 s \quad \text{4-46}$$

$$\hat{u} = \begin{bmatrix} \hat{u}_k \\ \hat{u}_{k+1} \\ \dots \\ \hat{u}_{k+v-1} \end{bmatrix} \quad \text{4-47}$$

In the figure below, the predicted trajectory can be seen after applying whole input of prediction horizon. So, In order to track the trajectory, the first entry,$\hat{u}_k$ is only applied to the system and repeat the whole process for time instant k+2.

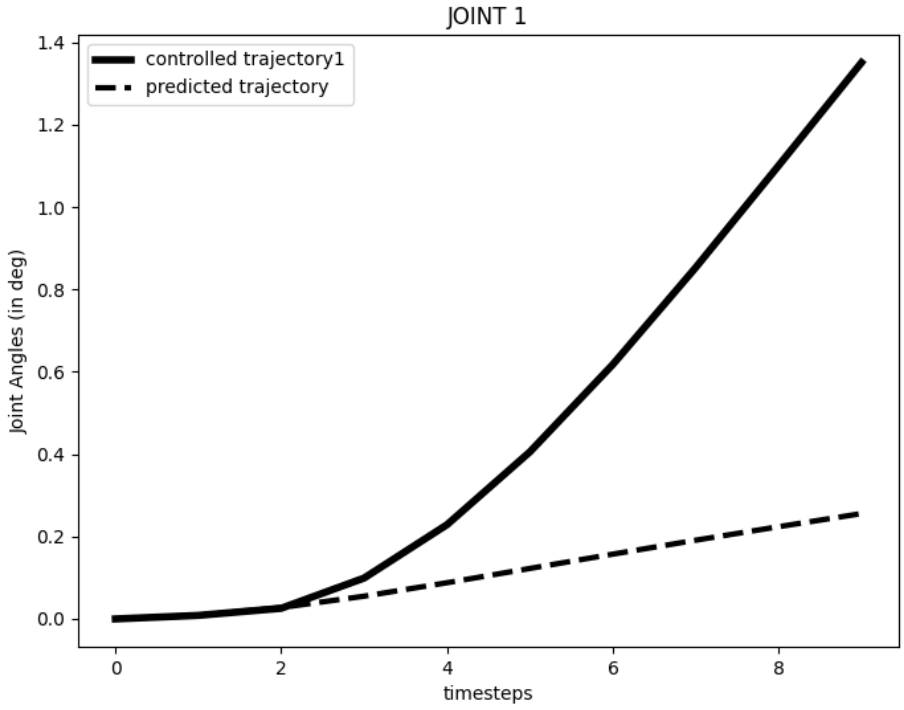


Figure 4-24: Controlled vs. Predicted Output

4.5 Speech Recognition System Architecture and Description

This section will explore the Squeezeformer model and its various modules with its layers in detail

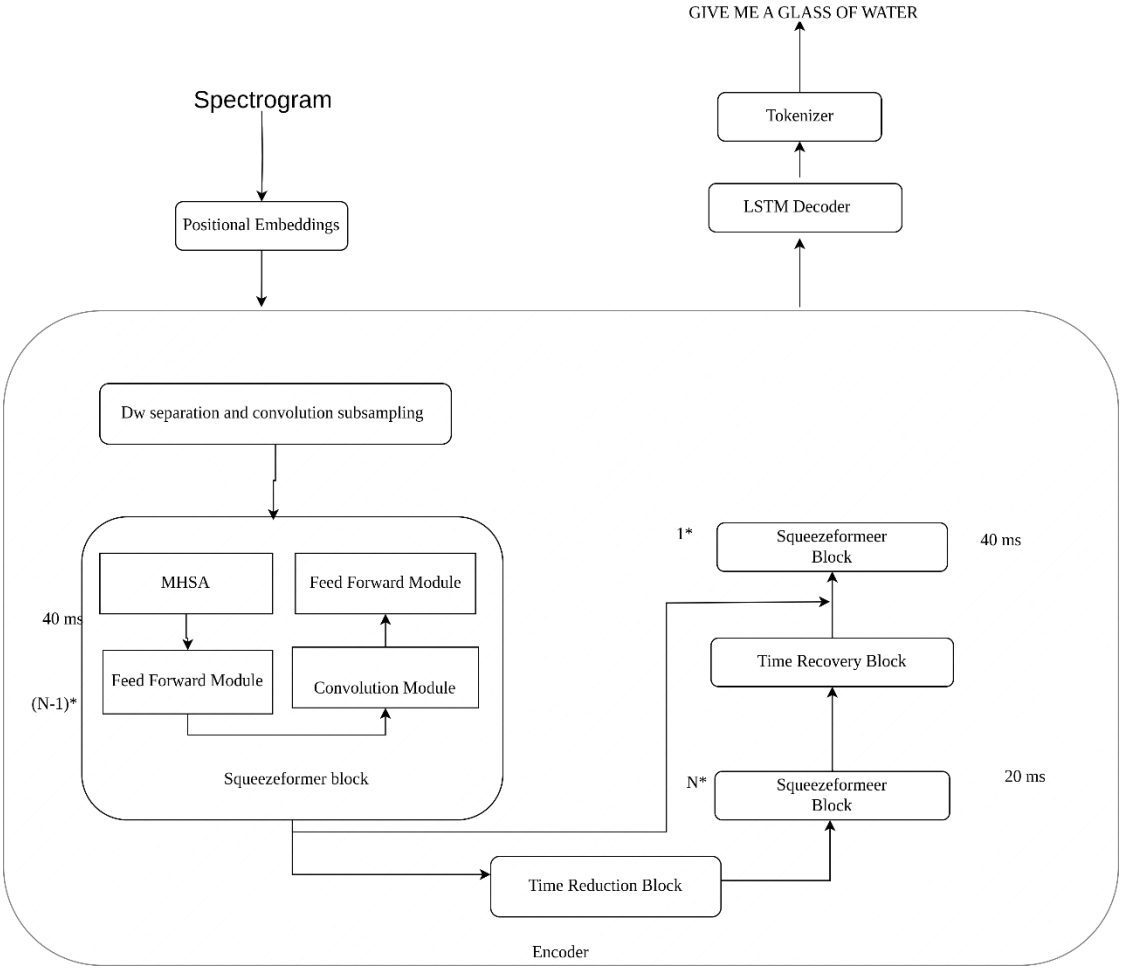


Figure 4-25: Speech Recognition System Architecture

Speech recognition system models are sophisticated algorithms used to convert spoken language into written text. Speech recognition system models often employ encoder-decoder architectures to convert spoken language into written text.

4.5.1 Encoder

The model encoder module takes raw audio input, such as spoken words or phrases, and processes it using transformer models to extract relevant acoustic features. These features capture the temporal patterns and spatial dependencies in the audio signal. The

output of the encoder is a transformed representation of the input audio, which is more abstract and informative. This transformed representation serves as the basis for further processing by the decoder module to generate textual representations of the spoken language. In essence, the encoder's role is to convert raw audio input into a structured format that can be easily interpreted by the decoder to produce accurate text output in the system

A. Positional Embedding

Positional encoding enhances transformer models by incorporating information about the sequential order of words in the input text. It takes tokenized words or elements as input and computes positional encodings using specific equations. These encodings are then added to the input embeddings, enriching them with positional information.

$$.PE(pos, 2i) = sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \quad \textit{4-48}$$

$$PE(pos, 2i+1) = cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \quad \textit{4-49}$$

Where 'pos' represents the position of the token in the sequence, 'i' denotes the dimension of the encoding, and‘d’ represents the model dimension. The sine function is applied to tokens at even indices, while the cosine function is applied to tokens at odd indices.

B. Depth Wise Separation and Convolution Subsampling

Depth-wise separation is a technique utilized in CNNs to reduce computational complexity. It divides the convolution operation into depth-wise and point-wise convolutions. In depth-wise convolution, each input channel is convolved with its filters, capturing local spatial information. Point-wise convolution, on the other hand, applies a 1x1 filter to the output of depth-wise convolution, capturing inter-channel interactions. This separation technique employs a kernel size of 3 (k=3) and stride of 2 (s=2), followed by a dropout layer.

Convolution subsampling down samples the spatial dimensions of feature maps in CNNs. It involves sliding a fixed-size window over the feature map and performing an aggregation operation within each window. This down sampling reduces spatial resolution, aiding in computational efficiency and higher-level feature extraction. For the Squeezeformer model, convolution with a kernel size of 3 (k=3) and stride of 2 (s=2) is performed, followed by a ReLU activation layer. These techniques collectively improve the efficiency and effectiveness of feature extraction within the model

C. Multi Headed Self Attention

This layer in the model converts input embeddings into query, key, and value vectors, allowing the model to compute attention scores and simultaneously focus on different parts of the sequence. This process significantly improves the model's ability to capture long-range dependencies and contextual information. Input involves a sequence of embeddings representing tokens or audio features, with the output comprising attended vectors enriched with contextual information. These attended vectors facilitate more effective processing by subsequent layers in the model.

$$\text{Dot\_Product}_{(i,j)} = \text{Q}_i^{\text{T}} * \text{K}_j \quad \text{4-50}$$

$$Scaled\_Dot\_Product_{(i,j)} = \frac{Dot\_Product_{(i,j)}}{\sqrt{d_k}} \quad \text{4-51}$$

$$Attention(Q, K, V) = softmax(Scaled\_dot\_Product_{(i,j)}) \times V \quad \text{4-52}$$

D. Feed Forward

The Feed Forward layer consists of several blocks designed to enhance the model's ability to capture complex patterns and improve generalization. It begins with a linear transformation, mapping the input dimension to a higher-dimensional space to capture intricate patterns. This is followed by the application of the Swish activation function, known for its effectiveness in deep neural networks. Dropout layers are then employed to prevent overfitting by randomly deactivating a portion of the output activations during training. Next, convolution operations are applied to introduce non-linear transformations and capture local spatial information. Batch normalization is utilized to stabilize and accelerate the learning process by normalizing the output of the

convolutional layers. The Swish activation function is again applied to introduce non-linearity to the output. Finally, another linear transformation is performed to reduce the dimensionality back to the original input dimension. Dropout is applied once more to further prevent overfitting. These operations collectively enhance the model's ability to learn intricate patterns and relationships within the data, ultimately improving its performance and generalization.

E. Time Reduction

The Time Reduction block integrates a DepthwiseConv2d layer and a Swish activation function to streamline processing time within the model. The DepthwiseConv2d layer, configured with stride (s) of 2 and no padding (p=0), conducts individual convolutions for each input channel. This approach minimizes required operations, boosting computational efficiency. Complementarily, the Swish activation function, combining linear and non-linear elements, elevates model performance while preserving computational speed.

F. Time Recovery

The Time Recovery Block operates by taking an input tensor and iteratively traversing a range that encompasses twice the size of the second dimension of the inputs. It stacks the elements of outputs along the second dimension, effectively restoring the input to its original resolution. This process ensures that the temporal information is recovered, enabling the model to maintain the integrity of the input sequence

4.5.2 Decoder

In the speech recognition system, the Decoder Block plays a vital role in converting encoded audio features into textual representations. This component employs specialized decoding techniques to ensure accurate transcription. Leveraging convolutional operations or other suitable methods, the Decoder Block captures temporal dependencies inherent in the audio data, facilitating the extraction of relevant features for text generation. It serves as a bridge between the encoded audio representations and the final textual output.

A. Convolution

The Convolutional Layer within the Squeezeformer encoder serves as a critical stage in the speech recognition system pipeline. Here, this layer takes the logits generated by the encoder and processes them to produce output probabilities. These probabilities represent the likelihood of different characters or tokens being present in the input audio.

II. Tokenizer

Tokenizer is responsible for converting numeric labels into their corresponding character representations. It takes a parameter label which is expected to be a NumPy array. If label is a 1-dimensional array, representing a single sentence, the method iterates through each label, excluding the end-of-sentence (eos_id) or blank (blank_id) tokens. It retrieves the symbol corresponding to each label from the dictionary and concatenates them to form a string representation of the sentence.

Table 4-6: Sample Data after Tokenization

Transcript	Tokens
‘HEY ARM GIVE ME A GLASS OF COFFEE'	'11 13 26 9 4 17 9 4 5 4 11 16 5 23 23 4 19 10 4 7 19 10 10 9 9'
‘HEY ARM GIVE ME A GLASS OF WATER'	'11 13 26 9 4 17 9 4 5 4 11 16 5 23 23 4 19 10 4 27 5 24 9 22 '

4.6 Computer Vision Module

This section incorporates the overall model explanation about the vision module used in the system for object detection and position estimation

4.6.1 YOLOv8 Architecture

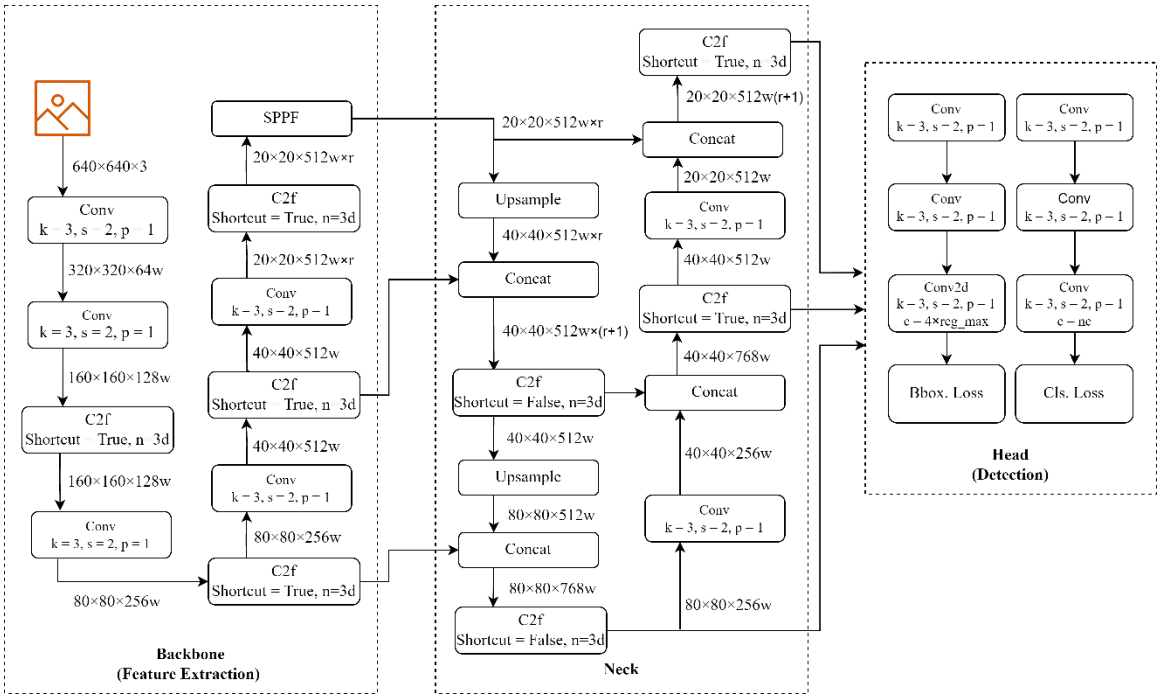


Figure 4-26: YOLOv8 Internal Architecture

A. Backbone

I. Convolutional Layer

The conv block is used in YOLOv8 for downsampling the feature maps and extracting higher-level features. It helps in reducing the spatial dimensions while increasing the receptive field of the filters, allowing the network to capture more global information from the input.

Kernel size (k): It represents the size of the convolutional filter or kernel. In this case, the kernel size is 3x3, meaning that the filter will have a receptive field of 3x3 pixels.

Stride (s): It defines the step size at which the convolutional filter moves across the

input image or feature map.

Padding (p): It determines the number of pixels added to the borders of the input image or feature map. Padding helps preserve spatial resolution and can be used to control the size of the output feature map. The formula used for upscaling the feature map is as follows:

Rounding off the dimensions ensures that the output feature map aligns correctly with subsequent layers and avoids fractional or partial pixel calculations.

II. Upscaling

In this model, upscaling is employed to augment the spatial dimensions of the feature maps, allowing for more detailed and comprehensive analysis.

III. C2f (Convolution to Fully Connect)

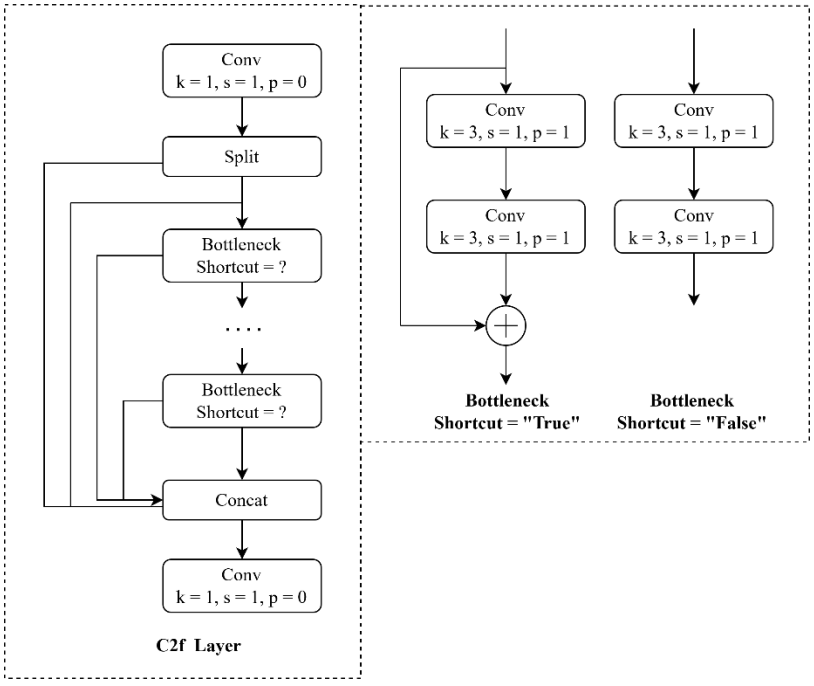


Figure 4-27: Convolution to Fully Connected Layer

The C2f (Convolution to Fully Connect) layer in the YOLO architecture plays an important role in increasing the network capacity without altering the spatial dimensions of the input feature maps.

When the "shortcut" parameter is set to "true", the C2f layer incorporates a shortcut connection. This means that a split is performed in the input feature maps, where one split is processed by the convolutional layers, and the other split remains untouched. The processed split and the untouched split are then concatenated together. The shortcut connection in the C2f layer allows for the preservation of high-resolution features from the input while also enabling the network to learn more abstract and higher-level representations through the convolutional layers.

IV. SPPF (Spatial Pyramid Pooling with Fuse)

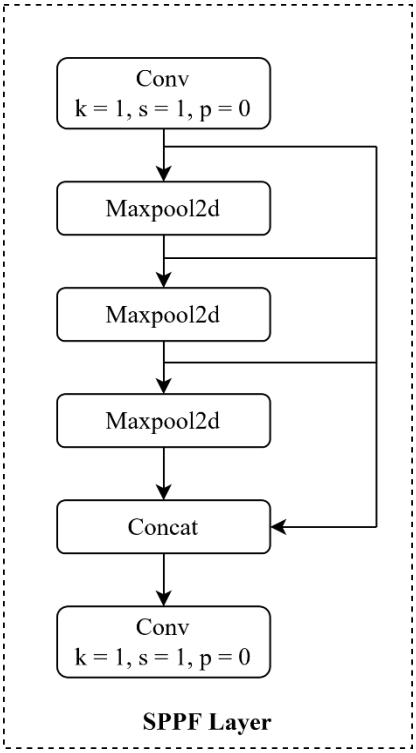


Figure 4-28: Spatial Pyramid Pooling with Fused Layer

The SPPF block consists of a series of MaxPool2d layers with different pooling sizes. Each MaxPool2d layer performs pooling operations on the input feature maps, resulting in feature maps with reduced spatial dimensions but preserved channel dimensions. The MaxPool2d operation involves dividing the input feature map into non-overlapping regions and taking the maximum value within each region. This pooling operation helps in extracting the most prominent features within each region, effectively reducing the spatial resolution.

B. Neck

The neck component of the YOLO architecture plays a crucial role in refining and enhancing the features extracted from the backbone. It consists of a series of upsampling, concatenation, C2f (Convolution to Fully Connected), and Conv layers.

The output is then passed through convolutional layers to further refine the features. These convolutional layers help to capture more localized patterns and fine-grained details. Finally, the output of the convolutional layers is concatenated with the previous C2f outputs, creating a hierarchical feature representation. This concatenation allows the network to combine features from different scales and levels of abstraction.

C. Head

The detection layer in YOLO architecture consists of two sets of convolutional layers followed by a Conv2d layer.

The first set of convolutional layers has a kernel size of (k=3, s=1, p=1), indicating a 3x3 spatial convolution operation with a stride of 1 and padding of 1. This operation helps capture spatial information from the input feature maps while maintaining the same spatial dimensions.

Following the first set of convolutional layers, there is a Conv2d layer with parameters (k=1, s=1, p=0, c=4*reg_max). This Conv2d layer performs a 1x1 convolution operation on the input feature maps. 'c' represents the number of output channels. In this case, the number of output channels is set to 4 times the maximum number of regression targets ('reg_max'). This Conv2d layer is responsible for predicting the bounding box coordinates and regression values associated with the detected objects.

Similarly, the second set of convolutional layers also has a kernel size of (k=3, s=1, p=1), indicating a 3x3 spatial convolution operation with a stride of 1 and padding of 1. It is followed by another Conv2d layer with parameters (k=1, s=1, p=0, c=nc), where 'nc' represents the number of classes or categories that the network is trained to detect. This Conv2d layer performs a 1x1 convolution operation and outputs a tensor with 'nc' channels, representing the predicted class probabilities for each bounding box.

4.7 Dataset Collection Method

The section encompasses the comprehensive data collection methodology for both speech recognition and object detection methods. It encompasses various stages including pre-tuning, fine-tuning, and other significant aspects involved in dataset collection.

4.7.1 Sound Data Collection

A. Pre-training

The Squeezeformer speech recognition model was trained on the NeMo ASRSET, a composite dataset comprising several thousand hours of English speech. This dataset includes the LibriSpeech corpus, consisting of 960 hours of English speech.

B. Fine-tuning

The model was finetuned using a custom dataset collected from Thapathali Campus and various school consisting of participants of varying age group.

C. Data Collection Method

The data collection process involved recording spoken utterances from students and teachers in a carefully controlled environment.

The components of data collection process are:

I. Participant

Participants of diverse age groups and roles within the educational sphere were included for finetuning dataset preparation. The participant pool consisted of students ranging from high school level to postgraduate studies, as well as teachers and campus staff ensuring representation across various age groups. The table below summarizes the participant distribution.

Table 6-5: Table of Participants

Location	Participants
Thapathali Campus	109
Nepal Mega School	30
Texas International School	71

II. Environment Setup

The recordings were primarily carried out in isolated rooms to minimize environmental noise and ensure optimal conditions for a high-quality dataset. Additionally, data collection took place in various outdoor and makeshift locations, such as park benches, further challenging the model's robustness to noise and diverse acoustic conditions. This approach aimed to strike a balance between controlled data collection and improvised environments, potentially enhancing the model's ability to generalize across different deployment scenarios.

III. Devices

The data recording utilized the Fantech MCX01 Leviosa, a professional condenser microphone with a cardioid pickup pattern for focused audio capture via USB interface. The microphone was positioned directly Infront of speaker to optimize speech clarity. The microphone performed well in noisy environments and captured high-fidelity audio.The data was captured using a personal computer or mobile phones due to their convenience in hosting the recording web application. Following capture, the raw data was archived on the PC for future processing and reference.

4.7.2 Object Detection Dataset Collection Method

A. Environment and its Objects for Data Collection

The data for the object detection model is gathered directly from the system environment itself. In this project, the system environment comprises the tabletop, encompassing various objects such as drinking cups, obstacles, dispensers, and the

robotic arm. The primary focus lies on tracking drinking glasses and obstacles, especially since their positions are dynamic and crucial for operating the robotic arm effectively. Images are captured at different instances to ensure the model's capability in handling various scenarios effectively.

I. Simulation Environment for Data Collection

The environment within CoppeliaSim is utilized to capture data for the detection model. Various objects are screenshotted from the environment, with images taken under different light intensities to enhance the model's compatibility across diverse lighting conditions.

The objects designated for simulation in CoppeliaSim are modeled in Fusion 360. These objects serve as the dataset for simulation purposes.

II. Physical Environment for Data Collection

In order for the model to function within a physical system, the datasets must also manifest as physical objects.

The physical environment of the project encompasses the tabletop along with all physical objects such as drinking glasses, obstacles, the robotic arm, and the dispenser. To collect the dataset, pictures are captured using camera devices. For this project, both a smartphone camera and a webcam integrated into the system are utilized to capture the image datasets.

B. Camera Parameters

These parameters define the specifications of the camera employed for capturing images, which constitute the dataset for the object detection model across both simulation and physical systems.

Table 4-7: Camera Parameters

Parameters	Value(Simulation)	Value(Physical)
Field of View	75°	60°
Resolution	356×356	1280×720

4.8 Data Augmentation Method

After the datasets are collected, they undergo augmentation for various purposes. This involves implementing several augmentation techniques tailored to object detection and speech recognition models.

4.8.1 Sound Augmentation Methods

In audio recognition tasks, data augmentation techniques play a vital role in improving the robustness and performance of the speech recognition model.

Algorithm: Audio Augmentation pipeline

Require: input_audio_files: a list of *N* input audio paths

1. **Loop** *N* times
2: **For** each input_audio_file in input_audio_path:
3: Load the input audio file
4: Apply various augmentation techniques:
5: 1. Apply gaussian noise
6: 2. Adjust pitch
7: 3. Time shift
8: 4. Add random gain
9: Save the augmented audio files
10: **Return** augmented files

Here are some commonly used transformations for data augmentation in audio recognition:

A. Gaussian Noise

Gaussian noise is a random signal that follows a Gaussian or normal distribution. By adding Gaussian noise to the dataset, the model becomes more robust to noise in real-world scenarios. The noise is calculated randomly and then added to the original data points.

$$\left[f(x;\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} exp\left[\frac{-1}{2}\left(\frac{x-\mu}{\sigma}\right)\right)^2\right] \quad \text{4-53}$$

B. Time-Shifting

Time-shifting involves shifting an audio signal along the time axis, either to the left or right, by a specified time value. Shifting the signal to the left results in an advancement, while shifting it to the right introduces a delay. Mathematically, this operation can be represented by the equation y(t) = x(t), where y(t) represents the shifted output signal, x(t) represents the original input signal, and Δt denotes the amount of time shift or delay. By subtracting or adding Δt from the time variable t, it can effectively shift the signal to the right or left introducing a delay.

$$y(t) = x(t + \Delta t) \quad \text{4-54}$$

$$y(t) = x(t - \Delta t) \quad \text{4-55}$$

C. Pitch Shifting

Pitch shifting is a transformation that changes the pitch of an audio signal without altering its speed. It can be used to simulate variations in the frequency content of the data. Raising or lowering the pitch can create new data points with different tonal characteristics.

$$y(t) = x(t/\alpha) \quad \text{4-56}$$

In this equation, the time variable “t” id divided by the pitch shifting factor α, effectively changing the playback speed of the signal. A value of α greater than 1 increases the pitch (higher frequency) of the output signal, while a value less than 1

decreases the pitch (lower frequency).

D. Random Gain

This scaling factor can be uniformly distributed or follow a specific probability distribution, depending on the desired characteristics of the augmentation. For example, a uniform distribution may be suitable when seeking random but equally probable gains, while a Gaussian distribution may mimic the natural variations in audio volumes more accurately.

$$Y(t) = X(t) * G \quad \text{4-57}$$

Where Y (t) represents the output signal at a specific time t, X (t) represents the input signal at the same time t, G represents the random gain applied to the input signal. By incorporating random gain into the audio processing pipeline, models can learn to be invariant to changes in volume, leading to improved generalization and performance on diverse audio data.

E. Time Masking

The purpose of time masking is to introduce temporal variations and modify the Spectro-temporal characteristics of the signal. This is achieved by applying a binary time mask, represented by M (t), to the input signal X (t).

$$Y(t) = X(t) * M(t) \quad \text{4-58}$$

The mask selectively allows certain segments to pass through while blocking or attenuating others.

F. Frequency Masking

Frequency masking is a technique used in audio signal processing to modify the spectral content of a signal by selectively attenuating or suppressing specific frequency components. Represented by the equation

$$Y(f) = X(f) * M(f) \quad \text{4-59}$$

Where Y (f) is the masked output signal, X (f) is the input signal, and M (f) is the binary frequency mask, frequency masking allows for targeted manipulation of spectral features. By applying the binary mask, specific frequency regions can be blocked while allowing others to pass through, resulting in modified spectral characteristics. This technique finds applications in tasks such as noise reduction, speech enhancement, and audio synthesis, enabling precise control over the spectral content of the signal to achieve desired audio processing goals.

4.8.2 Vision Data Augmentation

Photometric augmentations are excluded due to varied lighting conditions in the datasets. Instead, geometric augmentations are. Some geometric augmentation performed while trainingYOLOv8 models are.

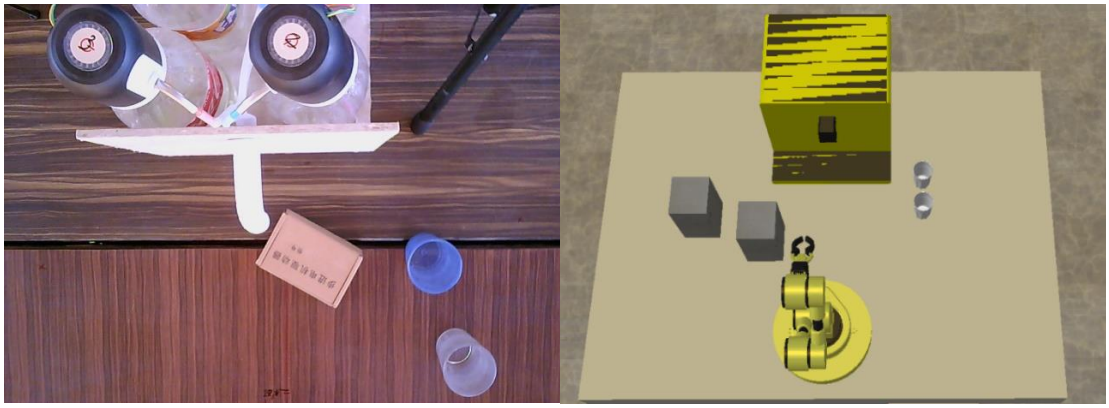


Figure 4-29: Original Image (Left-Physical and Right-Simulation)

Flip: Flipping can increase the diversity of the dataset and help the model learn to recognize objects from different perspectives. The transformation matrix for flipping vertically is

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{4-60}$$

Algorithm: Flip Image

Require: Image, flipAxis (horizontal or vertical)

1: if flipAxis = horizontal **then**

2: **for** each row in Image **do**

3: reverse the order of pixels in the row

4: **end for**

5: else if flipAxis = vertical **then**

6: **for** each column in Image **do**

7: reverse the order of pixels in the column

8: **end for**

9: end if

10: return Flipped Image

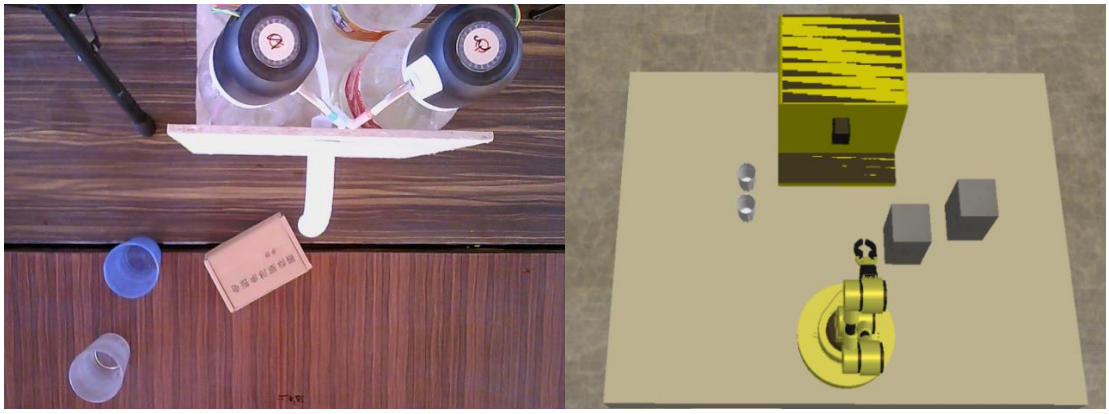


Figure 4-30: Flipped Image (Left-Physical and Right-Simulation)

Rotate: It helps the model handle objects in various orientations. The transformation matrix for rotation depends on the angle and can be represented as

$$\begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad 4\text{-}61$$

Where θ is the rotation angle in radians.

Algorithm: Rotate Image

Require: Image, rotationAngle

1: Compute rotationMatrix for rotationAngle

2: for each pixel in Image **do**

3: Compute new coordinates of the pixel after rotation using rotationMatrix

4: Interpolate the pixel value from the original image based on the new coordinates

5: Assign the interpolated pixel value to the corresponding position

6: end for

7: return Rotated Image

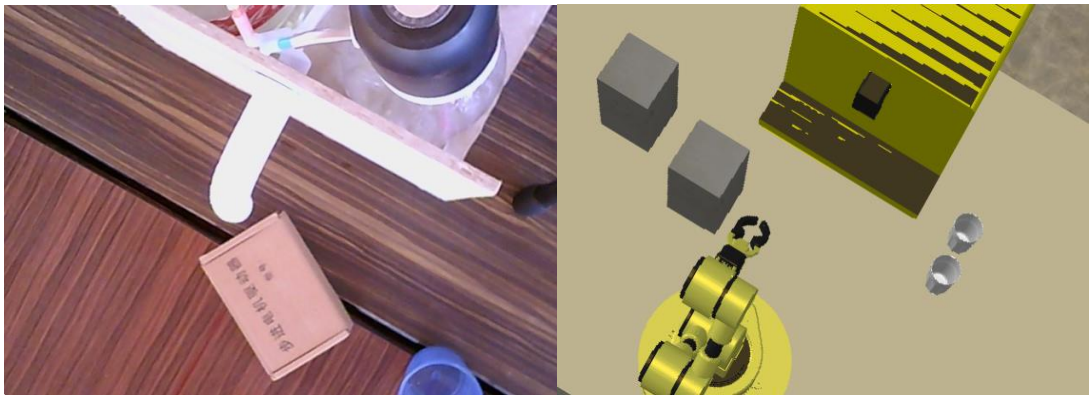


Figure 4-31: 30 Degree Rotation (Left-Physical and Right-Simulation)

4.9 Data Preprocessing Pipeline

Before training the model, it is crucial to preprocess the collected datasets. The sequence of dataset preprocessing for both object detection and speech recognition methods typically involves the following steps

4.9.1 Audio Preprocessing Methods

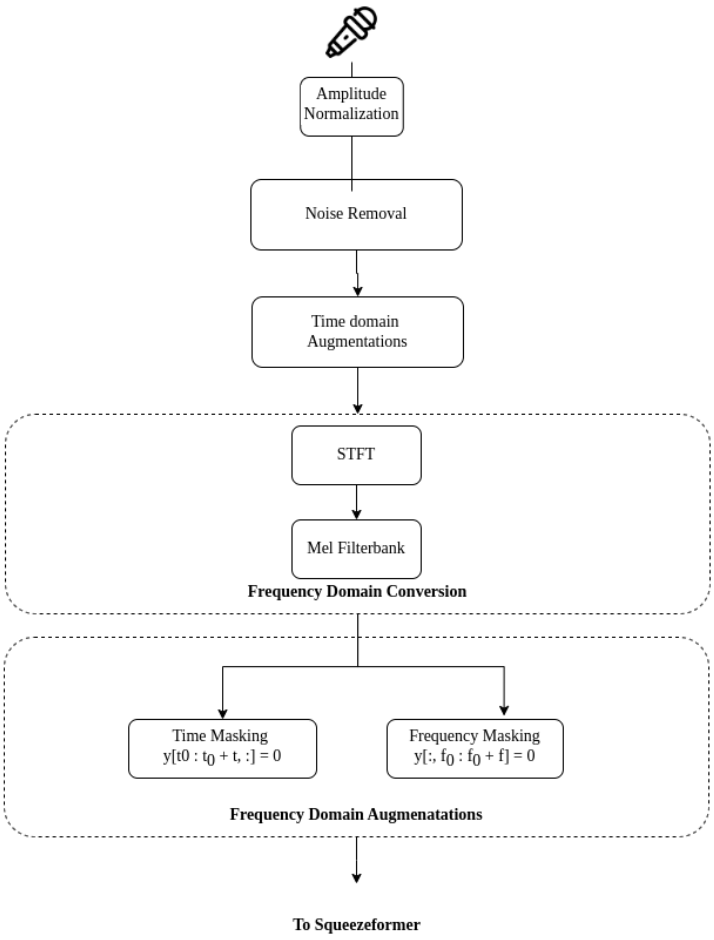


Figure 32: Data Preprocessing Architecture

Preprocessing techniques are applied to the raw audio data to enhance its quality and extract relevant features. Which included amplitude normalizing to avoid distortion across samples, clipping the silent ends to remove redundancy, and removing noise through band pass filter for better overall quality of sound sample.

A. Amplitude Normalization

It involves adjusting the volume level of an audio signal to reach a desired target level or dynamic range. The purpose of amplitude normalization is to eliminate variations in loudness, providing a balanced and enjoyable listening experience without altering the relative volume balance between different audio elements.

B. Background Noise Removal

Noise removal techniques are employed in speech recognition systems to enhance the quality of the input audio by reducing unwanted noise, thereby improving the model's ability to recognize and understand speech.

C. Frequency Domain Conversion

This transformation allows us to analyze and manipulate the spectral content of the signal, enabling tasks such as filtering, equalization, and spectral modification. The process of frequency domain conversion typically involves the Short-Time Fourier Transform (STFT), Mel filter bank, and Mel spectrogram.

I. STFT (Short-Time Fourier Transform)

It decomposes the signal into a series of time-dependent frequency components. This is achieved by dividing the signal into short frames of duration T and applying a windowing function to each frame. Commonly used window functions include Rectangular, Hamming, Hanning, Blackman, and Kaiser. The windowed frames are then transformed into the frequency domain using the Fourier Transform, resulting in a complex-valued representation with time index and discrete frequency bins.

$$STFT\{x(t)\}(\tau,\omega) \equiv \int_{\infty}^{-\infty} x(t)\,\omega(t-\tau) - i\omega t dt \qquad 4\text{-}62$$

$$c_i = \sum_{n=1}^{Nf} S_n \cos\left[i(n-0.5)\left(\frac{\pi}{Nf}\right)\right], i = 1,2\ldots,L \qquad 4\text{-}63$$

By computing the magnitude spectrogram, the time-varying spectral content of the audio signal can be visualized and further analyzed.

Table 4-8: STFT Parameters

Parameters	Values	No of Samples
Sample rate	16000 Hz	1600
Frame length	20 ms	320
Frame shift	10 ms	160

II. Windowing Function

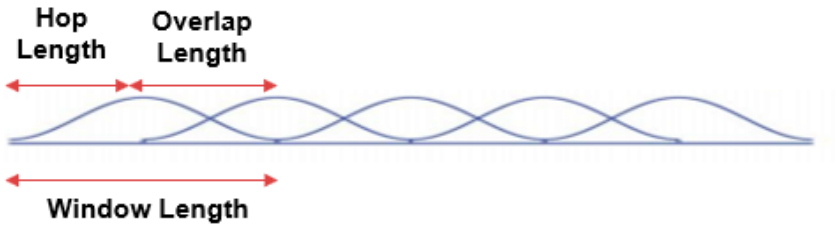


Figure 4-32: Windowing of a Signal

A windowing function, commonly used in signal processing, modifies finite-length signals or data segments to minimize spectral leakage and artifacts during analysis, particularly or spectral analysis. These functions, including types like Hamming, Hanning, and Blackman, taper signal values towards zero at the edges of the window, influencing how the signal is weighted across its duration. Choosing a windowing function depends on factors such as desired frequency resolution and trade-offs between main lobe width and side lobe levels, with applications spanning audio processing, vibration analysis, and spectrum analysis in various fields.

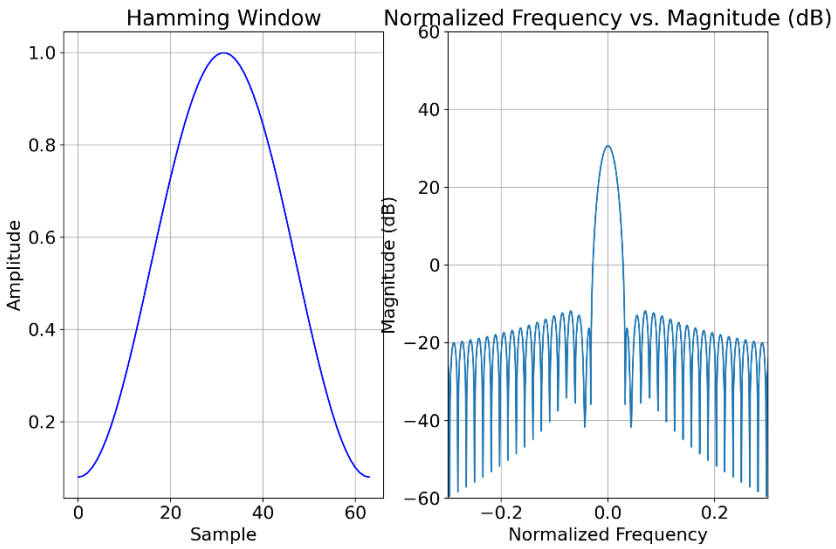


Figure 4-33: Hamming Window Plots

Table 4-9: Window Parameter Table

Parameters	Theoretical	Observed
Peak to peak side lobe amplitude (Relative, dB)	-41	-33.96
Width of main lobe (Frequency Bins)	0.0630	29
Roll off factor (dB/octave)	-18	-55.7

III. Mel Filter Bank

Mel filter bank is applied to the magnitude spectrogram. The filter bank divides the frequency axis into Mel-scale bins, which are perceptually relevant frequency regions. This process emphasizes the frequency regions that are more important for human perception.

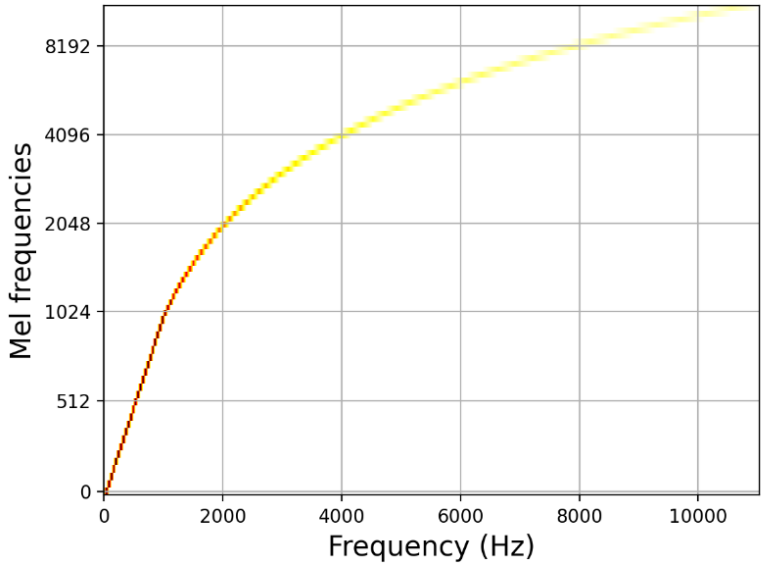


Figure 4-34: Mel Frequency vs. Frequency

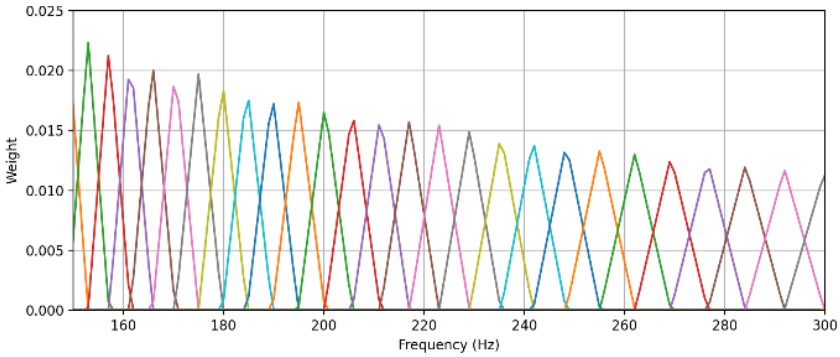


Figure 4-35: Mel Filter Banks for Frequency Conversion

IV. Mel Spectrogram

To refine the representation, the logarithm of the filter bank energies extracted from the Mel Spectrogram is computed to align with the logarithmic scale of human perception. These log filter bank energies effectively capture the intensity of frequency bands, mirroring human hearing characteristics. Mel Spectrograms provide a condensed yet informative representation of audio signals, emphasizing perceptually relevant frequency bands.

4.9.2 Object Detection Model Data Preprocessing

A. Annotation of Images

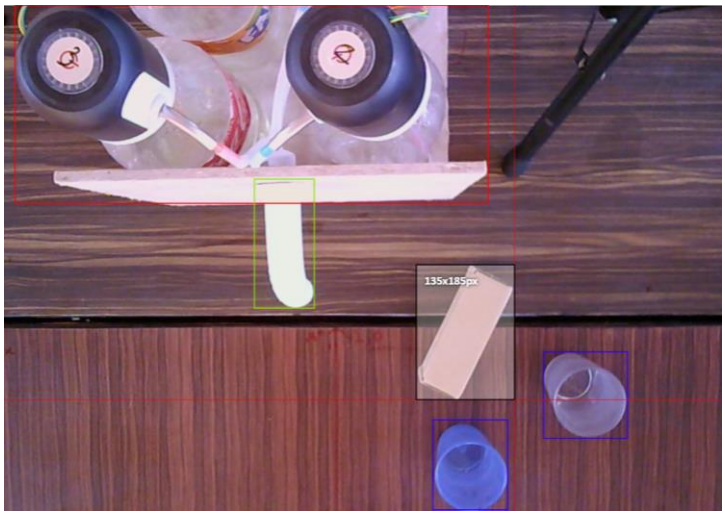


Figure 4-36: Annotating Images using CVAT

Table 4-10: Normalized Bounding Box Parameters for Above Annotations

Class Index	Class Name	Center Coordinates		Width	Height
		x	y		
1	Glass	0.39	0.18	0.50	0.37
1	Glass	0.75	0.74	0.09	0.16
2	Dispenser	0.63	0.87	0.08	0.17
3	Nozzle	0.62	0.61	0.10	0.25
4	Obstacle	0.43	0.45	0.06	0.24

The annotation process using CVAT involves labeling objects within image datasets collected from both simulation and physical environments. With CVAT, annotators create bounding boxes around objects of interest within each image, defining their position and extent. Additionally, assign appropriate class labels to each bounding box, indicating the type of object it represents (e.g., "glass," "obstacle," "fallencup"). Once bounding boxes are drawn and labeled, CVAT generates annotation files in Darknet format. Darknet format consist of two components:

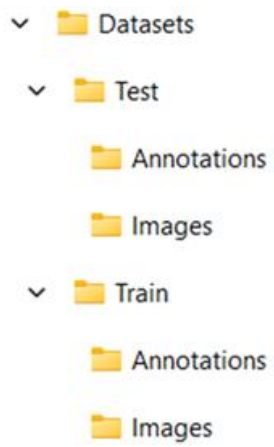


Figure 4-37: Darknet Format

Image Files: Each image in the dataset is paired with an annotation file containing the same name.

Annotation Files (TXT): These files contain information about the bounding boxes and class labels for objects within the corresponding images. Each line in the TXT file represents a bounding box, specifying its coordinates (e.g., top-left corner coordinates, width, height) and the associated class label.

5. IMPLEMENTATION DETAILS

This section delves into the practical implementation of the project, including a description of the virtual and physical environments used, the design procedures for the robotic arm and dispenser, the control mechanisms for motors and actuators, and the integration of vision and speech recognition modules.

5.1 Virtual Environment

The virtual environment consists of 4-DOF robotic arm as well as dispenser which is imported from Fusion 360. Except robotic arm and dispenser, everything is built-in in the CoppeliaSim itself.

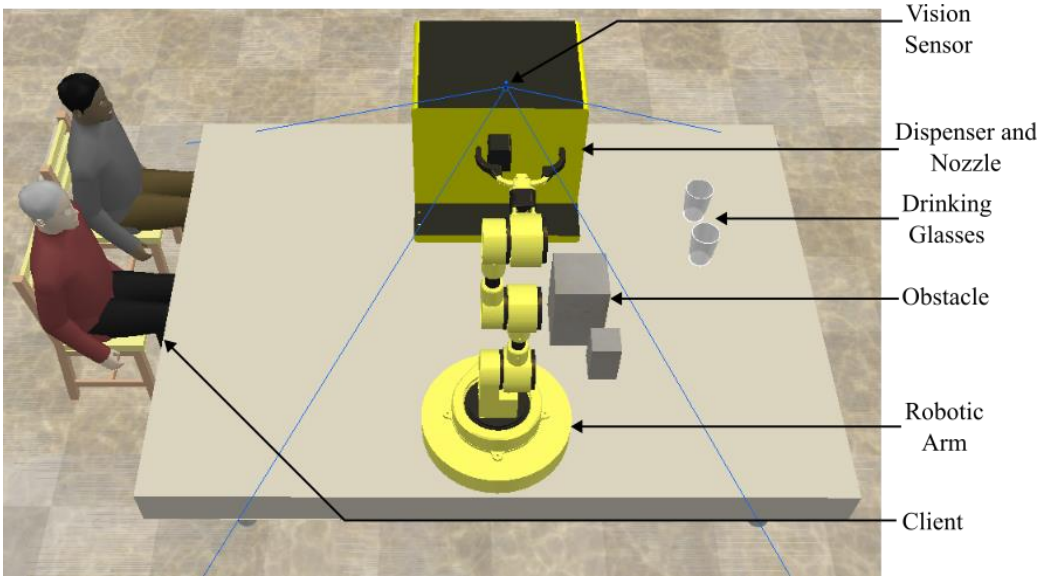


Figure 5-1: Virtual Environment

5.1.1 Four-Degree of Freedom Robotic Arm

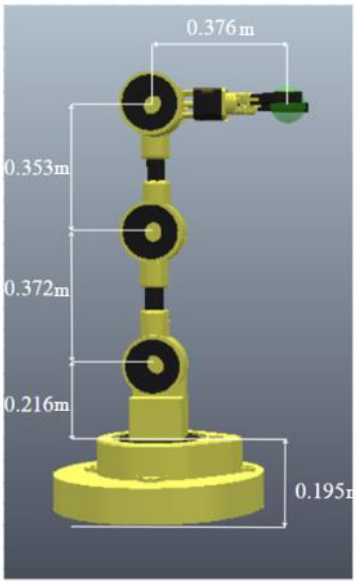


Figure 5-2: Kinematic Model

This is the 4-DOF arm in the simulation environment. It contains four revolute joints which in real is actuated by the cyclodial drive powered stepper motors and five links that connects to the joints. It consists of the two finger gripper that grips the drink glass which can be powered by servo motors with high torque.

The dimensions of each links is shown in the figure. For example the length of base link is 0.195m. Whereas the size of the whole robot is shown using the bounding box.

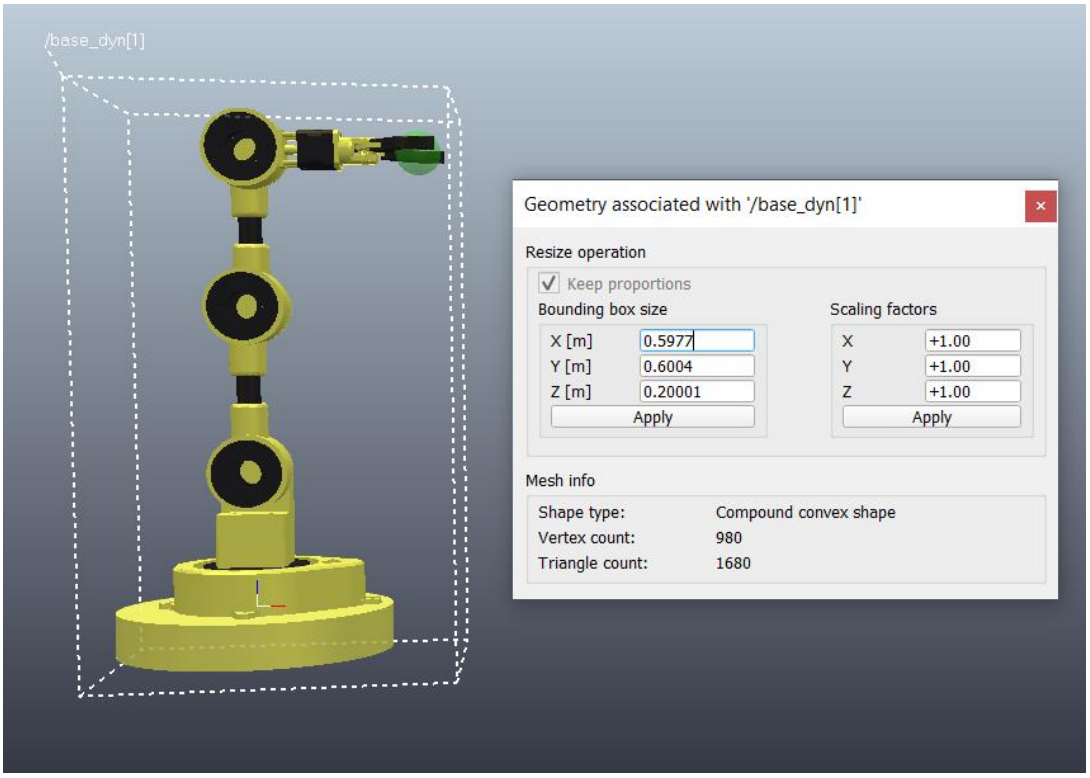


Figure 5-3: Robotic Arm Size

5.1.2 Dispenser

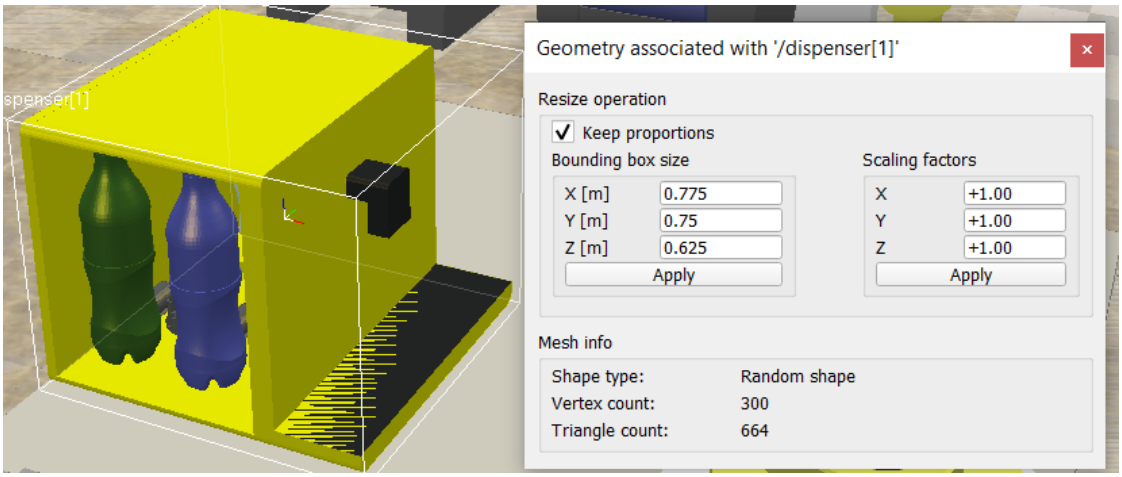


Figure 5-4: Dispenser Size

It is the component that pours the drink to the cup when the robot places empty cups on it. It consists of 4 different drinks i.e., coffee, sprite, orange juice and water. It houses motor pump, motor drivers, microcontrollers and interface to the main controller i.e., raspberry pi.

But in the simulation environment it only acts as a place holder for the cups. It doesn't have functionalities of the motor drivers, motors, microcontrollers etc. Simulation of fluid poured in the cup is also out of the scope of the project. Only thing that is done is change the color of the cups once after certain interval of time of placing the empty cups to the dispenser internally from code.

The size of dispensor is represented by the size of bounding box i.e. $0.775 \times 0.75 \times 0.625 \times m^3$

5.1.3 CoppeliaSim In-built Components

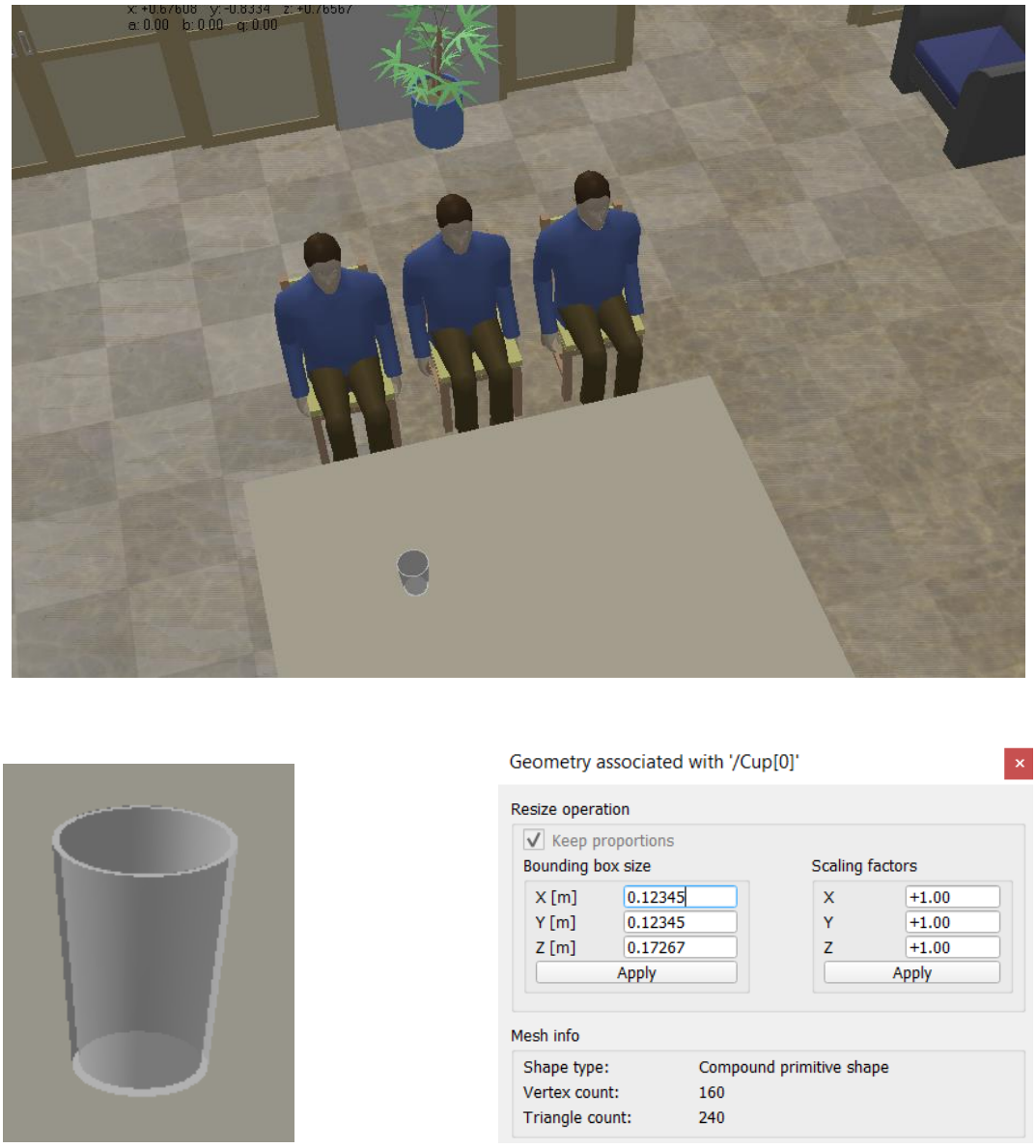


Figure 5-5: Customers and Cups

Customers, cups, chairs, sofa, decoration, floor, sliding door, walls or basically the environmental components are built using the inbuilt models present in the coppeliaSim. In other words, except robot arm and dispenser all the other components are used from the built-in library models. The size of the cup is also denoted by the bounding box size as shown in the figure.

5.1.4 Vison Sensor

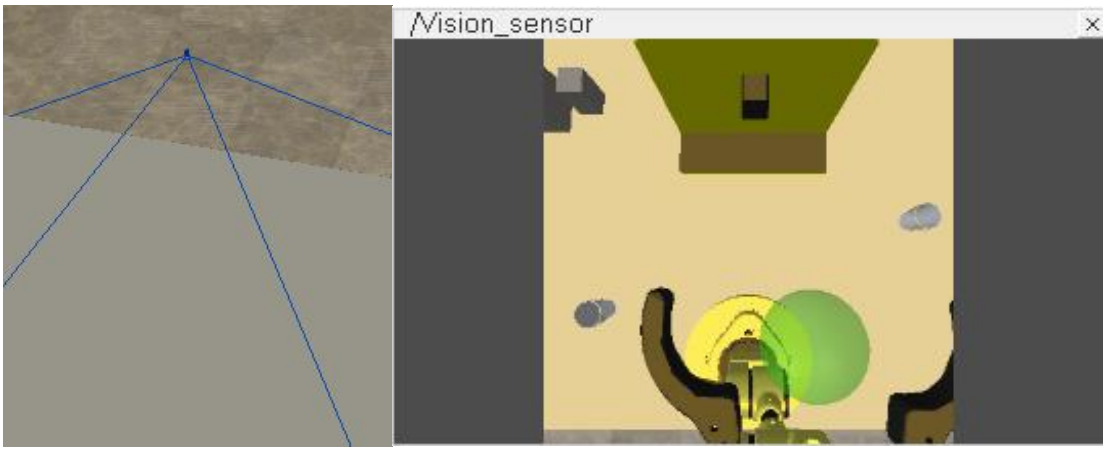


Figure 5-6: Vision Sensor

Vision sensors are also already present in the coppeliaSim environment. The vision sensor is represensted as a cuboid. The blue lines in the figure represents the perspective view of the vision sensor and the camera stand, although seems redundant as the vision sensor doesn't require any, it is there to estimate camera stand position and length since in real world it need something to support the actual camera Sensor view can be linked with a display window without any hassel. The sensor is placed directly on the top in such a way that it can see dispenser cups and client side space.

5.1.5 Python and CoppeliaSim (ZeroMQ)

ZeroMQ provides a socket-based API that abstracts the complexities of low-level network programming. It acts as a mediator for python and CoppeliaSim communication for this project. This API is similar to the traditional Berkeley Sockets API (used for IPC) commonly used for network communication in programming languages like C and Python. In the context of ZeroMQ, a "socket" is a communication endpoint that allows data to be sent and received between applications or processes.

ZeroMQ sockets provide an interface for various messaging patterns and transport protocols, and they are designed to be more flexible and feature-rich compared to traditional sockets. The socket-based API of ZeroMQ abstracts away the underlying transport mechanisms, making it easier for developers to focus on the communication patterns and logic of their distributed applications rather than dealing with network complexities.

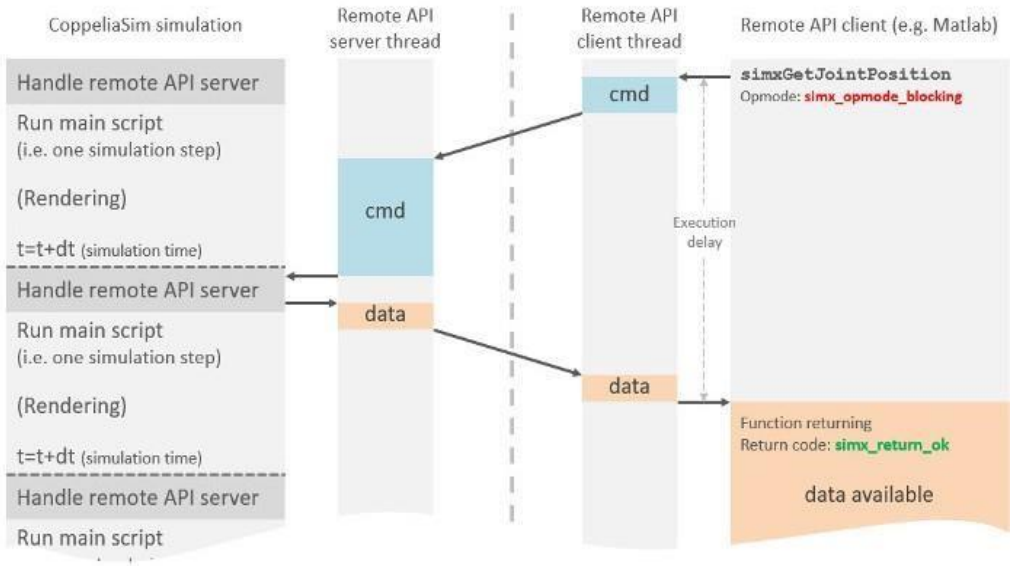


Figure 7-14: Client Server Interaction

After successful installation client side and for server side are complete. It has to include simAddOnZMQ remote API server.lua file in addOns folder in the CoppeliaSim installed directory. If the procedure is successful, it can see zeroMQ API running in the background of simulation environment.

5.1.6 Some Important Parameters

Table 5-1: Joint Parameters

Parameter Name	Value
Max torque	1000 N/m
Max angular speed	65 deg/s
Joint1 range of motion	360 deg
Joint2 range of motion	-60-0, 0-60 deg
Joint3 range of motion	-60-0, 0-60 deg
Joint4 range of motion	-70-0, 0-70 deg

The maximum torque is very to ensure that joint can apply enough torque at any conditions. Problems like jittering of motion and unable to reach the destination might occur when torque isn't enough. With experimentation 65 deg/s angular speed was appropriate. The range of motion of joint 3 is 180 due to the reasons specified in section

Table 5-2: Material Property

Density	1240 kg/m^3
RGB for Dark	[0.15,0.15,0.16]
RGB for Yellow	[1,1,0.4]

The uniform density 1240 is the density of the 3d printed PLA material. The bot consists of two colors yellow and black whose RGB values in the simulations are present in the table.

5.2 Physical Environment Setup

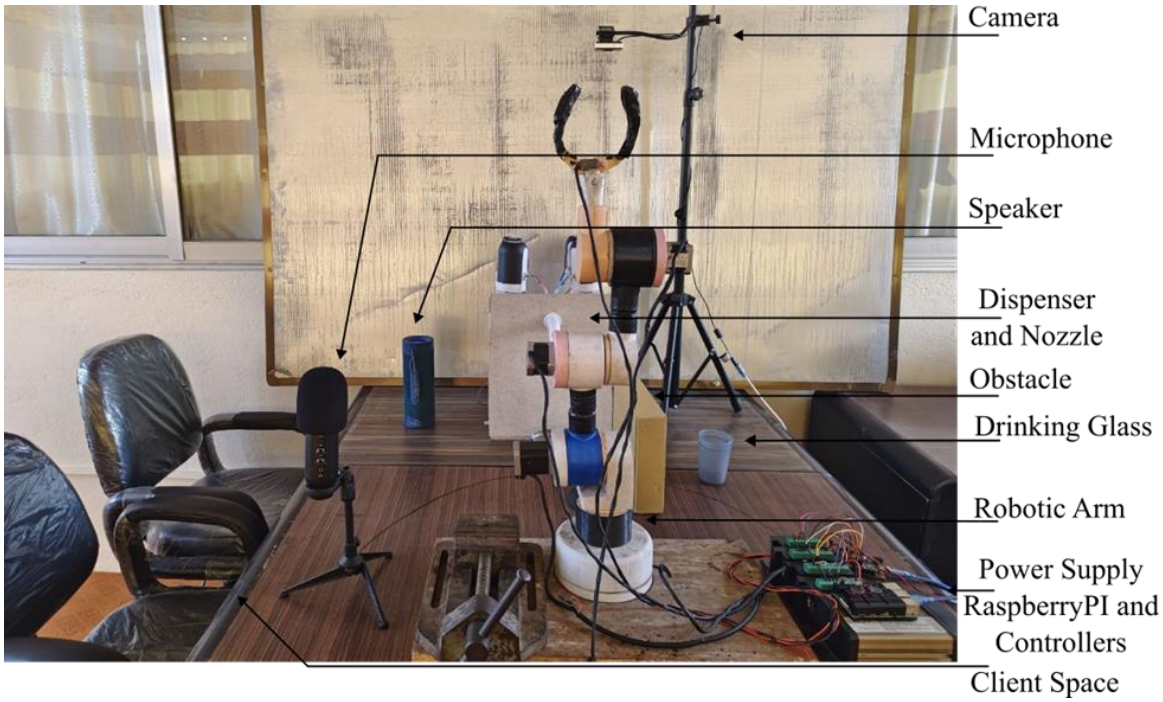


Figure 5-7: Complete Project Setup in Physical Environment

The physical environment setup comprises a tabletop hosting various components including dispensers, a camera, cups, and a robotic arm. Additionally, the setup includes a speaker and a microphone, which were absent in the simulation setup.

5.3 Design Procedure

5.3.1 Import CAD into CoppeliaSim

From fusion 360, single mesh in CoppeliaSim can be exported and then define joints links after it is imported in CoppeliaSim or it can directly export URDF file.

A. Mesh File

Before exporting the mesh file to the CoppeliaSim the robot arm must be simplified in fusion 360 as it contains various components when included in simulation, will result unnecessary overload.

When mesh file is imported from fusion 360 only single mesh file is present which

must be divided into individual parts using Shape grouping/Merging—divide. After the procedure, simulator generates all the possible parts on the scene hierarchy.

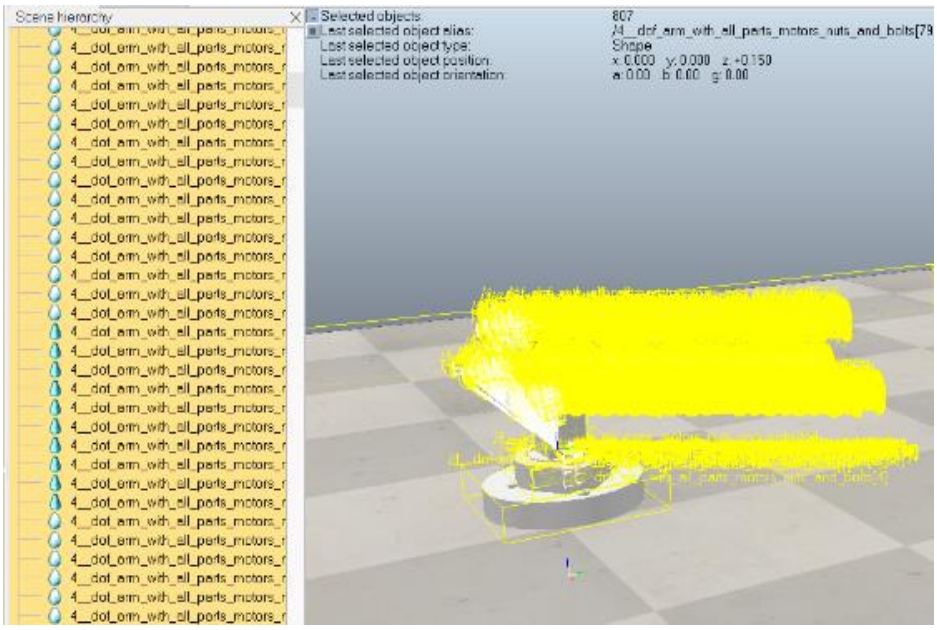


Figure 5-8: Mesh Before Simplification

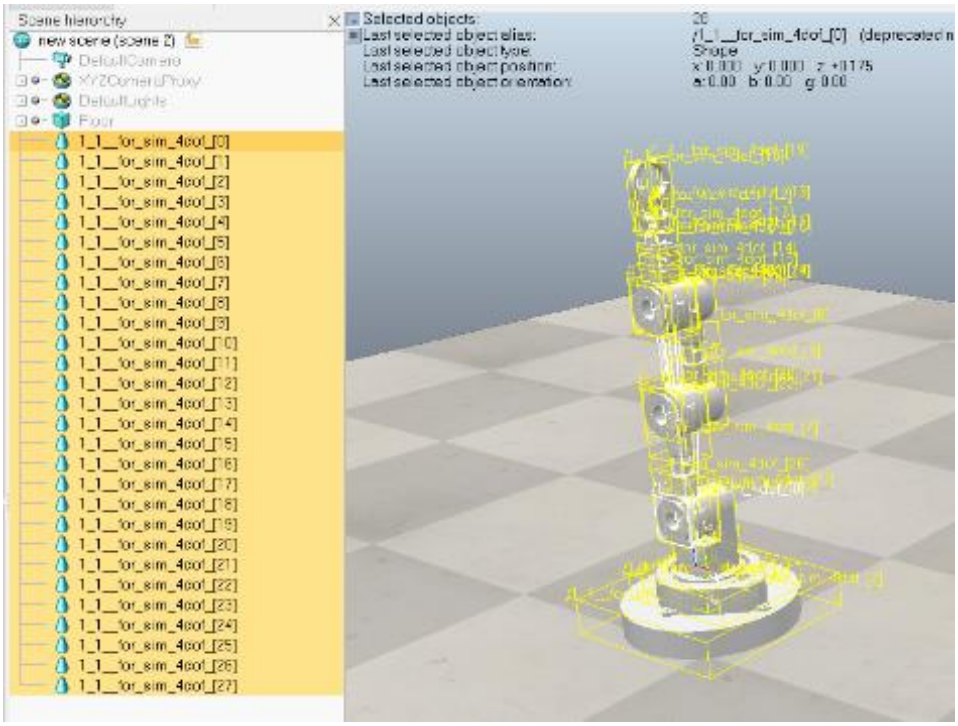


Figure 5-9: Mesh After Simplification

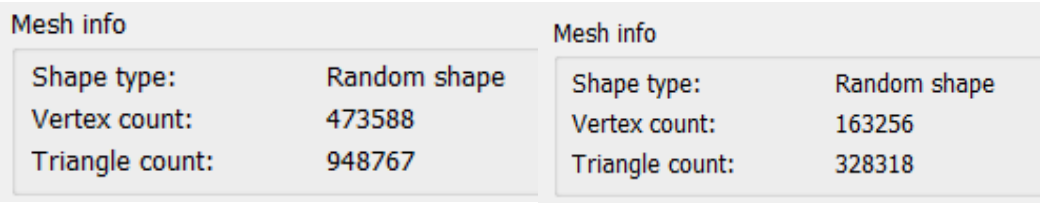


Figure 5-10: Mesh Triangle Counts Comparison

Low number of triangles without losing or distorting the appearance must be the set goal as low the number of triangles faster and more accurate the computation of the physics engine. Hence simplification is crucial.

B. Decimation of Shape

Even with the simplification the triangle count is high hence reduction of triangle counts is necessary and can be done by using Decimate shape feature.

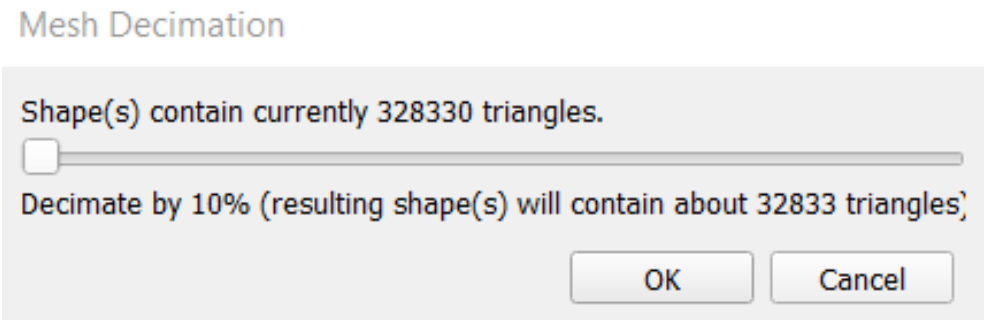


Figure 5-11: Decimation of Shape

From mesh decimation the triangles count reduced from 328330 to 32833 this process can be repeated to further reduce the number of triangles.

C. Hierarchical Links and Joints

The related parts from the scene hierarchy are grouped together as a link and joints. Once the links are defined and the joints are created, we can see them in the scene hierarchy of CoppeliaSim environment. But although they are defined the simulator doesn't know the relationship between them. Joints and links have parent child

relationship after hierarchy is defined.

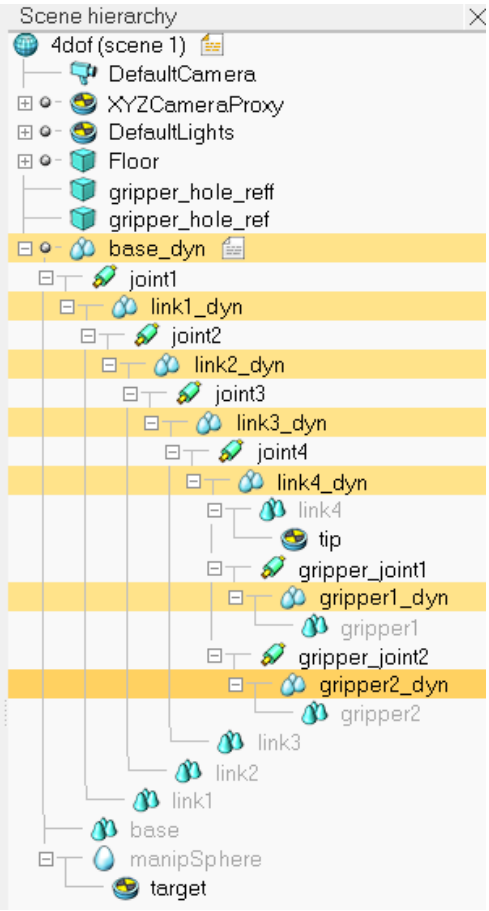


Figure 5-12: Hierarchical Structure of Robot

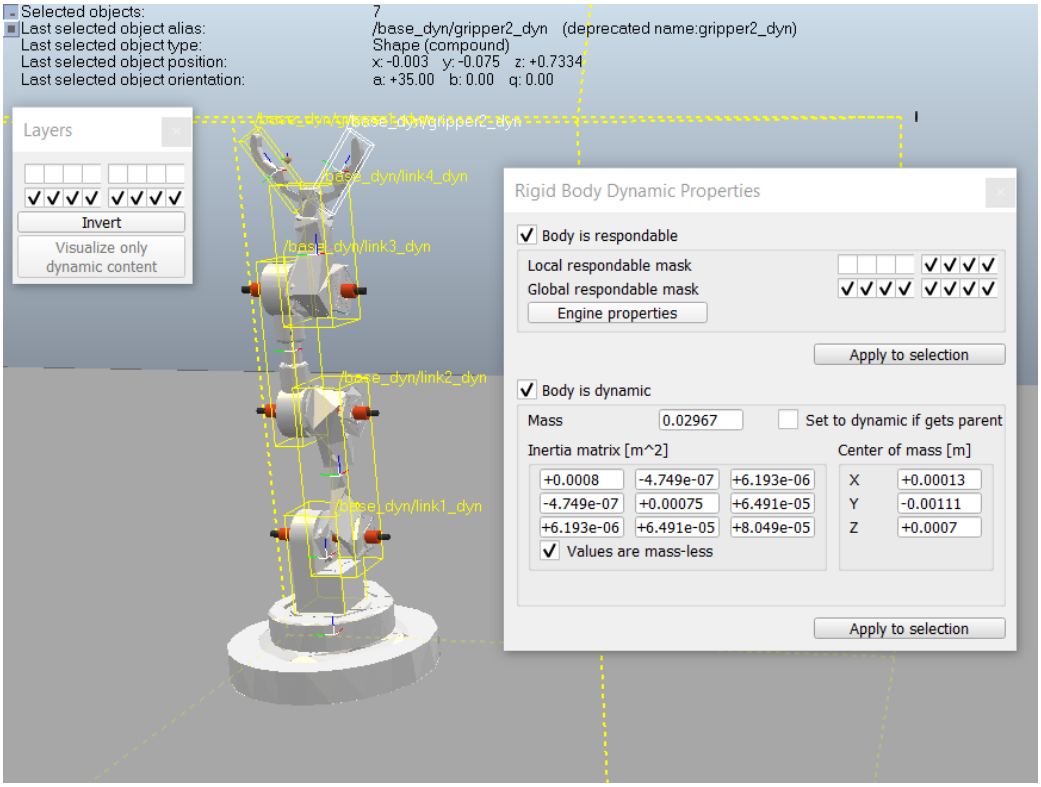


Figure 5-13: Visible Dynamic Respondable Objects

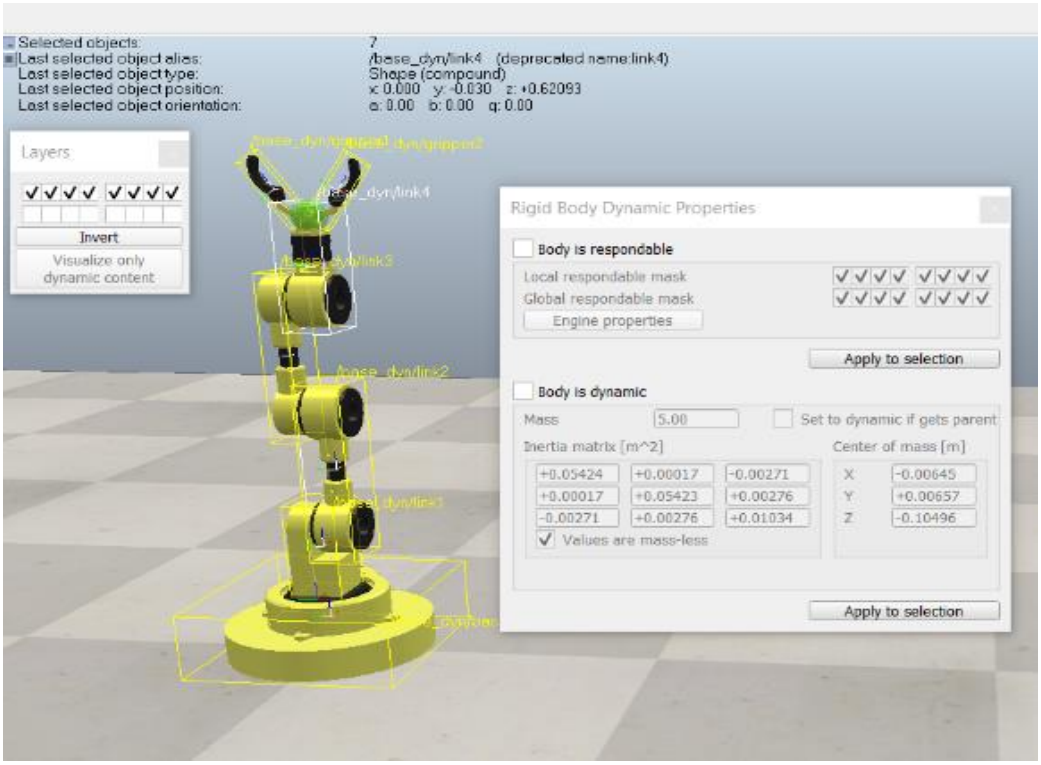


Figure 5-14: Visible Static Non-Respondable Object

D. Coloring Scene Objects

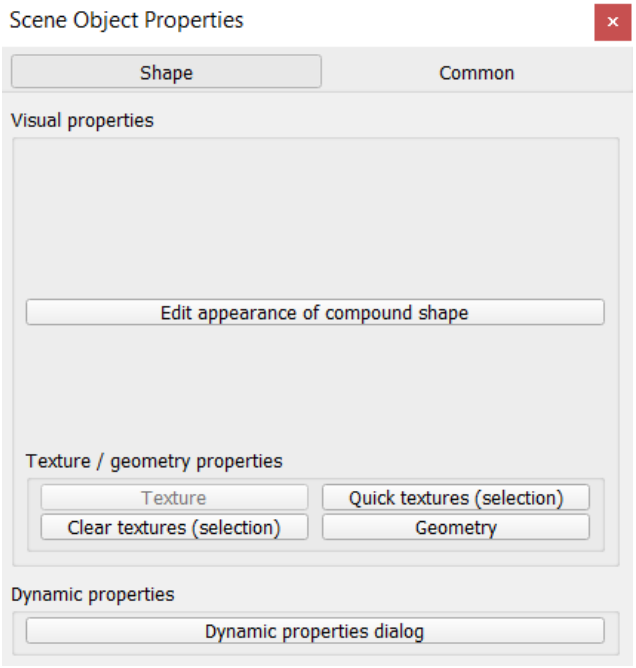


Figure 5-15: Scene Object Properties

All the objects can be colored using the scene object property's function. In the dialog edit appearance of a compound shape is present using which each shape can be selected and individual RGB values can be set.

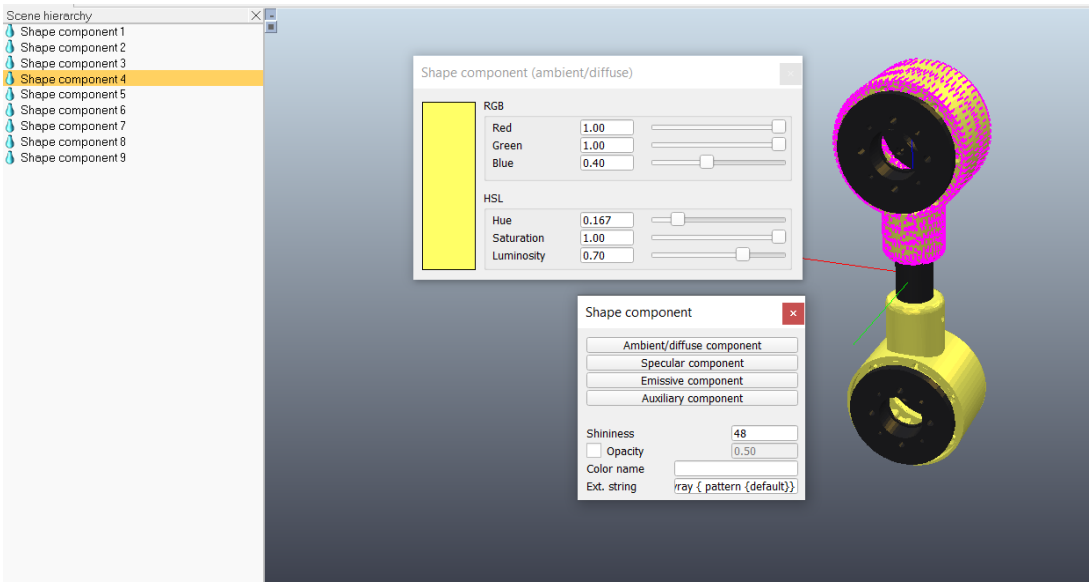


Figure 5-16: Coloring the Objects

The objects can be colored from the script or python code also which is very important in the project as changing color of glass will indicate the glass is filled with drink.

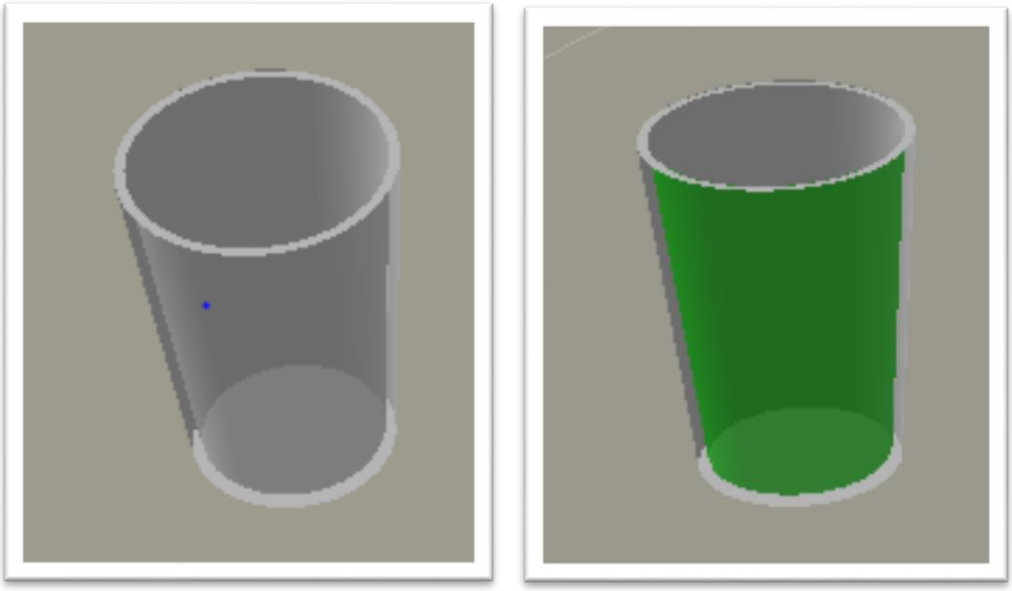


Figure 5-17: Empty Glass vs Filled Glass

Following code is used for the color changing

sim.setShapeColor(cup,None, sim.colorcomponent_ambient_diffuse,[0,1,0])

E. Local Masking

The links are connected by the joints but since the links are dynamic and respondable while moving, the interaction between links at the joints will cause some issues. In order to make the links move freely at joints there must be alternate local masking between simultaneous links.

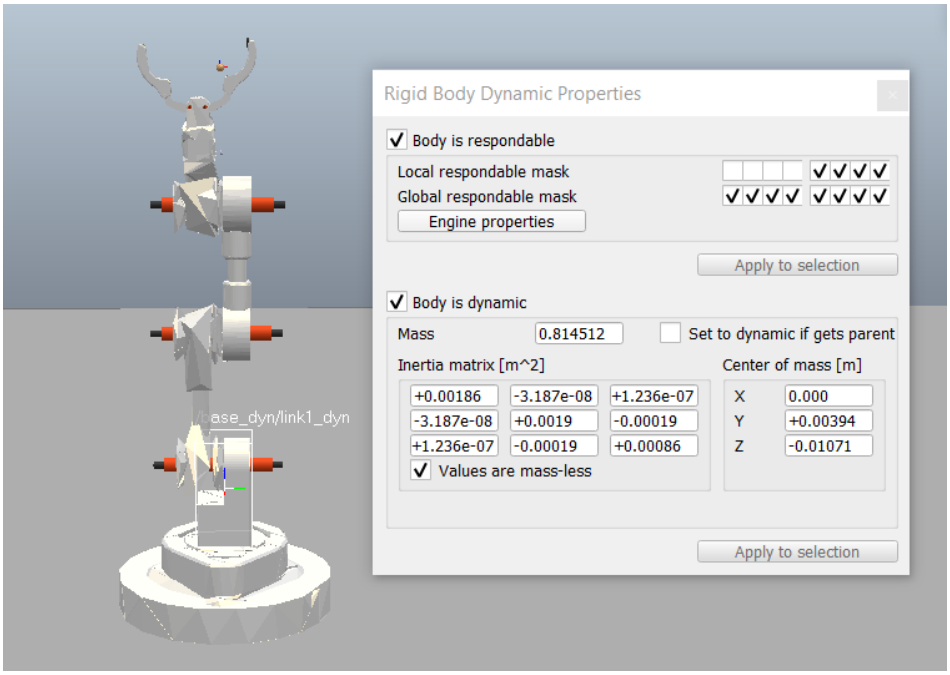


Figure 5-18: Link1 Local Respondable Masking

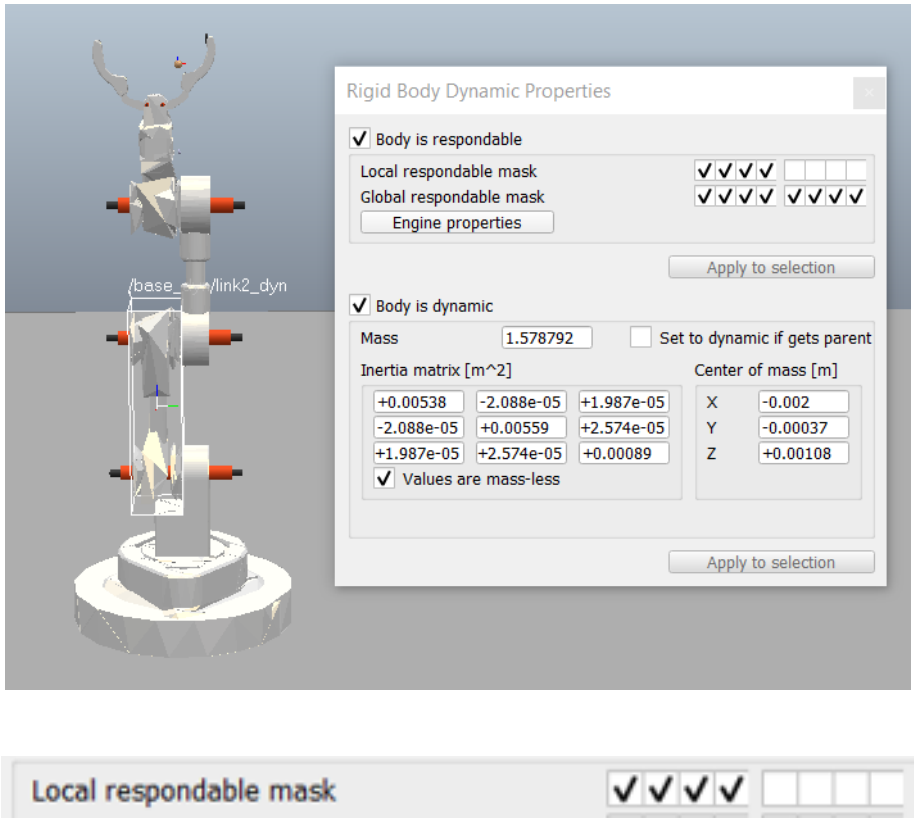


Figure 5-19: Link2 Local Respondable Mask

5.3.2 Design Parameters for Cycloidal Disk

The cycloidal disk as well a ring pins were designed using Fusion 360 add-Ins using the following parameters.

Table 5-3: Design Parameters

S.N.	Parameter Name	Value
1.	Reduction Ratio (N)	20
2.	Eccentricity (e)	1.21
3.	Ring Pin diameter (Rpd)	05.0 mm
4.	Ring pin Pitch diameter (RpD)	72.0 mm
5.	Disk Center hole diameter (Chd)	32.0 mm
6.	Around hole diameter (Ahd)	05.0 mm
7.	Center to hole distance (Cth)	21.5 mm

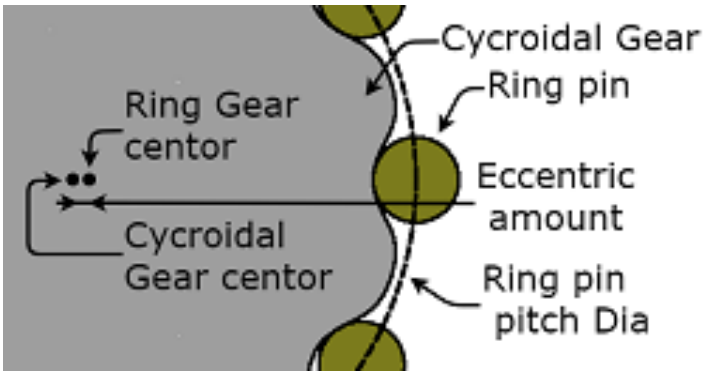


Figure 5-20: Disk and Ring Pins

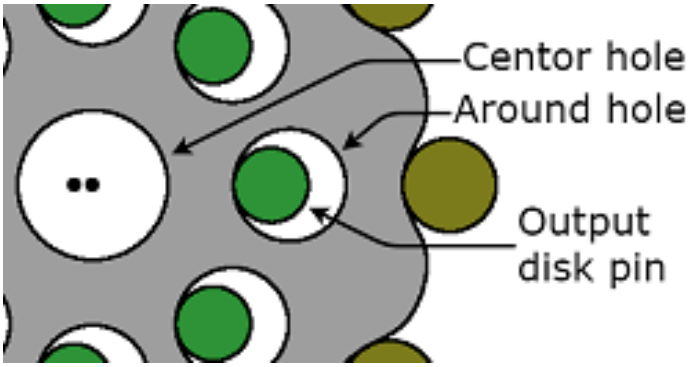


Figure 5-21: Output Shaft and Holes

After all the parameters are added to the fusion 360 add-in the following sketch is drawn which can be extruded to create a 3d object.

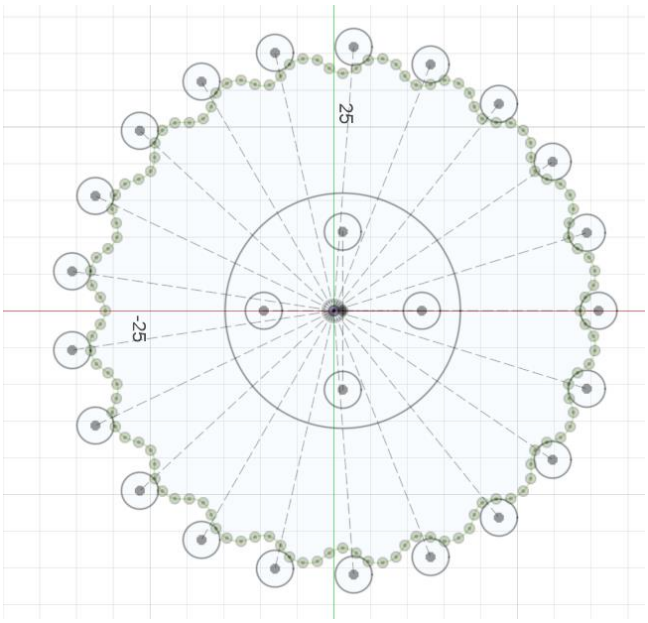
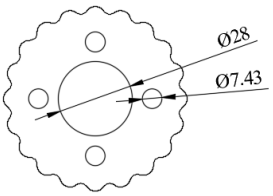
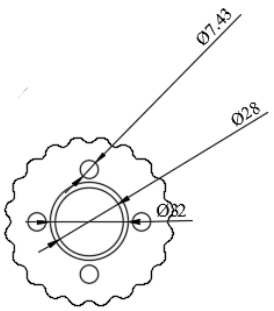
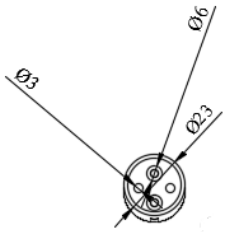
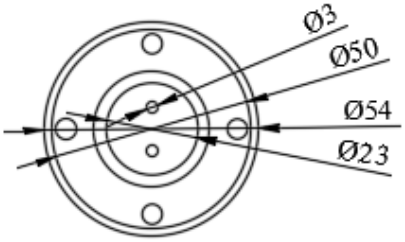


Figure 5-22: Drawn Sketch

5.3.3 Dimensions of Design

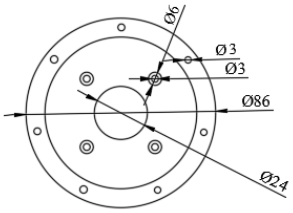
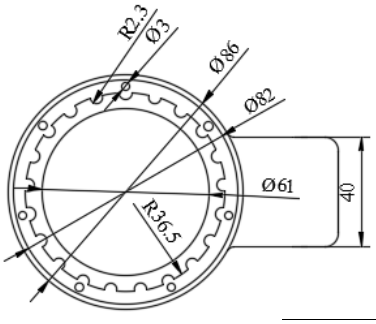
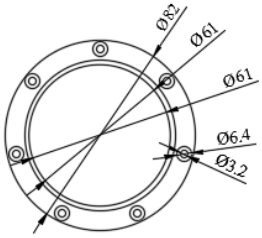
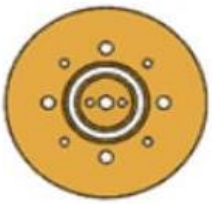
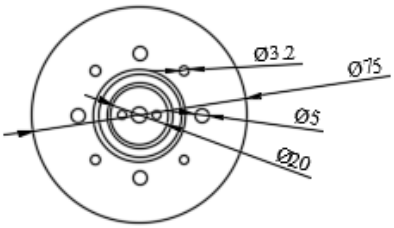
The dimensions provided are extracted from Fusion 360, a computer-aided design (CAD) software. They represent the top view inner parts of the cycloidal drive which includes cycloidal disk, eccentric ca and input/output coupler showcasing the spatial parameters and key measurements.

Table 5-4: Top View Inner Parts Dimensions

Design view	Drawing Dimensions	Part Name
		Cycloidal Disk1
		Cycloidal Disk 2
		Eccentric Cam
		Input and Output Coupler

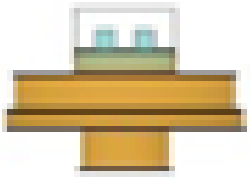
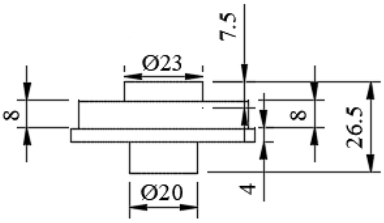
They top view representation of the cycloidal drive includes stepper mount, main housing, housing lid and input/output coupler.

Table 5-5: Top View Outer Parts Dimensions

Design View	Drawing Dimensions	Part Name
		Stepper Mount
		Main Housing
		Main Housing Lid
		Input and Output Coupler

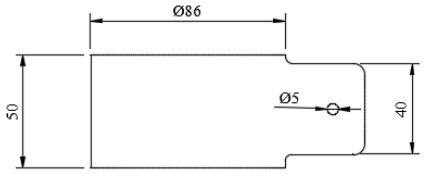
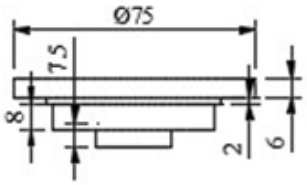
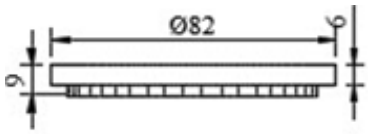
The dimensions provided are extracted from Fusion 360, a computer-aided design (CAD) software. They represent the side view inner parts of the cycloidal drive which includes main housing, housing lid, cycloidal disk and input/output coupler, showcasing the spatial parameters and key measurements.

Table 5-6: Side View Inner Parts Dimensions

Design View	**Drawing Dimensions**	**Part Name**
		Cycloidal Disk
		Input and Output Coupler

They side view outer parts representation of the cycloidal drive includes input and output coupler.

Table 5-7: Side View Outer Parts Dimensions

Design View	Drawing Dimensions	Part Name
		Main Housing
		Input Output Coupler
		Houing Lid

5.3.4 Gripper Design

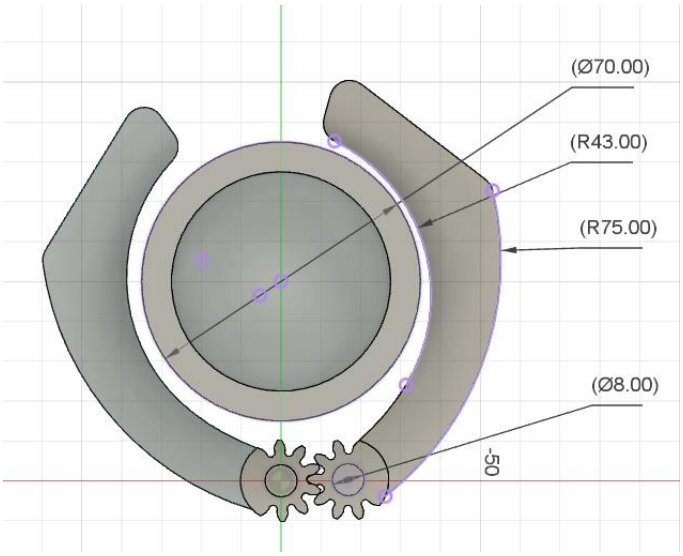


Figure 5-23: Gripper Dimensions

A claw gripper is a type of robotic or mechanical device designed to grasp and hold

objects securely. It typically features two or more opposing jaws or fingers that can open and close to surround an object, mimicking the action of a claw or pincer which is driven using servo motor in this project.

5.3.5 3D Printing

3D printing involves creating a three-dimensional object layer by layer based on a digital model. The specific procedure can vary depending on the type of 3D printer, the material used, and the complexity of the model. An overview of the 3D printing process can be for this project can be:

A. Create a 3D Model:

Use computer-aided design (CAD) software Fusion 360 to create a 3D model or download an existing model from online repositories.

B. Model Preparation:

Exporting the design as a mesh file.

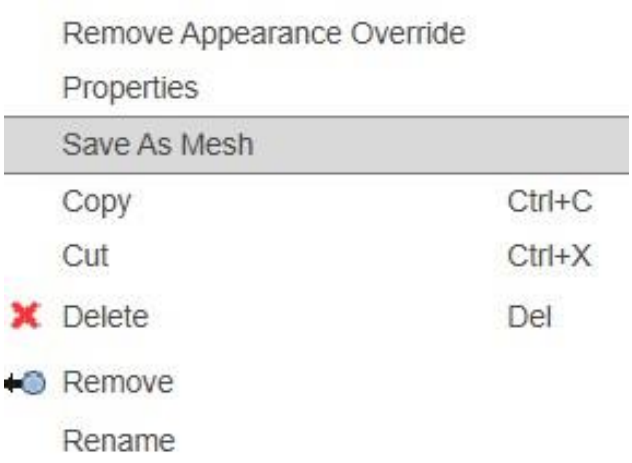


Figure 5-24: Fusion 360 Export Mesh File

Use slicing software Cura to prepare the model. Slicing involves dividing the 3D model into thin layers and generating the toolpaths for the printer.

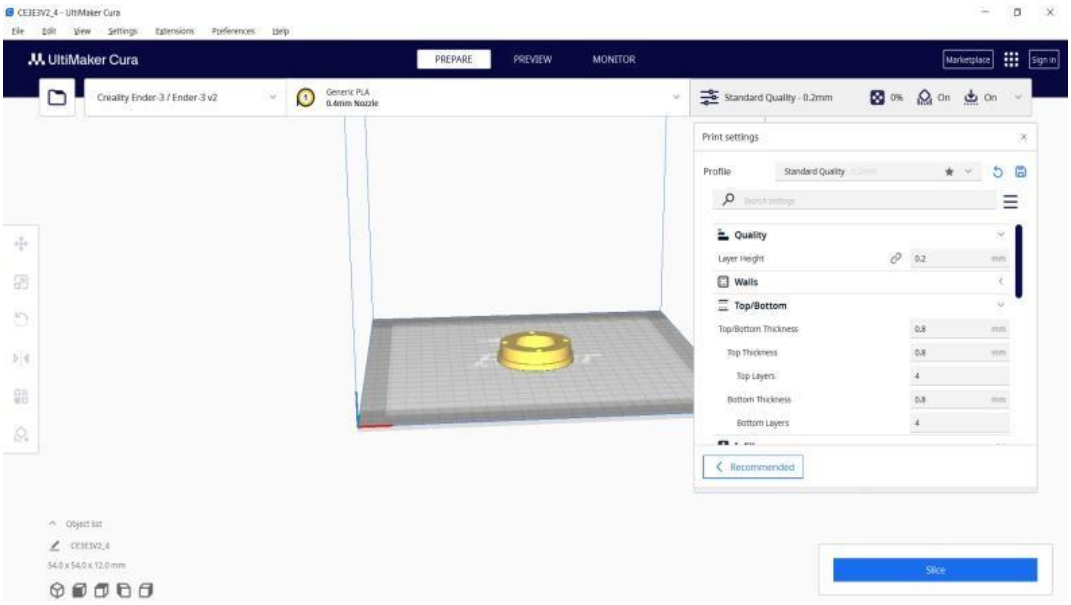


Figure 5-25: Slicing Software Cura

C. Select Printing Material:

Appropriate printing material should be chosen. Common materials include PLA, ABS, PETG for filament printers, and various resins for resin printers. PLA was used in this project

Figure 5-26: PLA Filament

D. Load Filament:

Filament is loaded to the printer where its end reaches to the nozzle.

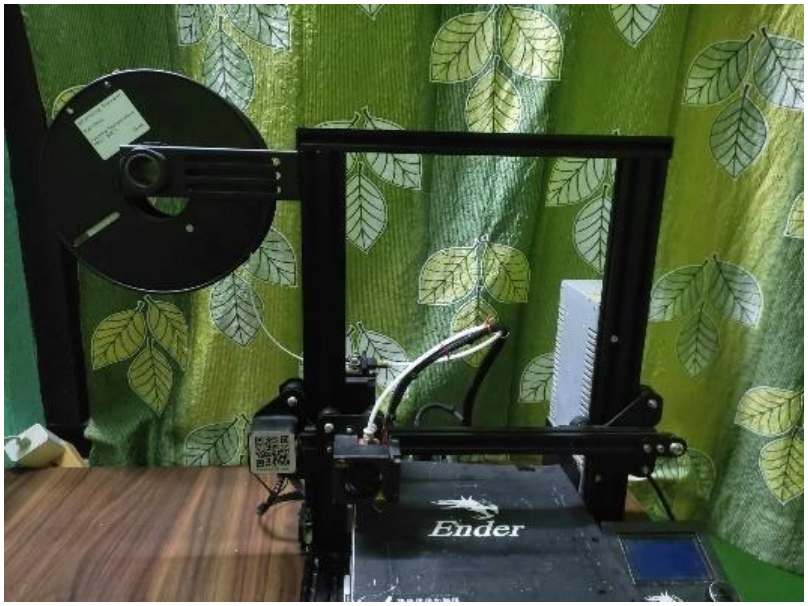


Figure 5-27: Filament Loaded to Ender 3 Printer

E. Calibrate the Printer:

The bed level should be properly calibrated using the four knobs at four points of the square bed.

Figure 5-28: Printer Bed with Two Visible Circular Knobs

F. Generate G-code:

After slicing, the slicing software generates G-code, which contains instructions for the 3D printer on how to move, extrude, and build each layer the G-code file is saved to an SD card or transfer it directly to the printer.

5.4 Controlling the Motors

5.4.1 Stepper Drivers (TB6600)

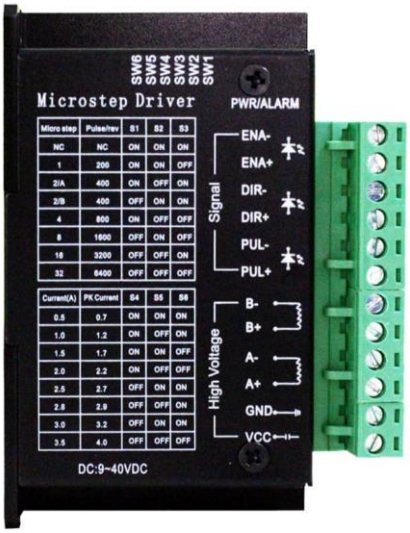


Figure 5-29: TB6600

The TB6600 is a commonly used stepper motor driver. It typically comes in a package with multiple pins for connection to control signals, power supply, and stepper motor coils.

VCC: This is the power supply pin. It usually requires a DC power supply within the specified voltage range (often around 9-42V DC).

GND: Ground connection for the power supply.

CLK-, CLK+: These are the pulse input pins for controlling the stepper motor. CLK- is connected to the ground, and CLK+ receives the pulse signal.

ENA-, ENA+: These are the enable pins. To enable the driver, ENA+ is connected to

the power supply and ENA- to the ground which is the default state so these can be left unconnected.

PUL-, PUL+: These pins receive the pulse signal to control the stepper motor. PUL- is connected to the ground, and PUL+ receives the pulse signal.

DIR-, DIR+: These pins control the direction of rotation. DIR- is often connected to the ground, and DIR+ determines the direction by the input signal (High for one direction, Low for the other).

A+ A- B+ B-: These are the connections for the stepper motor coils. They're connected to the respective coil wires of the stepper motor.

Various micro stepping and current limit can be set using the 6 switches. The modes are represented by the following figure.

Table 5-8: Micro Stepping Table

Micro Step	Pulse/rev	S1	S2	S3
NC	NC	ON	ON	ON
1	200	ON	ON	OFF
2/A	400	ON	OFF	ON
2/B	400	OFF	ON	ON
4	800	ON	OFF	OFF
8	1600	OFF	ON	OFF
16	3200	OFF	OFF	ON
32	6400	OFF	OFF	OFF

Table 5-9: Current Limit Table

Current(A)	PK Current	S4	S5	S6
0.5	0.7	ON	ON	ON
1.0	1.2	ON	OFF	ON
1.5	1.7	ON	ON	OFF
2.5	2.2	ON	OFF	OFF
2.5	2.7	OFF	ON	ON
2.8	2.9	OFF	OFF	ON
3.0	3.2	OFF	ON	OFF
3.5	4.0	OFF	OFF	OFF

Figure 5-30: DIP Switch

To control 4 stepper motors 4 such drivers are required. They are connected with the Arduino and power supply as shown in the figure below.

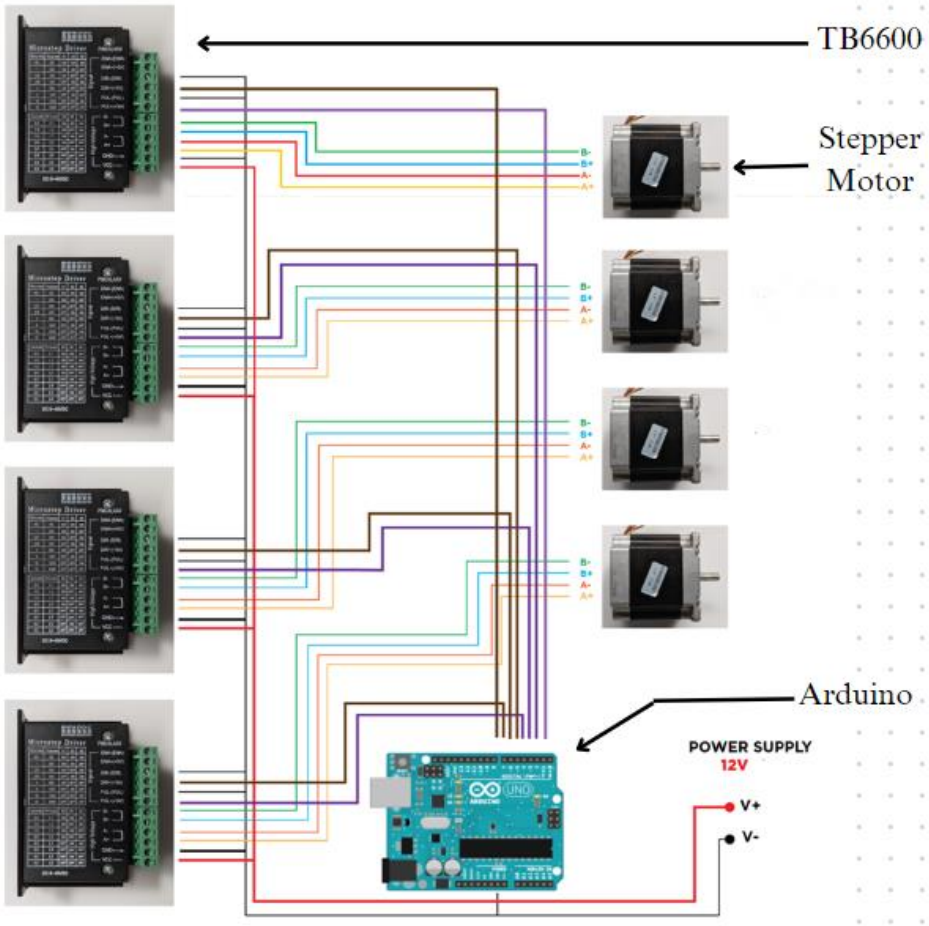


Figure 5-31: TB6600 Connection with Arduino

5.4.2 Servo (MG90)

The MG90 servo motor is a compact and reliable micro-sized servo known for its moderate torque output, typically ranging from 1.8 kg/cm to 2.5 kg/cm, and its fast response speed, rotating between 0.08 to 0.12 seconds per 60 degrees. Operating within a voltage range of 4.8V to 6.0V DC, it offers a rotation range of approximately 180

degrees and is constructed with durable plastic housing and metal gears. With standard 3-wire interface for control, including power, ground, and control signal, and dimensions of around 22.8mm x 12.2mm x 28.5mm.

The servo motor is used as a robotic gripper system using an Arduino microcontroller. Through wiring as shown in the figure and programming, the servo motor was configured to function as a gripper, capable of opening and closing to manipulate objects. The C++ code, utilizing the Servo library, enabled the Arduino to control the servo motor's actions, facilitating precise gripping movements. Basic functionality, including rotation to 0, 90, and 180 degrees.

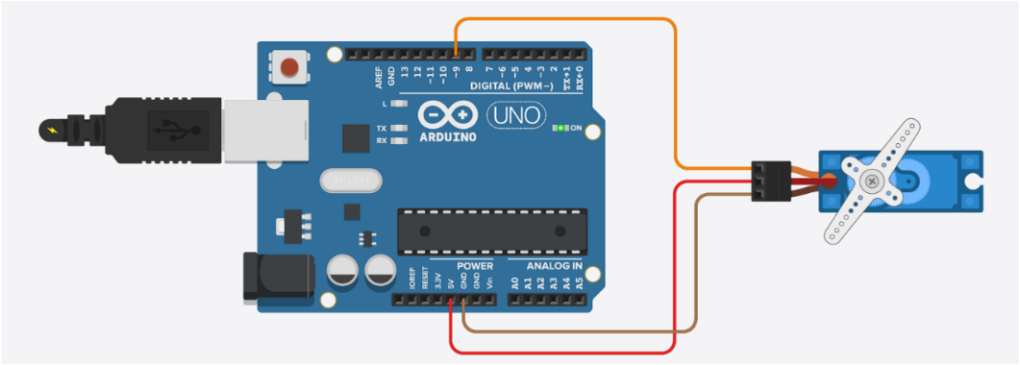


Figure 5-32: Arduino and Stepper

5.5 Raspberry PI/Python to Arduino

5.5.1 Python/Raspberry PI Side

After the angle calculation from the inverse kinematics the desired angles are obtained. The angles are converted into string and separated by commas with carriage return at the end to determine the end of string. The string might look like “40, 50, 60, 70\r”. The Python script sets up a serial connection with an Arduino board connected to the computer via certain com port at a baud rate of 9600. The data is then encoded in utf-8 format and transmitted to the Arduino through the serial connection.

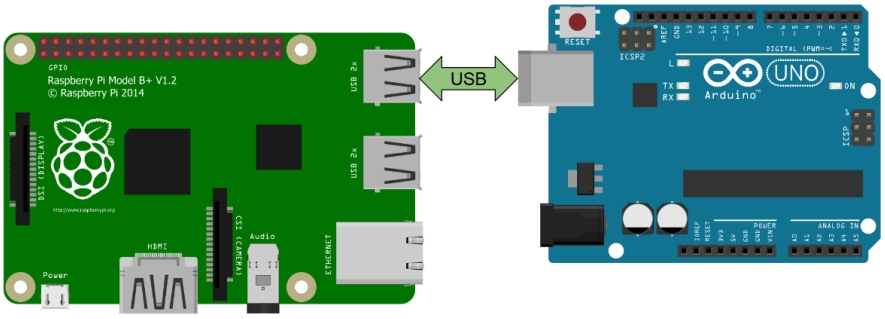


Figure 5-33: Arduino and Raspberry PI

5.5.2 Arduino Side

Arduino listens for data coming in from a connected device via the serial port. It waits until it receives a complete string of values separated by commas. Using carriage return '\r' it determines the end of string. Once the string is received, it splits it into individual pieces based on the commas and converts each piece into a number. These numbers are then stored in an array, ready for use in controlling the servo motors or any other part of the program. This process allows the Arduino to understand and respond to commands or data sent to it from an external source, like a computer or Raspberry Pi in this case, making it possible to control the behavior of the system remotely.

5.6 Torque Computation

Torque is a measure of the rotational force applied to an object, causing it to rotate around an axis. It is typically represented by the symbol "τ" (tau) and is measured in units such as Newton-meters (Nm). Torque is calculated by multiplying the force applied perpendicular to the distance from the axis of rotation.

Mathematically, torque (τ) is expressed as:

$$\tau = r * F * sin(\theta) \qquad 5\text{-}1$$

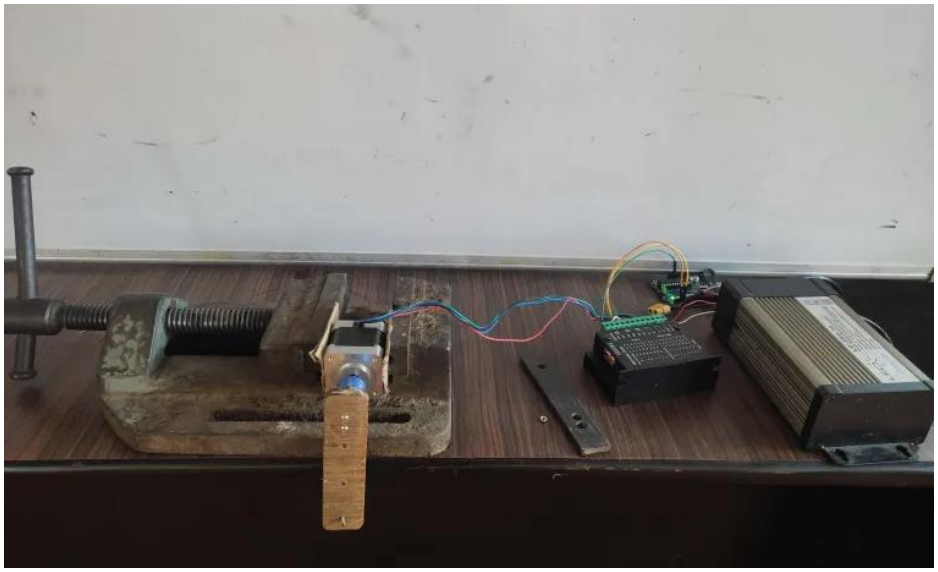


Figure 5-34: Stepper Torque Testing Setup

For the torque measurement an input shaft is attached to a wooden extension having 3 holes at different distance. The holes are the placeholder for the different number of weights. TB6600 was used for the stepper driver to drive Nema 17CS04A-170 in full step mode which was powered by 12V power supply.

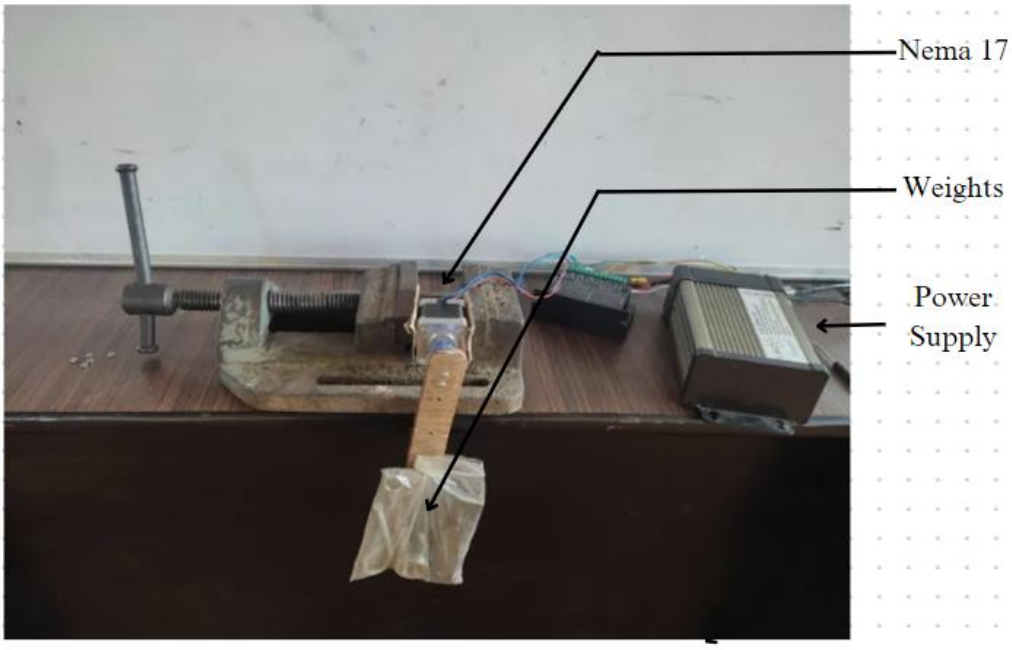


Figure 5-35: Stepper Motor with Weight

Figure 5-36: Weight Lifted

Figure 5-37: Weights

The hole is at 9cm from the center of the shaft. It could lift 273gm weight with ease. The wooden extension had a weight of 25gm whose center of gravity was approximately 3cm away from the center which produced 0.75Ncm torque.

Hence the total torque produced was 24.57Ncm + 0.75Ncm=25.32Ncm torque. So the motor has torque of at least 25Ncm.

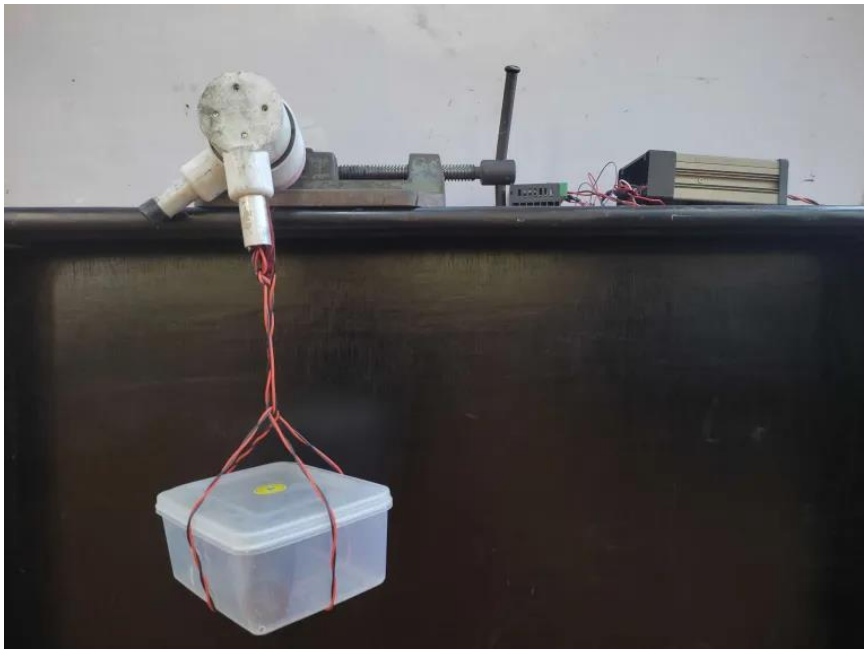


Figure 5-38: Gearbox Torque Test

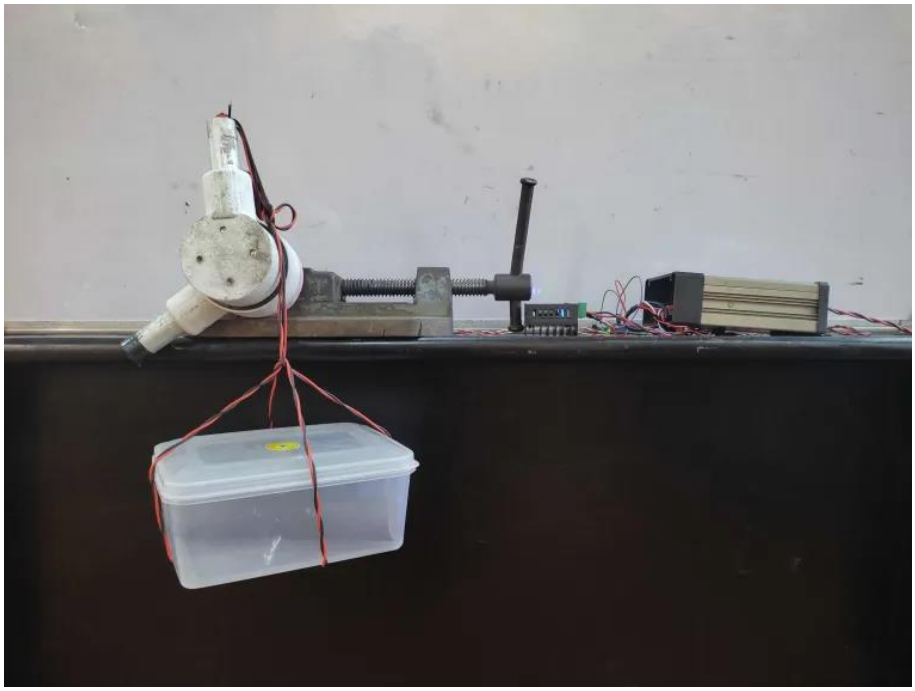


Figure 5-39: Weight Lifted

Figure 5-40: Total Weight

Table 5-10: Results of Torque Test

Length(cm)	Mass(kg)	Torque (Nm)	remark
10	0.5	0.5	Stable
10	1	1	Stable
10	1.5	1.5	Stable
10	2	2	Stable
15	1.74	2.61	Stable
15	2.5	3.75	Unstable

The experiment reveals that the maximal torque limit for the TB6600 controllers is 2.61Nm, with 17CS04A-170.

5.7 Forward and Inverse Kinematics Analysis

This section discusses about the forward and inverse kinematics solution used for the 4-DOF robotic arm.

5.7.1 Forward Kinematics

Forward kinematics is a concept in robotics that deals with the calculation of the position and orientation of an end-effector based on the known joint angles and link parameters of a robot's kinematic chain. The homogeneous transformation matrix derived from the kinematic modelling of the robotic arm is given by:

$$H_4^0 = \begin{bmatrix} c_1c_{234} & -c_1s_{234} & s_1 & c_1(a_4c_{234} + a_2c_2 + a_3c_{23}) \\ c_{234}s_1 & -s_1s_{234} & -c_1 & s_1(a_4c_{234} + a_2c_2 + a_3c_{23}) \\ s_{234} & c_{234} & 0 & a_4s_{234} + a_2c_2 + a_3s_{23} + a_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad 5\text{-}2$$

The first three row's element of fourth column of the given matrix provides the x, y, z position of the end-effector and is given by:

$$x = c_1(a_4c_{234} + a_2c_2 + a_3c_{23}) \quad 5\text{-}3$$

$$y = s_1(a_4c_{234} + a_2c_2 + a_3c_{23}) \quad 5\text{-}4$$

$$z = a_4s_{234} + a_2c_2 + a_3s_{23} + a_1 \quad 5\text{-}5$$

5.7.2 Inverse Kinematics

Inverse kinematics is the reverse of forward kinematics. While forward kinematics calculates the position and orientation of the end-effector given the joint angles, inverse kinematics aims to find the joint angles necessary to reach a particular position and orientation in the workspace.

The inverse kinematics problem is solved using gradient descent. This method relies on the Jacobian matrix, which relates the rate of change of the end-effector's position and orientation to changes in the joint angles of the robot.

I. Inverse Kinematics Design Parameters

The link lengths of the robotic arm play a vital role in the Inverse Kinematics solution. The design difference of robotic arm in terms of link length in between simulation environment and reality is represented below:

Table 5-11: Real Robot Link Lengths

S.N	Link Length (m)	Value (in meters)
1.	Link1	0.17
2.	Link2	0.20
3.	Link3	0.20
4.	Link4	0.20

Table 5-12: Virtual Robot Link Lengths

S.N	Link Length	Value (in meters)
1.	Link1	0.216
2.	Link2	0.372
3.	Link3	0.353
4.	Link4	0.376

Table 5-13 Nomenclature for Inverse Kinematics Problem

Symbol	Remarks
H_k	Current homogeneous transformation matrix of end effector w.r.t base
α	Step size
H_d	Desired homogeneous transformation matrix of end effector w.r.t base
Δx	Vector: difference in desired and actual pose of the end effector
e_p , e_o	Vectors: position and orientation error between current and desired pose
θ_k	Vector: joint angles at time instant k
$FK(\theta_i)$	Forward Kinematics method; provides end effector x, y, z position according to given joint angles
$\Delta\theta$	Vector: Change in joint angles
J^+	Pseudo-Inverse of Jacobian matrix

Algorithm 1 Inverse Kinematics via Gradient Descent

Require: H_d and H_k at step k, J^+, α, $FK(\theta_i)$

1: **loop** **Stop after** $\left|\left|\theta_k\right|\right| \approx 0$

2: Evaluate $\Delta x = \begin{bmatrix} e_p \\ e_o \end{bmatrix}$

3: Compute $\Delta\theta = J^+\Delta x$

4: Increment θ_k to converge end effector pose on the desired pose:

$$\theta_{k+1} = \theta_k + \alpha J^+ \Delta\theta_k$$

5.8 MPC Design Parameters

There are various design parameters that should be taken into account for the successful solution of MPC problem. In the case of reference trajectory tracking via MPC, the design parameters used are discussed below:

Table 5-14: MPC Design Parameters

S.N	Design Parameter	Value
1.	Prediction Horizon	10
2.	Control Horizon	5
3.	Weight matrix to penalize input changes (W_2)	$\begin{bmatrix} 10^{-3} & 0 & 0 & ... & 0 \\ 0 & 10^{-3} & 0 & ... & 0 \\ 0 & 0 & 10^{-3} & ... & ... \\ 0 & 0 & 0 & ... & 10^{-3} \end{bmatrix}$
4.	Weight matrix to penalize tracking error (W_4)	$\begin{bmatrix} 10 & 0 & ... & 0 \\ 0 & 10 & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & 0 & 10 \end{bmatrix}$

5.9 Audio Dataset Exploration

This section will explore the details about the various implementation steps adopted during dataset preparation

5.9.1 Dataset Collection

Various kinds of dataset were collected for various purposes including training of speech recognition and for feedback by the system.

A. Pretraining Dataset

The Librespeech dataset was used for pretraining of Squeezeformer model. The "LibriSpeech" dataset is a widely used corpus for training and evaluating speech recognition systems. It consists of English speech recordings derived from audiobooks, making it suitable for speech recognition research.

Table 5-15: Gender Distribution of Speakers for Various Subsets

Subset	**Hours**	**Minutes per-speaker**	**Female speakers**	**Male speakers**	**Total speakers**
dev-clean	5.4	8	20	20	40
test-clean	5..4	8	20	20	40
dev-other	5.3	10	16	17	33
test-other	5.1	10	17	16	33
train-clean-100	100.6	25	125	126	251
train-clean-360	363.6	25	439	482	921
train-other-500	496.7	30	564	602	1166

The LibriSpeech dataset is divided into three training subsets based on the duration and size of the recordings. Here's a description of the three training datasets:

The LibriSpeech dataset offers three subsets for training speech recognition models. "train-clean-100" contains 100 hours of high-quality, minimally-noisy speech recordings. "train-clean-360" expands this with 360 hours of similar data. In contrast, "train-other-500" provides 500 hours of diverse recordings with varying background noise levels and audio qualities, ensuring robust model training across real-world conditions.

B. Finetuning Dataset

For finetuning of Squeezeformer model a custom dataset was prepared the table below represents the demographic of that dataset collected.

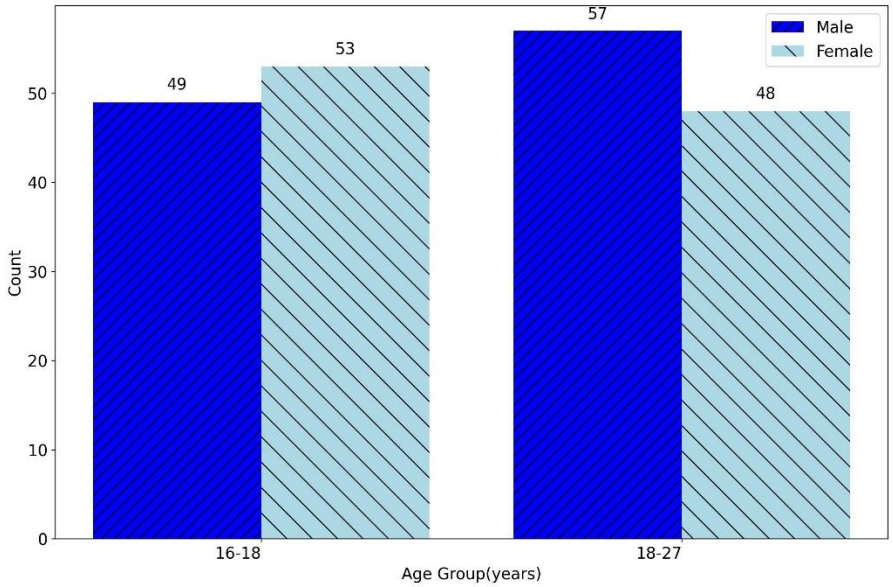


Figure 5-41: Finetuning Speech Dataset before Augmentation

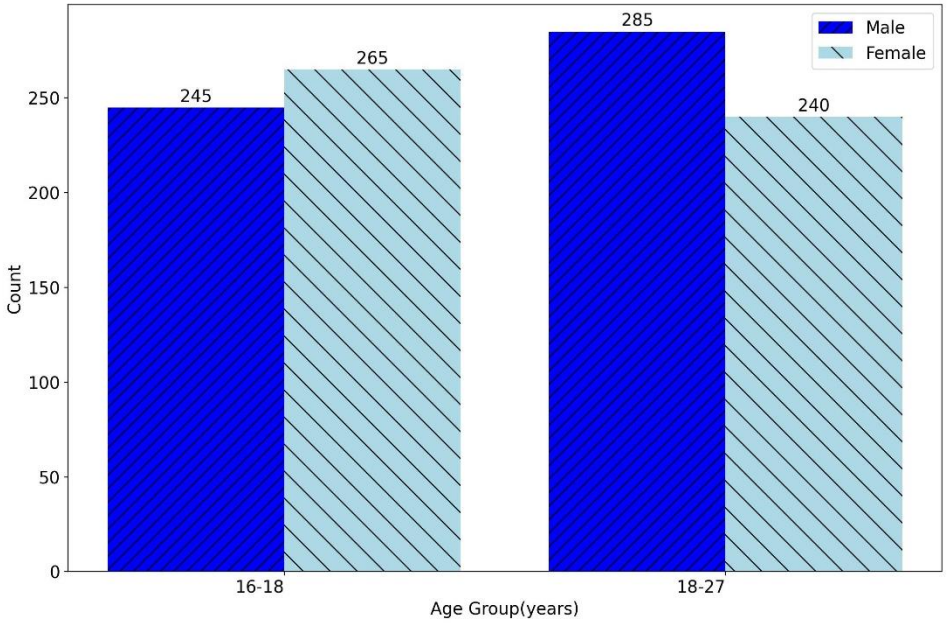


Figure 5-42: Finetuning Speech Dataset after Augmentation

The histogram above depicts the fine-tuning dataset, which exhibits a biased distribution with the majority of participants concentrated within the younger age spectrum. Due to challenges in collecting data across the entire age range, a decision was made to focus data collection efforts primarily on younger individuals, particularly school and campus students. This approach was chosen to streamline the data collection process and ensure a sufficient volume of representative data, albeit primarily from a younger demographic.

Different variation of sample commands was recorded from each participants following are the commands that were recorded.

Table 5-16: Recorded Commands

Drink	**Variation 1**	**Variation 2**	**Variation 3**	**Variation 4**	**Variation 5**
Water	Can I have a cup of water	Hey arm can I would like a cup of water	Can you get me a cup of cold water	Fill up a cup with water please	Serve me a cup of fresh water
Sprite	Pour me a cup of sprite	Hey arm can I have a cup of chilled sprite	Fill up a cup with sprite	I would like a cup of sprite	Serve me a cup of sprite
Orange Juice	Pour me a cup of orange juice	Fill up a cup with orange juice	Hey arm I would like a cup of orange Juice	Provide me with a cup of orange juice	Pour me a cup of delicious orange juice
Coffee	Provide me with a cup of	Fill up a cup with coffee	I would like a cup of coffee	Can you get me a cup of coffee	Hey Arm Make me a cup of delicious coffee

Recording a variety of commands for each drink was done to enhance the model's robustness, enabling it to handle diverse variations in user commands effectively. This approach ensures the model can accurately recognize and respond to different ways users may make their beverage requests, ultimately improving the overall user experience.

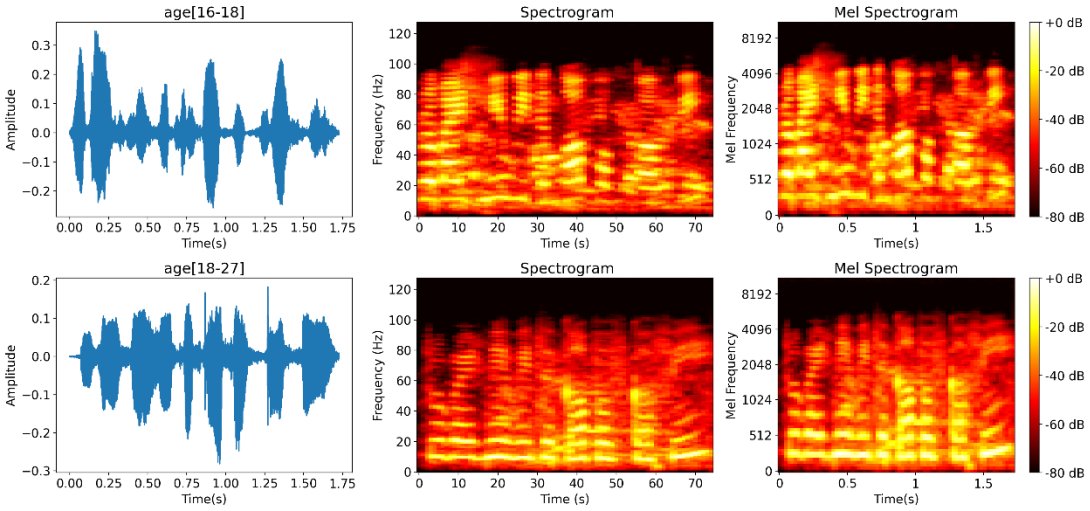


Figure 5-43: Visualizations of “Provide me with a cup of coffee”(Female)

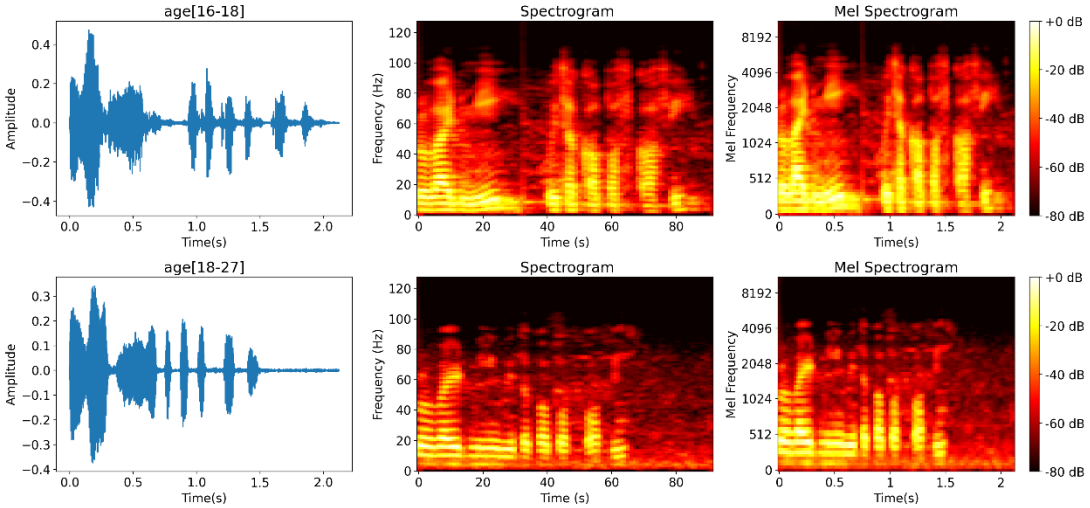


Figure 5-44: Visualizations of “Provide me with a cup of coffee” (Male)

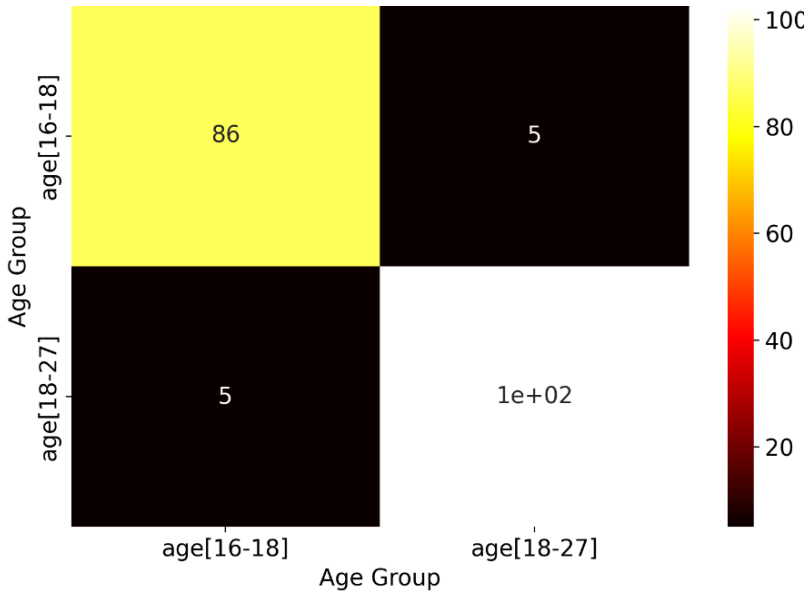


Figure 5-45: Cross-correlation Between Age Groups (Female)

The above heatmap reveals correlations between three age groups: 16-18, 18-27, and 30-50+ for female. Strong diagonal squares indicate positive correlations within each group. A moderate positive correlation exists between 18-27 and 30-50+, while weak correlations connect 16-18 with both older groups, suggesting potential differences in between them.

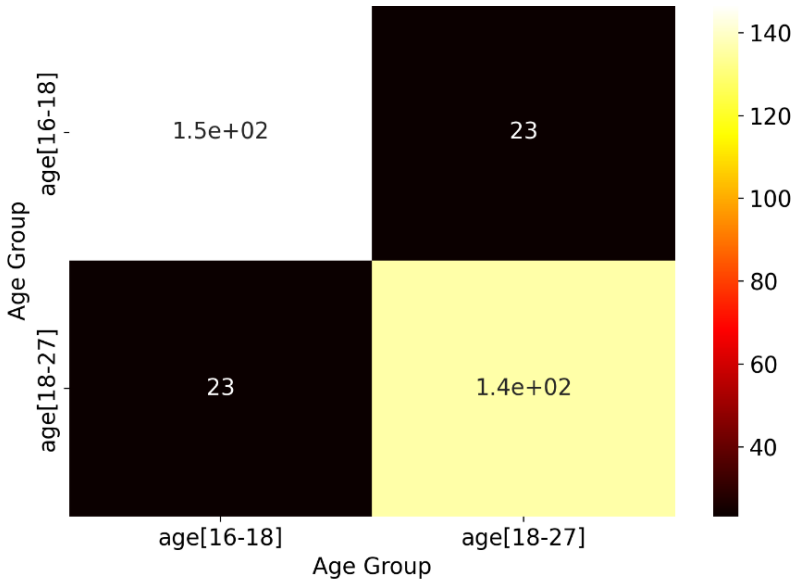


Figure 5-46: Cross-correlation Between Age Groups (Female)

The above heatmap reveals correlations between three age groups: 16-18, 18-27, and 30-50+ for male. Strong diagonal squares indicate positive correlations within each group. A moderate positive correlation exists between 18-27 and 30-50+, while weak correlations connect 16-18 with both older groups, suggesting potential differences in between them.

C. Feedback dataset

Various feedback sound samples were collected to give feedback to user in case some problems are faced by the system. The feedback dataset was collected form a student from Thapathali campus the various feedbacks collected are tabulated as below

Table 6-4: Feedback for Multiple Cases

Case	Feedback
No drink available	I'm sorry but we are currently out of that drink, would you like something else
Obstacle in the way	There's an obstacle in the way. Please clear the path
Incorrect Command	I'm sorry, I didn't understand your command, please give me a drink related command
Target position out of reach from arm	The target is out of my reach

5.9.2 AI Generation of Voice for System Feedback

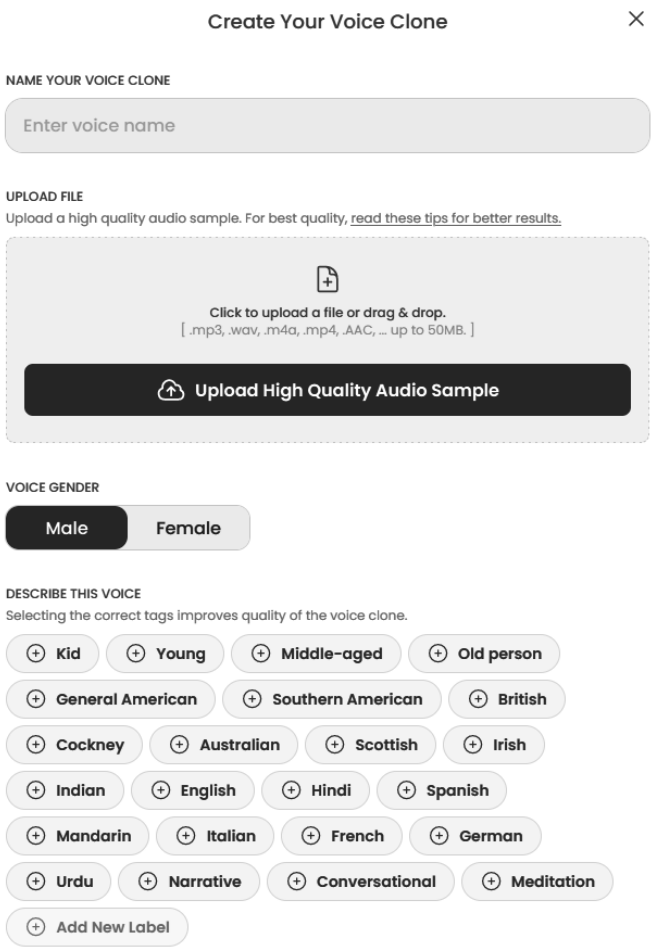


Figure 5-47: Setting Parameters for Voice Cloning

The image above depicts a feature within the 'play.ht' website, which was used for voice cloning. In this instance, a voice sample lasting 30 seconds was fed as input and parameters like gender, voice description was set and a voice clone instance called ‘combined’ was generated.

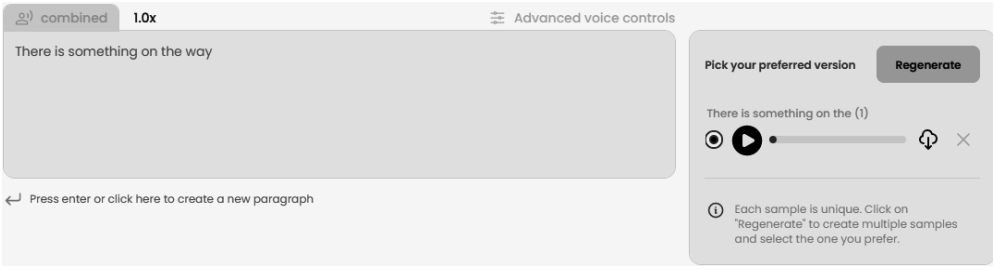


Figure 5-48: Creating Cloned Voice from Text

The image above displays the actual voice creation process, showcasing the use of previously created clone instance ‘combined’ and the text input along with a generated voice sample that can be downloaded.

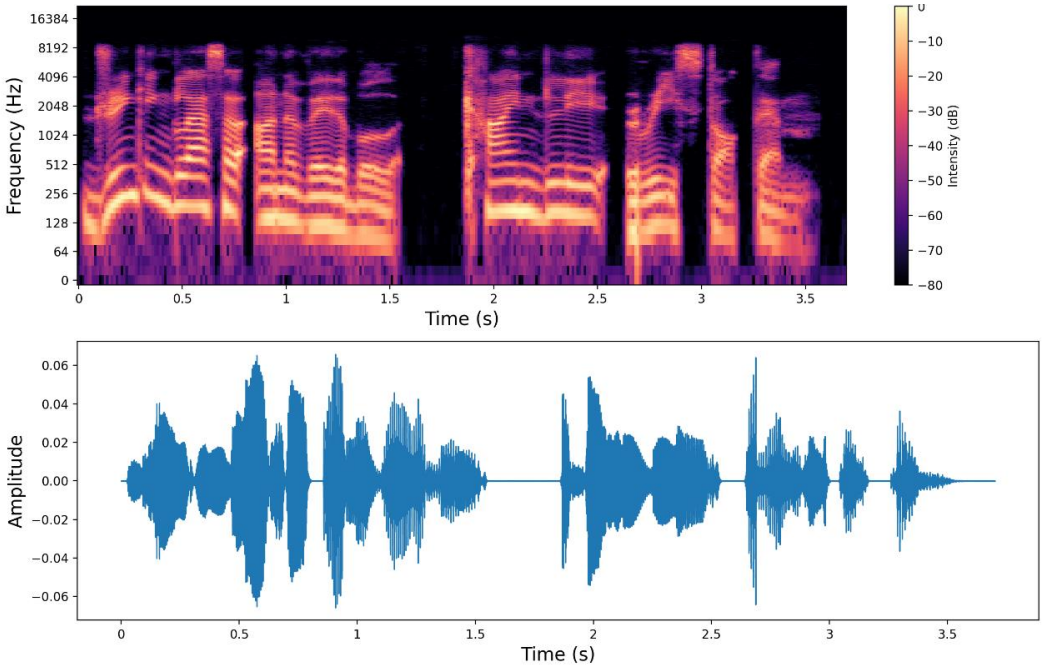


Figure 5-49: Visualizations for “drinking glass are unavailable”

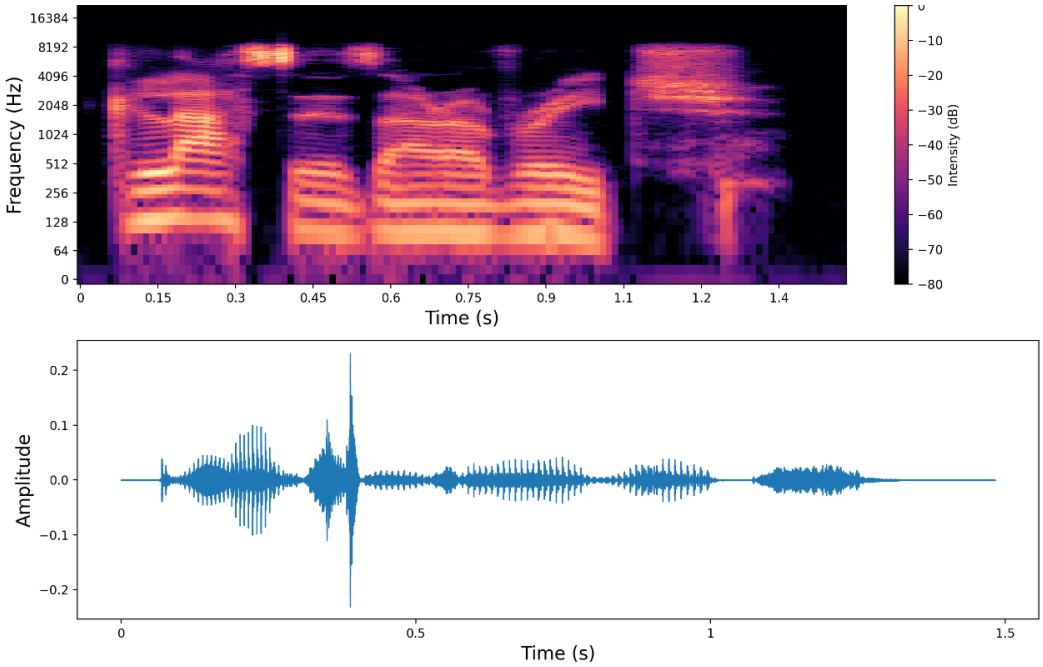


Figure 5-50: Visualizations for “glasses are out of reach”

The plots above depict the time domain signal and mel spectrograms of the cloned voice signals. It is evident from the plots that there is minimal noise present as they are cloned audio.

5.9.3 Noise Removal Techniques

Noise removal techniques are employed in speech recognition system to enhance the quality of the input audio by reducing unwanted noise, thereby improving the model's ability to recognize and understand speech.

Band-pass filters are one of the effective noise removal techniques used in speech recognition system. A band-pass filter allows only a specific range of frequencies to pass through while attenuating frequencies outside that range. By using a band-pass filter, speech recognition systems can target and eliminate noise that falls outside the frequency range of typical human speech, preserving the essential speech components within that range. The sounds were filtered using a band pass filter of range 90 Hz to 3000 Hz and of 4^{th} order for removal of various noises.

$$f_c = \sqrt{(f_1 \times f_h)} \quad 5\text{-}6$$

$$BW = f_h - f_1 \quad 5\text{-}7$$

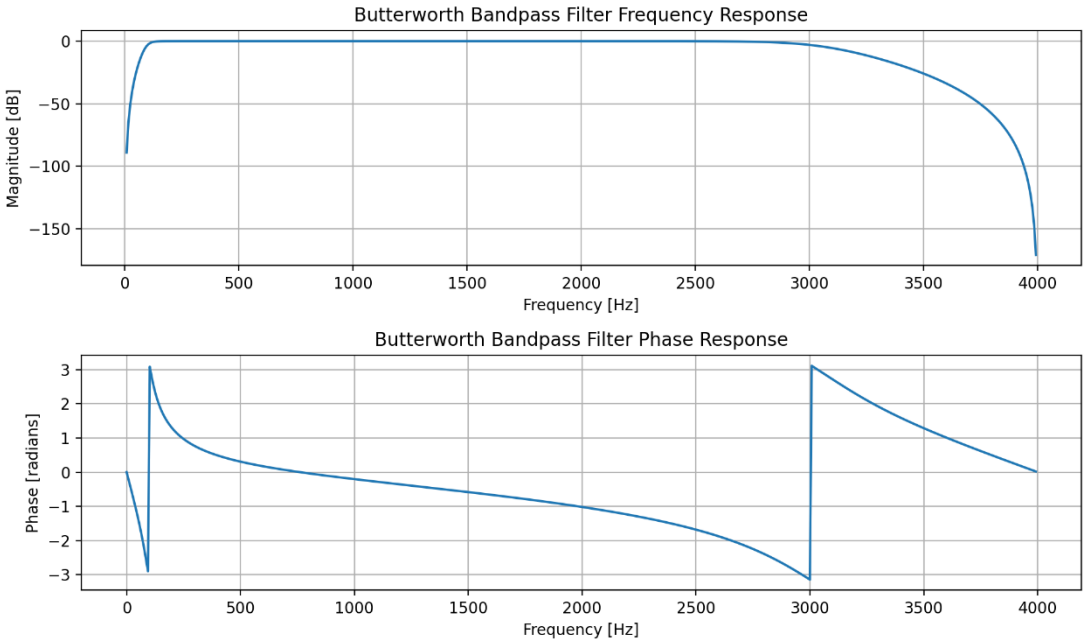


Figure 5-51: Magnitude and Phase Plot of Butterworth Filter

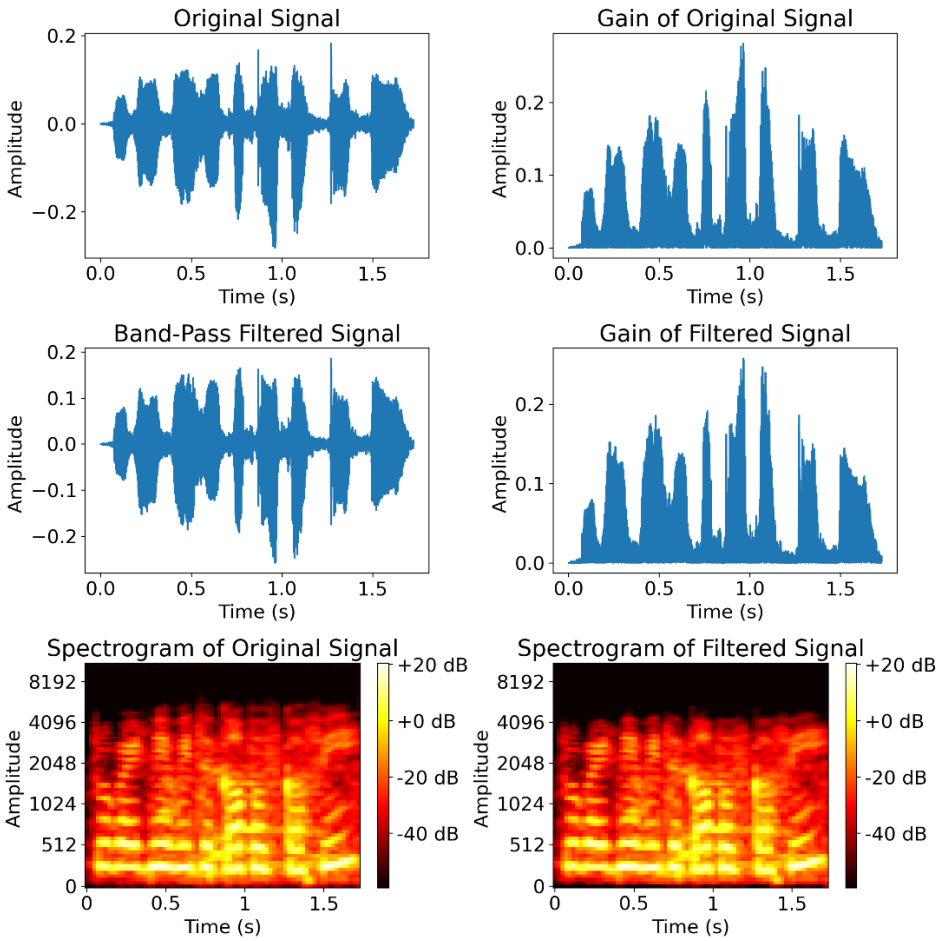


Figure 5-52: Gain and Spectrogram of Original and Filtered Signal

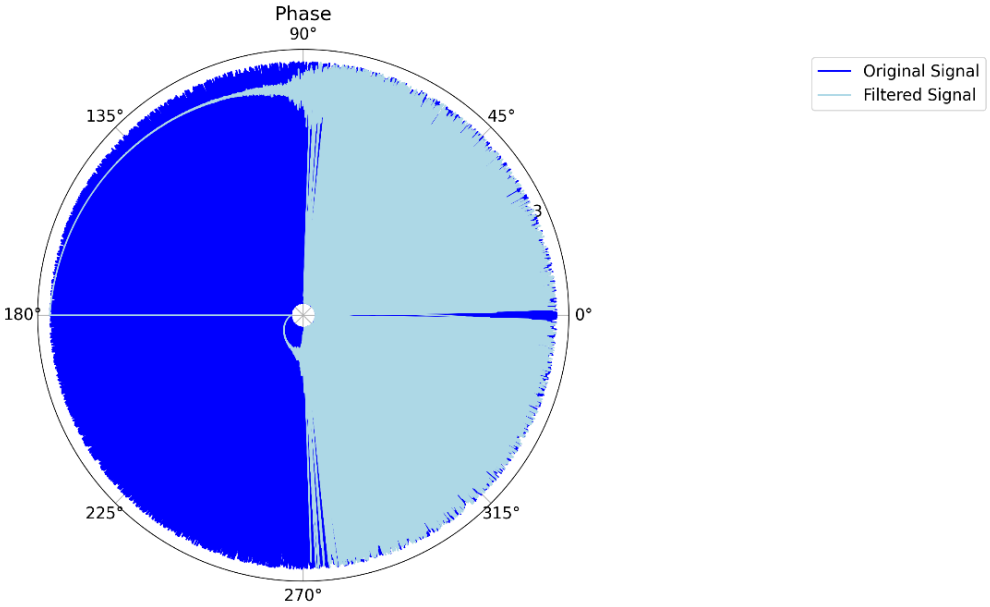


Figure 5-53: Phase of Original and Filtered Signal

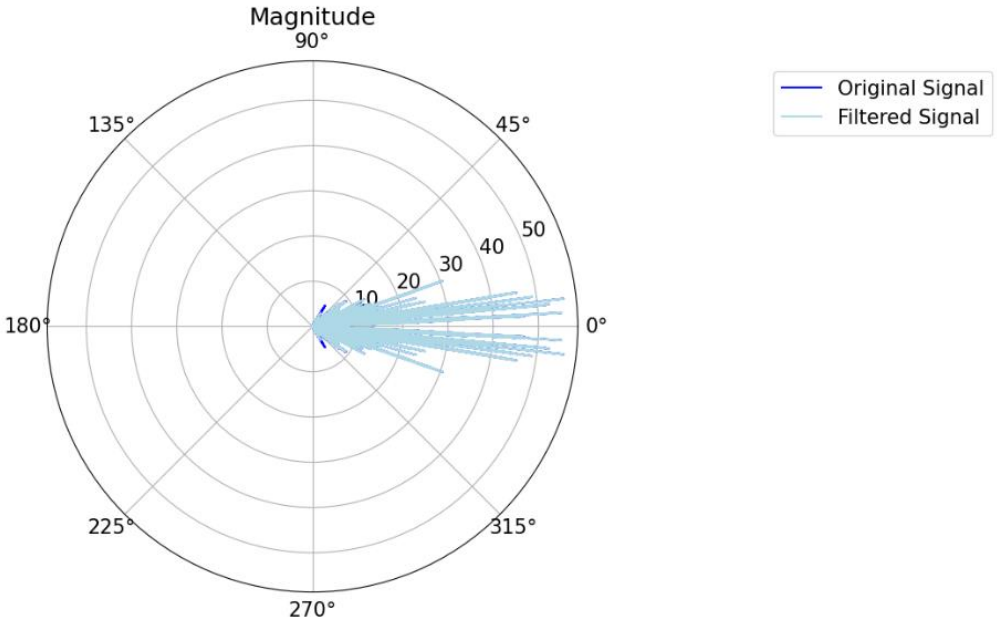


Figure 5-54: Magnitude Plot of Original and Filtered Signal

Applying a bandpass filter to a signal induces changes in both its phase and magnitude characteristics which can be seen from above plots. The magnitude response of the filter dictates the alteration in amplitude across different frequency components, allowing frequencies within the passband to pass relatively unaltered while attenuating those outside. Concurrently, the phase response introduces shifts in the temporal relationship between signal components across frequencies, potentially leading to phase distortion which can be seen in the plot above. These alterations are collectively represented above.

5.9.4 Dataset Augmentation

A. Frequency Masking

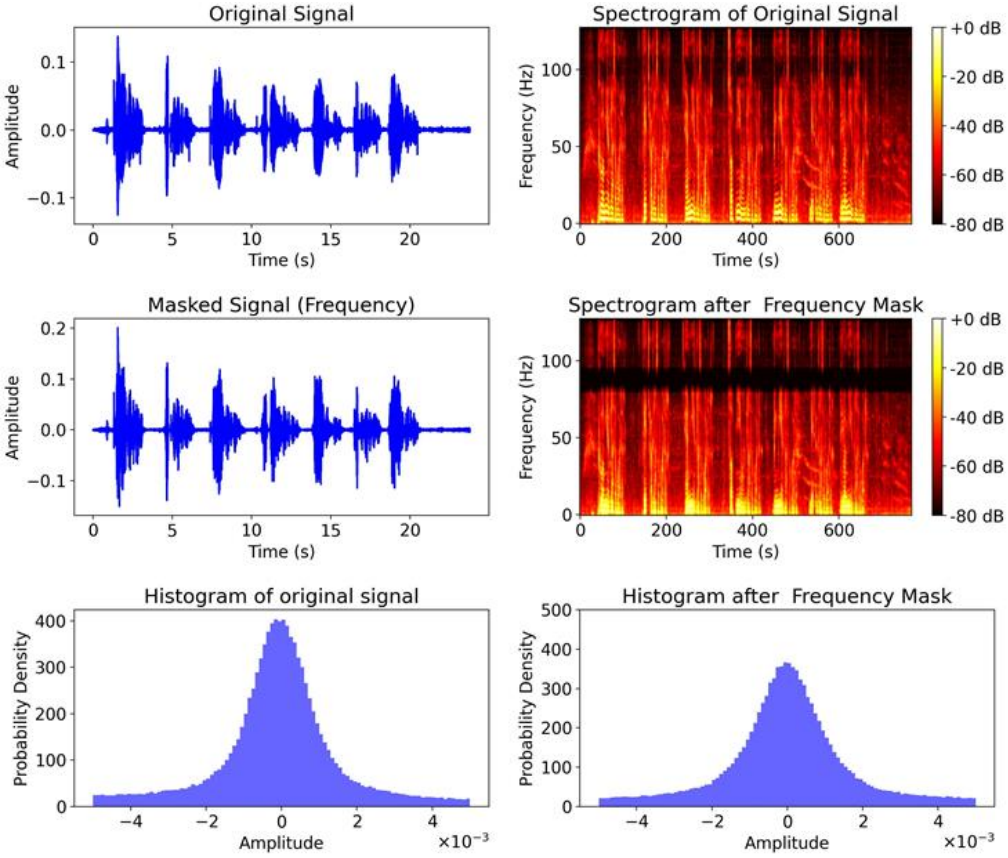


Figure 5-55: Visualizations after Frequency Masking

The plot illustrates the effects of frequency masking on a signal, influencing both its time and frequency domains. Initially, the original signal exhibits a diverse range of frequencies, evident from its waveform and spectrogram. However, with frequency masking applied, certain frequency components within the signal are attenuated or removed, leading to alterations in both domains. In the time domain, the waveform exhibits variations in amplitude or shape corresponding to the masked frequency components. Meanwhile, in the frequency domain, the masked frequency components are suppressed, resulting in a reduction or elimination of specific spectral features in the spectrogram. These changes are visually apparent in the modified waveform and spectrogram, highlighting the influence of frequency masking on both the temporal and spectral characteristics of the signal. Additionally, histograms before and after augmentation offer insights into the distribution of signal amplitudes or frequency

intensities. Before augmentation, the histogram displays a relatively uniform distribution. However, after applying frequency masking, the histogram exhibits blank regions corresponding to the masked frequency components, indicating localized reductions in amplitude or frequency intensity.

B. Gaussian Noise

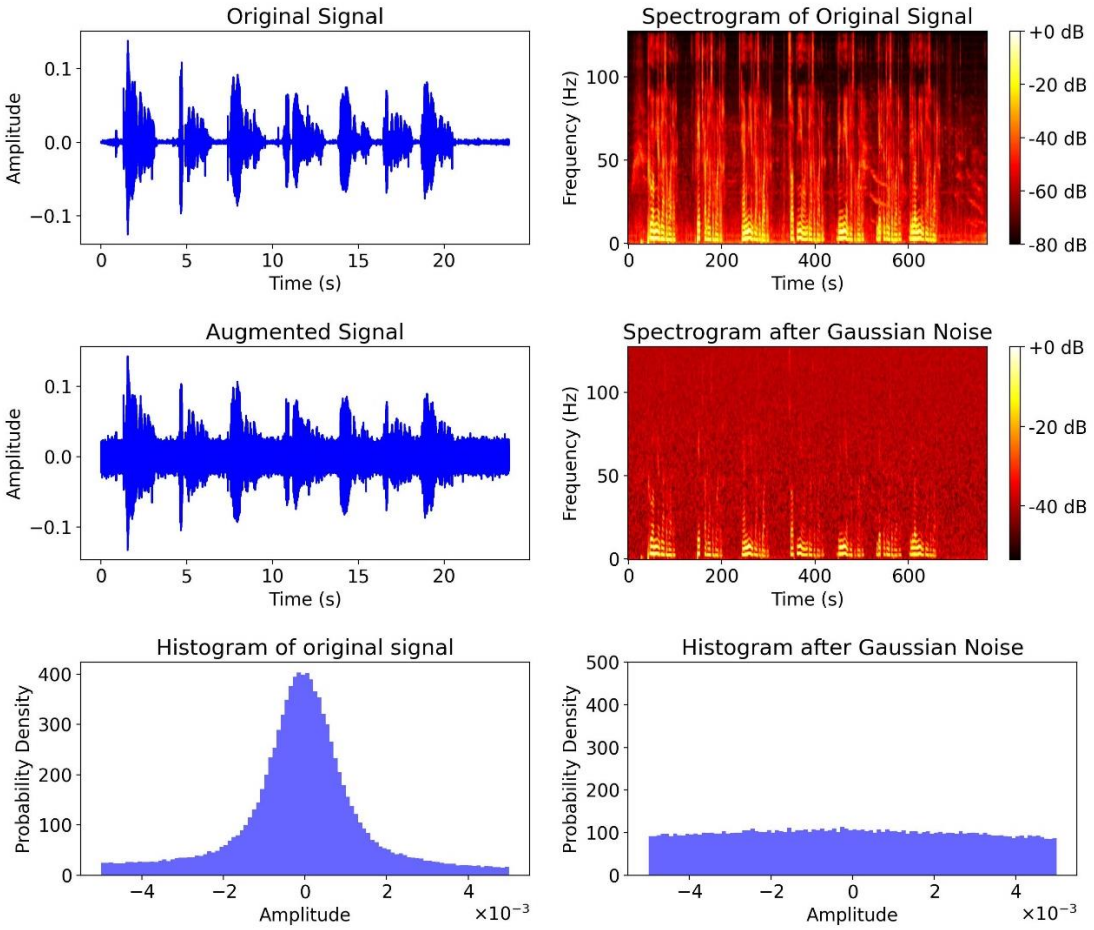


Figure 5-56: Visualizations after Gaussian Noise

The above plot illustrates the impact of adding Gaussian noise to a signal, affecting both its time and frequency domains. Initially, the original signal exhibits a diverse range of frequencies, as evident in spectrogram. However, upon introducing Gaussian noise, the signal's waveform becomes distorted with random fluctuations in amplitude over time. Simultaneously, in the frequency domain, the noise spreads across a wide range of frequencies, obscuring the original spectral components and reducing the signal-to-noise ratio.

C. Pitch Shift

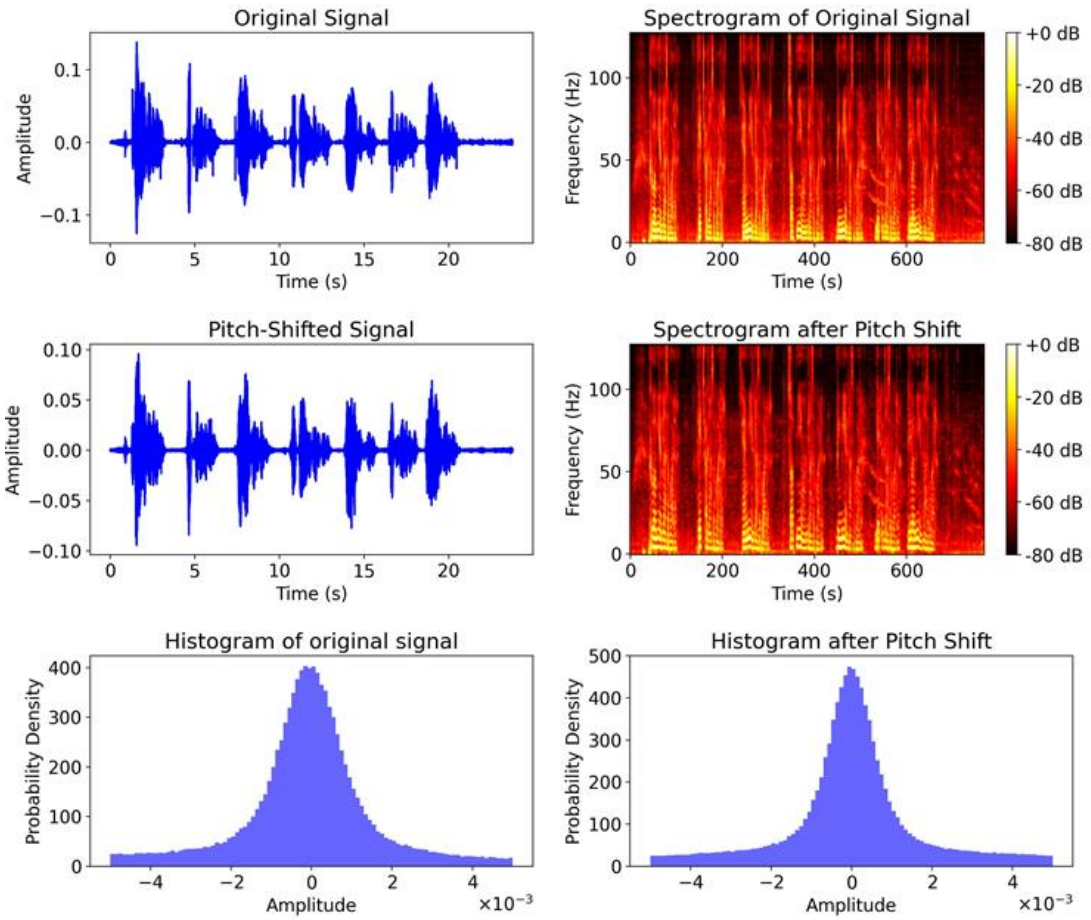


Figure 5-57: Visualizations after Pitch Shift

The plot illustrates the impact of pitch shifting on a signal, affecting both its time and frequency domains. Initially, the original signal exhibits a diverse range of frequencies, as evidenced by its waveform and spectrogram. However, with pitch shifting applied, alterations occur in both domains. In the time domain, the waveform undergoes compression accordingly to the magnitude of the pitch shift, altering the temporal structure of the signal without changing its overall duration. Meanwhile, in the frequency domain, the spectral components shift either upwards lower pitch shift,. Consequently, the positions of the spectral features in the spectrogram are adjusted upwards. These changes are visually depicted in the modified waveform and spectrogram, showcasing the influence of pitch shifting on both the temporal and spectral characteristics of the signal. The histograms further emphasize the distribution of pitch-shift-induced changes across the signal, providing additional insights into the alterations in both domains.

D. Random Gain

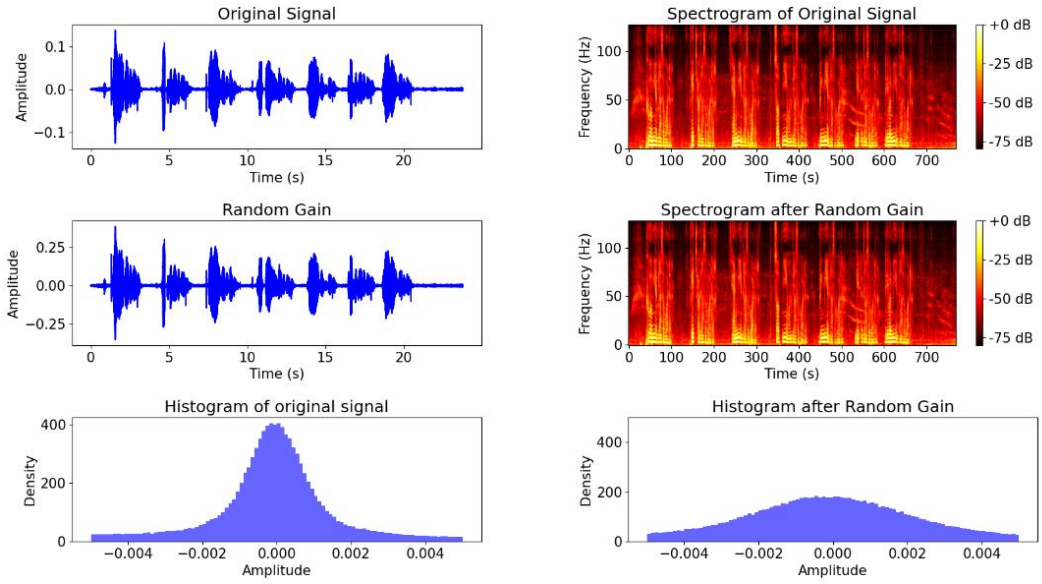


Figure 5-58: Visualizations after Random Gain

The plot illustrates the effect of random gain variation on a signal, influencing both its time and frequency domains. When random gain variation is introduced, changes occur in both domains. In the time domain, the waveform undergoes amplitude fluctuations over time due to the random gain adjustments. These fluctuations lead to variations in the signal's overall amplitude without altering its temporal structure. In the frequency domain, the random gain variation affects the amplitude of the spectral components across different frequency bands, resulting in changes to the signal's frequency content. These alterations are visually represented in the modified waveform and spectrogram, highlighting the influence of random gain variation on both the temporal and spectral characteristics of the signal. The histograms further illustrate the distribution of gain-induced changes across the signal, providing additional insights into the alterations in both domains.

E. Time Masking

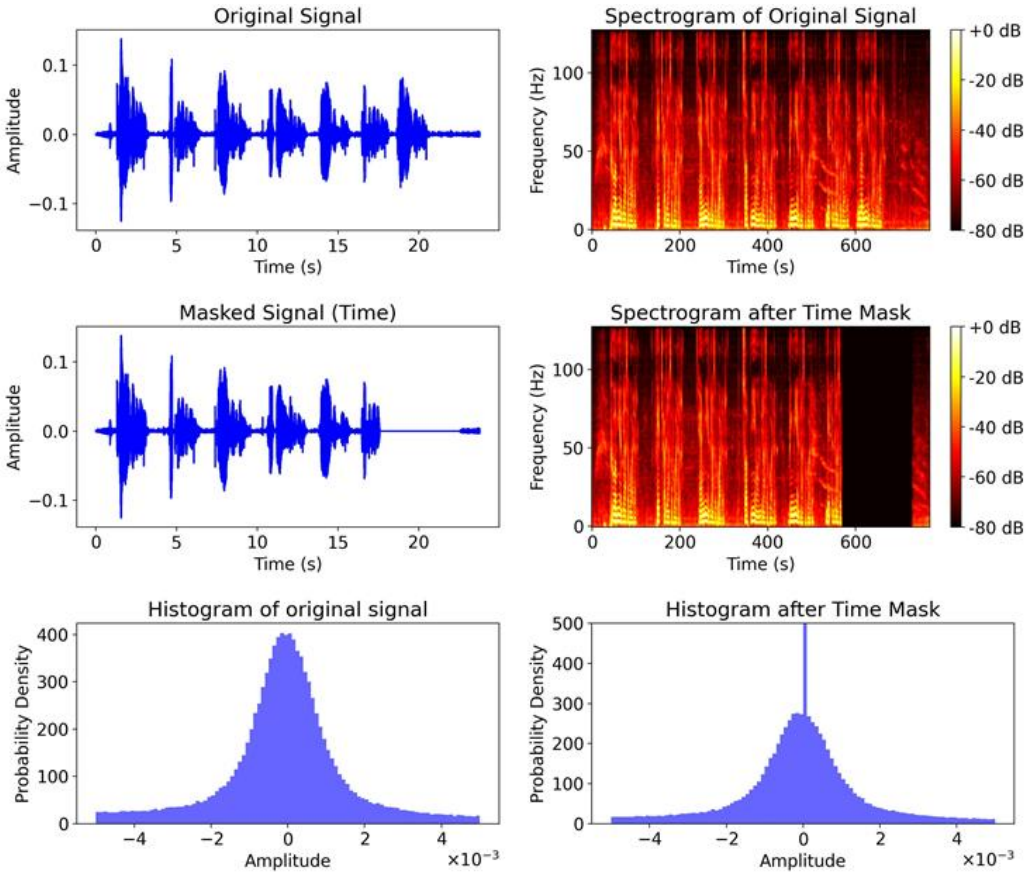


Figure 5-59: Visualizations after Time Masking

With time masking applied, certain segments of the signal's temporal structure are masked, leading to alterations in both domains. In the time domain, specific segments of the waveform are temporally suppressed, resulting in temporary silence or reduced amplitude. Meanwhile, in the frequency domain, the masked segments cause specific frequency components to be attenuated or removed, thereby modifying the spectral characteristics of the signal. These changes are visually apparent in the modified waveform and spectrogram, illustrating the influence of time masking on both the temporal and spectral attributes of the signal. Additionally, before augmentation, the histogram may displays relatively higher density. However, after applying time masking, the histogram exhibit decreased density because of the masked segments, indicating the reduction in amplitude or frequency intensity. This analysis underscores the impact of time masking on signal characteristics.

F. Time Stretch

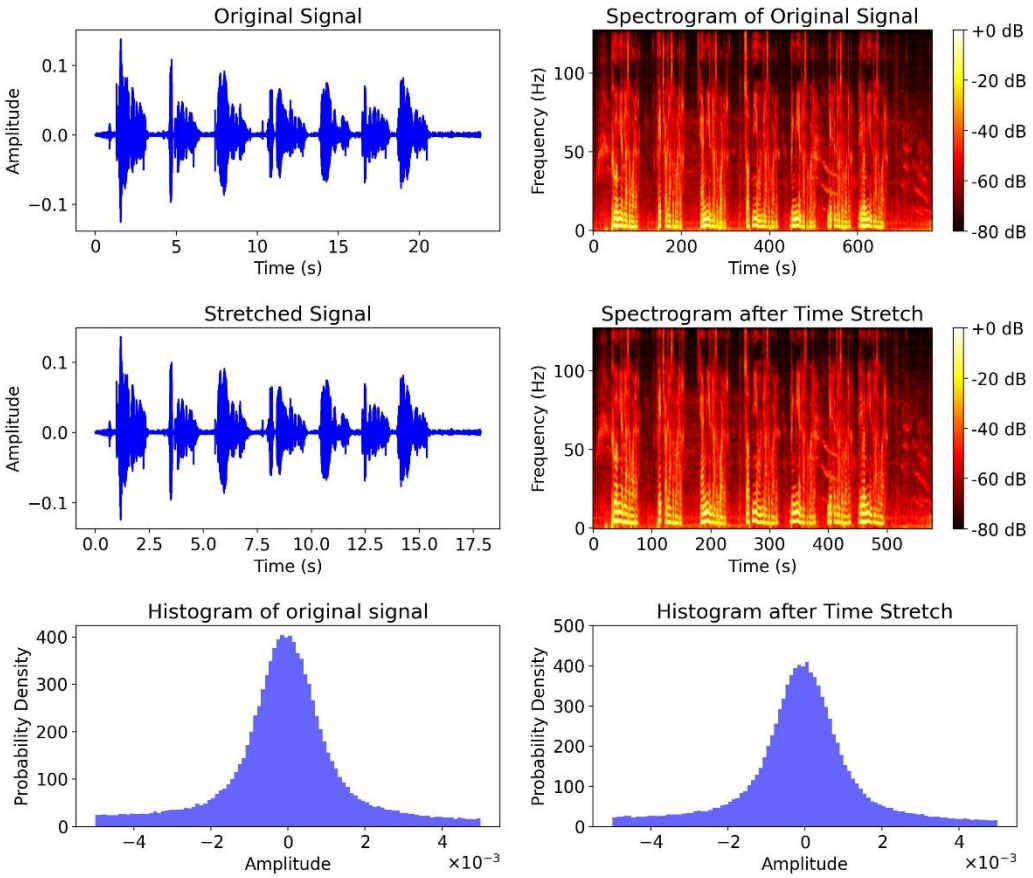


Figure 5-60: Visualizations after Frequency Masking

With time stretching applied, changes occur in both domains. In the time domain, the temporal structure of the waveform is altered, resulting in a compression or expansion of the signal's duration. This modification leads to a distortion of the temporal characteristics, with shorter or longer intervals between waveform peaks and troughs. Meanwhile, in the frequency domain, time stretching affects the distribution of spectral components, causing a compression or expansion of the frequency content. As a result, the positions of spectral features in the spectrogram are adjusted accordingly. These alterations are visually depicted in the modified waveform and spectrogram, highlighting the influence of time stretching on both the temporal and spectral attributes of the signal. Before augmentation, the histogram has relatively uniform distribution. However, after applying time stretching, the histogram exhibits shifts or stretches, indicating changes in the distribution of amplitude or frequency intensity across the signal. This analysis underscores the impact of time stretching on signal characteristics.

5.10 Fine Tuning Speech Recognition Model

5.10.1 Layer Freezing

Layer freezing in machine learning models is a technique employed during training, particularly in transfer learning or fine-tuning scenarios. It involves preventing certain layers, typically the lower or feature extraction layers, from being updated with new weights or gradients. By "freezing" these layers, their learned representations remain fixed, while only the weights of the unfrozen layers, usually the higher-level or task-specific layers, are adjusted during training. This strategy accelerates training convergence and helps mitigate overfitting, especially when dealing with small or similar datasets compared to the original pre-training data.

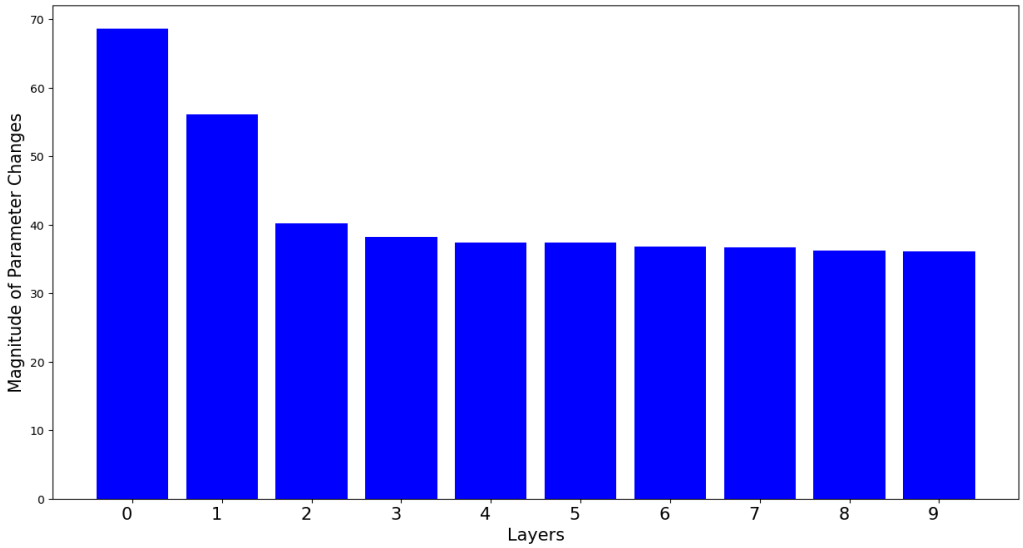


Figure 5-61: Change in Magnitude of Layers

Table 5-17: Layer Names

Layer	Layer name
0	Pre encoder layer
1	Decoder layer weight
2	16th Feed forward layer weight
3	7th Feed forward layer weight
4	5th Feed forward layer weight
5	6th Feed forward layer weight
6	7th Feed forward layer linear weight
7	6th Feed forward layer linear weight
8	4th Feed forward layer linear weight
9	17th Feed forward layer linear weight

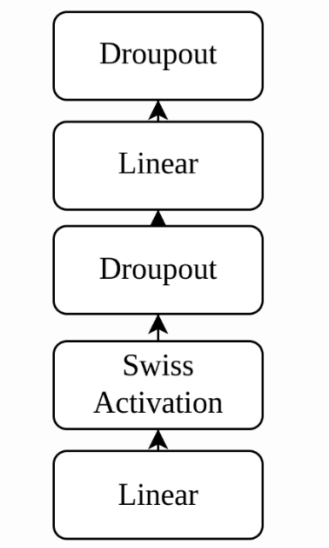


Figure 5-62: Feed Forward Layer

This layer received significant updates during finetuning. This layer, designed to extract high-level representations, may have required significant adjustments to better capture the nuances present in the specific dataset used for fine-tuning.

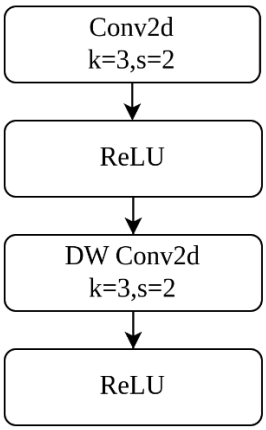


Figure 5-63: Pre-encoder Layer

This layer received the most updates among all layers. The pre-encoder layer receiving the most updates during fine-tuning could be attributed to several factors. Firstly, the pre-encoder layer plays a crucial role in extracting initial representations from the input data before passing them through subsequent layers. Therefore, during fine-tuning, the model might have prioritized adjusting this layer to better capture the specific characteristics and patterns present in the new dataset.

Analyzing the magnitude of updates for different weight parameters after fine-tuning provides valuable insights into the adaptation process of the pretrained speech recognition model to a specific task or dataset. These insights can inform further fine-tuning strategies, model architecture modifications, or areas of focus for improving performance.

A. Encoder Freezing

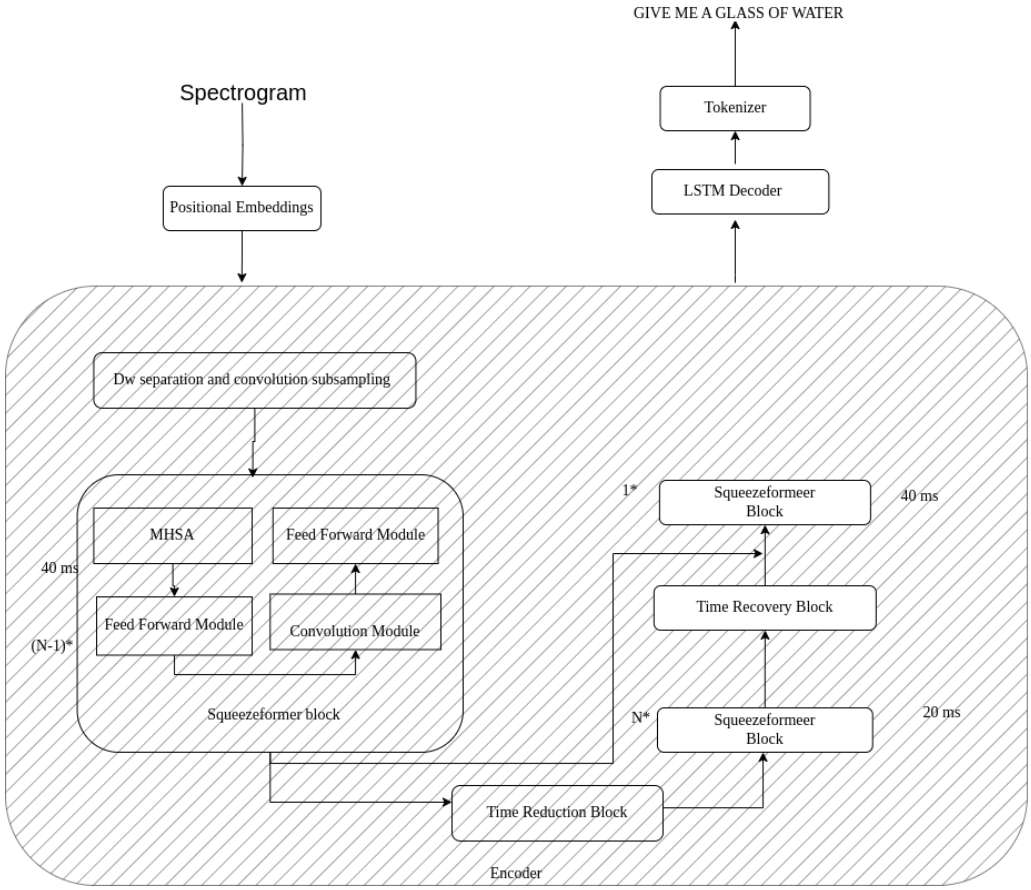


Figure 5-64 : Speech Recognition Architecture Showing Encoder Freezing

In the above figure the encoder is represented as hatched and the parameters of the encoder component are fixed or "frozen" during training, while only the decoder parameters are updated. This approach is often employed in transfer learning scenarios where the encoder has been pre-trained on a large dataset for a related task. By freezing the encoder, its learned representations are preserved, allowing the model to focus on learning the decoding process specific to the new task or dataset. Encoder freezing can lead to faster convergence during training, especially when dealing with limited data

or fine-tuning pre-trained models.

B. Decoder Freezing

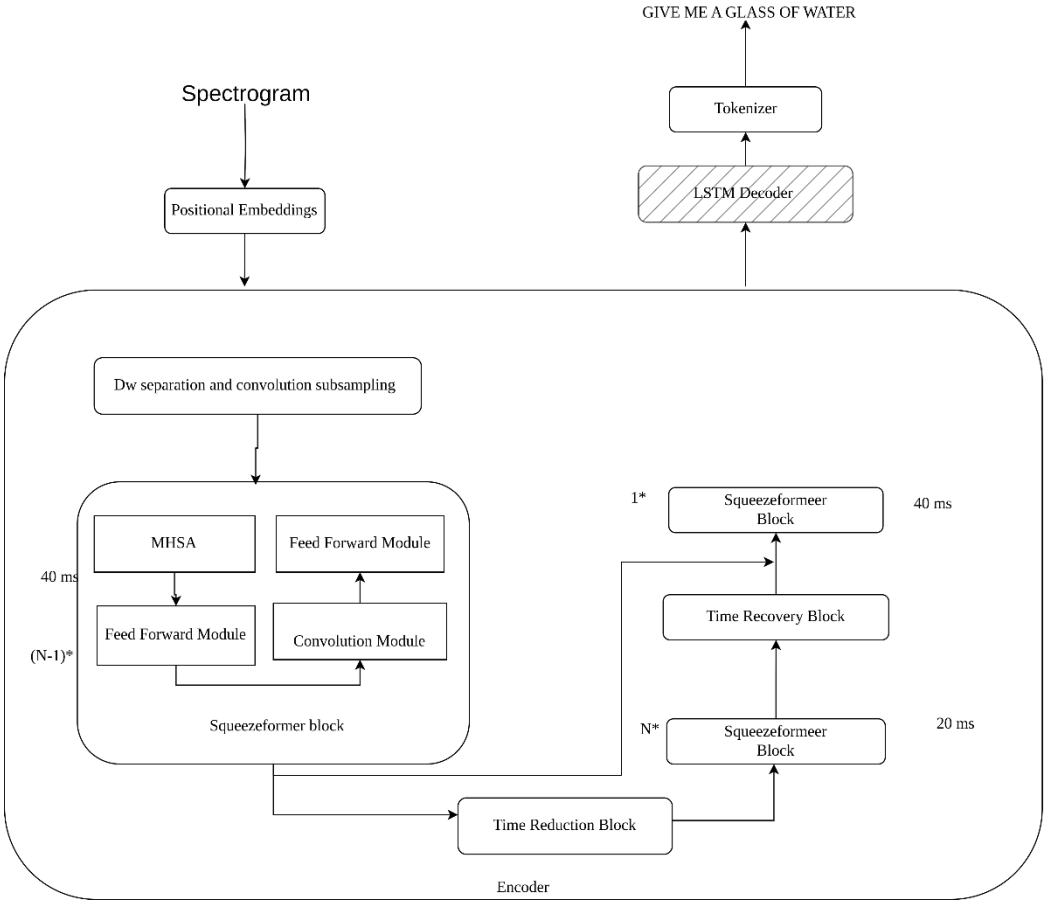


Figure 5-65: Speech Recognition Architecture Showing Decoder Freezing

The above figure shows the speech recognition architecture with decoder frozen. Parameters of the decoder component are kept fixed during the training process, while only the encoder parameters are updated. This approach is often employed in scenarios where the decoder needs to be adapted to a specific task such as updating vocabulary while leveraging pre-trained encoder representations. By freezing the decoder, the model focuses on fine-tuning the decoding process without altering the learned representations from the pre-trained encoder. Decoder freezing can expedite training and enhance model performance, particularly when dealing with limited data or fine-tuning pre-trained models.

5.10.2 Hyperparameters

Unlike model parameters, which are learned from data during training, hyperparameters need to be set before training begins and are not updated during the training process. Some relevant hyperparameters, description and the actual value used are tabulate below:

Table 5-18: Some Relevant Hyperparameters

Hyperparameters	Description	Values
Learning Rate	The step size at which the model adjusts its parameters during training. A higher learning rate may lead to faster convergence, but it can also cause instability and overshooting.	$1 \cdot e^{-3}$
Dropout Rate	The probability of dropping out units during training, used as a regularization technique to prevent overfitting.	0.1
Batch Size	The number of data samples used in each iteration of the training process. A smaller batch size may lead to noisier gradients but can speed up training with parallel processing.	50
Number of Epochs	The number of times the entire dataset is used to train the model. Setting this too low may result in under fitting, while setting it too high may lead to overfitting.	30

5.10.3 Optimizer Settings

In a speech recognition model, the choice of optimizer and its settings are crucial for efficient training and convergence. The optimizer determines how the model's parameters are updated during the learning process. Different optimizers use various algorithms to find the optimal set of parameters that minimize the model's loss function.

Here's a table describing common optimizers and their settings for speech recognition model:

Table 5-19: Optimizers and their Settings

Optimizer	Learning Rate	Beta1	Beta2	Epsilon
Adam	$1 \cdot e^{-3}$	0.9	0.99	$1 \cdot e^{-8}$

5.11 Image Dataset Exploration

5.11.1 Pre-trained Dataset Discussion

The COCO dataset consists of 200,000 meticulously annotated images, making it a pivotal resource for training and evaluating object detection models. It encompasses 80 distinct object categories, ranging from common entities like cars and bicycles to more nuanced items such as umbrellas and handbags, providing a comprehensive dataset for model training. Annotations, including bounding boxes, ensure precise object localization, crucial for training accurate models.

Table 5-20: Image and Object Count in Pretrained Dataset (COCO-2017)

Subset	Object Count	Image Count
Train2017	860,001	118,287
Val2017	36,553	5,000
Test2017	147905	20,288

5.11.2 Fine-tuning Datasets

A custom YOLOv8 model was trained to detect and estimate the position of different objects. Model was fine-tuned on the nano model of YOLOv8, utilizing datasets collected from both CoppeliaSim and physical environments.

A. Fine-tuning in Simulation Dataset

A total of 175 datasets were collected, and after augmentation final dataset comprised 384 images. These datasets were then separated into training and validation sets,

consisting of 353 training images and 31 validation images. Finally, the model was trained on this dataset for 80 epochs.

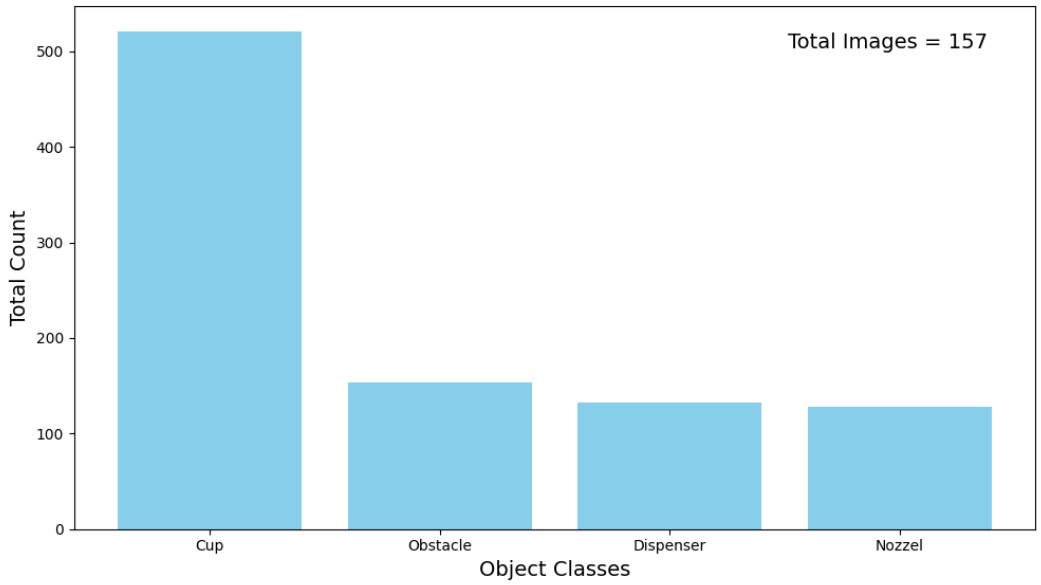


Figure 5-66: Object Distribution before Augmentation (Simulation)

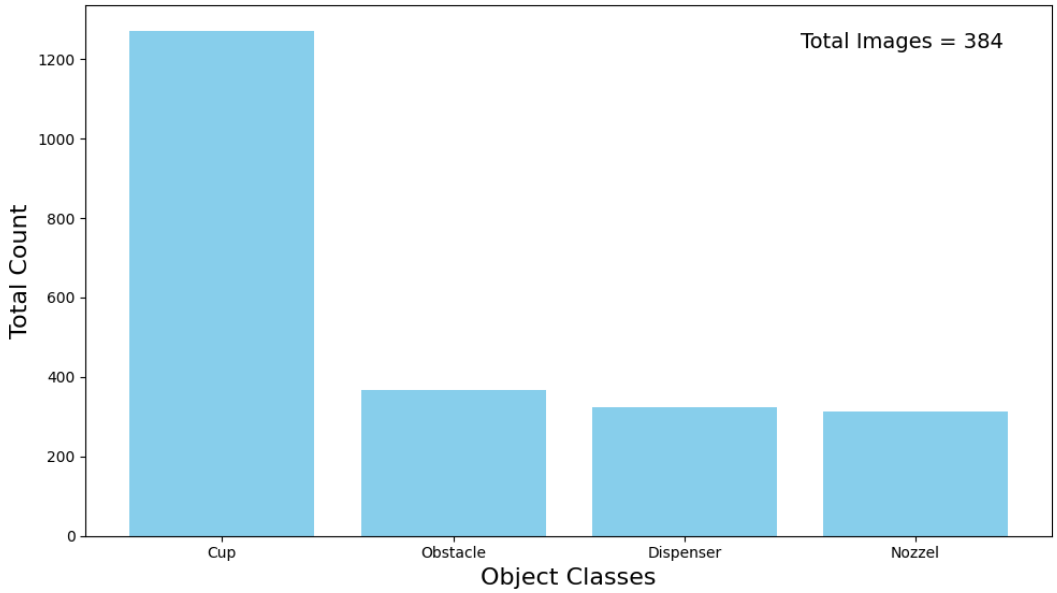


Figure 5-67: Object Distribution after Augmentation (Simulation)

B. Fine-tuning in Physical Dataset

A total of 194 datasets were collected, and after applying vertical flip the final dataset comprised 297 images. These datasets were then separated into training and validation sets, consisting of 249 training images and 48 validation images. Finally, the YOLOv8 model was trained on this dataset for 80 epochs.

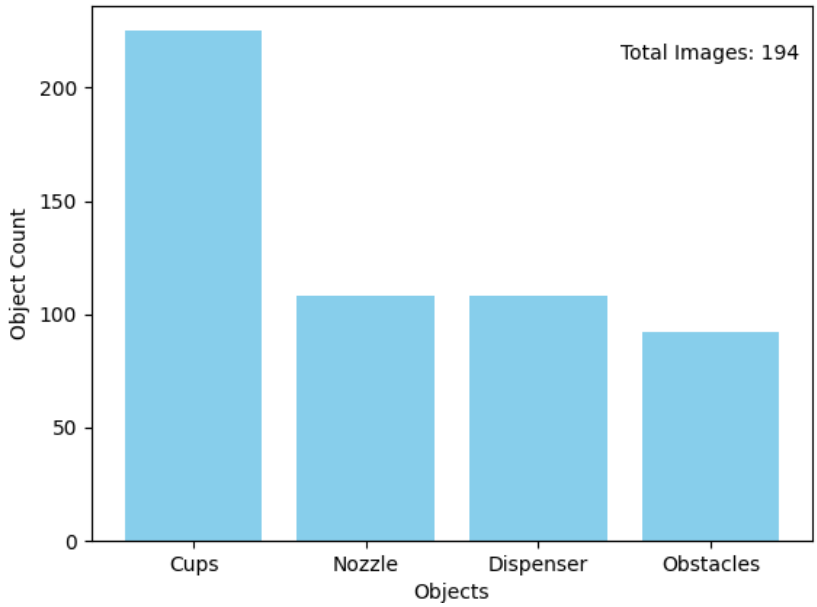


Figure 5-68: Object Distribution before Augmentation (Physical)

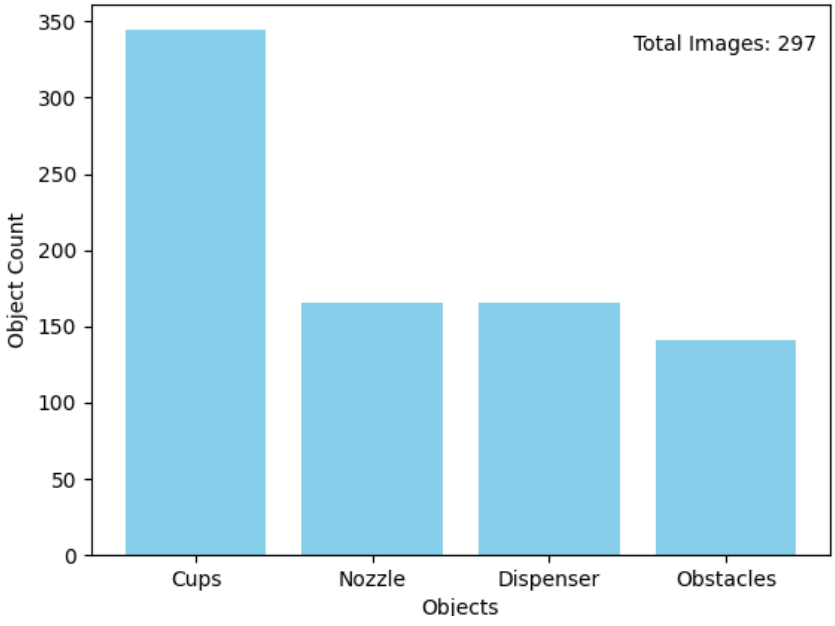


Figure 5-69: Distribution of Objects after Augmentation (Physical)

5.11.3 Raw Glimpse of Objects Utilized for Dataset

A. Simulation Objects

The objects designated for simulation in CoppeliaSim are modeled in Fusion 360. These objects serve as the dataset for simulation purposes.

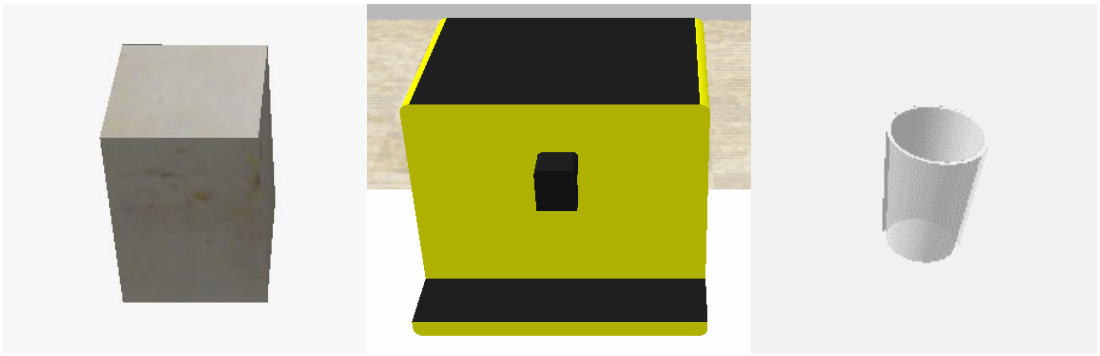


Figure 5-70: Objects in Simulation Environment (Obstacle, Dispenser and Cup)

B. Physical Objects

In order for the model to function within a physical system, the datasets must also manifest as physical objects.

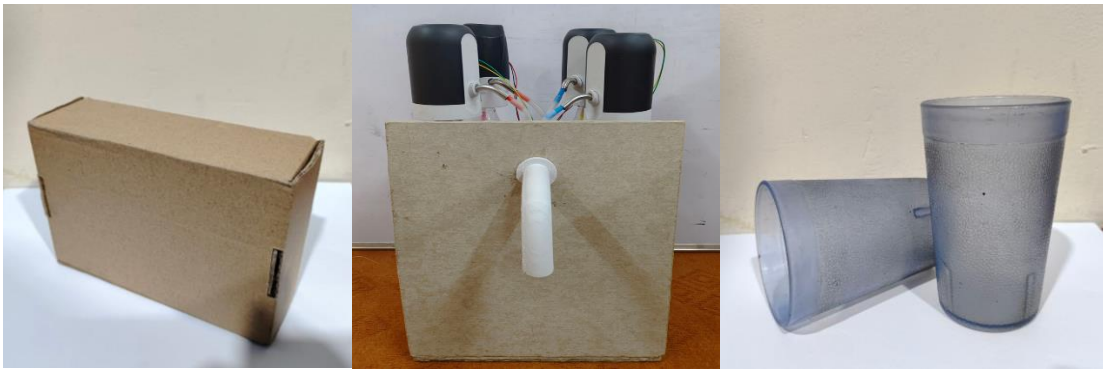


Figure 5-71: Objects in Physical Environment (Obstacle, Dispenser and Cup)

5.11.4 Hyperparameters of Object Detection Model

Table 5-21: Hyperparameters of YOLOv8 Training

Hyperparameters	Description
Epochs	Number of epoch used to train the model (80)
Batch	Number of images in each batch (16)
imgsz	Size of input image while training the model (640)
Optimizer	Optimizer AdamW (lr=0.001111, momentum=0.9)
Learning Rate	The learning rate of while training (0.001111)

5.12 Dispenser Working Mechanism

The dispenser system comprises four bottle holders, each equipped with a unique water pump attached to its neck, enabling the dispensing of individual drinks. When a client issues a command for a specific drink, the speech recognition model generates a transcription of the issued voice command. This transcription serves as the command for the dispenser system to dispense the requested drink.

It's essential to trigger the dispenser system when the robotic arm positions the empty glass below the corresponding nozzle. To achieve this synchronization, a switching circuit is employed, comprising four transistors. These transistors are responsible for activating the pump associated with the commanded drink at the required moment.

5.12.1 Switching Circuit for Dispenser

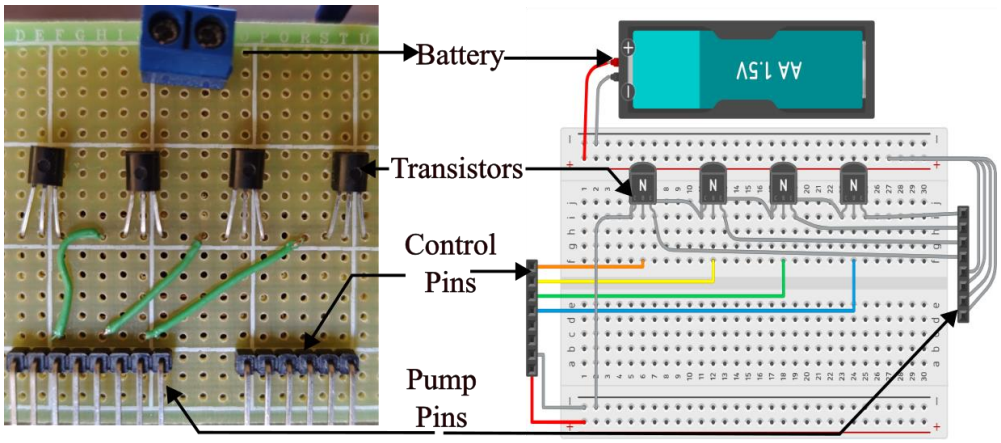


Figure 5-72: Switching Circuit for Dispenser

The circuit consists of four transistors, each corresponding to an individual pump. The commanded transcription serves as the command signal to activate the specific transistor associated with the requested drink. When the glass is positioned below the nozzle, the command is transmitted to the dispenser system through control pins, triggering the corresponding transistor and activating the pump trough pump pins to dispense the requested drink.

5.13 Coordinate Mapping and Position Estimation

In applications, such as robotic vision systems or augmented reality, there is a need to map coordinates from a camera frame to the coordinate system of a physical environment.

5.13.1 Source and Destination Coordinate Collection

In the context of the presented figures, the left figure delineates the camera frame of the environment, represented in pixels, with associated camera coordinates. Conversely, the right figure illustrates the real-world coordinates of the same system, expressed in centimeters.

A. Coordinate Collection for Simulated Environment

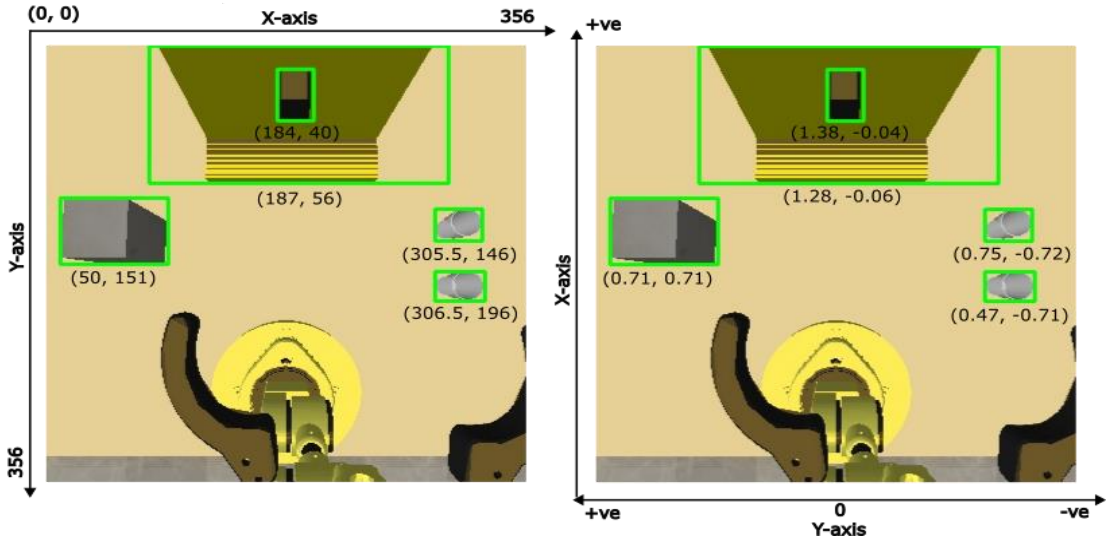


Figure 5-73: Camera and Simulation Coordinate System

Table 5-22: Source and Destination Coordinate (Simulation)

Source Coordinates (pixel)		**Destination Coordinates(unit)**	
x-axis	**y-axis**	**x-axis**	**y-axis**
81.5	102.0	1.0	0.54
273.5	189.0	0.50	-0.52
313.5	106.5	0.9	-0.77
146	101.5	1.0	0.17
327.0	189.0	0.50	-0.82
29.0	184.0	0.52	0.82
323.0	284.0	0.0	-0.77
86.5	184.0	0.52	0.49
29.5	28.5	1.45	0.87

B. Coordinate Collection for Physical Environment

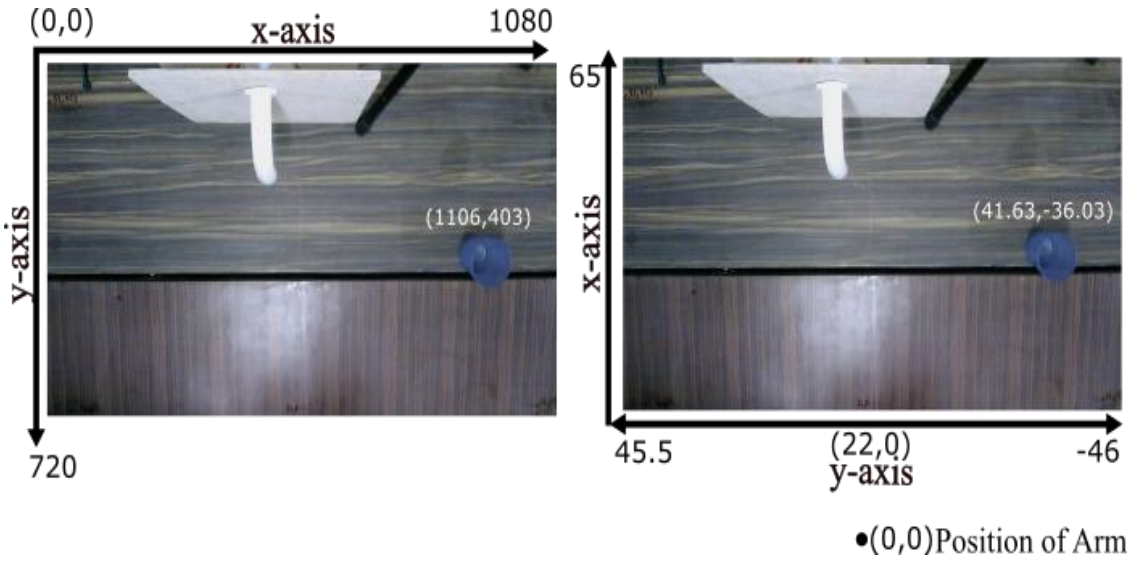


Figure 5-74: Camera and Physical Coordinate System

The origin of the robotic arm, positioned in the world coordinate system, resides outside the field of view (FOV) of the camera. Consequently, the origin point is not depicted within the figure.A total of 9 camera coordinates were meticulously obtained along with their corresponding real-world coordinates.

Table 5-23: Source and Destination Coordinate (Physical)

Source Coordinates (pixel)		**Destination Coordinates(cm)**	
x-axis	**y-axis**	**x-axis**	**y-axis**
5	80	59	45.5
612	75	59	0
1253	75	59	-46
31	450	42	45.5
632	444	42	0
1239	439	42	-46
45	707	22	45.5
637	703	22	0
1223	696	22	-46

5.13.2 Transformation Matrix Estimation

The coordinates acquired from the camera and the corresponding real-world points were utilized as input for the "findHomography" function provided by OpenCV. This process resulted in the derivation of the transformation matrix.

A. Transformation Matrix for Simulation

$$H = \begin{bmatrix} 9.69e-06 & -5.75e-03 & 1.63 \\ -5.97e-03 & -3.37e-08 & 1.06 \\ -6.63e-07 & 4.47e-04 & 1.00 \end{bmatrix} \quad \text{5-8}$$

B. Transformation Matrix for Physical Environment

$$H = \begin{bmatrix} -4.10e-04 & -6.12e-02 & 6.44e+01 \\ -7.12e-02 & 1.34e-03 & 4.42e+01 \\ 1.9e-08 & -1.38e-04 & 1.00 \end{bmatrix} \quad \text{5-9}$$

5.13.3 Coordinate Mapping for Drinking Glass

A. Mapping in Simulation

Following is the example of the camera coordinate of empty drinking glass from fig 7-64 (305, 146) being mapped into the simulation coordinate system.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 9.69e-06 & -5.75e-03 & 1.63 \\ -5.97e-03 & -3.37e-08 & 1.06 \\ -6.63e-07 & 4.47e-04 & 1.00 \end{bmatrix} \cdot \begin{bmatrix} 305.5 \\ 146 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.75 \\ -0.72 \\ 1 \end{bmatrix}$$

B. Mapping in Physical System

Following is the example of the camera coordinate of empty drinking glass from fig 7-65 (1106, 403) being mapped into the simulation coordinate system.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -4.10e-04 & -6.12e-02 & 6.44\text{e}+01 \\ -7.12e-02 & 1.34e-03 & 4.42\text{e}+01 \\ 1.9e-08 & -1.38e-04 & 1.00 \end{bmatrix} \cdot \begin{bmatrix} 1106 \\ 403 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 41.63 \\ -36.03 \\ 1 \end{bmatrix}$$

6. RESULTS AND ANALYSIS

This section includes results corresponding to different models of the project.

6.1 Oscilloscope Waveforms

The input and output of the A4988 was analyzed in the RIGOL DS1102E oscilloscope. Since it had only two channels the inputs and outputs were separately measured.

6.1.1 Input to the Servo

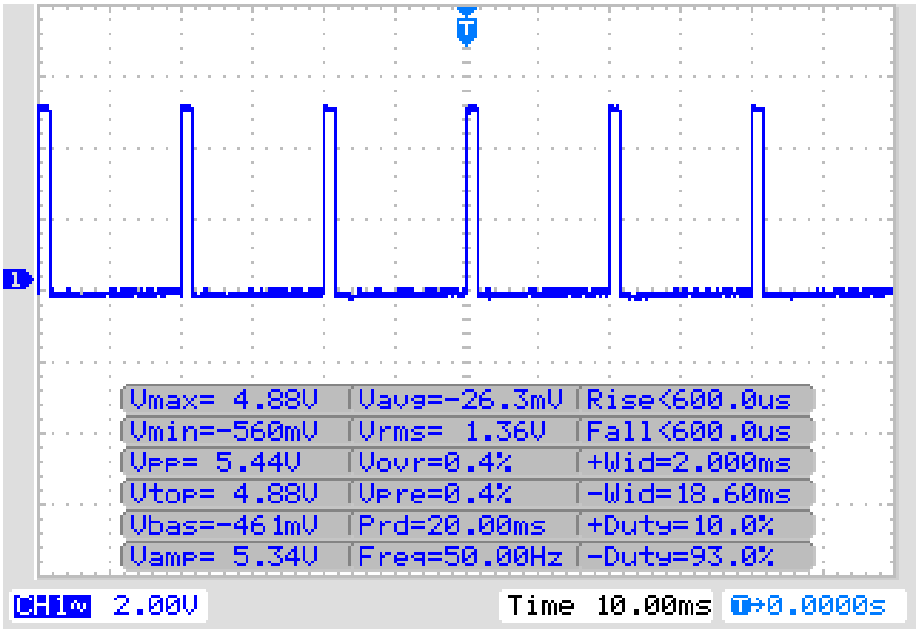


Figure 6-1: Input Signal for 90 Degree of Angle

Vpp stands for Volts peak-to-peak. It's a measurement of the voltage amplitude of a waveform, specifically the difference between its maximum positive and maximum negative peaks.

Vmax and Vtop both refer to the maximum positive voltage level reached by a waveform. They represent the peak voltage of the signal, typically measured from the zero-voltage reference point (baseline) to the highest positive peak.

Vamp stands for voltage amplitude. It represents the magnitude of the voltage variation in a waveform, usually measured from the baseline (zero voltage reference) to the peak (either positive or negative) of the waveform.

Vbas typically refers to baseline voltage which represents the voltage level of the horizontal line that serves as the reference for the waveform.

-Wid and +Wid are the most important values as it determines the angle the servo is supposed to rotate. In the above figure we have -Wid and +Wid as 18.60ms and 2ms respectively. This shows that the time period is approximately 20ms. Duty cycle is 10.0% and for this duty cycle the servo will have 90 degrees of rotation.

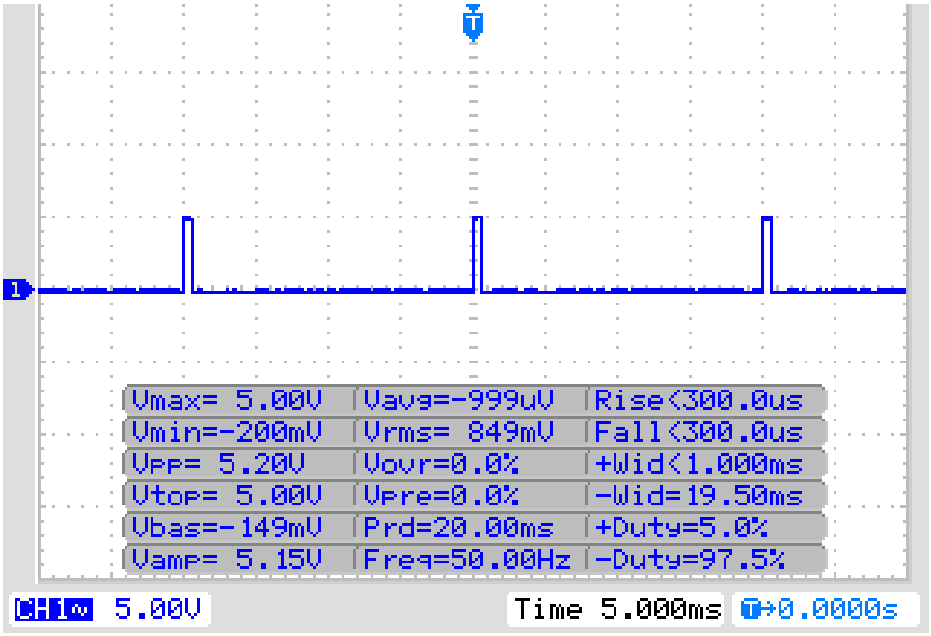


Figure 6-2: Input Signal for 0 Degree of Angle

In the above figure we have -Wid and +Wid as 19.50ms and <1ms respectively. This shows that the time period is approximately 20ms. Duty cycle is less than 5.0% and the corresponding angle is 0 degrees.

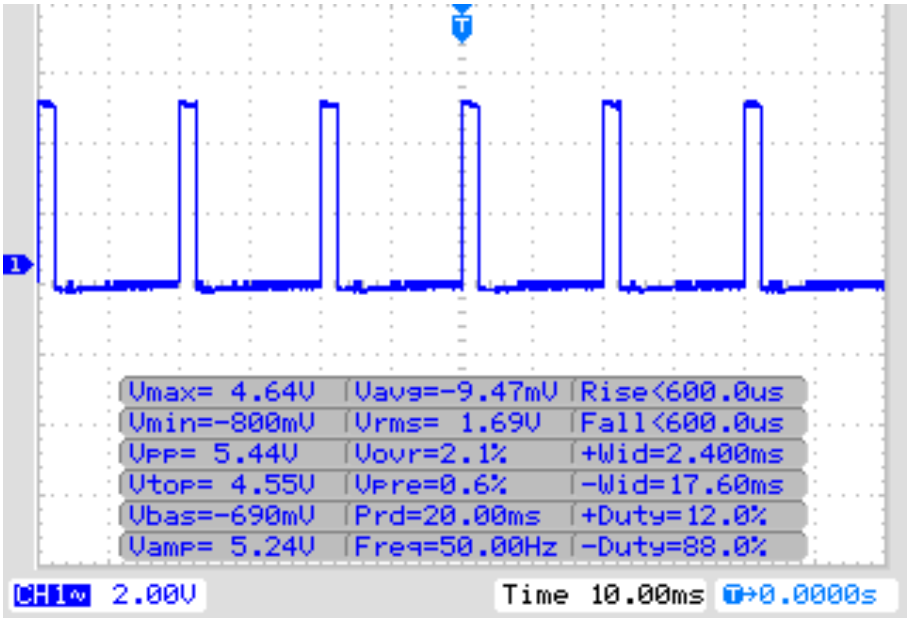


Figure 6-3: Input Signal for 180 Degree of Angle

In the above figure we have -Wid and +Wid as 17.60ms and 2.40ms respectively. This shows that the time period is approximately 20ms. Duty cycle is 12.0% and the corresponding angle is 180 degrees.

6.1.2 Input to the TB6600 from Controller

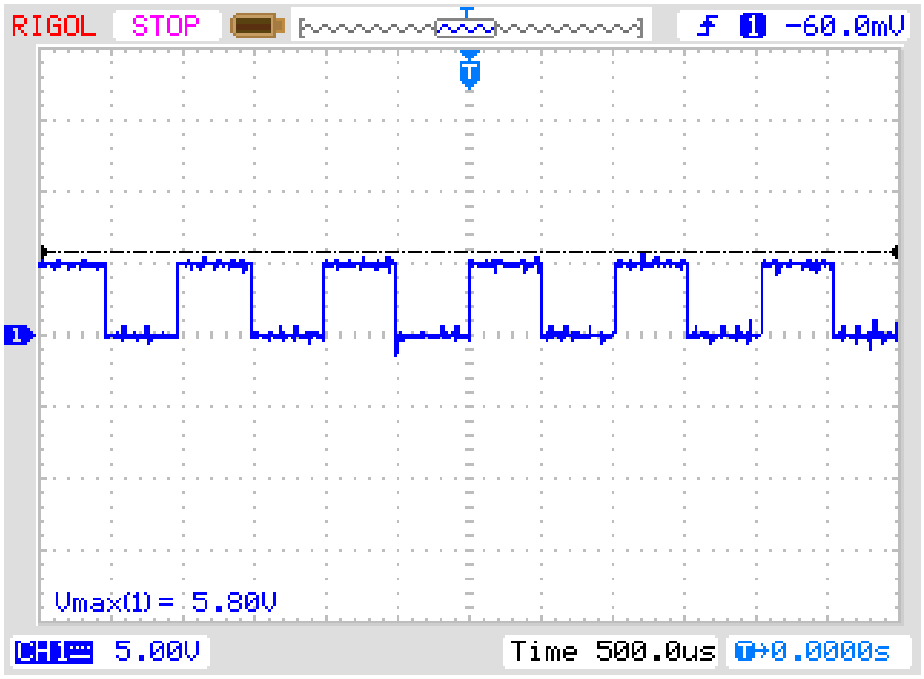


Figure 6-4: Stepping Signal from Arduino

Each horizontal division (larger box) was set to 500 microseconds whereas the vertical division was set to 5V. From the figure it can be deduced that the both on time and off time of the PWM is 500 microseconds making it the time period 1 millisecond. As both the on time and off time is equal the duty cycle of the input pulse is 50%.

$$Duty\ Cycle = \frac{On\ Time}{Time\ period} \times 100 \qquad 6\text{-}1$$

The driver detects the rising edge of the pulse to initiate step hence changing duty cycle will not change anything. Peak 5.80V voltage is seen which might be due to various noises and lack of isolation in the connection. The distortions are also caused by external noise.

6.1.3 Output from the TB6600

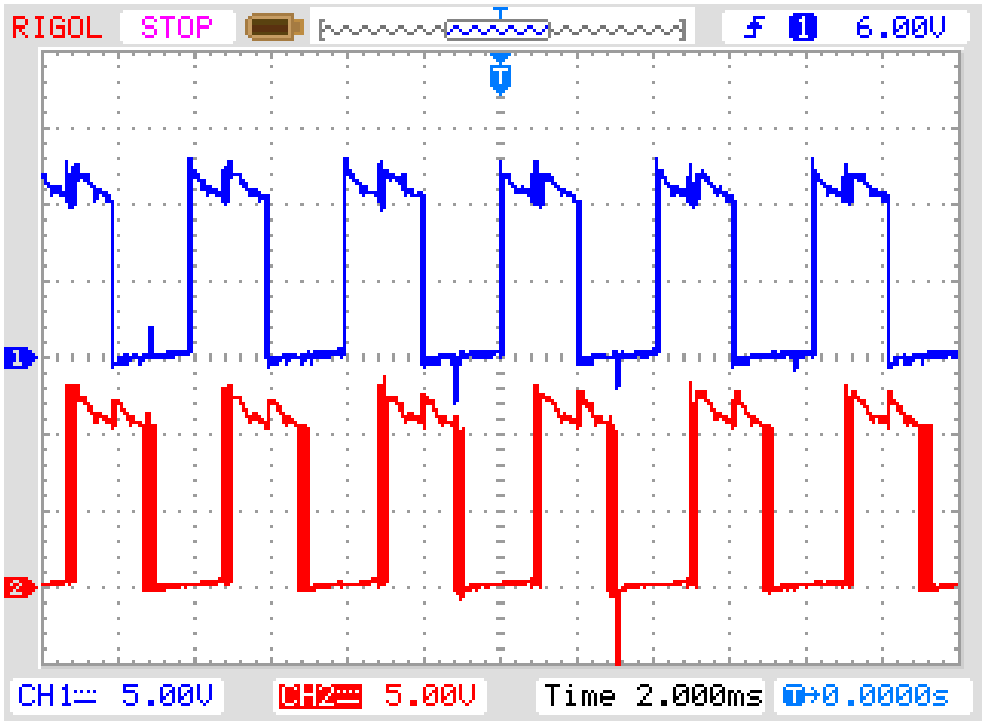


Figure 6-5: Output from Stepper Driver

Both channels of the oscilloscope were utilized to get the output of the A4988 which is supplied to the stepper motor. The four pins of the stepper motor are connected to the 1A, 1B (forms a coil) and 2A, 2B (forms a coil) pair. To get the output to the oscilloscope, one channel (blue) is connected to 1A and another channel is connected to 2A (red). The supply voltage is to the stepper driver is 10V. Both channels have

vertical division (larger box) of 5V and horizontal division of 2 milliseconds.

The sequence of energizing the coils A (blue channel) and B (red channel) can be explained by the following table.

Table 6-1: Excitation Table

S.N.	Coil A (V)	Coil B (V)
1.	10	10
2.	0	10
3.	0	0
4.	10	0

When the voltage is 10V the current flows from the connected point 1A or 2A to the other end of the coil 1B or 2B. When the voltage is 0V the terminal is at lower potential which allows the current to flow from the other end of the coil i.e., 1B and 2B. Hence when the voltage is 0 the current direction reverses.

6.2 Speech Recognition Performance

In this section, performance of the speech recognition model is evaluated, and the results are obtained to assess the accuracy and effectiveness of the model in recognizing speech based on the evaluation metrics used.

6.2.1 Evaluation Metrics

The speech recognition model's performance was assessed using two main metrics: Word Error Rate (WER) and Character Error Rate (CER). WER gauges the percentage of inaccurate words in the recognized transcriptions, while CER measures the percentage of incorrect characters. These evaluation metrics were employed to monitor the model's performance during both training and testing phases.

A. Training Loop

The training loop consisted of iteratively feeding the mini batches of spectrograms and their corresponding transcriptions into the speech recognition model. The model computed the forward pass, calculated the cross-entropy loss, and performed backward

propagation to compute gradients. The Adam optimizer updated the model parameters based on the gradients, iteratively improving the model's performance over epochs. Below are plots for different combination of training.

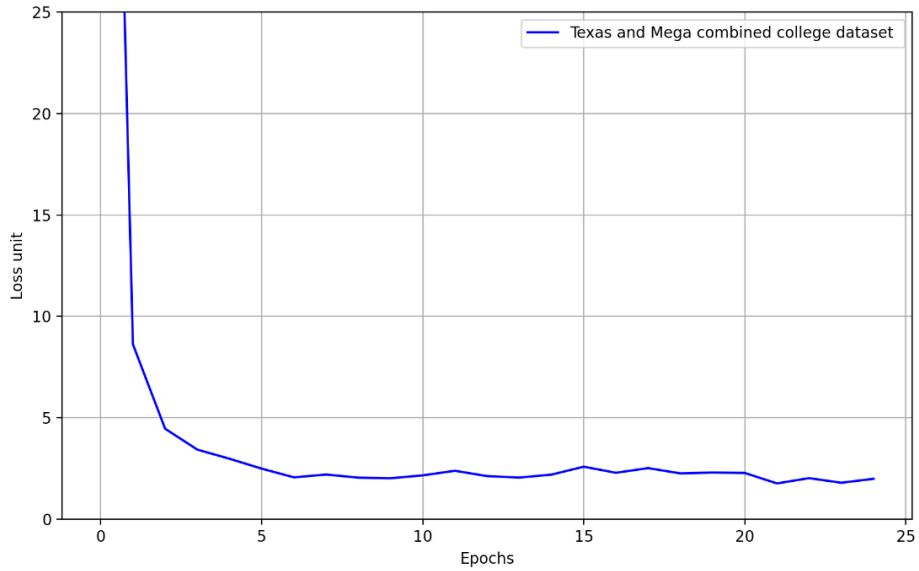


Figure 6-6: Training Loss Trend for Dataset from Texas and Mega School

The plot above illustrates the loss curve resulting from training the speech recognition model exclusively on the dataset gathered from young students attending Mega and Texas school. The loss consistently decreases up to a certain point.

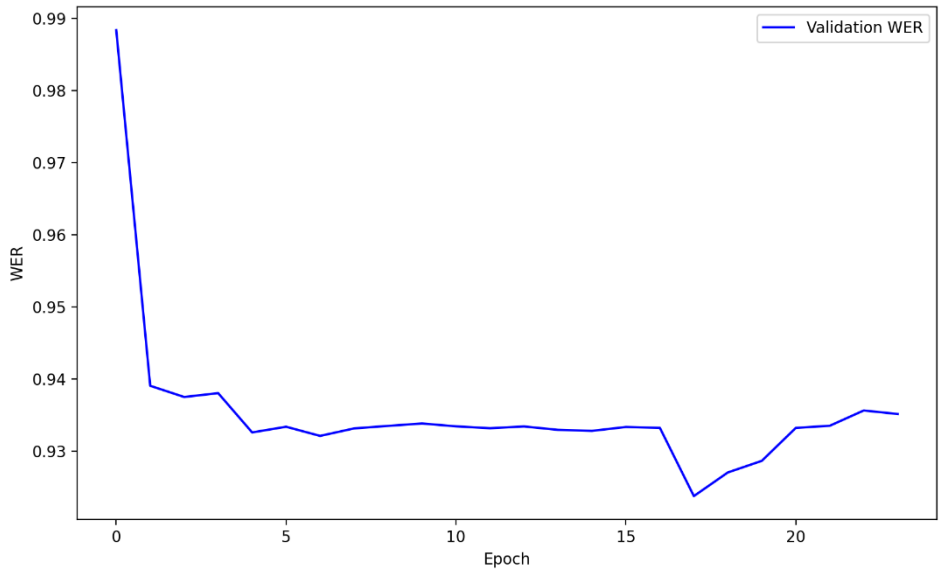


Figure 6-7: Validation WER of Texas and Mega School

The above plot depicts the training validation Word Error Rate (WER) of combined data from Texas and Mega schools. It is evident from the plot that the model is encountering challenges in converging to a lower WER.

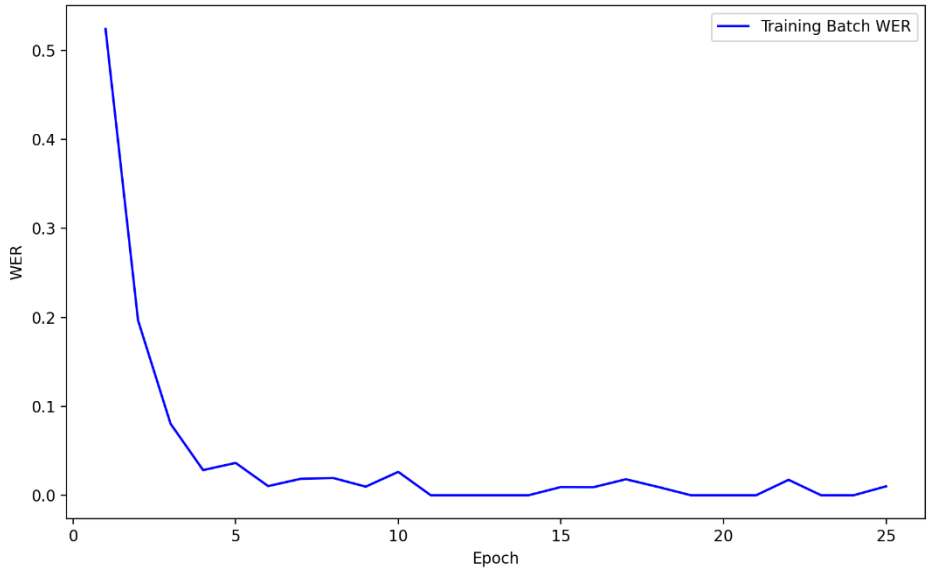


Figure 6-8: Training Batch WER of Texas and Mega School

The provided plot illustrates the training batch Word Error Rate (WER) of combined data from Texas and Mega schools. It is evident from the plot that the model has performed quite well, achieving a WER as low as 8%, which indicates satisfactory performance.

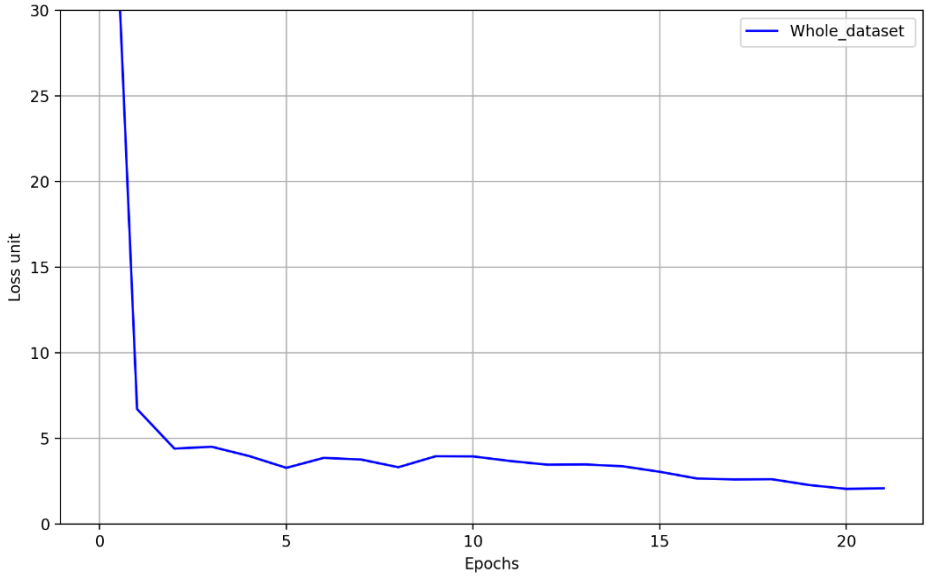


Figure 6-9: Whole Dataset Training Loss

The plot illustrating the training loss for the entire combined dataset, the loss curve showcases how the loss decreases over the course of training epochs. This decreasing trend is indicative of the model learning and improving its performance over time. The decreasing loss curve signifies that the model is effectively converging towards an optimal solution. However, it's important to monitor other metrics such as validation performance to ensure that the model generalizes well to unseen data. Overall, the decreasing trend in the training loss curve indicates that the model is learning and improving its performance, which is essential for successful training.

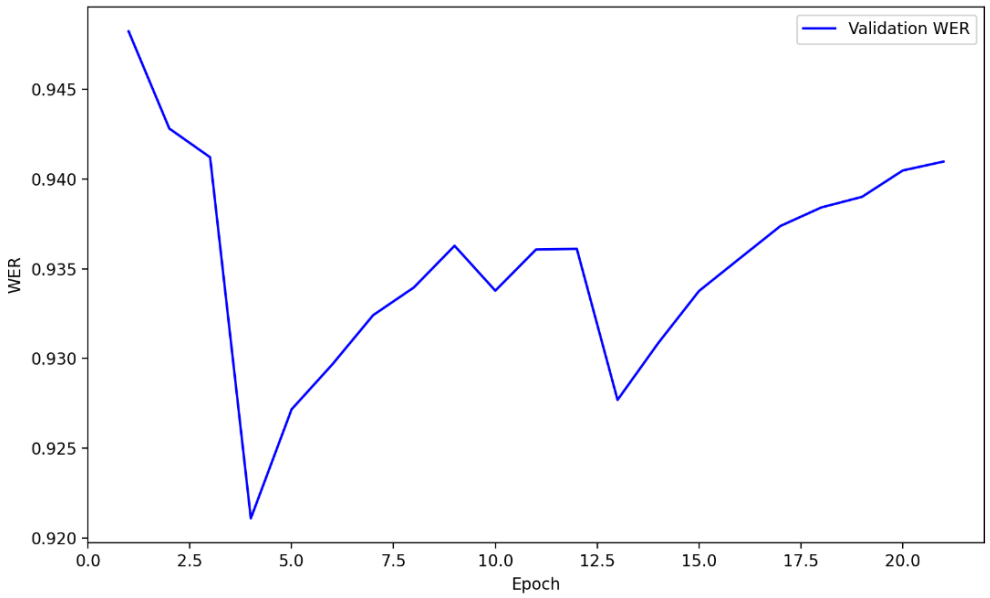


Figure 6-10: Whole Dataset Validation WER

The depicted plot showcases the validation Word Error Rate (WER) for the entire dataset, where initially, there's a decline in WER over the first few epochs, indicating the model's improvement in performance on the validation set. However, beyond a certain point, the WER begins to rise, suggesting potential overfitting or challenges in generalization. Selecting the optimal epoch involves finding a balance between minimizing WER and avoiding overfitting. Often, this entails choosing the epoch with the lowest validation WER, representing the model's best performance on unseen data while still benefiting from training. Alternatively, early stopping techniques can be employed by halting training when the validation WER consistently increases, signifying declining performance on unseen data. Through analysis of the validation

WER plot, an appropriate epoch can be identified to finalize the trained model, ensuring a balance between minimizing WER and preventing overfitting.

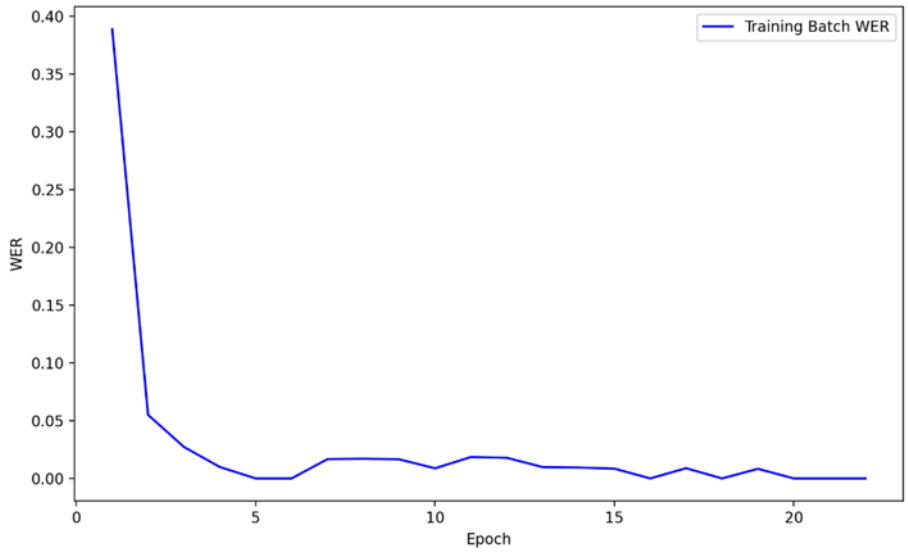


Figure 6-11: Whole Dataset Training batch WER

The training batch Word Error Rate (WER) for the entire dataset, showcasing a generally decreasing trend, which is desirable for model performance. However, notable fluctuations are observed throughout the plot. While the decreasing trend indicates the model's overall improvement, the fluctuations suggest variations in performance across different batches or iterations of training. Despite these fluctuations, the plot serves as a valuable metric for assessing model performance and determining the optimal point for early stopping. By monitoring the training batch WER, it’s possible to conclude how well the model is learning and decide when to halt training to prevent overfitting or ensure that the model's performance on unseen data remains favorable.

B. Test Dataset

For evaluation, test dataset consisting of 660 audio samples is utilized, recorded in various conditions.

I. Results

For a sample prediction and transcription, the WER and CER were calculated as if prediction is "can I have a cup of water” and ground truth is “can I have a cup of water" then Substitutions (S): 0 Deletions (D): 0, Insertions (I): 0, No errors (N): 7, Total words: 7

WER = (S + D + I) / N = (0 + 0 + 0) / 7 = 0

CER = (S + D + I) / total characters = (0 + 0 + 0) / 26 = 0

The speech recognition model achieved the following results on the test dataset:

Character Error Rate (CER): 6.4%,

Word Error Rate (WER): 7.08%

Table 6-2: WER and CER for Different Predictions

Age Group (Years)	Gender	Transcript	Prediction	WER (%)	CER (%)
16-18	Male	Hey Arm Fill up a cup with coffee	Hey Arm Fill up a cup with water	10.5	40
16-18	Female	Hey Arm	Hey Arm	0	0
18-27	Male	Hey arm can you get me a cup of hot coffee	Hey arm can you get me a cup of coffee	5	25
18-27	Female	Hey Arm I would like a cup of coffee please	Hey Arm I would like a cup of sprite please	10	14.29

From the calculated WER and CER values, it can be observed that the model performed well in some examples where both the WER and CER were 0.0%, indicating a perfect transcription with no errors. However, there were instances where the model made mistakes, resulting in higher WER and CER

Overall, the speech recognition model achieved an average WER of 11.5% and an average CER of 10.9% for all the examples. While these values demonstrate reasonably good accuracy, there is room for improvement, especially in cases where the output deviates significantly from the ground truth.

These metrics indicate the accuracy of the model in converting spoken language into written transcriptions. The achieved WER of 11.5% demonstrates that, on average, approximately 11.5% of the words in the recognized transcriptions differ from the ground truth.

6.3 Evaluation of Datasets

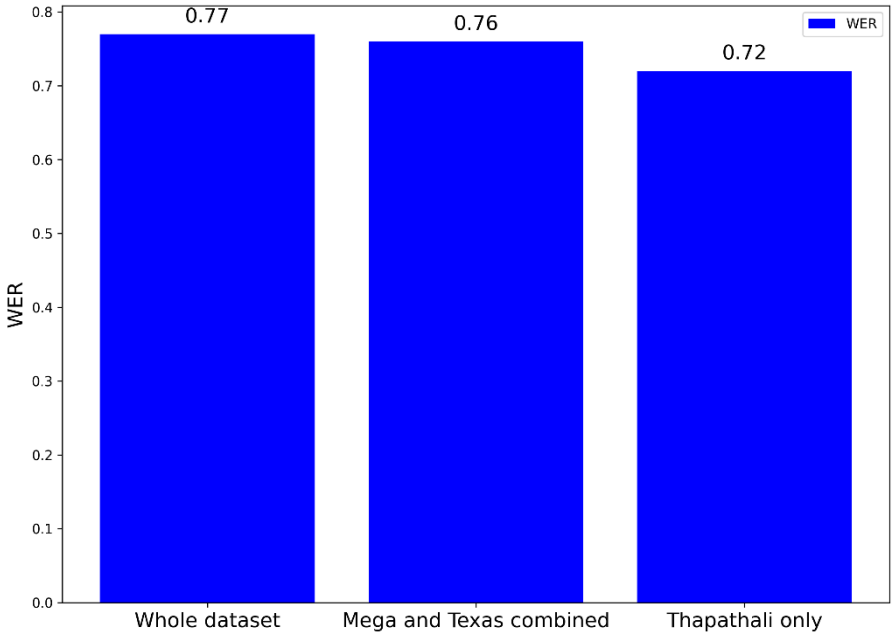


Figure 6-12: WER for Different Configurations

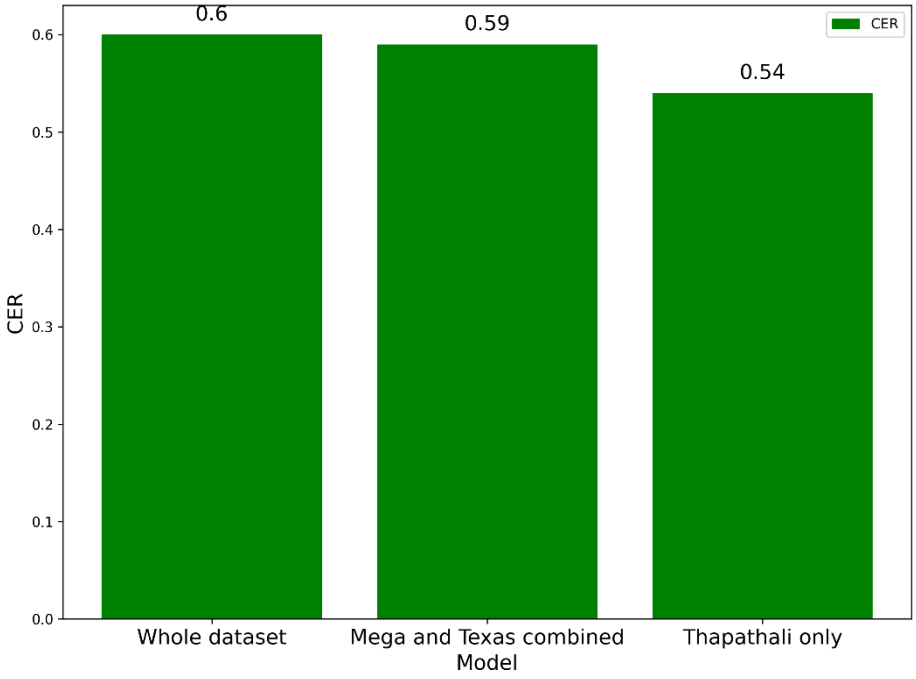


Figure 6-13: CER for Different Configurations

The above plots suggest that the model performs better on the Thapathali datasets compared to the Texas and Mega combined or whole dataset, indicating a possible

age bias of model or low-quality data from school children.

6.3.1 Confusion Histogram

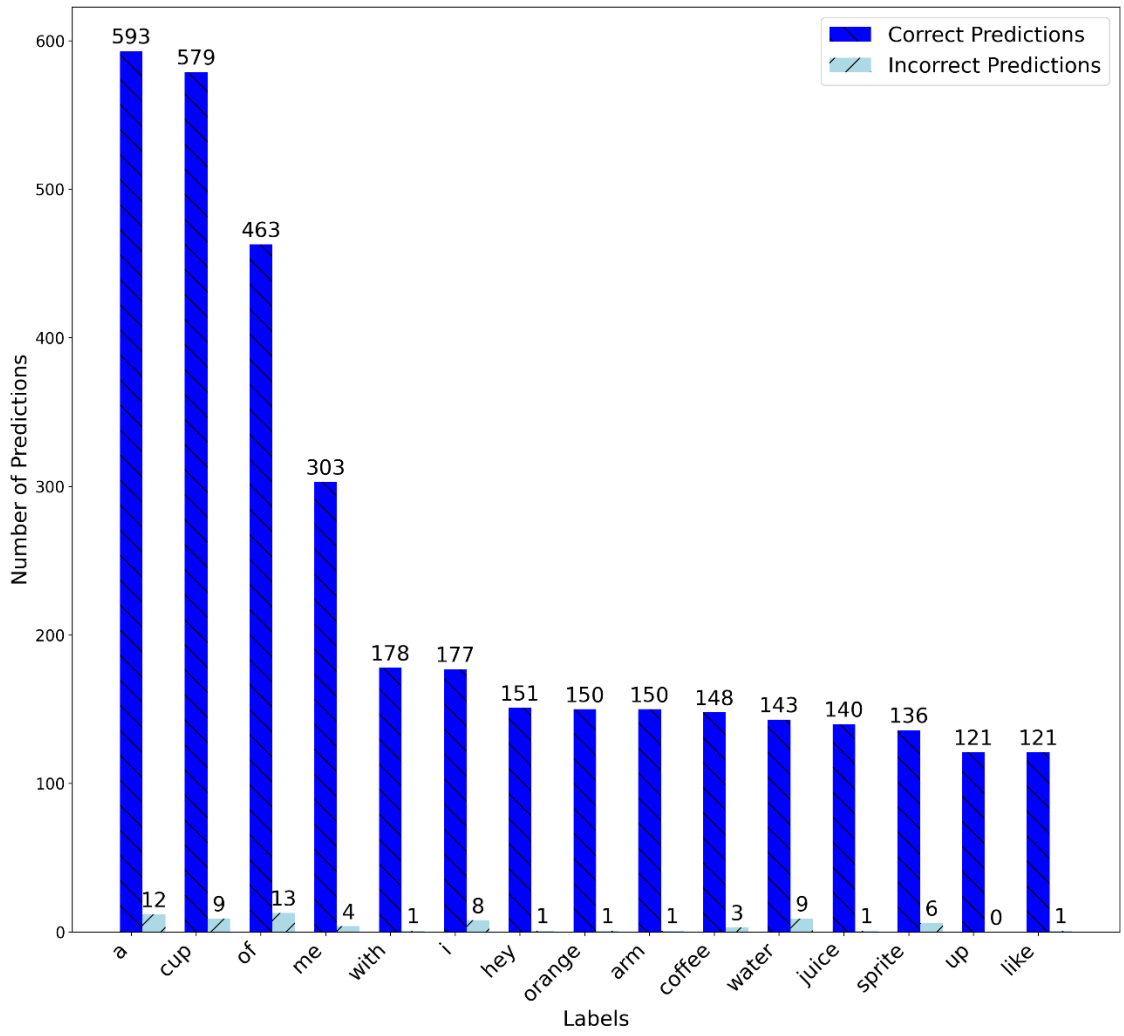


Figure 6-14: Confusion Histogram of First 15 Labels

The above plot illustrates the no of correct and wrong prediction for each label in the test set. It can be seen that the model has lest confusion on label ‘a’ and next it has least confusion on word ‘cup’.

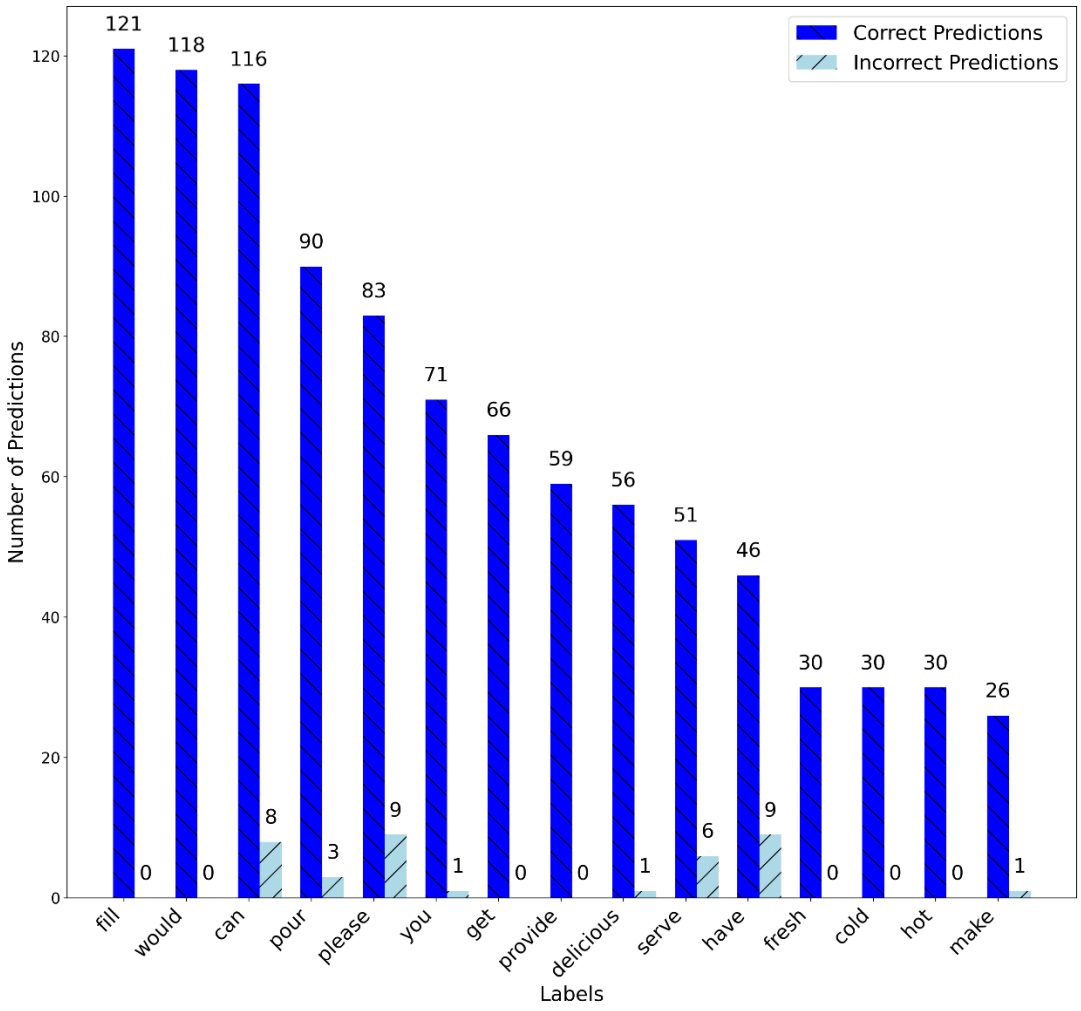


Figure 6-15: Confusion Histogram of Next 15 labels

The above plot illustrates the prediction of model for next 15 labels. It can be seen that there are comparatively more incorrect prediction in this group.

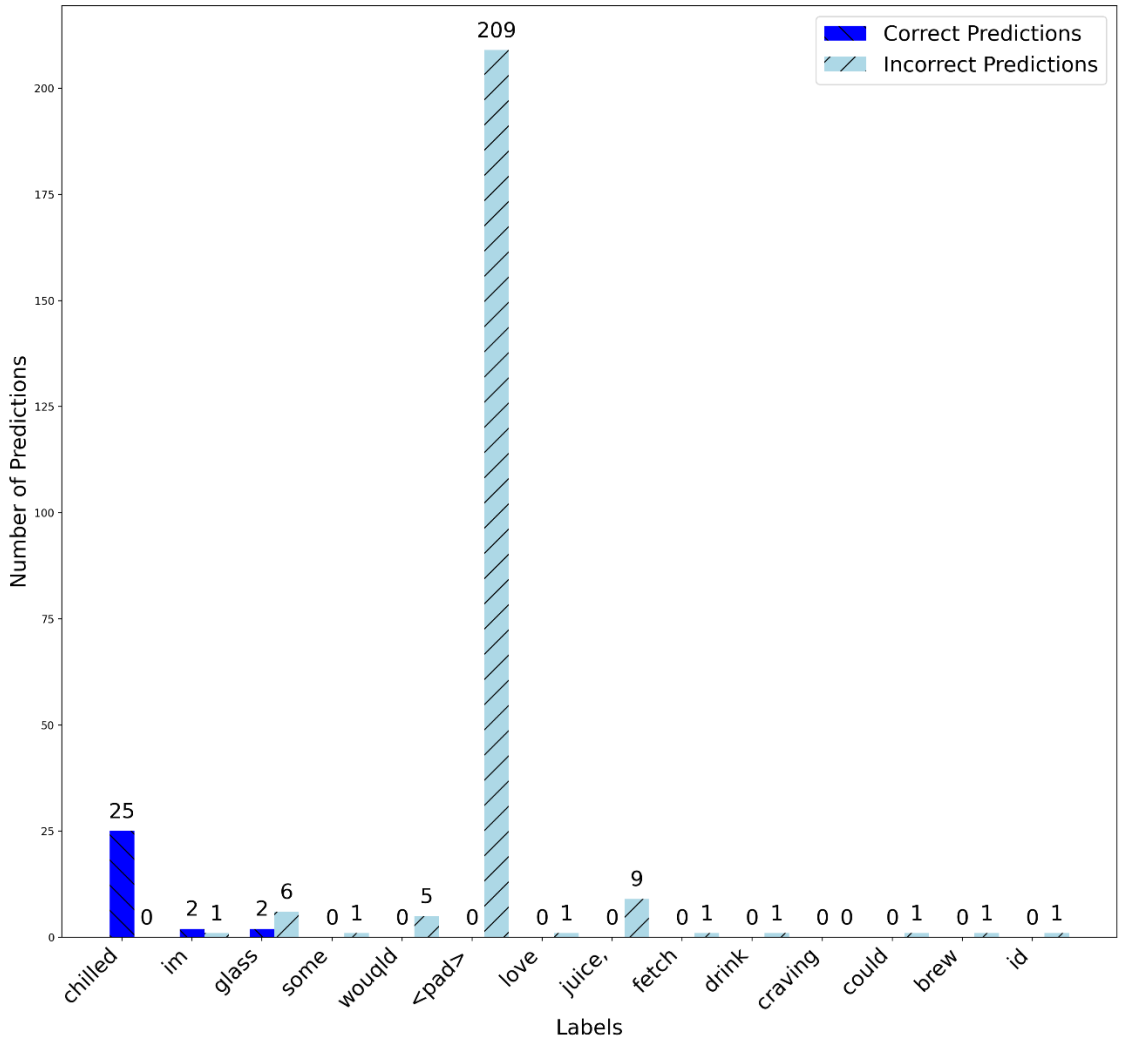


Figure 6-16: Confusion Histogram of Remaining Labels

The plot above has the remaining label and their corresponding correct and incorrect number of prediction. It can be seen from plot that the model has made most mistakes on juice compared to other labels in this group.

6.3.2 Error Analysis

An error analysis was conducted to understand the types of errors made by the speech recognition model. The analysis revealed that the most common errors were related to lack of proper pronunciation by the speaker and the quality of recorded sound as well as limited inclusion of old age population which may have made the model biased towards younger generations. The model mostly struggled with accurately transcribing words containing noise.

6.3.3 Discussion of Findings

From different explorations discussed above it can be concluded that freezing certain layers while fine-tuning a pretrained model, coupled with the utilization of a substantial dataset, led to a notable enhancement in model performance than achieved before without any freezing and limited dataset. By preserving the learned representations in the early layers and allowing the later layers to adapt to the specifics of our dataset, we effectively leveraged the pretrained knowledge to improve convergence and achieve superior results. Furthermore, the addition of data provided more examples for the model to learn from, enabling better generalization to unseen instances. These findings underscore the importance of thoughtful architecture modification and ample data availability in the fine-tuning process.

6.4 Object Detection Model Prediction

The section includes the evaluation metrics results of the object detection model trained on both virtual and physical datasets.

6.4.1 Model Predictions for Simulated Environment

A. Loss Curves

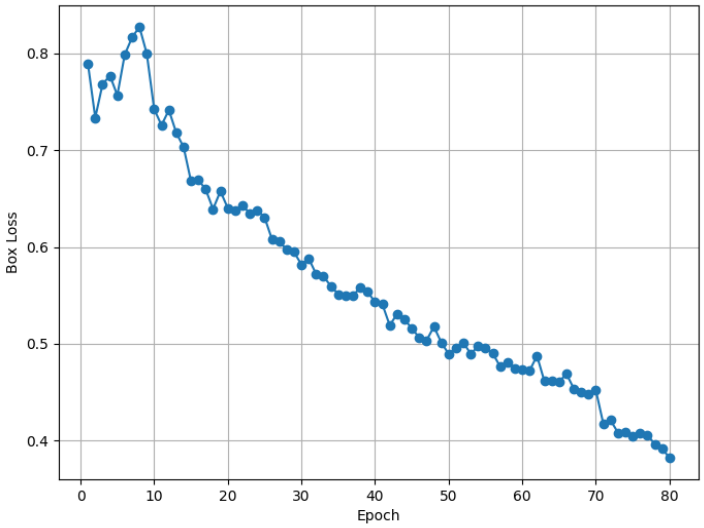


Figure 6-17: Bbox Loss Curve

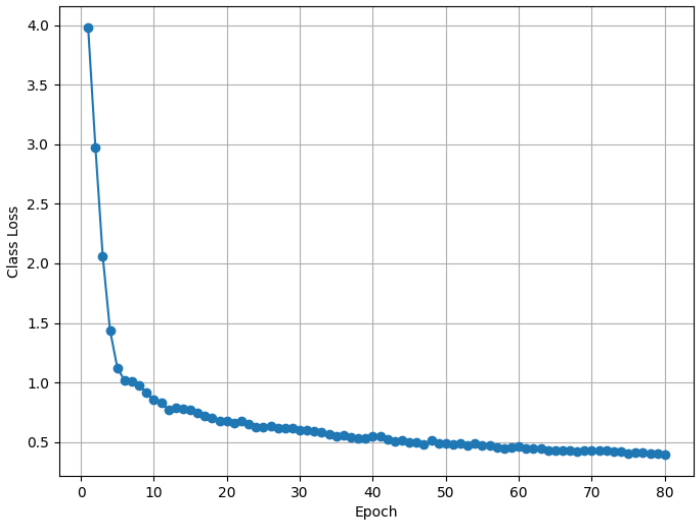


Figure 6-18: Class Loss Curve

The above plots depict the loss values in relation to the training epoch. The first figure displays the bounding box loss, while the second figure illustrates the class loss. Both losses exhibit a consistent decrease, indicating the model's learning process and improving performance on the training data.

B. Accuracy Metrics

I. Precision Metrics

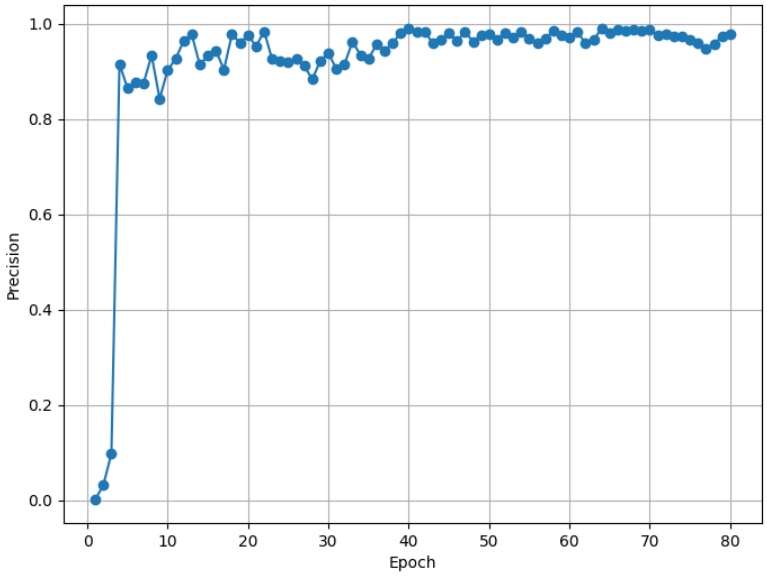


Figure 6-19: Precision Metrics

precision is a metric used to evaluate the performance of a model in correctly identifying positive samples (true positives) from the total samples that it classified as positive. It quantifies the proportion of true positive predictions out of all the samples that the model classified as positive.

II. Recall Metrics

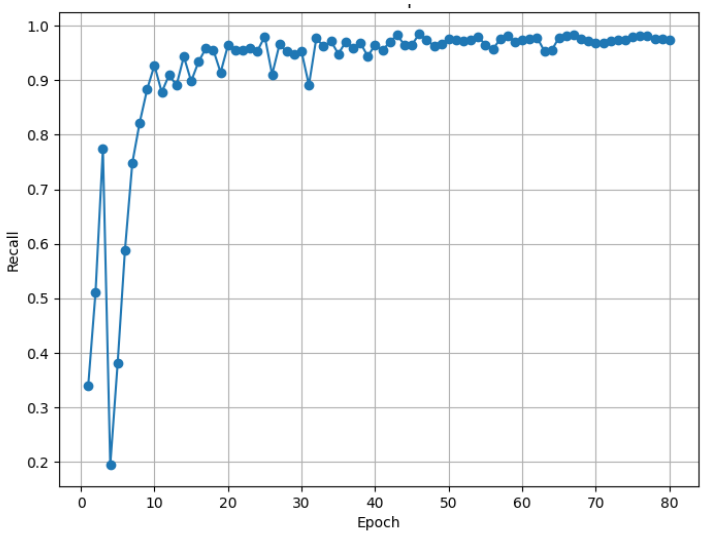


Figure 6-20: Recall Metrics

Recall (also known as sensitivity or true positive rate) is a metric used to evaluate the performance of a model in correctly identifying all the positive samples (true positives) out of the total positive samples present in the dataset. It quantifies the proportion of true positive predictions out of all the actual positive samples.

C. Mean Average Precision at 50 (mAP50)

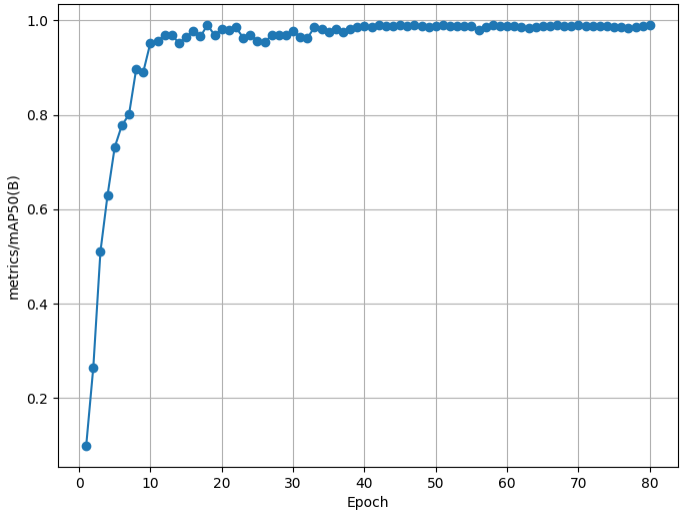


Figure 6-21: Mean Average Precision at 50

Mean Average Precision at 50 (mAP50) is a performance metric used to evaluate the accuracy of an object detection model. It measures how well the model can accurately detect objects in an image, specifically focusing on the precision of detections at a certain level of confidence.

The "50" in mAP50 refers to the threshold used for confidence score. It means that only detections with a confidence score greater than or equal to 50% are considered during the calculation of precision and recall.

D. Confusion Matrix

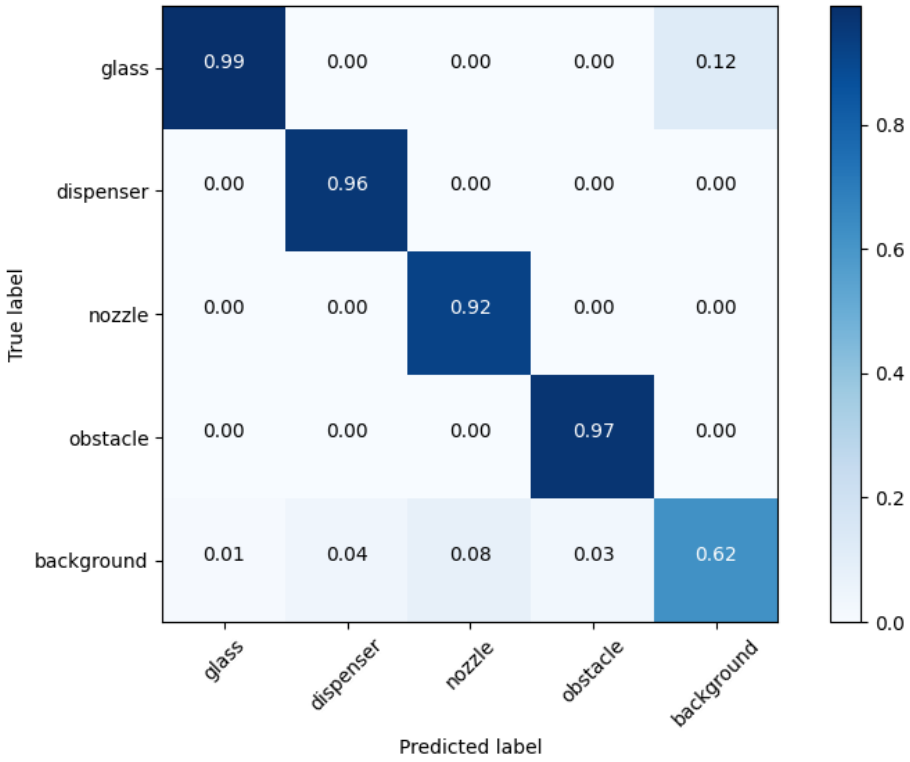
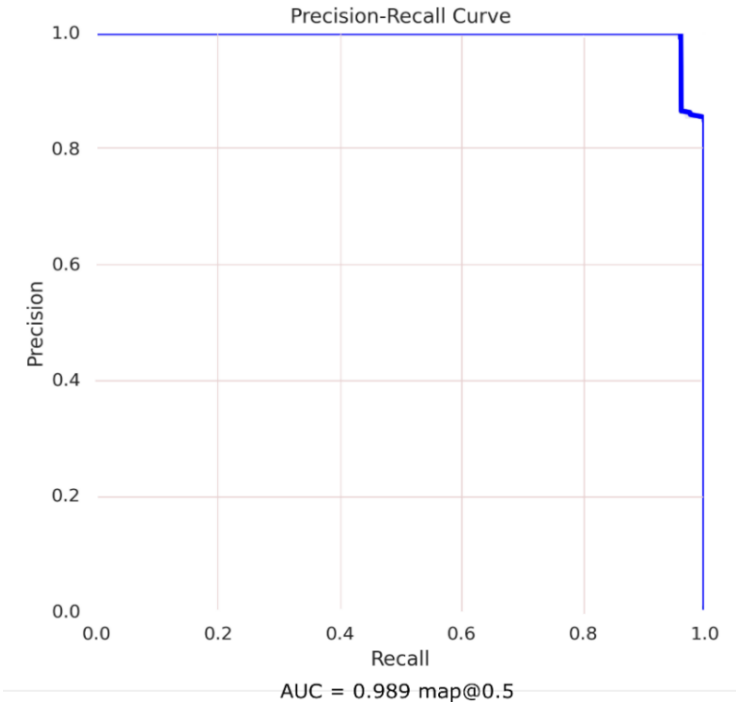


Figure 6-22: Confusion Matrix

Confusion Matrix provides a clear breakdown of how well the model has performed in classifying objects into their respective categories. Rows represent the true classes of objects, and columns represent the predicted classes by the model. The diagonal elements of the matrix represent the correctly classified objects, while off-diagonal elements indicate misclassifications.

E. Precision-Recall Curve



The Precision-Recall (PR) curve is a graphical representation used to evaluate the performance of a machine learning model, such as YOLO, in object detection tasks. It plots precision (positive predictive value) against recall (true positive rate) at various thresholds for object detection. A high-quality PR curve demonstrates the model's ability to maintain high precision while achieving high recall.

The area under the PR curve, often denoted as mAP (mean Average Precision), quantifies the model's overall performance. An mAP50 of 0.989 indicates that the YOLO model achieves an impressive average precision across different object classes

6.4.2 Model Predictions for Physical Environment

A. Loss Curves

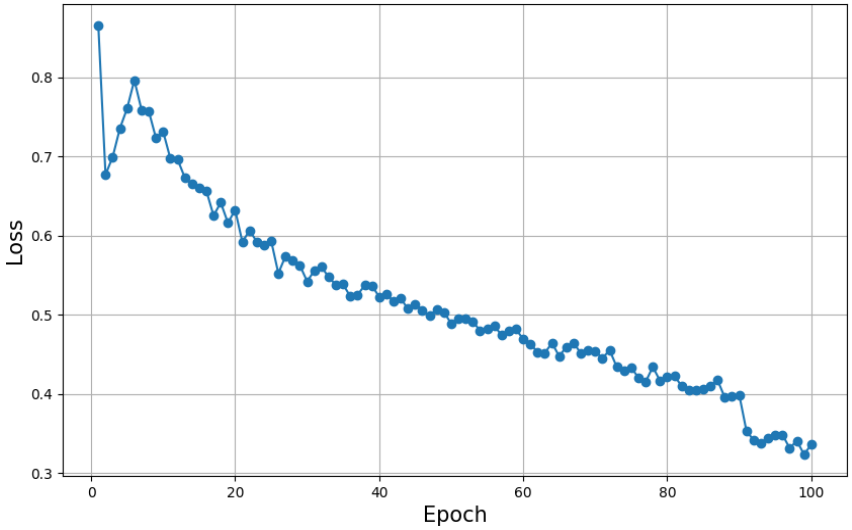


Figure 6-23: Bbox Loss Curve

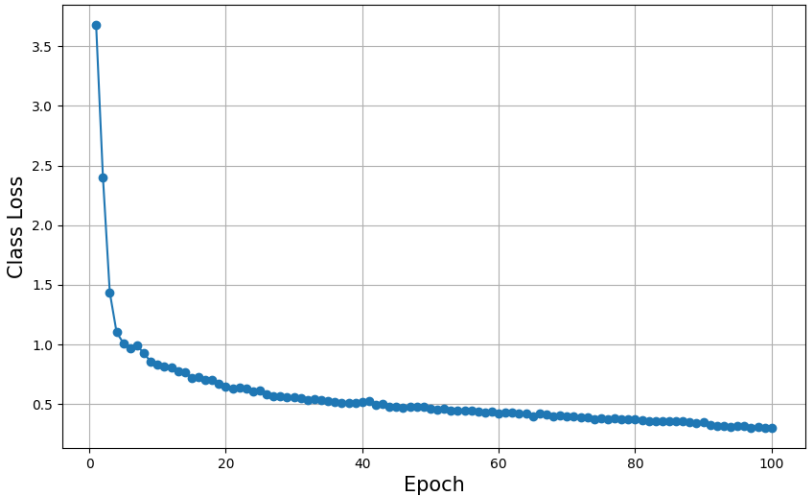


Figure 6-24: Class Loss Curve

The above plots depict the loss values in relation to the training epoch. The first figure displays the bounding box loss, while the second figure illustrates the class loss. Both losses exhibit a consistent decrease, indicating the model's learning process and improving performance on the training data.

B. Accuracy Metrics

I. Precision Metrics

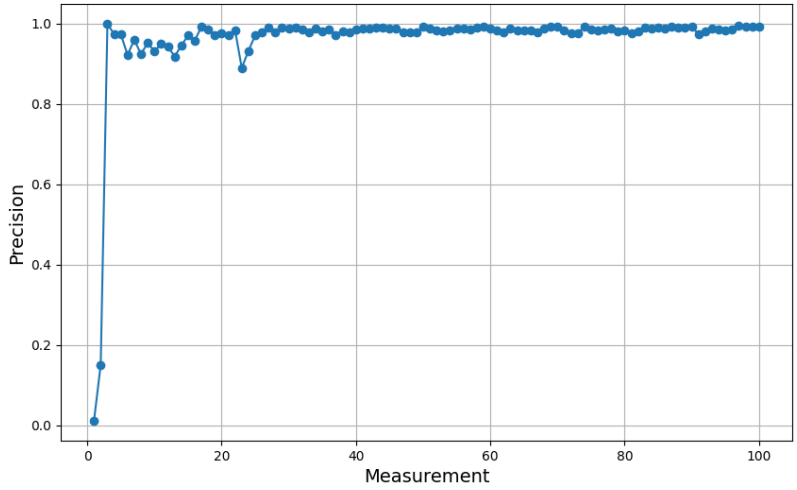


Figure 6-25: Precision Metrics

precision is a metric used to evaluate the performance of a model in correctly identifying positive samples (true positives) from the total samples that it classified as positive. It quantifies the proportion of true positive predictions out of all the samples that the model classified as positive.

II. Recall Metrics

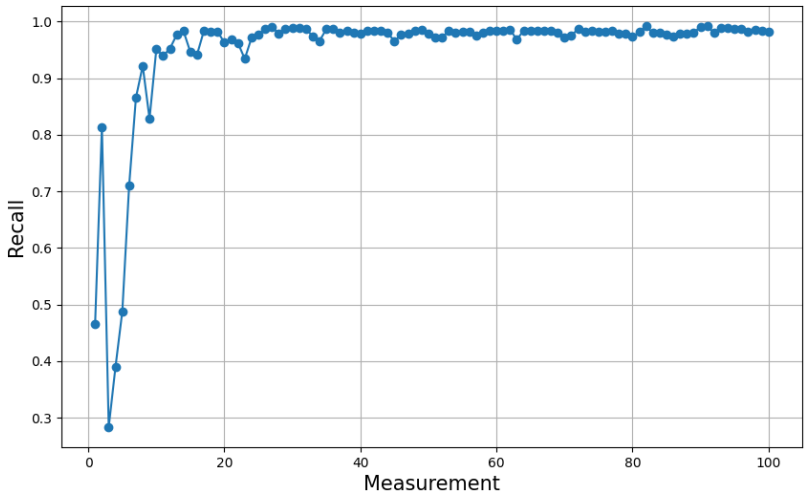


Figure 6-26: Recall Metrics

Recall (also known as sensitivity or true positive rate) is a metric used to evaluate the performance of a model in correctly identifying all the positive samples (true positives) out of the total positive samples present in the dataset. It quantifies the proportion of true positive predictions out of all the actual positive samples. The recall is calculated using the formula:

C. Mean Average Precision at 50 (mAP50)

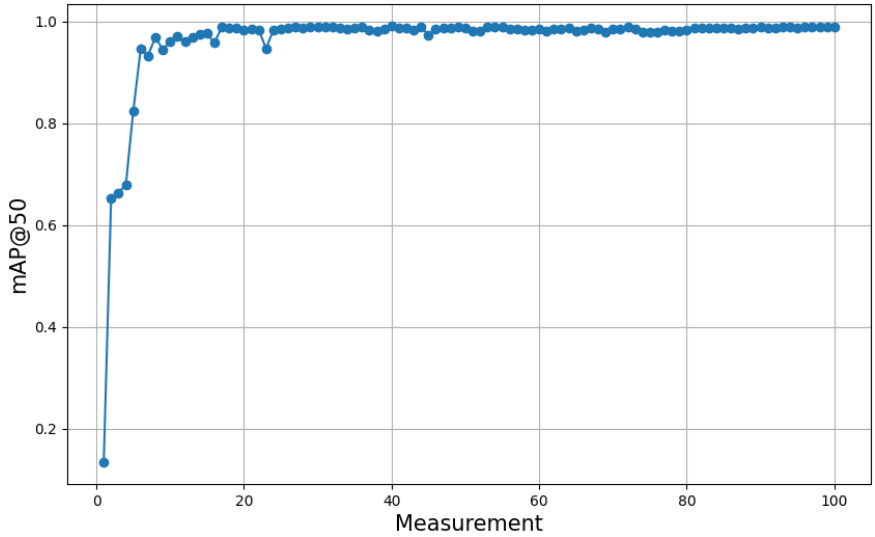


Figure 6-27: Mean Average Precision at 50

Mean Average Precision at 50 (mAP50) is a performance metric used to evaluate the accuracy of an object detection model. It measures how well the model can accurately detect objects in an image, specifically focusing on the precision of detections at a certain level of confidence.

The "50" in mAP50 refers to the threshold used for confidence score. It means that only detections with a confidence score greater than or equal to 50% are considered during the calculation of precision and recall.

D. Confusion Matrix

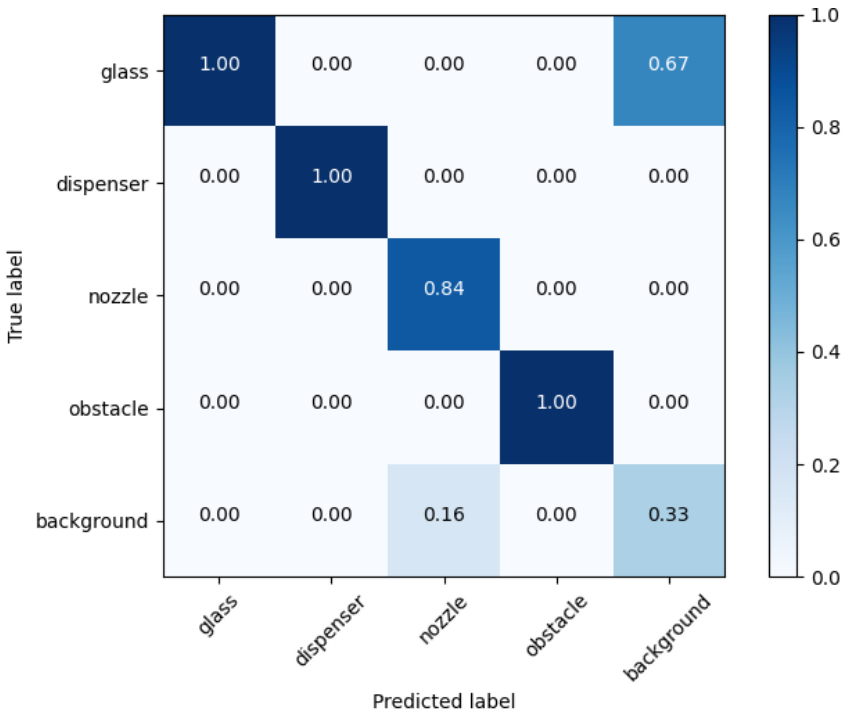


Figure 6-28: Confusion Matrix

Confusion Matrix provides a clear breakdown of how well the model has performed in classifying objects into their respective categories. Rows represent the true classes of objects, and columns represent the predicted classes by the model. The diagonal elements of the matrix represent the correctly classified objects, while off-diagonal elements indicate misclassifications.

E. Precision-Recall Curve

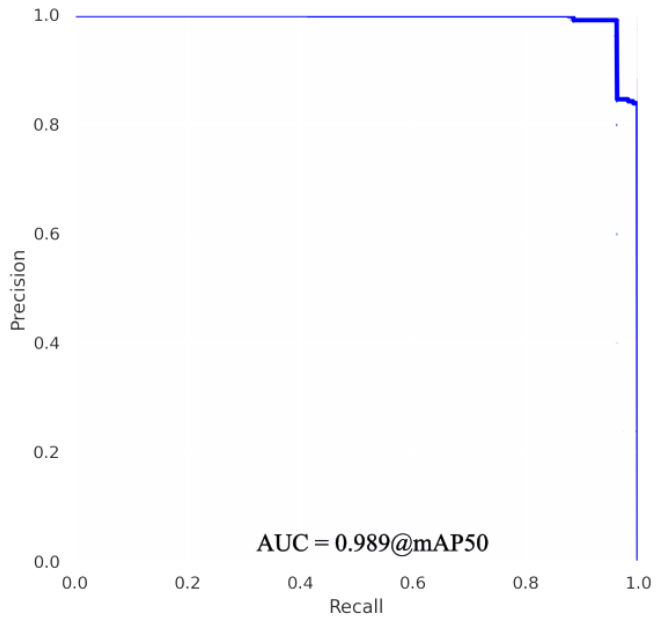


Figure 6-29: Precision-Recall Curve

The Precision-Recall (PR) curve is a graphical representation used to evaluate the performance of a machine learning model, such as YOLO, in object detection tasks. It plots precision (positive predictive value) against recall (true positive rate) at various thresholds for object detection. A high-quality PR curve demonstrates the model's ability to maintain high precision while achieving high recall.

The area under the PR curve, often denoted as mAP (mean Average Precision), quantifies the model's overall performance. A mAP50 of 0.989 indicates that the YOLO model achieves an impressive average precision across different object classes.

6.4.3 Detections of YOLOv8 Model

The results presented below, along with their corresponding confidence scores, represent the successful detection of the dispenser, nozzle, drinking glass and obstacles using the custom YOLOv8 model in both simulation and physical environment. The average score obtained for both the dispenser and nozzle detections exceeds 0.8, indicating a high level of confidence in the accuracy of the model's predictions.

A. Detection for Simulation Environment

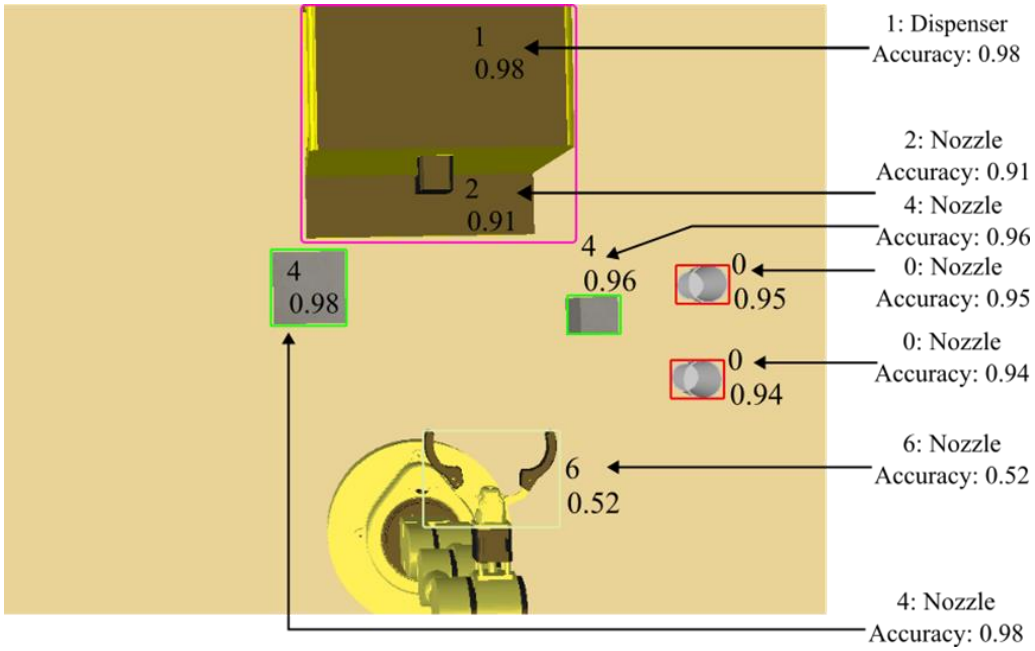


Figure 6-30: Detection on Simulation Environment

B. Detection for Physical Environment

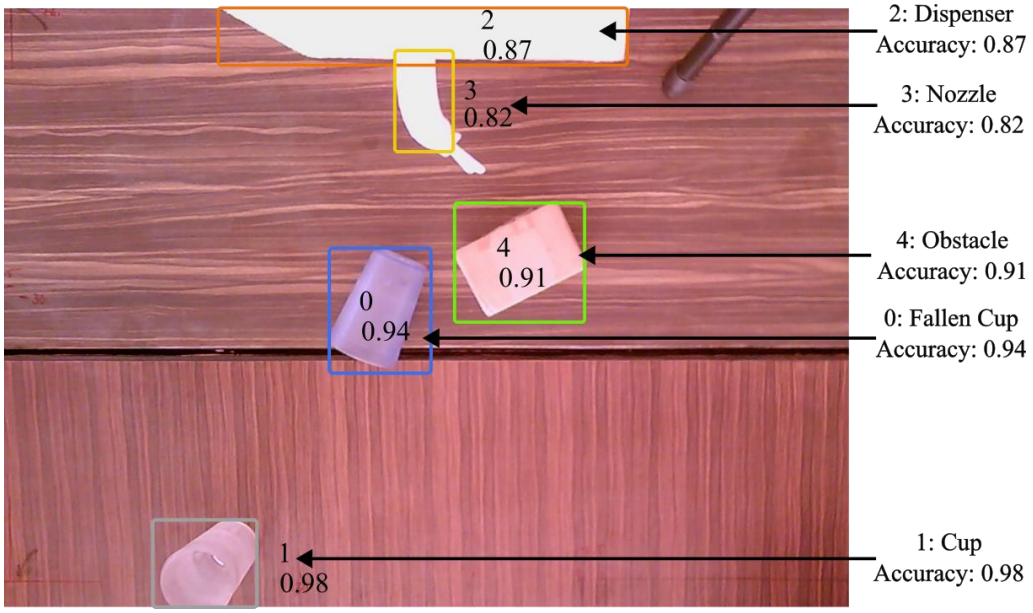


Figure 6-31: Detection on Physical Environment

6.5 Dispenser System

6.5.1 Dispenser Model in Simulation

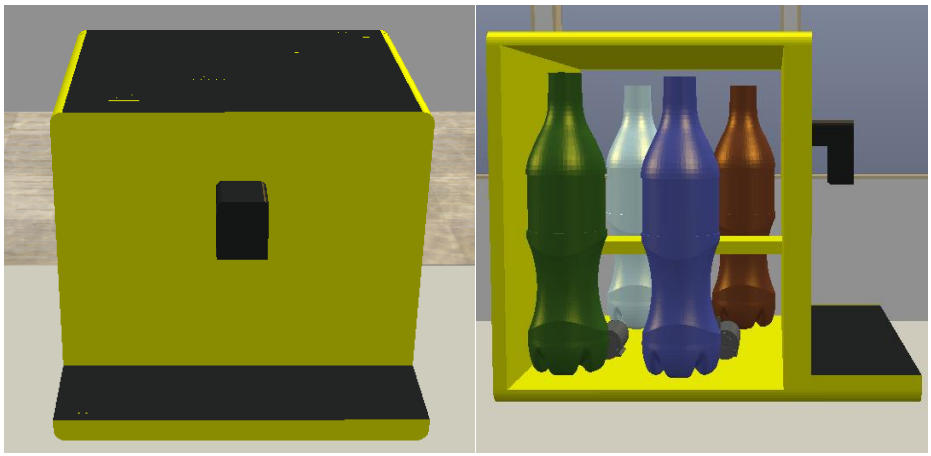


Figure 6-32 Dispenser Model for Simulation Environment

In the simulation system, the dispenser model features four container bottles. However, dispensing drinks into glasses is simulated by adjusting the color of the glass to match the ordered drink. This approach is employed due to limitations within the simulation environment, which may not fully replicate the physical process of dispensing liquids.

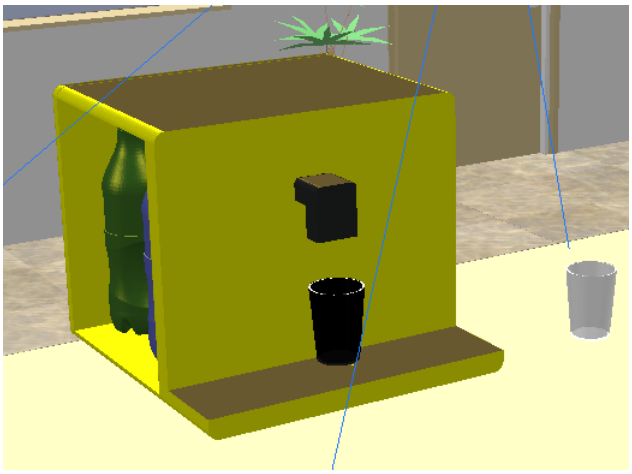


Figure 6-33: Dispensing Drink in Cup (Simulation)

6.5.2 Dispenser Model in Physical System

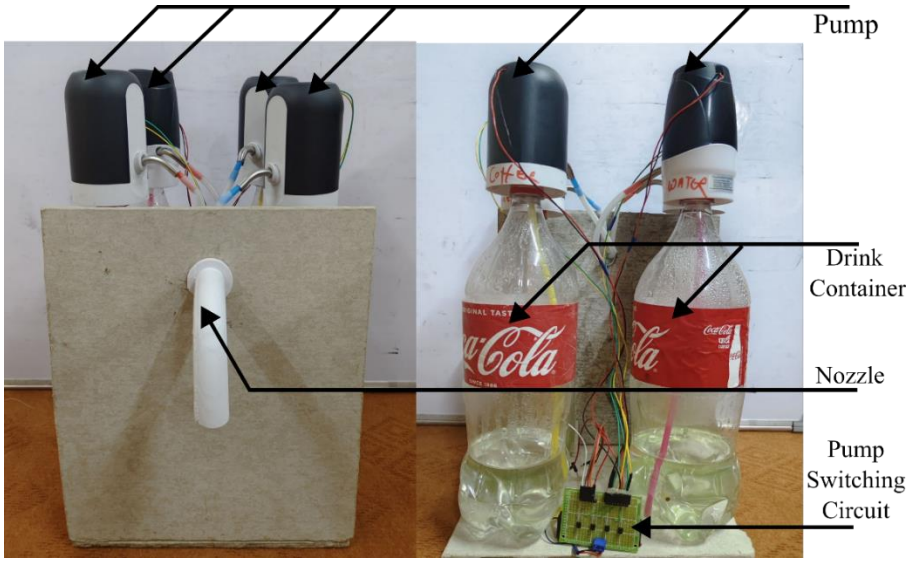


Figure 6-34: Dispenser Model for Physical Environment

The physical model of the dispenser system comprises four distinct containers, each equipped with individual pumps. Utilizing small diameter pipes, the system facilitates the smooth transfer of liquids from the container bottles to the nozzle for dispensing. Additionally, a switching circuit is integrated into the system to ensure that the pump corresponding to the selected drink operates effectively when that particular drink is ordered.

Figure 6-35: Dispensing Drink in Cup (Physical)

6.6 3D Printed Parts

The following parts of the cycloidal drive is printed using PLA material. The dimensions of the parts are presented in the appendix

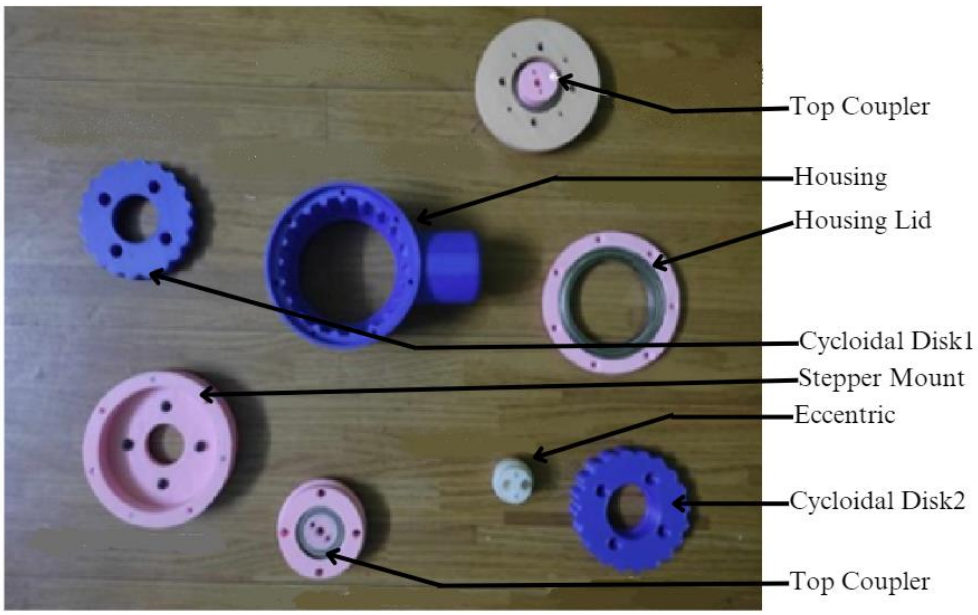


Figure 6-36: 3D Printed Parts (Top View)

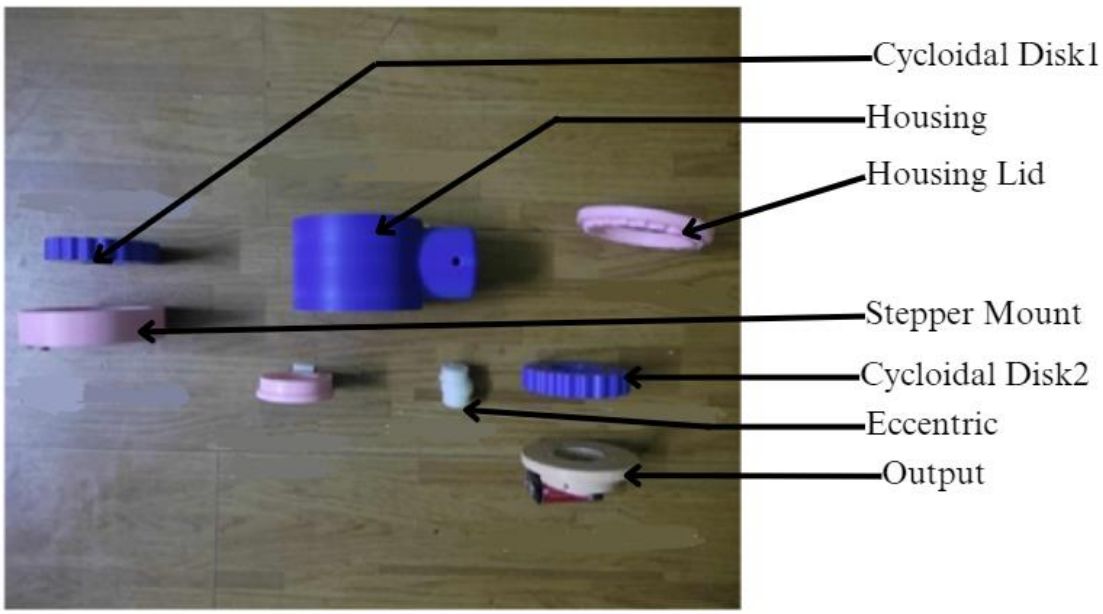


Figure 6-37: 3D Printed Parts (Side View)

6.7 Different Actions Performed Robotic Arm

The actions performed by the robotic arm from being in the default position to serving the drinks to client are demonstrated for both virtual as well as real environment.

6.7.1 Actions Performed in Simulation Environment

A. Robotic Arm at Default Position

At the beginning, Robotic Arm is rested at its default position with all joints angles at zero degrees.

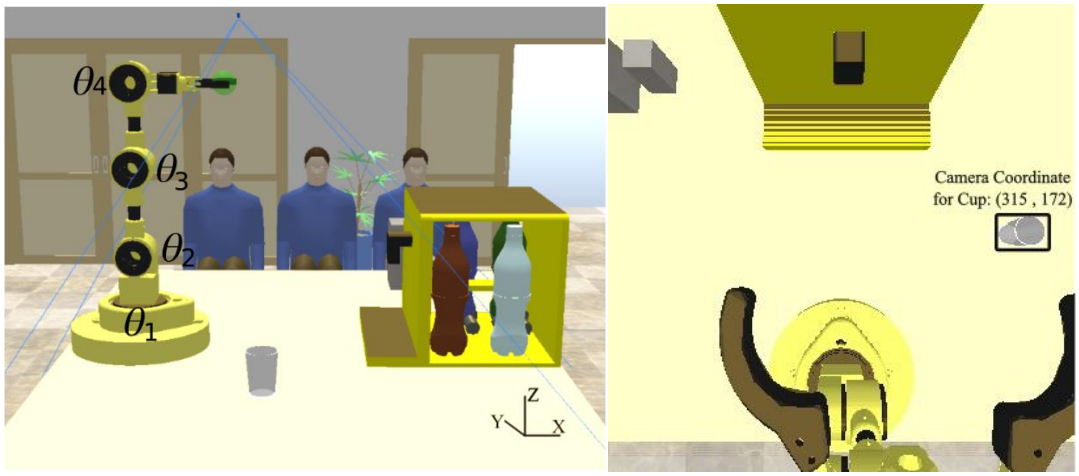


Figure 6-38: Default Position with Camera View

Table 6-3: Coordinated Mapped for Empty Glass (Simulation)

Camera Coordinate	**Simulation Coordinates**
(315,172)	(0.587,-0.764)

Table 6-4: Inverse Kinematics Parameters - 1

End Effector Position in Task Space (m)			**Joint Angles(degree)**			
X	Y	Z	θ_1	θ_2	θ_3	θ_4
0.33	0	1.012	0	0	0	0

B. Grabbing the Empty Glass

After receiving valid command of drink from the client, the glass location in the table is derived using overhead camera input, and inverse kinematics is performed to obtain the required joint angles.

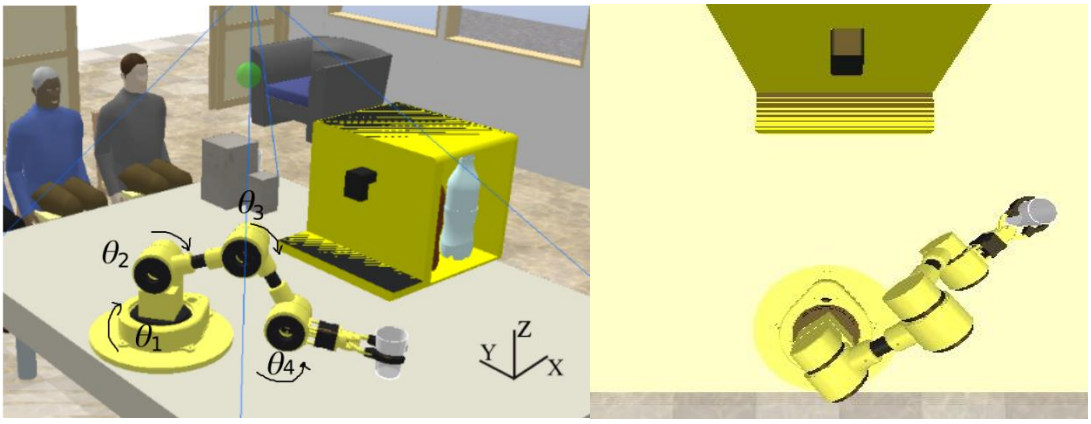


Figure 6-39: Grasping Empty Glass with Camera View

Table 6-5: Inverse Kinematics Parameters - 2

Desired End Effector Position in Task Space (m)			**Desired Joint Angles(degree)**			
X	Y	Z	θ_1	θ_2	θ_3	θ_4
0.587	-0.764	0.05	-52.5	-72	-66.0	131.4

Let's analyze the change in joint angles and their rotation (Clockwise-CW or Counter Clockwise- CCW) for each action performed by the robotic arm.

$$\Delta\theta = current\ joint\ angle - previous\ joint\ angle$$

Table 6-6: Change in Angles - 2

Change in Angles ($\Delta\theta$)			
$\Delta\theta_1$ (CW)	$\Delta\theta_2$ (CW)	$\Delta\theta_3$ (CW)	$\Delta\theta_4$ (CCW)
= -52.5 – 0 = -52.5	= -72 - 0 = -72	= -66.0 - 0 = -66.0	= 131.4 – 0 = 131.4

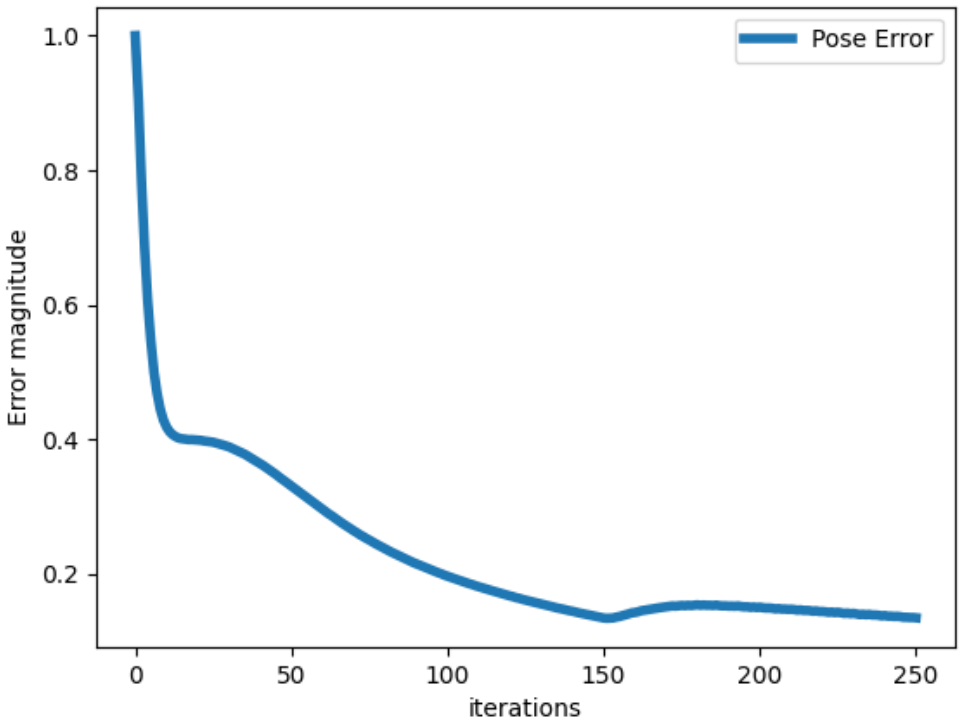


Figure 6-40: Pose Error while Grabbing the Cup

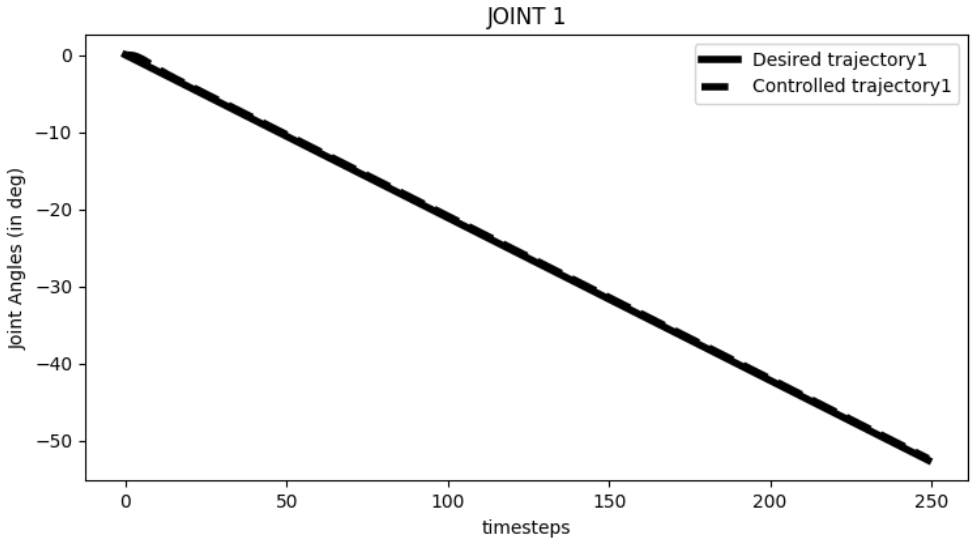


Figure 6-41: Trajectory Tracking for Joint 1 while Grabbing Cup

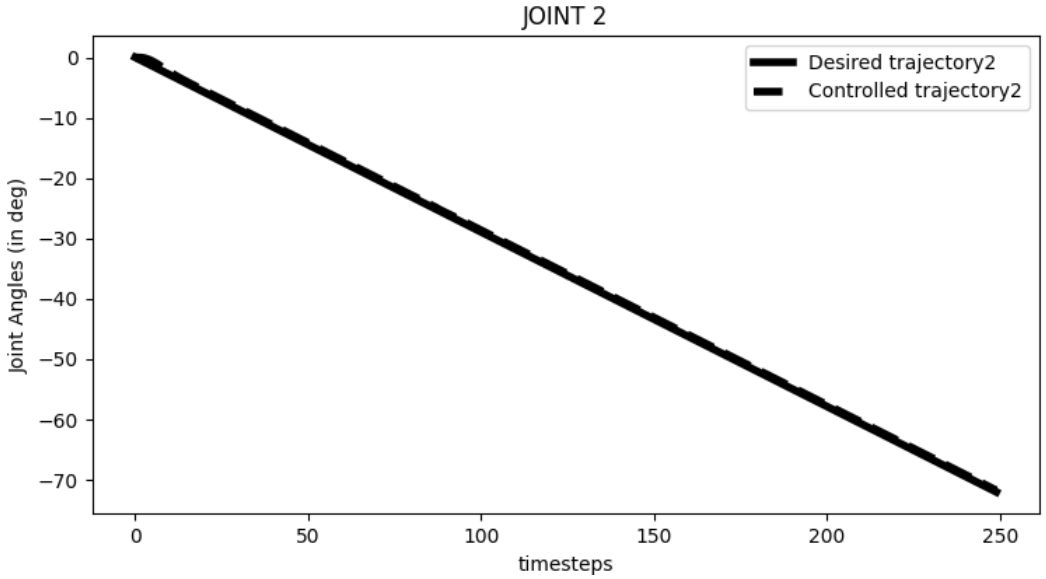


Figure 6-42: Trajectory Tracking for Joint 2 while Grabbing Cup

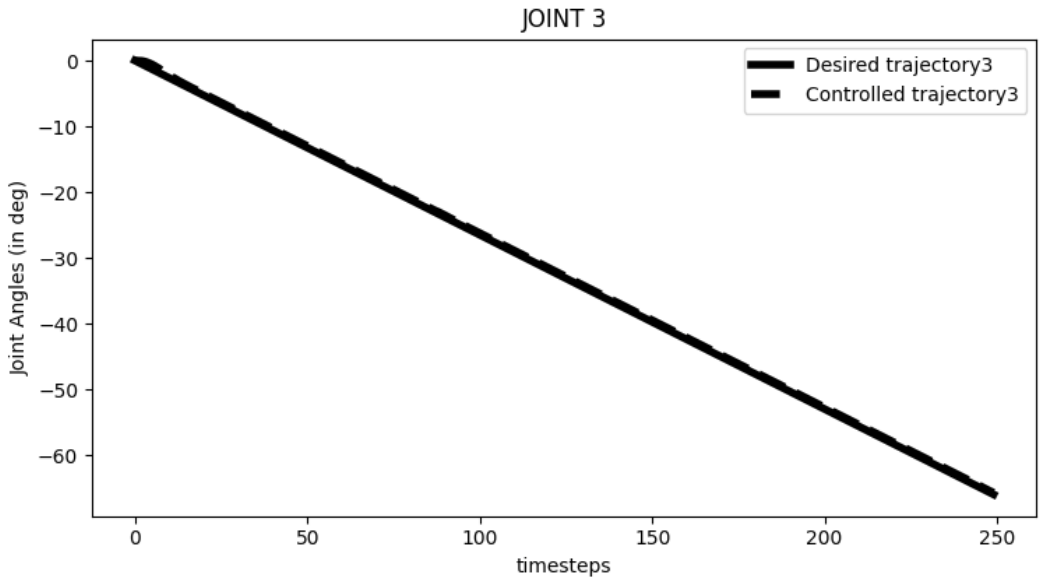


Figure 6-43: Trajectory Tracking for Joint 3 while Grabbing Cup

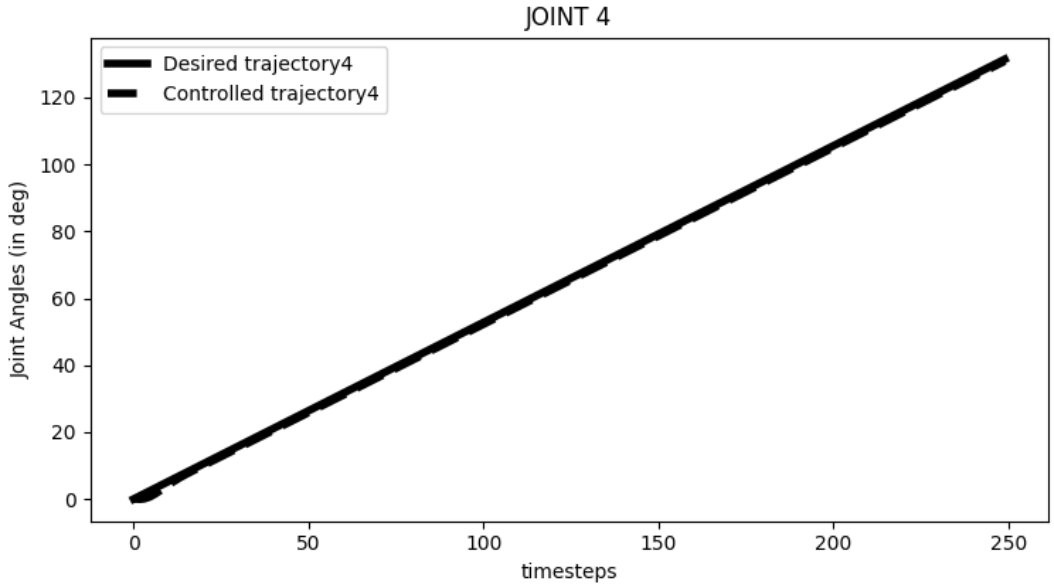


Figure 6-44: Trajectory Tracking for Joint 4 while Grabbing Cup

C. Towards the Dispenser

After grabbing the cup, the arm reached towards dispenser to fill the cup.

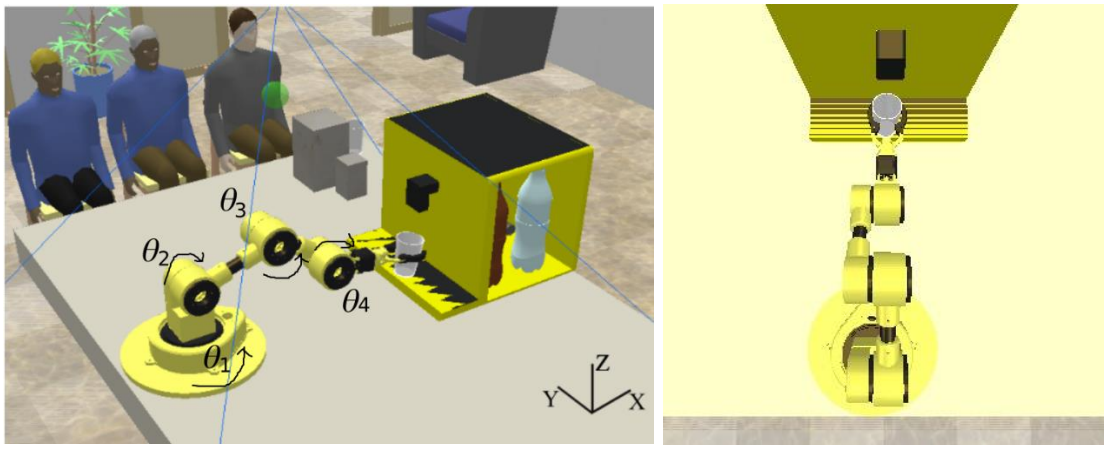


Figure 6-45: Towards the Dispenser with Camera View

Table 6-7: Inverse Kinematics Parameters - 3

Desired End Effector Position in Task Space (m)			**Desired Joint Angles(degree)**			
X	Y	Z	θ_1	θ_2	θ_3	θ_4
1.109	-0.008	0.1	0.4	-72	-50.1	115.8

Table 6-8: Change in Angles - 3

Change in Angles ($\Delta\theta$)			
$\Delta\theta_1$ (CCW)	$\Delta\theta_2$ (CW)	$\Delta\theta_3$ (CCW)	$\Delta\theta_4$ (CW)
= 0.4 - (-52.5) = 52.9	= -72 – (-72) = 0	= -50.1 - (-66.0) = 15.9	= 115.8 - 131.4 = -15.6

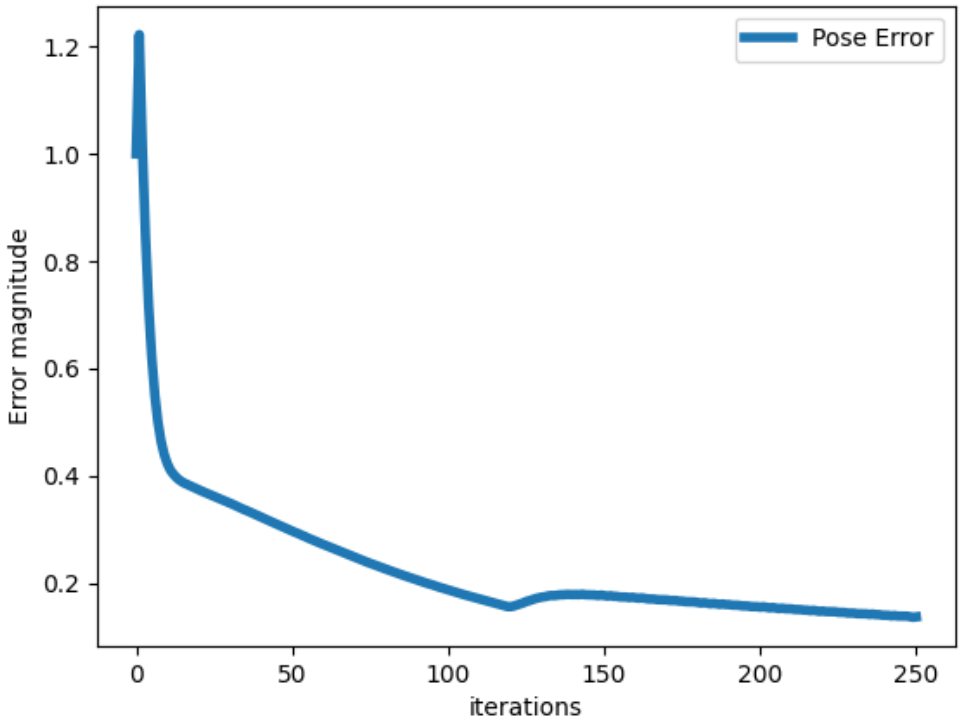


Figure 6-46: Pose Error while Filling the Cup

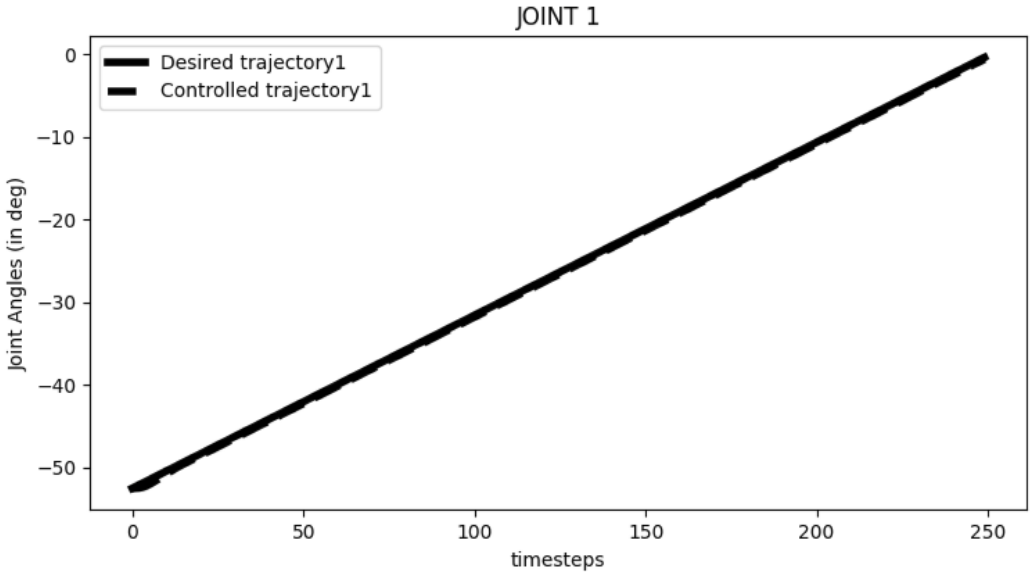


Figure 6-47: Trajectory Tracking for Joint 1 while Filling Cup

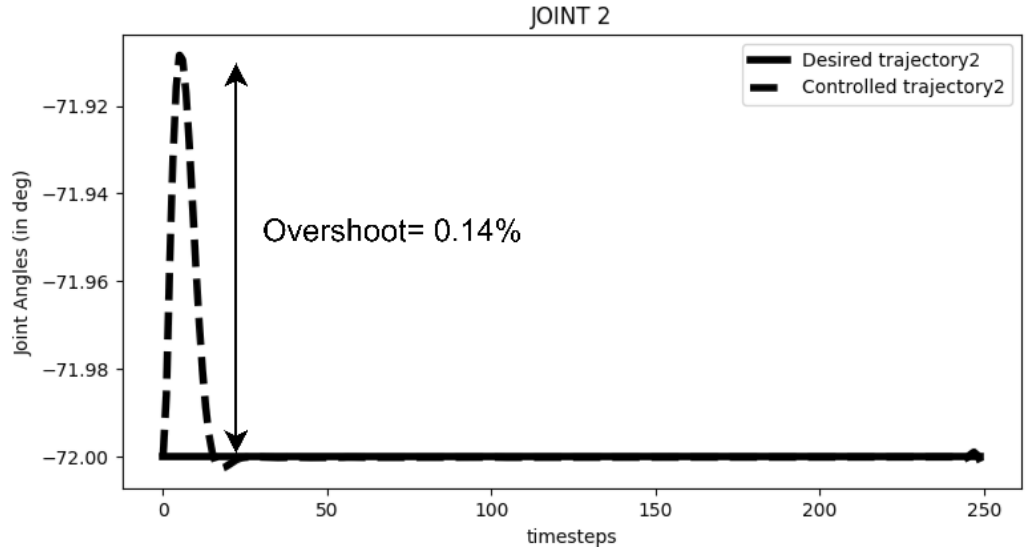


Figure 6-48: Trajectory Tracking for Joint 2 while Filling Cup

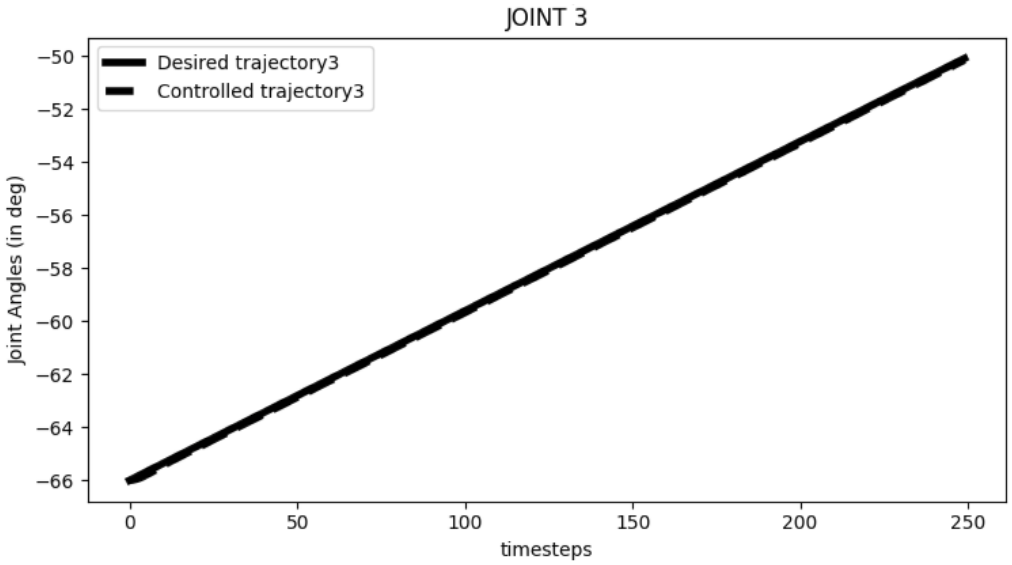


Figure 6-49: Trajectory Tracking for Joint 3 while Filling Cup

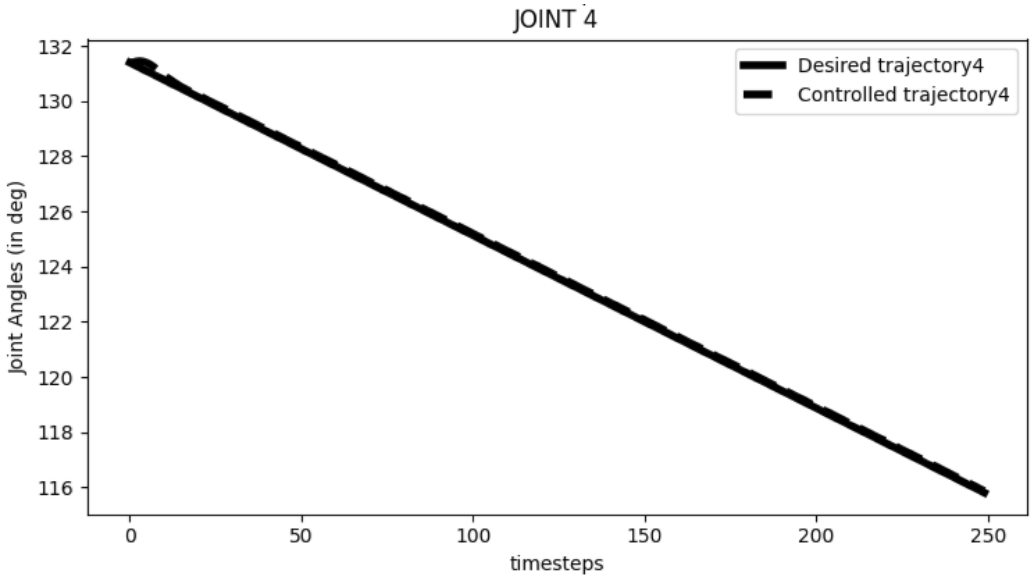


Figure 6-50: Trajectory Tracking for Joint4 while Filling Cup

D. Serving Drink to Client

The robotic arm serves the drink to the client.

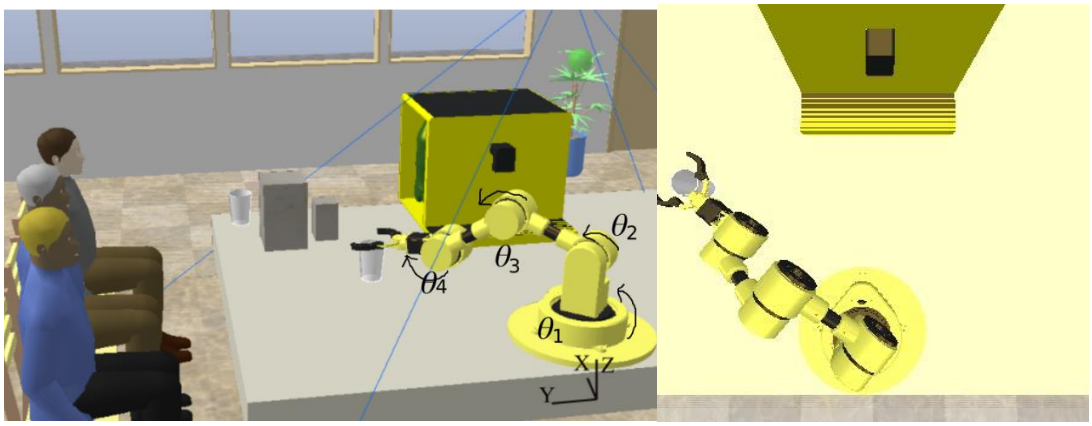


Figure 6-51: Serving Drink to Client with Camera View

Table 6-9: Inverse Kinematics Parameters - 4

Desired End Effector Position in Task Space (m)			**Desired Joint Angles(degree)**			
X	Y	Z	θ_1	θ_2	θ_3	θ_4
0.74	0.82	0.05	47.9	-72	-54.4	119.0

Table 6-10: Change in Angles - 4

Change in Angles ($\Delta\theta$)			
$\Delta\theta_1$ (CCW)	$\Delta\theta_2$ (CW)	$\Delta\theta_3$ (CW)	$\Delta\theta_4$ (CCW)
= 47.9 - 0.4 = 47.5	= -72 – (-72) = 0	= -54.4 - (-50.1) = -4.3	= 119 - 115.8 = 3.2

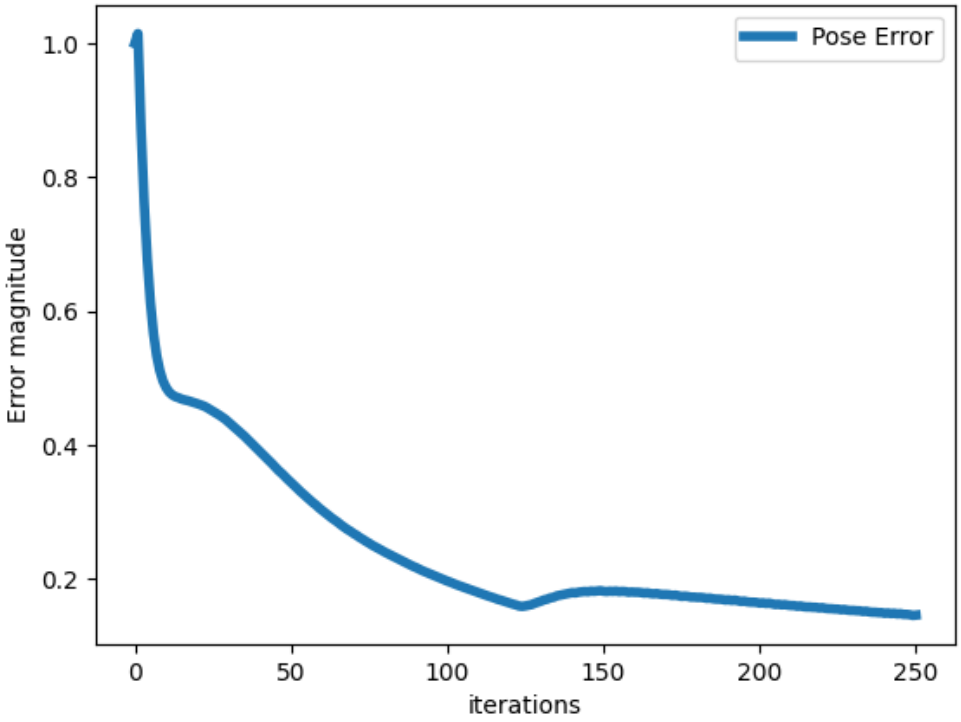


Figure 6-52: Pose Error while Serving the Drink

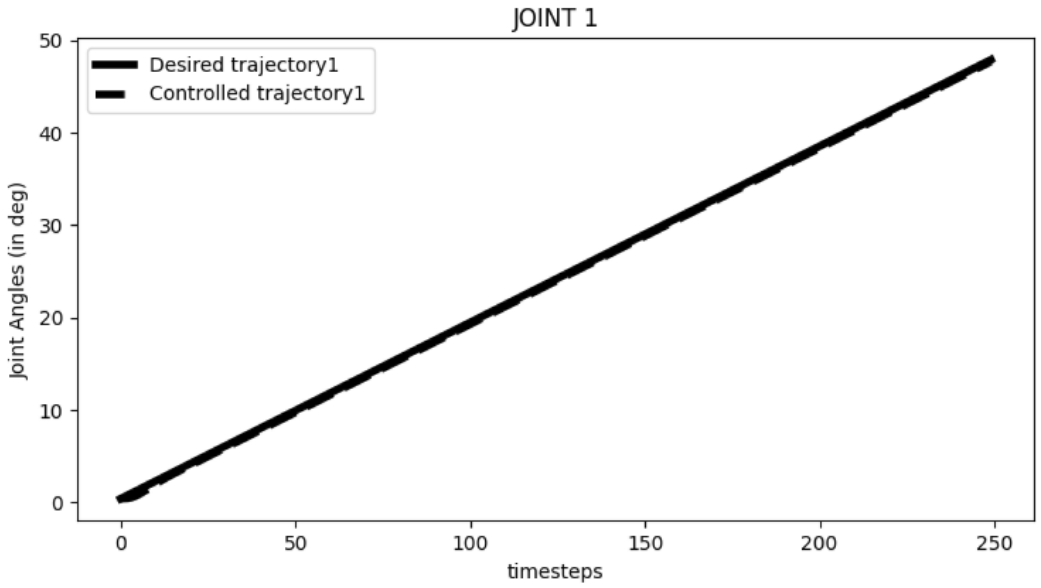


Figure 6-53: Trajectory Tracking for Joint 1 while Serving Drink to Client

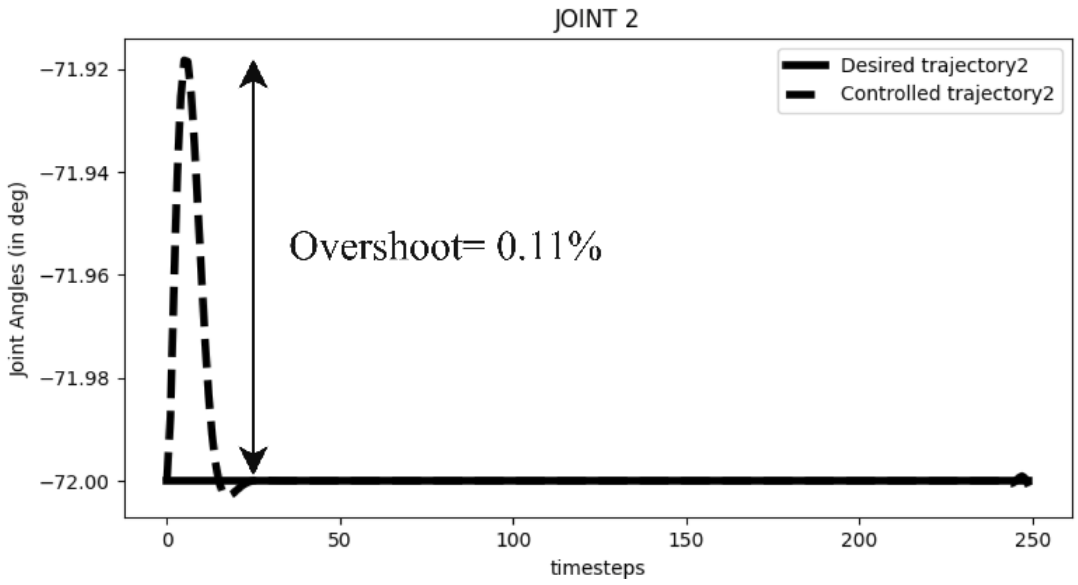


Figure 6-54: Trajectory Tracking for Joint 2 while Serving Drink to Client

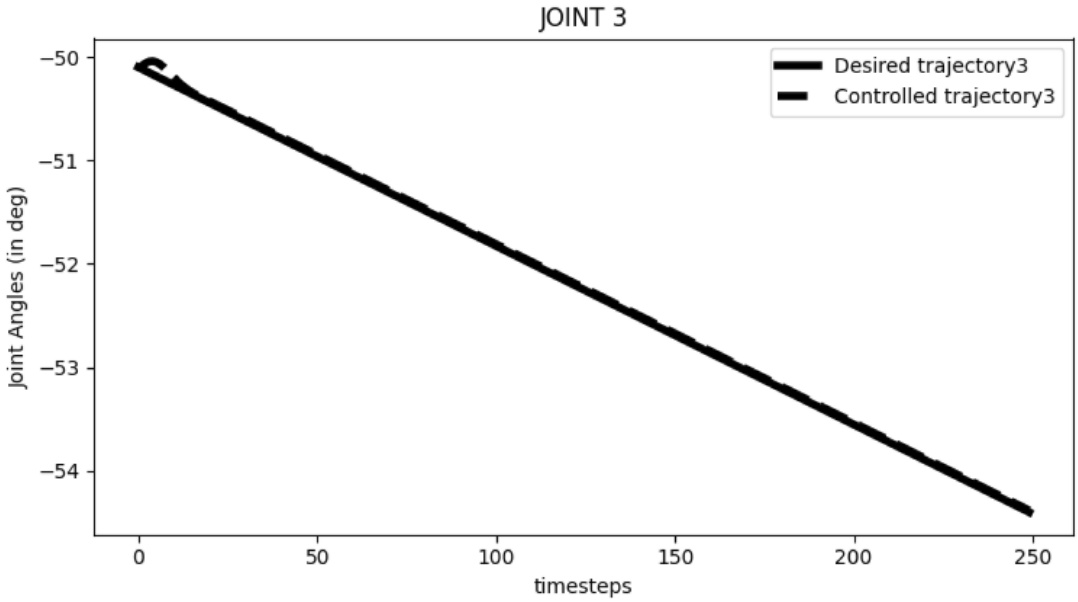


Figure 6-55: Trajectory Tracking for Joint 3 while Serving Drink to Client

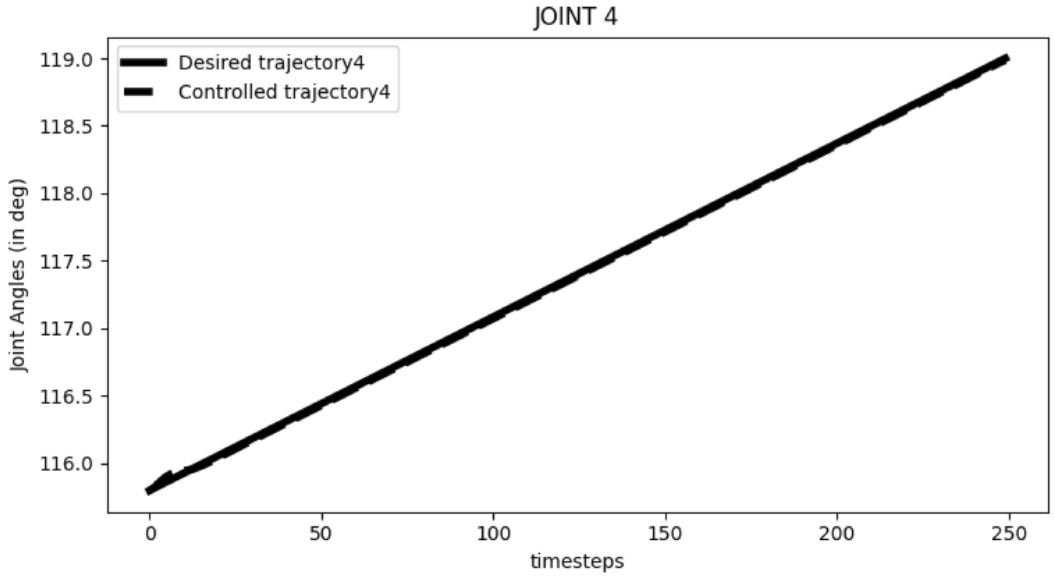


Figure 6-56: Trajectory Tracking for Joint 4 while Serving Drink to Client

E. Returning to Default Position

After serving the ordered drink to the client successfully, the robotic arm returned to the default position.

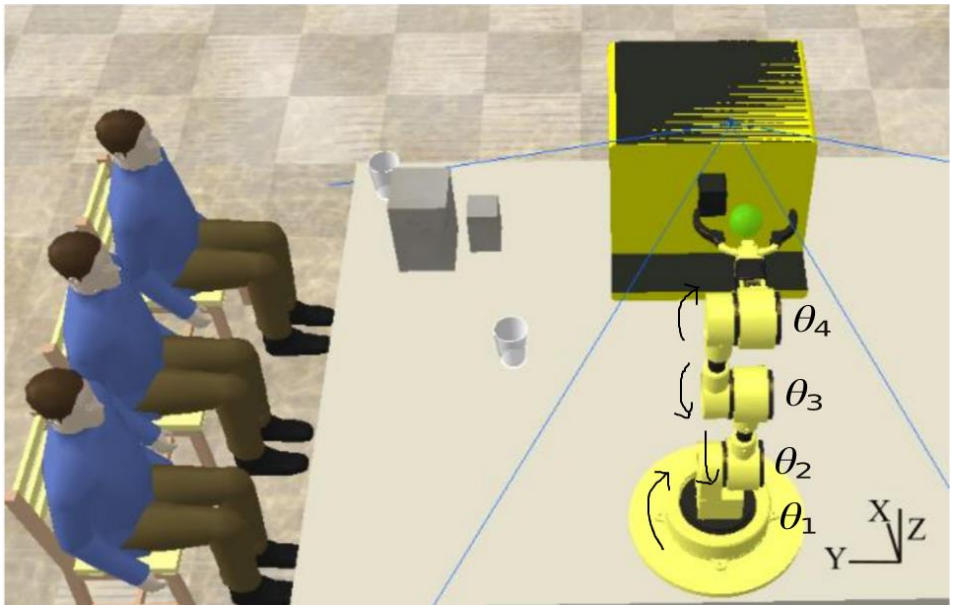


Figure 6-57: Returning to Default Position with Camera View

Table 6-11: Inverse Kinematics Parameters - 5

End Effector Position in Task Space (m)			**Desired Joint Angles(degree)**			
X	Y	Z	θ_1	θ_2	θ_3	θ_4
0.33	0	1.012	0	0	0	0

Table 6-12: Change in Angles - 5

Change in Angles ($\Delta\theta$)			
$\Delta\theta_1$ (CW)	$\Delta\theta_2$ (CCW)	$\Delta\theta_3$ (CCW)	$\Delta\theta_4$ (CW)
= 0 - 47.9 = -47.9	= 0 - (-72) = 72	= 0 - (-54.4) = 54.4	= 0 - 119 = -119

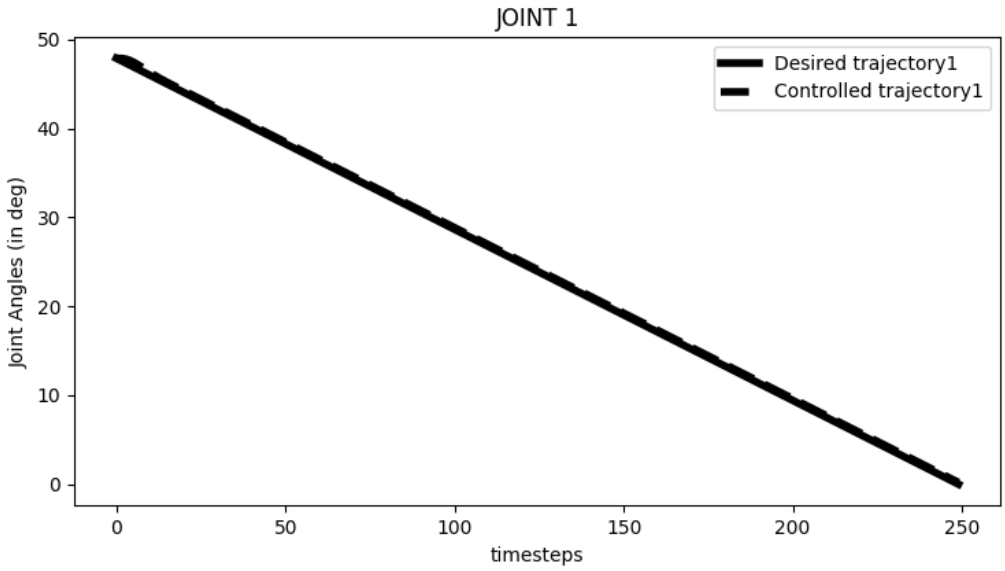


Figure 6-58: Trajectory Tracking for Joint 1 while Moving to Default Position

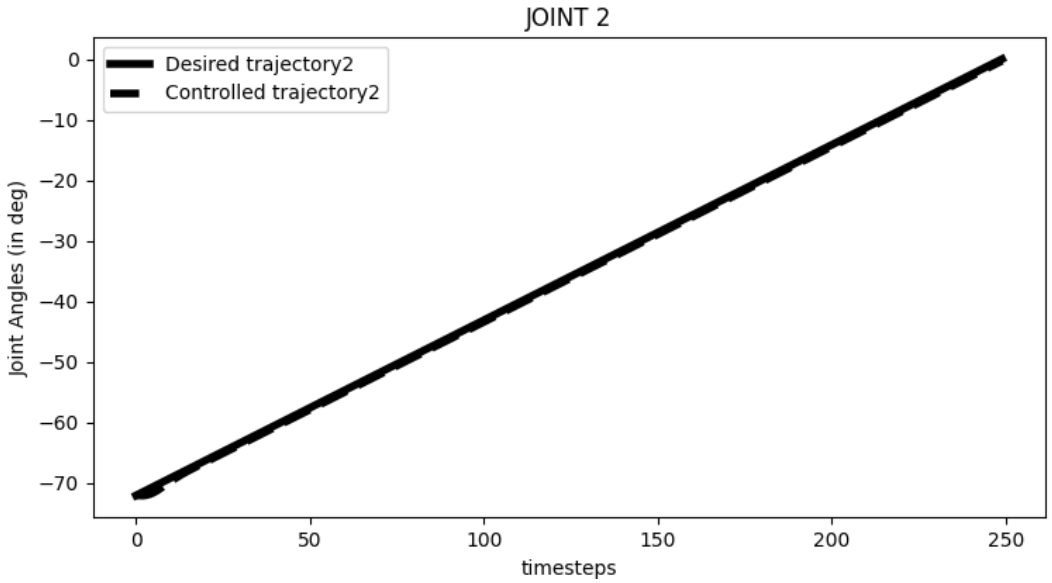


Figure 6-59: Trajectory Tracking for Joint 2 while Moving to Default Position

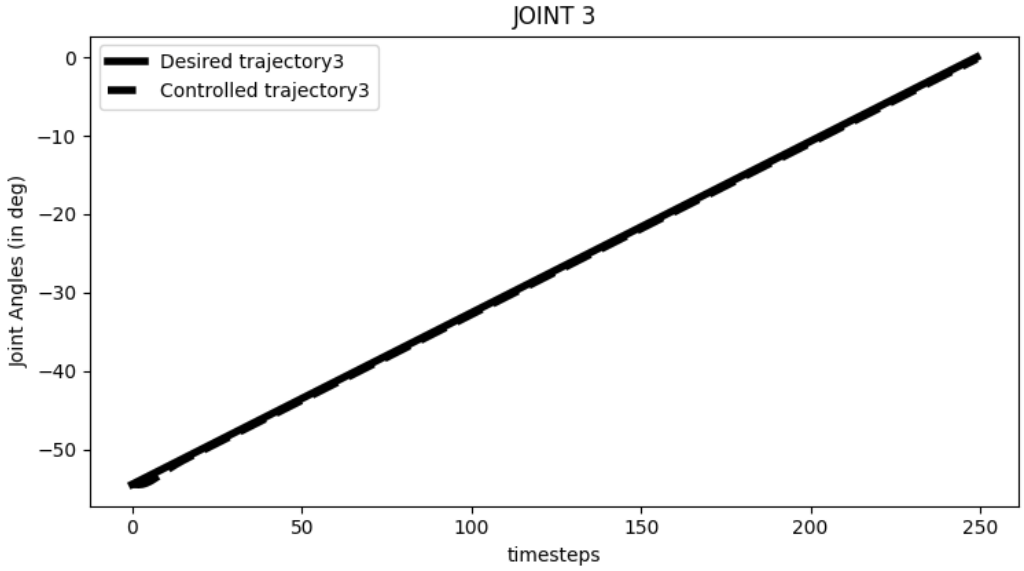


Figure 6-60: Trajectory Tracking for Joint 3 while Moving to Default Position

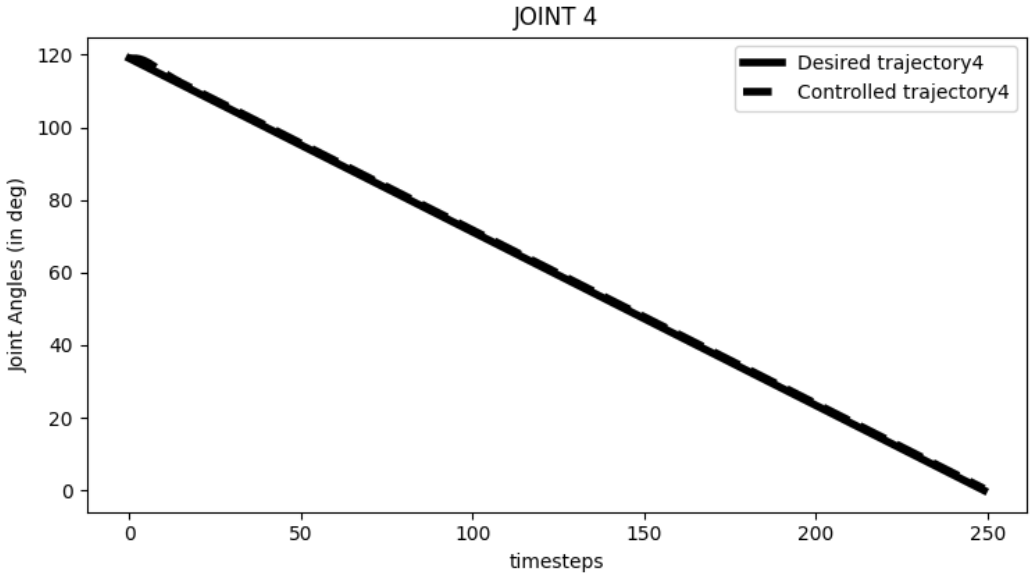


Figure 6-61: Trajectory Tracking for Joint 4 while Moving to Default Position

6.7.2 Actions Performed in Physical Environment

A. Robotic Arm at Default Position

At the beginning, Robotic Arm is rested at its default position with all joints angles at zero degrees.

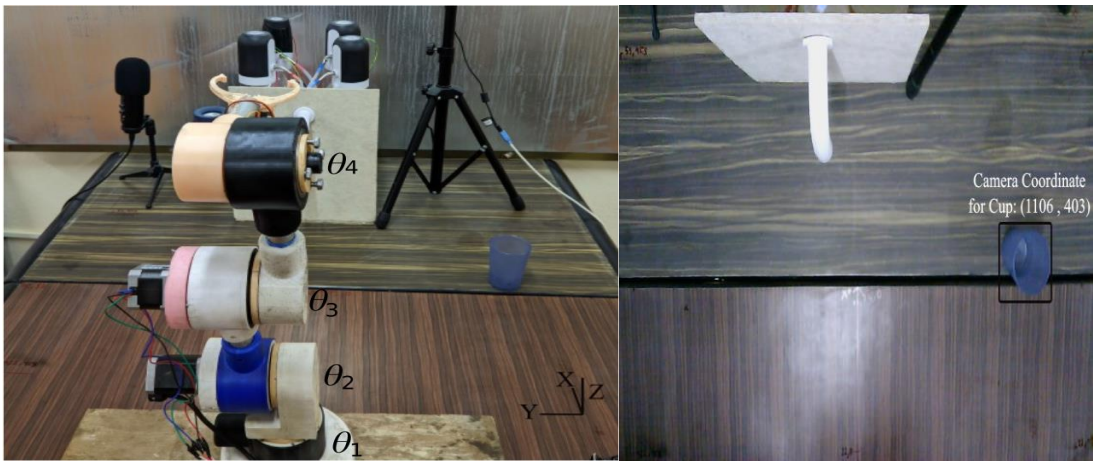


Figure 6-62: Default Position with Camera View

Table 6-13: Coordinated Mapped for Empty Glass (Physical)

Camera Coordinate	**Physical Coordinates(m)**
(1106,403)	(0.4163, -0.3603)

Table 6-14: Inverse Kinematics Parameters - 1

End Effector Position in Task Space (m)			**Joint Angles(degree)**			
X	Y	Z	θ_1	θ_2	θ_3	θ_4
0.20	0	0.57	0	0	0	0

B. Grabbing the Empty Glass

After receiving valid command of drink from the client, the glass location in the table is derived using overhead camera input, and inverse kinematics is performed to obtain the required joint angles.

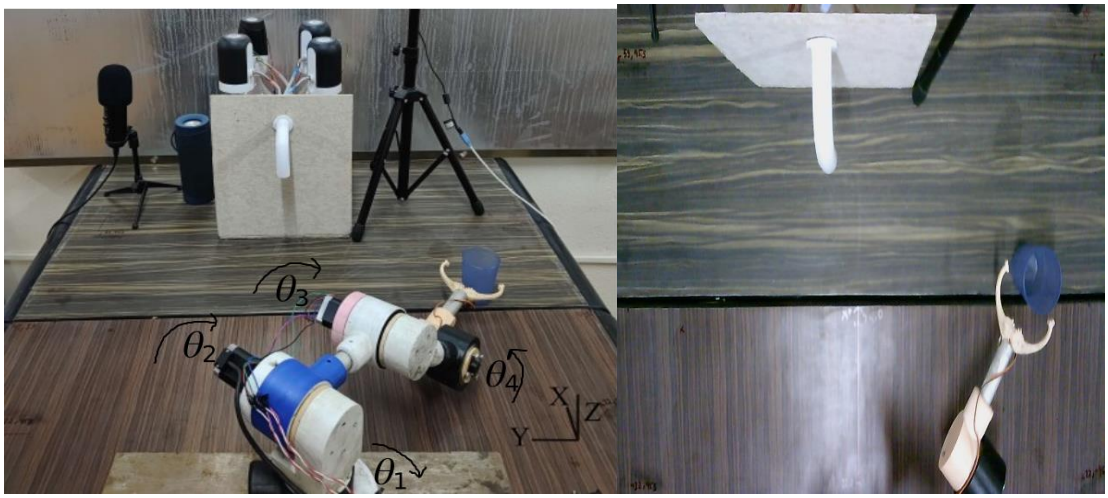


Figure 6-63: Grasping Empty Glass with Camera View

Table 6-15: Inverse Kinematics Parameters - 2

Desired End Effector Position in Task Space (m)			**Desired Joint Angles(deg)**			
X	Y	Z	θ_1	θ_2	θ_3	θ_4
0.4163	-0.3603	0.07	-40.9	-72	-41.7	107.7

Table 6-16: Change in Angles - 2

Change in Angles ($\Delta\theta$)			
$\Delta\theta_1$ (CW)	$\Delta\theta_2$ (CW)	$\Delta\theta_3$ (CW)	$\Delta\theta_4$ (CCW)
= -40.9 – 0 = -40.9	= -72 - 0 = -72	= -41.7 - 0 = -41.7	= 107.7 – 0 = 107.7

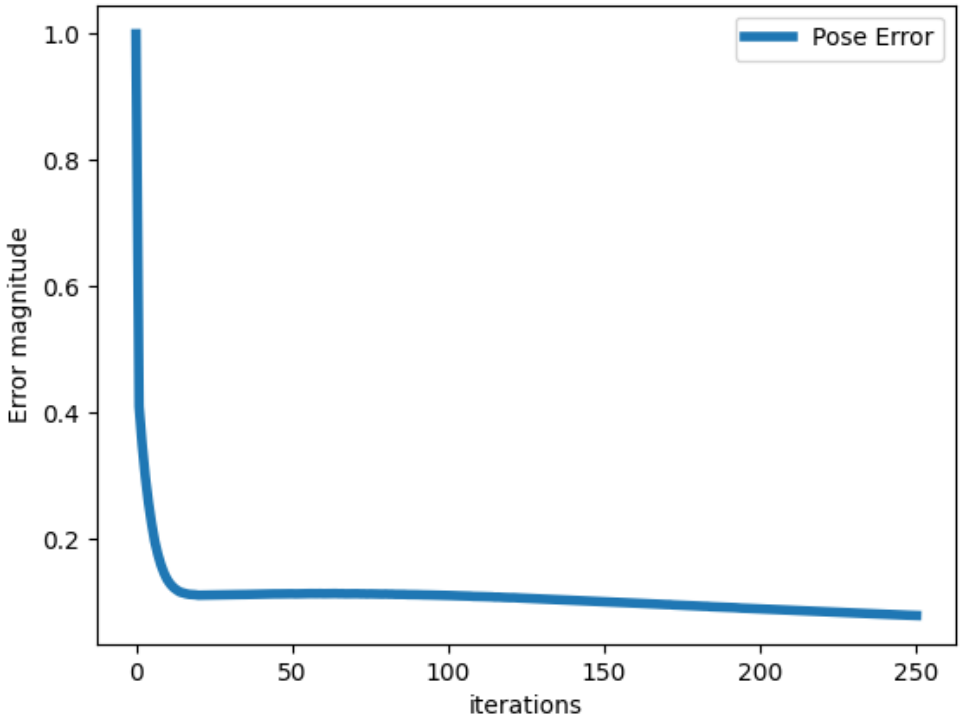


Figure 6-64: Pose Error while Grabbing the Cup

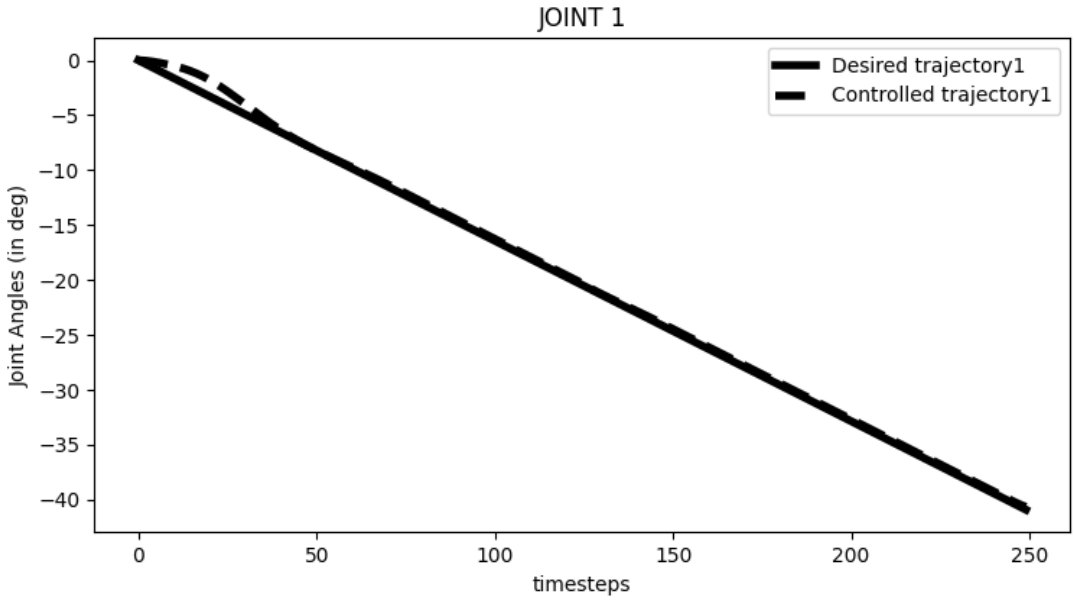


Figure 6-65: Trajectory Tracking for Joint 1 while Grabbing Empty Glass

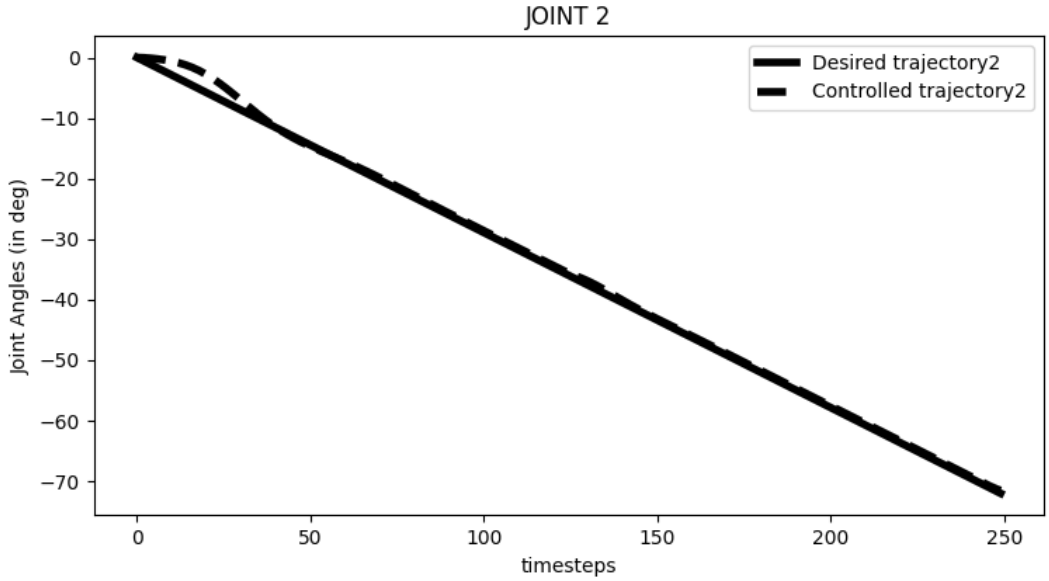


Figure 6-66: Trajectory Tracking for Joint 2 while Grabbing Empty Glass

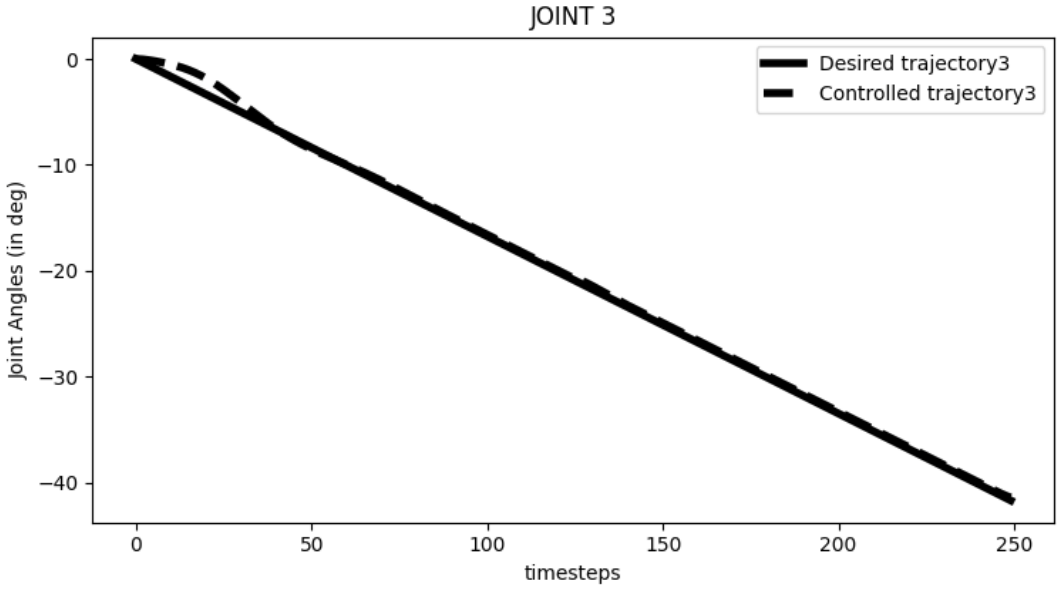


Figure 6-67: Trajectory Tracking for Joint 3 while Grabbing Empty Glass

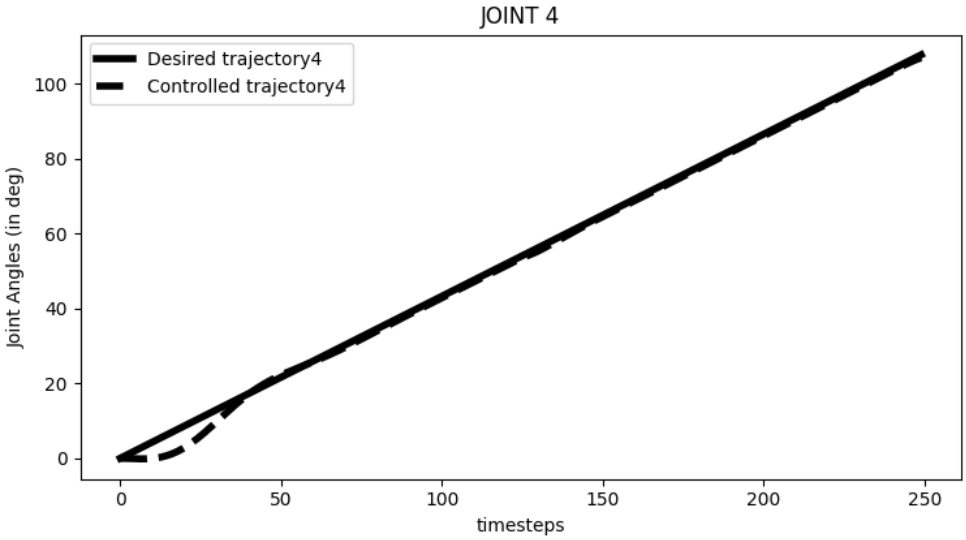


Figure 6-68: Trajectory Tracking for Joint 4 while Grabbing Empty Glass

C. Towards the Dispenser

After grabbing the cup, the arm reached towards dispenser to fill the cup.

Figure 6-69: Towards the Dispenser with Camera View

Table 6-17: Inverse Kinematics Parameters - 3

Desired End Effector Position in Task Space (m)			**Desired Joint Angles(deg)**			
X	Y	Z	θ_1	θ_2	θ_3	θ_4
0.4477	0.0316	0.20	3.6	-60.7	-48.5	104.4

Table 6-18: Change in Angles - 3

Change in Angles ($\Delta\theta$)			
$\Delta\theta_1$ (CCW)	$\Delta\theta_2$ (CCW)	$\Delta\theta_3$ (CW)	$\Delta\theta_4$ (CW)
= 3.6 - (-40.9) = 44.5	= -60.7 - (-72) = 11.3	= -48.5 - (-41.7) = -6.8	= 104.4 - 107.7 = -3.3

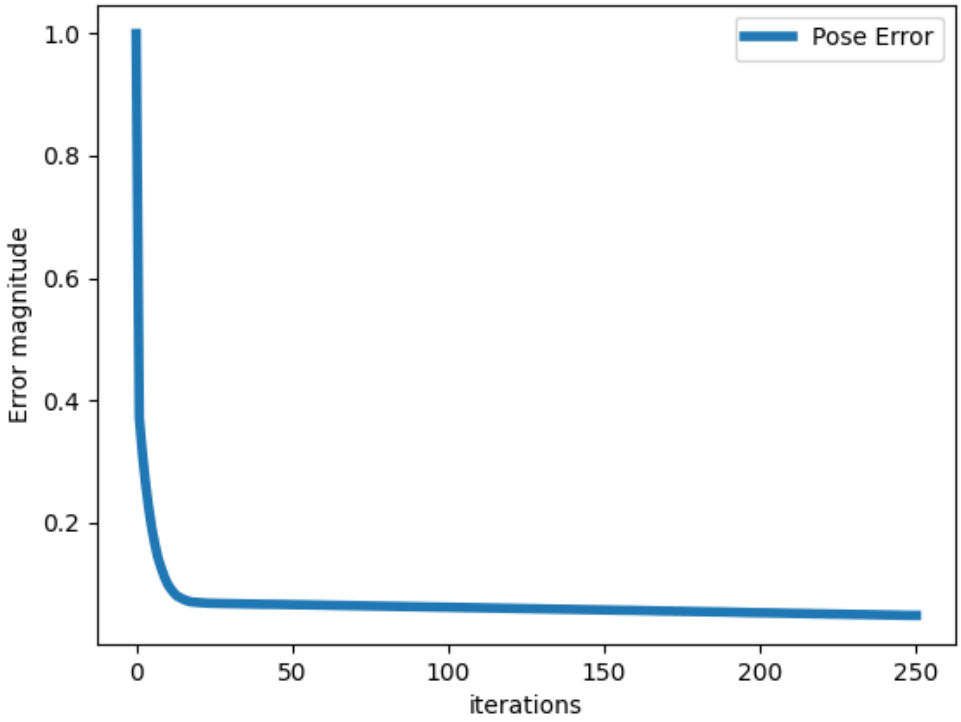


Figure 6-70: Pose Error while Filling the Cup

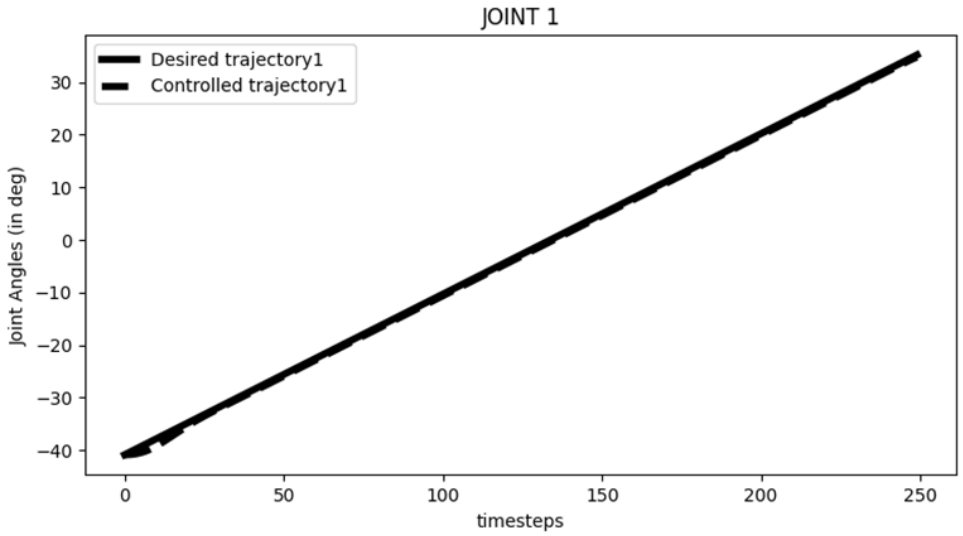


Figure 6-71: Trajectory Tracking for Joint 1 while Filling the Cup

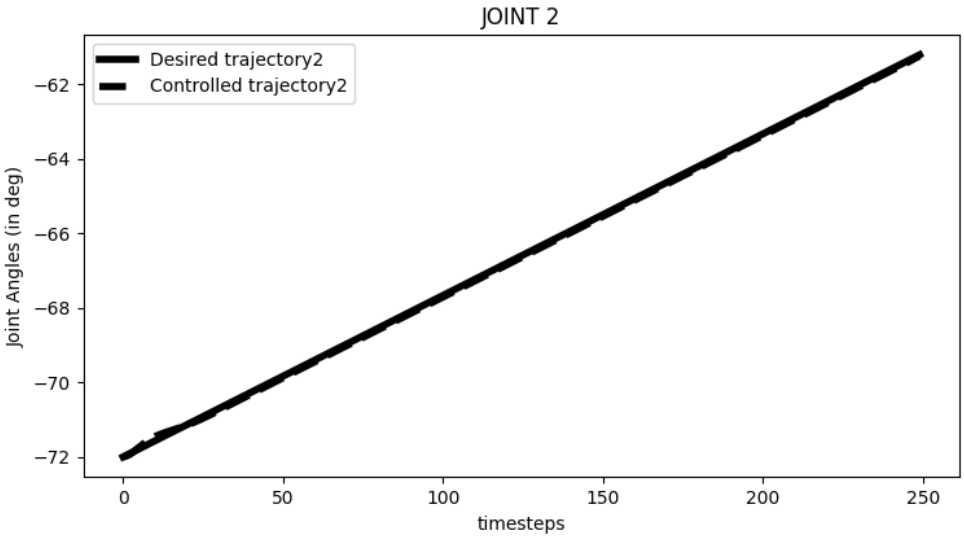


Figure 6-72: Trajectory Tracking for Joint 2 while Filling the Cup

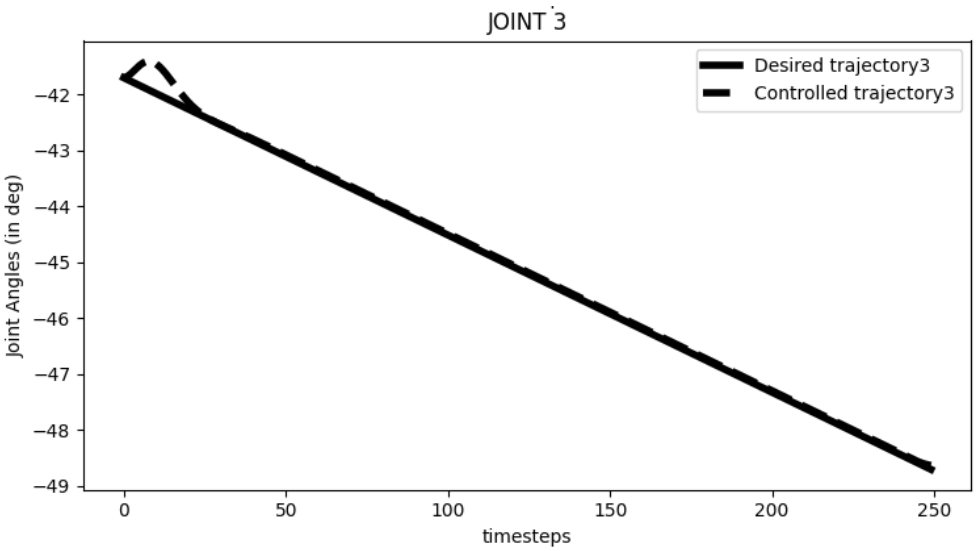


Figure 6-73: Trajectory Tracking for Joint 3 while Filling the Cup

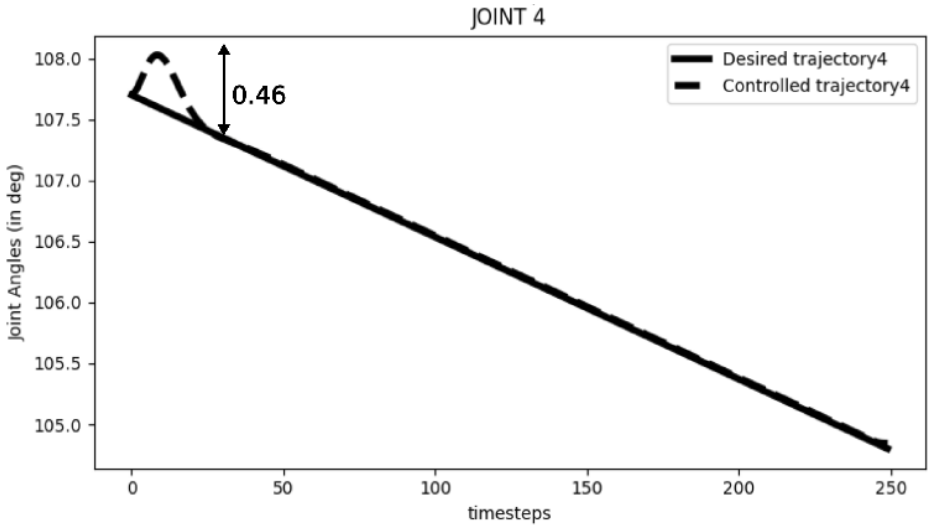


Figure 6-74: Trajectory Tracking for Joint 4 while Filling the Cup

D. Serving to Client

The robotic arm serves the drink to the client.

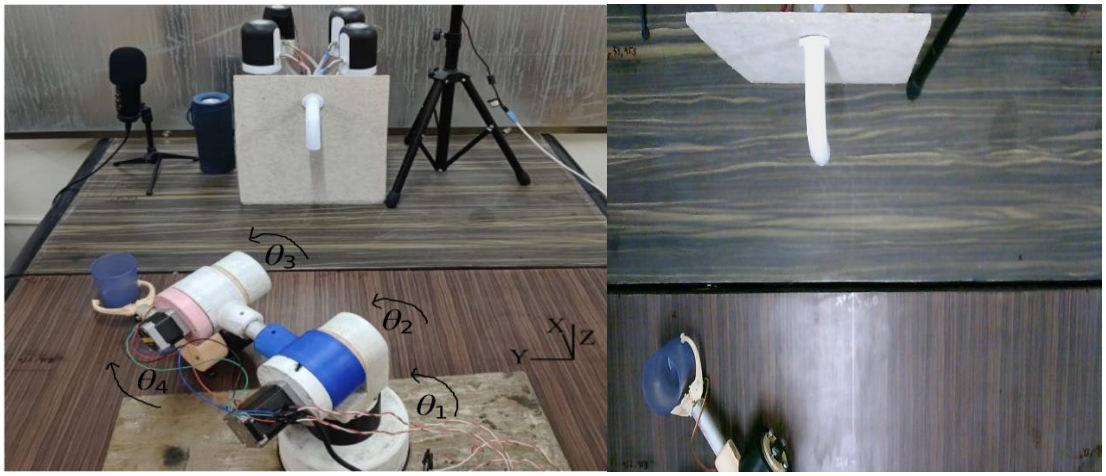


Figure 6-75: Serving to Client with Camera View

Table 6-19: Inverse Kinematics Parameters - 4

Desired End Effector Position in Task Space (m)			**Desired Joint Angles(degree)**			
X	Y	Z	θ_1	θ_2	θ_3	θ_4
0.3169	0.3701	0.07	49.3	-71	-54.7	119.4

Table 6-20: Change in Angles - 4

Change in Angles ($\Delta\theta$)			
$\Delta\theta_1$ (CCW)	$\Delta\theta_2$ (CW)	$\Delta\theta_3$ (CW)	$\Delta\theta_4$ (CCW)
= 49.3 - 3.6 = 45.7	= -71 - (-60.7) = -10.3	= -54.7 - (-48.5) = -6.2	= 119.4 - 104.4 = 15

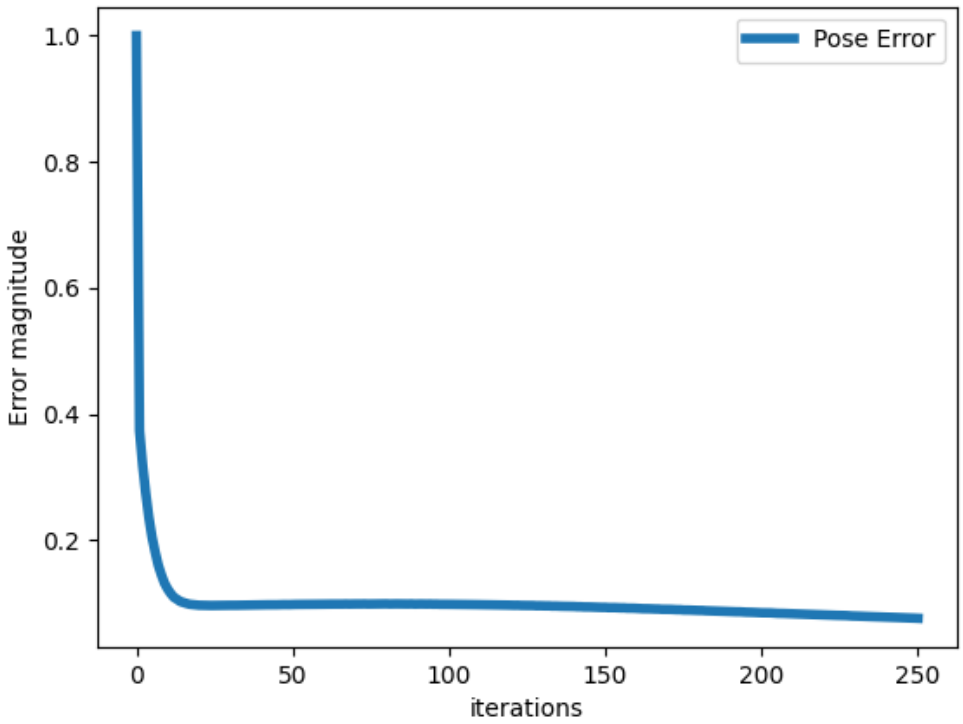


Figure 6-76: Pose Error while Serving Drink to Client

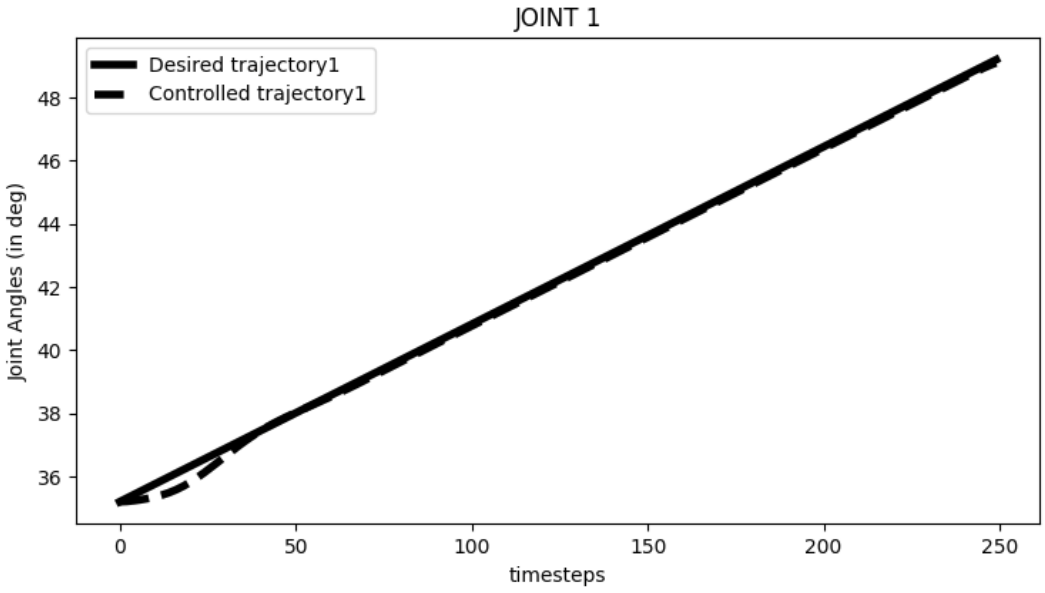


Figure 6-77: Trajectory Tracking for Joint 1 while Serving Drink to Client

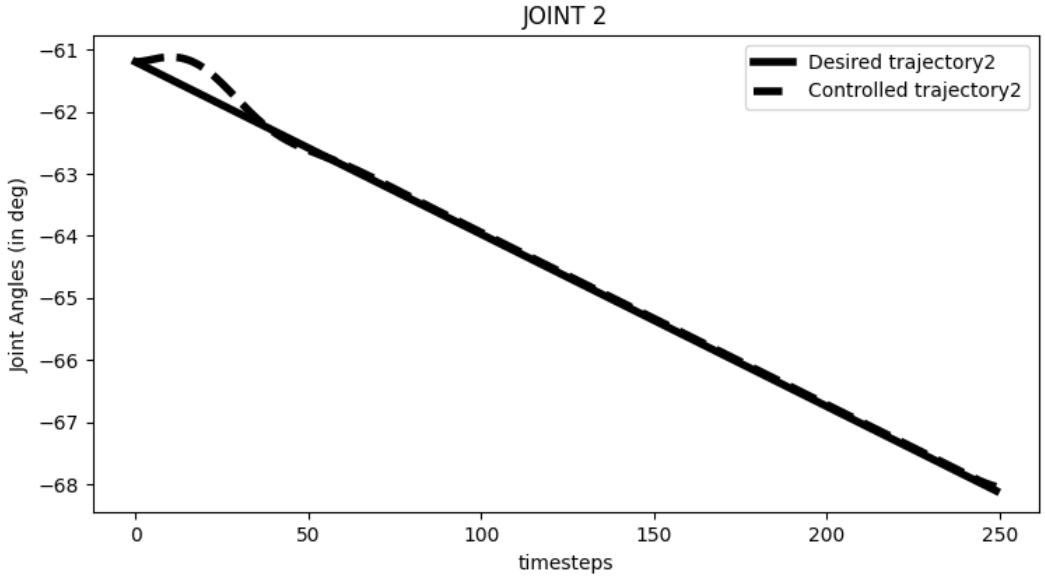


Figure 6-78: Trajectory Tracking for Joint 2 while Serving Drink to Client

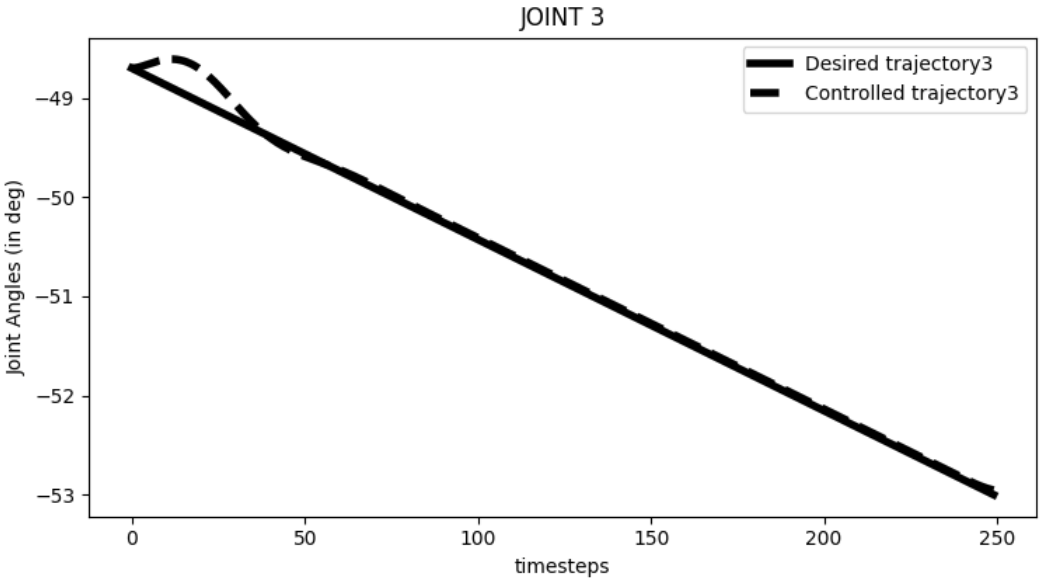


Figure 6-79: Trajectory Tracking for Joint 3 while Serving Drink to Client

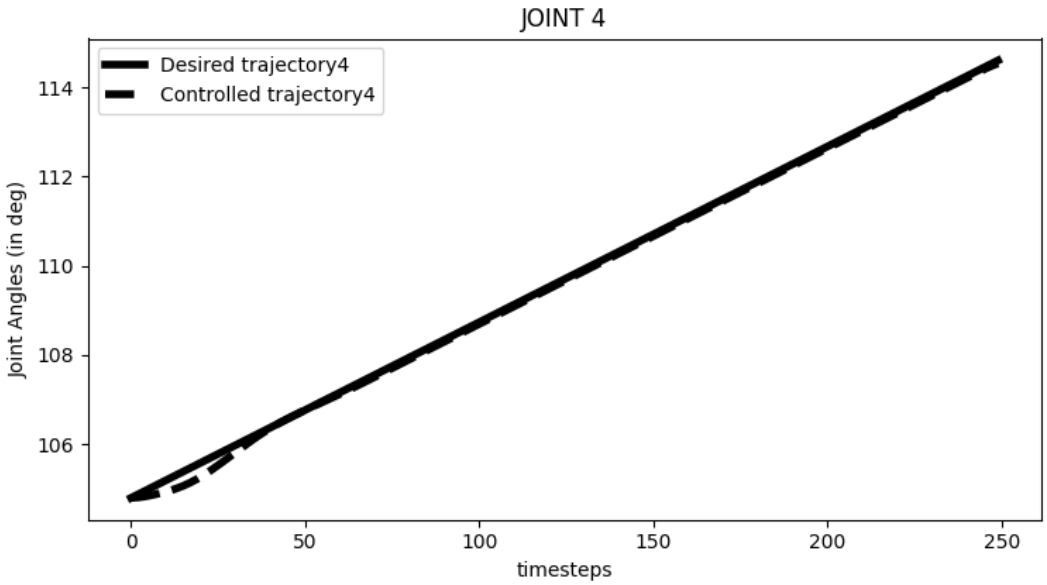


Figure 6-80: Trajectory Tracking for Joint 4 while Serving Drink to Client

E. Returning to Default Position

After serving the ordered drink to the client successfully, the robotic arm returned to the default position.

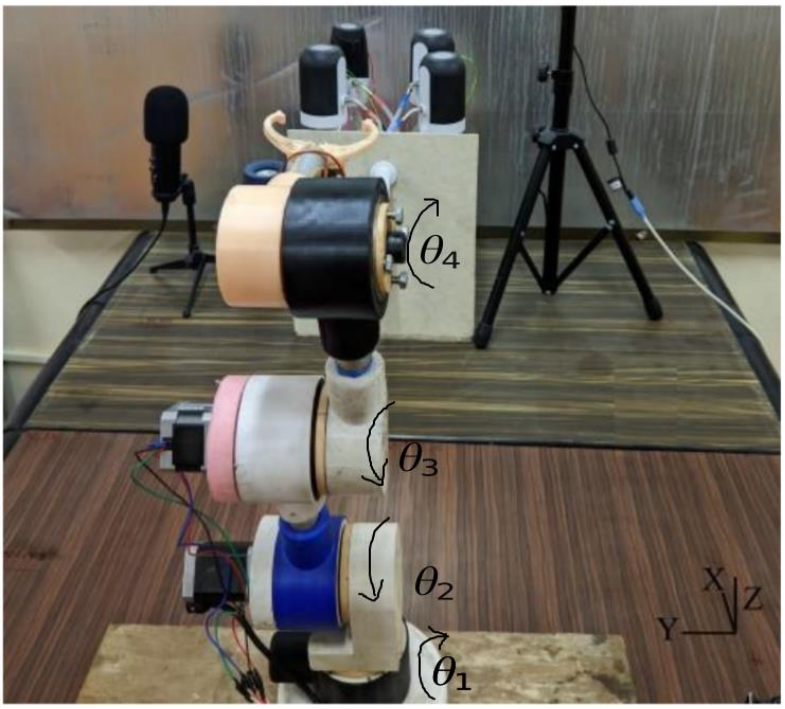


Figure 6-81: Returning to Default Position with Camera View

Table 6-21: Inverse Kinematics Parameters - 5

End Effector Position in Task Space (m)			**Deseired Joint Angles(degree)**			
X	Y	Z	θ_1	θ_2	θ_3	θ_4
0.20	0	0.57	0	0	0	0

Table 6-22: Change in Angles - 5

Change in Angles ($\Delta\theta$)			
$\Delta\theta_1$ (CW)	$\Delta\theta_2$ (CCW)	$\Delta\theta_3$ (CCW)	$\Delta\theta_4$ (CW)
= 0 - 49.3 = -49.3	= 0 - (-71) = 71	= 0 - (-54.7) = 54.7	= 0 - 119.4 = -119.4

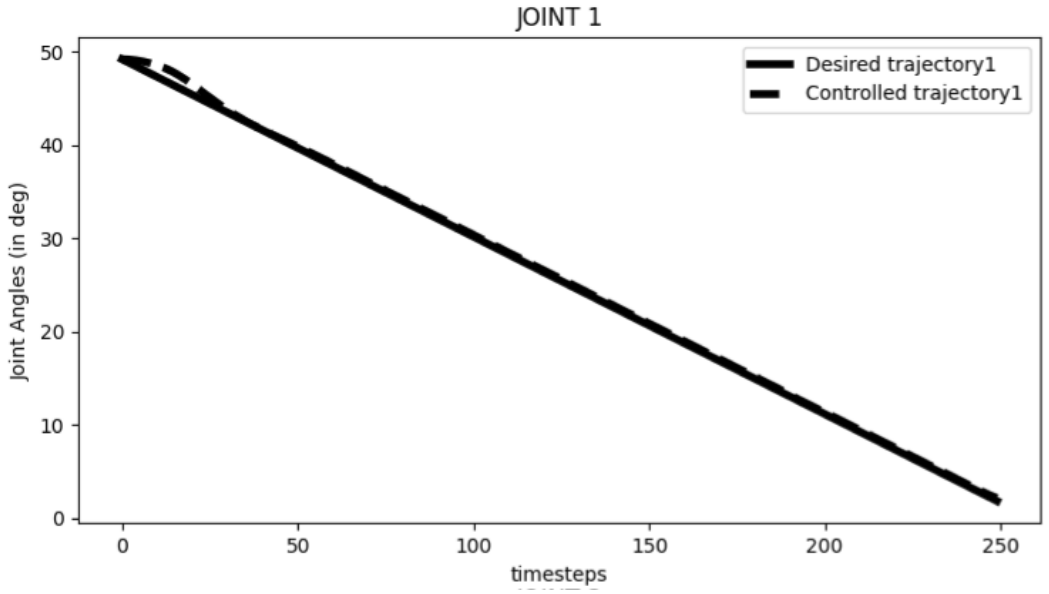


Figure 6-82: Trajectory Tracking for Joint 1 while Returning to Home

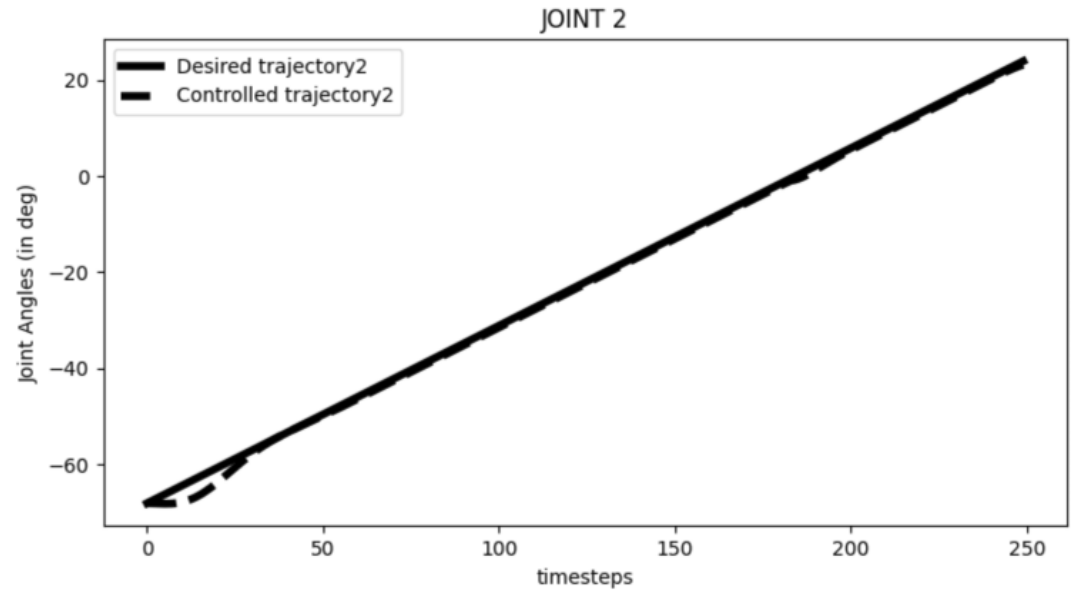


Figure 6-83: Trajectory Tracking for Joint 2 while Returning to Home

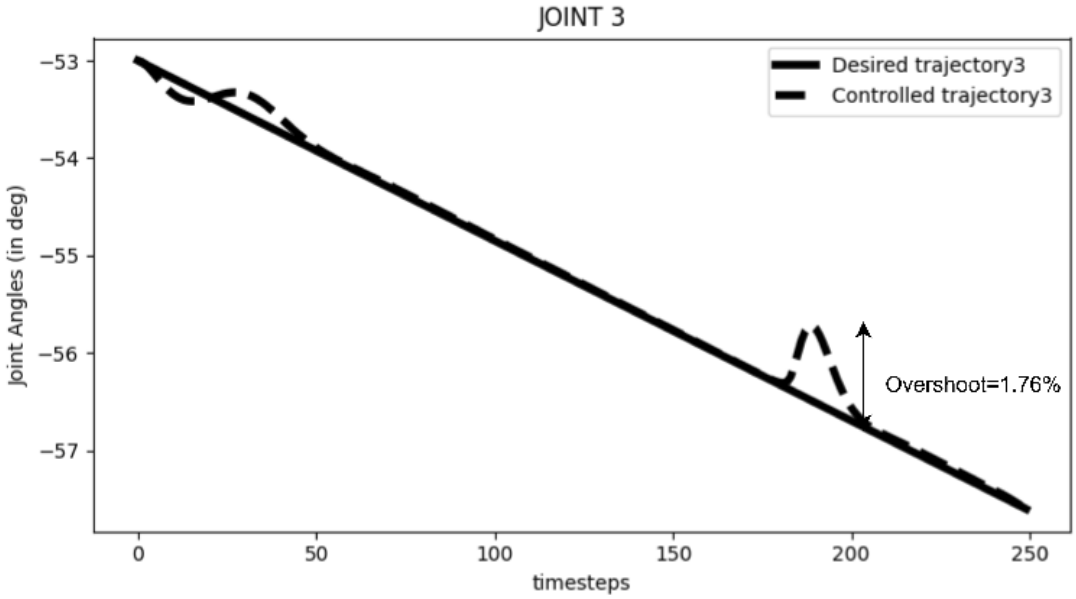


Figure 6-84: Trajectory Tracking for Joint 3 while Returning to Home

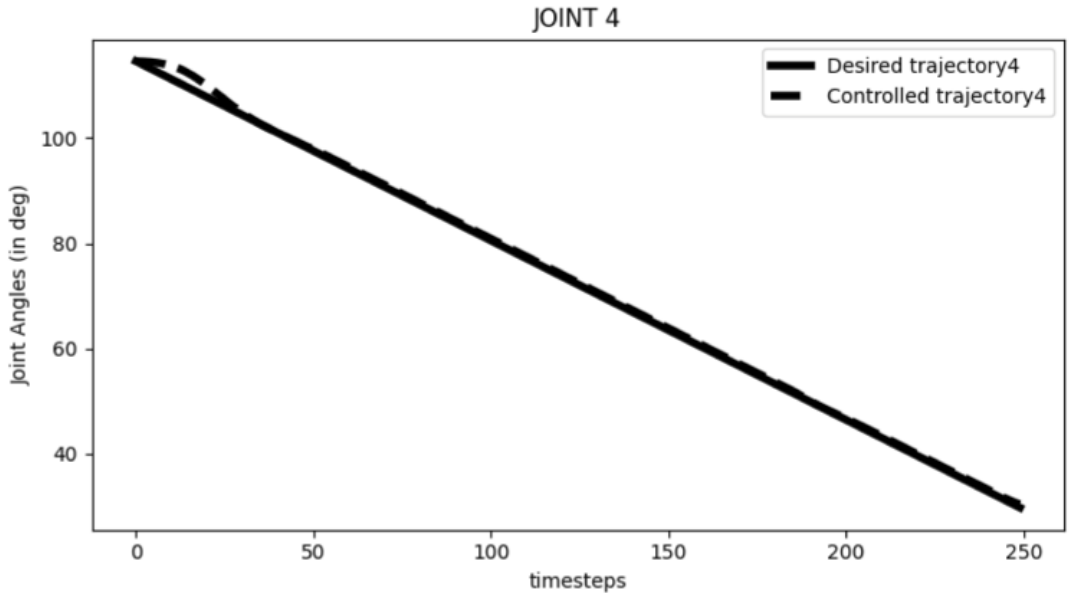


Figure 6-85: Trajectory Tracking for Joint 4 while Returning to Home

6.8 Anomalies

Various issues related to the vision and speech recognition models within the system prompt alerts to the client or operator through vocal feedback via the speaker. The following anomalies are identified along with their corresponding feedback.

6.8.1 Camera Sensor Blockage

When the system encounters an issue extracting frames through the camera sensor, typically due to the sensor being obstructed, the provided alert feedback is: "The vision is obstructed, Please check the camera."

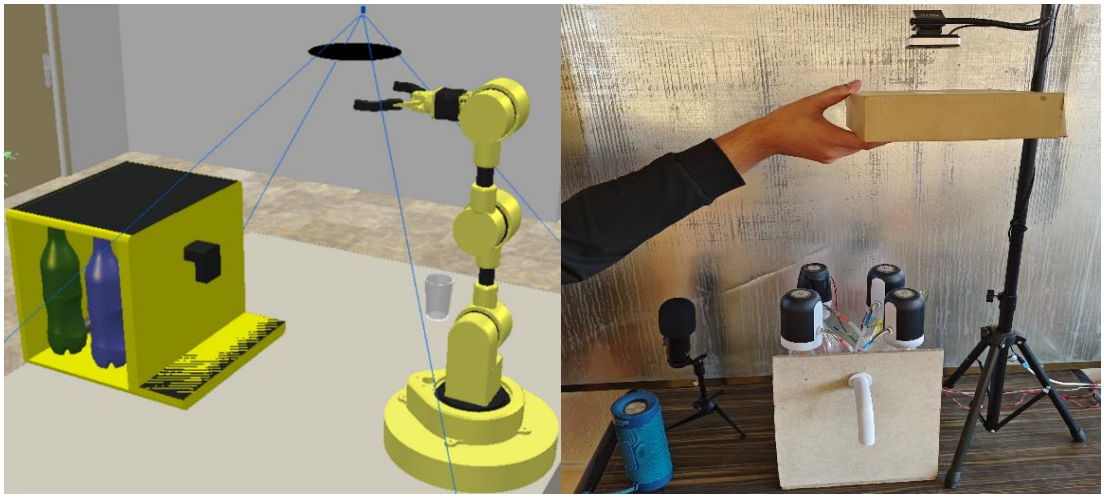


Figure 6-86: Camera Sensor Blockage (Left-Simulation and Right-Physical)

6.8.2 Path Obstruction

In the scenario where the path of the arm towards the dispenser is obstructed by an obstacle, the feedback provided is: "There is a obstacle in workspace, Please remove it."

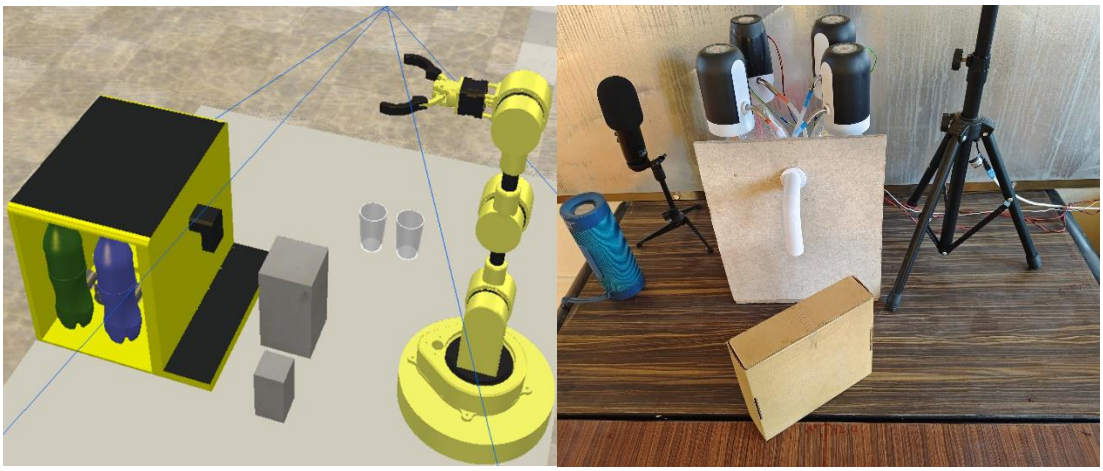


Figure 6-87: Dispenser Path Obstruction (Left-Simulation and Right-Physical)

6.8.3 Glass Unavailable

When detecting for an empty glass, if there are no glasses available for the arm to fill, the feedback provided is: "There are no glasses available. Kindly place some empty glasses on the table."

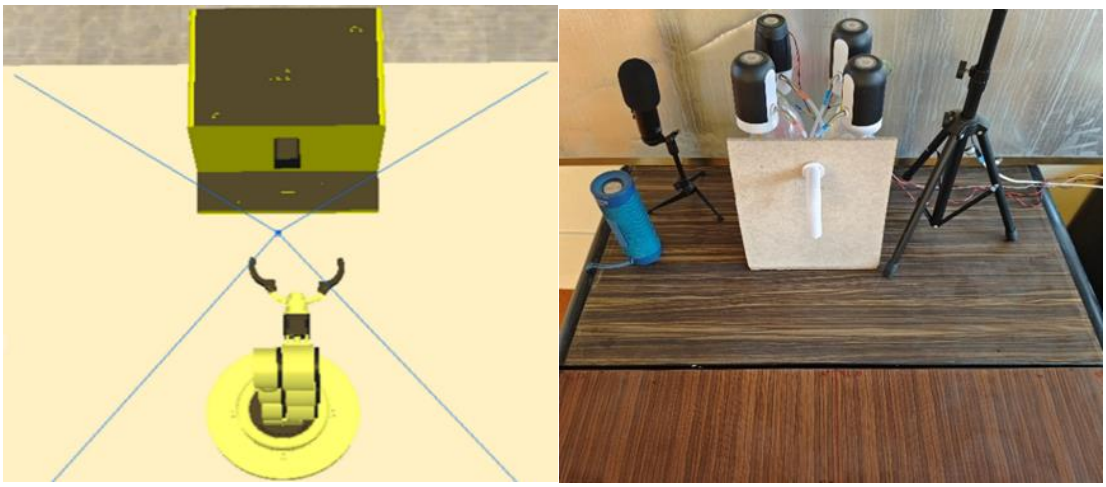


Figure 6-88: Glass Unavailable (Left-Simulation and Right-Physical)

6.8.4 Glass Unreachable

The arc in the table identifies the maximum reach of arm to grab empty glass. When the only remaining glasses on the tabletop are out of reach of the arm, the clients are alerted with the vocal feedback: "the glasses are out of reach, please place it within the reach of robotic arm"

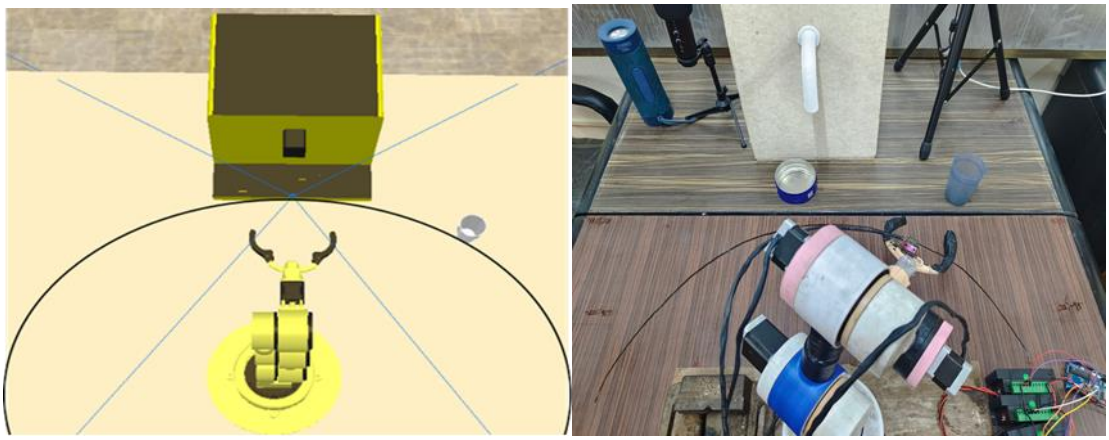


Figure 6-89: Glass Unreachable (Left-Simulation and Right-Physical)

6.8.5 Communication Error

If the system receives a command that cannot be interpreted, such as a missing drink name or wake word, it should provide feedback to the user indicating that the command cannot be understood and prompt them to repeat it. The feedback for such situation is: "Sorry, I cannot understand the order, please repeat."

Table 6-23: Possible Conditions for Communication Error Feedback

Transcribed	Remark
“provide me with cup of coffee”	No wake word
“hey arm, serve me a cup of tea”	Tea is not included in the system
“hey arm, give me a cup of ”	Missing drink name

6.9 Performance Comparison between Simulation and Reality

The following section presents a comparison of various project components between simulation and physical environments.

6.9.1 Speech Recognition Model

In comparing speech recognition model performance between simulated and real-world scenarios, it has been observed that while using pre-recorded, clean speech datasets, it achieves notably high accuracy and low error rates. However, when evaluated in real-world settings, where environmental factors such as background noise, and varying acoustic conditions prevail, the performance of this models often experienced a significant decline. Addressing this disparity involves developing new strategies and methodologies that enable speech recognition models to better generalize across diverse environments.

Table 6-24: Performance Comparison of Speech Recognition Model

Evaluation Metrics	**Noiseless Environment (%)**	**Noisy Environment (%)**
WER	7.08	15
CER	6.4	13

6.9.2 Object Detection Model

Table 6-25: Performance Comparison of Object Detection Model

Evaluation Metrics	**Simulation**	**Physical**
Box Loss	0.38233	0.3361
Class Loss	0.39422	0.30173
Precision Metrics	0.97882	0.99319
Recall Metrics	0.97295	0.98161
mAP@50	0.98941	0.98869

The table presents a comparison of evaluation metrics for an object detection model trained in both simulation and physical environments. It visually illustrates that the metrics for the model trained on physical datasets are notably superior. It was observed that the vision system operated smoothly in the simulation environment because the

camera remained static without any disruptions. However, in the physical environment, system disturbances necessitated frequent evaluation of the transformation matrix for coordinate mapping.

6.9.3 Design of Robotic Arm

A. Backlash

Backlash in a gearbox refers to the clearance or play between mating gears. It's a mechanical phenomenon that occurs due to imperfections in gear manufacturing and assembly. Essentially, it's the amount of movement or rotation that a gear can undergo before it engages with its mating gear.

Since the simulation didn’t include the intricate mechanism of cycloidal disk to roller pin interaction as well as the output torque coupling mechanism the joint didn’t have any backlash in the simulation resulting a smooth and precise motion.

The backlash in the gearbox was attributed to two primary factors. Firstly, it was from the output shaft that was machined on a lathe, which inherently introduced slight imperfections and tolerances in the gear meshing process. Secondly, tolerance variations inherent in 3D printing also contributed to the presence of backlash.

Figure 6-90: Assembled Gearbox with Bolts as Output Shaft

The 6mm bolts were turned in a lathe machine so that the bolts can have a smaller 5mm

diameter section of length 45mm. The cycloidal gearbox was designed so that the cycloidal disk can slide around the 5mm shaft and the hole diameter was determined according to the shaft diameter.

But after turning process the diameter was lot smaller than 5mm. The bolt had diameter in between 4mm to 4.5mm which was one of the reasons of the backlash in the gearbox.

Figure 6-91: Lathe Turned 6mm Bolts

To address the issue of the disk's inability to rotate smoothly due to imperfections in the 3D printer and inconsistencies in the designed dimensions, a 0.4mm tolerance was incorporated into the design, as depicted in the figure below. However, the combination of manually added tolerance and the printer's inherent tolerance led to noticeable disparities in the final 3D printed part. Consequently, the design showed a degree of looseness as well as backlash.

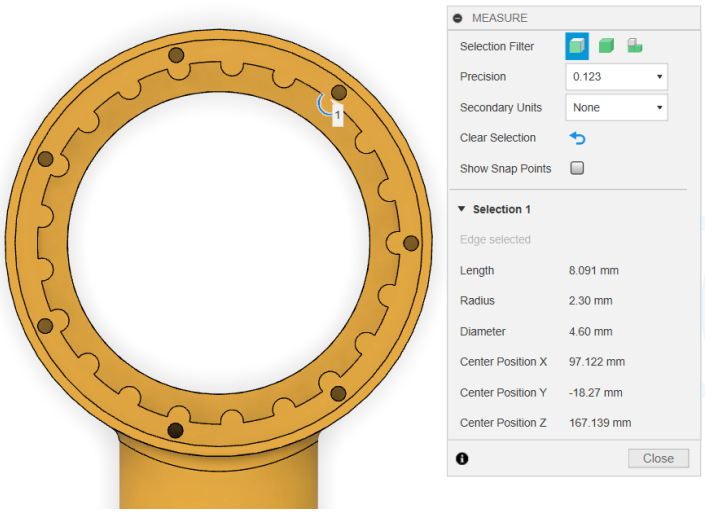


Figure 6-92: 5mm Ring Pins Reduced to 4.6mm

B. Gearbox Design

The cycloidal gearbox simulation couldn't be fully replicated due to the complex motion of the cycloidal disk. Therefore, in Fusion 360, the joint was simplified before exporting it to CoppeliaSim so that triangle counts of mesh file could be reduced and simulation could be rendered much faster.

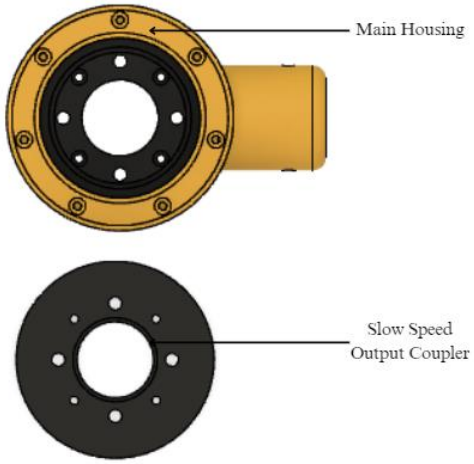


Figure 6-93: Simplification before Exporting to CoppeliaSim

The following figure shows all the components of the actual cyclodial gearbox which includes cycloidal disks, eccentric cams etc. which was absent in the simulation environment.

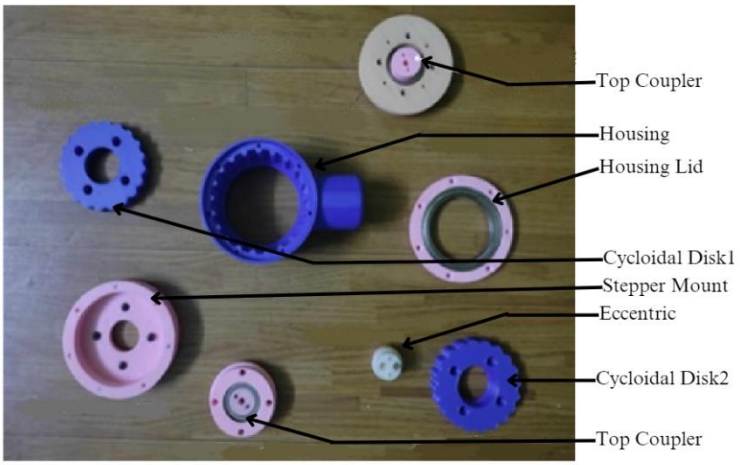


Figure 6-94: Actual Gearbox with More Components

C. Torque

In the simulation environment, the maximum torque was configured to 1000Nm to enable the robotic arm to lift objects reliably under any circumstances. It was observed that the arm struggled to lift loads when the torque was set to lower values. Essentially, the high torque setting ensures that the joint can consistently generate sufficient torque and can adapt automatically to the weights it needs.

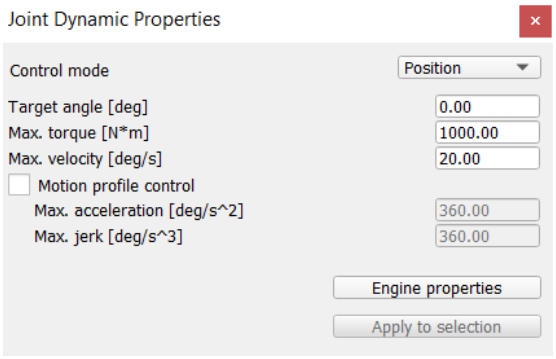


Figure 6-95: Joint Properties in CoppeliaSim

In reality, the maximum torque was constrained by factors such as the reduction ratio, reduction efficiency, and the maximum torque that the Nema 17 motor could produce. The Nema 17 motor could generate 25.32Ncm of torque, which was augmented to 2.61Nm with the gearbox. Consequently, the maximum torque parameter in reality would be 2.61Nm. This limitation meant that the arm could lift weights only up to the capacity required for carrying the cup and the drink.

6.9.4 Movement of Robotic Arm

There is no constraint on the value of torque in the simulation environment for the joints of the robotic arm. So, the robotic arm could perform any possible actions efficiently. For example, while returning to home position after serving drink to client, the virtual robotic arm moved easily.

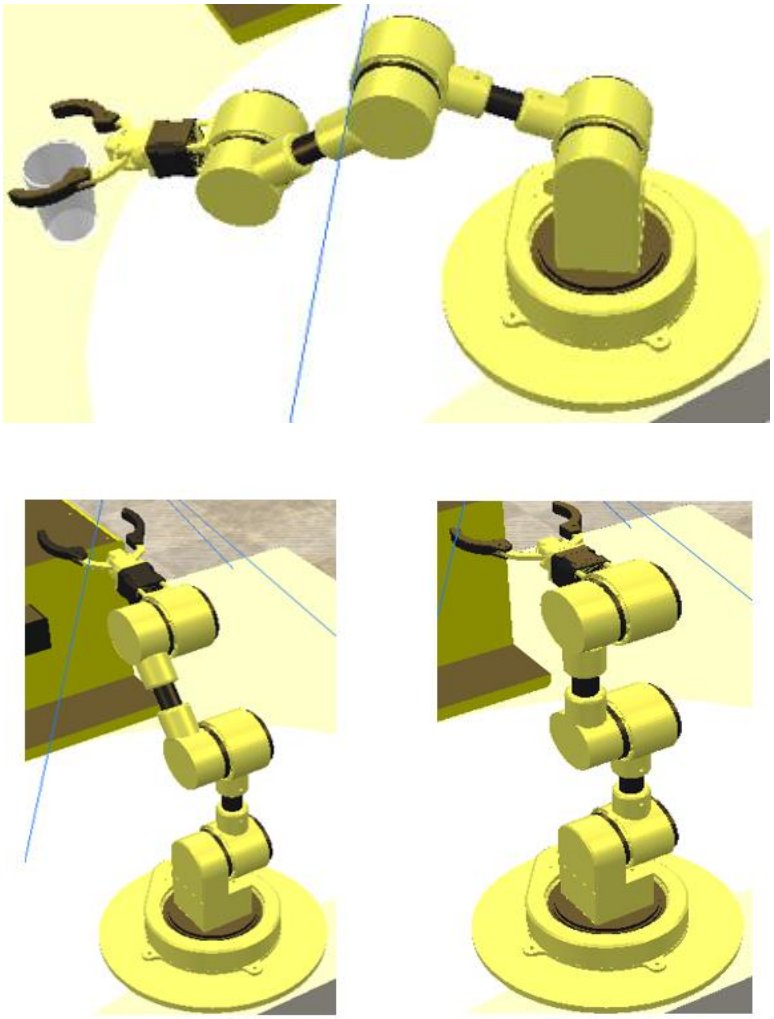


Figure 6-96: Virtual Arm Movement while Reaching to Default Position

But in case of the physical robotic arm, out of all four joints, the second joint needs the maximum torque of 5 Nm if it is to return to home position, after serving drink to client, without any jerky motion. But the stepper motor we have for second joint can only provide the maximum torque around 4.5Nm. Also, Due to the slippage in cycloidal gear box, it cannot move to the home position directly after serving drink to client.

So, to encounter this issue, rather than trying to lift the extended arm directly to the home position, we first rotate the fourth joint in the anticlockwise direction followed by the third joint in the same direction so that the second joint don't have to lift the extended arm which reduces the torque requirement drastically. This way, the physical robotic arm move to the home position efficiently.

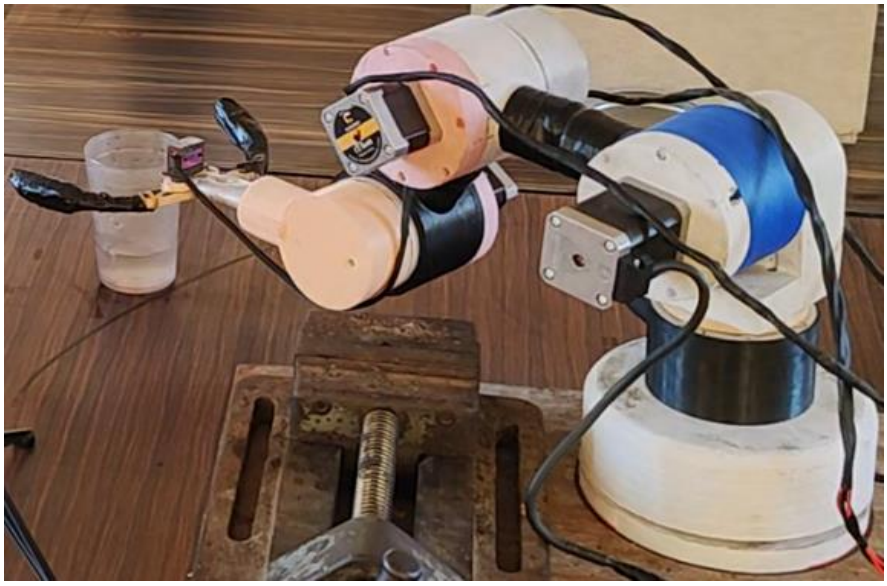


Figure 6-97: Physical Arm Movement while Reaching to Default Position

7. FUTURE ENHANCEMENT

For full movement of robotic arm in 3D space, it requires 6 degrees of freedom which includes translation and rotation across 3 axes. So, the robotic arm could be enhanced to make it a 6-DOF making it a redundant manipulator which allows multiple solution of inverse kinematics increasing flexibility. 3D printed parts introduces some kind of backlash due to various tolerance issues so using quality 3D printer with minimum tolerance could greatly increase the performance of the gearbox. Similarly, CNC-machined aluminum components could be used specially for cycloidal gearbox will enhance torque and reduction efficiency as well as precision in the robotic arm's operation. It could use Brushless DC Motor (BLDC) in joints for more precise control by adapting torque control of motors using the field-oriented control mechanism. Use of feedback sensors like AS9600 to get the current position of the joint can improve robotic arm’s reliability significantly. Currently the position tracking is done using software and the robotic arm must be manually changed to the initial defined position in case of power failure. Using such hall sensors, the problem will be solved.

The implementation of adaptive MPC algorithms that can adaptively update the model parameters online based on real-time data could lead to better performance in the presence of varying operating conditions. Vision system can be upgraded by adding depth cameras. These cameras help the system understand space better and track object accurately. Also, object detection model can be trained to recognize more things, like customers. This helps when serving drinks because the system can keep an eye on who it's serving. A speech recognition model can be trained to be multilingual, enabling its use by a diverse range of users. By incorporating datasets from various age groups, including older individuals, the model's scope can be significantly expanded. This approach not only enhances accessibility but also improves the inclusivity and effectiveness of the system for a wider user base.

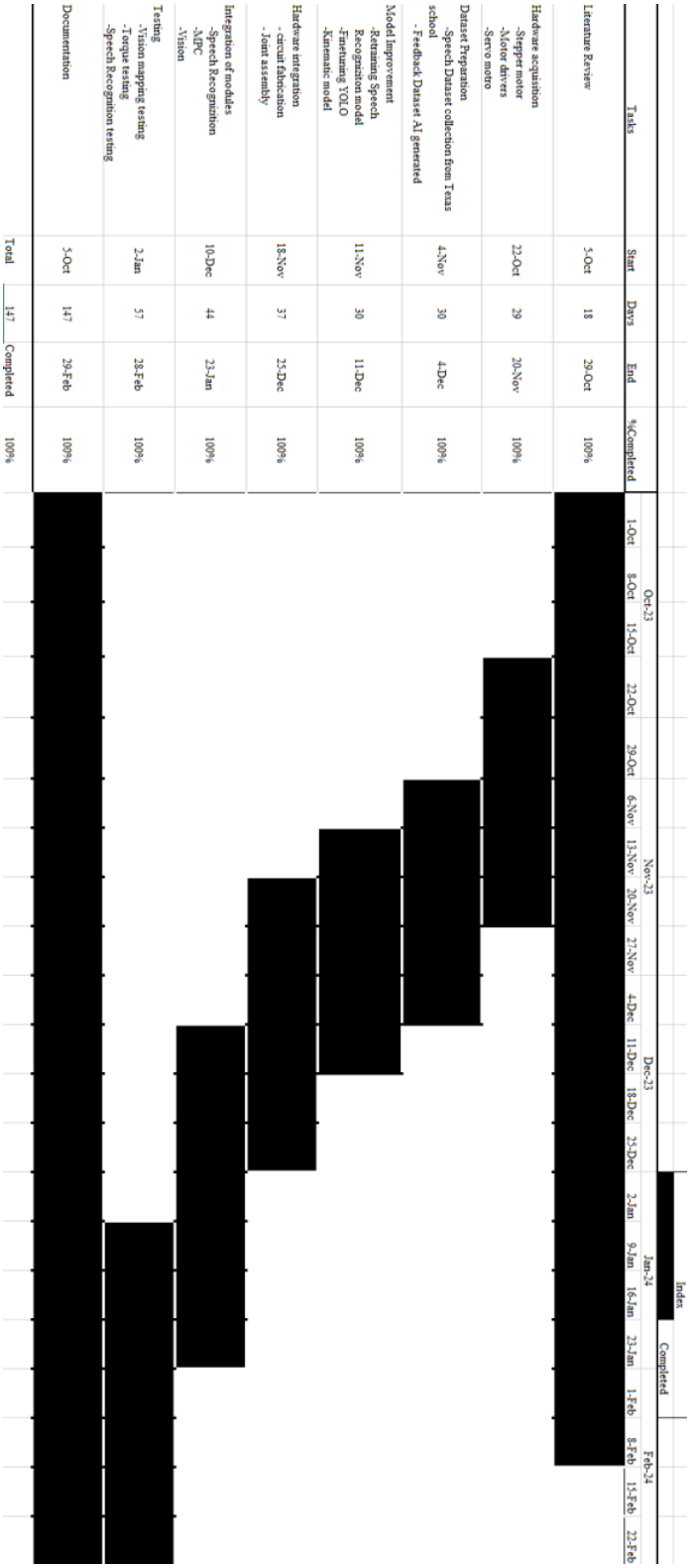
8. CONCLUSION

A voice-controlled 4-DOF robotic arm to automate the monotonous task of drink serving in a bartender domain is made. The complete functionality is first tested in the virtual environment as an objective for the Part-A of this project. Then, the 3D-printing of all parts of robotic arm like joints and cycloidal gear is done and the stepper motors, servo motor and motor controllers were purchased. The same models and algorithms that were used in the virtual environment for simulation is fine-tuned for the physical environment to have a complete working setup.

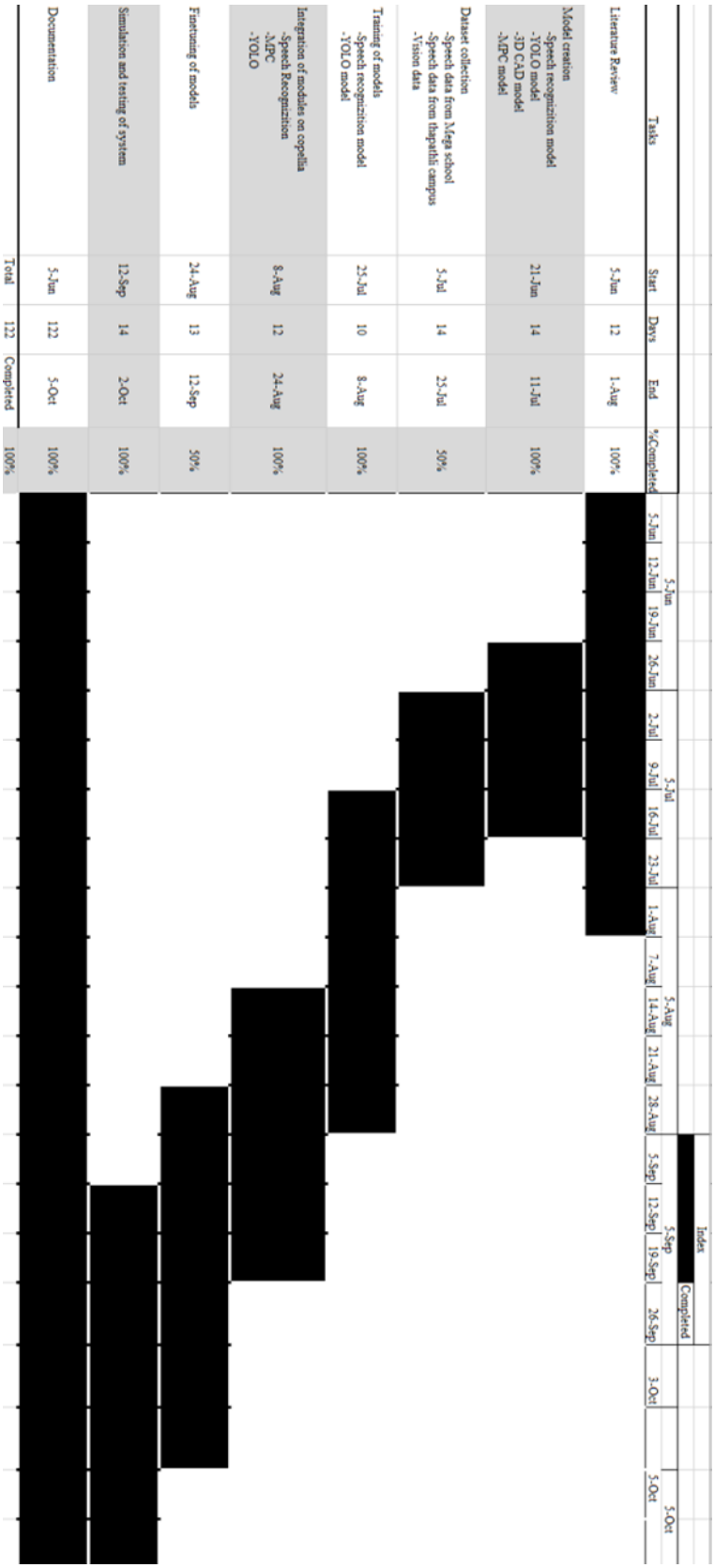
The project stands as a testament to the potential and opportunities inherent in the integration of hardware and software technologies. We highly recommend aspiring engineers, researchers, and developers to embark on projects that merge robotics, machine learning, and software development. These projects not only push the boundaries of technological innovation but also hold immense potential for creating impactful solutions across a wide range of industries and applications. By delving into projects like ours, individuals can gain a deeper understanding of the complexities involved in designing, implementing, and optimizing systems that seamlessly blend physical interaction with intelligent decision-making.

APPENDICES

Appendix A: Project Schedule (Part A)



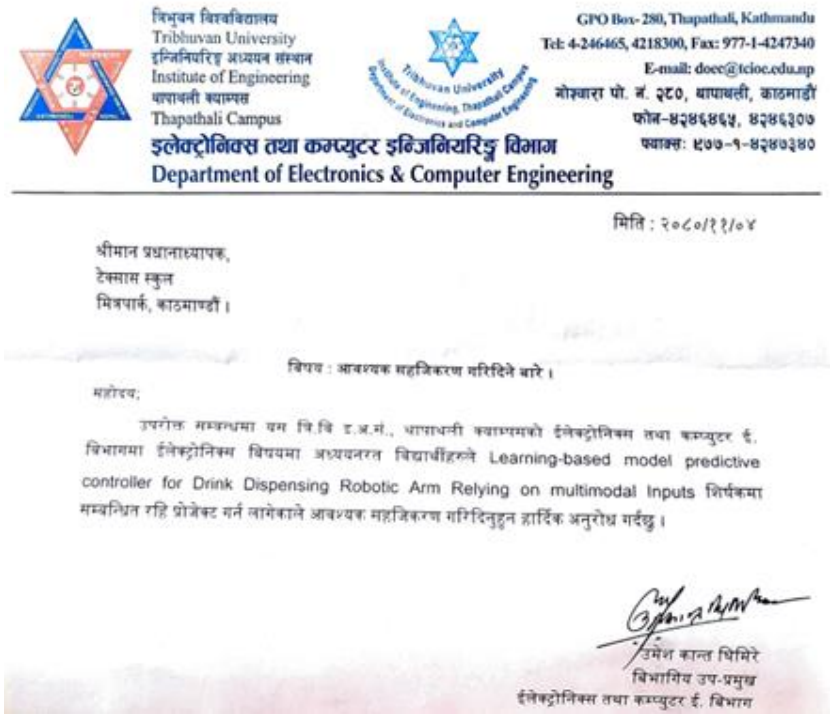
Appendix B: Project Schedule (Part B)



Appendix C: Budget Analysis

S.N.	Items	Description	Quantity	Unit Price (NPR)	Total (NPR)	Source
1.	6810	Ball bearings	8	1200	9600	Real Time Solutions
2.	6804	Ball bearings	16	400	6400	Real Time Solutions
3.	Nema 17	45 Ncm	3	1600	4800	Real Time Solutions
4.	Nema 17	60 Ncm	1	3320	3320	Real Time Solutions
5.	M3 screw	Screws	20	5	100	Real Time Solutions
6.	Camera	Vision	1	1700	1700	Real Time Solutions
7.	A4988	Stepper driver	4	210	840	Real Time Solutions
8.	6mm Bolts	Nut bolts	16	10	160	Self-Purchased
9.	Lathe Turning	For output shafts	16	100	1600	Self-Purchased
10.	DRV8825	Stepper driver	1	500	500	Self-Purchased
11.	3D printing	Gearbox 21 hours print time	4	1470	5880	Self-Purchased
12.	Grease	Lubricant	1	150	150	Self-Purchased
13.	Arduino	Stepper controller	1	1200	1200	Self-Purchased
14.	Dc Pump	Dispenser Pump	4	400	1600	Self-Purchased
15.	Raspberry pi 4B	Main controller	1	29500	29500	Self-Purchased
16.	TB6600	Stepper Driver	1	1200	4800	Self-Purchased
				Grand Total	**72,150**	

Appendix D: Dataset Collection (Texas International School)



त्रिभुवन विश्वविद्यालय
Tribhuvan University
इन्जिनियरिङ अध्ययन संस्थान
Institute of Engineering
थापाथली क्याम्पस
Thapathali Campus
इलेक्ट्रोनिक्स तथा कम्प्युटर इन्जिनियरिङ्ग विभाग
Department of Electronics & Computer Engineering

GPO Box- 280, Thapathali, Kathmandu
Tel: 4-246465, 4218300, Fax: 977-1-4247340
E-mail: doec@tcioe.edu.np
पोस्टबक्स नं. २८०, थापाथली, काठमाडौं
फोन-४२४६४६५, ४२४६३०७
फ्याक्स: ९७७-१-४२४७३४०

मिति : २०८०/११/०४

श्रीमान प्रधानाध्यापक,
टेक्सास स्कुल
मित्रपार्क, काठमाण्डौं ।

विषय : आवश्यक सहजिकरण गरिदिने बारे ।

महोदय;

उपरोक्त सम्बन्धमा यस त्रि.वि इ.अ.सं., थापाथली क्याम्पसको ईलेक्ट्रोनिक्स तथा कम्प्युटर ई. विभागमा ईलेक्ट्रोनिक्स विषयमा अध्ययनरत विद्यार्थीहरुले Learning-based model predictive controller for Drink Dispensing Robotic Arm Relying on multimodal Inputs शिर्षकमा सम्बन्धित रहि प्रोजेक्ट गर्न लागेकाले आवश्यक सहजिकरण गरिदिनुहुन हार्दिक अनुरोध गर्दछु ।

उमेश कान्त घिमिरे
विभागिय उप-प्रमुख
ईलेक्ट्रोनिक्स तथा कम्प्युटर ई. विभाग

The image shows an official letter of request prepared to obtain permission for collecting data from students of Texas school.

The image depicts students from a Texas school participating in the recording of audio for voice data collection purposes.

Appendix E: Dataset Collection (Nepal Mega College)

त्रिभुवन विश्वविद्यालय
Tribhuvan University
इन्जिनियरिङ अध्ययन संस्थान
Institute of Engineering
थापाथली क्याम्पस
Thapathali Campus
इलेक्ट्रोनिक्स तथा कम्प्युटर इन्जिनियरिङ विभाग
Department of Electronics & Computer Engineering

GPO Box- 280, Thapathali, Kathmandu
Tel: 4-246465, 4218300, Fax: 977-1-4247340
E-mail: doec@tcioe.edu.np
गोश्वारा पो. बं. २८०, थापाथली, काठमाडौं
फोन-४२४६४६५, ४२४६३०७
फ्याक्स: ९७७-१-४२४०३४०

मिति : २०८/०९/२४

श्रीमान प्रधानाध्यापक,
नेपाल मेगा स्कुल
अनामनगर, काठमाण्डौं ।

Tribhuvan University, Institute of Engineering, Thapathali Campus, Department of Electronics and Computer Engineering

विषय : आवश्यक सहजिकरण गरिदिने बारे ।

महोदय,

उपरोक्त सम्बन्धमा यस त्रि.वि इ.अ.सं., थापाथली क्याम्पसको ईलेक्ट्रोनिक्स तथा कम्प्युटर ई. विभागमा ईलेक्ट्रोनिक्स विषयमा अध्ययनरत विद्यार्थीहरुले Learning-based model predictive controller for Drink Dispensing Robotic Arm Relying on multimodal Inputs शिर्षकमा सम्बन्धित रही प्रोजेक्ट गर्न लाग्नेकाले आवश्यक सहजिकरण गरिदिनुहुन हार्दिक अनुरोध गर्दछु ।

किरण चन्द दाहाल
विभागिय प्रमुख
ईलेक्ट्रोनिक्स तथा कम्प्युटर ई. विभाग

The image shows an official letter of request prepared to obtain permission for collecting data from students at a Nepal Mega school

The image depicts students from a Nepal Mega school participating in the recording of audio for voice data collection purposes.

References

[1] Zhou, Z., Zhang, Y. and Li, Y. (2023) 'Model predictive control design of a 3-dof robot arm based on recognition of spatial coordinates', 2023 9th International Conference on Mechatronics and Robotics Engineering (ICMRE), doi:10.1109/icmre56789.2023.10106581.

[2] Voice Transformer Network: Sequence-to-Sequence Voice. https://www.isca-speech.org/archive_v0/Interspeech_2020/pdfs/1066.pdf

[3] L. Dong, S. Xu and B. Xu, "Speech-Transformer: A No-Recurrence Sequence-to-Sequence Model for Speech Recognition," 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 2018, pp. 5884-5888, doi: 10.1109/ICASSP.2018.8462506.

[4] S. Revay and M. Teschke, "Multiclass Language Identification using Deep Learning on Spectral Images of Audio Signals," May 2019, [Online]. Available: http://arxiv.org/abs/1905.04348

[5] L. Rafael Stefanel Gris and A. Candido Junior, "Automatic Spoken Language Identification using Convolutional Neural Networks." [Online]. Available: http://www.freesound.org

[6] P. Kaur, Q. Wang, and W. Shi, "Fall Detection from Audios with Audio Transformers," Aug. 2022, [Online]. Available: http://arxiv.org/abs/2208.10659

[7] A. A. Q. Mohammed, J. Lv, and M. D. S. Islam, "A deep learning-based end-to-end composite system for hand detection and gesture recognition," *Sensors (Switzerland)*, vol. 19, no. 23, Dec. 2019, doi: 10.3390/s19235282.

[8] M. Musaev, I. Khujayorov, and M. Ochilov, "Image Approach to Speech Recognition on CNN," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Sep. 2019. doi: 10.1145/3386164.3389100.

[9] Y. Gong, Y.-A. Chung, and J. Glass, "AST: Audio Spectrogram Transformer," Apr.

2021, [Online]. Available: http://arxiv.org/abs/2104.01778

[10] R. P. A. Petrick and M. E. Foster, “Planning for Social Interaction in a Robot Bartender Domain.” [Online]. Available: www.aaai.org

[11] A. V. Oppenheim and A. S. Willsky, Signals and Systems, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 1996. [Online]. Available:https://www.academia.edu/37486178/Signals_and_Systems_2nd_Edition_by_Oppenheim_

[12] Ultralytics, "Ultralytics/ultralytics: Open-source deep learning inference & training on YOLOv3/YOLOv4/PyTorch," GitHub. [Online]. Available: https://github.com/ultralytics/ultralytics