Final Year Project Report
on

Imitation of Human Arm on Robotic Manipulator

Submitted
in partial fulfillment of
the requirements of the Degree of Bachelor Of Technology

by

Vishnu Harikish Parammal (171090009)
Vednarayan Sriram Iyer (171090045)
Laukik Ashok Hase (171090052)
Ameya Vinay Apte (171090061)

Under the Guidance of

Dr. Faruk S. Kazi

DEPARTMENT OF ELECTRICAL ENGINEERING
VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE
(An Autonomous Institute Affiliated to Mumbai University)
(Central Technological Institute, Maharashtra State)
Matunga, MUMBAI - 400019

A.Y. 2020-2021

DECLARATION OF STUDENT

I declare that work embodied in this project titled "***Imitation of Human Arm on Robotic Manipulator***" forms my own contribution of work under the guidance of Dr. Faruk Kazi at the Department of Electrical Engineering, Veermata Jijabai Technological Institute, Mumbai. The report reflects the work done during the project.

Parammal Vishnu Harikish
Roll No. - 171090009
Date:
Place: VJTI, Mumbai

Iyer Vednarayan Sriram
Roll No. - 171090045
Date:
Place: VJTI, Mumbai

Hase Laukik Ashok
Roll No. - 171090052
Date:
Place: VJTI, Mumbai

Apte Ameya Vinay
Roll No. - 171090061
Date:
Place: VJTI, Mumbai

APPROVAL SHEET

The credit report "***Imitation of Human Arm on Robotic Manipulator***" by Vishnu Parammal, Vednarayan Iyer , Laukik Hase and Ameya Apte is found to be satisfactory and is approved for the Degree of Bachelor of Technology.

Prof. Dr. Faruk Kazi
Supervisor

Examiner

Examiner

Examiner

Place: *Veermata Jijabai Technological Institute, Mumbai.*
Date: / /

CERTIFICATE

This is to certify that Vishnu Parammal (171090009), Vednarayan Iyer (171090045), Laukik Hase (171090052) and Ameya Apte (171090061) students of B. Tech (Electronics and Telecommunication), Veermata Jijabai Technological Institute, Mumbai have successfully completed the project titled "***Imitation of Human Arm on Robotic Manipulator***" under the guidance of Prof. Dr. Faruk Kazi.

Prof. Dr. Faruk Kazi	Prof. Dr. Faruk Kazi
Supervisor	***Head of Electrical Department***

Place: *Veermata Jijabai Technological Institute, Mumbai.*
Date: / /

Contents

List of Tables iv

List of Figures v

List of Abbreviations vi

1 MOTIVATION 1
- 1.1 Technical Keywords . . . 1

2 LITERATURE SURVEY 2
- 2.1 Gap of Literature Survey . . . 3
- 2.2 Summary of Literature Survey . . . 3

3 PROJECT OBJECTIVES 4

4 PROPOSED SYSTEM AND IMPLEMENTATION 5
- 4.1 Hardware . . . 7
  - 4.1.1 Joint angle extraction . . . 7
  - 4.1.2 Manipulator . . . 8
- 4.2 Implementation . . . 9
  - 4.2.1 Joint Angle Extraction . . . 9
  - 4.2.2 Manipulator Simulation . . . 10
  - 4.2.3 Manipulator Control . . . 11

5 RESULTS 12
- 5.1 Joint angle extraction using IMU . . . 12
- 5.2 Comparing joint torques: RBDL v CoppeliaSim . . . 13
- 5.3 Results from RBDL under different sampling rates . . . 14
- 5.4 Comparison between quartic and cubic curve fit for Dynaban . . . 15
- 5.5 Dynamixel Motor Torque Analysis . . . 16
- 5.6 Comparison between position control and torque control imitation . . . 17
- 5.7 Drawing a Circle . . . 21
- 5.8 Pick and Place: Real Time . . . 22
- 5.9 Memorise and Repeat . . . 23

6 PLATFORM USED 24
- 6.1 Hardware . . . 24
  - 6.1.1 IMU . . . 24
  - 6.1.2 ESP32 DevKitC . . . 25
  - 6.1.3 Dynamixel . . . 25
- 6.2 Software . . . 25
  - 6.2.1 ESP-IDF . . . 25

6.2.2 MQTT . . . 25
6.2.3 ROS . . . 25
6.2.4 CoppeliaSim . . . 25
6.2.5 RBDL . . . 25
6.2.6 Pypot . . . 26
6.2.7 Dynaban . . . 26

7 CONCLUSION & FUTURE SCOPE **27**

Bibliography **28**

List of Tables

5.1 Inter-angle dependence: Pearson's Correlation Coefficient 13
5.2 Similarity for Quartic and Cubic curve fits: Percentage Normalised Dot Product 16
5.3 Similarity For Position and Torque control: Percentage Normalised Dot Product 20

6.1 9 axis IMU 24

List of Figures

2.1 Imitation using Kinect sensor . . . 3

4.1 IMU Motion Capture: Process Flow . . . 5
4.2 Manipulator Process: Flow . . . 6
4.3 Master PCB . . . 7
4.4 BNO055 Breakout . . . 7
4.5 Master PCB Schematic . . . 7
4.6 BNO055 Schematic . . . 7
4.7 Manipulator: Front View . . . 8
4.8 Manipulator: Side View . . . 8
4.9 Gripper . . . 9
4.10 URDF . . . 9
4.11 Reference Sensor Placement . . . 10
4.12 IMU Sensor Setup . . . 10

5.1 Euler Angles from MPU-9250 . . . 12
5.2 Euler Angles from BNO055 . . . 13
5.3 Joint Torques: CoppeliaSim . . . 13
5.4 Joint Torques: RBDL . . . 14
5.5 RBDL Torque Values at 500 Hz . . . 14
5.6 RBDL Torque Values at 30 Hz . . . 15
5.7 Quartic vs Cubic - Angles . . . 15
5.8 Quartic vs Cubic - Torques . . . 16
5.9 Dynamixel Motor: Torque Analysis . . . 16
5.10 Shoulder Pitch: Without Torque . . . 17
5.11 Shoulder Pitch: With Torque . . . 17
5.12 Shoulder Roll: Without Torque . . . 18
5.13 Shoulder Roll: With Torque . . . 18
5.14 Shoulder Yaw: Without Torque . . . 19
5.15 Shoulder Yaw: With Torque . . . 19
5.16 Elbow Pitch: Without Torque . . . 20
5.17 Elbow Pitch: With Torque . . . 20
5.18 Drawing a Circle: Manipulator . . . 21
5.19 Drawing a Circle: CoppeliaSim . . . 21
5.20 Pick and Place: Real Time . . . 22
5.21 Comparison between sent and received value time lags . . . 22
5.22 Pick . . . 23
5.23 Place . . . 23
5.24 Pick and Place: Human Joint Angles and Gripper State . . . 23

List of Abbreviations

DOF	Degree Of Freedom
AI	Artificial Intelligence
IMU	Inertial Measurement Unit
IR	Infrared
RGB	Red-Green-Blue
URDF	Universal Robotic Description Format
ROS	Robot Operating System
RL	Robotics Library
RBDL	Rigid Body Dynamics Library
RNEA	Recursive Newton-Euler Algorithm
ABA	Articulated Body Algorithm
CRBA	Composite Rigid Body Algorithm
SDK	Software Development Kit
MQTT	Message Queuing Telemetry Transport
PCB	Printed Circuit Board
JSON	JavaScript Object Notation
I2C	Inter Integrated Circuit
PID	Proportional Integral Derivative
QoS	Quality of Service
IDF	IoT Development Framework

Chapter 1

MOTIVATION

The recent years have brought an increasing interest in human-robot interaction. In the immediate future, robots will have to work among people, execute tasks that are assigned to them, and exhibit similar motions to their human counterparts for a friendly and predictable interaction. In this context, one of the most important aspects is motion planning.

Programming a robot to execute a prescribed movement is a very complex and tedious task as one has to establish the motor control sequences for every joint of the robot as well as their interactions while dealing with a large number of degrees of freedom simultaneously. Part of this burden can be eliminated by using intelligent approaches that allow a robot to imitate the movement of a human and learn from these demonstrations, ensuring a friendlier human-robot interaction and the predictability of the robot's movement.

The idea behind such an approach is to obtain information on the movements of a human using a motion capture system and replicate it on the robot within its limits and constraints. This is what our project aims to achieve i.e. to develop a manipulator capable of precise imitation of the human arm.

1.1 Technical Keywords

Robotic Manipulator, Human Imitation, Smart actuator system, Rigid body dynamics algorithms, RBDL, Forward and Inverse kinematics, Dynamixel, Dynaban, Closed-loop models, PID tuning, IMU, Sensor-fusion techniques, Complementary filter, Madgwick Filter, Simulation, CoppeliaSim.

Chapter 2

LITERATURE SURVEY

The imitation of the human arm by a robot manipulator will help remove the physical limitations of our working capacity e.g. to relocate a heavy object. Making the manipulator imitate would be much better than programming the arm to perform some task using some complicated algorithm. So, imitation is highly useful for multi-purpose robots who have to perform various tasks with which they have no prior experience. There has been prior research in the field.

First and foremost, it was necessary to understand the numerous methods tried out for human joint monitoring through a comparative study of the approaches based on their advantages, drawbacks and implementation challenges. The most commonly used methods viz. Imaging and Video-Based Tracking System and the Inertial Measurement Unit (IMU) Sensor-based tracking were chosen for further study, due to their several advantages over the rest.

The calculation of joint parameters using 3D depth images of the human arm employing an IR Transmitter-Receiver Pair and a 2D RGB camera has been implemented[2] . Also, there are previous works on calculation of the joint parameters using the readings of wearable sensors such as (1) inertial as accelerometers and gyroscopes, and (2) magnetic as magnetometers or (3) a combination of the previous sensors (IMU)[3][4].

The vision based-solution to track and estimate the human arm pose suffers a lot of drawbacks. Certain DOFs of the arm are difficult to track using a camera (For e.g. the shoulder roll) due to occlusions and fast motion. Also, external factors like inconsistent lighting conditions and nonoptimal camera position affect the precise estimation of the pose. Even though the IMU-based system was susceptible to noise and required complex sensor-fusion techniques, it proved to be best suited for our use cases due to compactness, low cost and ability to capture every DOF for a given human joint.

The stabilization of manipulators through dynamics is the most important part of the project. There has been tremendous research in the field of robot dynamics[5], centred around two methods namely, the Lagrangian method and the Newton-Euler method[6]. There are textbook references that help understand the concept of dynamics and help in the developmental stages of a manipulator. Some of the important ones like Fundamentals of Robotics[7] and Modern Robotics Mechanics, Planning, And Control[8] have been referred. The implementation details of the Newton-Euler algorithm has been discussed in the book Rigid Body Dynamics Algorithm by Roy Featherstone[9].

Based on the Newton-Euler recursive algorithm, the RBDL[10] (Rigid-body Dynamics Library) package will be used for the dynamic modeling of our manipulator. It is a self-contained free open-source software package that implements state-of-the-art dynamics algorithms including external contacts and collision impacts. It is based on Featherstone's spatial algebra notation and is implemented in C++ using highly efficient data structures.

Further, we investigated methods to interface dynamics and motor control, where Dynaban[11], an open-source alternative firmware for Dynamixel servo-motors, was found to be compatible with the manipulator and hardware we propose to use. By default, Dynamixel motors use a PID-only approach which always lag behind a moving command. In order to overcome this problem, the Dynaban firmware implements a generic model of the motor, more precisely -

1. A model of the electric motor (essentially the relationship between input voltage, rotation speed and output torque)
2. A model of the frictions (with an estimation of the static friction and the coulomb friction)
3. An inertial model

The project will be a part of AVITRA: The Humanoid[13]. The previous work on this system also forms an integral part of this project. Previous work based on imitation contains resources for control of dynamixel motors and extraction of angles using kinect sensors. The setup for imitation in this work is shown below. Although the project showed results, they were not satisfactory. Every step taken in this project will form an extension of the humanoid project and will focus on its improvement.

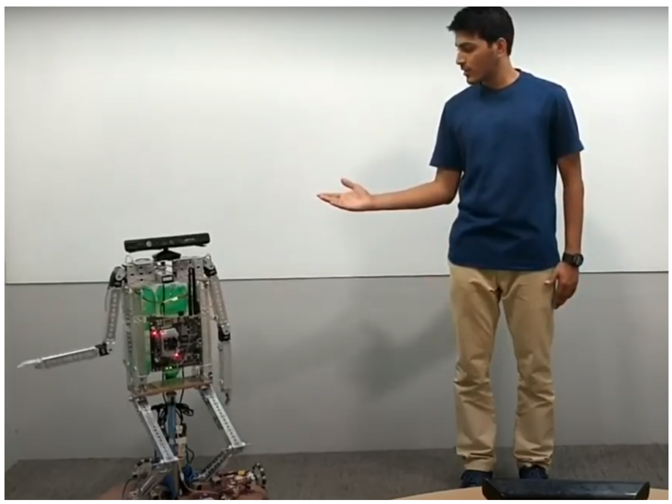

Figure 2.1: Imitation using Kinect sensor

2.1 Gap of Literature Survey

As is seen in previous sections, a lot of work has been done focusing merely on endpoint-based mapping and detecting posture using visual sensors. But these systems were unable to produce consistent results in different environments due to various factors. The idea proposed in this report will draw inferences from these preexisting solutions and introduce improvements for imitation using the IMU approach, which has been tested and implemented in research as well as industry.

2.2 Summary of Literature Survey

As a closing word, rather than implementing different complex algorithms on manipulator for multiple use cases, imitation is found to be the best and efficient method to use. This approach will ensure a friendlier human-robot interaction and high predictability of robot motion. We plan to create a system capable of executing our objective with high degrees of stability, precision, accuracy, and above all, consistency with different environments and configurations.

Hence we propose to use IMU sensors over vision based sensors for human arm pose tracking and implement dynamics using RBDL package and Dynaban firmware for the stabilisation of manipulator.

Chapter 3

PROJECT OBJECTIVES

1. **Acquiring the joint angles of human arm using IMU-based sensor capture system**
   - To identify the drawbacks of vision-based sensor capture systems and to overcome the same with IMU sensors
   - To extract joint angles with appropriate sensor fusion techniques and a decent sampling rate
2. **Publishing the joint angles in real time with low latency**
   - To publish the joint angles remotely over the Internet to the actual manipulator with minimum latency
3. **Perform imitation with inverse dynamics to achieve stable and accurate control of manipulator**
   - To perform imitation from the received joint angles
   - To compute inverse dynamics and implement Torque control, improving the overall accuracy and stability of the motion
4. **Comparative study of the dynamics approach (Torque control) and position control approach**
   - To compare results obtained after implementing torque control
   - To compare Dynaban firmware with the default firmware for Dynamixel motors

Chapter 4

PROPOSED SYSTEM AND IMPLEMENTATION

1. First, human joint angle data is extracted using various IMU sensors attached to the arm. The BNO055 sensor has been selected for this project due to its compactness, ease of use, and reliable output.
2. This sensor directly provides us with its absolute orientation, given by Euler angles (Roll, Pitch, and Yaw). These angles are then sent to the master micro-controller device which calculates the joint angles over the I2C protocol.
3. These angles are then packed in JSON format along with the corresponding timestamps and sent over MQTT to the remote manipulator.

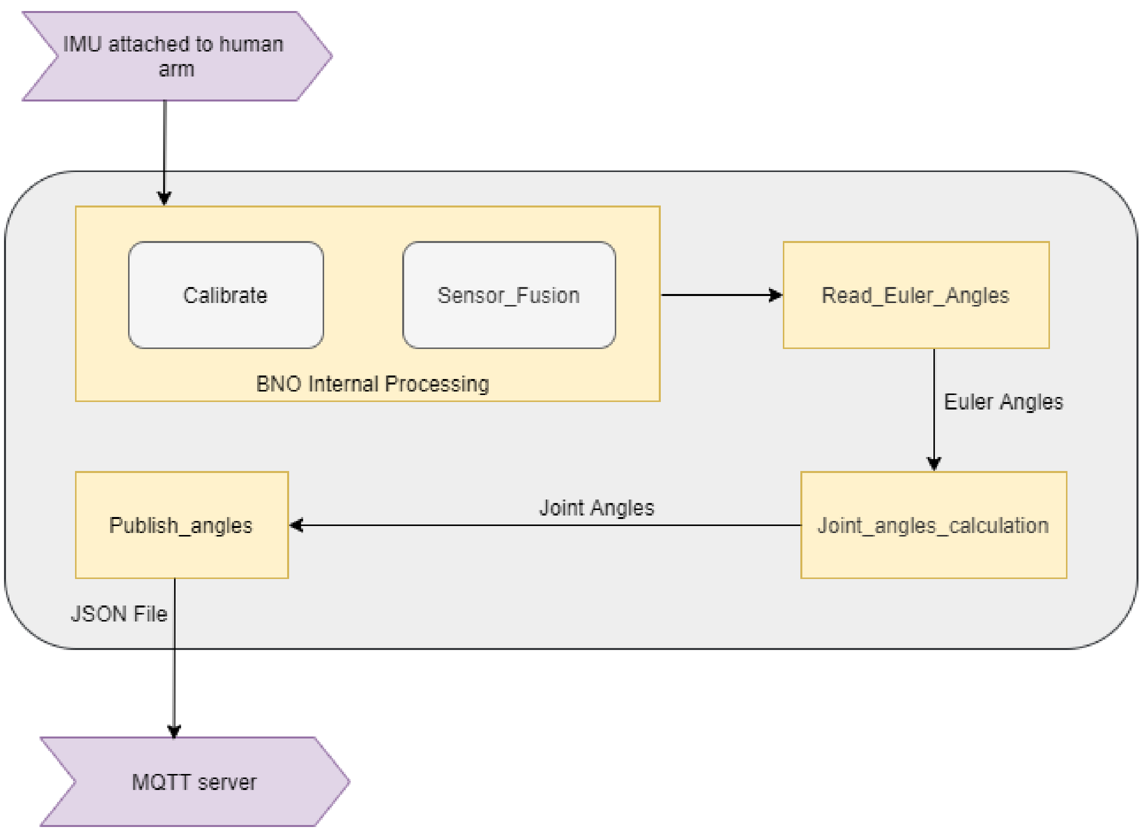

Figure 4.1: IMU Motion Capture: Process Flow

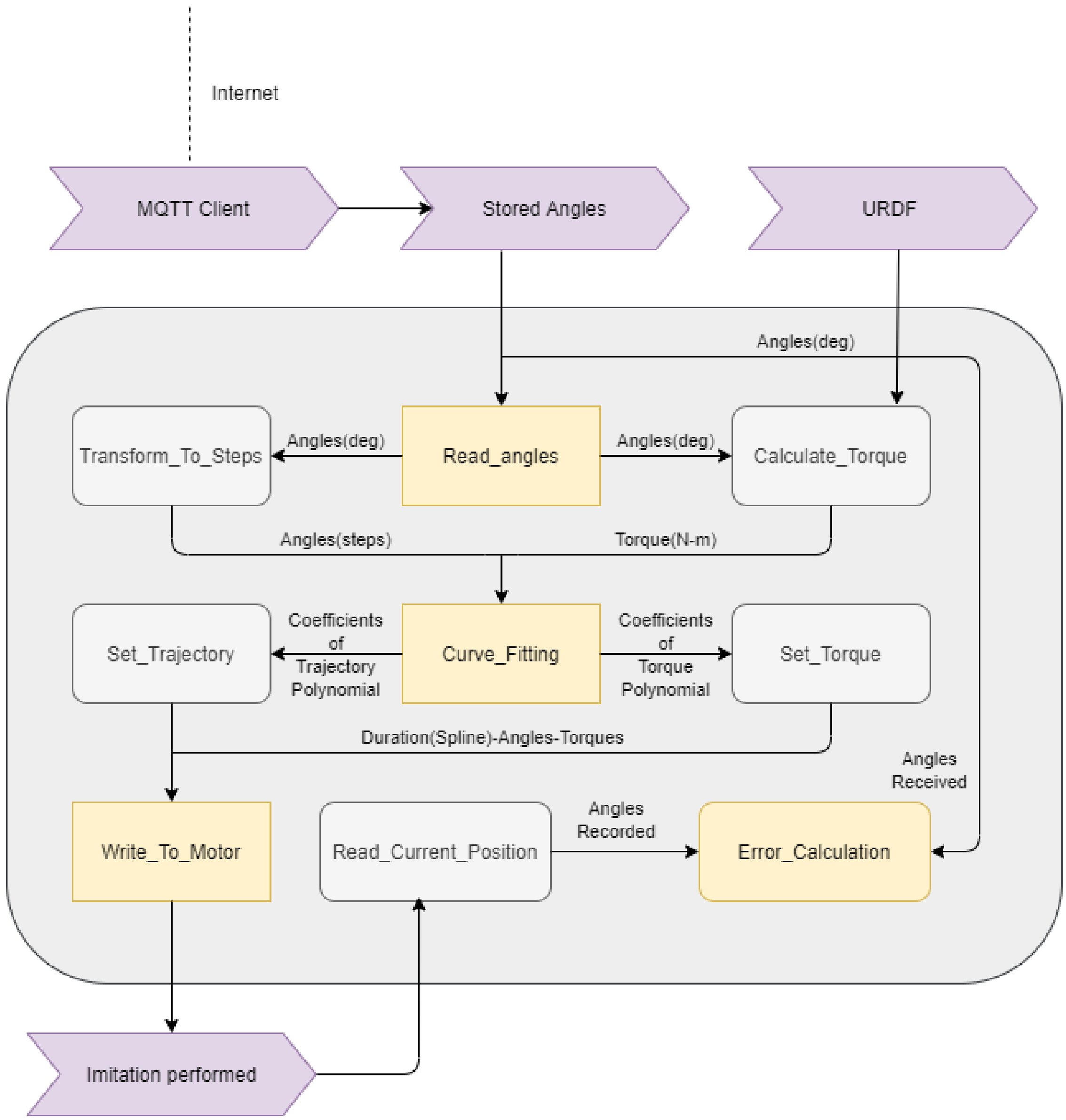

Figure 4.2: Manipulator Process: Flow

1. The other side of this project consists of the remote manipulator. An MQTT client is set up to receive the JSON data packets from the human side.
2. Joint angles (in degrees) and the timestamps along with the URDF of the manipulator are used to calculate the required joint torque. These angles are also converted to motor steps.
3. The calculated joint torques and the converted angles are then passed on to the curve fitting block to obtain coefficients of the trajectory polynomial and torque polynomial, further used by the motor to perform the required motion.
4. As the imitation is being performed, the angles of the motors and the torques are recorded for further analysis.

4.1 Hardware

4.1.1 Joint angle extraction

The ESP32 micro-controller is the master device, in charge of the sensor data extraction and communication with the remote manipulator. Multiple sensors communicate with ESP32 over the I2C protocol, connected over a bus multiplexer (TCA9548A). The above setup along with the power circuit and reverse voltage protection has been implemented on a PCB. Similarly, the BNO055 IMU requires breakout circuit boards for simplifying the interface with the master device. The sensor breakouts are attached on Velcro bands which provides a neat and hassle-free way for attaching, removing and re-positioning sensors on a human arm.

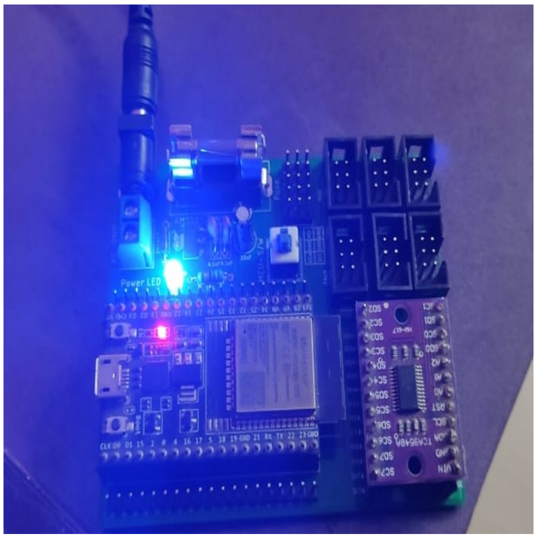

Figure 4.3: Master PCB

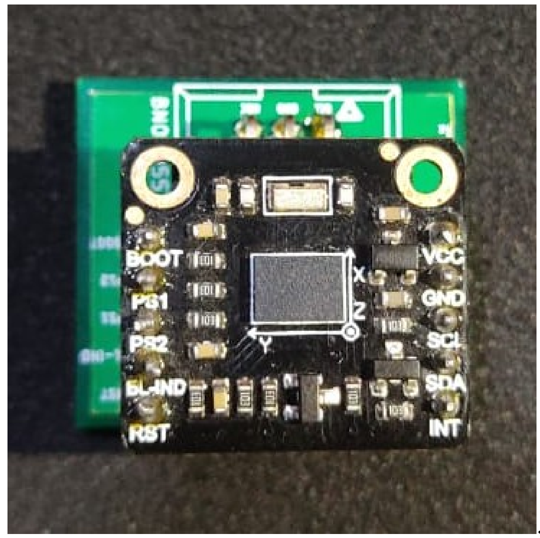

Figure 4.4: BNO055 Breakout

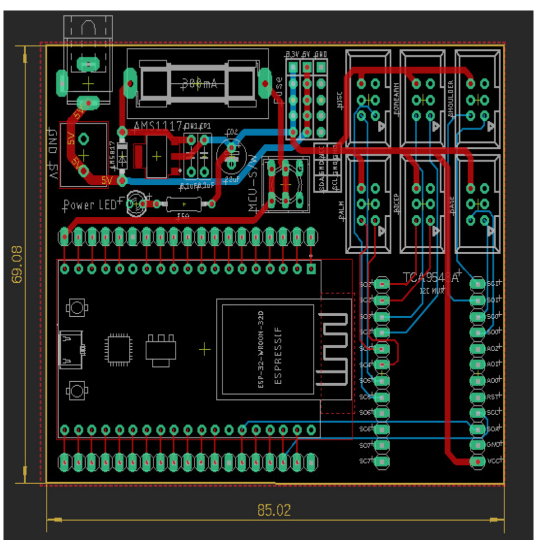

Figure 4.5: Master PCB Schematic

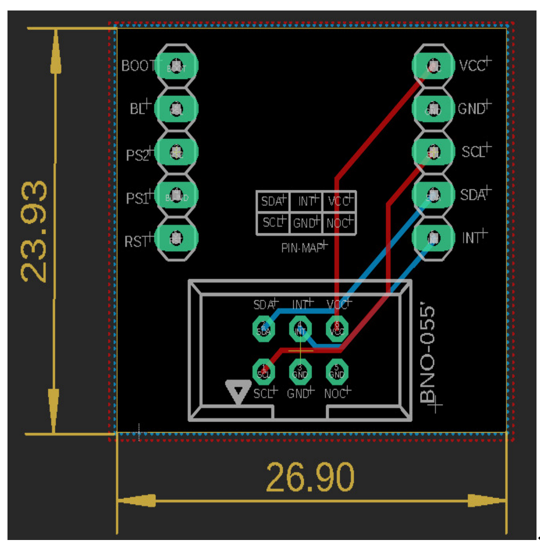

Figure 4.6: BNO055 Schematic

4.1.2 Manipulator

The project requires an autonomous, anatomically similar but scaled-down version of the human arm. The manipulator consists of four servo motors (MX-64) as joints and one motor(AX-12A) for gripper actuation; all of them are set up in a daisy-chain fashion. The manipulator possesses 4 DOFs out of which, 3 DOFs are on the shoulder (pitch, roll and yaw) and 1 DOF on the elbow (pitch). A gripper is attached at the end. The entire setup is fixed on a stand as shown below. A URDF file of the manipulator was also made for simulation and analysis purposes. The gripper still in use is the design of underactuated two-fingered rack and pinion based.

Figure 4.7: Manipulator: Front View

Figure 4.8: Manipulator: Side View

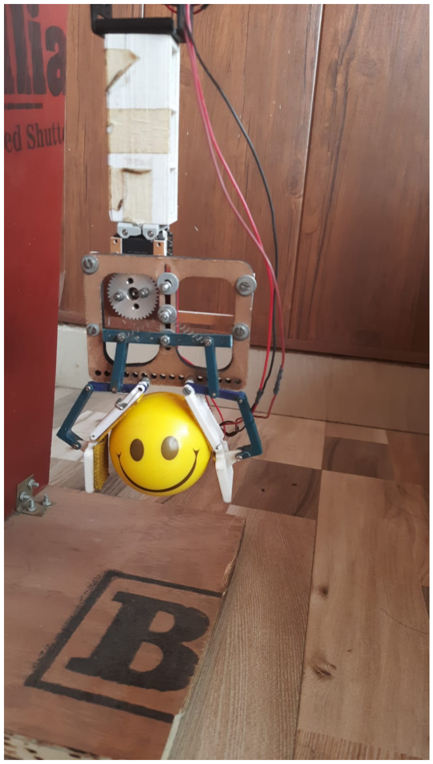

Figure 4.9: Gripper

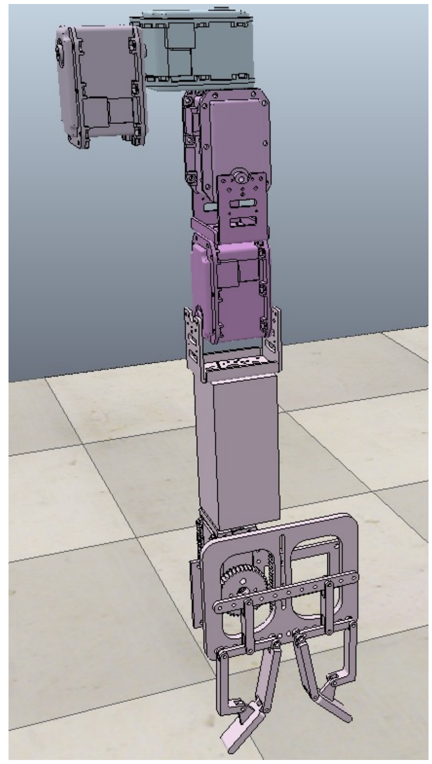

Figure 4.10: URDF

4.2 Implementation

4.2.1 Joint Angle Extraction

1. The human arm joint parameters will be obtained using a network of sensor bands embedded with 9-axis IMU sensors placed on the human arm as shown below[2]. We selected the Bosch BNO055 for this purpose due to its compactness and ease of use.

2. The BNO055 directly provides us with reliable Euler angles and Quaternion data, owing to its on-board micro-controller at a 100Hz sampling rate. It has a simpler calibration routine than comparable solutions.

3. The ESP32 micro-controller is the master device, in charge of the sensor data extraction and communication with the remote manipulator. The multiple sensors communicate with ESP32 over the I2C protocol, connected over a bus multiplexer (TCA9548A).

4. After attaching the sensors to the human arm in desired orientation, the joint angle was estimated in two steps - one, obtain the absolute orientation of each IMU sensor separately and then take the difference between the absolute orientations (on the links adjacent to the joint) in order to estimate the joint angle involved. E.g. Shoulder pitch can be obtained as the difference between the base (chest) IMU pitch and the bicep IMU pitch.

5. These angles are then packed in JSON format along with the corresponding timestamps by ESP32 and sent over MQTT to the remote manipulator. An on-board switch (ESP32) is responsible for controlling the gripper state on the manipulator.

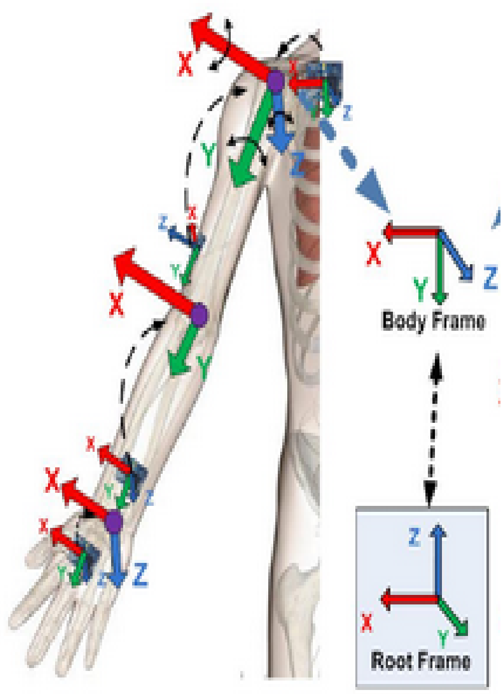

Figure 4.11: Reference Sensor Placement

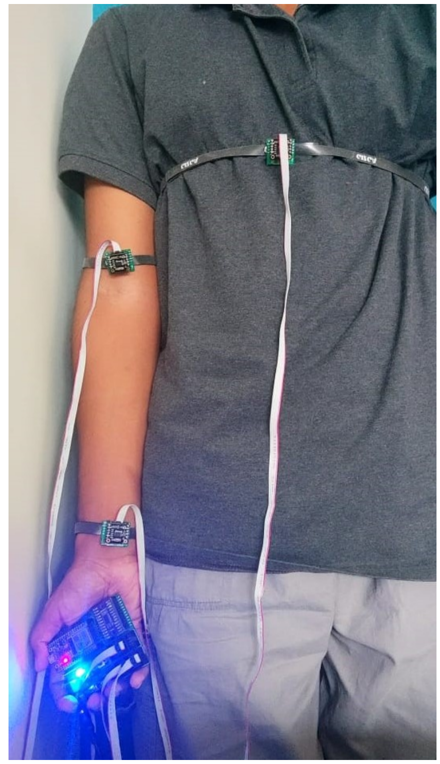

Figure 4.12: IMU Sensor Setup

4.2.2 Manipulator Simulation

1. **Imitating the human motion on CoppeliaSim:**
   (a) The angle values extracted from the human arm are given to the simulated manipulator through ROS.
   (b) This method helps verification of the angles obtained from IMU sensors before imitating those on the actual manipulator.
2. **Joint angle extraction using CoppeliaSim:**
   (a) Any desired motion can be performed using the simulated manipulator and the angle values can be obtained for all joints.
   (b) These angles can be given to actual manipulator for imitation thereby eliminating the need of actual sensors for joint angle extraction.

4.2.3 Manipulator Control

1. A communication pipeline has been established with MQTT to receive joint angles in real time over the internet. These values can be used directly for real time control, or can be stored for motion analysis.

2. The RBDL library uses the manipulator URDF as input and processes the received human joint angles and timestamps to obtain corresponding joint torques.

3. The human joint angles received go through a pre-processing stage which involves passing the data through a moving average filter, multiplying it with a transformation matrix that corrects any orientation differences between human and manipulator joints and then finally converting the joint angles from degrees(0-360) to motor steps(0-4096).

4. The joint angle and torque values are now split at equal intervals (500-1000 ms) and undergo a curve fitting process of degree 3. The polynomial coefficients for trajectory and torque are used to control the motors.

5. The Pypot library that has been extended for the Dynaban firmware will send the trajectory and torque polynomial coefficients. Then the manipulator will be set to PID and predictive control mode which is a feature unique to Dynaban.

6. The PID was tuned manually to try and achieve as stable and accurate motion as possible. During manipulator movement, the angle values were recorded from the manipulator. The two sets of values are then compared using normalised dot product analysis, as a metric of similarity given by the formula

$$\text{Similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2}\sqrt{\sum\limits_{i=1}^{n} B_i^2}} \tag{4.1}$$

Chapter 5

RESULTS

5.1 Joint angle extraction using IMU

In the first iteration, joint angles were extracted using the MPU-9250. As shown below, the results obtained were good, as there was no drift or influence of noise on the pitch, roll or yaw values, even at high data rates. We achieved data rates of 1000Hz for the pitch and roll angles and 100Hz for the yaw angle. But, the MPU-9250 suffered from inter-angle dependence - a major issue for our case. The yaw varied with the pitch and the roll angles as shown in the table below, measured by the Pearson's correlation coefficient (takes in the covariance of X and Y and their standard deviations) given as -

$$\rho_{X,Y} = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y} \tag{5.1}$$

Also, the MPU-9250 took almost 9-10 seconds to stabilise itself at the initial orientation. The same was then repeated with the Bosch BNO055 sensor. Joint angles were extracted at a lower sampling rate (100 Hz) but almost no inter-angle dependence was found. BNO055 also has a simpler calibration routine than its counterpart. Thus, it was decided to proceed with this sensor.

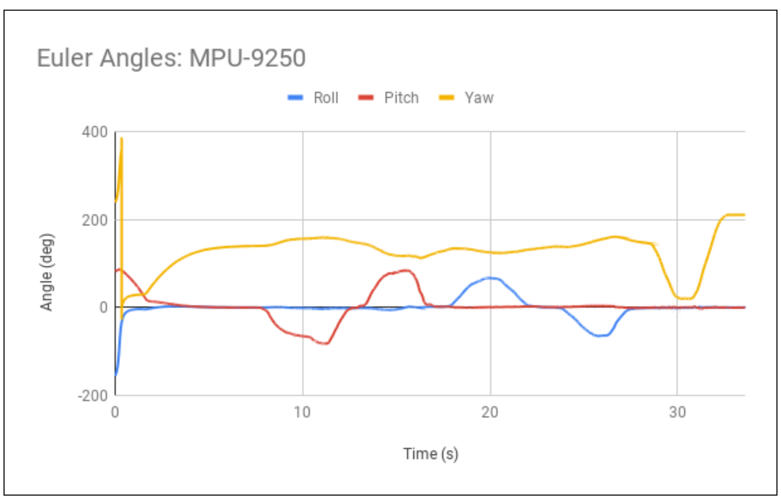

Figure 5.1: Euler Angles from MPU-9250

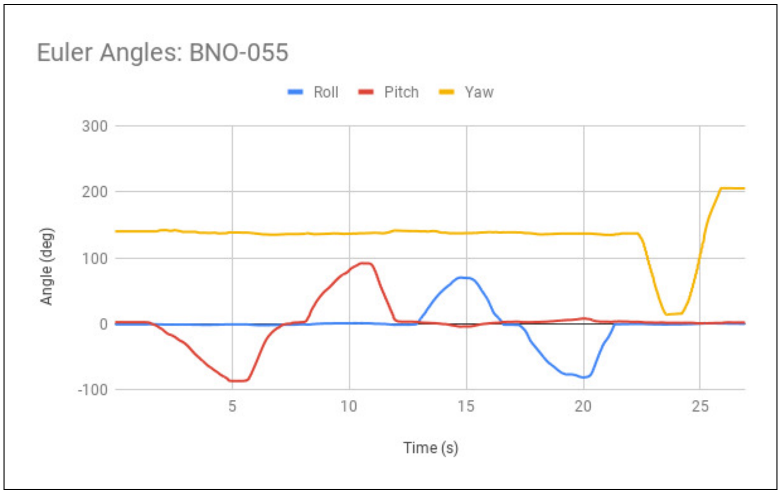

Figure 5.2: Euler Angles from BNO055

Sensor	Pitch v Yaw	Roll v Yaw
MPU-9250	-0.2568	-0.2102
BNO055	-0.0116	0.0066

Table 5.1: Inter-angle dependence: Pearson's Correlation Coefficient

5.2 Comparing joint torques: RBDL v CoppeliaSim

The torque values obtained from manipulator during simulation has been plotted alongside the torque values calculated from the RBDL library for the same motion. We observe that the graphs are equal in magnitude but inverted. This is because CoppeliaSim provides torque values in a positive range only, and the direction of the torque is reflected in the velocity value in simulation. The similarity in the graph verifies that in the absence of any external torque, RBDL performs inverse dynamics accurately. A comparative graph of torques for both cases are shown below.

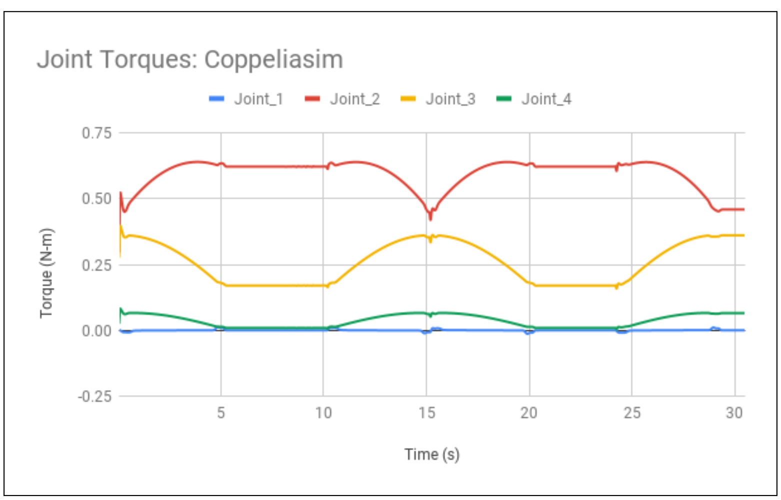

Figure 5.3: Joint Torques: CoppeliaSim

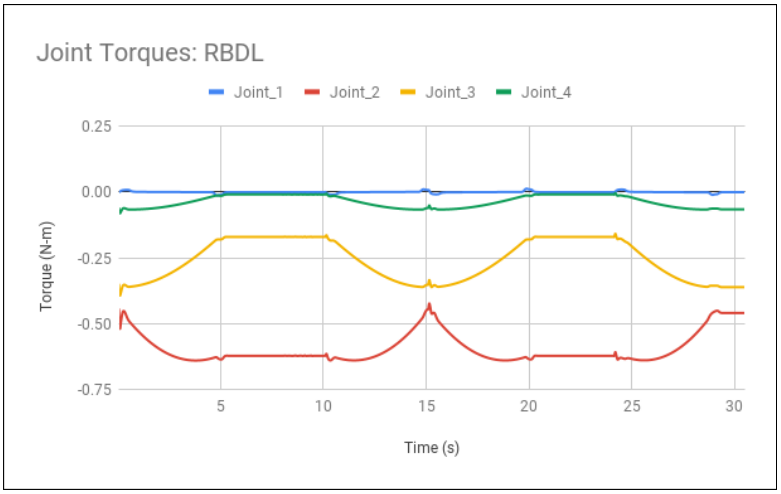

Figure 5.4: Joint Torques: RBDL

5.3 Results from RBDL under different sampling rates

The RBDL library exhibits different behaviour based on different sampling rates used during angle extraction. The library uses the difference in timestamps to calculate velocity and acceleration. As the sampling frequency increases, the magnitude of the resulting torque increases to the point where it is no longer practically possible to implement. Using lower sampling rates (around 30Hz) provides stable results. The torque values for 500Hz and 30Hz sampling rates as obtained from RBDL are given below. The torques read back from the manipulator as a result are also plotted. The graph provides a clear indication that angles obtained at a lower sampling rate provide a better result.

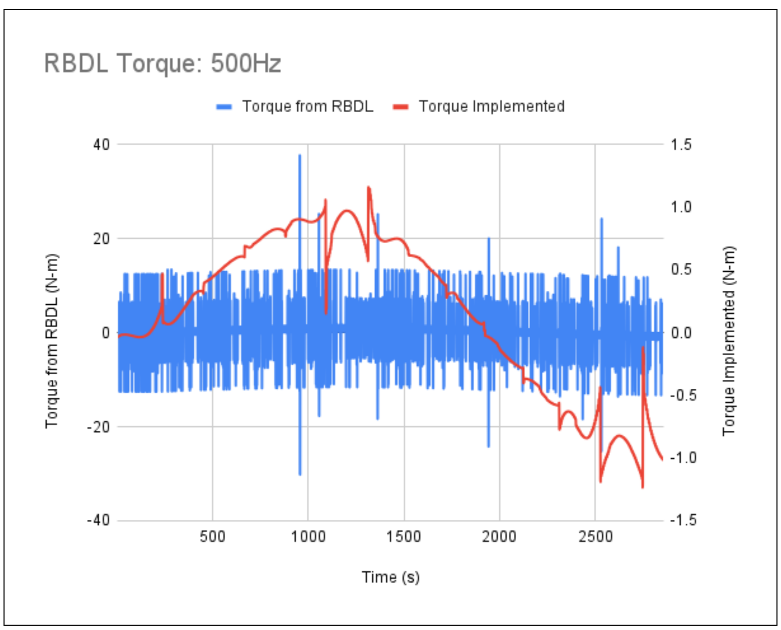

Figure 5.5: RBDL Torque Values at 500 Hz

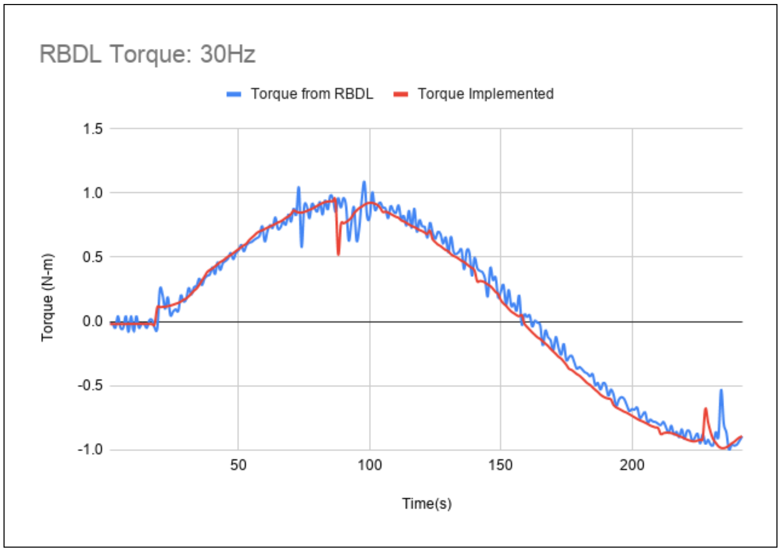

Figure 5.6: RBDL Torque Values at 30 Hz

5.4 Comparison between quartic and cubic curve fit for Dynaban

Dynaban requires trajectory and torque in a polynomial form to execute motion in predictive control mode and thus the joint angles and RBDL torque values had to go through a curve fit process. Trials were made with quartic as well as cubic curve fit for many iterations. The graph below shows one such instance of the required human joint angles along with the polynomials obtained using quartic and cubic curve fitting. The normalised dot product similarity values provide a clear indication that cubic curve fit is better for our purpose. A similar comparison has been provided for torque values.

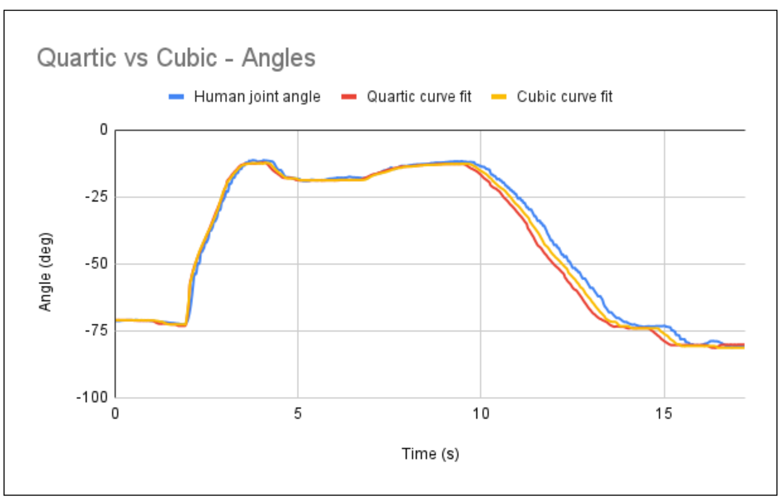

Figure 5.7: Quartic vs Cubic - Angles

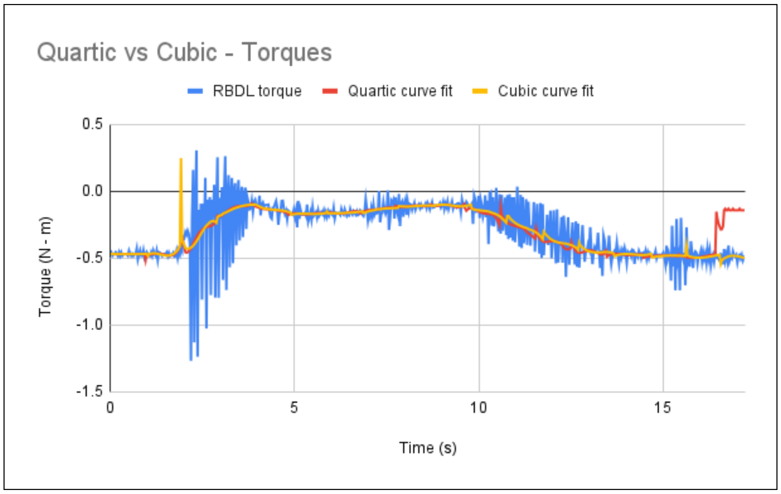

Figure 5.8: Quartic vs Cubic - Torques

Normalised Dot Product	**Quartic Curve Fit**	**Cubic Curve Fit**
Angles	99.7922%	99.9167%
Torques	90.4114%	92.3806%

Table 5.2: Similarity for Quartic and Cubic curve fits: Percentage Normalised Dot Product

5.5 Dynamixel Motor Torque Analysis

During only position control, when a certain trajectory is given to a Dynamixel motor, it applies a certain torque value for execution. On plotting these values it was found that these torque values majorly consisted of zeroes and some spikes. On the contrary, RBDL provided values with higher magnitude throughout the motion. After implementing torque control, the values recorded were in accordance with the values from RBDL library.

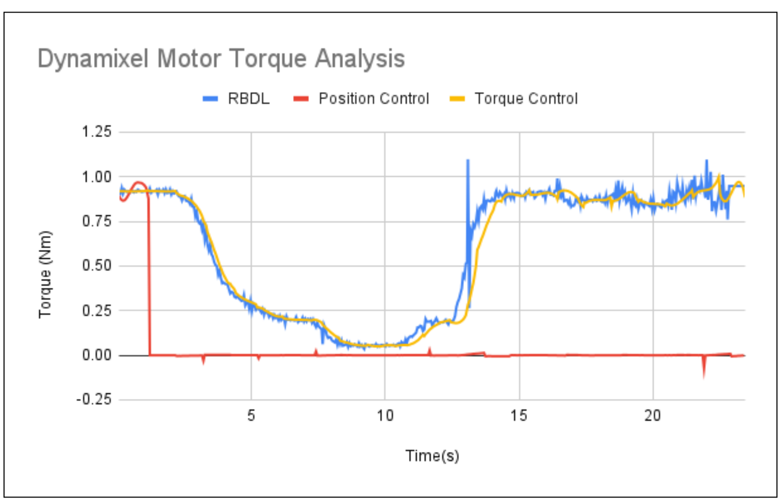

Figure 5.9: Dynamixel Motor: Torque Analysis

5.6 Comparison between position control and torque control imitation

During the process of imitation, the desired and obtained values of position have been plotted for each motor of the hardware manipulator as shown in the graphs below. There are two sets of the graph shown below - the first set showing the imitation comparison when only position control approach was used and the second showing the torque control approach. A table of results is also shown below with the normalised dot product for both with and without torque control, as a metric of similarity.

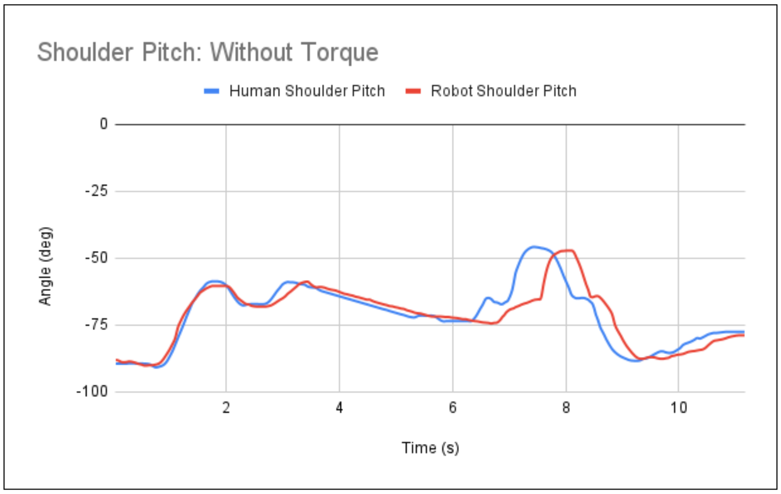

Figure 5.10: Shoulder Pitch: Without Torque

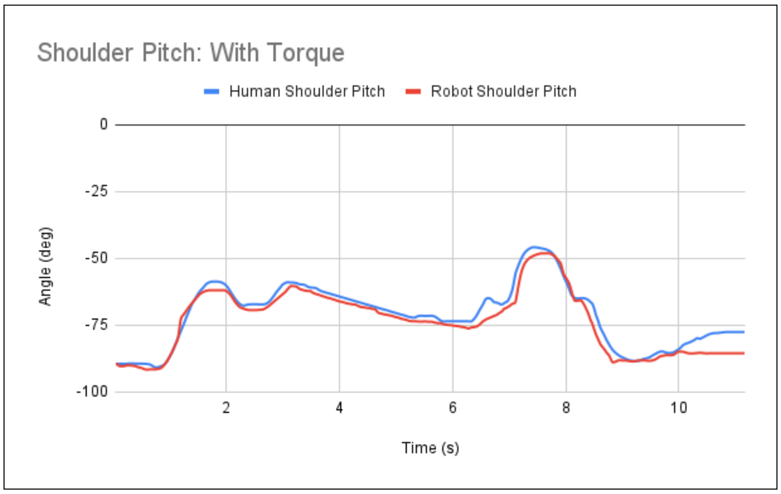

Figure 5.11: Shoulder Pitch: With Torque

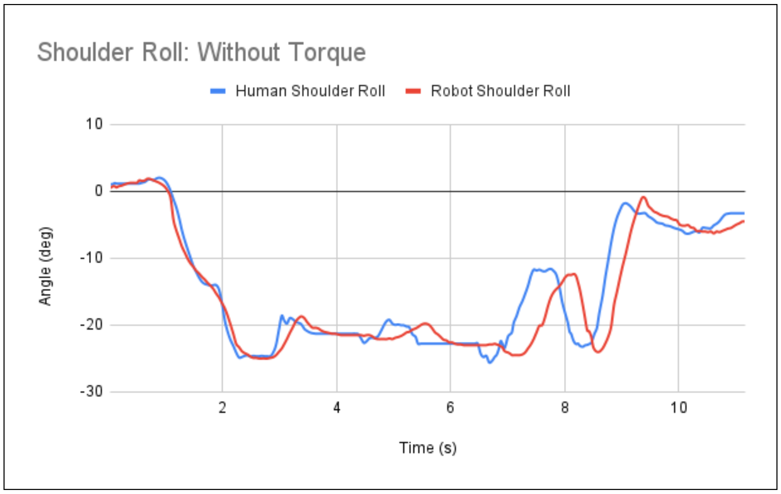

Figure 5.12: Shoulder Roll: Without Torque

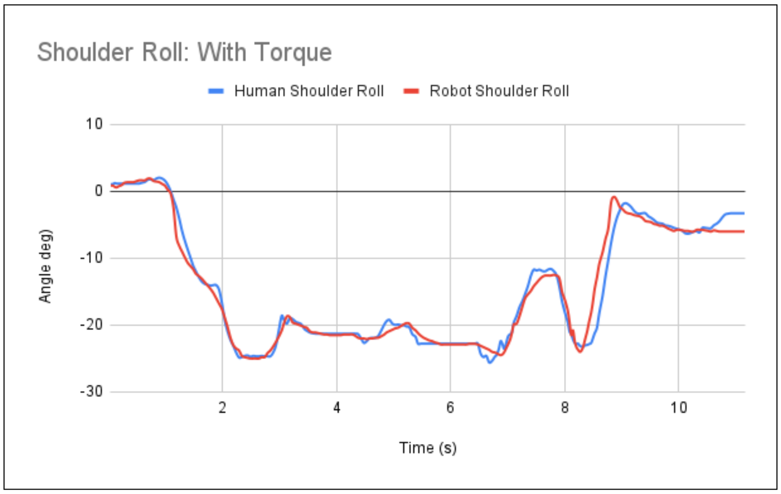

Figure 5.13: Shoulder Roll: With Torque

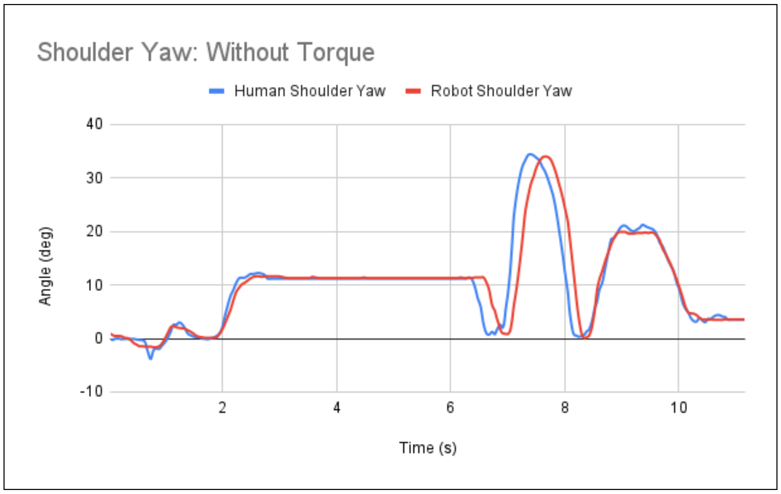

Figure 5.14: Shoulder Yaw: Without Torque

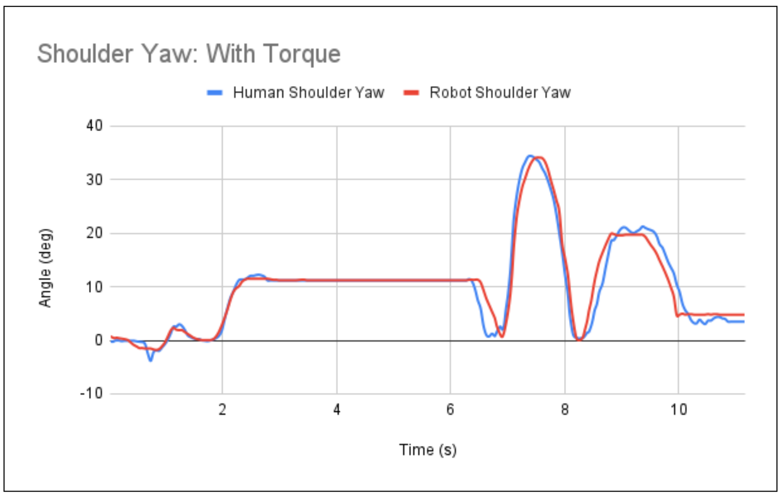

Figure 5.15: Shoulder Yaw: With Torque

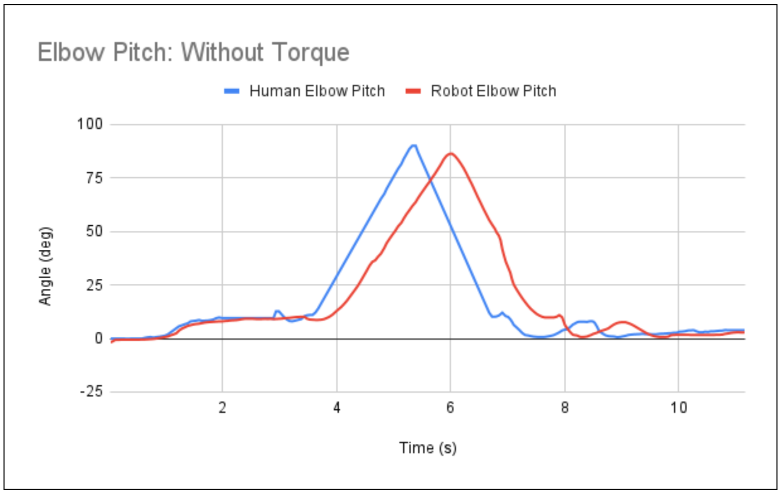

Figure 5.16: Elbow Pitch: Without Torque

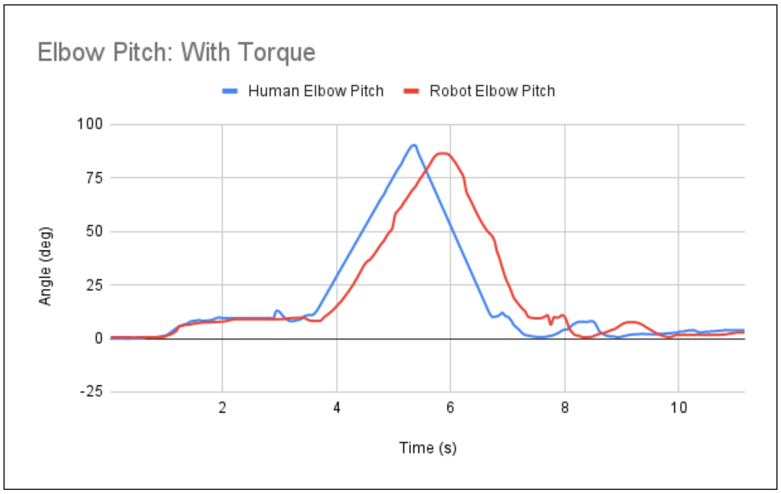

Figure 5.17: Elbow Pitch: With Torque

Normalised Dot Product	**Without Torque**	**With Torque**
Shoulder Pitch	99.6474%	99.9452%
Shoulder Roll	97.7874%	99.5361%
Shoulder Yaw	95.3855%	98.7544%
Elbow Pitch	86.0183%	90.3805%

Table 5.3: Similarity For Position and Torque control: Percentage Normalised Dot Product

As seen from the above results, it is evident that the torque control approach provides better accuracy and thus, smoother imitation.

5.7 Drawing a Circle

A circle was drawn on simulation (CoppeliaSim) using two motors on the shoulder - pitch and yaw. The angles were recorded from the simulated manipulator and then fed to its hardware counterpart. A marker was attached at the end effector and the motion was imitated first in the air and then on a board. Two cases were analysed - with and without torque control.

Figure 5.18: Drawing a Circle: Manipulator

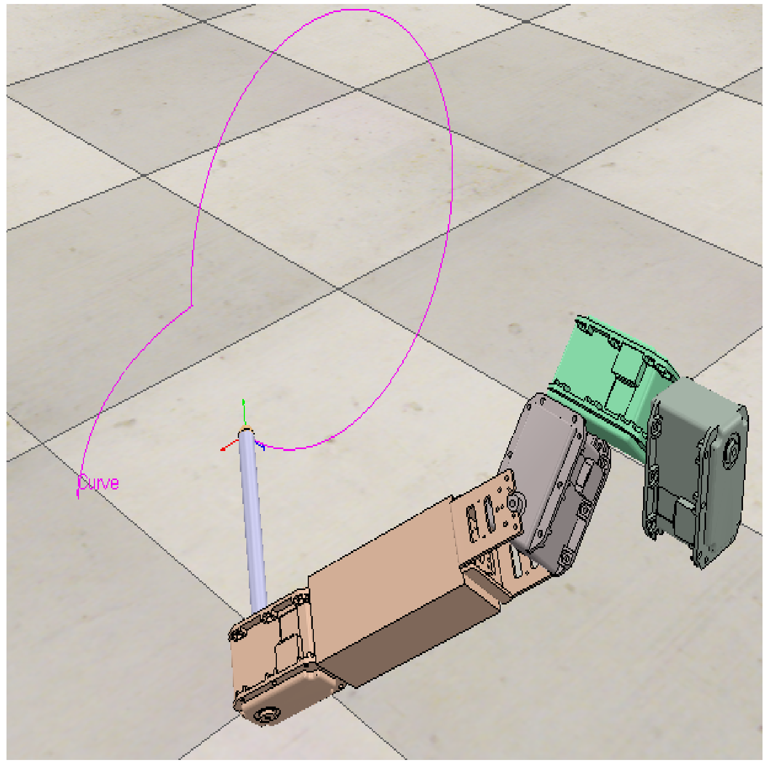

Figure 5.19: Drawing a Circle: CoppeliaSim

5.8 Pick and Place: Real Time

Using various results provided in the previous sections, manipulator control using IMU sensor setup was performed in real time. The angles recorded from BNO055 sensors were sent over MQTT and fed to the manipulator after reception. After analysing the results, it was observed that the processing time is small, but the MQTT server produced lag after a certain time interval (delay of ~ 2 seconds). The entire pick and place operation has been executed successfully and the same has been provided in a video demonstration.[32] The delay in real-time imitation for slow human arm movement (1000 steps/s or 87.89 deg/s) is found to be minimum at 2 sec. But, if the movement of the human arm is faster than the specified limit, the angles need to be written with some delay so as to to meet the limitations of Dynamixel motor and the current setup of the manipulator. Thus, the overall delay in real-time imitation becomes quite large at 20s. The graph below shows time delay between sent and received values through MQTT.

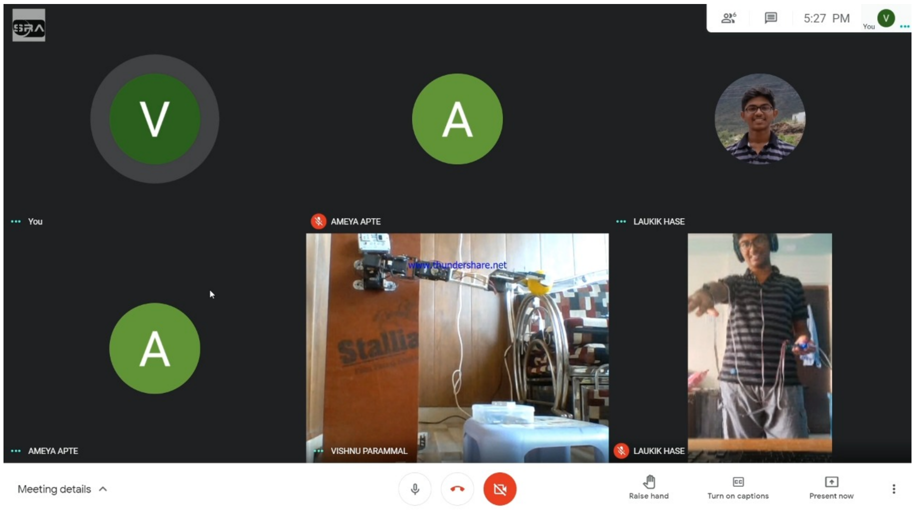

Figure 5.20: Pick and Place: Real Time

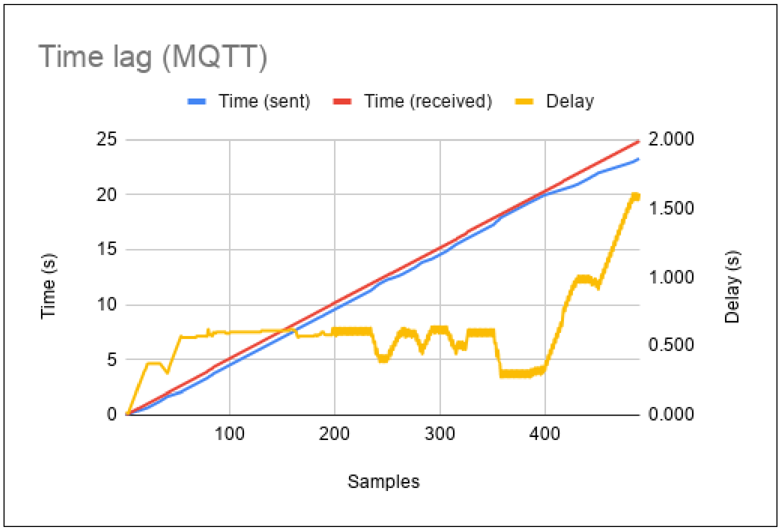

Figure 5.21: Comparison between sent and received value time lags

5.9 Memorise and Repeat

The angles obtained during pick and place motion were stored and the same could be used again to repeat the motion. Also in practical scenarios, there is always a scope for error, so the stored data for a corresponding motion can be processed to improve the precision of the motion and thereby minimizing the error. The graph at the end shows the motion performed by the human. The black curve shows the ON and OFF state of the gripper.

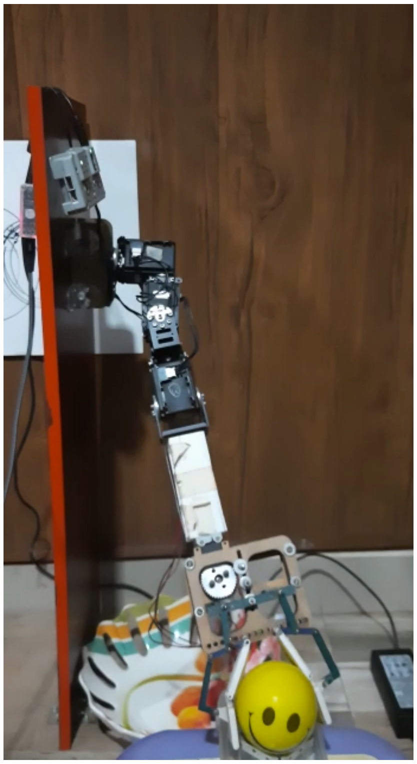

Figure 5.22: Pick

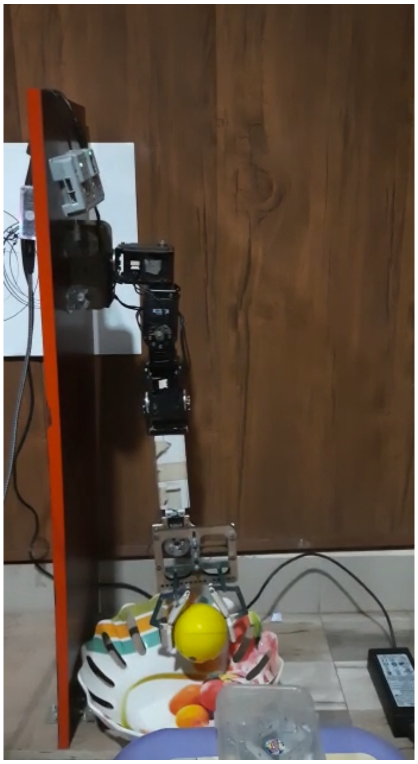

Figure 5.23: Place

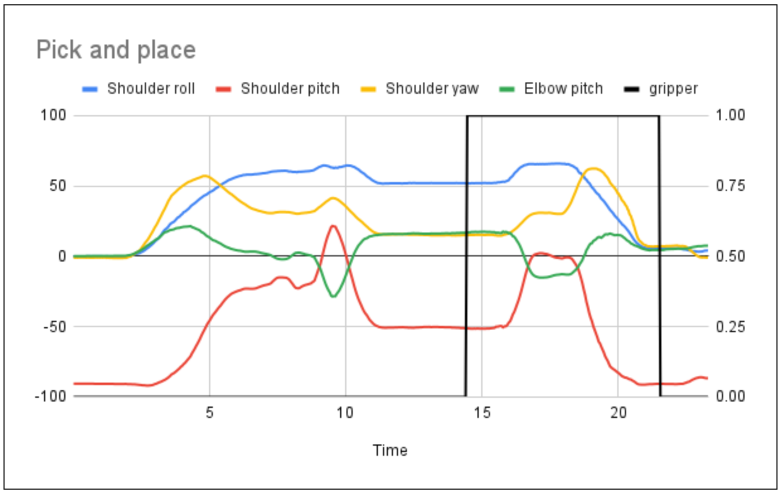

Figure 5.24: Pick and Place: Human Joint Angles and Gripper State

Chapter 6

PLATFORM USED

6.1 Hardware

6.1.1 IMU

Component	**Physical Quantity**	**Function**	**Output**
3-axis Gyroscope	Angular Velocity	Detect rotational changes	Pitch, Roll
3-axis Accelerometer	Linear Acceleration	Detect the rate of change in velocity	Pitch, Roll
3-axis Magnetometer	Magnetic Field	Detect the strength and direction of magnetic field	Yaw

Table 6.1: 9 axis IMU

The requirement for acquiring the joint parameters is that either the sensor directly provides the angles on which it is placed or it provides some data which can directly be mapped to the joint parameters. The 9-axis IMU provides us with the latter option, consisting of the 3 components as shown in the table above. To measure the relative orientation, these components are geometrically positioned to provide X, Y and Z coordinate-based measurements. The three angles, Pitch, Roll and Yaw namely, will then be used to calculate the joint parameters required for imitation. The human arm consists of three main joints viz. the shoulder joint, the elbow joint and the wrist joint. So, to calculate all the joint parameters we intend to use one 9-axis IMU on the shoulder, as the shoulder joint has 3-DOFs and one 6-axis IMU (does not contain magnetometer) on the elbow and wrist each as both of these joints have 1-DOF. Although the wrist has 3-DOFs we intend to use only the pitch motion as our manipulator has only one minor DOF.

BNO055

BNO055 is an IMU sensor from Bosch. It hosts an accelerometer, magnetometer and gyroscope with an on-board microcontroller for ready-to-use sensor fusion, providing absolute orientation in space. It uses a triple-axis sensors to measure tangential and rotational acceleration and strength of magnetic field.The sensor provides quick calibration routine. [21]

6.1.2 ESP32 DevKitC

ESP32 is a MCU with inbuilt Wi-Fi and Bluetooth modules from Espressif Systems. Robust design, ultra-low power consumption, high level of integration and rich peripherals are some of its features.[32]

6.1.3 Dynamixel

The Dynamixel is a joint actuator that is designed to be modular. They can be daisy-chained on any robot or mechanical design for powerful and flexible robotic movements. We are using the Dynamixel MX-64 and Dynamixel AX-12A servos for this project. Dynamixel SDK is a software development kit that provides Dynamixel control functions using packet communication. The Dynamixel SDK supports both C++ and python which are used for the development of this project. The dynamixel motors can be tested and controlled using its official software Dynamixel Wizard 2.0. [29][30]

6.2 Software

6.2.1 ESP-IDF

ESP-IDF provides SDK for general application development. It uses C, C++ as the programming languages. It is Espressif's open source official IoT Development Framework. Being open-source and production- ready, feature-rich software components and extensive documentation and examples are some of its features. [31]

6.2.2 MQTT

MQTT stands for Message Queuing Telemetry Transport. It is a communication protocol used by IoT devices. It is lightweight and hence ideal for remote devices. We have used MQTT for real time transmission of joint angle data over the internet. [23]

6.2.3 ROS

Robot Operating System is a middleware that provides functionalities like hardware abstraction, communication between different programming languages, and many commonly used features. ROS has been used as a communication medium between simulator and programs in our project. [24]

6.2.4 CoppeliaSim

To test the correctness of various algorithms used throughout the project, the CoppeliaSim simulation software will be used. It uses Lua language but also supports C++ and python as external APIs. It acts as a testbed for our project. It is also compatible with ROS which will be useful for testing various codes, libraries and packages that will be developed during the project. It also has a large library of existing models, which will be helpful for our project. [25]

6.2.5 RBDL

RBDL is a C++ library that helps in implementing the dynamics algorithm. The same will be used and tweaked as per our purpose for the project. RBDL library is written in C++. It contains dynamics algorithms like Articulated Body Algorithm (ABA), Recursive Newton-Euler Algorithm (RNEA)and Composite Rigid Body Algorithm (CRBA). ABA is used for forward dynamics, RNEA is used for inverse dynamics and CRBA is used to calculate joint space inertia matrix. It contains code for forward and inverse kinematics, jacobians, closed-loop models and handling of external constraints.[26]

6.2.6 Pypot

Pypot is a python library for controlling dynamixel motors. It provides control at a low level, for directly communicating with the motor registers, as well as a high-level abstraction that provides an easy interface to control robots. Pypot is also compatible with the CoppeliaSim simulator. It has been modified to suit the needs of dynaban firmware in the pypot dynaban edition library[27]

6.2.7 Dynaban

Dynaban is an open-source firmware for the Dynamixel MX-64 motor. It has the software features to fully use the hardware capabilities of the device. To improve the default controller, a friction model and an electric model of the motor are also included in it. It implements a feed-forward method to follow position, speed and torque trajectories. This approach is useful for the highly dynamic movement of robots whose torque trajectories are computed using classic rigid body inverse dynamics. The comparison between the default control strategy and the one implemented by Dynaban shows significant improvements in terms of accuracy, delay and repeatability. [28]

Chapter 7

CONCLUSION & FUTURE SCOPE

Control of manipulators through imitation is a new approach that has its pros and cons. Making the manipulator imitate a host human arm is much better than programming it to perform a particular task, but the control and stability requirements of the system are high.

This project, aiming to achieve them, has three important aspects. The first aspect is the data acquisition for capturing the joint parameters of the human arm required for imitation. After testing with both MPU-9250 and BNO055 sensors, the latter gave better results in terms of accuracy as well as calibration time. The second aspect is publishing these values over internet in real-time to the remote manipulator. This was achieved by using MQTT protocol which provided high reliability and low latency. The third aspect is achieving seamless motion and exact imitation wherein stability, accuracy, and precision are expected the most out of the manipulator. The concept of inverse dynamics comes into the picture here, providing precise and stable movement by using torque control along with position control. The results obtained using predictive torque control demonstrated better results when compared with only PID based position control. At present, along with the above-mentioned tasks we have also added a gripper at the end effector which can perform pick and place. A stationary marker was also added as an end effector to draw simple shapes and figures.

This project has an extensive scope of development and has further potential to be a long-term research project. One such example can be the implementation of haptics to obtain feedback from the manipulator giving us boundless applications and enhance the versatility of the manipulator. Other applications may include increasing the DOF of the manipulator to achieve delicate and precise imitation. The present sensor system can be improved by getting rid of wires entirely by making a custom PCB circuit with the ability to transmit data wirelessly and on-board power. By further analyzing motors, an indigenous custom firmware like Dynaban for other types of Dynamixel servos like AX-12 for better torque control can be implemented. Also, there is a vast scope in improving imitation with the advent of AI and ML. Thus, this project not only is a valuable asset for a robotic enthusiast but also encompasses multiple fields of application, proving it to be useful in several walks of life.

Bibliography

[1] Faisal, A.I.; Majumder, S.; Mondal, T.; Cowan, D.; Naseh, S.; Deen, M.J. Monitoring Methods of Human Body Joints: State-of-the-Art and Research Challenges. Sensors 2019, 19, 2629

[2] Varshney, Ayush et al. "Manipulator Control Using Human Arm Imitation." IFAC Proceedings Volumes 47 (2014): 109-114

[3] I. Prayudi and D. Kim, "Design and implementation of IMU-based human arm motion capture system," 2012 IEEE International Conference on Mechatronics and Automation, Chengdu, 2012, pp. 670-675

[4] Estimation of Angles from Upper and Lower Limbs for Recognition of Human Gestures using Wireless Inertial Sensors Irvin Hussein Lopez Nava, Angelica Munoz-Melendez, Technical Report No. CCC-15-004 December, 2015

[5] Armstrong, W. W. 1979. Recursive solution to the equations of motion of an n-link manipulator. Proc. 5th World Congress Theory Mach. Mechanisms, vol. 2. Montreal: ASME, pp. 1343-1346

[6] Ahmet SHALA, Ramë LIKAJ, Dynamic Modeling Of 3 Dof Robot Manipulator, ACTA TECHNICA CORVINIENSIS – Bulletin of Engineering, Tome XI, 2018

[7] R.J. Schilling, Fundamentals of Robotics Analysis and Control Prentice-Hall, Englewood Cliffs, NJ, 1990

[8] Kevin M. Lynch and Frank C. Park, Modern Robotics Mechanics, Planning, And Control, Cambridge University Press, 2017

[9] Featherstone R. Rigid Body Dynamics Algorithms In: In: Featherstone R, editor. 1st. New York: Springer: 2008

[10] Felis, M.L. RBDL: an efficient rigid-body dynamics library using recursive algorithms. Auton Robot 41, 495–511 (2017).

[11] Fabre, Remi & Rouxel, Quentin & Passault, Grégoire & N'Guyen, Steve & Ly, Olivier. (2017). Dynaban, an Open-Source Alternative Firmware for Dynamixel Servo-Motors. 169-177.

[12] Survey of imitation learning for robotic manipulation by Bin Fang1, Shidong Jia2, Di Guo1, Muhua Xu2, Shuhuan Wen3, Fuchun Sun1

[13] Gawali, Kothari, Agashe, Kumaran, Kuthe. Avitra: Manipulator, Veermata Jijabai Technological Institute, 2020

[14] A Comparison of Complementary and Kalman Filtering by WALTER T. HIGGINS, JR. (IEEE Thans. Aerospace and Electronic Systems, Vol. AES-li, no. 3, May 1975.)

[15] Chen, Xi, "Human Motion Analysis with Wearable Inertial Sensors. " PhD diss., University of Tennessee, 2013

[16] Wisama Khalil. DYNAMIC MODELING OF ROBOTS USING RECURSIVE NEWTON-EULER TECHNIQUES. ICINCO2010, Jun 2010, Portugal

[17] Favre J, Jolles BM, Aissaoui R, Aminian K. Ambulatory measurement of 3D knee joint angle. J Biomech. 2008;41(5):1029-1035

[18] M. El-Gohary and J. McNames, "Human Joint Angle Estimation with Inertial Sensors and Validation with A Robot Arm," in IEEE Transactions on Biomedical Engineering, vol. 62, no. 7, pp. 1759-1767, July 2015

[19] Seel, T.; Raisch, J.; Schauer, T. IMU-Based Joint Angle Measurement for Gait Analysis. Sensors 2014, 14, 6891-6909

[20] C.-L. Hwang and G.-H. Liao, “Real-Time Pose Imitation by Mid-Size Humanoid Robot With Servo-Cradle-Head RGB-D Vision System,” IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 49, no. 1, pp. 181–191, 2019.

[21] Bosch BNO055 IMU Fusion Datasheet

[22] IMU based motion sensing - Bijo Sebastian, Wordpress

[23] MQTT Protocol

[24] Robot Operating System

[25] CoppeliaSim - Robot simulator

[26] RBDL - ORB - GitHub

[27] Pypot Dynaban Edition

[28] Dynaban - Rhoban Project, GitHub

[29] Dynamixel AX-12A - Datasheet

[30] Dynamixel MX64 - Datasheet

[31] ESP - IoT Development Framework

[32] ESP32 - DevKitC

[33] Resource link: Google Drive

[34] Resource link: GitHub