CONTROLLED FLIGHT OF HIGH DOF HUMANOID ROBOTS

by

Gowtham Tummala

A Thesis
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Master of Science
Electrical Engineering

Committee:

_____ Dr. Daniel M. Lofaro, Thesis Director

_____ Dr. Kathleen E. Wage, Committee Member

_____ Dr. Nathalia Peixoto, Committee Member

_____ Dr. Monson Hayes, Chairman, Department of Electrical and Computer Engineering

_____ Dr. Kenneth S. Ball, Dean for Volgenau School of Engineering

Date: _____ Summer Semester 2021
George Mason University
Fairfax, VA

Controlled Flight of High DOF Humanoid Robots

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

By

Gowtham Tummala
Bachelor of Engineering
Anna University, 2017

Director: Dr. Daniel M. Lofaro, Professor
Department of Electrical and Computer Engineering

Summer Semester 2021
George Mason University
Fairfax, VA

Dedication

Every challenging work needs great efforts as well as guidance from elders especially those who are very close to our hearts. I dedicate this thesis to my loving big joint family, great grandmother Chitemma (deceased), grandfather Rama Rao, my parents Lokeswara Rao, Ratna Kumari, and my sister Harika, to my uncles Bhagavan, Narasimha Rao, Jagadeesh, aunts Hema Latha, Vijaya Lakshmi, Mohana Roopa, and cousins Bhuvitha, Medha, Reshmitha, Sumedh, Manas, Aalaya, Rishi, Dhruv. Whose affection, love, support, encouragement make me able to get such success and honor. And also to all my teachers along this journey.

Acknowledgments

First and foremost I would like to sincerely thank my thesis director, Dr. Daniel Lofaro, for introducing me to the world of research, patiently guiding me through the program, and provided all the support required in completing my thesis. Thanks to my thesis committee for serving and giving valuable time and insights.

Special thanks to my uncle Narasimha Rao for his support, care, and time throughout my master's journey.

I would also like to thank Alberto Herranz for his contribution in designing the robot, selecting the components required for the robot.

I want to thank Dr. Kathleen E. Wage and Dr. Nathalia Peixoto for serving on the committee and sharing her insights on improving this thesis document. I am also grateful for their excellent feedback to improve my writing. Thanks also to Dr. Monson H Hayes for supporting on the thesis submission and giving his valuable time and insights.

I cannot resist acknowledging all family members, relatives who supported me to reach such heights.

Lastly, I would also love to thank my friends for understanding, supporting, and providing ample memories throughout my master's journey.

Table of Contents

List of Tables

List of Figures

Abstract

CONTROLLED FLIGHT OF HIGH DOF HUMANOID ROBOTS

Gowtham Tummala

George Mason University, 2021

Thesis Director: Dr. Daniel M. Lofaro

This work aims to expand the abilities of humanoid robots, by implementing flying capabilities to the robot. Humanoid robots are designed to mimic the kinematics of a human, i.e. two arms, two legs, and a head. With this structure, humanoid robots can be designed and programmed to perform a variety of tasks. Some examples of full-body humanoid robots include Hubo [1], NAO [2], iCub [3] [4] (an open-source cognitive humanoid robotic platform), Atlas [5] by Boston dynamics, Honda's ASIMO [6], and Valkyrie [7] from NASA. Some examples of upper body humanoid robots, humanoid robots with wheels and tracks instead of legs, include Handle [8] by Boston Dynamics and Mitra [9] by Invento Robotics. Increasing mobility options for humanoid robot makes them more versatile.

In this work, we focus on adding a mobility option of flying to the humanoid robot. This was done by adding thrusters to the end-effectors of its high Degrees of Freedom (DoF) robotic arms and using control methodology for stabilization. Specifically, in this work, we study the ability of the latter robot to stabilize over the rotation in the x-axis. To test our algorithms, we built the physical robot via additive and subtractive manufacturing methods. We utilized computer-aided design (CAD) as well as computer-aided machining (CAM) for manufacturing the robot. This resulted in a 6-DOF upper body of a humanoid robot with ducted fans on its end-effectors that operates in the coronal plane. A test fixture

that allows for full motion of the robot and ground truth measurement was also created. We used system identification to create a mathematical model of the dynamics of the robot. This model was then used to design a controller for stabilization over the x-axis. We applied this controller in a simulation, then confirmed our results with the physical robot. In the results, we were able to achieve stability over the x-axis with an overshoot of about 30% and a settling time of approximately 1.12 seconds in simulation. On the robot the settling time is about 8 secs with no overshoot.

Chapter 1: Introduction

Robotics is not only a combination of science and engineering, but also psychology, anthropology, economics, etc. Robotics also involves a variety of tasks that include designing, operating, constructing, learning, controlling, feeding back data, processing data, and actuating. Its main objective is to make an intelligent machine.

The field of Robotics is becoming wider and wider with multi-department groups such as mechatronics, and the concentration of each group is further becoming multidimensional. In the past couple of decades, there has been a huge spike in research on humanoid robots. Researchers spend a lot of time in studying the behavior and structure of the human body while building humanoid robots. The main reason for developing these types of robots is they are able to work in the human environment without additional requirements. On the other hand, flying robots are also complex control systems in the huge 3-D world. There are a number of state-of-the-art flying robots which provide flying capabilities, for example, producing thrusts by rotating blades (helicopters, quad copters), propellers (aeroplanes), flapping-wing robots etc. Here, we are mainly focusing on the systems like quad rotors because of their position and functionality provided for the robot. Aerial vehicles such as Unmanned Aerial Vehicle (UAV), Urban Air Mobility (UAM), and drones are being used in a wide range of real-time industries such as e-commerce, surveillance, agriculture, military, photography, toys, and first responders. For these systems, more agility and/or more intelligence are incorporated by the researchers by implementing non-linear control laws that are capable of controlling the system with partial state measurements. To take a step forward in the complexity, there are robots with multiple modes of operation and multiple modes of mobility. Here we will see few examples of these specific robots. Robots with multiple mobility options are: aerial robots that can swim(loon copter) [10], walk [11], and roll [12]; humanoids that can swim (Swumanoid) [13], fly (iCub) [3] [4] [14] [15]. In

[16] and [17], the robots have different modes of operation. Researchers are building such models to improve versatility, portability, and to switch between energy-efficient mode of operation and performance operations. Such designs are not generic but used for special situations to overcome slow traverse in terrains and operate for long intervals. This paper takes a step forward in understanding and demonstrating the humanoid locomotion with additional flying capabilities.

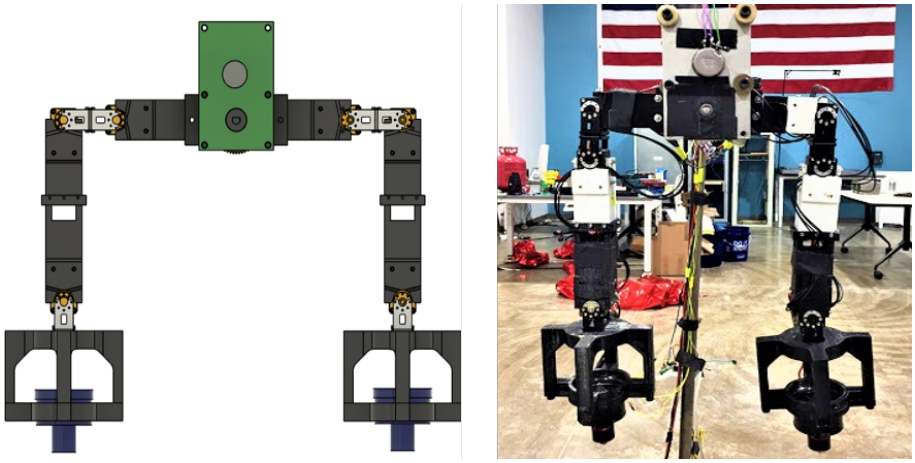


Figure 1.1: Introduction to the robot. (Left) Designed CAD Model of the 6 DOF robotic arm in coronal plane with thrusters at the end effectors and angle feedback sensor system at the center of the robot (green colored block). (Right) Real-world robot with 3-D printed links and joint actuators attached. Thrusters as the end effectors with a casing for them. Power and communication cables to all the electrical and electronic components.

This work is focused on building a robot that is similar to the humanoid robots upper body. This robot consists of two arms with a pivot in the middle supporting the arms on a gear system encapsulated with the aluminum structure that helps in balancing and hanging the whole system. Each arm has 3 joints namely; a shoulder joint, an elbow joint, and a wrist joint. The scope of this project is aimed at the coronal plane; that is, all the joints have single angle of rotation. For each arm, at the end effector, an electric ducted brushless DC motor is attached and used to provide thrust. The thrust generated helps the flight of the robotic arms, to control the roll angle and to balance the robot. The whole robotic arm structure is held by a rigid frame at a pivot point, which helps in performing

the experiments. The overall configuration of the robot is symmetric over the sagittal plane so that balancing the weight distribution should not be an additional task. Here, we will analyze the flight dynamics of this robot's structure in detail by creating a system equation and developing a controller in the forthcoming chapters.

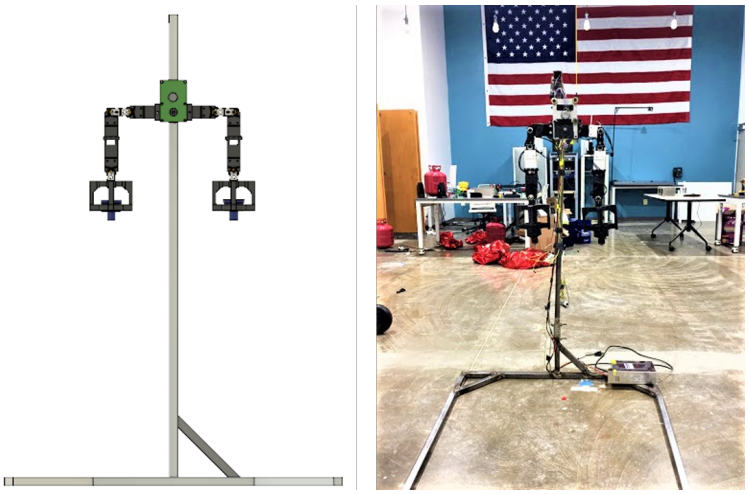


Figure 1.2: (Left) Front view of the CAD model of the robot hanging to support structure. This structure helps in testing the robot with different arm positions. Designing the robot model is clearly explained in the Chapter 3. (Right) Front view of the real-world system with the supporting structure with the power and communication cables. The whole robot is hanged to the support structure to measure the roll angle from the sensor system that is placed in between the two arms.

There are very less resources, simulations and experiments available on flying humanoid robots. The work done in [3] and [4] were simulations of the iCub robot with jet propellers attached to the limbs of humanoid. In this paper, we understand the control theory of the humanoid's stability and flight dynamics of aerial vehicles such as quadcoptors. We will also focus on improving the mobility of the humanoid by designing a stable controller using the PID control. The objective of this paper is to demonstrate the stable control of those systems while in flight. Here, the robot is designed using CAD tool and built using additive and subtractive manufacturing methods. With this robot, we first implemented PID controller and the gains are choosen by trail and error method until the response of the output is expected, we call this implementation a naive implementation of the PID

controller. This method is not feasible in the long run, if the arms' positions are changed we need different PID gain values. So, we gave a complicated task to the system, in finding the proper gains for the controller. For that, we need the system equation. To get the system equation, we used systems identification methods. To be more specific, we used black box modelling to get the system TF. First, we measured system response with an input u. Then using model matching, assuming the system is a second-order system we calculated the plant equation $G_p(S)$. This second order system equation to be calculated using damping co-efficient ζ and natural frequency ω_n terms of the system response; this is discussed in detail in the Systems Identification section in this document. Then, a lead-lag compensator is designed with the desired transient responses; this compensator adds a pole and a zero to the present system for the expected behaviour of the system. Finally, with the compensator TF the PID controller gains are calculated; as the PID TF and compensator TF are same. Thus, we achieve a smooth control with the desired transient responses. These calculated gains are tested both in the simulation platform and on the real robots to analyze the results.

Different experiments were performed to stabilize the system using PID gains at different maneuvers. In simulator the results shows that system reaches the stable state within 5 seconds with a percent overshoot of about 30% and a rise time less than 2 seconds. Tests on the robot shows a bit different from expected as this is approximated system. It took around 8 seconds to settle with no overshoot and less rise time.

1.1 Organization of Thesis

The rest of this thesis document consists of the following chapters. Chapter 2, Background, with the literature review on the base papers, humanoids, and aerial vehicles. Next, Chapter 3 Consists of the system construction and design. Chapter 4 System Identification, methods used to find the system equation. Chapter 5 designs the control system for the identified system. Chapter 6 is the complete set-up of the system used for this experiment. Chapter 7 has the results of tests performed.

Chapter 2: Background

Flying humanoids are the combination of humanoid robots and aerial systems. Humanoids and aerial vehicles such as quad rotors, drones, etc. are built with more elaborate control systems. In this chapter, we will go through papers on flying humanoids and go through the important concepts in building one from them. A conceptual design for the flying humanoid was developed by Ruangwiset in [14]. There he constructed a mathematical model on humanoids with two rotors on shoulders. This mathematical model helps in steering the flying humanoid robot, with the Center of Gravity (CG) shifting mechanism. They used the control laws of the Tommy i-SOBOT humanoid robot and developed the CG shifting model on it, using PID control laws. Performed number of numerical computations for the feasibility of the concept. The other flying humanoid is the simulation on iCub robot by Pucci, et. al. in [3] and Nava, et. al. in [4], it is powered by 4 thrusters on the limbs of an iCub humanoid robot, an open source robot developed by the researchers of Italian Institute of Technology to achieve flying capabilities. The iCub robot is made up of n joints and $n+1$ rigid body parts called links for its structure. This robot's whole body control laws are capable of stabilizing the overall system dynamics in both humanoid operations and aerial robot operations. These papers shows the simulation approach to the whole humanoid body. As the iCub is an open source project, it is available as a research material. To be more specific about the simulations performed by these researchers, the robot is attached with four thrusters on the four limbs, two on its arms and the other two on its legs. They used turbo engines for thrust force to lift the humanoid with a wide range of control objectives and error dynamics; centroidal momentum asymptotic stability and the ability to track the reference trajectory, setting error boundaries between reference orientation and robot frame. In addition to these objectives, the moment control, velocity, position control,and orientation control laws were defined and explained which provided stable results. For

optimization, stabilization, and controllability of the system, they assumed the humanoid system to be a VTOL (Vertical Takeoff and Landing) system. Here, with an under-actuated robot these equations are built based on Newton-Euler formalism. But in the simulation, errors were not resolved with the proposed control laws. Considering the results of the [3] that produced more errors, they designed a new approach in [4]; Task-based control. Task-based control's main objective is global tracking of desired Center of Mass (CoM) location. The results in this case [4] were much better than in previous paper's [3]. Using this control in the iCub, it could perform complex tasks but there are additional assumptions like omitting aerodynamic effects, not considering take-off and landing phases. Overall paper shows an approach to design and simulate the flying humanoid concept with certain laws into account. In the following sections we will focus on different control laws, performance, and implementations in detail for both humanoids and aerial vehicles.

2.1 Humanoid papers

Proper performance for a humanoid robot can be explained as the capability to traverse without falling on the ground. In this sense, the fundamental target of the control in a humanoid robot is to ensure a reasonable ground response strength to keep the dynamic balance explained in [18]. However, for better performance, the robot must adjust to unseen forces from outside called disturbances and noise for better performance. In general, the robot must follow desired trajectories that include actuation of basic position regulators which can be attained by utilizing dynamics based walking pattern generation. Because of the limited foot area, pure position control is lacking for executing these trajectories it can be a problem in defining the control laws. In this section, we will research a bit deeper to understand the structure of humanoids, different control laws to make the system stable and work on practical implementation of it.

In general, humanoids are divided into upper body and lower body [19]. The upper body consists of upper limbs (hands), sensors such as inertial measurement units, position,

lidar, proximity, etc. The lower body consists of legs used for locomotion with a different set of control strategies than the upper body for balancing [20], walking [21] [22], whole-body control [23], and trajectory planning [24]. Each limb, hand or leg, will be a robotic arm in a perspective with different end effectors explained in [25]. For example, if you take a hand, the origin will be at the shoulder and the end effector is fingers. In order to move a particular location, it uses inverse kinematics and maps the trajectory to the destination. While this operation is in progress the dynamics of the humanoid changes but whole robot should be stable. For all that to happen smoothly, a lot of control is involved. Kim, et. al. in [26] solves the major difficulties faced by the inverse kinematics solutions developed for humanoids scaling the design, motion postures and motor capabilities in detail.

In [27] the effects of uneven surface on humanoid robots are explained in detail. Each humanoid robot consists of an Inertial Measurement Unit (IMU) or a group of sensors that provide data of the system required to achieve horizontal motion, stability from high-frequency vibrations, and attitude stability[19]. Louis et. al. in [28] used Force Sensitive Resistors (FSR) while pushing the cart using humanoid robot. A few authors also proposed the Linear Inverted Pendulum Model (LIPM) [28] [29] [30] [31]; in this model they proposed a cascade controller design to withstand external disturbances that minimizes the vibrations. This model also follows the predefined trajectories of the Center of Mass (CoM) and horizontal motion with small variations and external disturbances. In addition, General State Equation (GSE) [32] and Zero Moment Point (ZMP) [21] [23] [32] [28] approaches are used for dynamic analysis for humanoids. Stephen, et. al.[31] proposed control equations that contains ZMP and LIPM to follow desired CoM using Dynamic Balance Force Controlling.

Control laws with a prediction method; Model-based Predictive Control (MPC) is used to predict the future trajectory of CoM [33] [34] and a layer of PID controller is cascaded that helps in modifying the ZMP reference [35]. Ivancevic, et. al. in [36] uses a different control model, the fuzzy-stochastic–Hamiltonian model and developed the humanoid as a 3 stages non-linear dynamic system. In the book [37], they proposed momentum-based

strategies for the stability of the humanoid robot and also tested it on the iCub robot and shown some satisfying results yet they need to compromise in the effectiveness in balancing. In [38], Hyon demonstrates the contact force control framework for humanoids on the rough terrain that does not require any prior knowledge of the terrain and frictional force. This is based on the torque exerted on the humanoid robot at the joints. In the paper [39], the robot performs dance motions satisfying dynamical consistency and mechanical constraints. Here is one more example for the application based development of humanoid robots, in the papers [40] and [41] illustrates an approach to throw an object using Jaemi Hubo robot and they are successful in achieving it.

2.2 Aerial Vehicles

Recently, the advancement of flying robots has rapidly evolved. Aerial vehicles such as UAV, UAM, and drones are used in a wide range of real-time applications. A few of such industries are as follows e-commerce [42], surveillance [43] [44], agriculture [45] [46], military, photography [47], toys [48], first responders [49] and etc. These applications have agility and intelligence incorporated in building them. The complexity is increasing version after another by implementing the finest non-linear control laws that are capable of controlling the system with partial state measurements [50], actuators saturation [51], robustness with variations of CoM [52], parametric uncertainties [53], aerodynamic pressure differences [54].

Aerial robots that fly can produce thrust in different ways, such as thrust produced by rotating blades [55] [56] [57] [54], and by flapping-wing robots [58] [59] [60] [61]. Some of the engineering milestones are as follows; quadrotor ball juggling[62], trajectory tracking [63], and so on. The control systems in such robots are getting more and more complex and performing tasks with more precision, accuracy, and without performance degradation. We are focusing on the actuators that produce thrust using rotating blades; that are made of plastic run on brushless motors [54]. In addition, the system of the quadrotors is also not quite the same as the usual robot controllers; these are very sensitive to slight disturbances.

Just 4 actuators, thrusters, in quadrotor be able to control 6 DOF movement. In such a case, four state variables of the quadrotor can simultaneously be balanced out into a steady-state position smoothly with a static feedback method. Azzam et. al. in [64] presented non-linear control law based on Newton-Euler formalism for the quadrotor.

In general, the quadrotor system is highly under-actuated as it performs number of maneuvers. The basic quadrotor system is capable of having 6 DoF with only four actuators and this may extend up to 18 in the real-time system. The Control system of such systems is complicated. There are two types of approaches: Centralized and Decentralized approach based on the task aerial system was built for [65]. In a centralized approach , the whole body is considered as a single system so as the control laws for stability and performance. In a decentralized approach, the aerial system and a manipulator are considered as two independent systems. The effects made by manipulators are considered as the external disturbances on the aerial system and vice versa.

Though there are different methods to lift the system into air, this type of application requires VTOL for its manipulation capabilities, versatility, agility. Both the approaches have their advantages and disadvantages based on the application. With all the above knowledge, we use a decentralized approach for our system in building the robot. We will see in detail in the further chapters.

Chapter 3: System Design

In this chapter, we will discuss the design and the construction of the robot. This robot is mainly built on 4 types of links; first piece, second piece, middle piece and thruster structure. The construction of the robot is the group of structures embedding actuators and sensors into 3-d printed parts. This is discussed in detail in the following sections. The scope of this project is limited to the coronal plane and the roll motion of the robot, so each link and joint is constructed with constrained movements.

To design the system with real-world metrics, we use CAD (Computer-Aided Design); it is a software tool to create, analyze, modify, and optimize a design. The CAD tool is used to increase the quality and productivity of the work. There are a lot of such CAD tools in the market. In this project, we are using Autodesk Fusion 360 to aid our robot with precise measurements. The design aim is to build two 3 DOF arms that resemble human upper limbs but operate only in the coronal plane. For the actuation, at the joints, we are using smart actuators (Dynamixels) to control the movement of the hands. While designing, certain things should be considered; the dimensions of the actuators, the freedom of actuation, designing the compatibility between the parts, room for power and communication cables between the actuators and the single board computer or micro controller. Dimensions of the actuators and sensors were taken from the website to build our robot[66].

3.1 Design of the robotic arms

For a human arm, we have 3 joints; the wrist joint, the elbow joint, and the shoulder joint. To join them we need two links. Here the first piece is used in two places as shown in Fig. 3.1; one between the shoulder joint and the middle piece, another between the elbow joint and the second piece (wrist joint link). The actuators are placed inside the first piece at

the top and screwed to hold it firmly. The first piece also has the space needed to connect the cables. Another important characteristic of it is the actuation freedom; actuators have 200° of actuation freedom. With all these in consideration we designed the link that is best fit for this robot.

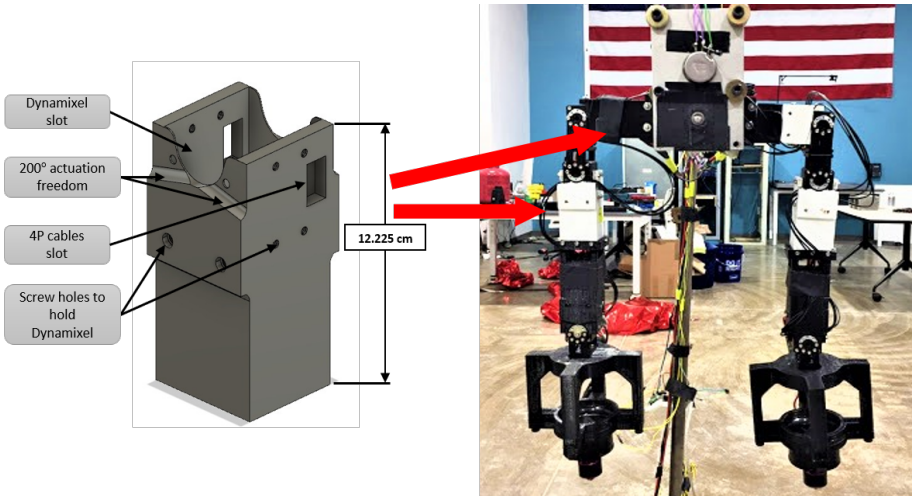


Figure 3.1: First piece: Holds the actuator with 200° actuation freedom and also provides slots for communication links for the actuator. This acts as the link between the angle feedback structure and the shoulder joint and also between the elbow joint and the second piece. Arrows indicate the location on the robot on one arm; it is also located on the same spot for another arm.

The first piece is good for its purpose.Now we will see the design of the second piece that could form the shape of an arm. The Fig. 3.2 is the second piece this acts as a link between the first piece and the wrist joint actuator. To design this part, we make sure the first part works with the actuators because the body of the first piece was used. The modifications were applied to the bottom part of the piece: this piece was made longer than the first piece; a square shape with a perimeter bigger than the piece that was added so it could hold the first piece and the actuator at the wrist joint. In addition, some holes were added so it could be screwed to the first piece. As the actuator part wasn't modified the properties of actuation remain the same.

Fig. 3.3 represents the manufactured parts using the additive manufacturing method

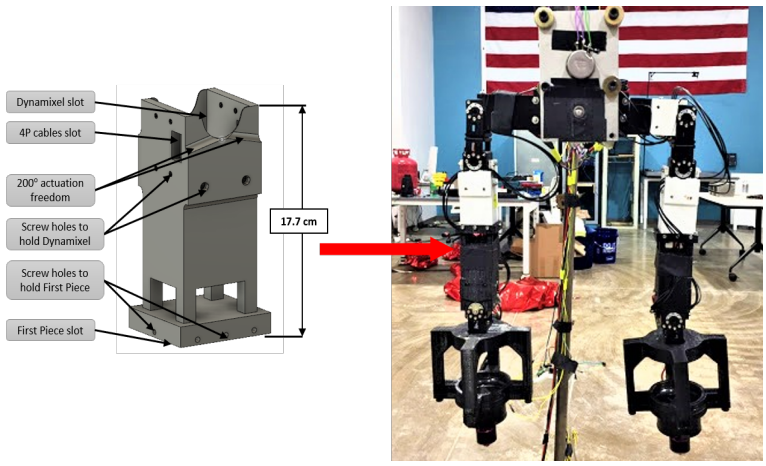


Figure 3.2: Second piece: This is developed from the first piece, so, this doesn't change the actuation properties that it provides 200° actuation freedom and also provides slots for communication and power links for the actuator. It acts as a link between the first piece and the wrist joint. The arrow indicates the location on the robot on one arm; it is also located on the same spot for another arm.

that is using 3D printing technology. The links are connected together using the FR12-H101 set from Robotis[66]. The white parts correspond to the first piece and the black part corresponds to second piece. This structure made the complete robotic arm with 3 DOF on single axis. Two of such arms completes our system i.e., 6 DOF robotic arms in the coronal plane. On the top, there is a gear system that balances both the sides and the arms on the bottom are fixed with a thruster actuator that has the thruster casing to hold it.

3.2 Connecting a angle feedback sensor (potentiometer) to the robot

In this section, we are going to discuss the middle structure; it is used to join both the arms of the robot and also holds the sensor. The Robot is an upper body structure with a total of 6 DOF robotic arms that are operated in the x-axis or the coronal plane. The middle piece is designed in such a way that it should be able to hold both the arms (the left and the right arms). So, for the strength it was made out of aluminum blocks using

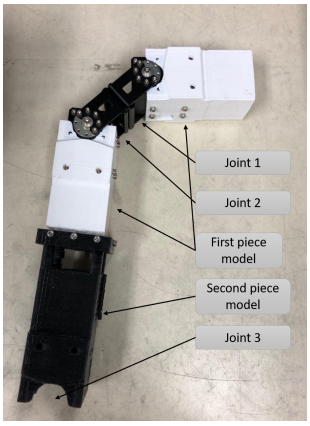


Figure 3.3: Using the additive manufacturing method, the single robotic arm with the 2 first pieces (printed with a white filament) and 1 second piece (printed with a black filament) are connected with joint actuators. Two of such arms were used to complete our robot. A middle structure helps to connect the two arms.

the subtractive manufacturing method. It not only holds the whole robot intact and also hanged to the support structure to perform the tests with different maneuvers (the support structure is discussed in Section. 3.4).

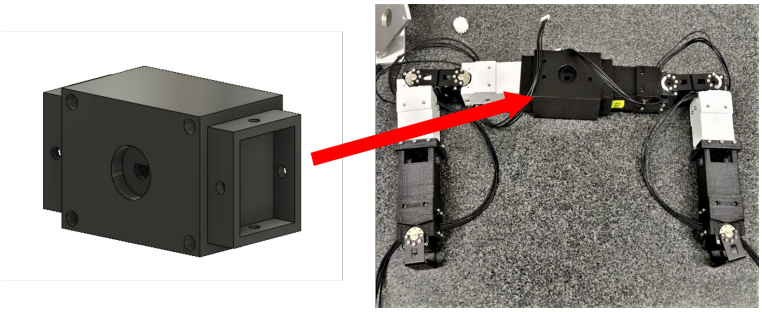


Figure 3.4: Middle piece: Used to hold the gear system that is attached to the angle feedback sensor. The two arms are attached on either side to complete the two robotic arms structure. This middle piece stays inside the middle structure.

The main goal of the middle piece is to measure the roll angle, the tilt in the x-plane. The sensor (Potentiometer) measures the roll angle of the robot and sends it to the micro-controller (Arduino) then control law is applied to generate the thrust value. That way it is feasible to know how the robot would behave with different input values. To get the

appropriate roll angle by the sensor, the middle structure is designed in such a way that the potentiometer is integrated with a couple of gears, made of PLA, which translate the rotational movement of the robot to the sensor.

One of the most important aspects of the design is how to let the potentiometer rotate freely while it is attached to the middle piece. We made a design with two gears, shown in Fig. 3.6, one that is attached to the bar that would go through the middle piece (robot), and another one is connected to the potentiometer to replicate the movement. The next step in the design process is to maintain the potentiometer and the gears in position while doing operations. To achieve that, the middle piece is placed in between two aluminum plates. These plates are not directly connected to middle piece but to provide support for hanging the robot and also to connect the whole robot to the support structure.

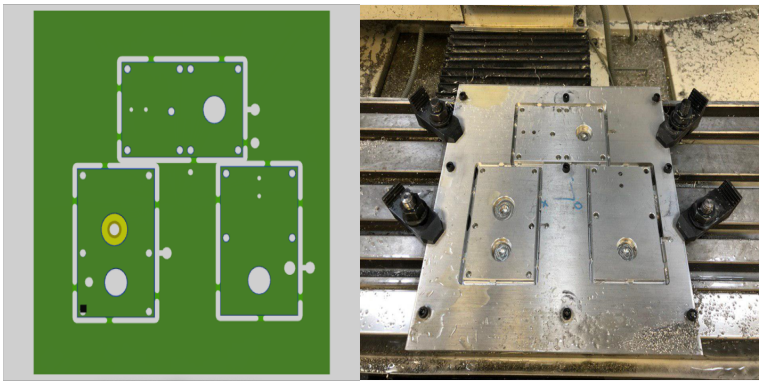


Figure 3.5: (LEFT) CAD model design of the aluminum pieces. These are used to hold the robots' middle piece, angle feedback sensor and gear system that connects firmly to the support structure. (RIGHT) The Computer Numerical Control (CNC) machine cut of aluminum pieces that is used to hold the robot to the support structure. It uses a number of operations like screwing, pocketing and contouring to get the appropriate finishing to the piece.

As mentioned earlier, the design of the structure was made in Auto desk Fusion 360; the plates, the gear system, the static parts and the relation between all of them were modeled and simulated. Three aluminum plates were cut off in the workshop; the following design is also created in the CAD software. In order to cut the aluminum pieces, a CNC (Computer Numerical Control) machine is used. Before doing operations on the CNC

machine, the aluminum block should be screwed to a metallic base so it would not move while the operations were in process. Different tools are used for this design, as we choose Aluminum the following are the operations to get the final design: screwing, pocketing, and contouring. Before the first operation, screwing, we defined some points in the design so that the CNC machine would punch little holes where the screw holes would be created later. Punching helps the CNC machine to not slip the bit while screwing. Then from the screwed holes the bit cuts the aluminum in sideways. The next step is pocketing; this takes large chunks of aluminum for the final shape if the design has big holes. Finally, contouring is done for the fine finishing with a different bit than the one used for pocketing. With all the components that are manufactured using both additive and subtractive manufacturing methods, now it is time to join the aluminum pieces (aluminum plates) with the 3D printed piece (middle piece) and the support structure. Here standoffs were used to connect the aluminum plates, leaving the 3D printed piece in the middle. In Fig. 3.6 the aluminum pieces are shown in green and the standoffs are in silver color and the middle piece (3D print piece) in black. This whole process of designing and manufacturing is an iterative process until we are satisfied with the design.

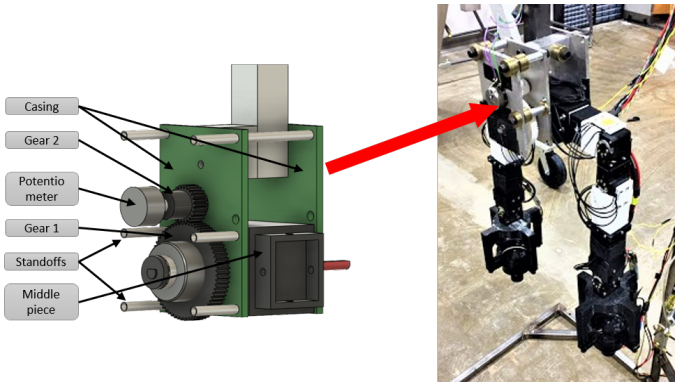


Figure 3.6: Middle Structure: Holds the middle piece, angle feedback sensor and, gear system to the support structure. The roll angle is captured by the sensor with the help of this gear system. Stand offs were used to provide enough space for the components to move freely.

3.3 Motor thrusters casing

The motors section for this system divided the work done in two different parts; one is selecting the motor with the correct specifications to be able to lift the robot, another is designing the structure that would keep the motors connected to the robot arms as the end effectors.

3.3.1 Selection of the motors

Selecting the right motor for the project requires some research about some important parameters and their definitions. Kv rating is the constant velocity of the motor, it is defined as the speed at which the motor would spin if 1 volt is supplied to it without applying the additional load. For projects like this, a motor that is small and has high Kv is selected. The revolution per minute (RPM) and amperes graph of each motor in the data sheet provides us a good estimation of how it would behave. Next, the thrust provided by the motor is also an important aspect to consider. After calculating the part's weights, selected a motor with eleven 50mm blades that can lift up to 770 grams of weight. The details of the motor are discussed in the chapter Experimental Setup.

3.3.2 Designing the thruster structure

The objective of this structure is to connect the motors to the robotic arms as the end effectors produce lift force to manipulate the roll angle. All the measurements of the thruster were taken manually as we could not get a datasheet. With these measurements we designed a model in CAD software that would fit the robotic arm. Also, the design should be narrow at the bottom and wide at the top so it could be able to hold the motors. The designed piece is separated from the arm at a fixed distance so the air flows through the motors to produce thrust. Screws were used to join the arm with the structure firmly. The Fig. 3.7 shows the final design of the piece responsible of holding the robot.

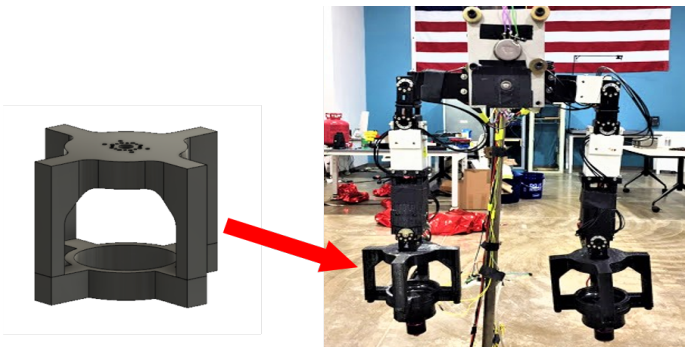
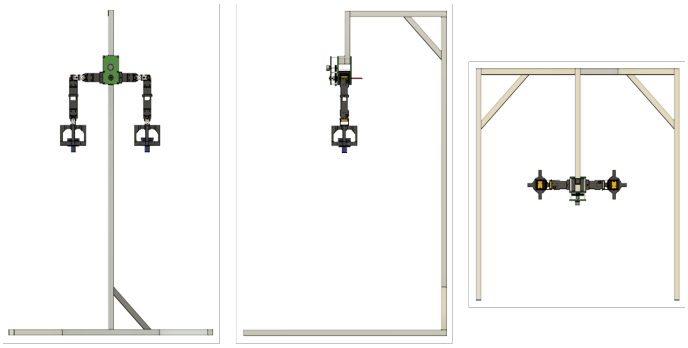


Figure 3.7: Thruster structure: Holds the electric ducted fan. The structure is designed in a way it has room to draw the air and create thrust. This is connected to Wrist joint to manipulate the direction of thrust. This figure also highlights the location of the structure on the robot.

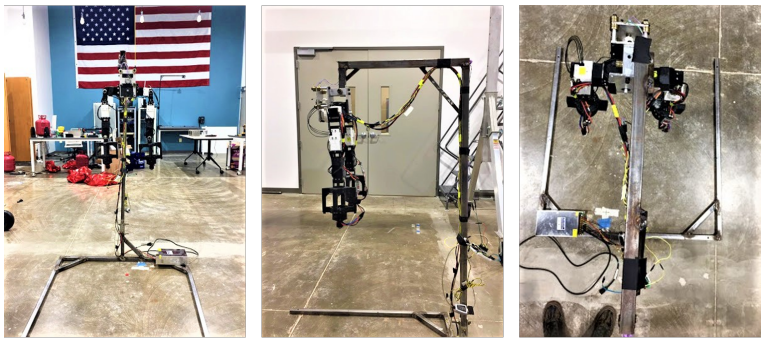
3.4 Creating the support structure

For the support structure, we choose to use a hot rolled one-inch square aluminum rod. This structure was meant to hang the robot arms and balance them over a pivot. The height at which we hanged the robot is around one meter from the ground so that the system would be accessible for anyone. In the workshop using the metal saw these rods were cut into required sizes as per the design. We also used a CAD model for the support structure as you can see in the Fig. 3.8. Once the parts are ready we welded the bars to make the structure as per design. We used shielded metal arc welding machine to join the aluminum rods. Shielded metal arc welding creates an electric arc in which the temperature reaches around 6500 F^{o}. Additionally, corner joints were placed for extra support (triangulating the corners). The below Fig. 3.8 and 3.9 shows three views of the whole designed structure in CAD and the real time model respectively. Leftmost is the front view, the middle figure is the side view and the rightmost figure is the top view of the overall structure.



(a) (b) (c)

Figure 3.8: CAD model of the robot with support structure in different views (a) Front View, (b) Side view, (c) Top view. It is rigidly attached to the support structure at a height around 1 meter from the ground. The robot is freely able to rotate around x-axis.



(a) (b) (c)

Figure 3.9: Physical robot with support structure in different views a) Front View, (b) Side view, (c) Top view. It is rigidly attached to the support structure at a height around 1 meter from the ground. The robot is freely able to rotate around x-axis.

Chapter 4: Systems Identification

In this chapter, we will discuss the Systems Identification concept in detail. The main objective is to develop suitable mathematical model for our dynamic system. Mostly, systems identification is experimental, which requires basic knowledge of the experimental setup and time to build good low-order systems. We can still develop models with unknown physical parameters for a complex dynamical system. In this project, we generated a model to build a controller for manipulating a few parameters like peak overshoot, rise time and settling time.

4.1 Introduction to Systems Identification

Systems Identification is a field of study in the control systems. This branch finds parameters or system equations of the dynamical system with only inputs and outputs from the system. It also incorporates ideal test plans for creating data effectively. A typical methodology is to begin from estimations of the behavior of the system with outside noises and try to decide a numerical connection between them without going into high precision details[67]. The progress and use of models are extraordinary in all objectives and purposes.

When exploring practical models with external disturbances, or models for fluid dynamics of definite fluids under explicit conditions, or models for a mechanical creation process in some robotized condition, the models are stressed on a variety of tests. When in uncertainty, designing the systems have one basic trademark: they are stressed over relations between parts and are described in physical quantities which can constantly be assessed by some appropriate implementations. One can measure force, speed, temperature, time, pressures etc., and this possibility of assessing gives the expert a huge information during the time spent while creating and operating the models.

While designing the control system model the following strategies are taken into consideration: system is treated as dynamical design, and determined to use in a couple of domains. Here are different types of models that can be used while designing one:

1. **Mental models**

   These are the same as the common idea that people have of the system in their minds.

2. **Programming models**

   These reflect structural representation that are in programming, mostly if-else conditions, discrete-event sorts of operations, and lookup tables.

3. **Graphical models**

   System representations as (nonlinear) characteristics and outlines, unit step, and step responses.

4. **Logical models**

   Structure delineations as logical relations as (partial) differential and qualification conditions

In this, we will concentrate on models of the last two arrangements, and explicitly on logical models, while the essential focus will be on direct time-invariant models as differentiation conditions. Inside the control block, one will adapt the strategy with models of dynamic systems, having different sorts of usages as a principle need. Some ordinary model applications are:

1. **System design**

   Learning about the (dynamic/static) system can be used for optimizing. In case a system isn't generally controllable, it must be redesigned to improve controllability and, accordingly, on achieving performance.

2. **Control design**

The Control system design should establish stability, controllability, reference tracking, etc.; or a control system that improves or redesigns the efficiency/performance of a dynamical system.

3. **Simulation**

   Testing the system under different data circumstances, getting ready for implementation, etc.

4. **Estimation**

   Measuring the output that are not naturally quantifiable; redoing the state of a system.

5. **Diagnosis**

   Detecting the system that it has any faults or errors.

In general, models are of three types, 1. White box, 2. Black box and, 3. Grey box.

1. **White box**

   A physical/logical representation of the dynamical system for which a lot of data such as parameters or system equation is known.

   Once the model description is chosen, system identification includes estimating the unknown parameters of the transfer function directly or performing tests to gather the parameters indirectly.

2. **Black box**

   An observational representation of the dynamics of a system for which no data is known. The system model is created from the responses when a variety of inputs provided to the system.

   Our system in this project is designed on this concept.

3. **Grey box**

   A mix of the two. If the "known" white-box part can be deducted out, the unknown part might be recognized by utilizing a few techniques.

4.2 System Modeling

The system is designed in the CAD software that resembles human's upper limbs, built on six actuators called joints that are connected with seven 3-D printed parts called links and two thrusters called end effectors. For 3-D printing, we used polylactic acid (PLA) filament that is biodegradable and environmentally friendly material. Here, the whole system is considered as a 6 DoF system and the origin is at the center. The system is symmetrical over the X-Z plane also known as the sagittal plane can see in Fig. 4.1.

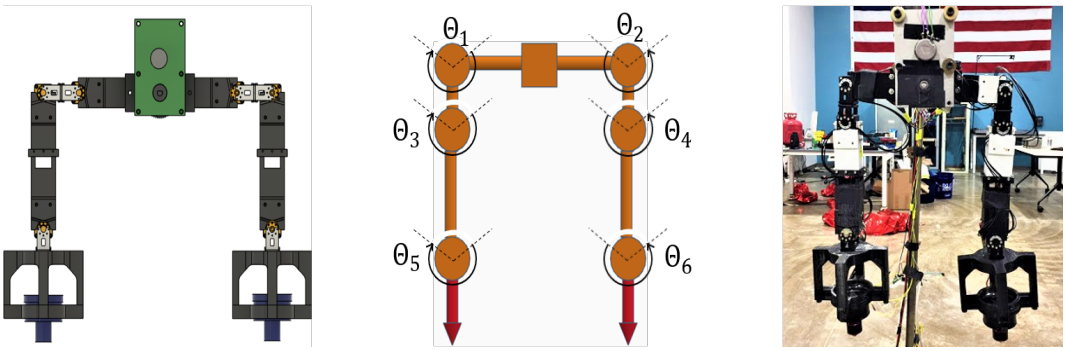


Figure 4.1: Three versions of the 6 DoF Robot: (Left) CAD Model, (Middle) Kinematic Model, (Right) Real Model. This model is symmetrical to Sagittal plane. We are focusing on the roll motion of the robot. Assuming our system is a second-order system to calculate the plant equation $G_p(S)$. Robot has 6 actuators at joints, a sensor to measure the roll angle with reference to ground at the center and the thrusters at the end effectors of each arm.

We assume our system as a whole is a second order system as it reacts around the x axis with the data from the system model above. Using transient specifications we can find the plant equation. The transient specifications are: *rise time* (t_r), *peak time* (t_p), *2% steady state time* ($t_s2\%$), *percent overshoot* ($PO(\%)$). The pictorial representation of these transient specifications can be seen and in Fig. 4.2 [68].

The following is the Second-order system Equation,

$$G_p(S) = \frac{1}{S^2 + 2\omega_n\zeta S + \omega_n^2} \tag{4.1}$$

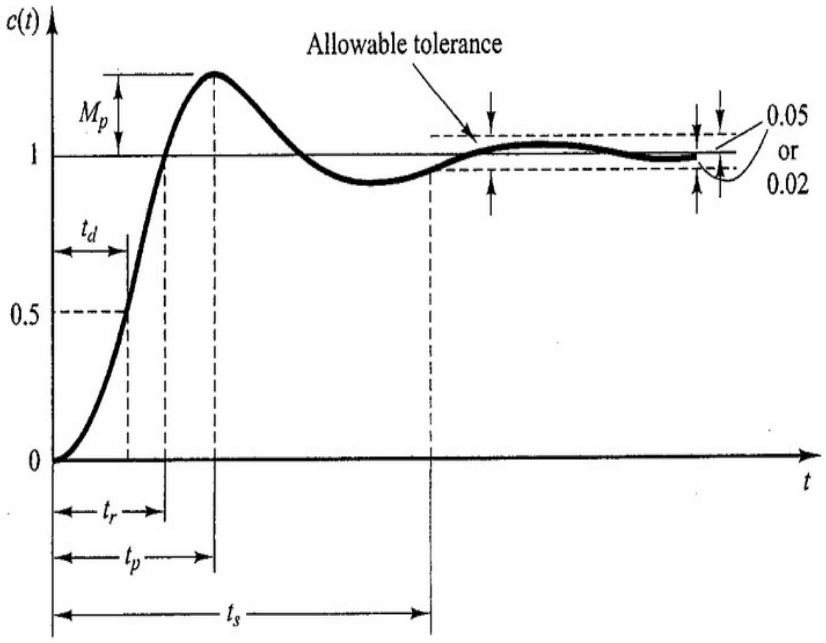


Figure 4.2: Transient specifications response of a second order system with natural frequency ω_n and damping coefficient ζ. From the response plot we can find the following transient specifications;*delay time* (t_d), *rise time* (t_r), *peak time* (t_p), *settling time* (t_s), *percent overshoot* (M_p *or* $PO(\%)$).

Transient specification Equations

$$PO(\%) = 100\ e^{\left(\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}\right)} \tag{4.2}$$

$$t_p = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}} \tag{4.3}$$

$$t_r = \frac{\pi - \tan^{-1}\left(\frac{\sqrt{1-\zeta^2}}{\zeta}\right)}{\omega_n\sqrt{1-\zeta^2}} \tag{4.4}$$

$$t_s(2\%) = \frac{4}{\zeta\omega_n} \tag{4.5}$$

With the above equations (4.2) (4.4) (4.3) (4.5), using two or more specifications to compute the damping factor (ζ) and the resonant frequency (ω_n), we developed a second order system model Eq. (4.1).

The Fig. 4.3 represents the system for modelling, while the input u to the system $G_p(S)$ will provide the response of the system for the given input. The time and theta values are recorded to know the system response. Here, we now know the inputs and outputs but not the system equation. Finding a system equation based on input and outputs is known as a black-box model.

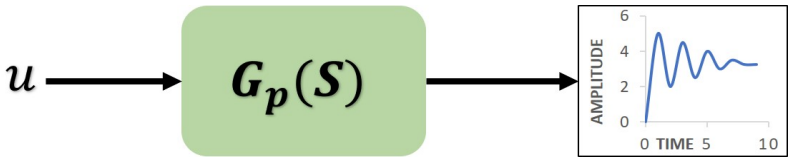


Figure 4.3: System response Model. With a step input u to the system we get the system response as shown in Fig. 4.2 to calculate the transfer function using the transient specifications from the system response. The second order system transfer function $G_p(S)$ is referenced Eq. 4.6. The calculated values for the plant equation is in Eq. 7.1.

For example, in order to model the system a step input signal, u, is given to the plant, $G_p(S)$, in this case, 50% duty cycle that produces 280 grams of thrust is given to only one thruster either left or right in the coronal plane as shown in Fig. 4.4. The potentiometer which is an analog sensor that is located at the pivot measures the rotation of the system in the coronal plane. Then the measured data of the system is recorded from the potentiometer. This data is the system response for a given amount of thrust shown in Fig. 4.5.

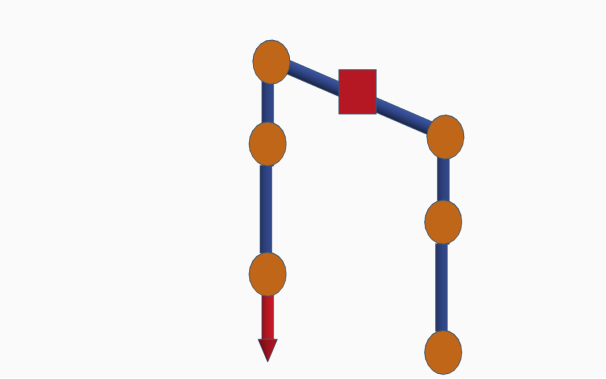


Figure 4.4: Kinematic model demonstrates the robot posture when a step input given to a single thruster of the 6 DoF robotic arm. As the robot has its free rotation in the x-axis, the actuation of the robot is shown. The sensor in the middle structure records the data of the angle feedback to create the plant model.

4.3 System Matching

We have a real system model with a higher-order, to get the plant equation for simulating we have to match the system model to the real system. By manual tuning with the addition of gain (K) and slight changes in damping coefficient and natural frequency gives the best matching case for the system model. With the final system equation $G_p(S)$ we can simulate the controller.

$$G_p(S) = \frac{K}{S^2 + 2\omega_n \zeta S + \omega_n^2} \tag{4.6}$$

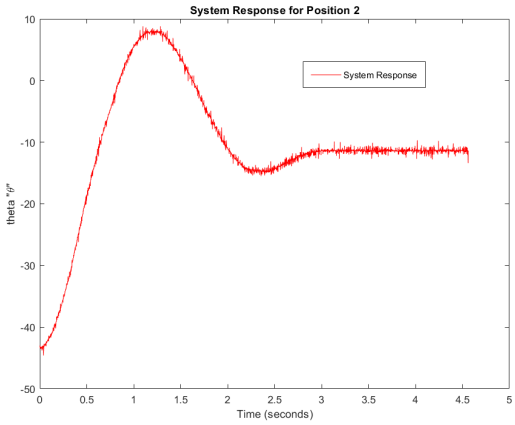


Figure 4.5: This plot is an example for the system response when a step input u is given to the robot. This plot represents the roll angle of the robot w.r.t. time from t = 0 when the step input u given. The system has lot of noise because of its long wires. For this plot, a 50% Duty cycle is fed to the system as input u.

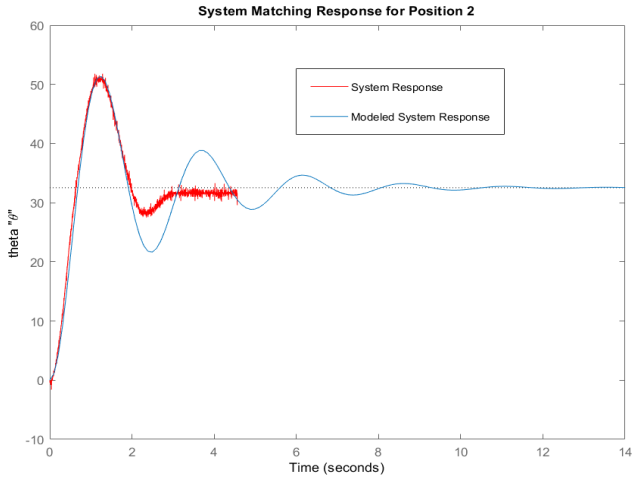


Figure 4.6: Blue color line represents the step response of the calculated second-order mathematical equation of the system response from Fig. 4.5 represented in red. Matched system (blue line) is having approximated natural frequency ω_n and damping coefficient ζ as of the system response in Fig. 4.5.

Chapter 5: Control System

In this chapter, we will discuss the design and implementation of the control system for our robot. The main objective here is to design the controller for the dynamic system that we developed using Systems Identification. We will discuss about the compensators and PID controller in detail and the necessary equations to build a control law.

5.1 Introduction

A controller is added to the system to improve the performance, decrease the unwanted noise and properties that enhances the final result. To design a controller the following properties need to be considered: the steady-state error, the maximum overshoot, the rise time, the peak time, etc. We have a variety of controllers to enhance the system, depending on the specifications and cost we have to choose the right one.

The advancements in technology and engineering, just as the synergetic combination of various principal engineering areas made the engineering issues get more earnestly, more extensive, and more profound. Issues that are in different fields and require a multidisciplinary engineering system, such methodology is called mechatronics approach, and such multidisciplinary systems are called mechatronics systems. It is characterized as a multidisciplinary idea, it is a synergistic joining of mechanical engineering, electrical engineering, electronic systems, control systems. To manage vulnerability, communication through the structure, and assembling of items from the beginning planning the process is important. In this manner empowering complex dynamic system, outstanding degrees of exactness and speed of cutting edge gear including the capacity to perform complicated and high-quality movements is part of developing a mechatronics system. Mechatronics systems work with high precision and speed inspite of unfavorable consequences of system nonlinearities and

vulnerabilities. Since accomplishing and confirming precision in mechatronics systems' exhibition is of concern, the most basic choice in the mechatronics configuration process is the selection and design of both control unit and control algorithm which are directly related to one another. There could be many alternative control methodologies for a particular scenario but the designer should choose the best controller based on the application use-case.

For our mechanical system, a controller designed with expected behavior from the system response parameters such as the steady-state error, maximum overshoot, rise time, peak time, etc. that are determined while designing the controller.Controllers alternatives including yet not constrained to: Micro-controller, Programmable Logic controller (PLC), Computer control, Digital Signal Processing (DSP) integrated circuits. Following are some types of controllers

- Bang-Bang (ON-OFF) Controller
- PID Controller
- Compensators
- Fuzzy Controller
- Adaptive control
- Neural network control
- Model Predictive Controller (MPC)
- Model Reference Adaptive Control (MRAC)
- Optimal Control

5.2 PID Control

PID control is one of the most well-known methods in control systems. PID controllers are regularly utilized in various fields such as industries, educational institutes, automobiles, etc. and a huge industrial facility may have a huge number of them, in machines, lab hardware etc. In system applications built for engineering tasks, the controllers show up in a wide range of areas.

In this subsection, we will discuss about the fundamentals of PID control, and many parts of control can be comprehended dependent on linear analysis discussed in [69]. And also, we will see the implementation of PID controllers, similar techniques can be utilized

to execute numerous different controllers shown in detail by Jadoon et. al. in[70].

In the PID controller, k_p, k_i, k_d as the proportional, integral and derivative gains of the controller respectively where, $e(t)$ is the error between the desired and actual value. The proportional part contains linear variation of the error $e(t)$, provides stable operation but gives steady state error. The integral part, $\int_0^t e(\tau)d\tau$, is the sum of the present error and the leftover error taken from the previous time steps when the error is zero the integral part stops growing this value gives the necessary action to decrease the steady state error. The derivative part, $\frac{de}{dt}$, is the rate of change of error as the input for the controller. It predicts the future error of the system and makes the system operate smooth and within the bounds. In general, systems with very low time constant and system that is less effect with the noise don't use derivative constant because it is designed to react to the rate of change of error.

This equation is represented in both time domain and laplace domain. Eq. 5.2 is the parallel form and Eq. 5.3 is the ideal form representation of the PID control system equation[71].

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau)d\tau + k_d \frac{de}{dt} \tag{5.1}$$

Also written as,

$$G(S) = K_P + K_I \frac{1}{S} + K_D S \tag{5.2}$$

$$G(S) = K_P\Big(1 + \frac{1}{T_I S} + T_D S\Big) \tag{5.3}$$

where,

$e(t) = d(t) - y(t)$ $\quad$ $d(t) = desired\ output$

$u(t) = input$ $\quad$ $y(t) = output$

$k_p = Proportional\ Gain$ $\quad$ $k_i = Integral\ Gain$

$k_d = Derivative\ Gain$ $\quad$ $\frac{de}{dt} = rate\ of\ change\ of\ error$

$T_I = Integral\ time\ constant$ $\quad$ $T_D = Derivative\ time\ constant$

These constant gains differ from system to system and also varies with different requirements for the same system. These gains are calculated before the system is released for the tests or practical implementation. The table 5.1 shows the effects of the gains on the system. In general, derivative gain is used when system should be settle faster and system overshoot is reduced.

Table 5.1: Effects on response of the system of a PID controller with the respective gains

Response	Rise time	Overshoot	Settling time	Steady state error
k_p	Decrease	Increase	No trend	Decrease
k_i	Decrease	Increase	Increase	Eliminate
k_d	No trend	Decrease	Decrease	No trend

This paper provides a PID controller design based on the natural frequency of the system and damping coefficient of the system. The effectiveness of the control system design depends on the location of poles and zeros on the real-imaginary plot, and also on the gain of the system. To reach all the desired specifications with accuracy and to know the reach of the system we will use the root-locus method and the graphical method to analyse the system stability in the complex S-Plane. O'Brien et. al.[72] explained the concepts root-locus and bode compensator design in detail.

5.3 Control Design

We build the compensator based on the following specifications.

- Percentage overshoot or maximum overshoot $(PO)\%$ for the step input
- Rise time t_r for the step input

With the above transient specifications that our plant should behave we will design a compensator. We know t_r & $(PO)\%$, using (4.4) and (4.2) respectively to find ζ and ω_n with the following steps we can design the compensator. A controller is designed with a desired rise time, overshoot and settling time

Compensator design:

$$S_d = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2} \tag{5.4}$$

$$\phi = \angle G_p(S_d) \tag{5.5}$$

$$\phi_c = -180 - \phi \tag{5.6}$$

Computing poles and zeros of the transient compensator

$$\phi_{zc} = \angle(S_d - Z_c) \tag{5.7}$$

$$\phi_{pc} = \phi_{zc} - \phi_c \tag{5.8}$$

$$P_c = Re(S_d) - \frac{Im(S_d)}{\tan(\phi_{pc})} \tag{5.9}$$

Computing gain of the compensator

$$G_{ct}(S) = K_{ct}\frac{S - Z_c}{S - P_c} \tag{5.10}$$

Result of product of magnitude of G_p and G_{ct} should be equal to one.

$$|G_{ct}(S_d)||G_p(S_d)| = 1 \tag{5.11}$$

$$K_{ct} = \frac{|S_d - P_c|}{|S_d - Z_c||G_p(S_d)|} \tag{5.12}$$

Matching compensator to the PD control

$$G_{pd} = \frac{K_p + SK_d}{\tau S + 1} \tag{5.13}$$

Here to find the K_p and K_d values we are matching lead compensator and PD control equation

$$K_{ct}\frac{S - Z_c}{S - P_c} = \frac{K_p + SK_d}{\tau S + 1} \tag{5.14}$$

After solving the (5.14)

$$\begin{aligned} \tau &= \frac{-1}{P_c} \\ K_p &= \frac{K_{ct}Z_c}{P_c} \\ K_d &= K_{ct}\tau \end{aligned} \tag{5.15}$$

where,

$\tau = time\ constant$

Using the above K_p & K_d values the controller $C(S)$ is designed

The below Fig. 5.2 is the simulink implementation of the system. The output of this contains how the system reaches steady state and be stable based on the calculated proportional, derivative and integral constants in using above method. And the Fig. 5.3 is the PID control box of the whole system.

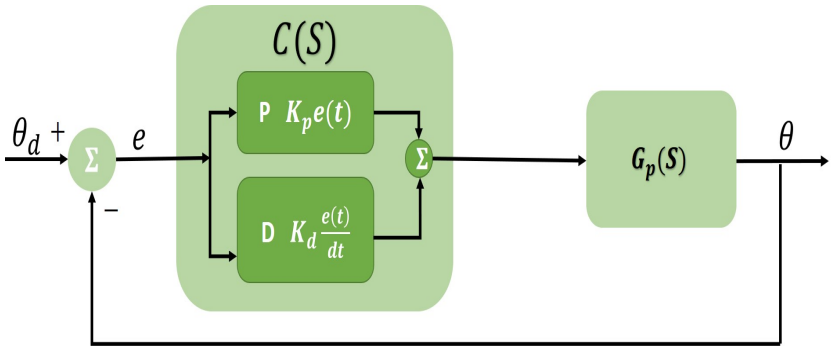


Figure 5.1: Block diagram of closed loop system with PID controller in the loop. This system continuously calculates the error value (e) which is the difference between set value (θ_d) and process variable (θ). For this we are considering integral gain K_i is 0 because the lead compensator and PD gain equations matches.

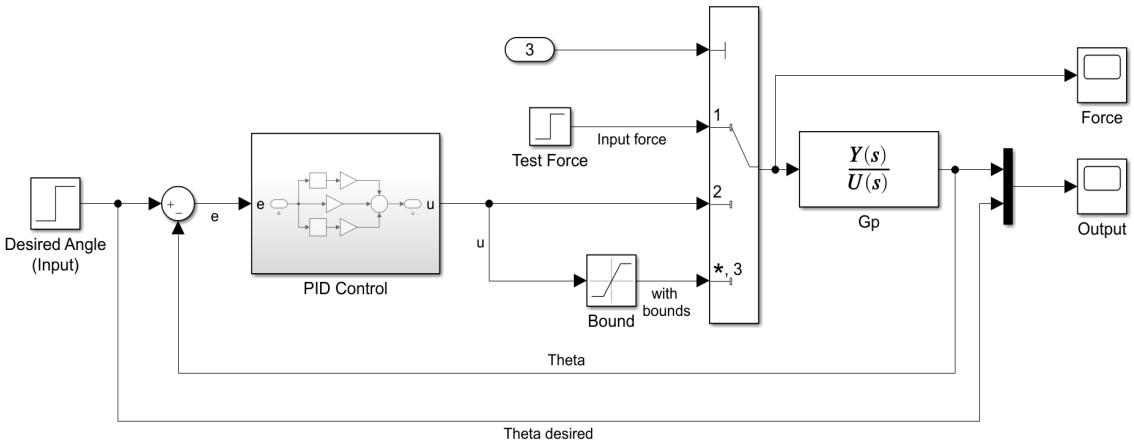


Figure 5.2: Simulink model for the system $G_p(S)$ with PID control system. This functional block is designed to see the simulated response of the whole system using the calculated K_p, K_d gains are used. Fig 7.6 is the plot captured from the simulated output that is generated using the calculated gains.

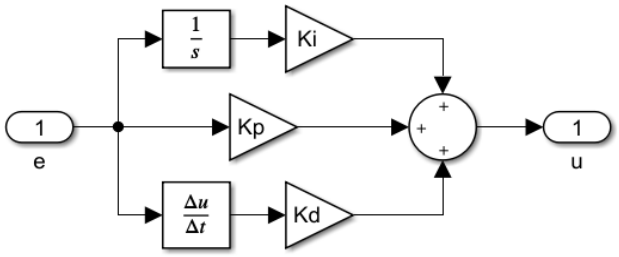


Figure 5.3: Simulink model of the PID control system with the error (e) as the input and control variable (u) as the output. Here error (e) is the difference between set point/desired angle (θ_d) and process variable/current theta (θ). The gains calculated for this block are from the Eq. 5.15

Chapter 6: Robot Component Set-Up

In this chapter, we will go through all the electronic and electrical components that we used to build our robot. Also, results of the component testing that we performed to check whether it is functioning as per the datasheet or the way it should behave. We will also go through the pseudo code, the flow charts of the brains of the robot, designed functional blocks to get a better understanding of the set up.

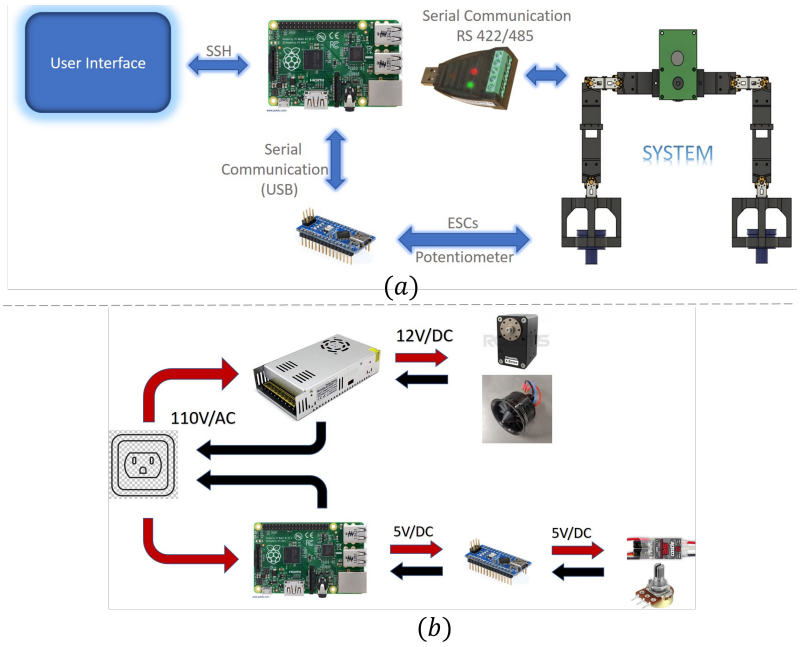


Figure 6.1: Block diagrams (a) communication block diagram (b) power supply block diagram. Communication block diagram provides the details of communication channel between two components. Power supply block diagram provides the details of the input voltage that the components work. We will discuss about each component in this chapter.

6.1 Introduction to Components

The robot contains 6 dynamixels as joint actuators, 2 Electric Ducted Fans (EDF) as thrusters at end effectors and a potentiometer as an angle feedback sensor. Each component should be working fine to generate the expected results. So, to make sure, we performed component testing. In this section we are also going through the list of hardware and software used to make our functional robot.

6.1.1 Electric Ducted Fan

The thrusters used for this system are Electric Ducted Fan (EDF) made by Powerfun. These thrusters are cleverly engineered, the unit is pre-balanced and properly positioned with a machine. The following are the features of this product.

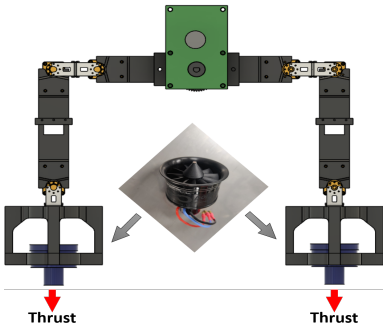


Figure 6.2: Electric Ducted Fan: Used as thrusters. It is a 3 phase brushless DC motor used to control the flight of the robot. It is controlled by Electronic Speed Controller(ESC) for variable thrust using PWM signals. Used 2 EDF's as end effector.

- It has new design and high performance with 50 mm 11 blades.
- Light weight, with overall weight of 77g
- The motor used is high performance 4300 Kv Brush-less Motor.

- With an operating Voltage of 11.1V(3S) - 14.8V(4S).
- While working the static thrust is up to 770 gms (can lift an object of up to 770 gms)

6.1.2 Actuators

We used Dynamixels, XH430-W350-R, as joints of the robotic arm. The Fig. 6.3 is Dynamixel and arrows in the figure indicates the physical location of dynamixels on the robot.

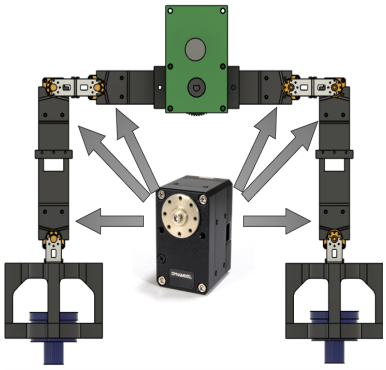


Figure 6.3: Joint Actuators: Used 6 Dynamixel X-series smart actuator as joints. This is used to manipulate the arm posture. Arrows represents the locations of the actuators in the robot. It is controlled by single board computer using RS485 communication system at 1 Mbps baudrate.

We have chosen these smart actuators because of their high performance when they are connected in a network. Following are the features of the Dynamixels.

- ARM CORTEX-M3 (72 [MHz], 32Bit) MCU (Micro Controller Unit)
- Position sensor with 12 Bit (4096 steps for 360°) (0.088°/step)
- Modes of Operation
  - Torque Control
  - Position Control
  - Velocity Control

- PWM Control
- Extended Position Control
- Current based Position Control

Robotis[[66] has developed good GUI interface for Dynamixels in wide range of platforms and developed SDK's in C, C++, Python, LabView, Matlab, JAVA, C#. In a network, communications takes place via RS-485 and TTL links, RS485 is a D+ and D- 2-wire half duplex serial communication type. As they are half duplex, it can be a multi tap system which means all devices can communicate on a single line. This simplifies the communication network but the disadvantage is, it uses more bandwidth. It is an Asynchronous Serial Communication with 8bit, 1stop, No Parity with an adjustable Baudrate up to 4.5Mbps.

6.1.3 Angle feedback Sensor

A potentiometer is used as an angle feedback sensor that is located inside the middle structure attached to a gear system to measure the roll angle of the system. It is an 1kΩ 3/4" single turn potentiometer. A 5 Volts power supply is given to the potentiometer to operate and at approximately 2.5 Volts it is set to give a 0° angle. The data generated by the feedback sensor is read by microcontroller, Arduino, to calculate the control law. Arduino uses 10-bit ADC to convert an analog value to a digital value. To calculate the error sign, we used the right-hand thumb rule to represent the roll angle. When it tilts left side potentiometer output will be less than 2.5 Volts then in Arduino the error value is a negative value and vice versa.

$$\theta = \frac{k - k_r}{k_r - k_{min}}(\theta_{ref} - \theta_{min}) \tag{6.1}$$

where,

k is value recorded at particular position in arduino varies from 0 to 1024

k_r is reference value

k_{min} is the minimum value

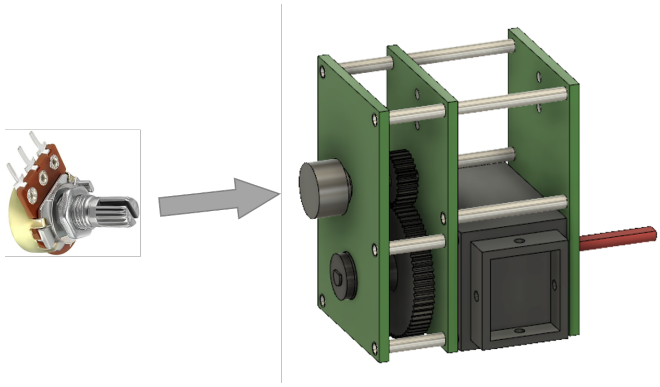


Figure 6.4: Potentiometer: Used as an angle feedback sensor to get the roll angle of the robot. It is placed inside the middle structure with a gear system that translates the rotation of the robot to sensor. It is an analog device connected to Arduino. Data generated is used in control law for desired output curve.

θ_{ref} is theta reference

θ_{min} is the theta minimum

6.1.4 Electronic Speed Control

The Electronic Speed Controllers (ESC's) are used to control the thrusters (EDF). For this system Castle Creations Multi-Rotor 35 ESCs are used. This has a lot of features to list but the following are the main features that made to choose this ESC [73].

- Linear throttle response
- Low latency throttle response ($<75\mu$s)
- Adaptive Timing system for Increased power and efficiency
- Low internal resistance (1.40 mΩ)
- Auto Detecting Servo PWM

Adaptive Timing System dynamically adjusts the motor timing while performing an operation to maintain the efficiency. In general, ESC's have fixed motor timing set by the

user, which means it produces optimized results only for specific RPM and load. Adaptive Timing System makes the motor to operate in peak efficiency at any RPM and load. This system always leads to the increase in power output, flight times and reduces the ESC tuning when compared to general configuration.

A traditional servo PWM signal can also be used to control ESC. The pulse width signal varies from 1.0 ms to 2.0 ms that is between 0% and 100% throttle. The operating frequency of the signal is 50Hz but can be increased upto 490Hz. The throttle curve the motor created is a linear curve with a reduced noise. This significantly shows the improved flight stability, flight times and simplifies flight controller PID tuning.

6.1.5 Single board Computer

Raspberry Pi is a single-board credit-card-sized computer developed in the United Kingdom by the Raspberry Pi Foundations, used in the fields of development, research, and educational institutes in areas like robotics, computer science, computer, and electrical engineering. They have been releasing a series of generations by integrating complexity in different areas such as computational power, protocols, updating hardware to the present world needs, etc.

Algorithm 1: Pseudo code for Single Board Computer 'Raspberry Pi'

```
Result: Change the End effector (wrist joint) position based on Theta θ
initialization;
declaration;
while True do
    if Data Waiting then
        GET theta θ from micro controller;
        Compute Position ;
        SET both the end-effector values;
    end
end
```

We choose Raspberry Pi 3 B+ in this project because of its computational power with higher speeds, RAM of 1 GB to install Robot Operating System for the Dynamixels to

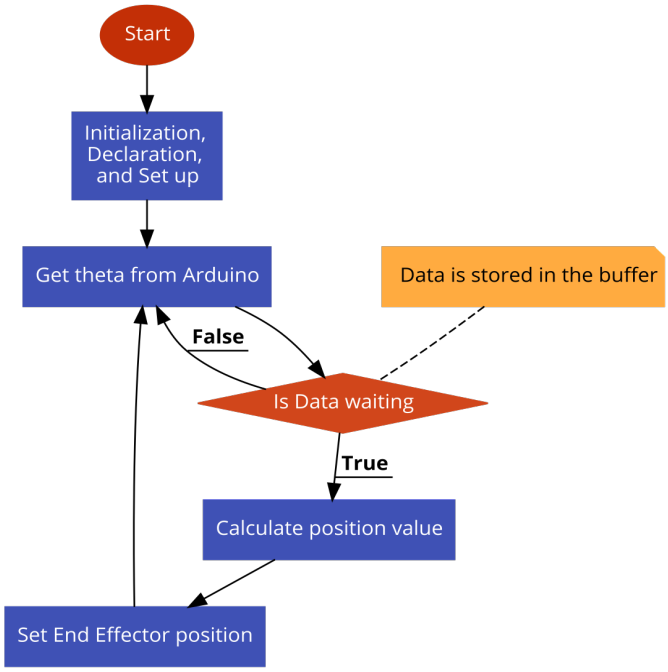


Figure 6.5: Flowchart of a Single Board Computer, Raspberry Pi: It reads data from the Arduino and calculates and commands the actuators' location. Used one serial communication channel to talk with dynamixels at 1Mbps baudrate and another with communicate with Arduino. Simultaneously, all the data is logged into the .txt file.

compute parallel tasks and download DynamixelSDk from ROS community, inbuilt Wi-Fi for connectivity for updates and operating remotely without additional setup with low power consumption. It also has a lot of other peripherals for different types of communication like serial communications, that we have relied more on this project.

Role of Raspberry Pi in this project, it controls the actuators, and receives data from microcontroller(Arduino). We used dynamixelsdk for set up and data transfer over RS485. And received current theta from Arduino over USB. The Fig. 6.5 is the flowchart of the raspberry pi functionalities how it performs the tasks and control the actuators. First it initializes, declares, set up takes place. In this state sets ID to the actuators, enabling serial communication ports, torque for actuators to hold the position and set up min and max limits of actuators as we are using position controlled mode of operation. Then it checks for the data in the serial buffer and wait for the data. On receiving the data it evaluates the

theta and computes the required step value of the actuators. This is to make the thrusters face down all the time. This process is repeated, and data is recorded. We are connected through the command line interface that is SSH interface of raspberry pi for updating the code. we also used GIT a version control tool for reliability.

6.1.6 Micro controller

Arduino is open-source hardware and software which produces single-board micro controllers for developing digital systems like Embedded Systems, IoT systems and also used in educational institutes, research, and development fields for the new addition to the knowledge. Arduino products are licensed under Creative Commons (CC-BY-SA) license and software are under the GNU General public License (GPL).

Algorithm 2: Pseudo code for micro controller 'Arduino NANO'

Result: Calculate control variable and send theta to single board computer
initialization;
declaration;
SET desired theta θ_d;
GET theta θ;
while *error* (θ_d - θ) $<$ *threshold* **do**
  Compute proportional error $e(t)$;
  Compute integral error $\int_0^t e(\tau)d\tau$;
  Compute derivative error $\frac{de}{dt}$;
  Compute control variable $u(t) = k_p e(t) + k_i \int_0^t e(\tau)d\tau + k_d \frac{de}{dt}$;
  if *u(t) > 0* **then**
    $PWM\ value \leftarrow get\_PWM(u(t))$;
    $Left\ Motor \leftarrow PWM\ value$; `// PWM value should be in range`
  else
    $PWM\ value \leftarrow get\_PWM(u(t))$;
    $Right\ Motor \leftarrow PWM\ value$; `// PWM value should be in range`
  end
  GET theta θ;
  SEND theta θ => single board computer;
end

We will see the role of Arduino Nano microcontroller in this project. It is used to read the analog data from angle sensor(potentiometer) and calculates the thrust required to reach the desired roll angle. Arduino has on board ADC to digitalize the analog output of potentiometer and Servo PWM signal generator libraries to control the thrusters.

The Fig. 6.6 is the flowchart of the operations and functionalities. Initialization, Declaration of the data pins, desired theta is the first step then it gets the theta from the sensor, simultaneously theta is sent to raspberry pi to calculate the wrist joint's actuation. Next error value is calculated to check whether the angle is below the threshold or not; if it is lesser than the threshold no action is needed because it reached the required angle but if it is greater than threshold it computes the error. After that error value is sent to the control law i.e., to PID control equations to generate the control variable. Finally, with the resulted control variable $u(t)$ the thrust required is calculated. Here, it also calculates to which arm thrust should be provided and generates the necessary PWM signal value to produce enough thrust. Until it reaches the desired angle this process is repeated.

6.1.7 MATLAB and Simulink

Matrix Laboratory, in short, is called MATLAB, which is a multi-paradigm programming language, which means it is built on more than one programming styles such as procedure-oriented, object-oriented, function-oriented, etc. and also a proprietary programming language developed MathWorks Corporation. On the other hand, Simulink is also a MathWorks product based on the graphical-programming environment that can be used for modeling, simulating, analyzing a dynamical system from multiple domains. MathWorks provides multiple toolboxes such as Control Systems Toolbox, System Identification Toolbox, Predictive Maintenance, etc..

In this project, we are using MATLAB and Simulink for modeling the system with the available data then simulating the system by adding a control system and analyzing the results, understanding the system behavior, testing with multiple scenarios.

6.2 Communication

Communication among devices is significant to transfer the valuable data they contain to serve the whole system on the right path and desired results. There is a wide range of protocol stack to serve this purpose depending on the necessity and in which layer the communication is taking place. The Fig. 6.7 below shows the communication layout for the system, the on-board computer, Raspberry Pi 3B+, used to take commands from the user with a different OS through Secured Shell (SSH) scripting.

Raspberry Pi is installed with Raspbian-Jessie operating system and Robot Operating System (ROS) to get the dynamixelSDK running. To communicate with dynamixels we used RS 485 based serial communication using 4P connectors. And also connected to the Arduino nano board to operate the ducted fans that control the roll motion of the system based on the input sensor, Potentiometer, placed at the center of the system on which the whole system hanging on. Started a serial communication through USB port between Arduino and Raspberry pi.

6.3 Power Supply

Powering the system is also an important part of the designing phase, there should not be any device which is provided with less than what it needs and over than what it is capable of else situation will be out of control. To power the joints and thrusters we used Switched-Mode Power Supply (SMPS) to convert the 110V AC voltage from the regular power socket to 12V DC. Also powered Raspberry pi with sufficient power that can run with and provide also to Arduino Nano. The rest of the components are provided with 5V to run such as ESC's, potentiometer.

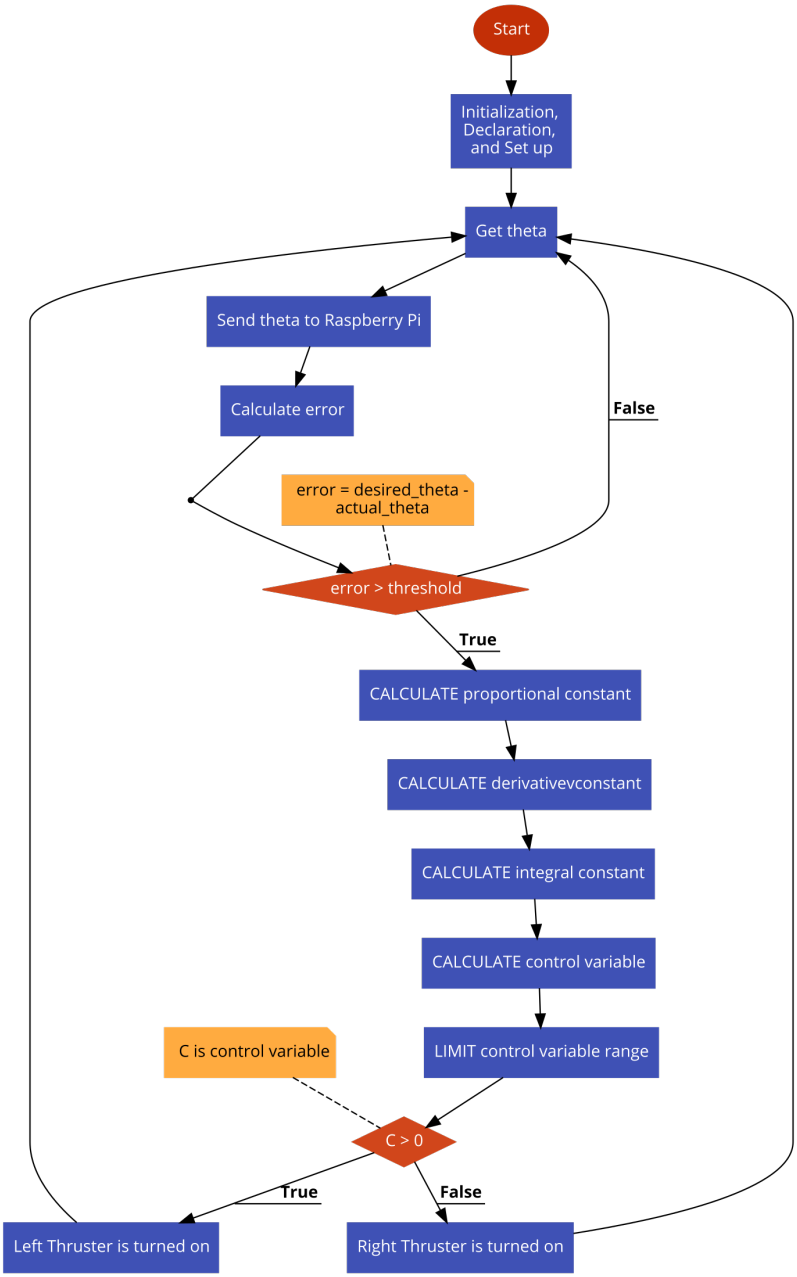


Figure 6.6: Flowchart diagram represents the operations and functionalities performed by arduino microcontroller. It gets input from the position sensor, calculates the PID control variable and provides the appropriate PWM signal to generate enough thrust to maintain design angle.

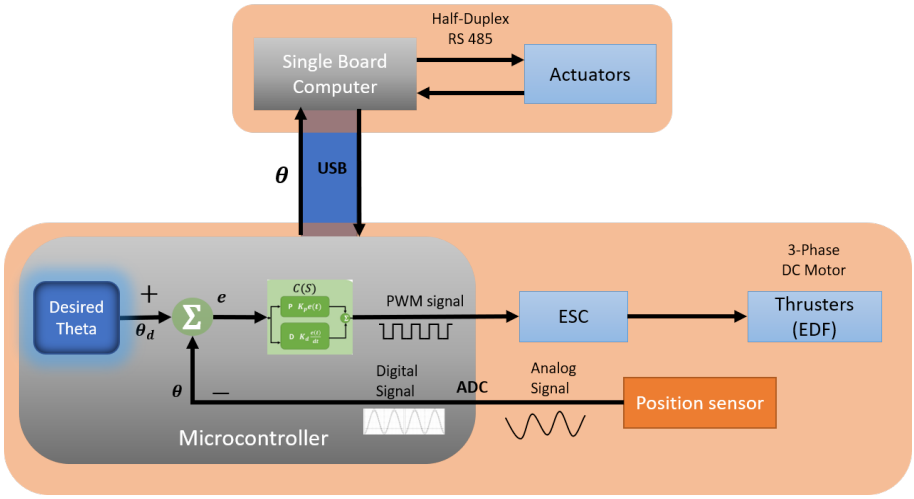


Figure 6.7: Overall block diagram of the system. Single board computer controls joint actuators through RS485 communication system. Micro controller gets the input from the angle sensor then calculates the appropriate thrust to be produced using some control laws. Single Board Computer and Micro controller communicates using serial port to share the current angle used to manipulate end effectors, wrist joints, to face down all the time.

Chapter 7: Experiments

In this chapter, we will see the results of the robot's response at different positions with different inputs. We considered four arm positions and recorded data at each position. Fig. 7.1 represents the natural responses, without any input u, that are recorded for the respective position. Though we are not using this data for our experiments, they have clear responses of how the system behaves. We compared the robot's response with a simple pendulum response and assumed the system is a second order system. In the later section we will see the experiments with an input u.

The first position: the shoulder joints are at 90° angle facing down and all the corresponding elbow and wrist are also following the same that is normal human attention position. In the Fig. 7.1 top - left plot represents the response for this position.

The second position: the shoulder joints and elbow joints are wide open i.e., parallel to the ground plane and wrist joints are facing ground for the thrust. In this position, the system is at maximum length open out widely. In the Fig. 7.1 top - right plot represents the response for this position.

The third position: the shoulder joints are horizontal and the elbow joints are at 90° angle facing down and the wrist joints are also following downwards. In this position, the system will be in the medium range not too wide and not too short. In the Fig. 7.1 bottom - left represents the response for this position.

The fourth position: the shoulder joints and the elbow joints are at 45° angle that results in facing down the wrist joint. In the Fig. 7.1 bottom - right represents the response for this position.

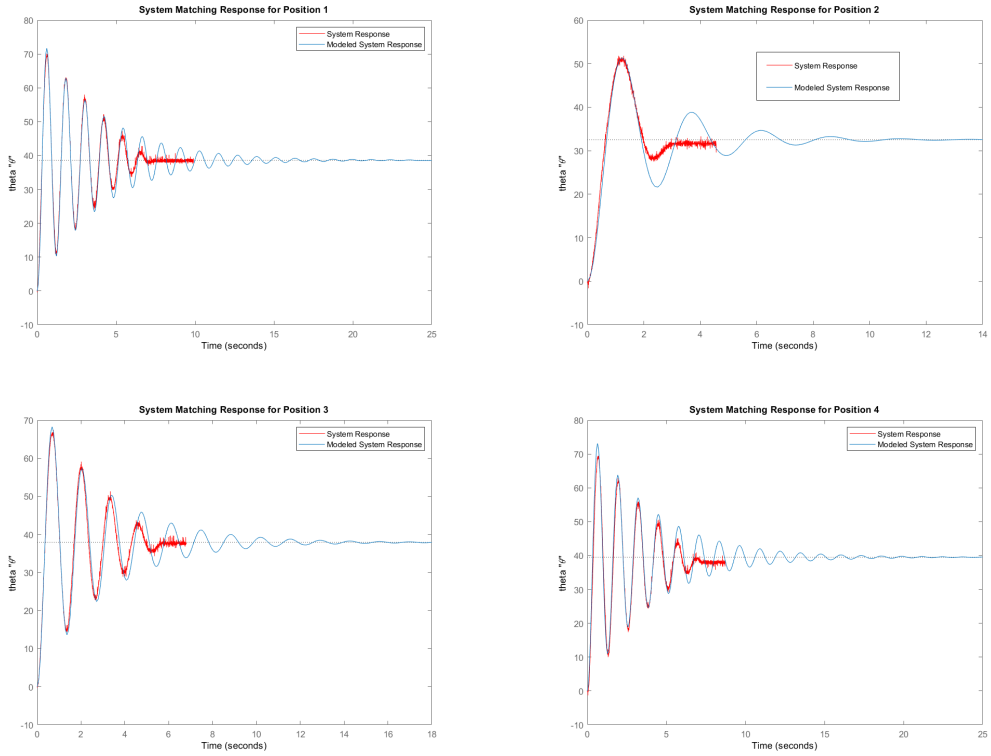


Figure 7.1: System' natural responses at four different positions. First picture (Top-Left) is the natural response data and matched system response of first position. Second picture (Top-Right) is the natural response data and matched system response of second position. Third picture (Bottom-Left) is the natural response data and matched system response of third position. Fourth picture (Bottom-Right) is the natural response data and matched system response of fourth position.

7.1 Results - Simulation Experiments

In this section, we will discuss the experiment in detail. We recorded the system response when input is constant from a thruster. Here, for this experiment, input u i.e., the thrust produced was 280 grams at 50% duty cycle (The thruster didn't produce enough thrust as mentioned in the specifications, it should produce around 335 grams). The system response for this input u at position 1 is shown in the Fig. 7.2

Manually calculated the rise time, peak overshoot and settling time from the generated system response in the Fig. 7.2. With these values from transient specifications equations

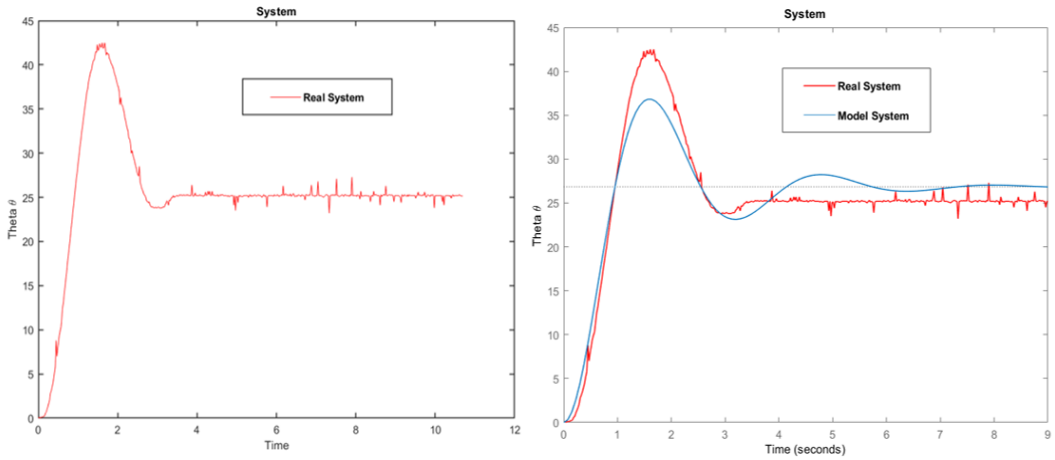


Figure 7.2: (Left) Represents the system response at 50% duty cycle to ESC as a step input that produces 280 grams of thrust, (Right) Step response of the calculated model system $G_p(S)$ that matches system response on the left. As the data from the feedback sensor has more noise for this experiment we added a basic LPF circuit(RC circuit)

with the calculated natural frequency ω_n and damping coefficient ζ, approximated and calculated second order transfer function of the system using the gain term and manual tuning of the ω_n and ζ. The Fig. 7.2 also represents the step response plot with the matched system that is the step response of the calculated second order equation.

Following plant Equation 7.1 is evaluated using calculated gain (K), natural frequency (ω_n) and damping coefficient (ζ) as 435, 2.0662 and 0.3000 respectively.

$$G_p(S) = \frac{435}{S^2 + 1.24S + 4.269} \tag{7.1}$$

The above table 7.1 is the step function info of the modeled system or the second order transfer function. This table has the information of the system response specifications, like rise time, settling time, percent overshoot etc. these specifications are modified by adding a pole and a zero to the system we will look into that in detail later. The Fig. 7.3 is the pole zero plot of the model system or second order transfer function. It has 2 poles on the left side of the real axis and they are symmetrical to real axis. If the poles are on left side of real axis the system stable, if the poles are near to imaginary axis it is over damped and

Table 7.1: Step Info of the second order model system that is equivalent to the real system

Step info
RiseTime : 0.6406
SettlingTime : 5.4352
SettlingMin : 23.1272
SettlingMax : 36.8207
Overshoot : 37.1410
Undershoot : 0
Peak : 36.8207
PeakTime : 1.5602

if they are symmetrical and far from the real axis it takes more time to settle. But this is not the response we are expecting for a system. So, next we want to create a compensator that adds a pole and a zero on the left half of real axis so that system will be stable and it should have quick rise time, less settling time and less percent overshoot.

The following steps are to design a lead compensator with the desired rise time as 3 seconds and the percentage overshoot of 10%. Using transient specifications Equations 4.2 4.3 4.4 & 4.5, any two out of four that can provide results of ζ, damping coefficient, and ω_n, resonant frequency. From the steps in the chapter 5 we compute poles of the desired system. From the calculated ζ and ω_n, using Equation 5.4 the poles of the desired are -5.4494 $\pm$ 5.7147i. From these poles we find the compensator phase from Equation 5.6. Finally we will calculate the gain of the compensator, a pole and a zero of the compensator for the model system $G_p(S)$ to reach the desired transient responses for our current system model. Computing the steps for the compensator design, section 5.3, the control design section. The final lead compensator $G_{ct}(S)$ equation is given as follows with the gain of the compensator (K_{ct}), a pole P_c and a zero Z_c of the compensator as 1.298, 63.34 and 5.449 respectively.

$$G_{ct}(S) \;=\; 1.298 \left(\frac{S \;+\; 5.449}{S \;+\; 63.34} \right) \tag{7.2}$$

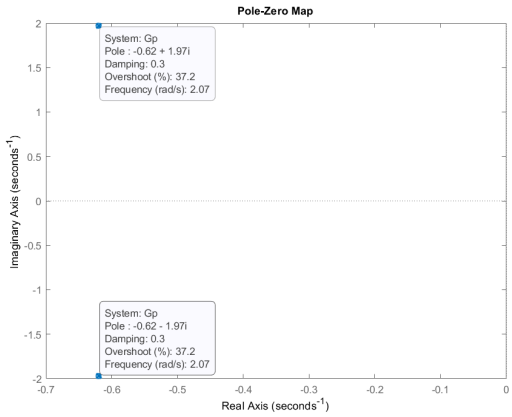


Figure 7.3: Pole-Zero plot of the plant $G_p(S)$. This gives an idea of poles and zero location and can see the system is stable or not. The system is a stable system as the poles lie in the left side of the real axis. Fig. 7.4 represents an added pole and a zero to the system to have the desired rise time and overshoot.

Our final goal is to find the PD gains for the PD controller. Here, we know the equation of PD controller $G_{pd}(S)$ mentioned in the section 5.3. We designed the lead lag compensator $G_{ct}(S)$ in such a way that it matches the PD controller $G_{pd}(S)$ to find the PD gains. Here, we compare both the equations $G_{ct}(S)$ and $G_{pd}(S)$ to find the gain constants k_p, k_d and the time period τ. In this PID controller K_i is 0.

$$G_{pd}(S) \;=\; \left(\frac{0.01687\ S \;+\; 0.09191}{0.01299\ S \;+\; 1} \right) \tag{7.3}$$

After evaluating Equation 7.2 and Equation 5.13 the PD gain values are,

$k_p = 0.0919$,

$k_d = 0.0169$,

$\tau = 0.0130$

The above table 7.2 is the step function info of the system with implemented PD control to the modeled system. We can see there is an improvement in the rise time, settling time, overshoot, peak time.

Table 7.2: Step Info of the system with PID controller

Step info
RiseTime: 0.1272
SettlingTime: 0.6958
SettlingMin: 0.8351
SettlingMax: 1.1200
Overshoot: 23.9642
Undershoot: 0
Peak: 1.1200
PeakTime: 0.3425

After implementing the designed PID control to the system with integral part as zero, the pole zero plot is changed as shown in the Fig. 7.4. We can see the results with an extra pole and a zero were introduced to make the system more stable and improve the settling time, percentage overshoot.

The expected behavior of the system after implementing the PID controller to the present system with the evaluated gain constants with a unit step function as input should be as the Fig. 7.5. There produces some steady state error because its a PD control that does not eliminate steady state error.

Fig. 7.6 is the output of the plant with PD control implemented in simulation using Simulink software. From the plant model 5.2, a step input of, desired theta θ_d, 15° is given as an input to the system that is at 0°. Initially, at time $t = 0$, there is a lot of error between the θ and θ_d system produces enough thrust to put the arms at desired value θ_d. The output in fig. 7.6 is the response of the system. It is satisfying with the conditions and met the requirements. It also has the steady state error because the PD control does not eliminate steady state error for the step input.

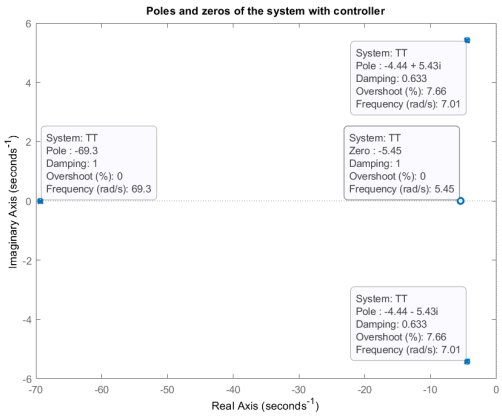


Figure 7.4: Pole-Zero plot of the system $G_p(S)$ with PID controller $C(S)$ that is designed with the desired transient specifications. These values are derived using the equations in the Chapter 5: Control System. An extra pole and a zero are added to make the system more stable and improve the settling time, percentage overshoot

7.2 Results - Hardware in the Loop Experiment

When the calculated PID gain constants are introduced to the real system, Fig. 7.7 shows the results of the theta (θ) when desired theta θ_d is 15°. The results show the robot took about 8 seconds to reach a steady state with less rise time and no overshoot. The data read from the sensor is too much noisy.

As there is much noise in the output shown in the Fig. 7.7, that's due to the long cables travelling from the potentiometer to the arduino between power cables and other 4P wires that contains information to the Dynamixels and brushless DC motors. Introducing a moving median filter to that output gives the better result without the heavy spikes in the plot. The following Fig. 7.8 contains the results of filtered output with moving median filter.

An high level definition of Moving-median filter, it is a non-linear digital filter, mostly used to reduce the noise from a signal. For the plot we generated we only want to remove the spike of the noise, this will be better solution. The output of the plot after using the Moving-median filter is shown in Fig. 7.8. We used the "smoothdata" function to smooth

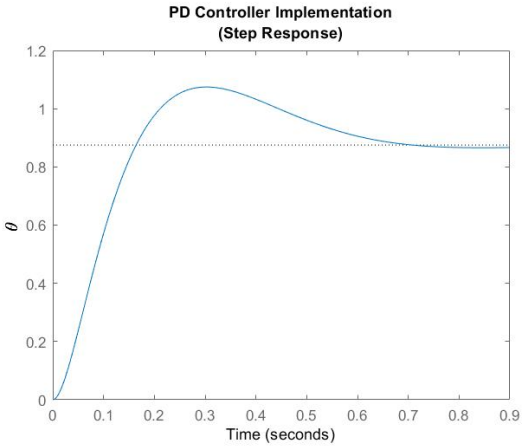


Figure 7.5: Expected system response with PID control (integral gain $K_i = 0$ in this case). The plot represents the system has very less rise time, less percentage overshoot and settled fast. We can also see steady state error for the system because PD controller don't eliminate the error.

out the noisy data[74] in MatLab. In smoothdata function, we have a number of filters like Gaussian, moving-median, Savitzky-Golay, Linear regression, Robust linear regression, etc. When the window size is not specified in this function, smoothdata, computes a default window size based on a heuristic.

There are few take away points here: the output on the implemented results has more steady state error than the simulation results, the system has very low rise time and percent overshoot is not so high as resulted in the simulation. This is because of the few assumptions made initially, from the system response it is clear that the system is not the second order system but we assumed it as a second order system. Lack of integral error produces more steady state error. For more accurate results: the integral gain can be introduced, designing the system model with higher order transfer function.

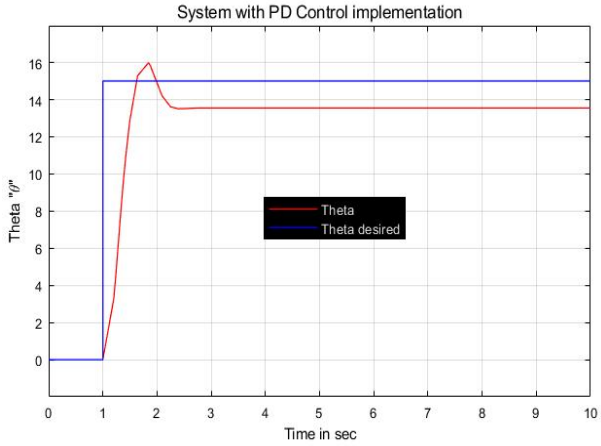


Figure 7.6: The plot represents the simulation result of the plant $G_p(S)$ with calculated PID gains using equations in the chapter *Control Design*. The input, desired angle θ_d is set to 15° (Blue line). The output, system response is shown in red line. As this controller is the PD controller we expect small steady state error.

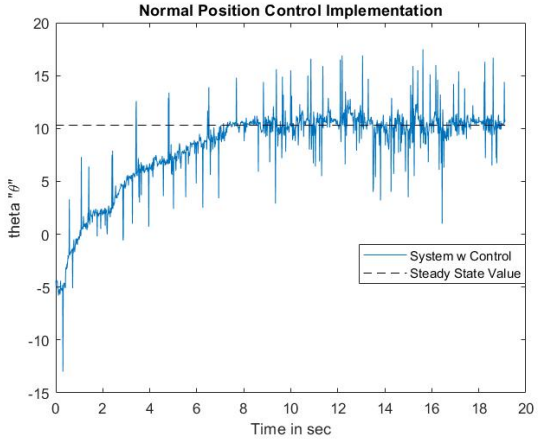


Figure 7.7: Plot shows the system's response when implemented using the calculated PID gains on the real robot for the same desired angle θ_d is 15°used in simulation, Fig. 7.6. This signal has a lot of noise because of the long cables and the magnetic interference produced by high speed EDF. The plot shows it reached steady state about 8 secs after the test and less rise time.

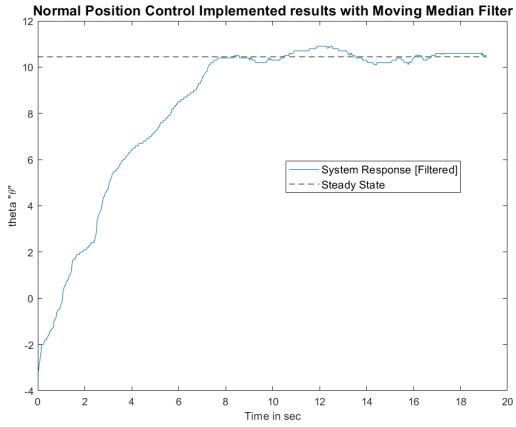


Figure 7.8: System response after passing the noisy output of Fig. 7.7 through the moving median filter. This filter is a 'smoothdata' function from Matlab software to analyze the clean data. The steady state error is more compared to simulation results. The transient responses of desired output are not met on the real robot.

Chapter 8: Conclusion

8.1 Summary

This paper presented how we developed an extra mobility option, gives the advantage of operating the robot in multiple modes. With many options for locomotion, a method with thrusters on humanoids' upper body in the coronal plane, helps the robot commute faster in difficult terrains and operate efficiently in specific research applications. Overall, the skeleton of the system was designed using CAD software, built using additive and subtractive methods. The robot is hanged to a support system it was built using CAM and CNC machines in the DASL (Drones and Autonomous Systems Lab) workshop at GMU. Built the robot with the following assumptions: the system is a second-order system, all the experiments are performed in the laboratory setup. The equations are developed using systems identifications, PID controller and lead-lag compensator concepts. As the system's equation is 4.1 the fundamental variables in developing the system equation are natural frequency ω_n and damping coefficient ζ. Developed a PID controller to control the overshoot, rise time, and steady-state for this system. To ensure the system is performing we did component testing, unit testing for the hardware and software and also integration testing between actuators, sensors, arduino and raspberry pi. Calculating the pole-zero locations to know the stability information of the system and compared with different scenarios.

With the completed simulation model and real robot we performed tests, results are as follows, the simulated results show there is an overshoot of 30% with a rise time of 0.12 secs and reaches the steady-state within 1.12 secs. On the real robot the tests show that it took 8 seconds to reach the steady-state with no peak overshoot. The following reasons might be the reasons in the noisy system: the long wires that carry analog and digital data, noise

due to power cables around signal wires. The reason for the non responsive system is that the gear system developed has more friction to operate on the coronal plane as the gears were made of plastic (PLA material).

8.2 Suggestions for future work

From the observations of the test results on the robot, the system is not 100% identical for designing more accurate system, considering the system to be a MIMO (Multi Input Multi Output) system as the machine has two thrust vectors. To take the complexity of the robot to next level we can extend the dynamics of the system to the sagittal and transverse planes with 6 DOF actuation for the system, which helps in traversing in 3 dimensional space. For such a complex system advanced algorithms, control methods such as altitude and attitude control laws need to be implemented to the system. A better arm design: that provides space to multiple actuators, gives more flexibility to the joints and more actuation freedom can be built with sophisticated design using CAD software. With the complex arm design the complexity of the actuation also increases, we can use inverse kinematics concept manipulate the end effector location.

Appendix A: Parts used in the project

Table A.1: Parts List

Part	Name	Model
Single Board Computer	Raspberry Pi	3 B+
Micro Controller	Arduino	Nano
Joint Actuators	Dynamixel	XH430-W350-R
Thruster	Powerfun	50mm EDF with 4300KV/4S
Sensor	Potentiometer	WH148
ESC	Castle	Multi-Rotor35
Actuator Cable	Cable	4P Cable

Appendix B: Code

B.1 Single Board Computer (Raspberry Pi) code

This code used to control the joint actuators (Dynamixels) over RS485 links through serial communication to set their position based on the the serial data received from micro controller (Arduino NANO). Simultaneously it records the necessary data in the .txt file in the local directory.

```
from __future__ import print_function
import os
import ctypes
import time
import serial

if os.name == 'nt':
    import msvcrt
    def getch():
        return msvcrt.getch().decode()
else:
    import termios, fcntl, sys, os
    from select import select
    fd = sys.stdin.fileno()
    old_term = termios.tcgetattr(fd)
    new_term = termios.tcgetattr(fd)

    def getch():
        new_term[3] = (new_term[3] & ~termios.ICANON & ~
           termios.ECHO)
        termios.tcsetattr(fd, termios.TCSANOW, new_term)
        try:
            ch = sys.stdin.read(1)
        finally:
            termios.tcsetattr(fd, termios.TCSADRAIN, old_term)
        return ch

    def kbhit():
        new_term[3] = (new_term[3] & ~(termios.ICANON |
           termios.ECHO))
        termios.tcsetattr(fd, termios.TCSANOW, new_term)
        try:
            dr,dw,de = select([sys.stdin], [], [], 0)
            if dr != []:
                return 1
        finally:
```

```
            termios.tcsetattr(fd, termios.TCSADRAIN, old_term)
            sys.stdout.flush()
        return 0

from dynamixel_sdk import *         # Uses Dynamixel SDK
    library

# Control table address
ADDR_OPERATING_MODE         = 11    # Control table address is
     different in Dynamixel model
ADDR_TORQUE_ENABLE          = 64
ADDR_GOAL_POSITION          = 116
ADDR_PRESENT_POSITION       = 132

# Protocol version
PROTOCOL_VERSION            = 2.0   # See which protocol
    version is used in the Dynamixel

# Default setting
DXL1_ID                     = 91    # theta 1
DXL2_ID                             = 92    # theta 2
DXL3_ID                             = 93    # theta 3
DXL4_ID                             = 94    # theta 4
DXL5_ID                             = 95    # theta 5
DXL6_ID                             = 96    # theta 6
BAUDRATE                    = 1000000 # Dynamixel default
    baudrate : 1000000
DEVICENAME                  = '/dev/ttyUSB0' # Check which
    port is being used on your controller

TORQUE_ENABLE               = 1     # Value for enabling the
    torque
TORQUE_DISABLE              = 0     # Value for disabling the
    torque
DXL_MOVING_STATUS_THRESHOLD = 10    # Dynamixel will rotate
    between this value

ESC_ASCII_VALUE             = 0x1b
SPACE_ASCII_VALUE           = 0x20

#index = 0
dxl1_goal_position          = 2560  # Goal position
dxl2_goal_position          = 1536  # Goal position
dxl3_goal_position          = 1536  # Goal position
dxl4_goal_position          = 2560  # Goal position
dxl5_goal_position          = 2048  # Goal position
dxl6_goal_position          = 2048  # Goal position

THETA_MAX, THETA_MIN        = 45, -45
dxl_MIN, dxl_MAX            = 1024, 3072

# Initialize PortHandler instance  # Set the port path
```

```
# Get methods and members of PortHandlerLinux or
   PortHandlerWindows
portHandler = PortHandler(DEVICENAME)

# Initialize PacketHandler instance  # Set the protocol
   version DXL1_ID
# Get methods and members of ProtocolPacketHandler or
   Protocol2PacketHandler
packetHandler = PacketHandler(PROTOCOL_VERSION)

# Open port
if portHandler.openPort():
    print("Succeeded␣to␣open␣the␣port")
else:
    print("Failed␣to␣open␣the␣port")
    print("Press␣any␣key␣to␣terminate...")
    getch()
    quit()

# Set port baudrate
if portHandler.setBaudRate(BAUDRATE):
    print("Succeeded␣to␣change␣the␣baudrate")
else:
    print("Failed␣to␣change␣the␣baudrate")
    print("Press␣any␣key␣to␣terminate...")
    getch()
    quit()

# Function to enable DXL Torque
def enableTorque(DXL_ID):
    dxl_comm_result, dxl_error = packetHandler.write1ByteTxRx(
       portHandler, DXL_ID, ADDR_TORQUE_ENABLE, TORQUE_ENABLE)
    if dxl_comm_result != COMM_SUCCESS:
        print("%s" % packetHandler.getTxRxResult(
           dxl_comm_result))
    elif dxl_error != 0:
        print("%s" % packetHandler.getRxPacketError(dxl_error)
           )
    else: print(DXL_ID,"has␣been␣successfully␣connected")

# Calling the function to enable torque on all DXL
enableTorque(DXL1_ID)
enableTorque(DXL2_ID)
enableTorque(DXL3_ID)
enableTorque(DXL4_ID)
enableTorque(DXL5_ID)
enableTorque(DXL6_ID)

## Setting Static goal position for DXL 1,2,3,4,5,6
# Function to Write goal position
```

```
def setGoalPosition(DXL_ID,dxl_goal_position):
    dxl_comm_result, dxl_error = packetHandler.write4ByteTxRx(
       portHandler, DXL_ID, ADDR_GOAL_POSITION,
       dxl_goal_position)
    if dxl_comm_result != COMM_SUCCESS:
        print("%s" % packetHandler.getTxRxResult(
           dxl_comm_result))
    elif dxl_error != 0:
        print("%s" % packetHandler.getRxPacketError(dxl_error)
           )

# Calling the function to set the initial position for DXL
setGoalPosition(DXL1_ID,dxl1_goal_position)
setGoalPosition(DXL2_ID,dxl2_goal_position)
setGoalPosition(DXL3_ID,dxl3_goal_position)
setGoalPosition(DXL4_ID,dxl4_goal_position)
setGoalPosition(DXL5_ID,dxl5_goal_position)
setGoalPosition(DXL6_ID,dxl6_goal_position)

## Getting the current position for DXL
# Function to read the current position

def getPosition(DXL_ID):
    # Read present position DXL1
    dxl_present_position, dxl_comm_result, dxl_error =
       packetHandler.read4ByteTxRx(portHandler, DXL_ID,
       ADDR_PRESENT_POSITION)
    if dxl_comm_result != COMM_SUCCESS:
        print("%s" % packetHandler.getTxRxResult(
           dxl_comm_result))
    elif dxl_error != 0:
        print("%s" % packetHandler.getRxPacketError(dxl_error)
           )
    return dxl_present_position

# Calling the function to read the current position of the DXL
dxl1_present_position = getPosition(DXL1_ID)
dxl2_present_position = getPosition(DXL2_ID)
dxl3_present_position = getPosition(DXL3_ID)
dxl4_present_position = getPosition(DXL4_ID)
dxl5_present_position = getPosition(DXL5_ID)
dxl6_present_position = getPosition(DXL6_ID)

print("␣␣[ID:%03d]␣:␣%03d,␣[ID:%03d]␣:␣%03d,␣[ID:%03d]␣:␣%03d,
   ␣[ID:%03d]␣:␣%03d,␣[ID:%03d]␣:␣%03d,␣[ID:%03d]␣:␣%03d" % (
   DXL1_ID, dxl1_present_position, \
    DXL2_ID, dxl2_present_position,DXL3_ID,
       dxl3_present_position, DXL4_ID, dxl4_present_position,
       DXL5_ID, dxl5_present_position, DXL6_ID,
       dxl6_present_position))
```

```
# Initializing port settings for Arduino
port = "/dev/ttyUSB1"
ser = serial.Serial(port,115200)
ser.flushInput()

while 1:    # Getting data from Arduino and record to the *.
   txt file
        with open('Data.txt', mode='a') as data:
            if ser.inWaiting()>0:       # Reading serial input
                theta = float(ser.readline().strip())
                print(theta)
                endtime = time.time()
                data.write("%f, %f\n" % (endtime,theta))    #
                   Write data to a txt file
            if kbhit():
                c = getch()
                if c == chr(ESC_ASCII_VALUE):
                    print("STOPPED!!!!")
                    break

# Function to Disable Dynamixel Torque DXL
def disableTorque(DXL_ID):
    dxl_comm_result, dxl_error = packetHandler.write1ByteTxRx(
       portHandler, DXL_ID, ADDR_TORQUE_ENABLE, TORQUE_DISABLE
       )

    if dxl_comm_result != COMM_SUCCESS:
        print("%s" % packetHandler.getTxRxResult(
           dxl_comm_result))
    elif dxl_error != 0:
        print("%s" % packetHandler.getRxPacketError(dxl_error)
           )
    else: print(DXL_ID,"has been successfully disconnected")

disableTorque(DXL1_ID)
disableTorque(DXL2_ID)
disableTorque(DXL3_ID)
disableTorque(DXL4_ID)
disableTorque(DXL5_ID)
disableTorque(DXL6_ID)

# Close port
portHandler.closePort()
```

B.2 Micro controller (Arduino NANO) code

Micro controller gets the input from the sensor (potentiometer) and the desired value given in the program with those as system inputs it calculated the PID control value, with that control value we calculate the amount of thrust to be generated to each motor to reach the desired roll angle. It also updates the raspberry pi the current angle using serial communication.

```
#include <Servo.h>

// Desired Values
double theta_D = 15.0;          // angle in degree
int timer = 1000;

const int mot[] = {5,10}; // Initializing motor pins
const int mot_L = 0;      // mot[mot_L] is to control left
   motor. (yellow wire)
const int mot_R = 1;      // similarly for right motor (white
   wire)

const double mot_PWM_offset[] = {60,60}; // Motor offset value
   .

Servo left;               // Creating 2 servos left and right
Servo right;
const double servoMin = 0.0;  // servo PWM min value
const double servoMax = 100.0; // servo PWM max value

int potentiometerPin = A1;
const int potRefVal = 550;  // Refereance Value
const int potMinVal = 250;  // Left most value -45 degree
const int potMaxVal = 850;  // Right most value +45 degree

// Gain constants
const double kp = 0.0919;
const double ki = 0.0;
const double kd = 0.0169;
double p_err = 0.0;
double i_err = 0.0;
double d_err = 0.0;
double prev_error = 0.0;
int const dt = 1000;

double getAngle() {
  int potentiometerValue = analogRead(potentiometerPin);
  double degree = enc2deg(potentiometerValue);
  return degree;
```

```
}

double enc2deg(double enc){
  double deg = (enc - potRefVal)/(potRefVal - potMinVal)*30;
  return deg;
}

int val2PWM(double val){
  double thrustInPWM = val * (servoMax - servoMin) + servoMin
     ;
  if( thrustInPWM > servoMax ) thrustInPWM = servoMax;
  if( thrustInPWM < servoMin ) thrustInPWM = servoMin;
  return (int)thrustInPWM;
}
int setMotor(double val){
  double absVal = abs(val);
  val = min(val, 1.0);
  val = max(val,-1.0);
  if (val > 0.0){
    setMotorThrust(mot[mot_L],absVal);
    setMotorThrust(mot[mot_R],servoMin);
    return 0;
  }
   else{
    setMotorThrust(mot[mot_R],absVal);
    setMotorThrust(mot[mot_L],servoMin);
    return 0;
  }
  return 1;
}

int setMotorThrust(int servo, double err){
  if (servo == mot[mot_L]){
    int thrust = val2PWM(err);
    thrust = min(thrust , servoMax +  mot_PWM_offset[mot_L]);
    thrust = max(thrust + mot_PWM_offset[mot_L] ,
       mot_PWM_offset[mot_L]);
    left.write(thrust);
    return 0;
  }
  else{          // (servo == mot[mot_R]){
    int thrust = val2PWM(err);
    thrust = min(thrust , servoMax +  mot_PWM_offset[mot_R]);
    thrust = max(thrust + mot_PWM_offset[mot_R] ,
       mot_PWM_offset[mot_R]);
    right.write(thrust);
    return 0;
  }
  return 1;
```

```
}

int startUp(){
  left.write(45);   // Left Motor start up value
  right.write(45);  // Right Motor start up value
  delay(2000);
  return 0;
}

void setup() {
  Serial.begin(115200);
  pinMode(mot[mot_L],OUTPUT);
  pinMode(mot[mot_R],OUTPUT);
  left.attach(mot[mot_L]);
  right.attach(mot[mot_R]);
  startUp();
  left.write(65);
  right.write(65);
  delay(1000);
}

void loop() {
  double theta = getAngle();
  double error = theta_D - theta;
  i_err += (dt/1000)*error;
  d_err = (error-prev_error)/(dt/1000);
  double c = kp*error + ki*i_err + kd*d_err;
  prev_error = error;

  // Adding threshold
  c = max(min(c,0.411),0.0);
  setMotor(c);
  Serial.print(c);
  Serial.print("\t");
  Serial.println(theta);
  /*
  // Code for oscillating between +20 and -20 degrees
  Serial.print(' ');
  //Serial.println(c); //Control Value
  if ( timer < 0 ){
    timer = 1000;
    theta_D = -theta_D;
  }
  timer--;
 */
  delay(10);
}
```

B.3 Matlab code

B.3.1 Systems Identification equations

We used MATLAB for some mathematical computation in calculating the Systems Identification Equations

```
syms a;
%% Peak overshoot def
eq = 100*exp((-a*pi)/sqrt(1-a^2)) == 25;
ep = eval(solve(eq));
zeta = ep(1,1)
%% peak time def
syms b;
eq2 = pi/(b*sqrt(1-zeta^2)) == 1.363;
wn = eval(solve(eq2))

%% rise time def
syms c;
%eps = 1.504;
%wn = 2.6424;
eq3 = (pi-atan(sqrt(1 - zeta^2)/zeta))/(wn*sqrt(1-zeta^2))==c;
eval(solve(eq3))

%% settling time
syms a,b;
eq4 = 4/(wn*zeta) == b;
eval(solve(eq4))

%% Closed loop poles
zeta = 0.0709;
wn = 0.4511;
Sd1 = (-zeta*wn + i*wn*sqrt(1-zeta^2))
Sd2 = (-zeta*wn - i*wn*sqrt(1-zeta^2))

%% Calculating Phase
% Current phase phi = Gp(Sd)

phi = phase(evalfr(Gp, sd))

%% system
zeta =  0.2660 ;%- 3.0727i;
wn =  1.7400;

sys = tf([120],[1 2*zeta*wn wn^2])
step(sys)
[z,gain] = zero(sys)
stepinfo(sys)
```

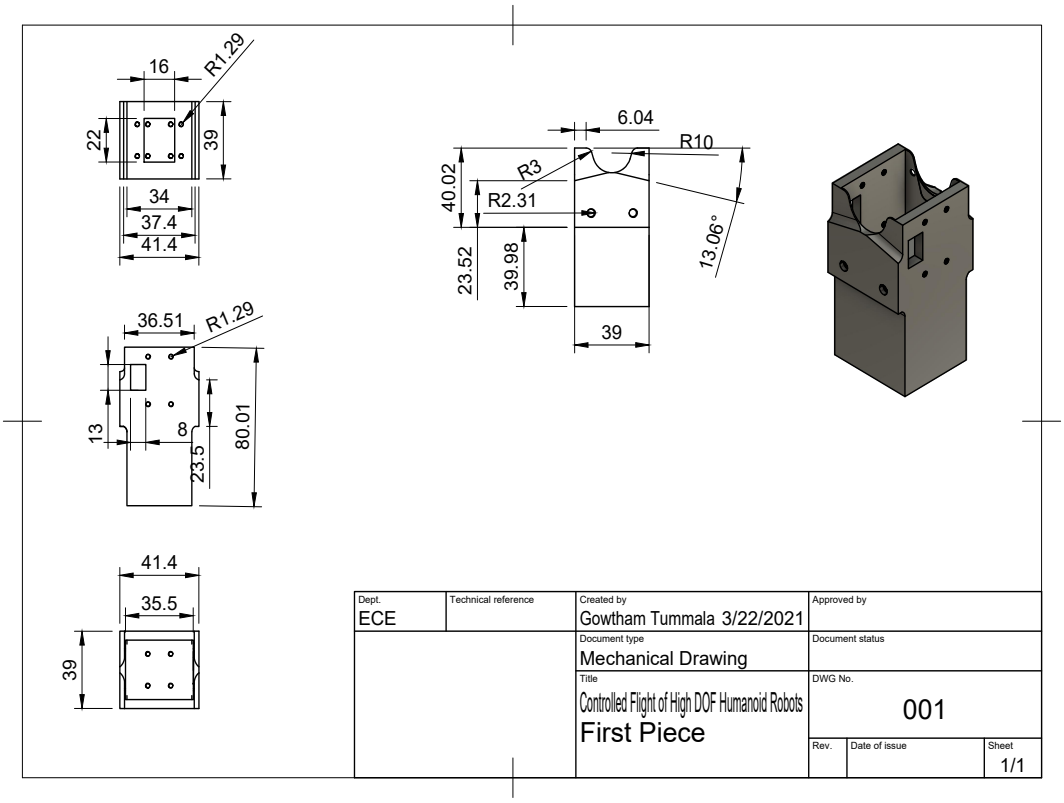
B.3.2 PID Gain Calculation

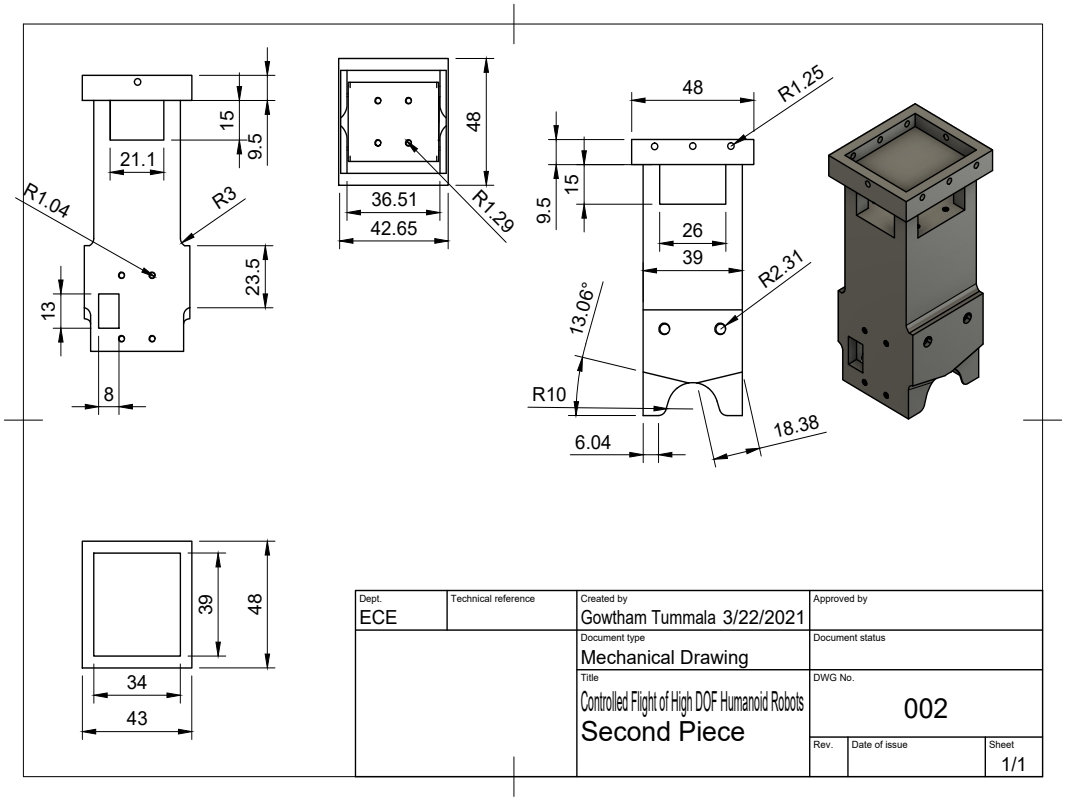
Here, we used MATLAB for the PID gain calculation and analysing the controller output when implemented the gains to the plant.

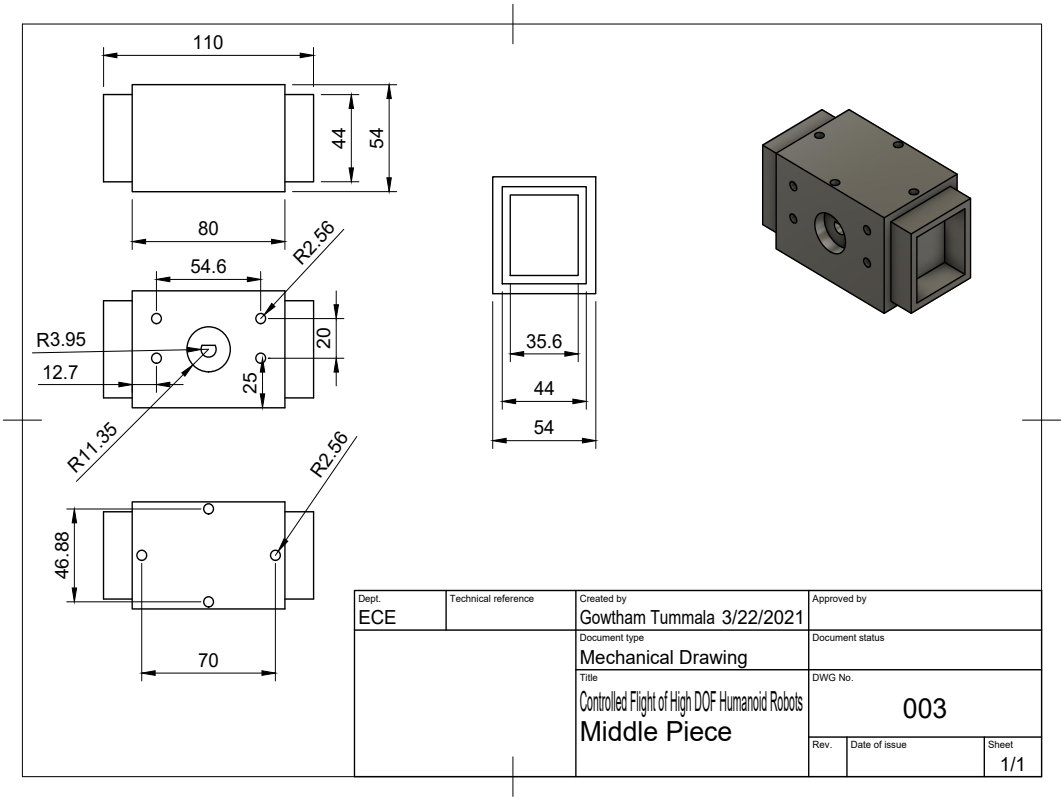
```
close all;clear all;
%% Plant Equations
load('normalPosData3.mat') % Data
% Initialization
iniForce = 0.0901;
zeta =  0.3;
wn =  2.0662;
gain = 435;

pos5_x = normalPosData3(:,1);
pos5_y = normalPosData3(:,2);
% Plant equation
Gp = tf(gain,[1 2*zeta*wn wn^2]);
[gpnum,gpden] = tfdata(Gp)
figure(5);
plot(pos5_x,pos5_y - pos5_y(1,1),'-r')
%hold on
%step(iniForce*Gp)
%% Designing a compensator 1
% Poles of the desired systems
syms a b;
rt = 3;
po = 5;
[a,b]= solve(pi-atan(sqrt(1 - a^2)/a)/(b*sqrt(1-a^2))-rt ,
   100*exp((-a*pi)/sqrt(1-a^2))-po);

zetaDes = double(a);
wnDes = double(b);

Sd1 = (-zetaDes*wnDes + 1i*wnDes*sqrt(1-zetaDes^2));
Sd2 = (-zetaDes*wnDes - 1i*wnDes*sqrt(1-zetaDes^2));
%Sd1=Sd2;
phi = phase(freqresp(Gp,Sd1))*180/pi-360;
phiC = -180 - phi;
zc = real(Sd1);
phiZc = phase(Sd1-zc)*180/pi;
phiPc = phiZc - phiC;
pc =    real(Sd1) - imag(Sd1)/tand(phiPc);
% Gain
K = abs(Sd1-pc)/(abs(Sd1-zc)*abs(freqresp(Gp,Sd1)));
% Lead compensator
Gct = K*tf([1 -zc],[1 -pc]);
essDes = 1;
KvDes = 1/essDes;
```

```
KvCurr =  abs(freqresp(Gp*Gct,essDes));
alphaLag = KvDes/KvCurr;
Zc = real(Sd1);
Pc = Zc/alphaLag;
% Compensator for steadystate
%Gcss = tf([1 -Zc],[1 -Pc]);
%Gc = Gct*Gcss;
% System
%G = Gc*Gp;
%% PD Controller system
T = -1/Pc;      // Time constant
Kp = K*Zc/Pc;   // Proportional gain
Kd = K*T;       // Derivative gain

C = tf([Kd Kp],[T 1]);
%% Plotting
%C1 = pid(0.01,0,0.005); // alternative testing method PID
   implementation

TT = feedback(C*Gp,1);
CF = feedback(C,Gp);
opt = stepDataOptions('StepAmplitude',15);

figure(2)
step(TT)
stepinfo(TT)
hold on;
%plot(impData1(:,1),impData1(:,2),'r-',impData1(:,1),impData1
   (:,3),'-b')
```

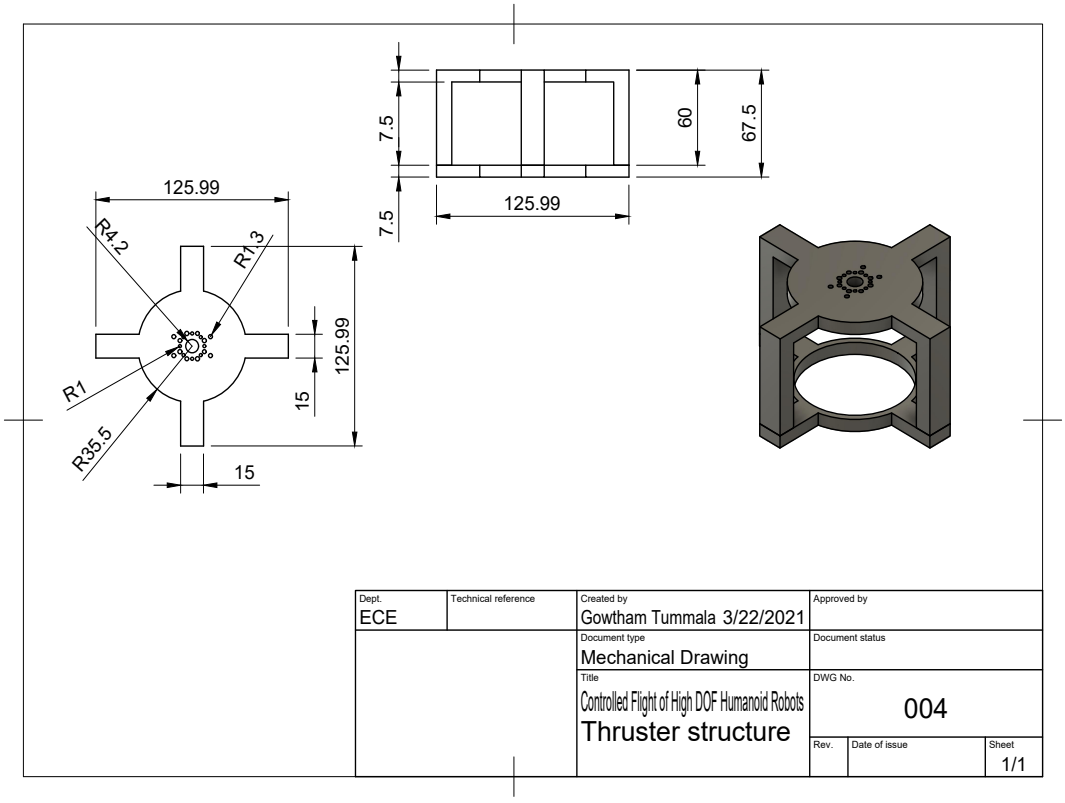
Appendix C: Mechanical Drawings



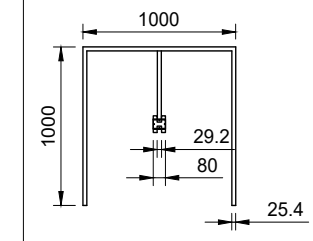

15
9.5
21.1
R1.04
R3
23.5
13
8
48
36.51
42.65
R1.29
48
R1.25
9.5
15
26
39
R2.31
13.06°
R10
6.04
18.38
39
48
34
43
Dept.
ECE
Technical reference
Created by
Gowtham Tummala 3/22/2021
Approved by
Document type
Mechanical Drawing
Document status
Title
Controlled Flight of High DOF Humanoid Robots
Second Piece
DWG No.
002
Rev.
Date of issue
Sheet
1/1

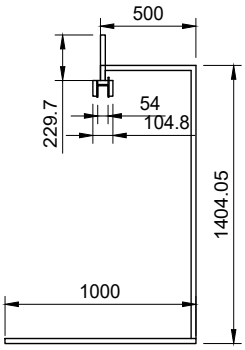


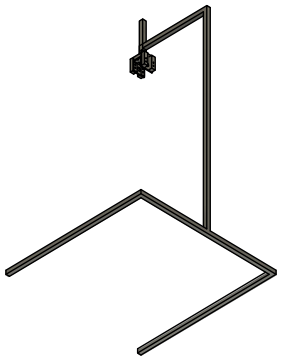
Dept.	Technical reference	Created by	Approved by
ECE		Gowtham Tummala 3/22/2021	
		Document type: Mechanical Drawing	Document status
		Title: Controlled Flight of High DOF Humanoid Robots — Middle Piece	DWG No.: 003
			Rev. / Date of issue / Sheet: 1/1

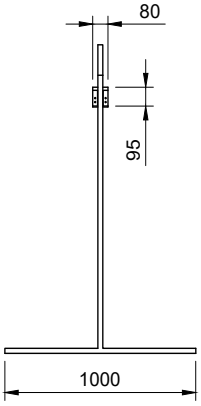


Dept.	Technical reference	Created by	Approved by
ECE		Gowtham Tummala 3/22/2021	
		Document type: Mechanical Drawing	Document status
		Title: Controlled Flight of High DOF Humanoid Robots — Thruster structure	DWG No.: 004
			Rev. / Date of issue / Sheet: 1/1

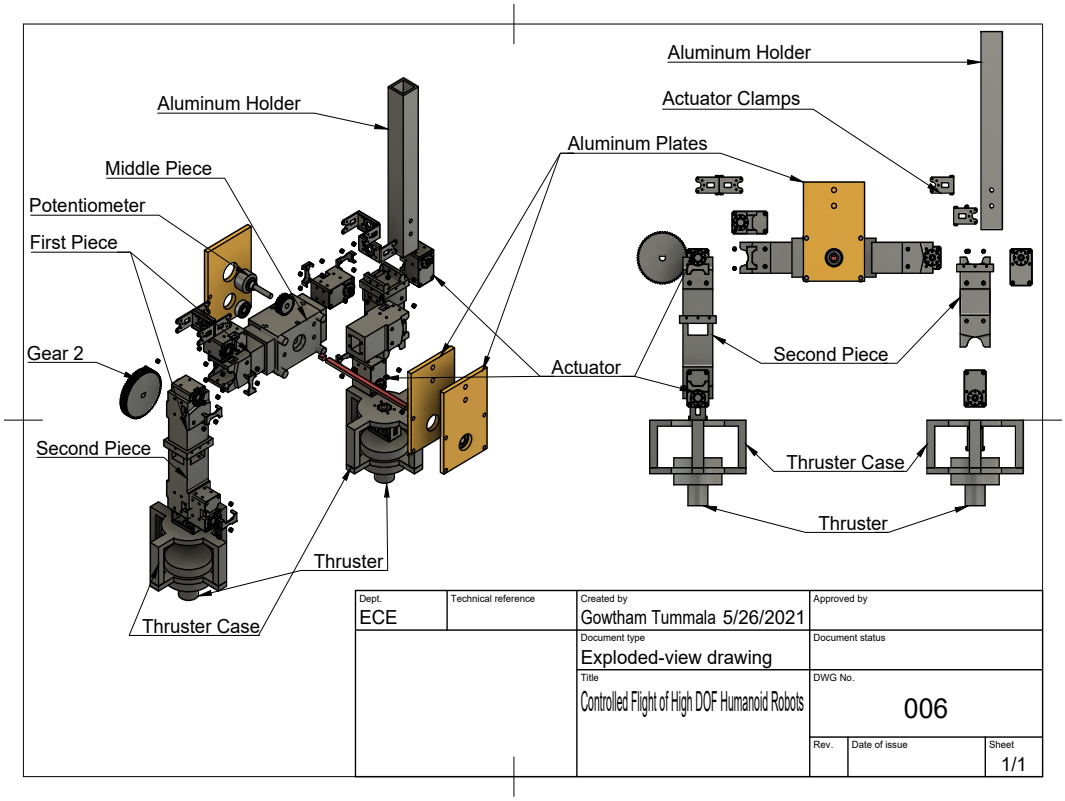








Dept.	Technical reference	Created by	Approved by
ECE		Gowtham Tummala 3/22/2021	
		Document type: Mechanical Drawing	Document status
		Title: Controlled Flight of High DOF Humanoid Robots — Aluminum Holder	DWG No.: 005
			Rev. / Date of issue / Sheet: 1/1


Aluminum Holder
Aluminum Holder
Actuator Clamps
Aluminum Plates
Middle Piece
Potentiometer
First Piece
Gear 2
Actuator
Second Piece
Second Piece
Thruster Case
Thruster
Thruster
Thruster Case
Dept.
ECE
Technical reference
Created by
Gowtham Tummala 5/26/2021
Approved by
Document type
Exploded-view drawing
Document status
Title
Controlled Flight of High DOF Humanoid Robots
DWG No.
006
Rev.
Date of issue
Sheet
1/1

Appendix D: List of Abbreviations and Symbols

D.1 Abbreviations

3D 3 Dimensional

AC Alternating Current

ADC Analog to Digital Converter

ARM Advanced RISC Machines

CAD Computer Aided Design

CAM Computer Aided Machining

CG Center of Gravity

CNC Computer Numerical Control

CoM Center of Mass

DASL Drones and Autonomous Systems Lab

DC Direct Current

DOF Degrees of Freedom

DSP Digital Signal Processing

EDF Electric Ducted Fans

ESC Electronic Speed Controller

F^{o} Fahrenheit

FSR Force Sensitive Resistors

Gb Giga bits

GPL General public License

GSE General State Equation

GUI Graphical User Interface

IMU Inertial Measurement Unit

IoT Internet of Things

LIPM Linear Inverted Pendulum Model

LPF Low Pass Filter

MCU Micro Controller Unit

MIMO Multi Input Multi Output

MPC Model-based Predictive Control

MRAC Model Reference Adaptive Control

OS Operating System

PID Proportional Integral Derivative

PLA Poly Lactic Acid

PLC Programmable Logic controller

PWM Pulse Width Modulation

RISC Reduced Instruction Set Computing

ROS Robot Operating System

RPM Revolution Per Minute

SDK Software Development Kit

SMPS Switched-Mode Power Supply

SSH Secured SHell

TF Transfer Function

TTL Transistor-Transistor Logic

UAV Unmanned Aerial Vehicle

UAM Urban Air Mobility

USB Universal Serial Bus

VTOL Vertical Takeoff and Landing

ZMP Zero Moment Point

D.2 Symbols

$G_p(S)$ Plant Model

$C(S)$ Control system

$G_{ct}(S)$ Compensator TF

ζ Damping Co-efficient

ω_n Natural Frequency

K_v Constant Velocity of the motor

t_r Rise Time

t_p Peak Time

t_s Settling Time

$PO\%$	Percentage Overshoot
k_p	Proportional gain in PID controller
k_d	Derivative gain in PID controller
k_i	Integral gain in PID controller
ϕ	phase of the pole
ϕ_c	phase of the compensator
τ	Time constant
θ	Actual/present angle of the system
θ_d	Desired angle of the system
k_r	Reference value
u	input of the system
K	Gain of the system
$e(t)$	error between the desired and actual value at time t
$d(t)$	desired output of the system
$y(t)$	output of the system
G_{ct}	Compensator TF
K_{ct}	Gain of the compensator TF
Z_c	Zero of the compensator TF
P_c	Pole of the compensator TF

Bibliography

Bibliography

[1] I.-W. Park, J.-Y. Kim, J. Lee, and J.-H. Oh, "Mechanical design of humanoid robot platform khr-3 (kaist humanoid robot 3: Hubo)," in *5th IEEE-RAS International Conference on Humanoid Robots, 2005.* IEEE, 2005, pp. 321–326.

[2] S. Shamsuddin, L. I. Ismail, H. Yussof, N. Ismarrubie Zahari, S. Bahari, H. Hashim, and A. Jaffar, "Humanoid robot nao: Review of control and motion exploration," in *2011 IEEE International Conference on Control System, Computing and Engineering*, 2011, pp. 511–516.

[3] D. Pucci, S. Traversaro, and F. Nori, "Momentum control of an underactuated flying humanoid robot," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 195–202, Jan 2018.

[4] G. Nava, L. Fiorio, S. Traversaro, and D. Pucci, "Position and attitude control of an underactuated flying humanoid robot," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, Nov 2018, pp. 1–9.

[5] M. Simon, "Boston dynamics' atlas robot does backflips now and it's full-tilt insane," *Wired, https://www. wired. com/story/atlas-robot-does-backflips-now*, 2017.

[6] M. Hirose and K. Ogawa, "Honda humanoid robots development," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 365, no. 1850, pp. 11–19, 2007.

[7] N. A. Radford, P. Strawser, K. Hambuchen, J. S. Mehling, W. K. Verdeyen, A. S. Donnan, J. Holley, J. Sanchez, V. Nguyen, L. Bridgwater *et al.*, "Valkyrie: Nasa's first bipedal humanoid robot," *Journal of Field Robotics*, vol. 32, no. 3, pp. 397–419, 2015.

[8] N. Owano, "Handle: Boston dynamics robot on wheels performs on stage [electronic resource]," *N. Owano*, 2017.

[9] "Mitra," Aug 2020. [Online]. Available: https://mitrarobot.com/

[10] H. Alzu'bi, I. Mansour, and O. Rawashdeh, "Loon copter: Implementation of a hybrid unmanned aquatic–aerial quadcopter with active buoyancy control," *Journal of field Robotics*, vol. 35, no. 5, pp. 764–778, 2018.

[11] C. J. Pratt and K. K. Leang, "Dynamic underactuated flying-walking (duck) robot," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 3267–3274.

[12] A. Kalantari and M. Spenko, "Design and experimental validation of hytaq, a hybrid terrestrial and aerial quadrotor," *2013 IEEE International Conference on Robotics and Automation*, pp. 4445–4450, 2013.

[13] C. Chung and M. Nakashima, "Swimming humanoid robot "swumanoid" as an experimental platform for research of human swimming," *Journal of Robotics and Mechatronics*, vol. 26, no. 2, pp. 265–266, 2014.

[14] A. Ruangwiset, "Conceptual design of twin rotors flying humanoid robot steered by cg shifting," in *2010 2nd International Conference on Mechanical and Electronics Engineering*, vol. 2, 2010, pp. V2–65–V2–69.

[15] M. Cooney, F. Zanlungo, S. Nishio, and H. Ishiguro, "Designing a flying humanoid robot (fhr): effects of flight on interactive communication," in *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2012, pp. 364–371.

[16] M. Zhao, T. Anzai, F. Shi, X. Chen, K. Okada, and M. Inaba, "Design, modeling, and control of an aerial robot dragon: A dual-rotor-embedded multilink robot with the ability of multi-degree-of-freedom aerial transformation," *IEEE Robotics and Automation Letters*, vol. 3, pp. 1176–1183, 2018.

[17] A. Ollero, G. Heredia, A. Franchi, G. Antonelli, K. Kondak, A. Sanfeliu, A. Viguria, J. R. Martinez-de Dios, F. Pierri, J. Cortes, A. Santamaria-Navarro, M. A. Trujillo Soto, R. Balachandran, J. Andrade-Cetto, and A. Rodriguez, "The aeroarms project: Aerial robots with advanced manipulation capabilities for inspection and maintenance," *IEEE Robotics Automation Magazine*, vol. 25, no. 4, pp. 12–23, 2018.

[18] E. M. Hoffman, N. Perrin, N. G. Tsagarakis, and D. G. Caldwell, "Upper limb compliant strategy exploiting external physical constraints for humanoid fall avoidance," in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2013, pp. 397–402.

[19] ——, "Upper limb compliant strategy exploiting external physical constraints for humanoid fall avoidance," in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2013, pp. 397–402.

[20] B. Stephens, "Integral control of humanoid balance," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 4020–4027.

[21] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 2, 2003, pp. 1620–1626 vol.2.

[22] Qiang Huang, K. Yokoi, S. Kajita, K. Kaneko, H. Arai, N. Koyachi, and K. Tanie, "Planning walking patterns for a biped robot," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 280–289, 2001.

[23] L. Sentis, J. Park, and O. Khatib, "Compliant control of multicontact and center-of-mass behaviors in humanoid robots," *IEEE Transactions on robotics*, vol. 26, no. 3, pp. 483–501, 2010.

[24] J. Chestnutt, M. Lau, K. M. Cheung, J. Kuffner, J. K. Hodgins, and T. Kanade, "Footstep planning for the honda asimo humanoid," in *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2005.

[25] J. Yang, Q. Huang, J. Li, C. Li, and K. Li, "Walking pattern generation for humanoid robot considering upper body motion," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 4441–4446.

[26] C. Kim, D. Kim, and Y. Oh, "Solving an inverse kinematics problem for a humanoid robot 2019s imitation of human motions using optimization." 01 2005, pp. 85–92.

[27] A. Takanishi, Hun-ok Lim, M. Tsuda, and I. Kato, "Realization of dynamic biped walking stabilized by trunk motion on a sagittally uneven surface," in *EEE International Workshop on Intelligent Robots and Systems, Towards a New Frontier of Applications*, 1990, pp. 323–330 vol.1.

[28] L. Hawley and W. Suleiman, "Control strategy and implementation for a humanoid robot pushing a heavy load on a rolling cart," 09 2017, pp. 4997–5002.

[29] S. Kajita and K. Tani, "Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, 1991, pp. 1405–1411 vol.2.

[30] ——, "Experimental study of biped dynamic walking in the linear inverted pendulum mode," in *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, vol. 3, 1995, pp. 2885–2891 vol.3.

[31] B. J. Stephens and C. G. Atkeson, "Dynamic balance force control for compliant humanoid robots," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 1248–1255.

[32] F. Gravez, B. Mohamed, and F. B. Ouezdou, "Dynamic simulation of a humanoid robot with four dofs torso," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 1, 2002, pp. 511–516 vol.1.

[33] N. Bilgin and M. K. Ozgoren, "A balance keeping control for humanoid robots by using model predictive control," in *2016 17th International Carpathian Control Conference (ICCC)*, 2016, pp. 60–65.

[34] B. Henze, C. Ott, and M. A. Roa, "Posture and balance control for humanoid robots in multi-contact scenarios based on model predictive control," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 3253–3258.

[35] J. A. Castano, C. Zhou, and N. Tsagarakis, "From non-reactive to reactive walking in humanoid robots," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, 2018, pp. 280–283.

[36] V. G. Ivancevic and M. Snoswell, "Fuzzy-stochastic functor machine for general humanoid-robot dynamics," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 31, no. 3, pp. 319–330, 2001.

[37] G. Nava, F. Romano, F. Nori, and D. Pucci, "Stability analysis and design of momentum-based controllers for humanoid robots," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 680–687.

[38] S. Hyon, "Compliant terrain adaptation for biped humanoids without measuring ground surface and contact forces," *IEEE Transactions on Robotics*, vol. 25, no. 1, pp. 171–178, 2009.

[39] S. Nakaoka, A. Nakazawa, K. Yokoi, H. Hirukawa, and K. Ikeuchi, "Generating whole body motions for a biped humanoid robot from captured human dances," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 3, 2003, pp. 3905–3910 vol.3.

[40] D. M. Lofaro, R. Ellenberg, P. Oh, and J. Oh, "Humanoid throwing: Design of collision-free trajectories with sparse reachable maps," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 1519–1524.

[41] D. M. Lofaro, C. Sun, and P. Oh, "Humanoid pitching at a major league baseball game: Challenges, approach, implementation and lessons learned," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, 2012, pp. 423–428.

[42] P. E. I. Pounds, D. R. Bersak, and A. M. Dollar, "Grasping from the air: Hovering capture and load stability," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2491–2498.

[43] K. W. Weng and M. S. b. Z. Abidin, "Design and control of a quad-rotor flying robot for aerial surveillance," in *2006 4th Student Conference on Research and Development*, 2006, pp. 173–177.

[44] A. Jaimes, S. Kota, and J. Gomez, "An approach to surveillance an area using swarm of fixed wing and quad-rotor unmanned aerial vehicles uav(s)," in *2008 IEEE International Conference on System of Systems Engineering*, 2008, pp. 1–6.

[45] X. Hu, M. Wang, C. Qian, C. Huang, Y. Xia, and M. Song, "Lidar-based slam and autonomous navigation for forestry quadrotors," in *2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC)*, 2018, pp. 1–6.

[46] T. H. Pham, Y. Bestaoui, and S. Mammar, "Aerial robot coverage path planning approach with concave obstacles in precision agriculture," in *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, 2017, pp. 43–48.

[47] A. Sagitov and Y. Gerasimov, "Towards dji phantom 4 realistic simulation with gimbal and rc controller in ros/gazebo environment," in *2017 10th International Conference on Developments in eSystems Engineering (DeSE)*, 2017, pp. 262–266.

[48] X. Ding, P. Guo, K. Xu, and Y. Yu, "A review of aerial manipulation of small-scale rotorcraft unmanned robotic systems," *Chinese Journal of Aeronautics*, vol. 32, 06 2018.

[49] K. Guevara, M. Rodriguez, N. Gallo, G. Velasco, K. Vasudeva, and I. Guvenc, "Uav-based gsm network for public safety communications," in *SoutheastCon 2015*, 2015, pp. 1–2.

[50] A. Abdessameud and A. Tayebi, "Global trajectory tracking control of vtol-uavs without linear velocity measurements," *Automatica*, vol. 46, no. 6, pp. 1053 – 1059, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109810001299

[51] J. F. Guerrero-Castellanos, N. Marchand, A. Hably, S. Lesecq, and J. Delamare, "Bounded attitude control of rigid bodies: Real-time experimentation to a quadrotor mini-helicopter," *Control Engineering Practice*, vol. 19, pp. 790–797, 2011.

[52] T. Hamel, R. Mahony, R. Lozano, and J. Ostrowski, "Dynamic modelling and configuration stabilization for an x4-flyer," 07 2002.

[53] L. Marconi and R. Naldi, "Robust full degree-of-freedom tracking control of a helicopter," *Automatica*, vol. 43, no. 11, pp. 1909 – 1920, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109807002154

[54] J. Pflimlin, P. Souères, and T. Hamel, "Hovering flight stabilization in wind gusts for ducted fan uav," vol. 4, 01 2005, pp. 3491 – 3496 Vol.4.

[55] M. Nguyen Duc, T. N. Trong, and Y. S. Xuan, "The quadrotor mav system using pid control," in *2015 IEEE International Conference on Mechatronics and Automation (ICMA)*, 2015, pp. 506–510.

[56] L. Lipera, J. D. Colbourne, M. B. Tischler, M. H. Mansur, M. Rotkowitz, and P. Patangui, "The micro craft istar micro air vehicle: Control system design and testing," 2001.

[57] T. J. Koo and S. Sastry, "Output tracking control design of a helicopter model based on approximate linearization," in *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No.98CH36171)*, vol. 4, 1998, pp. 3635–3640 vol.4.

[58] R. J. Wood, "The first takeoff of a biologically inspired at-scale robotic insect," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 341–347, 2008.

[59] K. Y. Ma, P. Chirarattananon, S. B. Fuller, and R. J. Wood, "Controlled flight of a biologically inspired, insect-scale robot," *Science*, vol. 340, no. 6132, pp. 603–607, 2013. [Online]. Available: https://science.sciencemag.org/content/340/6132/603

[60] L. Y. Matloff, "Birds of a feather: Designing feathered morphing wings for soft aerial robots," Ph.D. dissertation, Stanford University, 2020.

[61] A. J. Ijspeert, "Biorobotics: Using robots to emulate and investigate agile locomotion," *Science*, vol. 346, no. 6206, pp. 196–203, 2014. [Online]. Available: https://science.sciencemag.org/content/346/6206/196

[62] M. Müller, S. Lupashin, and R. D'Andrea, "Quadrocopter ball juggling," in *2011 IEEE/RSJ international conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 5113–5120.

[63] W. Dong, G.-Y. Gu, X. Zhu, and H. Ding, "High-performance trajectory tracking control of a quadrotor with disturbance observer," *Sensors and Actuators A: Physical*, vol. 211, pp. 67–77, 2014.

[64] A. Azzam and Xinhua Wang, "Quad rotor arial robot dynamic modeling and configuration stabilization," in *2010 2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR 2010)*, vol. 1, 2010, pp. 438–444.

[65] F. Ruggiero, V. Lippiello, and A. Ollero, "Aerial manipulation: A literature review," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1957–1964, July 2018.

[66] Y. Name, "Robotis e." [Online]. Available: http://emanual.robotis.com/

[67] P. Van den Hof, X. Bombois, and L. N. D. Course, "System identification for control," *Delft Center for Systems and Control, TU-Delft. Lecture notes, Dutch Institute for Systems and Control (DISC)*, 2004.

[68] K. Ogata, *Modern control engineering*, 5th ed., ser. Instrumentation and control series. Boston, MA: Prentice-Hall, 2010.

[69] P. K. Janert, *Feedback control for computer systems*. North Sebastopol, California: O'Reilly.

[70] Z. K. Jadoon, S. Shakeel, A. Saleem, A. Khaqan, S. Shuja, S. A. Malik, R. Ali Riaz *et al.*, "A comparative analysis of pid, lead, lag, lead-lag, and cascaded lead controllers for a drug infusion system," *Journal of healthcare engineering*, vol. 2017, 2017.

[71] Kiam Heong Ang, G. Chong, and Yun Li, "Pid control system analysis, design, and technology," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 559–576, 2005.

[72] R. O'Brien and J. M. Watkins, "A unified approach for teaching root locus and bode compensator design," in *Proceedings of the 2003 American Control Conference, 2003.*, vol. 1. IEEE, 2003, pp. 645–649.

[73] *Multi-Rotor35 Brushless DC Speed Control*, Castle, 2016. [Online]. Available: http://www.castlecreations.com/medias/CastleCreationsMulti-Rotor35DatasheetRev1.1.pdf

[74] "Smoothdata matlab." [Online]. Available: https://www.mathworks.com/help/matlab/ref/smoothdata.html

Curriculum Vitae

Gowtham Tummala is a Master of Science student in the Department of Electrical and Computer Engineering (ECE) at George Mason University. He received the Bachelor of Engineering degree in Electrical and Communications Engineering from Anna University, Chennai, India in 2017. In Spring 2018, he started his Master's degree in Electrical Engineering with a focus in controls and robotics. He worked as a graduate teaching assistant during Spring 2019 and Fall 2019. His primary research interests are control systems, robotics, embedded systems and artificial intelligence.