B. TECH. PROJECT REPORT

On

Task-Space Position Tracking Performance Investigations of a Planar Vehicle-Manipulator

BY

A G SAI KIRAN

BADABAGNI HITESH

DISCIPLINE OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

November 2018

Task-Space Position Tracking Performance Investigations of a Planar Vehicle-Manipulator

A PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degrees

of

BACHELOR OF TECHNOLOGY

in

MECHANICAL ENGINEERING

Submitted by:

A G SAI KIRAN

BADABAGNI HITESH

Guided by:

Dr. Santhakumar Mohan

Associate Professor

INDIAN INSTITUTE OF TECHNOLOGY INDORE

November 2018

CANDIDATE'S DECLARATION

We hereby declare that the project entitled **"Task-Space Position Tracking Performance Investigations of a Planar Vehicle-Manipulator"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in 'Mechanical Engineering' completed under the supervision of **Dr. Santhakumar Mohan, Associate Professor, Mechanical Engineering,** IIT Indore is an authentic work.

Further, we declare that we have not submitted this work for the award of any other degree elsewhere.

(A G SAI KIRAN)

DATE:_____

(BADABAGNI HITESH)

DATE:_____

CERTIFICATE by BTP Guide

It is certified that the above statement made by the students is correct to the best of my knowledge.

(Dr. Santhakumar Mohan)

Associate Professor

Mechanical Engineering

Preface

This report on “Task-Space Position Tracking Performance Investigations of a Planar Vehicle-Manipulator" is prepared under the guidance of Dr. Santhakumar Mohan.

Through this report we have tried to give a detailed explanation of new control scheme for Mobile Manipulators and to cover every aspect of the new control scheme including mechanical designs, electronic components and programming logics.

We have tried to the best of our abilities and knowledge to explain the content in a lucid manner. We have also added specifications of components 3D models and figures to make it more illustrative.

This report contains 7 chapters and outline is given in the table of contents page. Real-time working images are also included.

A G SAI KIRAN
B.Tech. IV Year
Discipline of Mechanical Engineering
IIT Indore

BADABAGNI HITESH
B.Tech. IV Year
Discipline of Mechanical Engineering
IIT Indore

Acknowledgements

We wish to thank Dr. Santhakumar Mohan for his kind support and valuable guidance.

It is his help and support, due to which we became able to complete the design and technical report.

Without his support this report would not have been possible.

A G SAI KIRAN
B.Tech. IV Year
Discipline of Mechanical Engineering
IIT Indore

BADABAGNI HITESH
B.Tech. IV Year
Discipline of Engineering
IIT Indore

Abstract

Unlike conventional mobile robots which use inverse of higher order rectangular Jacobian matrix in order to calculate input variables for control resulting in instability and needing high performance microprocessors, a resolved control for mobile base and manipulator is being used which reduces the complexity to some extent and can be controlled with a low cost microcontroller (Arduino). The project focuses on positioning the base such that the point of interest lies in the dexterous workspace of the manipulator for the end-effector to be able to reach the point with any orientation. Feedback sensors are used to track and control the position and orientation of the base and manipulator in a closed-loop manner. Worked with SolidWorks for designing, MATLAB-Simulink and Arduino as electronic prototyping platforms. The control scheme is implemented and the performance is analysed.

Table of Contents

List of Figures

List of Tables

Chapter 1. Overview

1.1 Introduction

Robots have replaced humans in performing repetitive and dangerous tasks which humans prefer not to do and tasks that are unable to do because of size limitations, which take place in extreme environments such as outer space or the bottom of the sea. By definition a Robot is "a reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through various programmed motions for the performance of a variety of task." If locomotion is added along with manipulation, it is called Mobile Manipulator.

As name suggests, locomotion(mobile) and performing a certain task(Manipulator) is what Mobile Manipulator or Mobile Robot is intended to do. Locomotion is facilitated by mobile base which comprises wheels, Continuous tracks etc. Manipulation is achieved with the help of open or closed kinematic chains powered by different modes of actuation. It is a system with the following functional characteristics

- Mobility: total mobility relative to the environment
- A certain level of autonomy: limited human interaction
- Perception ability: sensing and reacting in the environment

Mobile Robots have wide range of applications like Cleaning large buildings, Transportation industry and Surveillance buildings indoors and have outdoor applications like Agriculture, Forest, Space, Military, Firefighting, Sewage tubes, Mining etc. In any of these situations a robot may be bounded to work in a limited space where doing a task is restricted in some orientations as obstacles present are hindrance to the base. This project focuses on reaching the job in all orientation keeping the base at a fixed location after approaching that point, so that robot can scale up its performance to tighter spaces. It also allows the robot to

have flexibility in choosing a particular orientation with minimum force requirement to do a certain task at a point.

1.2 History

Mobile robots have been in action since early 20th century. The first AGV systems were developed in the 1950s by Barrett Electronics, USA. Grey Walter from University of Bristol built three wheeled built three wheeled, turtle like, mobile robotic vehicles. These vehicles had a light sensor, touch sensor, propulsion motor, steering motor, and a two vacuum tube analog computer. Later on in 1969, Stanford University developed the first vision controlled mobile robot, Shakey. It was intended to perform simple tasks like recognizing the object using vision, find its way to it and perform some action on it.

Legged robots like Genghis in 1989 and Dante II in 1994 added more capabilities to mobile robots to work in extreme environments. Robots like Nomad helped in exploring the remote Antarctic region of Elephant Moraine in search of new meteorite samples enabling autonomous discovery of Antarctic meteorites. In space applications robots like Curiosity helps in exploring other planets. Mobile robots came way far since their origin helping humans in replacing dangerous tasks, exploring places, assisting tasks etc. with their ability to move and work on certain object. Developing new control schemes to enhance their capabilities contributes more to the development of the field.

1.3 Motivation

In any construction site, when we observe an earth mover like backhoe loader working, we might notice that it operator always position the vehicle at a certain distance from the point that he intended to work on. This distance has a significance, it's not too near or far from the vehicle i.e., not in a position that Backhoe has to reach extreme positions to work. This is because extreme positions of the backhoe limit possible orientations that he can work on the object with loader(shovel/bucket). So positioning at right distance gives more freedom in terms of orientation i.e., giving many possible orientations in reaching a specific point helping in excavations or other intended actions with ease by fixing the vehicle's position. Like backhoe, all other manipulators have this kind of classification in workspaces around them where in reachability is maximum in terms of orientations at a specific distance

from base. Dexterous workspace is where all orientations all possible and this project focuses on positioning the base such that point of interest (object that we are working on) falls into dexterous workspace enabling the manipulator to reach that point in all orientation without changing the base position.

Fig 1.1 Extreme positions with minimum flexibility upon ground

Fig 1.2 Optimum position with maximum flexibility upon ground

Chapter 2. Base And Manipulator Modeling

2.1 Basic Concepts Of Base

As discussed earlier, the purpose of base is locomotion or mobility relative to the environment.A mobile robot needs locomotion mechanisms that enable it to move unbounded throughout its environment. Locomotion mechanisms used in biological (nature) systems (land based) are Crawling, Sliding, Walking and jumping etc. biological systems succeed in moving through a wide variety of harsh environments. Therefore it can be desirable to copy their selection of locomotion mechanisms. However, replicating nature in this regard is extremely difficult for several reasons.

Locomotion is the complement of manipulation and in locomotion, the environment is fixed and the robot moves by imparting force to the environment. Key issues for locomotion:

- Stability
    - Number of contact points
    - Center of gravity
    - Static/dynamic stabilization
    - Inclination of terrain
- Characteristics of contact
    - Contact point or contact area

- Angle of contact
- Friction

- Type of environment
  - Structure
  - Medium (soft or hard ground)

Owing to these limitations, mobile robots generally use either wheeled mechanisms, or a small number of articulated legs. Legged locomotion requires higher degrees of freedom and therefore greater mechanical complexity than wheeled locomotion. Wheels, in addition to being simple, are extremely well suited to flat ground. On flat surfaces wheeled locomotion is one to two orders of magnitude more efficient than legged locomotion.But as the surface becomes soft, wheeled locomotion accumulates inefficiencies due to rolling friction whereas legged locomotion suffers much less because it consists only point contacts with the ground.The efficiency of wheeled locomotion depends greatly on environmental qualities, particularly the flatness and hardness of the ground. The efficiency of legged locomotion depends on the leg mass and body mass, both of which the robot must support at various points in a legged gait. Nature favors legged locomotion to operate on rough and unstructured terrain.Stability is not typically an issue in wheeled robot designs because all wheels are on the ground at the same time. Three wheels are sufficient to guarantee stable balance. So in order to keep it simple we are dealing with wheeled mechanisms.

Wheeled mechanism is further classified as follows

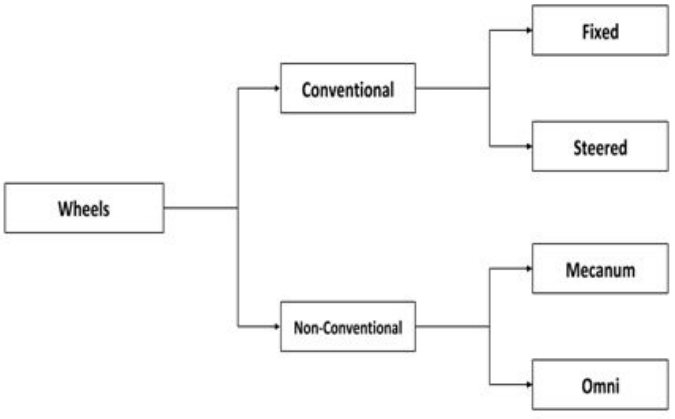


Fig 2.1 *Classification of wheels*

2.1.1 Conventional Wheels

Conventional wheels are mechanically simple, have high load capacity and high tolerance to work surface irregularities. However, due to their non-holonomic nature, they are not truly omni-directional. Designs have been proposed to achieve near omni-directional mobility using conventional wheels. The most common designs are those using steered wheels. Vehicles based on this design have at least two active wheels, each of which has both driving and steering actuators. They can move in any direction from any configurations. However, this type of system is not truly omni-directional because it needs to stop and re-orient its wheels to the desired direction whenever it needs to travel in a trajectory with non-continuous curvatures.

Most special wheel designs are based on a concept that achieves traction in one direction and allow passive motion in another, thus allowing greater flexibility in congested environments One of the more common omni-directional wheel designs is that of the Mecanum wheel, invented in 1973 by Bengt Ilon, an engineer with the Swedish company Mecanum AB. Many of the other commonly currently used designs are based on Ilon's original concept.

2.1.2 Mecanum wheels

Ilon's Mecanum wheel is based on the principle of a central wheel with a number of rollers placed at an angle around the periphery of the wheel(as shown in fig.2). The angled peripheral rollers translate a portion of the force in the rotational direction of the wheel to a force normal to the wheel direction. Depending on each individual wheels direction and speed, the resulting combination of all these forces produces a total force vector in any desired direction thus allowing the platform to move freely in the direction of the resulting force vector, without changing the direction of the wheels themselves.

The design in figure 2.2 shows a traditional Mecanum wheel with the peripheral rollers held in place from the outside. This design, though having a good load carrying capacity, has the disadvantage that, when encountering an inclined or an uneven work surface, the rim of the wheel can make contact with the surface instead of the roller, thus preventing the wheel form operating correctly. This is illustrated in figure 2.3.

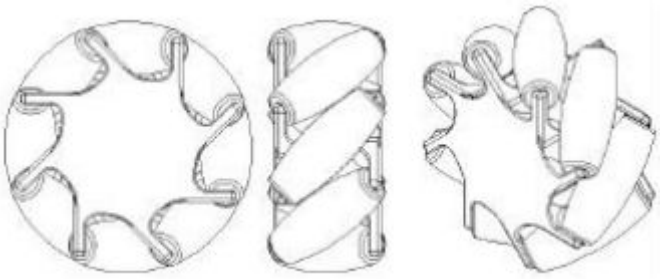


Fig 2.2 *Mecanum wheel based on Ilon's concept*

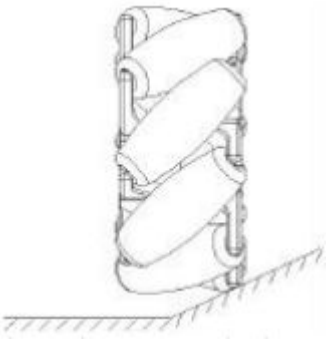


Fig 2.3 *Mecanum wheel on inclined surface*

Controlling direction using mecanum

For a vehicle using Mecanum wheels to achieve omni- directional movement, the wheels are arranged on the platform in a configuration as shown in figure 4. Depending on the direction of the motors, the wheels always produce both a forward or reverse force, as well as an inward or outward force, caused by the angled peripheral rollers. Depending on the resulting combination of these forces, the platform can be controlled to move in any direction, as seen in fig.5. A variety of other motions are possible by varying the direction and speeds of the wheel. Though incredibly versatile, the standard Mecanum wheel has an unfortunate side effect which reduces its efficiency considerably. Its wide range of mobility is due to the fact that the peripheral rollers translate a portion of the motor force into a force perpendicular or at an angle to that produced by the motor. This means that are large portion of the force in one direction is lost through the translation into a resulting force by the rollers. As an extreme example of this inefficiency, when the platform travels diagonally, only a front and rear opposing wheels are spinning whilst the rollers on the other two wheels cause direct drag that the motors must fight against.

Fig 2.4 *Mecanum Wheel Configuration*

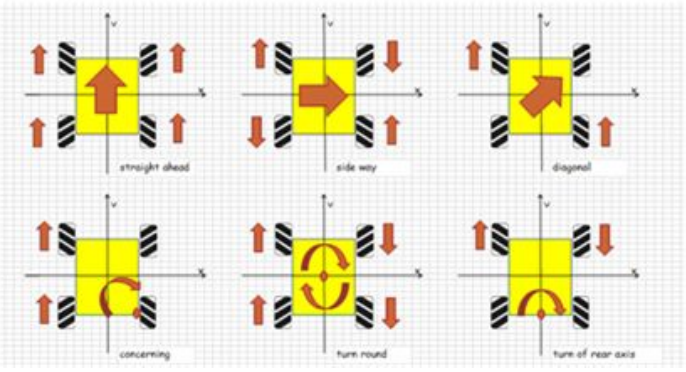


Fig 2.5 *Resultant base motion direction for different wheel spins*

2.2 Basic concepts of Manipulator

The purpose of the manipulator is to able to work on a certain object with certain tool attached to its End effector(that has effect on environment). It is simply series of links(bodies) connected together with the help of joints(allows relative motion between links) in order to form open or closed kinematic chains. A serial manipulator is a open kinematic chain and a parallel manipulator is the one that is formed with one or more closed kinematic loops in which links are connected to each other with different types of joints like parallel, rotary etc.

This project deals with planar serial manipulator with 3 rigid links and 3 revolute joints(3-DOF RRR). The following section consists some manipulator terminology with a brief explanation which will be used in further topics.

2.2.1 Manipulator Terminology

Reachable Workspace:

Locus of all points in the space around manipulator that can be reached by end effector with at least one possible orientation with respect to base frame.

Dexterous Workspace:

Locus of all points in the space around manipulator that can be reached by the end effector with any orientation with respect to base frame.

Task-Space(Operation space) variables:

- Variables in Space or Frame in which robot is working(Ground frame).
- Desired position and orientation coordinates.

Joint-Space(Configuration space) variables:

- Variables in the Robotic frame of reference.
- Orientation of each link or angular displacement of each joint of a robotic link

2.2.2 Dexterous workspace of Manipulator

As we have seen in the previous section, Dexterous workspace is the locus of points that can be approached with every orientation (0^0 - 360^0). Dexterous workspace can be found by assuming our manipulator to be a four-bar mechanism with distance separating end effector and first joint at the base as fixed link length and applying Grashof's condition. Generally in 3-DOF RRR manipulator last link (away from base) is the shortest or one of the shortest links.

The Grashof's law states that for a four-bar linkage system, the sum of the shortest and longest link of a planar quadrilateral linkage is less than or equal to the sum of the remaining two links, then the shortest link can rotate fully with respect to a neighboring link.

$$S+L <= P+Q$$

Where,

S = Shortest link,

L = Longest link,

P,Q = Other two links

To find dexterous workspace fix the manipulator's first joint(joint connecting first link with the base) at point M_1 in a 2D coordinate system. Let first, second and third links lengths be L_1, L_2, L_3 respectively and first, second and third joints be M_1, M_2, M_3 respectively. Apply Grashof's condition to the manipulator along any line in that 2D plane between minimum and maximum stretch (L_1+ L_2+ L_3) of the manipulator for varying fixed link length (L_f) . This gives specific zones along the line where dexterous is possible and by sweeping that line about M_1 gives Dexterous workspace (as those zones are symmetric about M_1 and gives area where Grashof's law is obeyed when sweeped to form a circle with M_1 as centre).

2.3 Kinematic model of a mecanum wheel drive base

The robotic base has four Mecanum wheels which are independently actuated by four individual motors. Each Mecanum wheel has peripheral rollers inclined at a constant angle a=45^0. Motion of the robot is assumed to be smooth and always in horizontal plane. Moreover, the center of gravity and the center of mass of the robot coincide with each other.

Figure 2.6 shows the kinematic structure of the robotic system that has three coordinate frames as follows: Q_{cwi}(i=1, 2, 3, 4)wheel coordinate frame, R_{cxy} is the robot(base) coordinate frame, and Q_{cxy} is the fixed coordinate frame. The velocity vector of wheel i, v_{ci} = $[\dot{x}_{cwi}, \dot{y}_{cwi}, \dot{\theta}_{cwi}]^T$ in the wheel coordinate frame is given as follows:

$$\begin{bmatrix} \dot{x}_{cwi} \\ \dot{y}_{cwi} \\ \dot{\theta}_{cwi} \end{bmatrix} = \begin{bmatrix} 0 & r_i \sin(\alpha_i) & 0 \\ R_i & -r_i \cos(\alpha_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\theta}_{ix} \\ \dot{\theta}_{ir} \\ \dot{\theta}_{iz} \end{bmatrix} \qquad \text{Equation 2.1}$$

Where,

$\dot{\theta}_{ix}$(i=1, 2 , 3 , 4): rotational speed of the roller around the hub.

$\dot{\theta}_{ir}$(i=1, 2, 3, 4): rotational speed of the roller.

$\dot{\theta}_{iz}$ (i=1, 2, 3, 4): rotational speed of the wheel around the point of contact with surface.

R_i: radius of the wheels and r_i : radius of the roller.

α_i(i=1, 2, 3, 4): roller slope angles of each wheel (inclination of peripheral rollers).

$\dot{P}_{cxy} = [\dot{x}_r, \dot{y}_r, \dot{\theta}_r]^T$is the velocity vector in the robot (moving) frame. The relation between the velocities in the wheel frame and the robot frame can be given as follows:

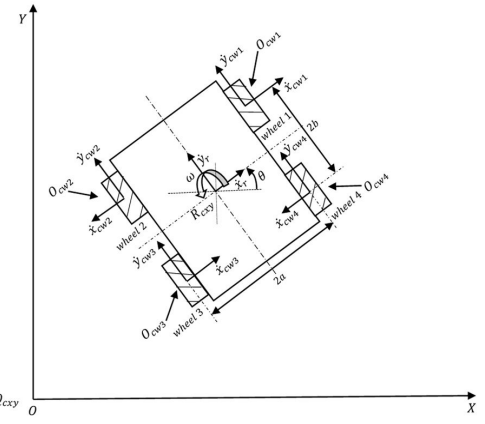


Fig 2.6 *Kinematic structure of mecanum base*

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} \cos(\varphi^r_{cwi}) & -\sin(\varphi^r_{cwi}) & d^r_{cwiy} \\ \sin(\varphi^r_{cwi}) & \cos(\varphi^r_{cwi}) & d^r_{cwix} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_{cwi} \\ \dot{y}_{cwi} \\ \dot{\theta}_{cwi} \end{bmatrix} \quad \text{Equation 2.2}$$

Where,

φ^r_{cwi}(i=1, 2, 3, 4) is the rotational angle between O_{cwi}(i=1, 2, 3, 4)and R_{cxy} frames. Here d^r_{cwix} and d^r_{cwiy} are the longitudinal and lateral distances between these two coordinate frames. From the kinematic structure of the base, φ^r_{cwi}=0. Hence, Equation 2.2 can be written as follows:

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} 1 & 0 & d^r_{cwiy} \\ 0 & 1 & d^r_{cwix} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_{cwi} \\ \dot{y}_{cwi} \\ \dot{\theta}_{cwi} \end{bmatrix} \quad \text{Equation 2.3}$$

Using Equations 2.1 and 2.3, the velocity of the robot in moving frame with regard to the rotational velocities of the roller can be determined as follows:

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} 0 & r_i \sin(\alpha_i) & d^r_{cwiy} \\ R_i & -r_i \cos(\alpha_i) & d^r_{cwix} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\theta}_{ix} \\ \dot{\theta}_{ir} \\ \dot{\theta}_{iz} \end{bmatrix} \quad \text{Equation 2.4}$$

where, the Jacobian matrix (J_i) for wheel i is given as

$$J_i = \begin{bmatrix} 0 & r_i \sin(\alpha_i) & d^r_{cwiy} \\ R_i & -r_i \cos(\alpha_i) & d^r_{cwix} \\ 0 & 0 & 1 \end{bmatrix} \quad \text{Equation 2.5}$$

From the geometrical structure of the robot, all the wheels are identical, so r_i(i=1, 2, 3, 4)=r. The roller angles are α_1=45^0, α_2= -45^0, α_3=45^0, α_4= -45^0 and longitudinal and lateral distances are $d^r_{cw1x} = d^r_{cw4x}$ =a, d^r_{cw2x}=d^r_{cw3x}= -a; d^r_{cw1y}=d^r_{cw2y}=b, d^r_{cw3y} =d^r_{cw4y}= -b. Using these parameters, the Jacobian matrix for each wheel can be given as follows:

$$J_1 = \begin{bmatrix} 0 & \frac{r}{\sqrt{2}} & b \\ R & -\frac{r}{\sqrt{2}} & a \\ 0 & 0 & 1 \end{bmatrix}$$

$$J_2 = \begin{bmatrix} 0 & -\frac{r}{\sqrt{2}} & b \\ R & -\frac{r}{\sqrt{2}} & -a \\ 0 & 0 & 1 \end{bmatrix}$$

$$J_3 = \begin{bmatrix} 0 & \frac{r}{\sqrt{2}} & -b \\ R & -\frac{r}{\sqrt{2}} & -a \\ 0 & 0 & 1 \end{bmatrix}$$

$$J_4 = \begin{bmatrix} 0 & -\frac{r}{\sqrt{2}} & -b \\ R & -\frac{r}{\sqrt{2}} & a \\ 0 & 0 & 1 \end{bmatrix}$$

Equation 2.6

From these equations, the inverse kinematic relationship can be given as follows:

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta} \end{bmatrix} = \mathbf{J} \begin{bmatrix} \dot{\theta}_{\text{cw1}} \\ \dot{\theta}_{\text{cw2}} \\ \dot{\theta}_{\text{cw3}} \\ \dot{\theta}_{\text{cw4}} \end{bmatrix}$$

Equation 2.7

where,

$$J = \frac{R}{4} \begin{bmatrix} -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ \frac{1}{a+b} & -\frac{1}{a+b} & -\frac{1}{a+b} & \frac{1}{a+b} \end{bmatrix}$$

Equation 2.8

Here $Q_{cxy}=[x_q, y_q, \theta_q]^T$ represents the position and the orientation of the robot Qcxy frame. The velocity of robot in the Q_{cxy} can be calculated as follows:

$$\begin{bmatrix} \dot{x}_q \\ \dot{y}_q \\ \dot{\theta}_q \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} \qquad \text{Equation 2.9}$$

Where,

$(\dot{x}_q, \dot{y}_q, \dot{\theta}_q)$ = velocities in fixed coordinate frame.

$(\dot{x}_r, \dot{y}_r, \dot{\theta}_r)$ = velocities in robot(base) coordinate frame.

On integrating equation 2.9 we get position and orientation (x_q, y_q, θ_q) in fixed coordinate frame or operational space.

2.4 Kinematic Model of a 3-DOF RRR Planar Manipulator

Geometric Approach

In a geometric approach to find a manipulator's solution, we try to decompose the spatial geometry of the arm into several plane-geometry problems. For many manipulators (particularly when the link twist angle $\alpha_i = 0$ or ± 90) this can be done quite easily. Joint angles can then be solved for by using the tools of plane geometry. For the arm with three degrees of freedom shown in Fig. 2.7a and 2.7b, because the arm is planar, we can apply plane geometry directly to find a solution.

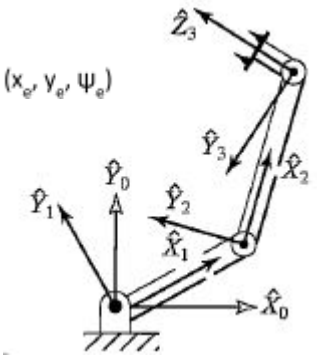


Fig 2.7 (a) *3 link manipulator with individual frames*

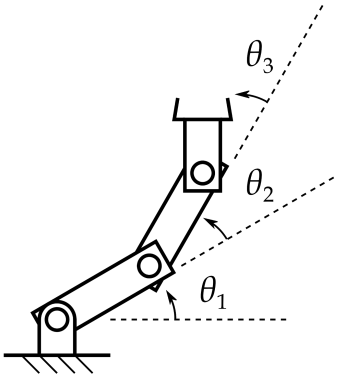


Fig 2.7 (b) *Angles of each link with preceding frame*

(x_e, y_e, ψ_e) are the position and orientation coordinates of end effector(tip of link 3). (θ_1, θ_2, θ_3) are the angle of each link with respect to each other as shown in the figure 2.7(b). Let first, second and third links lengths be L_1, L_2, L_3 respectively. By using plane geometry we can write following equations

Let $\cos(\theta_1) = C_1$; $\sin(\theta_1) = S_1$; $\cos(\theta_1+\theta_2) = C_{12}$; $\sin(\theta_1+\theta_2) = S_{12}$ and so on.

$$x_2 = L_1C_1$$

$$y_2 = L_1S_1$$

$$x_3 = L_1C_1 + L_2C_{12}$$

$$y_3 = L_1S_1 + L_2S_{12}$$

$$\theta_3 = \theta_1+\theta_2$$

$$x_e = L_1C_1 + L_2C_{12}+L_3C_{123} \qquad \text{Equation 2.10}$$

$$y_e = L_1S_1 + L_2S_{12}+L_3S_{123} \qquad \text{Equation 2.11}$$

$$\Psi_e = \theta_1+\theta_2+\theta_3 \qquad \text{Equation 2.12}$$

Let us take two variables K_1, K_2 such that

$$K_1 = x_e - L_3\cos(\theta_m)$$

$$K_2 = y_e - L_3\sin(\theta_m)$$

From Equations 2.10, 2.11 and 2.12; K_1 and K_2 can be written as

$$K_1 = L_1C_1 + L_2C_{12} \qquad \text{Equation 2.13}$$

$$K_2 = L_1S_1 + L_2S_{12} \qquad \text{Equation 2.14}$$

By squaring and adding Equations 2.13 and 2.14, we get

$$K_1^2 + K_2^2 = L_1^2 + L_2^2 + 2L_1L_2C_2$$

$$C_2 = (K_1^2 + K_2^2 - L_1^2 - L_2^2)/2L_1L_2$$

$$|\cos(\theta_2)| \leq 1$$

Also, $\sin(\theta_2) = \sqrt{(1-\cos(\theta_2)^2)}$

$$\theta_2 = \mathbf{Tan^{-1}(\sin(\theta_2)/\cos(\theta_2))}$$

$$\theta_1 = \mathbf{Tan^{-1}(K_2/K_1) - Tan^{-1}(L_2S_2 / L_1+L_2C_2)} \qquad \text{Equation 2.15}$$

From Equation 2.12,

$$\theta_3 = \Psi_e - \theta_1 - \theta_2 \qquad \text{Equation 2.16}$$

Hence joint angles (θ_1, θ_2, θ_3) can be found for given link lengths and orientation of the end effector.

2.5 Dynamic Model of a 3-DOF RRR Planar Manipulator

Lagrangian Dynamic model

Consider three links of manipulator of lengths L_1,L_2,L_3 have masses m_1, m_2, m_3 respectively located at a distance of L_{g1}, L_{g2}, L_{g3} from their preceding joint measured along the link. From section 2.4, we can see that these position can be written as a function of angles.

$$X_{g1} = L_{g1}C_1$$

$$Y_{g1} = L_{g1}S_1$$

$$X_{g2} = L_1C_1 + L_{g2}C_{12}$$

$$Y_{g2} = L_1S_1 + L_{g2}S_{12}$$

$$X_{g3} = L_1C_1 + L_2C_{12} + L_{g3}C_{123}$$

$$Y_{g3} = L_1C_1 + L_2C_{12} + L_{g3}C_{123} \qquad \text{Equation 2.17}$$

On differentiating Equation 2.17 with time, we get velocities $\dot{X}_{g1}$, $\dot{Y}_{g1}$, $\dot{X}_{g2}$, $\dot{Y}_{g2}$, $\dot{X}_{g3}$, $\dot{Y}_{g3}$

.

Kinetic energies are obtained as

$K.E_i = (½)m_i v_i^2 = (½)m_i(\dot{X}_{g1}^2 + \dot{Y}_{g1}^2)$ Equation 2.18

And Potential energy $P.E_i = m_i g Z_i$ Equation 2.19

Where, Z_i = height of manipulator plane from ground

Now,

Lagrangian, $L_i(\theta, \dot{\theta}) = \Sigma_{i=1}^{3} (K.E_i - P.E_i)$ Equation 2.20

To find torques

$$\tau_i = \frac{d}{dt}\left[\frac{\partial(Li(\theta, \dot{\theta}))}{\partial \dot{\theta} i}\right] - \left[\frac{\partial(Li(\theta, \dot{\theta}))}{\partial \theta i}\right]$$ Equation 2.21

General form of torque equations obtained is as follows

$\tau = M(\theta)\ \ddot{\theta} + N(\theta, \dot{\theta})$ Equation 2.22

$M(\theta)\ \ddot{\theta} \rightarrow$ Mass matrix

$N(\theta, \dot{\theta}) \rightarrow$ Mass matrix

Chapter 3.Structural Development

3.1 Overview

As discussed, this mobile robot has a base and a manipulator. Base is powered with 4 motors of same type and has 4 mecanum wheels at respective locations. Manipulator(3-DOF RRR) has 3 links (or arms) with 3 motors one at each joint connecting them. As this project focuses on displaying the control scheme budget is taken into consideration. As a result of these factors, a very simple design was developed which fulfilled requirements.

3.2 Base

Base need to accumulate space for 4 motors, Controllers , motor driver and electronic circuit. For this, it has metal plates(chassis as shown in fig 3.2) serving as platforms and aluminium columns holding them together. Lower platform has four motors at each corner and controller unit placed at the center as shown in figure 3.1 and the platform above serves as manipulator mounting.

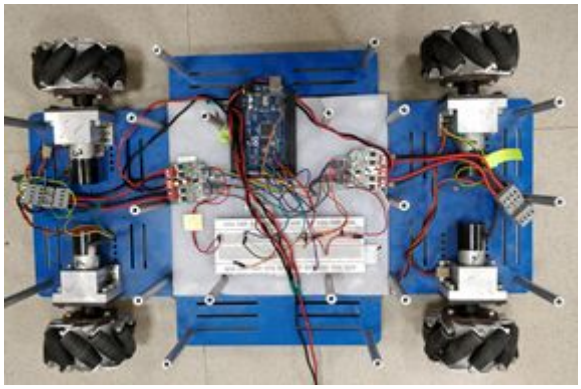


Fig 3.1 *Lower platform with motors and circuit*

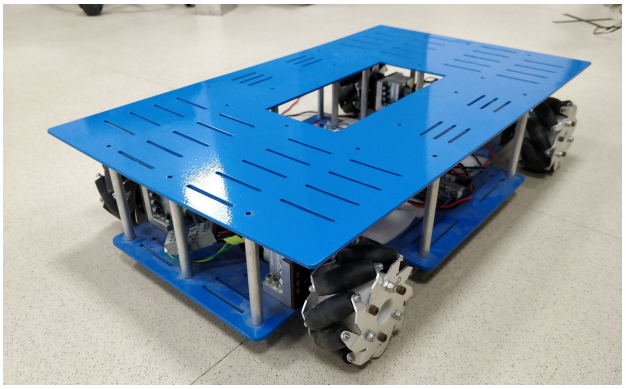


Fig 3.2 *Mobile Base*

3.3 Manipulator

Three links of 3-DOF RRR planar manipulator has lengths 40cm, 20cm, 10cm for first, second and third link(end effector link) respectively. Manipulator must have high bending stiffness in order to remain in a plane undisturbed during the base movement which might have slight vibrations due to mecanum wheels. Wood is the material that makes up the manipulator body(links) that have fairly high bending stiffness and are light in weight. Wood is economical better compared to aluminium also it weighs less and easy to manufacture. Joints are made up of a bioplastic Polylactic Acid(PLA) using additive manufacturing techniques(fig 3.3) i.e., 3D printing with the help of Prusa plus 3D printer.

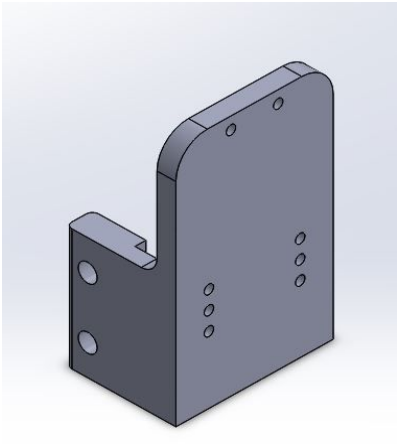


Fig 3.3 *3D printed PLA part for joint*

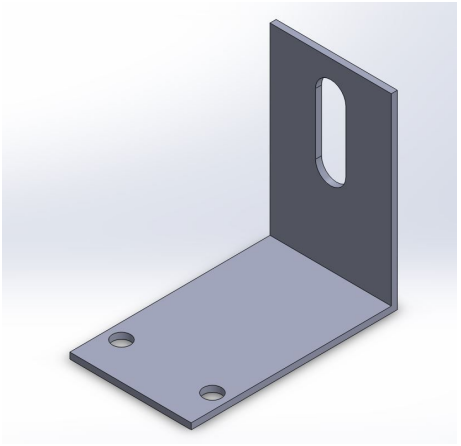


Fig 3.4 *Metal bracket*

Rotary sensor is mounted using a metal bracket (as shown in fig. 3.4) made of 2mm thick mild steel sheet fitted to PLA part. Motor is mounted to PLA joint using metal bracket that is designed to hold motor and align its shaft with shaft of the rotary sensor which is mounted on the top using metal bracket. In order to avoid damaging motor shaft, A castor wheel arrangement is made at the first motor in order to take the compressive load coming from the manipulator due to its weight. Overall design assembly of the mobile robot is shown in the figure 3.5

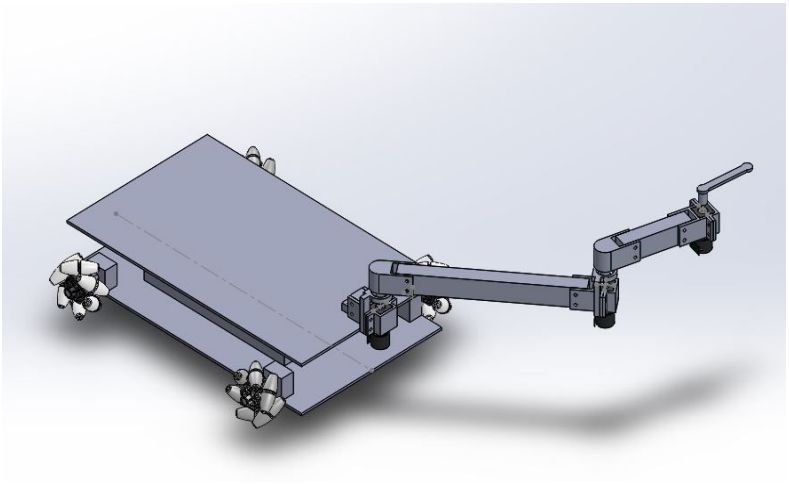


Fig 3.5 *Mobile Robot with Mecanum wheel base and 3-DOF RRR manipulator*

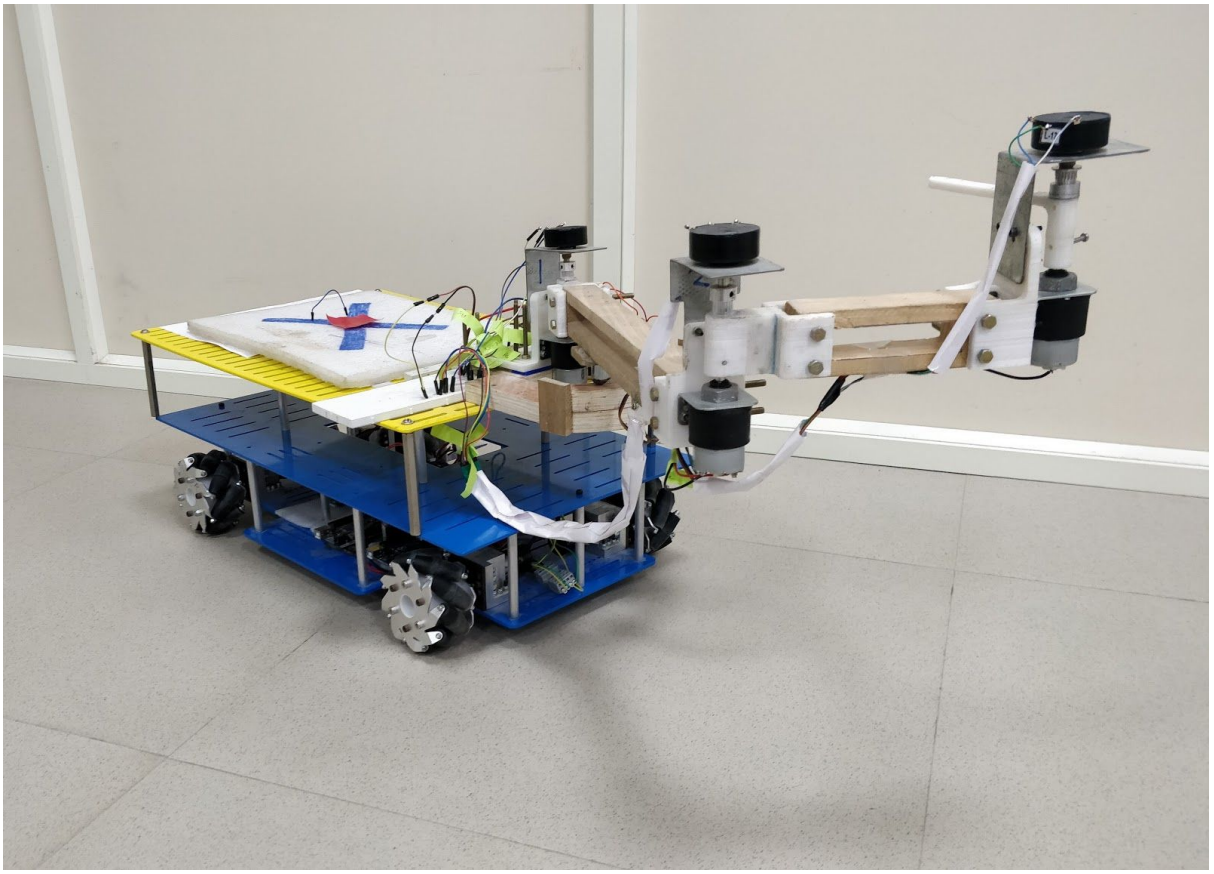


Fig 3.6 *In house fabricated Mobile Robot prototype*

Chapter 4. Electronic System Design

4.1 Overview

For the mobile robot to run and operate, it totally has 7 motors (4 for the base and 3 for the manipulator) and so it need 7 motor drivers. As it is planned to implement a separate control for the base and the manipulator two microcontrollers are used. A breadboard and jumper wires make up the rest of circuit. To power the entire mobile robot SMPS is used during testing and experimental phase, batteries can be used for long run.

4.2 Controller

4.2.1 Arduino Mega 2560

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it

with a AC-to-DC adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for the Uno and the former boards Duemilanove or Diecimila.

Table 4.1 Arduino Mega 2560 Specifications

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

Controller Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(),digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50 k ohm. A maximum of 40mA is the value that must not be exceeded to avoid permanent damage to the microcontroller.

In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX). Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega16U2 USB-to-TTL Serial chip.

- External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2). These pins can be configured to trigger an interrupt on a low level, a rising or falling edge, or a change in level. See the attachInterrupt() function for details.

- PWM: 2 to 13 and 44 to 46. Provide 8-bit PWM output with the analogWrite() function.

- SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). These pins support SPI communication using theSPI library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Arduino /Genuino Uno and the old Duemilanove and Diecimila Arduino boards.

- LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

- TWI: 20 (SDA) and 21 (SCL). Support TWI communication using the Wire library. Note that these pins are not in the same location as the TWI pins on the old Duemilanove or Diecimila Arduino boards.

The Mega 2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

- AREF. Reference voltage for the analog inputs. Used with analogReference().

- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Programming

The Arduino Mega 2560 can be programmed and controlled with open source Arduino Software (IDE) or high performance technical computing software like MATLAB.

4.2.2 Arduino UNO R3

Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller.

Table 4.2 Arduino Uno R3 Specifications

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V

Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Controller Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50k ohm. A maximum of 40mA is the value that must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller.

In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.

- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.

- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.

- LED: 13. There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

- TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. There are a couple of other pins on the board:

- AREF. Reference voltage for the analog inputs. Used with analogReference().

- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Programming

Similar to Arduino Mega 2560

4.3 Motors

GEARED DC MOTOR 12V 200RPM

- 200 RPM 12V DC motors with Metal Gearbox and Metal Gears
- 18000 RPM base motor
- 6mm Dia shaft with M3 thread hole
- Gearbox diameter 37 mm.
- Motor Diameter 28.5 mm
- Length 63mm without shaft
- Shaft length 30mm Motor
- 170gm weight
- 32 kg cm torque
- No-load current : 800mA, Load current : upto 7.5 A(Max)

GEARED DC MOTOR 12V 10RPM

- 10 RPM 12V DC motors with Gearbox
- 12V DC Supply
- 6mm shaft diameter with threaded hole & screw
- 5 kg cm torque
- No-load current = 60 mA(Max), Load current = 250 mA(Max)

4.4 Battery

LI-ION 11.1V 4400MAH

- Very Small in size and weight(270g) compared to Ni-Cd, Ni-MH and Lead Acid Batteries
- Discharge current upto 8A
- Full Charge in 40 to 90 minutes depending upon special charger
- Long life with full capacity for upto 1000 charge cycles
- 6X Li-ion 3.7V 2200mAh cells (3S2P)
- Low maintenance
- Inbuilt charge protection circuit

4.5 Sensors

4.5.1 Rotary potentiometer

The potentiometer, commonly referred to as a “pot”, is a three-terminal mechanically operated rotary analogue device which can be found and used in a large variety of electrical and electronic circuits. They are passive devices, meaning they do not require a power supply or additional circuitry in order to perform their basic linear or rotary position function.

The name “potentiometer” is a combination of the words Potential Difference and Metering, which came from the early days of electronics development. It was thought then that adjusting large wire wound resistive coils metered or measured out a set amount of potential difference making it a type of voltage-metering device.

Rotary potentiometer (the most common type) vary their resistive value as a result of an angular movement. Rotating a knob or dial attached to the shaft causes the internal wiper to sweep around a curved resistive element as shown in figure 4.1. The most common use of a rotary potentiometer is the volume-control pot.

Rotary potentiometers can produce a linear or logarithmic output with tolerances of typically 10 to 20 percent. As they are mechanically controlled, they can be used to the measure the rotation of a shaft, but a single-turn rotary potentiometer normally offers less than 300 degrees of angular movement from minimum to maximum resistance. However, multi-turn potentiometers, called trimmers, are available that allow for a higher degree of rotational accuracy.

Multi-turn potentiometers allow for a shaft rotation of more than 360 degrees of mechanical travel from one end of the resistive track to the other. Multi-turn pots are more expensive, but very stable with high precision used mainly for trimming and precision adjustments.

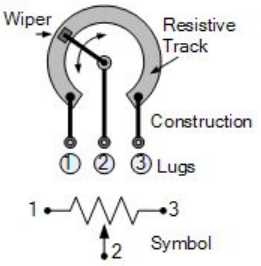


Fig 4.1 *Rotary Potentiometer Construction*

Table 4.3 Potentiometer specifications

Output type	Linear
Angular movement	270^0
Coil resistive value	100K□
Shaft type	Metal
Shaft dia	6mm

4.5.2 Xbox Kinect V1

Kinect is a line of motion sensing input devices that was produced by Microsoft for Xbox 360 and Xbox One video game consoles and Microsoft Windows PCs. Based around a webcam-style add-on peripheral, it enables users to control and interact with their console/computer without the need for a game controller, through a natural user interface using gestures and spoken commands.

It has wide range of applications in Gaming, Health Care, Fitness, Medicine, VR, AR, Robotics etc. In this it is used to track the motion of the mobile robot in two dimensional

plane. It has various components like Color sensor, Depth sensor and Audio sensor as shown in figure 4.2.

Color sensor

- Common camera sensor to acquire RGB images

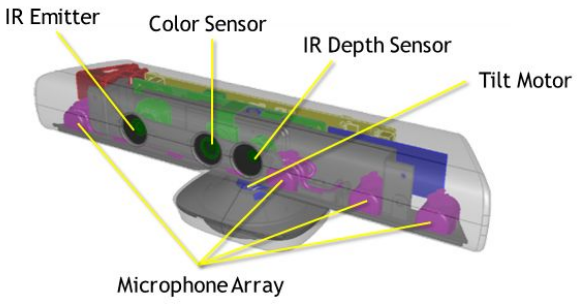


Fig 4.2 *Kinect V1 Components*

Depth camera

- IR emitter + IR sensor use infrared light to acquire depth images
- Uses time of flight

Audio sensor

- Microphone array to capture sound signal and sound position

Table 4.4 Xbox Kinect V1 Specifications

Frame Rate	30 FPS
Color stream	640 x 480
Depth stream	320 x 240
Depth distance	0.4m - 4m
Horizontal field view	57^0
Vertical field view	43^0
No.of Skeleton joints trackable	20
No.of bodies trackable	2 (6 recognizable)

4.5.3 Inertial Measurement Unit (IMU)

MPU 9250

MPU-9250 is a multi-chip module (MCM) consisting of two dies integrated into a single QFN package. One die houses the 3-Axis gyroscope and the 3-Axis accelerometer. The other die houses the AK8963 3-Axis magnetometer. Hence, the MPU-9250 is a 9-axis Motion Tracking device that combines a 3-axis gyroscope, 3-axis accelerometer, 3-axis magnetometer and a Digital Motion Processor (DMP) all in a small 3x3x1mm package available as a pin-compatible upgrade from the MPU6515. With its dedicated I2C sensor bus, the MPU-9250 directly provides complete 9-axis MotionFusion output. The MPU-9250 Motion Tracking device, with its 9-axis integration, on-chip MotionFusion, and runtime calibration firmware. For precision tracking of both fast and slow motions, the parts feature a user-programmable gyroscope full-scale range of ±250, ±500, ±1000, and ±2000°/sec (dps), a user-programmable accelerometer full-scale range of ±2g, ±4g, ±8g, and ±16g, and a magnetometer full-scale range of ±4800μT.

For this position and orientation tracking of the mobile robot only Accelerometer and Gyroscope are required.

Accelerometer

The accelerometer provides acceleration information and movement data along the axes. Accelerometers can also provide tilt data which can supplement the gyroscope output for accurate drift-free angular measurement. Specifications the triple-axis MEMS accelerometer in MPU-9250 are listed below:

- Digital-output triple-axis accelerometer with a programmable full scale range of ±2g, ±4g, ±8g and ±16g and integrated 16-bit ADCs
- Accelerometer normal operating current: 450μA
- Low power accelerometer mode current: 8.4μA at 0.98Hz, 19.8μA at 31.25Hz

Gyroscope

Gyroscope is primarily used to measure angular displacement. Specifications of the triple-axis MEMS gyroscope are listed below

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full scale range of ±250, ±500, ±1000, and ±2000°/sec and integrated 16-bit ADCs
- Digitally-programmable low-pass filter
- Gyroscope operating current: 3.2mA

4.6 Motor Driver Board

DC Motor Driver (RKI-1340) is used to control speed and direction of motors. It also has a braking feature that can immediately halt the shaft of motors even in most high power applications. Other characteristics of board are as follows

Electrical Characteristics

- Input Voltage: 7V minimum to 30V maximum
- Continuous Current (< 1 second) - 20A Continuous
- Current (< 10 seconds) - 10A
- Continuous Current (> 10 seconds) - 5A
- Absolute Maximum Peak Current - 50A

Control IO Description

- GND – connect to GND on controlling board
- DIR – Pulled down to GND Forward by default and Backward when 5V (logic high)
- PWM – Pulse Width Modulation input to control speed of motor (recommended freq 20 Hz to 400Hz)
- BRK – braking input to halt the motor in operations when 5V (logic high)
- 5V – regulated 5V output from motor driver board (maximum 50mA supply)

Chapter 5. Control Scheme and Programming Logic

5.1 Conventional vs Resolved control

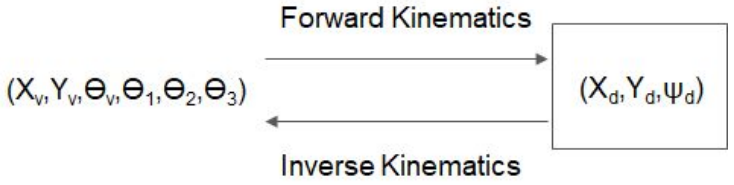
5.1.1 Conventional control

For a 3-DOF RRR planar mobile manipulator, we have three operational space variables to achieve with end effector i.e., point of interest or desired point (X_d, Y_d, ψ_d) where, ψ_d is desired orientation.

In conventional control, on total we have 6 variables to control in which three are position and orientation of the base (X_v, Y_v, Θ_v) and other three are orientation of each link of the manipulator $(\Theta_1, \Theta_2, \Theta_3)$. When forward kinematics is considered, three desired coordinates are function of 6 controllable coordinates as follows

$$\begin{bmatrix} X_d \\ Y_d \\ \psi_d \end{bmatrix} = f \begin{bmatrix} X_v \\ Y_v \\ \Theta_v \\ \Theta_1 \\ \Theta_2 \\ \Theta_3 \end{bmatrix} \eta$$

Task-Space, ρ



In Inverse kinematics, we have six controllable parameters as inputs i.e.,(X_v,Y_v,Θ_v,Θ_1,Θ_2,Θ_3) and three Desired variables(X_d,Y_d,ψ_d) as output.

In forward Kinematics, from

$\rho = f(\eta)$

$\rho = J \times \eta$ Equation 5.1

In inverse Kinematics, Equation 6.1 can be written as

$\eta = J^{-1} \times \rho$ Equation 5.2

Where,

Order of ρ = (3x1)

Order of η = (6x1)

And hence, order of J is (3x6)

Inverse of a rectangular matrix is to be calculated in inverse kinematics in order to calculate inputs i.e, 'η', for which Moore-Penrose Pseudo Inverse theory is used which results in instability and also needs high-performance microprocessors in long run.

Also conventional control, the concept of the workspace being dexterous is not considered (planar case), the number of desired coordinates are three (2 positions and 1 orientation) and the number of controllable parameters are 6 (3 vehicle and 3 manipulator). This results in kinematic redundancy thereby giving rise to an infinite number of possible solutions, a great many of which might be non-dexterous. This shows that every orientation at desired coordinates is not possible and mobile robot needs to change its position for a new orientation at the same desired point by selecting a set of inputs from many possible solutions for new desired orientation.

5.1.2 Resolved control

In resolved control scheme, the idea is to position the mobile robot such that the point of interest falls into the dexterous workspace of the manipulator with respect to the base. This can be achieved by controlling the base and the manipulator separately. This separate control removes the kinematic redundancy of the robot as the base now has 3 desired parameters (x_{vd}, y_{vd}, Θ_{vd}) and 3 controllable parameters (x_v, y_v, Θ_v). Further, the manipulator too will have 3 desired parameters (x_d, y_d, Θ_d) and 3 controllable parameters (Θ_1, Θ_2, Θ_3).

Dexterous workspace for the link taken lengths is a ring of inner diameter 30mm and outer diameter of 50mm from the first joint fixed at the base. For simplicity, consider that mobile robot always face the point in forward direction and base will be in a position that first joint 40mm away from the final desired point.

5.2 MATLAB Environment

MATLAB is a multi-paradigm numerical computing environment and proprietary programming language. It is used for a range of applications like signal processing and communications, image and video processing, control systems, test and measurement etc. In this project Image processing and Simulink are used along with arduino libraries to control the mobile manipulator.

5.2.1 Coding in Editor

Editor window allows write a program or script called a m-file (has a .m extension). The Editor Window is a word processor specifically designed for Matlab

commands so it automatically executes specific commands that are included in code. In order to give commands to microcontroller(arduino), an arduino support package is installed. This help in communication with arduino board using a USB cable. To write a value into pins, respective pins are declared and commands are coded using digitalWrite() or anologWrite() functions. Similarly syntax for other commands will be obtained along with the package installed.

5.2.2 Simulink

Simulink is an application in MATLAB, that provides an interactive, graphical environment for modeling, simulating, and analyzing of dynamic systems. It enables rapid construction of virtual prototypes to explore design concepts at any level of detail with minimal effort. For modeling, Simulink provides a graphical user interface (GUI) for building models as block diagrams. It includes a comprehensive library of predefined blocks to be used to construct graphical models of systems using drag-and-drop mouse operations. The user is able to produce an "up-and-running" model that would otherwise require hours to build in the laboratory environment. It supports linear and nonlinear systems, modeled in continuous-time, sampled time, or hybrid of the two.

Arduino variables are available as blocks in Simulink library to read and write, digital or analog values, from or to pins. Control in matlab runs in loop till specified time and helps in executing closed loop control by reading and writing values from and to arduino respectively.

5.2.3 Image Acquisition Toolbox

Image Acquisition Toolbox provides functions and blocks that enable us to connect industrial and scientific cameras to MATLAB. It includes a MATLAB app that lets you interactively detect and configure hardware properties. The toolbox enables acquisition modes such as processing in-the-loop, hardware triggering, background acquisition, and synchronizing acquisition across multiple devices. This project uses Xbox Kinect v1 as a camera along with Image acquisition toolbox for live image tracking.

5.3 Control Theory

PD control

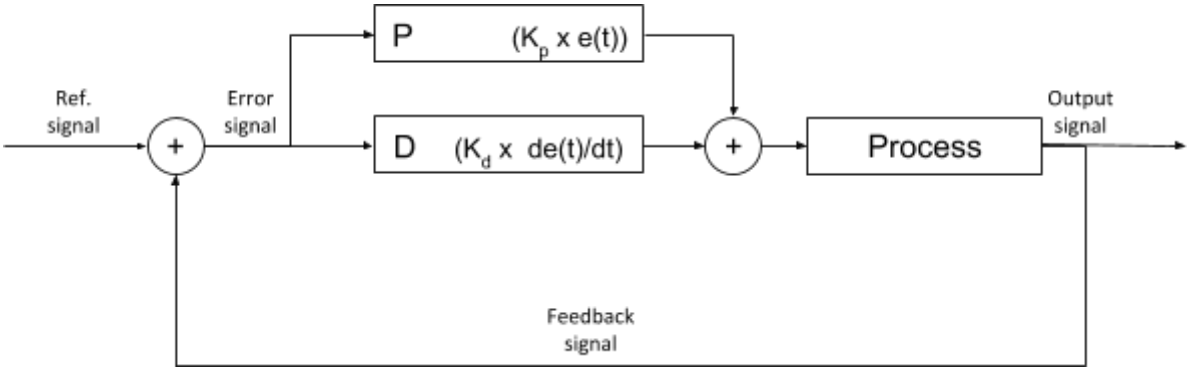


Fig 5.1 *PD logic*

Proportional (P)

The command is proportional to how much the current value is away from the desired value i.e., proportional to the value of error. It helps in pushing the current angle (or position) to desired angle (or position). K_p is proportional gain.

Derivative (D)

The command is proportional to the rate of change of the error i.e to how fast the error signal is raising or shrinking. In other words, it predicts the future and responds to how fast we are approaching the goal and and gives a value to slow down or speed up the process to reduce overshooting. K_d is derivative gain

Mass Damper Analogy

It can be assumed as similar case as adding spring and damper(shock absorber) to a mass. The spring stiffness K_p is proportional gain. The damping rate K_d is the derivative gain.

- Increasing K_P pushes the mass to mean value faster, but also results in more overshoot and oscillations
- Increasing K_d slows down the rate and dampens out oscillations

Tuning a system for most suited K_p and k_d value results in reaching a goal with optimum speed and very less overshoot.

5.4 Feedback sensors programming for position and orientation

5.4.1 Orientation Data

As mentioned in specification, in this project we are using rotary potentiometer to measure the angle of each link with respect to preceding link(i.e, joint angles). This potentiometer produces linear outputs and communicates to program through analogread pins of arduino.

Potentiometer terminal pins are connected to 5V and Gnd pins on the arduino board and wiper pin is connected to analogread pin. Each pot has 270^0 of range and returns an integer between 0-1023 using analogRead(), for any position in its range. Hence scaling factor is calculated as 270/1024 and is multiplied with current count of the potentiometer to read current angle to keep live tracking throughout the experiment.

5.4.2 Position Data

(i) Image tracking for position

Xbox kinect v1 is mounted above the working area in order to capture frames continuously and track an object of particular color mounted on top of the mobile manipulator which is in XY plane. To track position coordinates of a red object following steps are to be followed

- Connect the cam and specify the format (format = 'RGB_1280x960').
- Capture the video frames by defining an object using the video input function(videoinput('kinect', 1, format)).
- Set the properties of the video object like Frames per trigger, Colorspace and Frame grab interval.
- Set a loop and start the video acquisition. Get snapshot of current frame into a variable data = getsnapshot(vid).
- Now to track red objects in real time, we have to subtract the red component

- From the grayscale image to extract the red components in the image use diff_im, [3 3] = imsubtract(data(:,:,1), rgb2gray(data)) i.e extracting the data from first of three layers of the RGB matrix.
- To filter out the noise, use a median filter (medfilt2(diff_im, [3 3]))
- Convert the resulting grayscale image into a binary image based on a threshold level using im2bw(diff_im,0.2)
- To remove any smaller objects that are in frame, remove all those objects that are less than 100px diff_im = bwareaopen(diff_im,100).
- Label all the connected components left in the image using bwlabel() function
- And find the centroid of the object using regionprops() function. It returns X and Y coordinates of the centroid of the object in the frame.
- This will be running in a loop and continuously feeding the live position data to Matlab. *(all the samples for these steps are shown in Appendix-B)*

We can also find orientation of the mobile robot about Z axis by placing another object of different color (e.g. blue) and finding coordinates of the both the centroids to find slope between those points there finding the angle.

(ii) Accelerometer (in IMU)

IMU is mounted on the mobile robot according to the axes specified on it. Libraries of MPU 9250 are available for Matlab and are used in reading data from IMU continuously through Tx and Rx pins.

Accelerometer of IMU returns acceleration data along respective axes in terms of 'g'. Integrating this acceleration data over time gives velocity, which on further integration with respect to time gives the position data along a particular axis. But, accelerometer data is continuously fluctuating due to some imparted noise. This continuous fluctuating noise upon integration adds up to give wrong position information i.e., even when our robot is at rest.

So, it is suggested to use and IMU along with another position sensor like GPS(which lacks in resolution) which will be continuously correcting the IMU data between particular intervals(its resolution). IMU also have gyroscope which give information about angular displacement or orientation

Owing to these limitations, image processing techniques are used in order to get position information of base as feedback.

5.5 Closed loop Mobile base control scheme (for position tracking using resolved motion)

In this section we are interested in investigating the resolved control scheme. So, here we are doing tracking position of the mobile base i.e., a fixed L shaped path(along X and Y) is selected in order to do so.

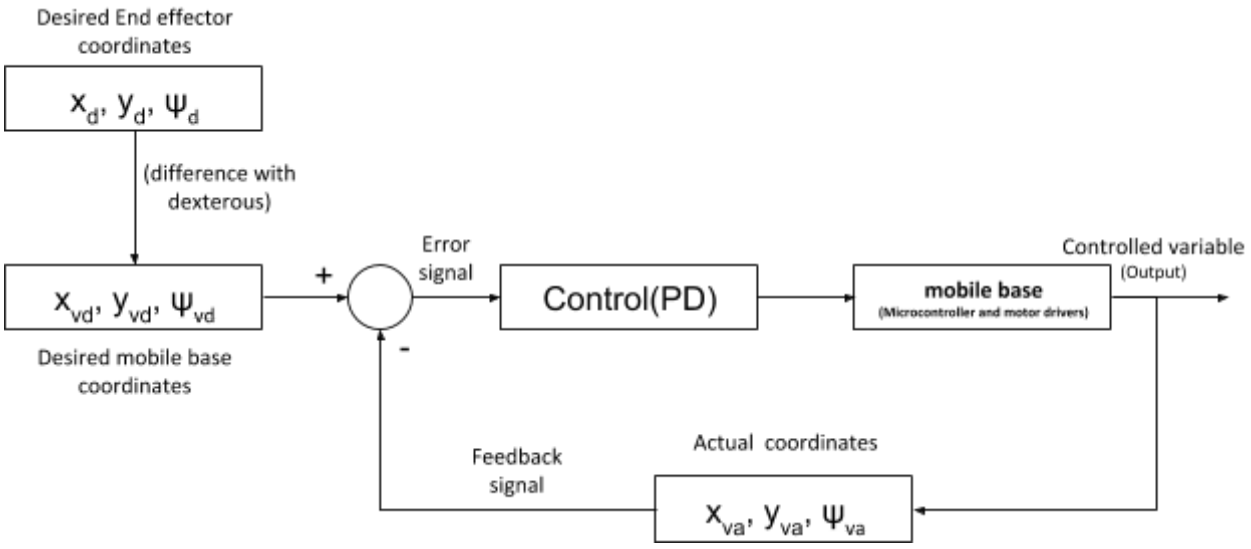


Fig 5.2 *Closed loop control logic for mobile base*

Desired position and orientation coordinates where the end effector point has to be placed are taken as input from user. And tracking is done with image processing using red object placed on top of mobile base and is done as follows

- Position of mobile base will be 40 cm away from the first joint mounted on the base. So, calculate distance between red object placed on the base and first joint(motor) placed on the base.
- For simplicity, orientation is always maintained such that robot will always face the final desired point (x_d, y_d, ψ_d) with its side where manipulator is mounted(Front side).

- Red object is placed at centroid of base and hence Y- coordinate of both desired end effector point(x_d, y_d, ψ_d) and desired mobile base position (x_{vd}, y_{vd}, ψ_{vd}) are same
- Now x_{vd} is found by calculating difference between x_d and sum of centroid distance of mobile base from joint one and 40 cm.
- (x_{vd}, y_{vd}, ψ_{vd}) are obtained and are subtracted from actual coordinates and sent to PD control block to get direction and PWM outputs.
- These outputs from control block are sent to arduino from Matlab and there by arduino communicates with motor drivers that drives the wheels accordingly.
- Mobile robot will start moving and actual coordinates of base and obtained with Xbox kinect image sensor and are feeded back. These actual value are continuously subtracted from desired value and that error signal is continuously passed into loop until the error becomes zero i.e., base reaches (x_{vd}, y_{vd}, ψ_{vd})

5.6 Closed loop Manipulator Control Scheme

As we already positioned the mobile base at the required point(x_{vd}, y_{vd}, ψ_{vd}), we now control the manipulator from that position to reach the final desired point (x_d, y_d, ψ_d) with end effector.we can track position with any random end effector path or with a specified trajectory. Both the procedures are discussed below

5.6.1 Simple Position Tracking

We have two ways to achieve a particular position. One method deals with configuration space variable and is called Joint-space control. The other method deals with operational space variables and is called Task-space control. Control based on both these methods are discussed below

5.6.1 (a) Joint-Space Control Loop

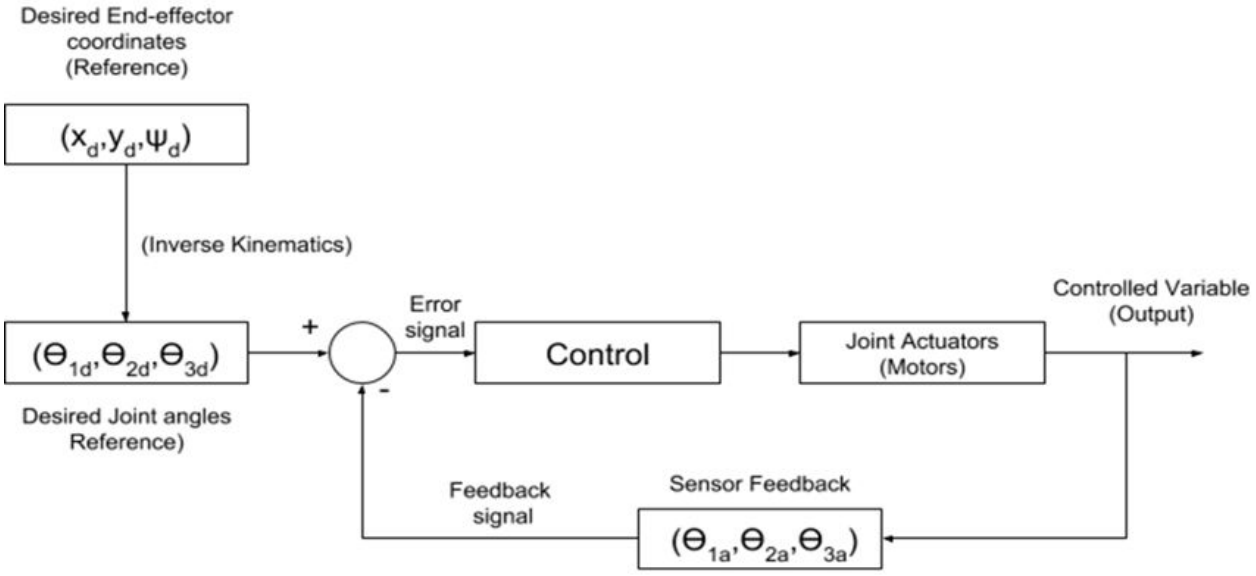


Fig 5.3 *Joint-space closed loop control logic*

Above loop represent closed loop control to achieve a particular point with end effector by taking desired coordinates as input from user. Control in closed loop goes as follows

- During mobile base control, desired coordinates for end effector (x_d, y_d, ψ_d) are taken as input. From that point all the three desired angles(θ_{1d}, θ_{2d}, θ_{3d}) can be calculated from inverse kinematics as shown in section 2.4
- We have desired joint angles (θ_{1d}, θ_{2d}, θ_{3d}) and this angles will be subtracted from actual angles(θ_{1a}, θ_{2a}, θ_{3a}) that manipulator links are currently oriented. This is the error signal and this is sent to control block(PD) to calculate analog output value(PWM). This is communicated to microcontroller from Simulink and it communicates with motor driver boards according to the commands specified.
- Motors are driven by drivers and orientation changes continuously which will be measured with the help of rotary potentiometers mounted to the motor shaft and will continuously update the current angles as feedback(θ_{1a}, θ_{2a}, θ_{3a})
- This loop continues till error signals becomes Zero i.e., desired angles(θ_{1d}, θ_{2d}, θ_{3d}) are obtained or desired coordinates for end effector(x_d, y_d, ψ_d) is obtained.

Joint space control is effective in achieving the position and orientation, but the problem arises when there is some error in link lengths(Measuring or manufacturing or assembly) as it only controls the angles.

5.6.1 (b) Task-Space Control Loop

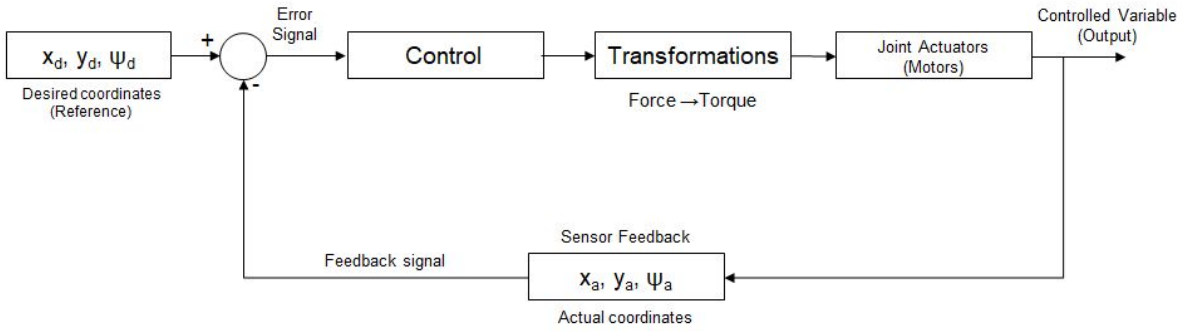


Fig 5.4 *Task-space closed loop control logic*

Above loop represent closed loop control to achieve a particular point with end effector by taking desired coordinates as input from user. Control in closed loop goes as follows

- Desired coordinates for end effector (x_d, y_d, ψ_d) are taken as input and is subtracted from actual position and orientation coordinates (x_a, y_a, ψ_a) to generate an error signal
- This error signal is sent to PD control block and corresponding linear force signals at each link. Which further transformed into torques for each motor using Jacobian matrix and is communicated to the microcontroller which controls motor driver accordingly to power motors.
- Actual position of end effector keeps on getting closer to the desired value and continuous feedback is obtained with the help of Xbox kinect working as image processing sensor.
- This loop continues till error signals becomes Zero or desired coordinates for end effector(x_d, y_d, ψ_d) is obtained.

Task-space helps in achieving desired point with more accuracy as it focuses only on making taskspace error zero rather than focusing on configuration space variables. But the problem arises when starting angular position of links are (0,0,0) because inverse of jacobian matrix doesn't exists at the point. However it overcomes this issue when started at any other angles.

Chapter 6. Experiments

6.1 Manipulator Position Tracking

As mentioned in earlier sections, block coding for manipulator is done in simulink and desired position coordinates are taken as (0,40) i.e., in dexterous space exactly on the same x-coordinate as the mobile base. And a random orientation of 125^0 is given and performance of manipulator is noted down as shown in figures below.

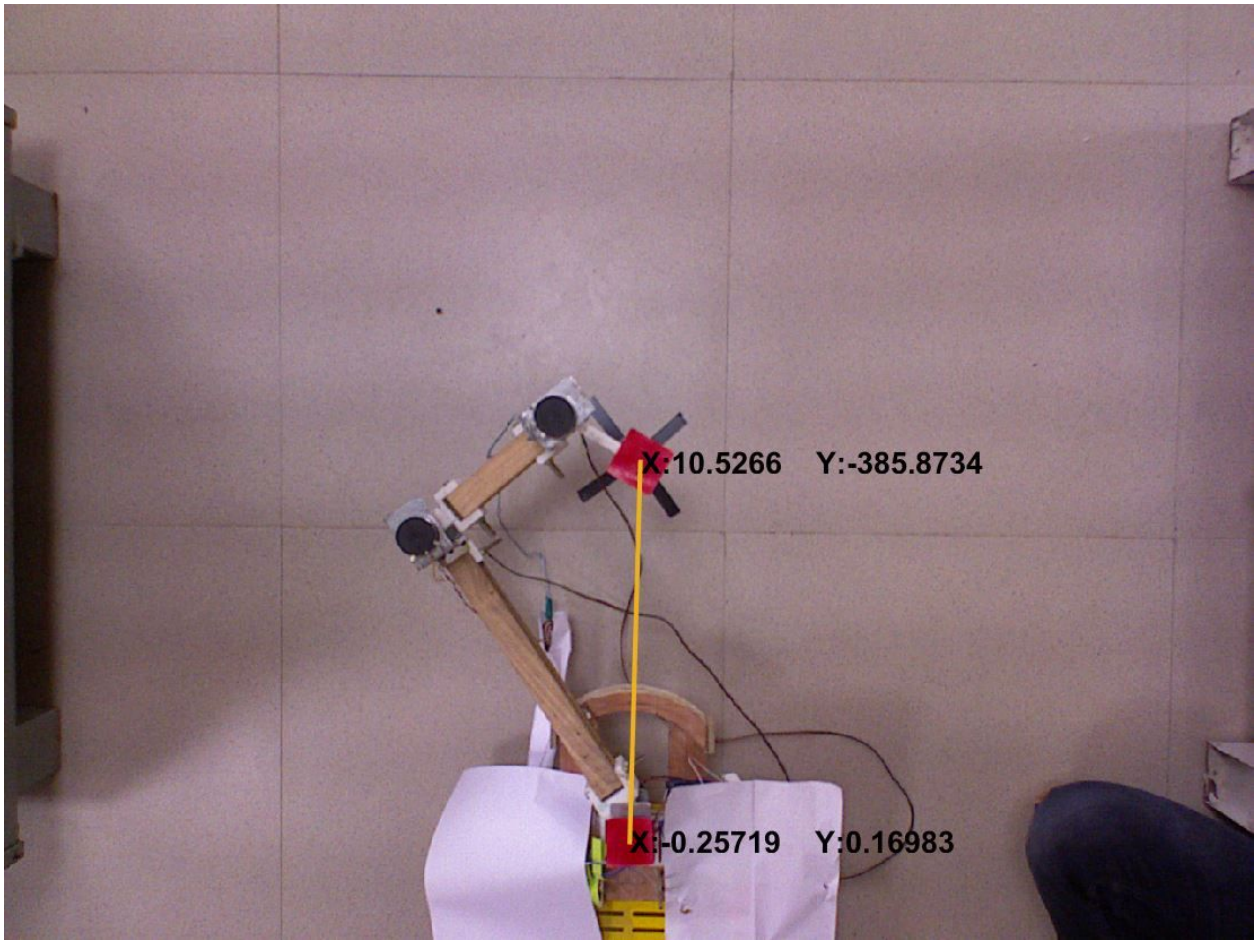


Fig 6.1 *Position Tracking performance of Manipulator*

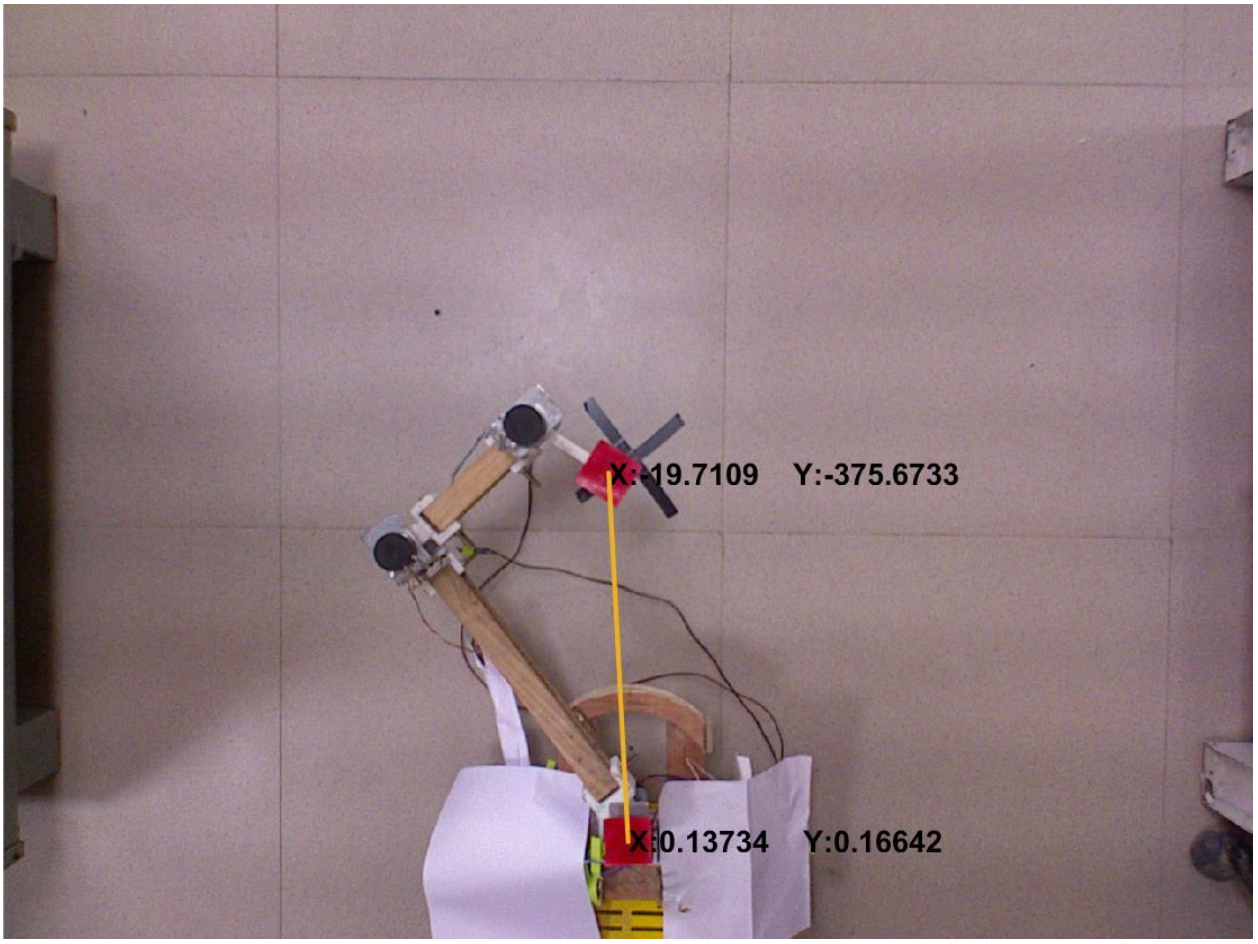


Fig 6.2 *Position Tracking performance of Manipulator*

6.2 Mobile Base Position Tracking

Target of the experiment is to position the mobile base at its desired coordinates given as input by the user. Red object is placed on top of the mobile base at its center. For simplicity, an L - shaped predefined path is given as our focus here is on tracking final position without changing orientation of the base. Experiments are performed for various coordinates and performances are recorded and are shown in figures 6.3,6.4 and 6.5.

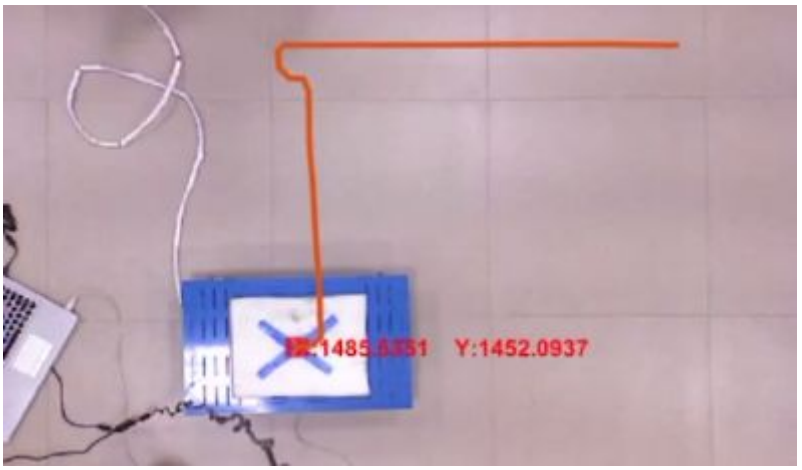


Fig 6.3 *Position Tracking performance of mobile base for the point (150cm, 150cm)*

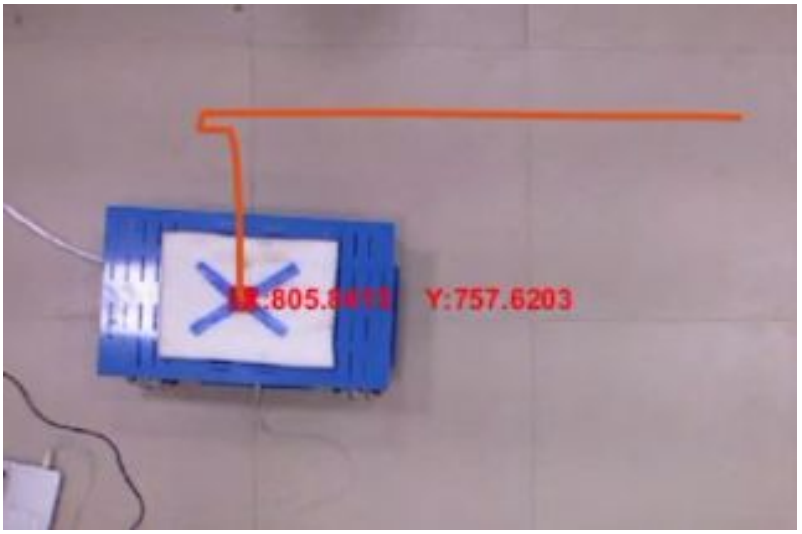


Fig 6.4 *Position Tracking performance of mobile base for the point (80cm, 80cm)*

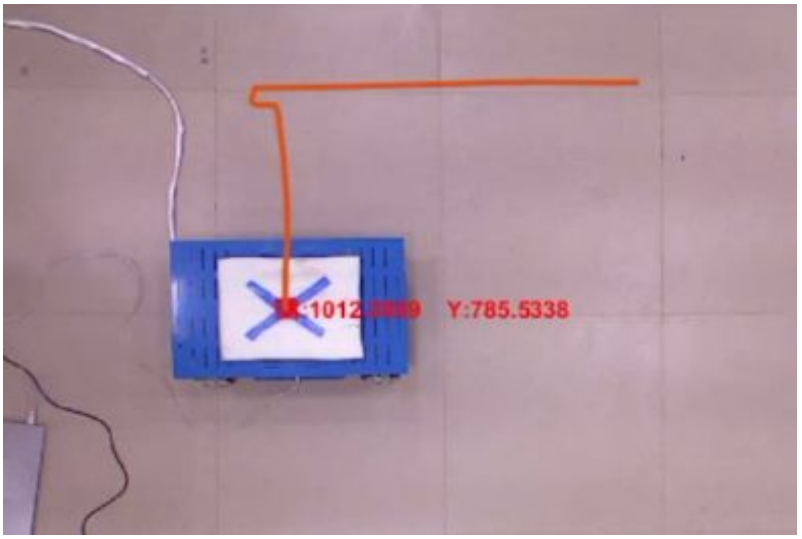


Fig 6.5 *Position Tracking performance of mobile base for the point (100cm, 80cm)*

6.3 Mobile Robot Position Tracking

In this experiment, both the tasks i.e, locomotion and manipulation are performed simultaneously. Only final desired point of the end effector(which is center of the black object as shown in fig. 6.7) is taken as input from the user. Mobile base calculated its desired position and reached that point as shown in fig. 6.6 and then manipulator calculated its desired angles and reached final desired position with its end effector. Performances are recorded and a sample is shown below

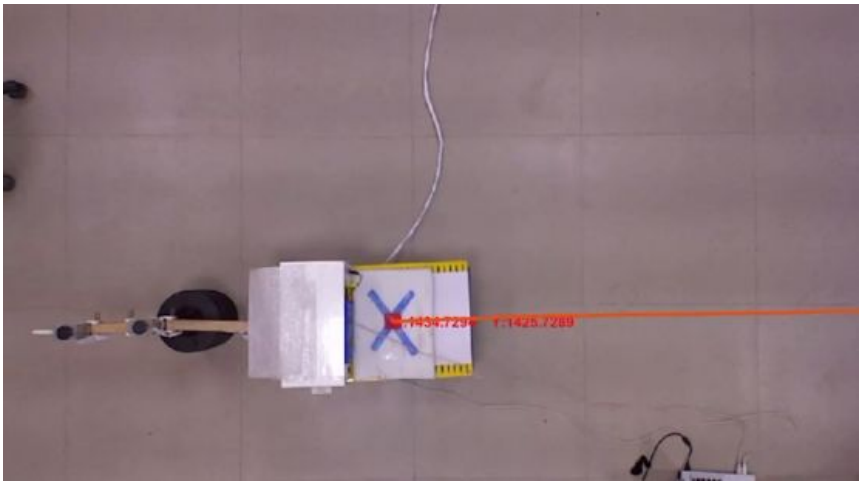


Fig 6.6 *Position Tracking performance of mobile robot showing center coordinates*

Fig 6.7 *Position Tracking performance of mobile robot showing end effector*

6.4 Manipulator Trajectory Tracking Simulations

Other than position tracking experiments, manipulator trajectory tracking simulations are also performed in which manipulator is given an ellipse to track and data is plotted for both Task-space and Joint-space as shown below.

Task-Space

Critically Damped system(damping ratio ξ=1)

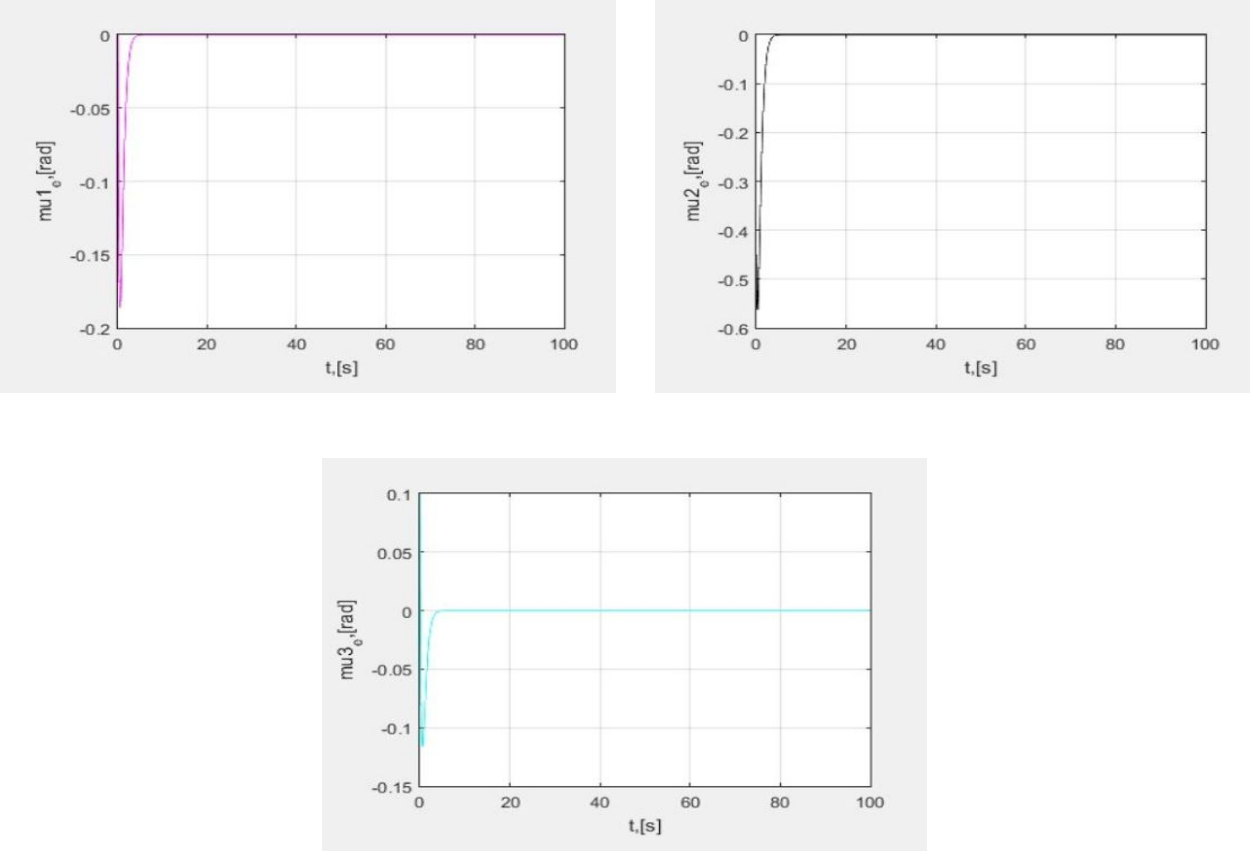
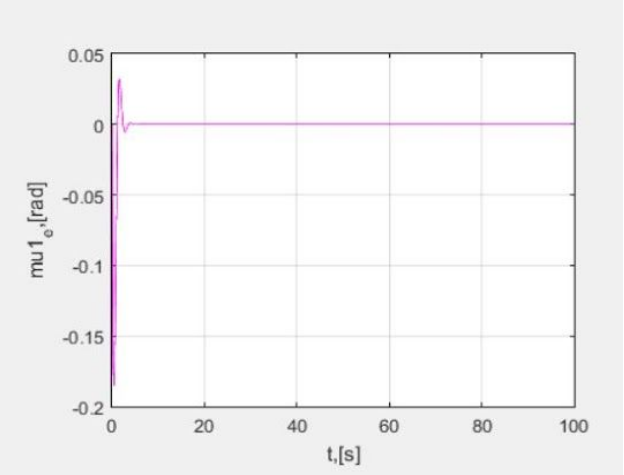
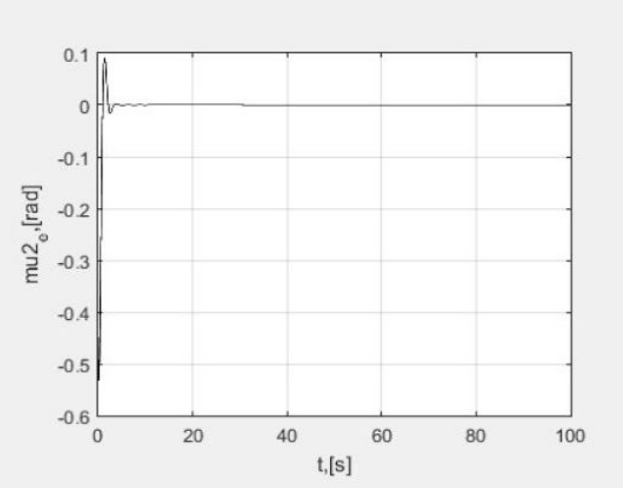


Fig 6.8 *Graphs showing errors in position(x_t,y_t) i.e, mu_{te} of each link with time for Critically Damped system*

Underdamped system (ξ=0.5)





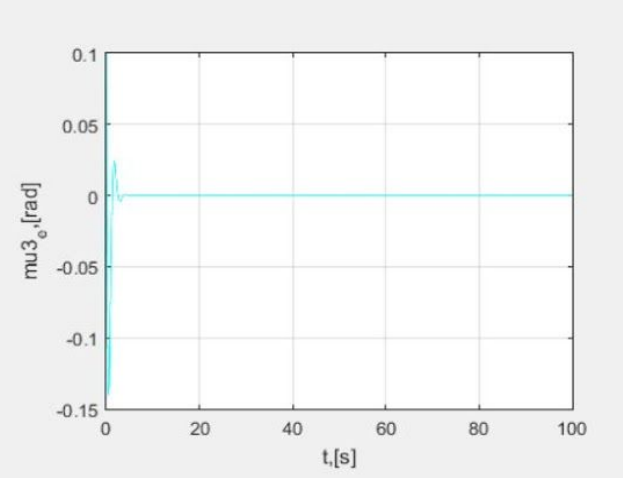


Fig 6.9 *Graphs showing errors in position(x_e,y_e) i.e, mu_{ie} of each link with time for Underdamped system*

Joint-Space

Critically damped (ξ=1)

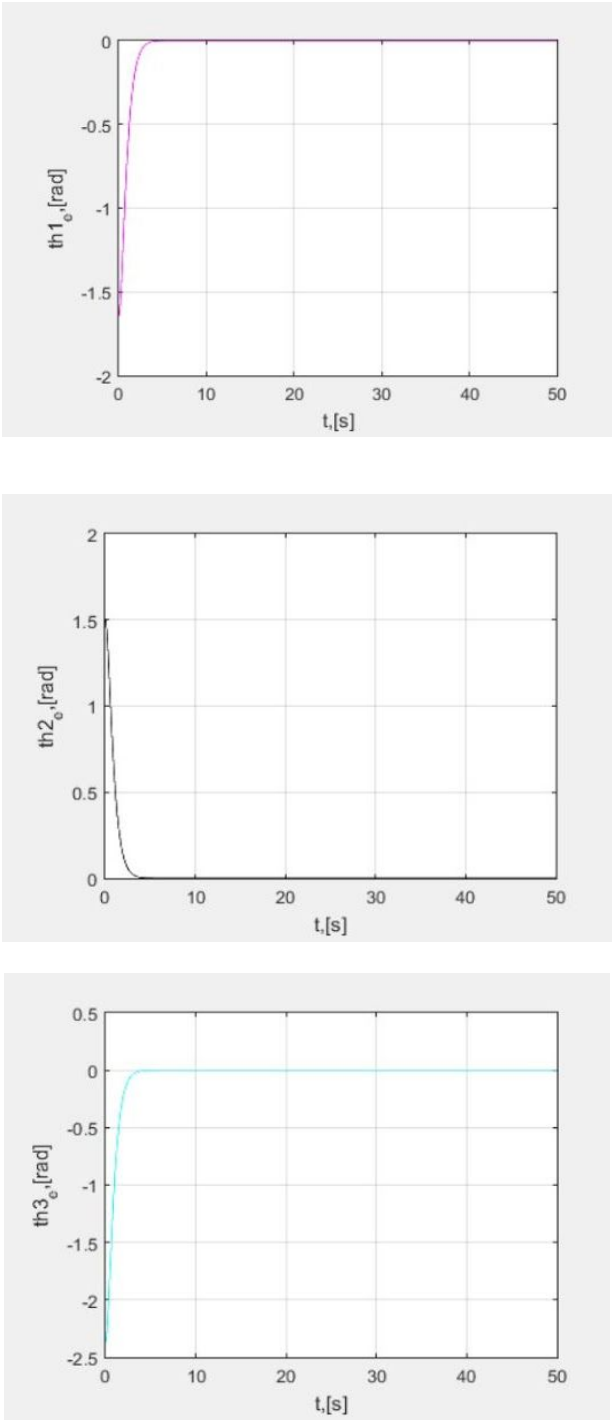
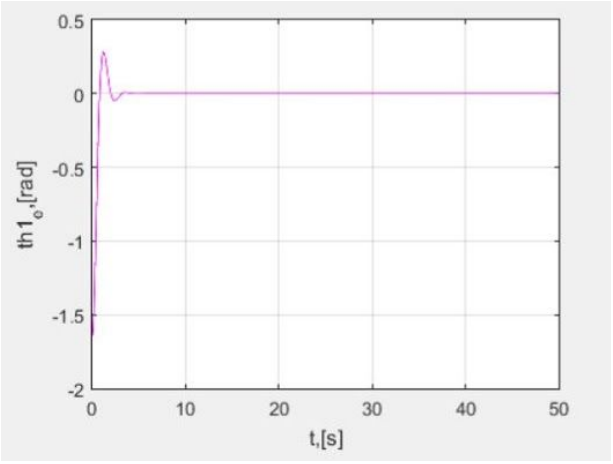
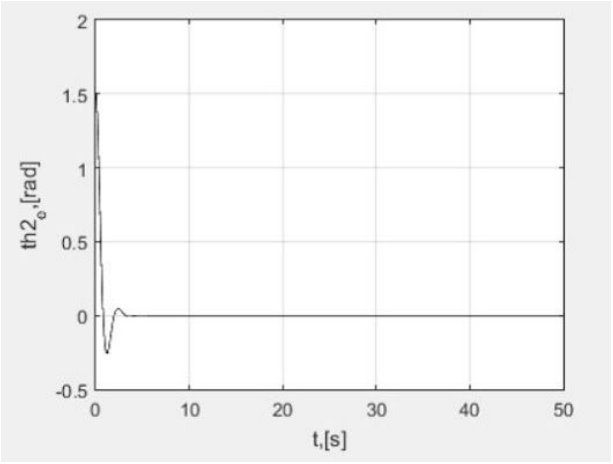


Fig 6.10 *Graphs showing errors in position(x_f,y_f) i.e, th_{ie} of each link with time for Critically Damped system*

Underdamped system($\xi = 0.5$)





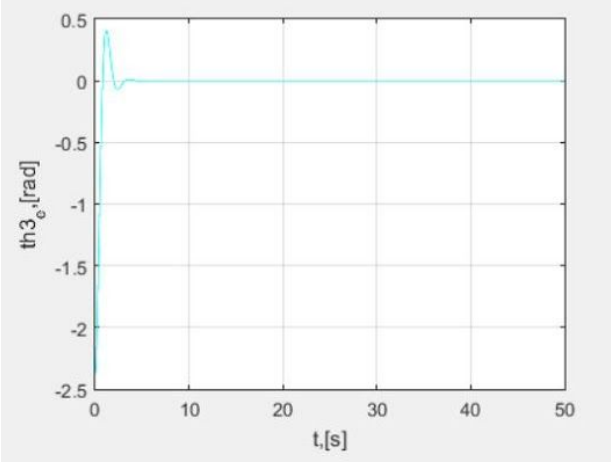


Fig 6.11 *Graphs showing errors in position(x_t,y_t) i.e, th_{ie} of each link with time for Underdamped system*

Chapter 7. Discussion of Results and Future Scope

7.1 Results

(i) Manipulator Position Tracking Results

Table 7.1 Results of Tracking (0, 40)

Desired - X (cm)	Desired - Y (cm)	actual - X (cm)	actual - Y (cm)	Distance error (cm)
0	40	1.08	38.60	1.768
0	40	-1.98	37.58	3.127
0	40	-1.55	38.24	2.345

Average Error *= 2.413cm*

Standard Deviation *= 0.682cm*

(ii) Mobile Base position Tracking Results

Table 7.2 Results of tracking (150, 150)

Desired - X (cm)	Desired - Y (cm)	actual - X (cm)	actual - Y (cm)	Distance error (cm)
150	150	148.2	151.09	2.104
150	150	147.63	149.72	2.386
150	150	148.54	145.22	4.998
150	150	153.7	152.12	4.264
150	150	144.8	145.47	6.896

Average Error *= 4.130 cm*

Standard Deviation *= 1.973 cm*

Table 7.3 Results of tracking (80, 80)

Desired - X (cm)	Desired - Y (cm)	actual - X (cm)	actual - Y (cm)	Distance error (cm)
80	80	79.09	79.41	1.0857
80	80	82.94	78.03	3.539
80	80	80.50	75.70	4.329

Average Error *= 2.984 cm*

Standard Deviation *= 1.692 cm*

Table 7.4 Results of tracking different points

Desired - X (cm)	Desired - Y (cm)	actual - X (cm)	actual - Y (cm)	Distance error (cm)
100	80	101.28	78.65	1.860
150	150	148.2	151.09	2.104
80	80	79.09	79.41	1.085

Average Error *= 1.683 cm*

Standard Deviation *= 0.533 cm*

(iii)Mobile Robot position Tracking Results

Table 7.5 Results of tracking (80, 140)

Desired - X (cm)	Desired - Y (cm)	actual - X (cm)	actual - Y (cm)	Distance error (cm)
80	140	70.2	144.6	10.826
80	140	72.4	146.5	10.000
80	140	74.5	136.2	6.685

Average Error *= 9.170 cm*

Standard Deviation *= 2.192 cm*

Above results shows average error and standard deviation of values obtained with respect to desired values. A maximum error of 3cm is obtained for manipulator where as, the value about 5cm for mobile base alone. When functioning together as a Mobile manipulator, it has a maximum error of 10.8cm. These errors can be even reduced to smaller values by taking care of manufacturing errors in links and correcting orientation of the mobile base which is slightly changing due to slip.

7.2 Future Scope

Mobile Base Trajectory Selection for Obstacle Avoidance

Obstacle avoidance can be employed to base tracking algorithm in order to make the base choose it own path while passing through objects or obstacles in the environment that it is working.

Manipulator Dual Loop Control for Accurate Tracking

As we have seen both Task-space and joint-space control have some complexities which imparts error to tracking. This can be reduced by developing dual loop algorithms where in both techniques are used simultaneously for accurate tracking.

IMU Sensor for Position Tracking along with any other Sensor to Operate in Outdoor Conditions

In this project, Xbox Kinect V1 served as the position sensor which is mounted to the roof of the lab and it restricts the workspace to about 3.3 x 2.45 m^2 area. In long run, when it is required to work in warehouses, shop floors and various outdoor conditions where this technique is difficult to employ, we can use IMU as position sensor which can be mounted on mobile robot and can be operated along with any other sensors like iGPS or RFID sensors for accurate results.

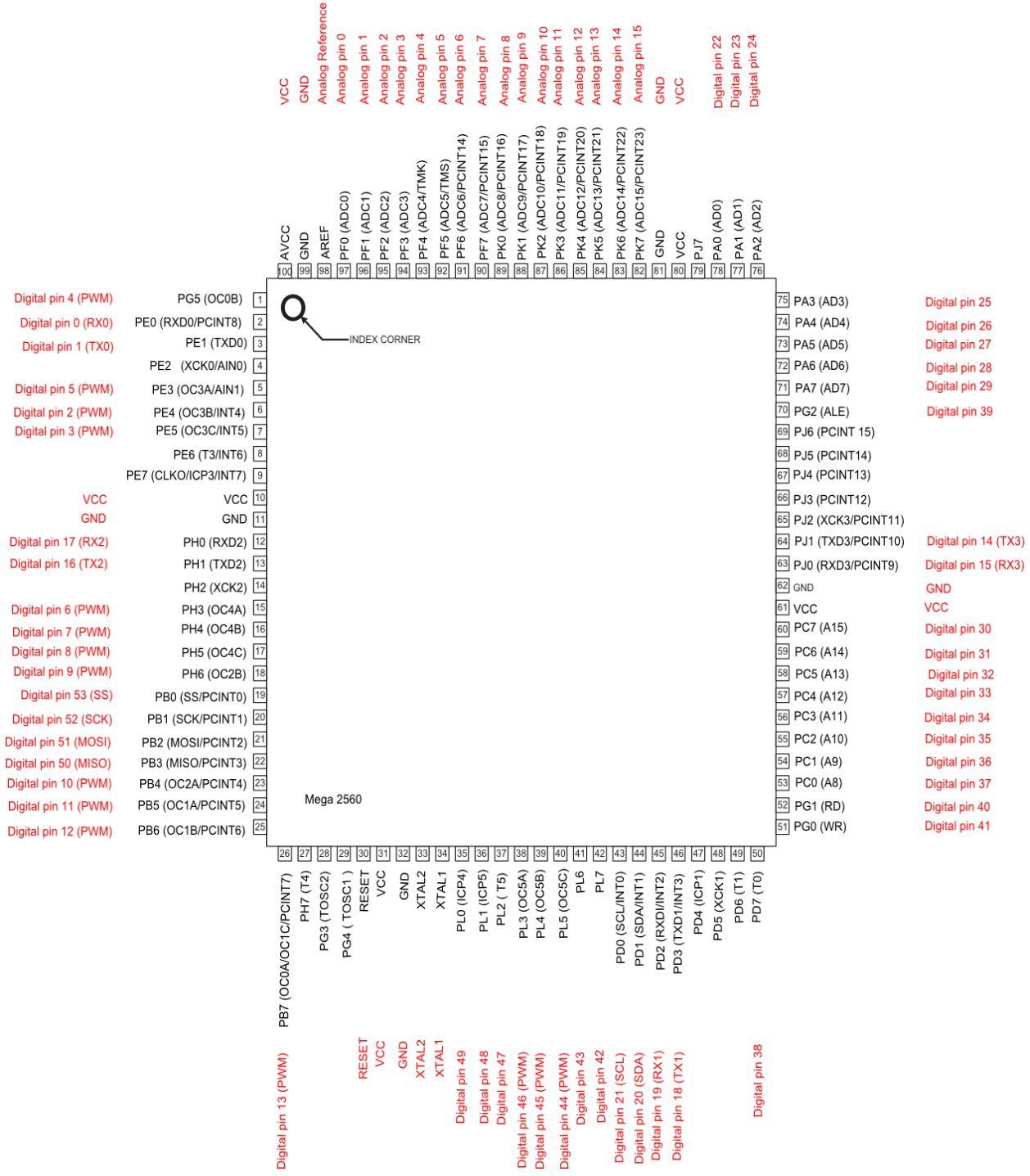
Algorithms for Selection of Optimal Orientation and Obstacle Avoidance

Algorithms can also be developed in order to enable the robot choose an optimal orientation of the manipulator in order to obtain an reduces the forces at joints and also to avoid any obstacles.

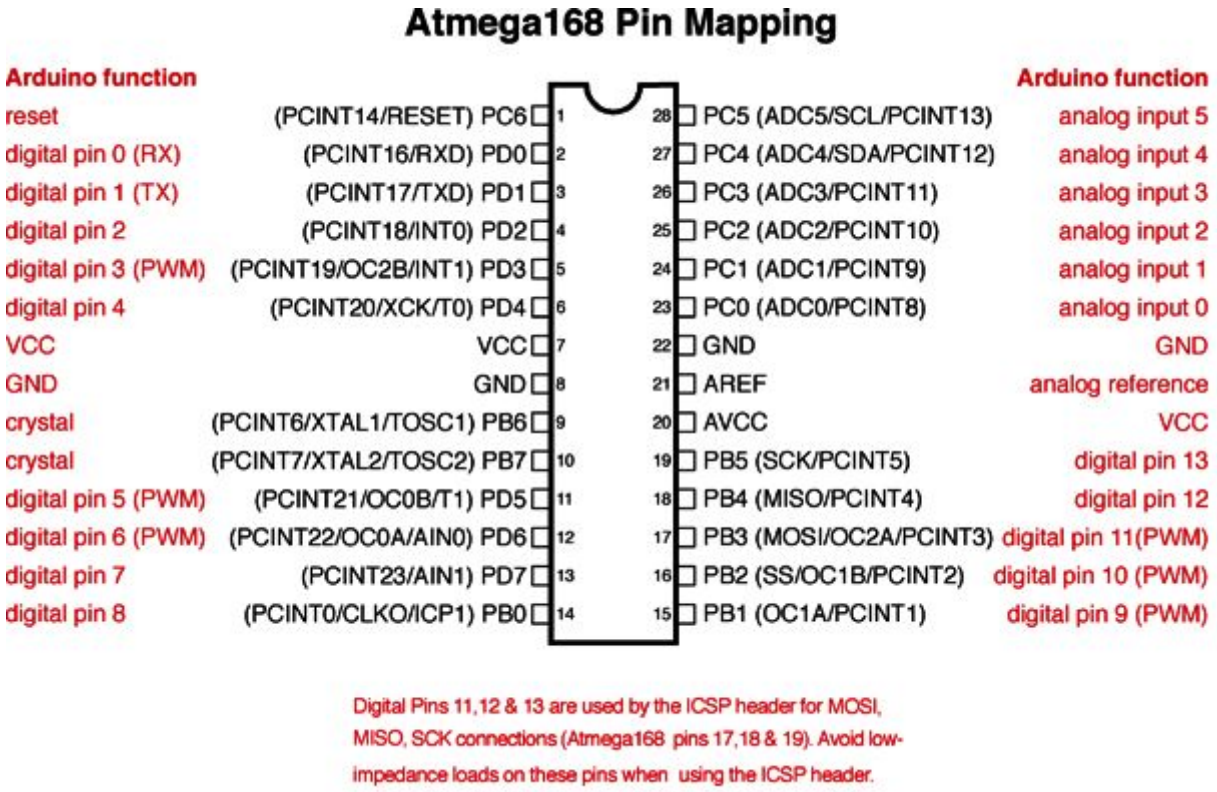
Appendix

A. PCB Design Schematics

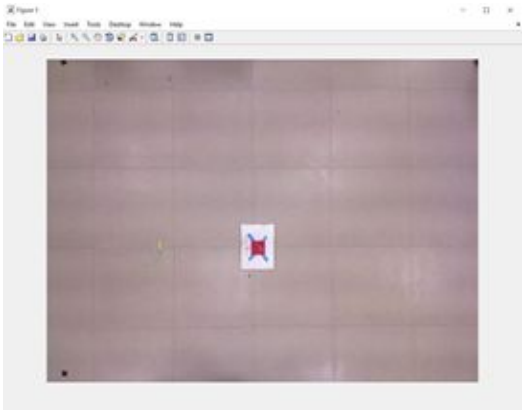
A.1 Arduino Mega 2560 board Schematics



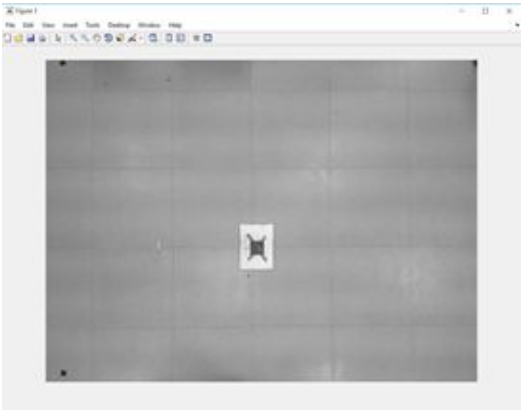
A.2 Arduino UNO R3 board Schematic



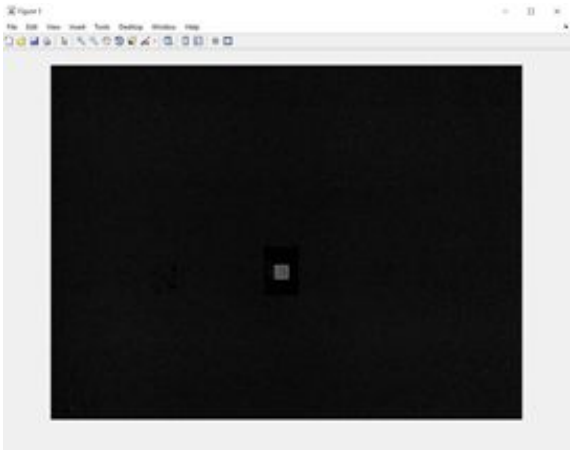
B. Image processing Samples



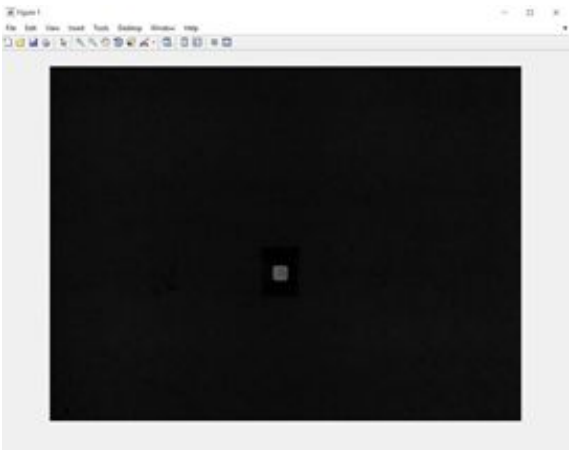
1



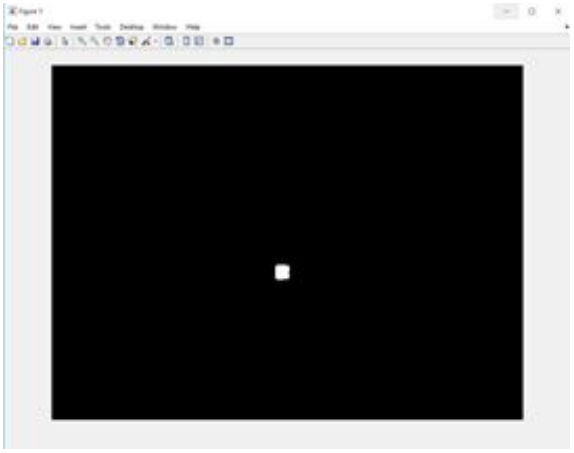
2



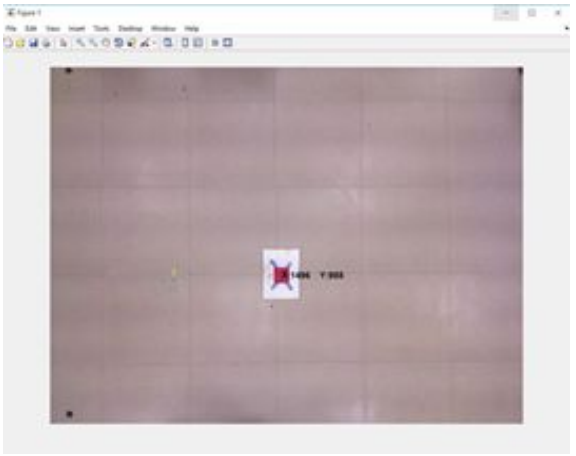
3



4



5



6

1-6 represents sequence of steps as mentioned in section 5.4.2

C. Lagrangian Dynamic modelling code

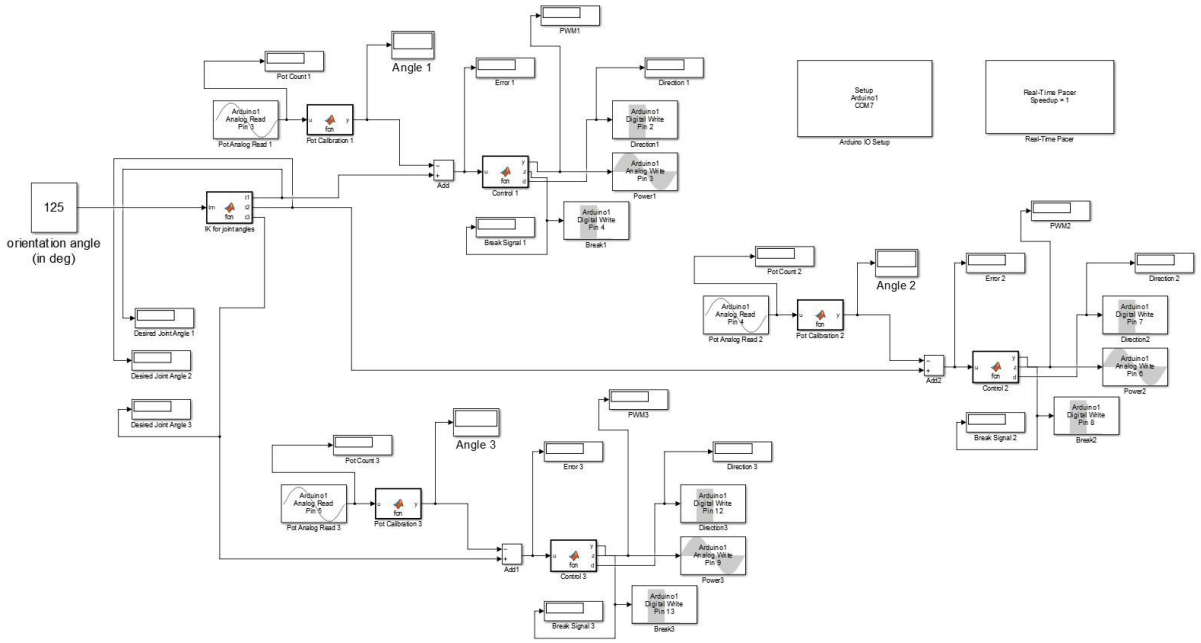
```
clear all; clc; close all;

%% SYMBOLS DECLARATION
syms m1 m2 m3;
syms L1 L2 L3;
syms th1(t) th2(t) th3(t);
syms th1_dot(t) th2_dot(t) th3_dot(t);
syms th1_ddot(t) th2_ddot(t) th3_ddot(t);
syms x1 x2 y1 y2 x3 y3;
syms p1 p2 p3;
syms v1(t) v2(t) v3(t);
syms a1 a2 a3;
syms KE1 KE2 KE3;
syms t;

%% POSITIONS
x1 = 0.5*L1*cos(th1) ;
% X-coordinate of the CoM of the first link
y1 = 0.5*L1*sin(th1) ;
% Y-coordinate of the CoM of the first link

x2 = L1*cos(th1) + 0.5*L2*cos(th1 + th2);
% X-coordinate of the CoM of the second link
y2 = L1*sin(th1) + 0.5*L2*sin(th1 + th2);
% Y-coordinate of the CoM of the second link

x3 = L1*cos(th1) + L2*cos(th1 + th2) + 0.5*L3*cos(th1 + th2 + th3);
% X-coordinate of the CoM of the third link
y3 = L1*sin(th1) + L2*sin(th1 + th2) + 0.5*L3*sin(th1 + th2 + th3);
% Y-coordinate of the CoM of the third link

p1 = [ x1 ; y1 ];
% Position matrix of Link 1
p2 = [ x2 ; y2 ];
% Position matrix of Link 2
```

```
p3 = [ x3 ; y3 ];
% Position matrix of Link 3

%% VELOCITIES

v1 = diff( p1, t);
v1 = subs(v1, diff(th1(t), t), th1_dot);
v2 = diff( p2, t);
v2 = subs(v2, diff(th1(t), t), th1_dot);
v2 = subs(v2, diff(th2(t), t), th2_dot);
v3 = diff( p3, t);
% Velocity = d/dt(Position) ; column matrix contains x and y components
of Velocity
v3 = subs(v3, diff(th1(t), t), th1_dot);
% Angular Velocity = d/dt(th1) = th1_dot
v3 = subs(v3, diff(th2(t), t), th2_dot);
v3 = subs(v3, diff(th3(t), t), th3_dot);

%% KINETIC ENERGIES & LAGRANGIAN

KE1 = simplify(sum(0.5*m1*(v1).*(v1)));
% Kinetic Energy = 0.5 * mass * velocity^2
KE2 = simplify(sum(0.5*m2*(v2).*(v2)));
KE3 = simplify(sum(0.5*m3*(v3).*(v3)));

KE = KE1 + KE2 + KE3;
% Total Kinetic Energy
PE = 0;
% Total Potential Energy (Planar Bot)
L = KE - PE;
% Lagrangian

%% LANGRANGIAN EQUATION TO DETERMINE TORQUE 1

dLdth1 = functionalDerivative(L, th1);
dLdth1dot = simplify(functionalDerivative(L, th1_dot));
DLDt1 = (diff(dLdth1dot, t));
```

```
DLDt1 = subs(DLDt1, diff(th1_dot, t), th1_ddot);
DLDt1 = subs(DLDt1, diff(th2_dot, t), th2_ddot);
DLDt1 = subs(DLDt1, diff(th3_dot, t), th3_ddot);
DLDt1 = subs(DLDt1, diff(th1(t), t), th1_dot);
DLDt1 = subs(DLDt1, diff(th2(t), t), th2_dot);
DLDt1 = subs(DLDt1, diff(th3(t), t), th3_dot);
DLDt1 = simplify(DLDt1);
torque1 = DLDt1 - dLdth1;
% Joint 1 Torque

%% LANGRANGIAN EQUATION TO DETERMINE TORQUE 2

dLdth2 = functionalDerivative(L, th2);
dLdth2dot = simplify(functionalDerivative(L, th2_dot));
DLDt2 = (diff(dLdth2dot, t));
DLDt2 = subs(DLDt2, diff(th1_dot, t), th1_ddot);
DLDt2 = subs(DLDt2, diff(th2_dot, t), th2_ddot);
DLDt2 = subs(DLDt2, diff(th3_dot, t), th3_ddot);
DLDt2 = subs(DLDt2, diff(th1(t), t), th1_dot);
DLDt2 = subs(DLDt2, diff(th2(t), t), th2_dot);
DLDt2 = subs(DLDt2, diff(th3(t), t), th3_dot);
DLDt2 = simplify(DLDt2);
torque2 = simplify(DLDt2 - dLdth2);
% Joint 2 Torque

%% LANGRANGIAN EQUATION TO DETERMINE TORQUE 3

dLdth3 = functionalDerivative(L, th3);
dLdth3dot = simplify(functionalDerivative(L, th2_dot));
DLDt3 = (diff(dLdth3dot, t));
DLDt3 = subs(DLDt3, diff(th1_dot, t), th1_ddot);
DLDt3 = subs(DLDt3, diff(th2_dot, t), th2_ddot);
DLDt3 = subs(DLDt3, diff(th3_dot, t), th3_ddot);
DLDt3 = subs(DLDt3, diff(th1(t), t), th1_dot);
DLDt3 = subs(DLDt3, diff(th2(t), t), th2_dot);
DLDt3 = subs(DLDt3, diff(th3(t), t), th3_dot);
DLDt3 = simplify(DLDt3);
```

```
torque3 = simplify(DLDt3 - dLdth3);
% Joint 3 Torque

%%
disp(torque1);
disp(torque2);
disp(torque3);
```

D. Manipulator Simulink Code



References

[1] Robot Institute of America, 1979. Robotics: A Brief History - Stanford. https://cs.stanford.edu/people/eroberts/courses/soco/projects/1998-99/robotics/history.html

[2] Institute for Systems and Robotics, Lisboa. Isr.tecnico.ulisboa.pt

[3] frc.ri.cmu.edu/projects/meteorobot2000/

[4] http://au.gehl.com/images/default-source/backhoe-loaders/bl-818-center-mount-1_thumb.png?sfvrsn=f2ac286a_4

[5] https://media.gettyimages.com/illustrations/diagram-of-backhoe-loader-depicting-both-the-digging-and-loading-of-illustration-id71558864

[6] http://meconstructionnews.com/wp-content/uploads/2017/11/Backhoes-Case-570T.jpg

[7] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.227.5044&rep=rep1&type=pdf

[8] http://www.researchgate.net/profile/V_Spinu/publication/221786657/figure/fig5/AS:282593363808259@1444386916273/DOFs-in-a-Mecanum-wheel-Song-Byun-2004-The-rollers-are-neither-actuated-nor-sensed.png

[9] https://me-mechanicalengineering.com/grashofs-law/

[10] Intuitive dynamic modeling and flatness-based nonlinear control of a mobile robot. http://journals.sagepub.com/doi/abs/10.1177/0037549717741192

[11] https://upload.wikimedia.org/wikipedia/commons/thumb/c/c5/Planar-three-link-manipulator.svg/2000px-Planar-three-link-manipulator.svg.png

[10] https://www.arduino.cc

[11]
https://robokits.co.in/motors/encoder-dc-servo/high-torque-quad-encoder-geared-dc-motor-12v-200rpm
[12]
https://www.rhydolabz.com/robotics-dc-geared-motors-c-155_156/dc-motor-with-gearbox-10rpm-p-814.html?gclid=EAIaIQobChMI_oGo3aC_3gIVzxwrCh18mQiqEAYYBCABEgJ18vD_BwE&zenid=l49remnp5s7lgirlbdh6mgncn3
[13]
https://www.electronics-tutorials.ws/resistor/potentiometer.html
[14]
https://www.uv.es/imaging3/PDFs/2016_OEng_56_41305.pdf
[15]
https://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf
[16]
http://robokits.download/documentation/DC_Motor_Driver_24V_20A.pdf
[17]
https://ewh.ieee.org/r1/ct/sps/PDF/MATLAB/chapter8.pdf
[18] Introduction to robotics: mechanics and control, Third edition by John J. Craig.