

MECHANICAL AND CONTROLLER DESIGN OF A MODULAR MECHATRONIC DEVICE – MECHACELL

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

By
Stefan Ristevski
August, 2015

Mechanical and Controller Design of a Modular Mechatronic Device –
MechaCell

By Stefan Ristevski

August, 2015

We certify that we have read this thesis and that in our opinion it is fully adequate,
in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Melih Çakmakcı(Advisor)

Assist. Prof. Dr. Onur Özcan

Assoc. Prof. Dr. Uluç Saranlı

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

MECHANICAL AND CONTROLLER DESIGN OF A MODULAR MECHATRONIC DEVICE – MECHACELL

Stefan Ristevski

M.S. in Mechanical Engineering

Advisor: Assist. Prof. Dr. Melih Çakmakçı

August, 2015

Since ancient times people have been building tools to aid them in their life. Robots evolved from being purely mechanical to mechatronic, from immobile to mobile and became smaller in scale. As the technology in building robots matured researchers, began working to build robotic systems that cooperate similar to the ones in nature. Ability of ants to accomplish tasks beyond the capability of a single ant intrigued scientists in robotics society to mimic that feature of ants and develop simple modules that alone cannot accomplish much, but together can complete complex assignments. Our motivation is to develop a miniaturizable mechatronic module–MechaCell.

Mechanical design focuses on a novel locomotion system having a mechanism that converts vibrations into translational motion. Two independent controllers, one for steering and one for translational speed control are designed such that MechaCell can follow a complex path and group of MechaCells can guide an object to follow a complex path.

Simulation results from the model of the MechaCell developed in SimMechanics are presented. Experimental setup comprising of a Bluetooth enabled PC, a platform, an overhead camera and four MechaCells is set up and simulation results are experimentally verified.

Possible application of coordinated object manipulation is in manufacturing systems that have limited fixture capabilities and desired precision in sub–centimeter levels.

Keywords: Modular, Mechatronic, Coordinated object manipulation, Locomotion system, Vibration utilization.

ÖZET

MODÜLER MEKATRONİK ALETİN MEKANİK VE KONTROLCÜ DIZAYNı–MECHACELL

Stefan Ristevski

Makine Mühendisliği, Yüksek Lisans

Tez Danışmanı: Assist. Prof. Dr. Melih Çakmakcı

Ağustos, 2015

Eski çağlardan beridir insanlar kendi hayatlarını kolaylaştırmak için çeşitli araçlar üretmekteledir. Robot teknolojisi de zamanla gelişmektedir. Örneğin günümüz robotları mekatronik, hareketli ve daha küçüktürler. Robotik alanında bu gelişmeler araştırmacıları doğa ile robotik sistemleri birleştirmeye teşvik etmiştir. Karıncalar ele alındığında grup olarak yapabildikleri ile bir karıncaya göre üstün olurlar. Bu da araştırmacıların dikkatini doğayı taklit edebilen çok modüllü, komplike sistemlere çekmiştir. Bizim motivasyonumuz ise küçülebilen mekatronik modül–MechaCell– geliştirmektir.

MechaCell’in mekanik dizaynı ve kontrol algoritması detaylıca ilerideki bölümlerde anlatılmıştır. Buradaki mekanik dizaynda önemli olan unsur geliştirilen yeni hareketli bir sistemdeki mekanizmanın ortaya çıkardığı titreşimi doğrusal harekete (translational motion) çevirmesidir. Ayrıca MechaCell’in hız kontrolü, yön kontrolü ve sürtünme kompensasyon kontrolü yapılarak modülün komplike bir rotayı takip edebilmesi sağlanmıştır. Buna ek olarak dört modül grup olarak bir objenin karışık bir yön izlemesini sağlayabilir.

MechaCell’in SimMechanics programı ile elde edilen simülasyon sonuçları detaylıca belirtilmiştir. Dört modül, bilgisayar ve kamera ile yapılan deneyler simülasyon sonuçlarını doğrulamaktadır.

Bu tezde sunulan sistemin potansiyel kullanım alanları arasında sabitleme problemi olan ve hassas submillimetrik pozisyonlama ihtiyacı olan üretim sistemleri vardır.

Anahtar sözcükler: Modüler, Mekatronik, Koordineli hareket ettirme, Locomosyon sistemi, Titreşimden yararlanma.

Acknowledgement

Firstly, I would like to express my sincere gratitude to my advisor Assist. Prof. Melih akmakcı for the continuous support of my graduate study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my graduate study.

Besides my advisor, I would like to thank the rest of my thesis committee: Assist. Prof. Onur  zcan and Assoc. Prof. Ulu Saranlı, for their insightful comments and encouragement, but also for the hard question which incited me to widen my research from various perspectives.

I thank my fellow office mates, Elif Altıntepe, Baak Bayraktar, Alper Tiftikci, Serdar Taze and the rest of the graduate students in the department of Mechanical Engineering for all the fun we have had in the last three years.

I would like to thank my family: my parents and to my brother for supporting me spiritually throughout writing this thesis and in my life in general.

Lastly, I would like to thank the European Commission through Seventh Framework Program for Research and Technological Development which provided financial support for this project (MC-IRG No: 268420).

Contents

1	Introduction	1
1.1	Introduction	1
1.2	Overview	6
2	Mechanical Design & Electronics	8
2.1	Mechanical Design	8
2.1.1	Locomotion System	9
2.2	Actuators	12
2.3	Electronics	14
2.3.1	Sensors	19
2.3.2	Software	20
2.4	Main Body	21
2.5	Manufacturing	22
3	Modeling & Control	26

3.1	Mathematical Model	26
3.2	Model of the MechaCell in SimMechanics	29
3.3	Force Models in SimMechanics	30
3.3.1	Contact Forces	32
3.3.2	Friction Forces	33
3.4	Control Algorithm	34
3.4.1	Translational Control (t-controller)	35
3.4.2	Steering Control (s-controller)	38
3.5	Single MechaCell Simulations in SimMechanics	39
3.6	Cooperative Workpiece Manipulation Model in SimMechanics	44
3.6.1	Object Caging	44
3.7	Simulations of Cooperative Workpiece Manipulation in SimMechanics Environment	48
4	Experimental Setup & Experiments	54
4.1	Experimental Setup	55
4.1.1	Position Tracking	56
4.1.2	Camera Calibration	59
4.2	Experiments	60
4.2.1	Mathematical Model Verification	62

4.2.2	Linear controller (PID) tuning	63
4.2.3	Single MechaCell Path Following–Experiments	66
4.2.4	Cooperative Workpiece Manipulation–Experiments	68
5	Conclusion & Future Work	72
5.1	Conclusion	72
5.2	Future Work	73
A	Electronic Components	80
A.1	Arduino ProMini Board	80
A.2	Motor Driver	81
A.3	Bluetooth Module - RN42	81
A.4	Vibration Motor	83
A.5	Servo Motor	85
B	Matlab-Code	86
B.1	Identification Algorithm	86
B.2	Servo Motor Angle Calculation	89
B.3	IDE-Code Uploaded to the Arduino	90

List of Figures

1.1	Archytas of Tarentum mechanical bird–The Pigeon	2
1.2	Leonardo da Vinci’s knight with inner workings	3
1.3	Example of legged robots: A) ASIMO bipedal robot; B) Big Dog 4 legged robot; C) RHex 6 legged robot	4
1.4	Examples of swarm robots: A) Alice robot; B) Swarm of Alice robots; C) Kilobot robot; D) Swarm of Kilobot robots	5
1.5	Simple sketch of the motivation of the study	6
2.1	Rotating unbalance presented in 3D	10
2.2	Graphical representation of the conversion of vibration motion to translational motion as the unbalanced mass, m , passes through four stages	11
2.3	Acceleration, velocity and displacement plots supporting the work- ing principle of the locomotion system based on Figure (2.2) . . .	12
2.4	Different types of cellphone vibration motors available on the mar- ket, vibration motor used in the MechaCell is highlighted	13

2.5	Actuator candidates for angular position fix of the steering cylinder, stepper and servo motors	14
2.6	Actuators of the MechaCell, servo motor on the left and vibration motor on the right	15
2.7	Different Arduino boards available on the market	16
2.8	L293C motor driver with pin labels	17
2.9	RN42 Bluetooth module from Roving Networks	19
2.10	Sensors that will be implemented in the future MechaCell	20
2.11	CAD drawing of parts of Section I of the Main Body design	23
2.12	CAD drawing of parts of Section II of the Main Body design	24
2.13	CAD drawings of mechanical components of the MechaCell and assembly	24
2.14	Manufactured parts of the MechaCell	25
2.15	CAD and Actual MechaCell	25
3.1	Projected view on XZ -plane of Figure (2.1) and corresponding FBD	28
3.2	MechaCell represented as a point mass in the XY -plane of the inertial coordinate system	29
3.3	CAD drawings of the parts used in SimMechanics to model the MechaCell; <i>A</i>) Exploded view of all three parts; <i>B</i>) Rotating unbalance is fixed in the steering cylinder; <i>C</i>) Steering cylinder is fixed inside the main body;	30

3.4	Snapshot of the SimMechanics environment where a single Mecha-Cell is resting on a floor	31
3.5	Two bodies in contact	31
3.6	Graphical representation of circle–line and circle–circle force laws	33
3.7	Graphical representation of friction force law	33
3.8	Coulomb’s friction model	34
3.9	Block diagram of the control structured implemented in the Mecha-Cell	35
3.10	Block diagram of the t–controller consisting of a PID controller and a friction compensation algorithm	36
3.11	Top view of the MechaCell, its position, desired position are sketched	37
3.12	Prescribed path for single MechaCell simulations, B–shaped path	40
3.13	Simulation results of a MechaCell following a prescribed complex path	42
3.14	Snapshot of the SimMechanics environment: Single MechaCell following a path	43
3.15	Graphical representation of the deviation from path numerical error parameter, ϵ	43
3.16	Form object closure caging example, object is not caged in part a, object is caged in part b	45
3.17	A disk shaped object is caged with form closure caging strategy by 3,4 and 5 mobile robots	46

3.18	Form closure caging of an object by 3, 4 and 5 mobile robots and definitions of parameters used in Equation (3.20)	47
3.19	Different planar shapes: Regular polygon inscribed in a circle, Irregular polygon inscribed in the smallest circle, Non polygonal shape inscribed in the smallest circle	48
3.20	Deformed rotated D-shape path that is used in the workpiece manipulation simulations	49
3.21	Snapshot of SimMechanics environment where four MechaCells and a workpiece rest on a floor and MechaCell are distributed around the workpiece to achieve form closure caging condition . .	50
3.22	Top view of Figure (3.21) with visible coordinate systems	51
3.23	Workpiece path guiding task simulation results	53
4.1	Experimental Setup	56
4.2	Examples of bounding boxes for different shapes	57
4.3	Color coding of MechaCells, with red background color and green foreground color	58
4.4	Snapshot of the color tracking algorithm	60
4.5	Graphical interpretation of the work flow between components of the experimental setup during experiments	61
4.6	Mathematical model with point mass assumption validation . . .	63
4.7	Feedback control structure of the MechaCell with only PID controller	64
4.8	Step responses for different PID-controller configurations	65

4.9	Best step response from Figure (4.8)	66
4.10	Best step response from Figure (4.8) and response improvement with after integration of friction compensation algorithm	67
4.11	Complex path used in the path following experiments identical to the path in Section 3.5	68
4.12	Results of the path following tasks of a single MechaCell	69
4.13	Complex path used in the path guidance experiments identical to the path in Section 3.7	71
4.14	Workpiece path guidance experiment's results	71
5.1	Future design of the MechaCell–mini MechaCell	74
A.1	L293C motor driver pin-out	81
A.2	RN42 Bluetooth module pin-out	83
A.3	Rotational speed with current variations of the shaft-less vibration motor	84
A.4	TowerPro SG90 servo motor	85

List of Tables

2.1	L293C motor driver pin connections	17
2.2	L293C motor driver input pin status combinations that determine the direction of rotation of the vibration motor	18
2.3	Bluetooth module to arduino board pin connections	18
3.1	ANGLE NOMENCLATURE FOR FIGURE (3.11)	38
3.2	Numerical values of key parameters used in the SimMechanics sim- ulation	41
3.3	Numerical values of key parameters used to perform the workpiece manipulation simulations	52
3.4	Numerical error parameters for four MechaCells and the workpiece handled by the MechaCells	53
4.1	MechaCell numbering according to the color codes	59
4.2	Parameters used in the mathematical model with numerical values and definitions	64

4.3	Numerical values of the error parameters from the experiment and the simulations	68
4.4	Numerical values of error parameters for the workpiece guiding tasks	70
A.1	Arduino ProMini board's spec summary	80
A.2	Basic specifications of the shaftless vibrational motor	84
A.3	TowerPro SG90 servo motor basic specifications	85

Chapter 1

Introduction

1.1 Introduction

Robot (Origin 1920) — *a machine that has the capability to complete complex series of tasks automatically, one that can be programmed by a computer.* [1]

Even though, the word robot appeared in the nineteenth century, people have been building robots from much earlier. Many inventors and engineers in ancient China, ancient Greece and Ptolemaic Egypt have made attempts in constructing self-operating machines. Some of these machines resembled humans and/or animals. [2–4] Around fourth century B.C. a Greek mathematician Archytas of Tarentum, proposed a theoretical mechanical steam-operated bird, called The Pigeon, shown in Figure (1.1). [5] During the renaissance period, the Italian inventor Leonardo da Vinci had designed a human-like robot. Leonardo's notebooks contained detailed sketches of the robot which looked like a knight, and was able to wave, move its head and jaw and sit up. This robot was also known as Leonardo's knight, shown in Figure (1.2). [6] Interest in building robots has been constantly growing since.

Mass production, which started with the industrial revolution had a positive effect

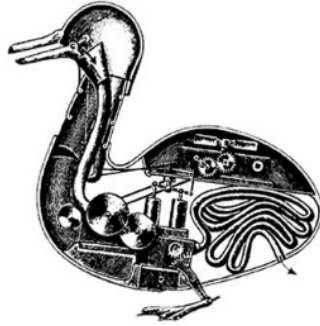


Figure 1.1: Archytas of Tarentum mechanical bird–The Pigeon

on the field of robotics (field of design, construction, operation and application of robots). [7] To speed up, decrease number of faults and replace humans in the process of mass production; new, huge and expensive robots were constructed – Industrial Robots. Speed, automation, programming and compactness of industrial robots today, are as a result from the tremendous advances made in the fields of nano–technology and computer science. Everything was triggered by Feynman’s talk *“There’s Plenty of Room at The Bottom”* in 1959. This talk is also considered as a motivation, and a start of the field of nano–technology. And an academic publication by Alan Turing published in 1936 thought of as the start of digital age, and predecessors of modern computers. [8,9] Industrial robots are very versatile, their size varies from centimeters to few meters, can be operated manually or be fully automatic. They are able to perform different tasks, such as: pick and place, material handling, painting, welding. However, having fixed physical location is a disadvantage and focus of researchers in robotics society has shifted toward mobile robots.

Mobile robots are defined as robots that are capable of changing their physical location, either in a medium (water or air) or on ground. They are classified as: Autonomous Underwater Vehicles (AUVs), Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs), based on the medium. Further classification of UGVs can be done according to the type of featured locomotion system: legged robots (human–like or animal–like), wheeled robots (locomotion system using wheels). Extensive research has been done on designing robots with a locomotion system featuring extremities. Robots resembling humans (2

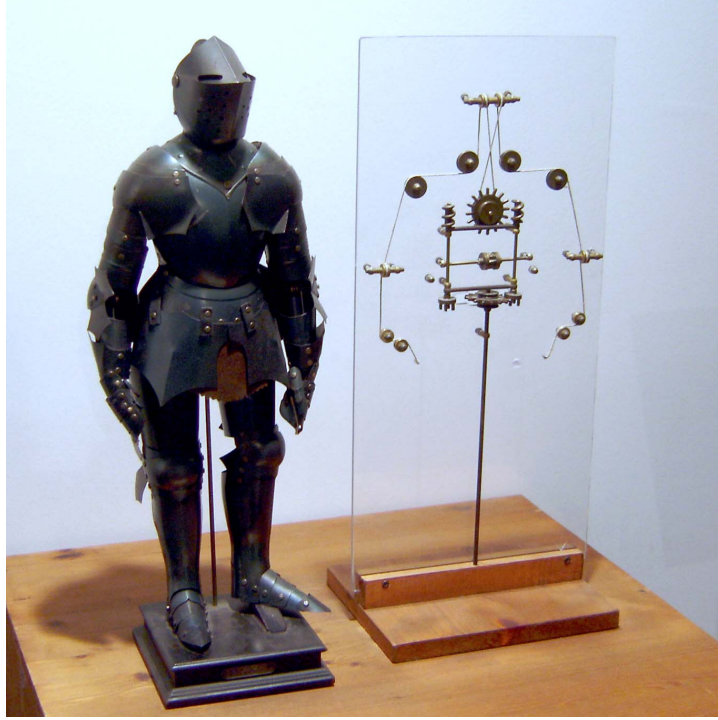


Figure 1.2: Leonardo da Vinci's knight with inner workings

extremities) or animals (4 or 6 extremities). ASIMO (Advanced Step in Innovative Mobility), is a humanoid robot developed by Honda. ASIMO with height of 130 [cm] has the ability to walk and run in an unknown environment. [10] Robots with higher number of extremities have been designed by Boston Dynamics, 4 legged Big Dog (91 [cm] in height and 76 [cm] in length) and 6 legged Rhex (39 [cm] in height and 54 [cm] in length). Both robots are able to walk and run in an unknown environment. [11, 12] The literature abounds with works on gait stability, gait speed, path planning etc, on legged robots. [13–17] ASIMO, Big Dog and Rhex are shown in Figure (1.3, A), Figure (1.3, B) and Figure (1.3, C) respectively.

At this point it is clear that mobile robots are constructed to resemble humans or animals. Mimicking biological beings in robot design, in addition to the physical mechanisms for locomotion, includes social behavior as well. Interactions between robots in a large group are modeled based on an swarm behavior observed in ants

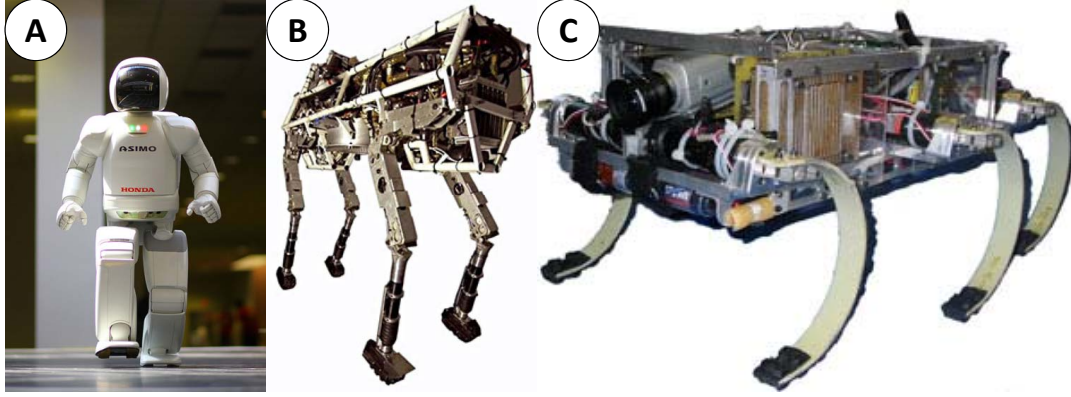


Figure 1.3: Example of legged robots:

A) ASIMO bipedal robot; B) Big Dog 4 legged robot; C) RHex 6 legged robot

and bees. [18–20] Swarm in robotics is defined as a large group of simple robots. Swarm Robots are able to complete more complex tasks when part of a swarm, than they are able to complete when they are alone. [21,22] Building large number of robots is not an easy task, hence robots that are part of a swarm, should be small in size and physically simple. With locomotion system based on wheels like *Alice*, or alternatively, based on vibrational motion like *Kilobot*, which converts vibrational motion into translational motion. [23, 24] Figure (1.4, A–B) shows *Alice* robot and a swarm of *Alice* robots respectively and in Figure (1.4, C–D) *Kilobot* robot and swarm of *Kilobot* robots are shown.

We focus on designing a simple, mechatronic device that can track complex paths, and a group of these devices can manipulate an object in plane to follow complex paths – MechaCell.

A review for optimal path planning of mobile robots is given in [25], path planning of mobile robots based on swarm intelligence is given in [26]. Object manipulation is an important task that robots should perform; research on different approaches of object manipulation in plane by a swarm of robots is studied by many researchers. [27–31]

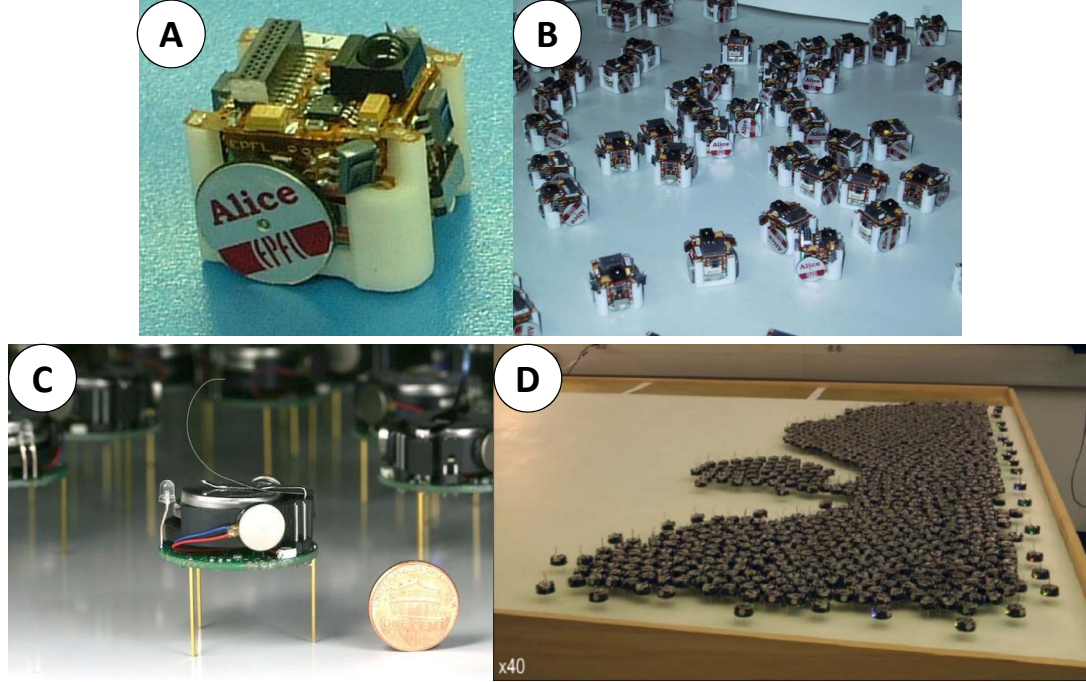


Figure 1.4: Examples of swarm robots:
A) Alice robot; B) Swarm of Alice robots; C) Kilobot robot; D) Swarm of Kilobot robots

After considering many different possibilities, a locomotion system based on conversion of vibrational motion to translational motion is used as reported by others [24,32–34]. However, different than the existing literature, our locomotion system is omni-directional (it can go to any target location by following the straight line connecting the target and current location), compact and completely closed system; further description is given in Chapter 2.

Machining processes, laser cutting, or other types of manufacturing techniques which require object (workpiece) attachment to a platform, can sometimes damage the workpiece due to insufficient or improper fixture. Or, they are limited to be able to fix workpieces with certain shape and/or hardness only. [35, 36]

Our motivation in this project is to develop an alternative fixture in manufacturing, using swarm of robots to *cage* the workpiece. Caging condition is achieved,

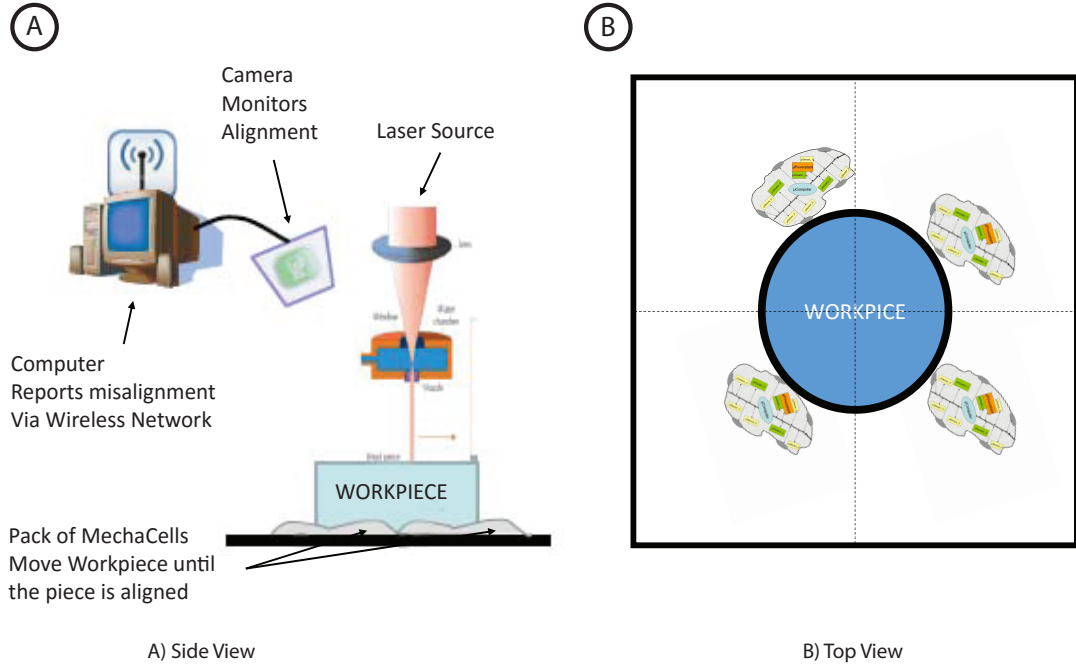


Figure 1.5: Simple sketch of the motivation of the study

when sufficient number of robots (depending on the workpiece's geometry), surround the workpiece and make a rigid formation around it. Robots have no physical attachments with the workpiece, and only apply forces aiding the motion. [37, 38] Theoretical manufacturing system with fixture-less approach, using swarm of robots to manipulate the workpiece on a platform, is shown in Figure (1.5). In this theoretical system, the workpiece is guided by the MechaCells, while being processed by the laser.

1.2 Overview

The rest of this thesis is structured as follows:

In Chapter 2, Mechanical design of the MechaCell is presented. Mechanical design is divided into body of the MechaCell and locomotion system of the MechaCell. Vibration forces produced by a rotating unbalance are converted into translation by the locomotion system. This chapter also includes electronic components of

the MechaCell.

Chapter 3 develops a mathematical model of the MechaCell and a control algorithm. Mathematical model is developed and simulated in Simulink/SimMechanics environment. Simulations present single MechaCell path following and four MechaCells object manipulation. Control algorithm comprises speed control and steering control.

Chapter 4 outlines the experimental setup in the laboratory, experiments and results. Experiments with one MechaCell following a path and four MechaCells manipulating a workpiece are performed. Comparison between simulations and experiments is presented.

Chapter 5 is the final chapter and concludes the work. Ideas for future work such as shrinking the size of the current MechaCell and few others are presented.

Chapter 2

Mechanical Design & Electronics

This chapter will present the mechanical design of the MechaCell and electronic components. Locomotion system of the MechaCell is in the focus of the mechanical design. Design of the locomotion system, its working principle and implementation are discussed in detail. Electronic components included in the MechaCell design are outlined and discussed.

2.1 Mechanical Design

Mechanical design comprises of two parts, part one is the main body of the MechaCell and part two is the locomotion system of the MechaCell. Size of the MechaCell is the leading design parameter in the mechanical design, such that The MechaCell would fit in space of $[5 \times 5 \times 5] [cm^3]$ as proposed in the EU project in 2010. Even though a cube with the same dimensions looks like the most logical choice, taking into account that this device is to be used in an object manipulation in plane using a pushing action only, main body of the MechaCell is designed as a cylinder with equal height and diameter, such that the contact with an object is always a theoretical point, unless the object has a shape of concave polygon.

Our MechaCell system has two main parts:

1. A mechanical locomotion system
2. The controller module with embedded sensors which commands the locomotion parts

In designing the locomotion system, a new approach has been taken, where vibration motion is converted into translational motion. Principle of conversion of vibrating motion to translating motion is not very commonly used in the design of locomotion systems of robots, however is found in the Kilobot and the bristle-bot. [24, 34] Locomotion system developed in this work differs from both system mentioned above, it is omni-directional and has only one vibration motor compared to the Kilobot, it is steerable and does not use bristles compared to the bristle-bot.

2.1.1 Locomotion System

Locomotion system should be compact, completely closed and fit in a restricted space of the main body which has a cylindrical shape. Locomotion system based on vibration forces is designed. Vibrations are produced by a rotating unbalanced mass. Figure (2.1) shows a simple 3D sketch describing the principle of the locomotion system. In this figure mass, m , is the unbalanced mass and is attached to a rod which than is connected to cylinder 2 at point A . In Figure (2.1) there are two coordinate systems, coordinate system one is XYZ —an inertial coordinate system, and coordinate system two is $X'Y'Z'$ —a non-inertial (steering) coordinate system. The steering coordinate system is attached to cylinder 2 – the steering cylinder. These two coordinate system are related to each other such that Z -axis and Z' -axis are coinciding. The steering cylinder is attached to cylinder 1 – the base-cylinder with one degree of freedom (DOF), rotation about Z' -axis as indicated by the arrow in Figure (2.1). System presented in Figure (2.1) always moves in either positive or negative X' -direction, depending on the direction

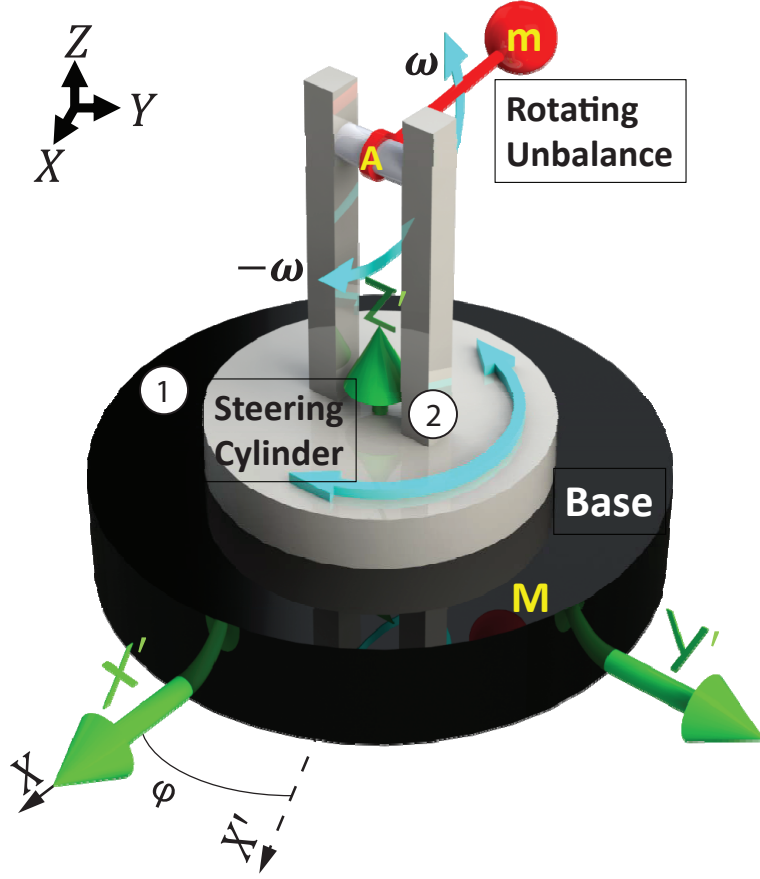


Figure 2.1: Rotating unbalance presented in 3D

of the angular velocity of the unbalance mass m , ω . Positive ω will locomote the system in negative X' -direction and negative ω in positive X' -direction. Direction of motion of the system in Figure (2.1) in the inertial coordinate system is defined by the angle, ϕ , constructed with the positive X -axis and the positive X' -axis. Using this configuration the system can move in any direction in the XY -plane, which results in an omni-directional drive.

Working principle of the locomotion system is based on the fact that the centrifugal force produced by the rotating unbalanced mass, m , affects the normal force between the base-cylinder and ground which affects the friction force. Let unbalanced mass, m , be rotating with a positive angular speed, $+\omega$, and let angle, θ , be defined as the angle between the positive X' -axis and the rod connecting the unbalanced mass, m , to point, A , of the steering cylinder. Let the system be

at rest and initial angular position, θ , and angular speed, ω , of the unbalanced mass, m , be 180° and ω respectively.

Translational motion of the system can be mapped into four stages. Stages are defined according to the angular position, θ , of the unbalanced mass, m . Each stage is represented by a quadrant of the non-inertial coordinate system, $X'Z'$. In Figure (2.2), mapping of the translational motion of the system according to the angular position, θ , of the unbalanced mass, m , is graphically shown.

As the unbalanced mass, m , passes through quadrants 3 and 4 of the non-inertial coordinate system, $X'Z'$, the system stays at rest. However, when angular position of the unbalanced mass, m , reaches quadrant 1 of the non-inertial coordinate system, $X'Z'$, the system starts accelerating. Finally when the unbalanced mass, m , is in quadrant 2 of the non-inertial coordinate system, $X'Z'$, the system starts to decelerate, and the unbalanced mass, m , has completed one full cycle. In Fig-

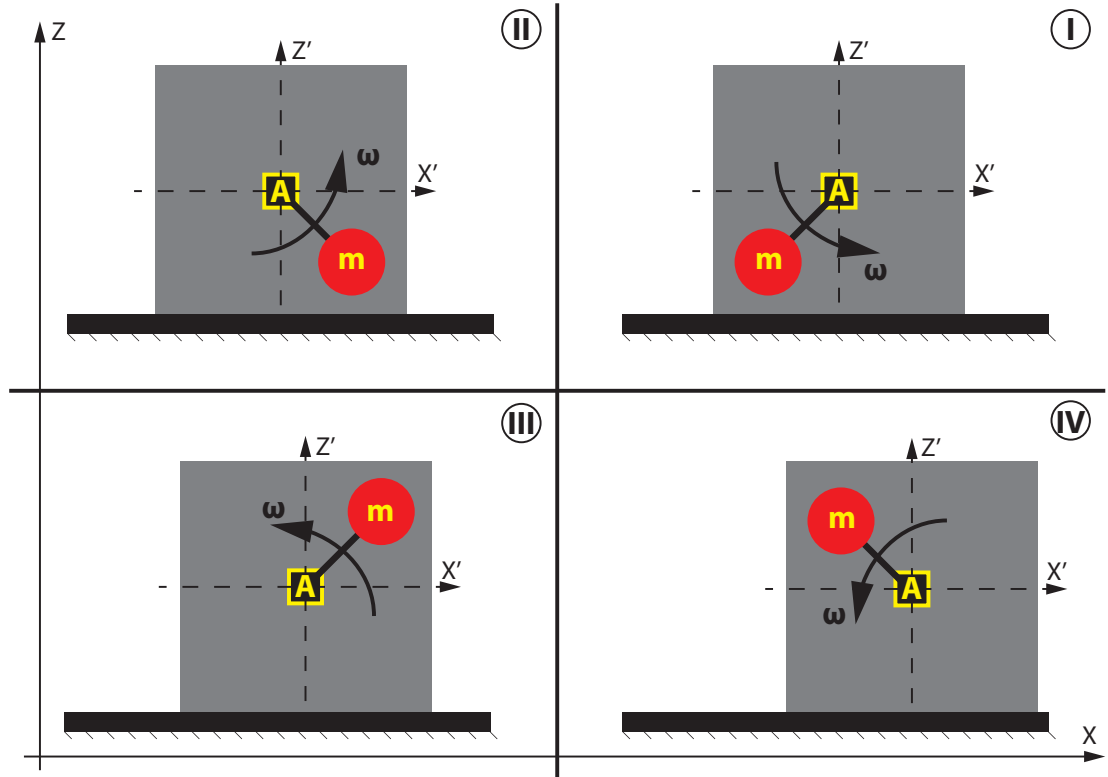


Figure 2.2: Graphical representation of the conversion of vibration motion to translational motion as the unbalanced mass, m , passes through four stages

Figure (2.3) acceleration, velocity and displacement plots of the base-cylinder are shown as the unbalance mass makes one full rotation.

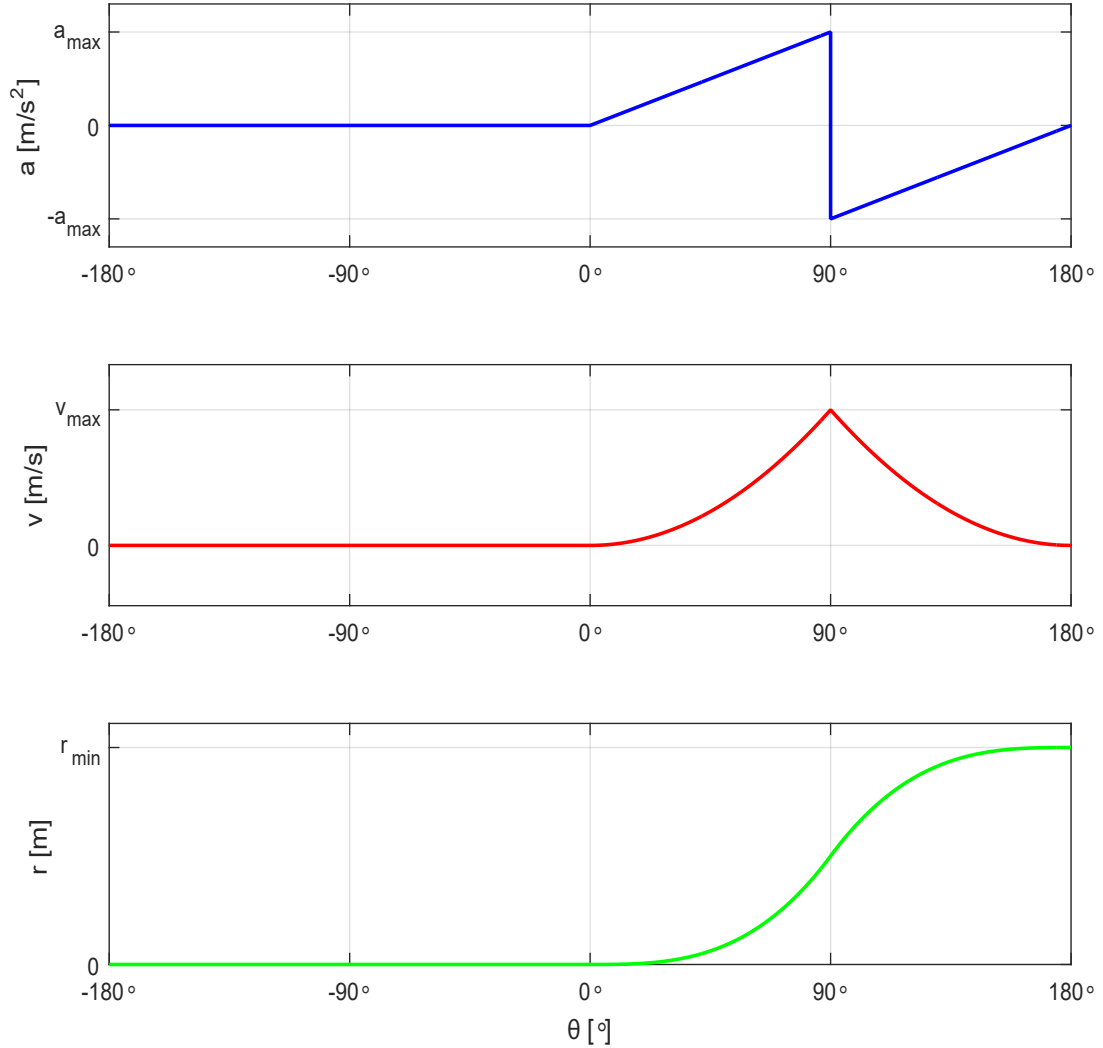


Figure 2.3: Acceleration, velocity and displacement plots supporting the working principle of the locomotion system based on Figure (2.2)

2.2 Actuators

The locomotion system presented in the previous section requires an actuator that will rotate the unbalance mass and an actuator that will control the angular

position of the steering-cylinder in the inertial coordinate system, XYZ . Off-the-shelf actuators that are able to accomplish above tasks are vibration motor and servo motor. Additional requirement in the choice of actuators is the spatial limitation, as stated in the chapter's introduction, the MechaCell would have cylindrical shape with height and diameter both 5 [cm].

Many different designs of vibration motors that fit the requirements of the locomotion system can be found, Figure (2.4) presents different types of cellphone vibration motors available on the market. One more additional design imposition on the vibration motor is symmetry requirement such that the rotating unbalance would be in the middle. For these purposes the shaft-less vibration motor is chosen, bottom left in Figure (2.4) highlighted with red. This vibration motor has diameter of 1.2 [cm] and thickness of 0.3 [cm], more detailed specs of the vibration motor are presented in Appendix A.4.

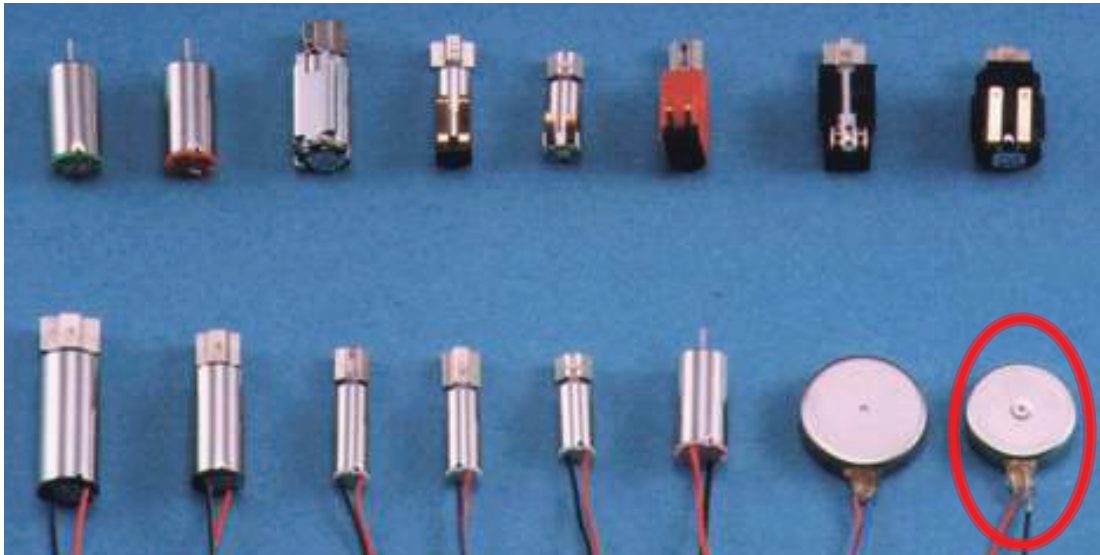


Figure 2.4: Different types of cellphone vibration motors available on the market, vibration motor used in the MechaCell is highlighted

Two types of actuators exist for the second actuator that can fix the angular orientation of the steering-cylinder in world coordinate system, one type of motors are the servo motors and the second type of motors are the stepper motors. Servo motors have integrated feedback system for the angular position of the

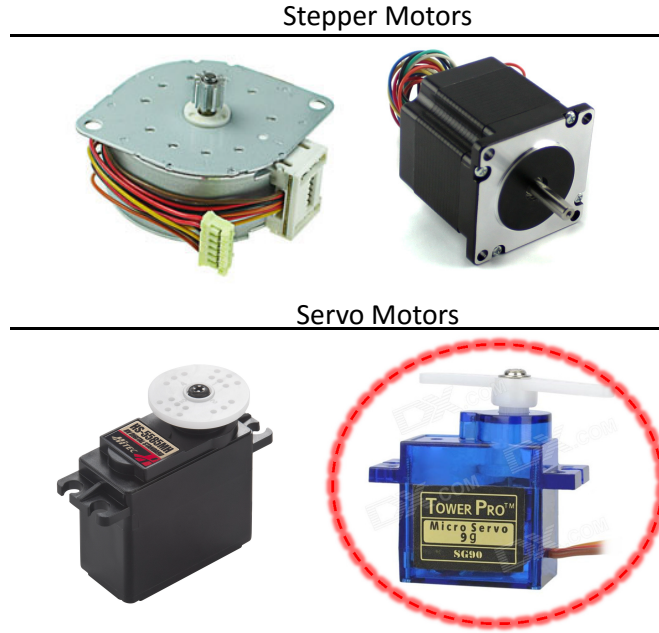


Figure 2.5: Actuator candidates for angular position fix of the steering cylinder, stepper and servo motors

shaft according to an internal reference, hence servo motor is chosen as the second actuator. In Figure (2.5) stepper and servo motors are shown, in the same figure in the bottom left the servo motor used in the MechaCell is presented and highlighted. Again due to the spatial limitations the smallest servo motor on the market is chosen, that is a "TowerPro SG90" servo motor with dimensions: $[l \times w \times h] = [3.0 \ 1.2 \ 28.6] \ [cm]$, more detailed specs of the servo motor are presented in Appendix A.5.

In Figure (2.6) both actuators are presented, the servo motor on the left and the vibration motor on the right.

2.3 Electronics

In this section, electronic components of the MechaCell are explained in detail. It should be noted that all electronic components are readily available on the market.

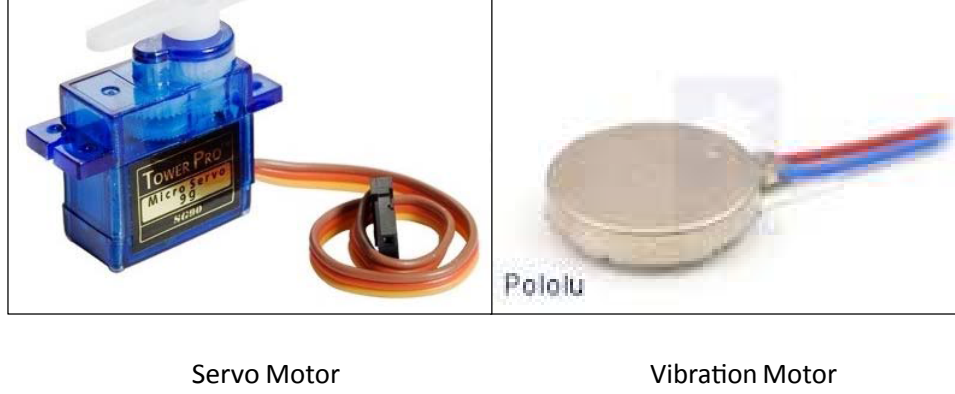


Figure 2.6: Actuators of the MechaCell, servo motor on the left and vibration motor on the right

Section 2.2 contains the actuation system of the MechaCell. Electronic components are chosen with accordance to that system. Actuation system consists of two actuators: a servo motor and a vibration motor. Electronic components chosen should be able to control both actuators, should have memory for a control algorithm and, should be able to communicate.

Arduino board, a programmable board with easy to use user interface and many digital input/output pins, is chosen.

There are eighteen different Arduino boards on the market all with very similar capabilities, all different Arduino boards are shown in Figure (2.7). MechaCell device has small size, hence the smallest Arduino board available is chosen, that is Arduino ProMini board with dimensions $[l \times w] = [3.3 \ 1.8] \ [cm]$, highlighted in Figure (2.7). More detailed specs of the Arduino ProMini board can be found in Appendix A.1. Due to the lack of USB or mini USB connection on the ProMini board, a FTDI (USB to ttl serial connection converter) cable was used to program the arduino ProMini.

Analog pins on all arduino boards including the ProMini can only be used as inputs. To provide an allusion of analog output using a digital pin, Pulse Width Modulation (PWM) technique is used (implemented in the design of the arduino

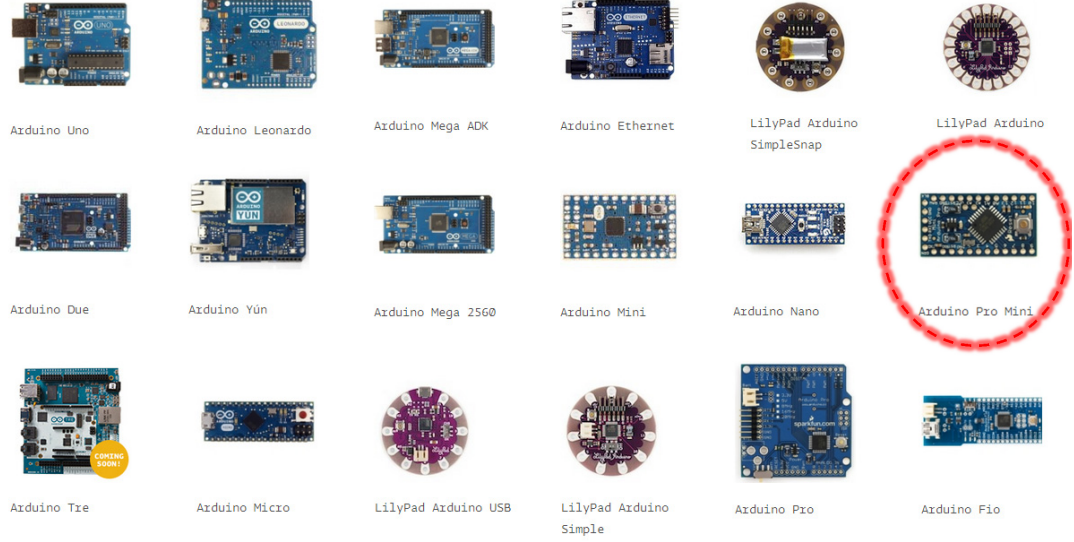


Figure 2.7: Different Arduino boards available on the market

boards). [39] From the fourteen digital pins on arduino boards, six of those can be used as PWM pins, rest of the digital pins are of on/off type.

Servo motor has three connections, power, ground and signal connection. Power and ground connections are directly connected to the $3.3V$ source and GND pins on the arduino boards respectively, and the signal connection can be attached to any digital pin on the arduino board.

Translational speed of the MechaCell is directly dependent on the rotational speed of the unbalance mass, hence translational speed is controlled through the control of the rotational speed of the rotating unbalance mass. This requires that the speed of the vibration motor is controlled. In addition, direction of rotation (CW/CCW) of the rotating unbalance mass should be controlled due to the limitation imposed by the the servo motor which can only rotate 180° .

Vibration motor is a DC-motor and has only two connections, however direct connection to the arduino board is not recommended because maximum DC-current that can supplied from any IO pin on the arduino board is $40 [mA]$, and the required DC-current for the motor is $120\text{--}150 [mA]$. In addition the DC-motor has to be driven in both direction (Clock Wise-CW and Counter Clock

Wise-CCW) which requires a polarity change that cannot be done from the arduino board. DC-current requirements and bidirectional drive are accomplished by implementing a motor driver that is controlled by the board. L293C motor driver is used to drive the vibration motor, cheap and readily accessible in the lab, L293C motor driver is shown in Figure (2.8). L293C motor driver can supply up to 600 [mA] of DC-current. Motor driver consumes power directly from the power supply of the MechaCell (in this work MechaCell is powered with an external power source). Pin connections between the arduino board and the motor driver are listed in Table (2.1). Speed of the vibration motor is controlled with

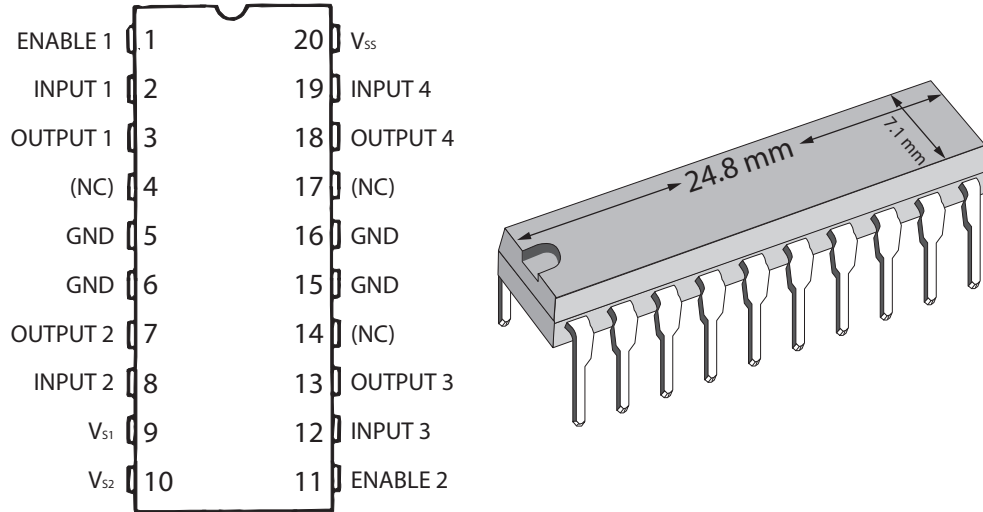


Figure 2.8: L293C motor driver with pin labels

Table 2.1: L293C motor driver pin connections

Motor Driver (<i>L293C</i>)	CONNECTIONS
Power	Power Supply
GND	GND on Arduino
INPUT 1 (DIR)	Digital pin 1 (arduino)
INPUT 2 (DIR)	Digital pin 2 (arduino)
OUTPUT 1	Vibration Motor
OUTPUT 2	Vibration Motor
LOGIC (V_{ss})	Digital pin 3 (arduino)
ENABLE 1	PWM pin (arduino)

the PWM pin on the arduino board that is connected to the ENABLE pin on the L293C motor driver. Direction of rotation is controlled by two digital pins on the arduino board connected to the two input pins on the L293C motor driver. In Table (2.2) combinations of the input pins statuses (digital pin status: LOW or HIGH) on the L293C motor driver that determines the rotation direction are listed. Vibration motor terminals are connected to the output pins of the L293C motor driver. The logic pin on the L293C (pin 20) motor driver determines the

Table 2.2: L293C motor driver input pin status combinations that determine the direction of rotation of the vibration motor

Pin Status		Direction
Input Pin 1	Input Pin 2	
HIGH	LOW	CW
LOW	HIGH	CCW
HIGH	HIGH	STOP
LOW	LOW	STOP

digital HIGH and LOW levels and is connected to a digital pin on the arduino board, digital HIGH is 3.3 [V].

MechaCell to PC and MechaCell to MechaCell communication is established wirelessly using Bluetooth technology. Bluetooth technology's low power consumption is the reason for choosing it over wireless LAN-networking. RN42-Bluetooth module from Roving Networks is powerful, small and easy to use and is the communication component of the MechaCell, shown in Figure (2.9). In Table (2.3) pin connections between the arduino board and the Bluetooth module are shown.

Two data pins and two power pins from the Bluetooth module are connected to

Table 2.3: Bluetooth module to arduino board pin connections

Bluetooth Pns	Arduino Pins
<i>Rx</i>	<i>Tx</i>
<i>Tx</i>	<i>Rx</i>
<i>GND</i>	<i>GND</i>
<i>V_{in}</i>	3.3 V

the arduino board. The Receive (*Rx*) pin on the Bluetooth module is connected

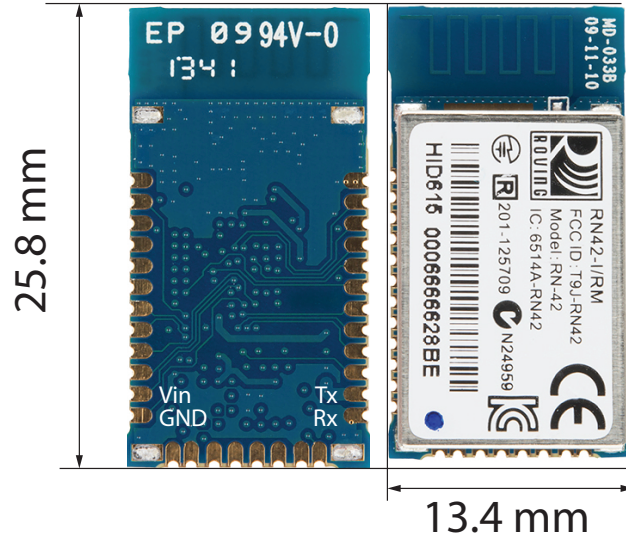


Figure 2.9: RN42 Bluetooth module from Roving Networks

to the Transmit (Tx) pin on the arduino board and the Tx pin on the Bluetooth module is connected to the Rx pin on the arduino board, ground (GND) pin of the Bluetooth module is connected to the GND pin of the arduino board and power pin (V_{in}) of the Bluetooth module is connected to the power pin of the arduino board.

2.3.1 Sensors

An electronic component that can detect a change in a physical quantity and transform that into an electric signal is a sensor. In the MechaCell, sensors that can detect change in orientation, rotational speed, acceleration and force will be implemented in the future.

Changes in rotational speed and orientation will be detected by a triple-axis gyroscope. ITG-3200 single chip digital-output triple-axis gyroscope from InvenSense. The chip will be used with a breakout board from Sparkfun electronics.

Change in acceleration will be detected by a small, thin, low-power, three-axis

ADXL345 accelerometer from Sparkfun electronics. Breakout board from Sparkfun electronics will be used in the accelerometer implementation in the MechaCell.

An FSR pressure sensor will be integrated in the MechaCell for force detection. MechaCell is thought to manipulate objects in plane, alone or in swarm, by only applying pushing force on the perimeter of the manipulated object; hence, sensing the contact force exerted on the object is important. For this purpose the FSR pressure sensor from Sparkfun electronics will be used.

In Figure (2.10) triple-axis ITG3200 gyroscope, triple-axis ADXL345 accelerometer with breakout boards and the FSR force sensor are shown.

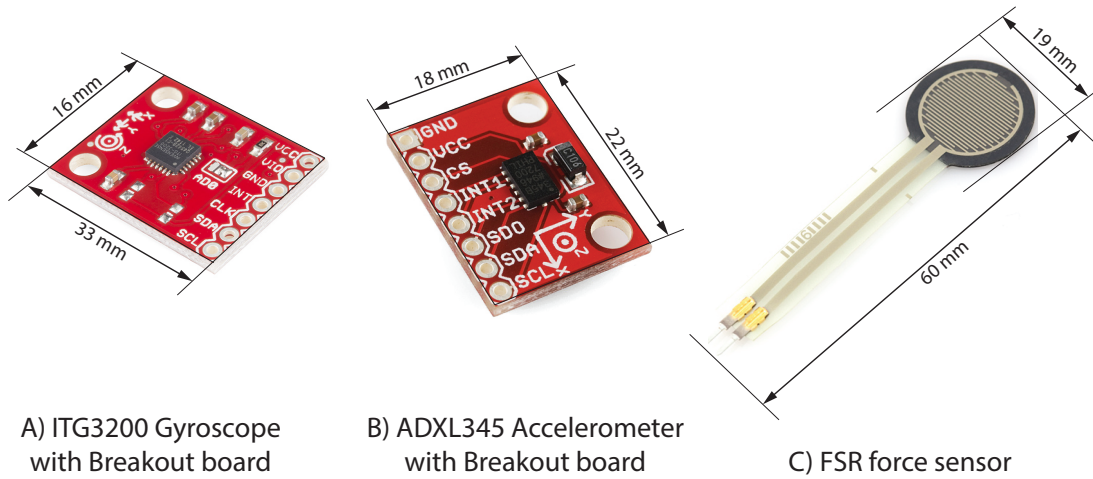


Figure 2.10: Sensors that will be implemented in the future MechaCell

2.3.2 Software

Arduino ProMini board is programmed with Arduino IDE environment, which can be downloaded for free at the arduino homepage [40]. Control algorithms can be directly downloaded to the Arduino board through this software over a USB

cable. Controller design and tuning is performed in Simulink, and using the real system. This requires uploading new program for every new configuration of the controller. To bypass this issue, a program developed by Giampiero Campa is used. This ArduinoIO package can be downloaded from Matlab file exchange web page [41]. The package includes code written in IDE environment to be uploaded to the Arduino board shown in Appendix B.3 and a Matlab script ran in Matlab. The script creates an Arduino object in Matlab and Simulink such that live communication is carried out between the Arduino board and Simulink. Connection between the Arduino board and the PC is established either through a USB cable or Bluetooth.

Both the IDE code and the Matlab script developed by Campa were intended for Arduino UNO board and communication with a PC through USB cable. Using this program with the ProMini board and Bluetooth technology required some changes in the source code of both the IDE code and the Matlab script.

Default communication baudrate of a serial port is set to 112 000 $[Bd]$, however the ProMini board cannot communicate at that speed. ProMini board communication speed is limited at 57 000 $[Bd]$ baudrate.

In the Matlab script developed by Campa, serial object should be replaced with a Bluetooth object and baudrate speed set to 57 000 $[Bd]$, so that Bluetooth communication is established with the ProMini board.

2.4 Main Body

In this section, design of the main body of the MechaCell is discussed in detail. MechaCell is a compact device in which all elements (actuators and electronic) are enclosed within. As stated in Section 2.1, shape of the MechaCell is a cylinder with equal height and diameter, $D = H = 5 [cm]$. Cylindrical shape is chosen such that when the MechaCell participates in an object manipulation task as a swarm, its contact with the object is a point.

In Section 2.2 and Section 2.3 actuators and electronic components of the MechaCell were presented. Main body of the MechaCell is designed such that it can fit all components inside the cylindrical shape. The body is divided into two sections. Section one is the locomotion system design and the fixture of the vibration motor. Section two is the fixture of the servo motor to the main body, connection to the locomotion mechanism, and electronic components fixtures.

Section I: In Figure (2.11) part 1 is the base of the cylinder in contact with the floor. This part has extensions on it like bristles but rigid, such that it will aid in the stick-slip motion of the MechaCell. Base cylinder and steering cylinder that were discussed in Section 2.1.1 are shown in Figure (2.11) with numbers 1–5. The steering cylinder is part number 3 and vibration motor (labeled with v) is attached to it. The base cylinder consists of parts 1,2,4 and 5, such that part one is the contact part as discussed above, part 3 (steering cylinder) is squeezed between parts 2 and 4 such that it is able to rotate freely and part 2 is fixed on part 1. Part 5 is fixed on part 4 and provides support for the connections to the servo motor fixture, connecting parts (parts 6, 7 and 8) and the FSR force sensor (shown in Figure (2.12)).

Section II: Servo motor (labeled with S) is fixed on part 9 (servo motor’s fixture) shown in Figure (2.12). The steering cylinder is connected to the servo motor’s shaft through parts 6 and 8 and servo motor’s fixture is fixed on part 7, visible in Figure (2.11). All electronic components are located on the servo motor’s fixture and this space is enclosed with a cover (part 10) given in Figure (2.12). In Figure (2.13), assembly steps of the MechaCell are shown.

2.5 Manufacturing

In Section 2.4, parts of the MechaCell and their assembly steps were discussed. Assembly of the parts is summarized in Figure (2.13).

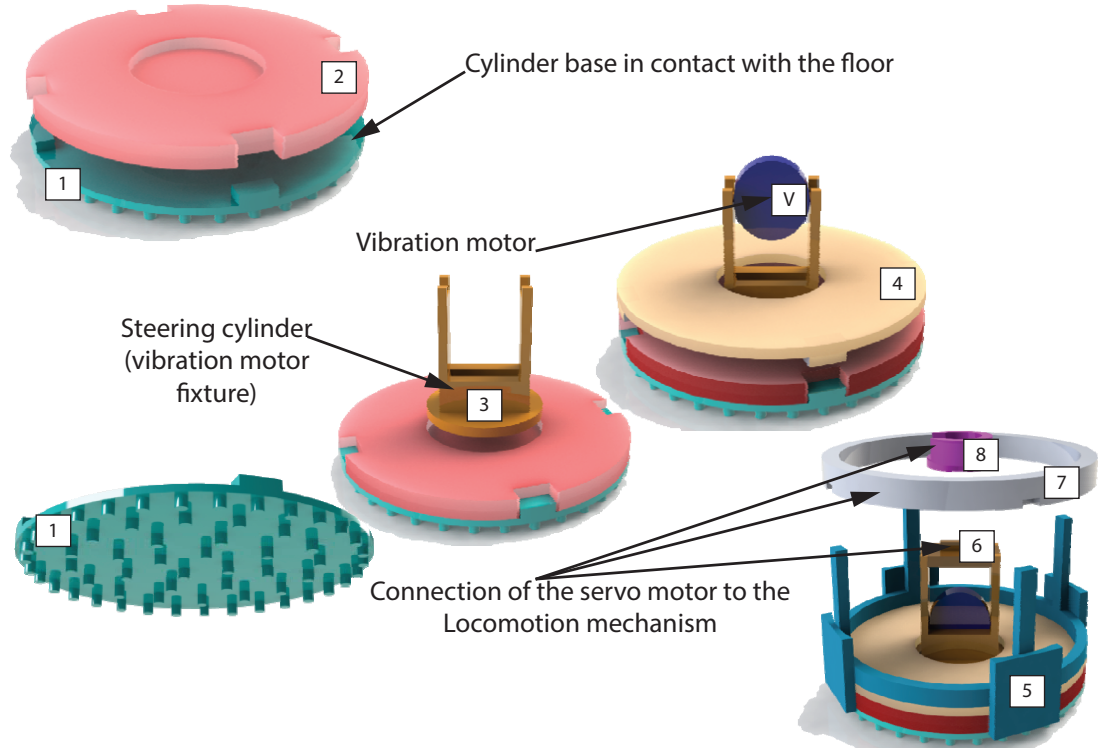


Figure 2.11: CAD drawing of parts of Section I of the Main Body design

Additive manufacturing process (3D–printing) was used to manufacture MechaCell’s parts. 3D printer used in the manufacturing process was *Dimension sst1200*. Immobile parts were glued together with super glue. Manufactured parts are laid out in Figure (2.14). Assembled CAD drawing and assembled manufactured MechaCell is shown in Figure (2.15).

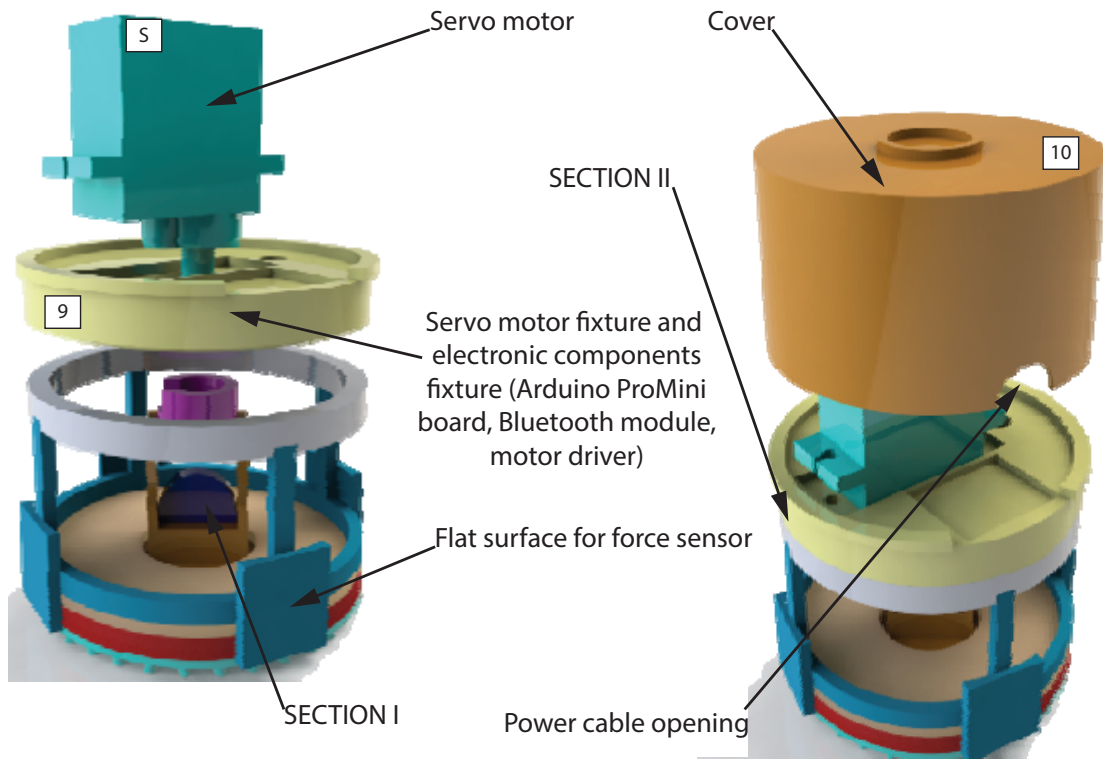


Figure 2.12: CAD drawing of parts of Section II of the Main Body design

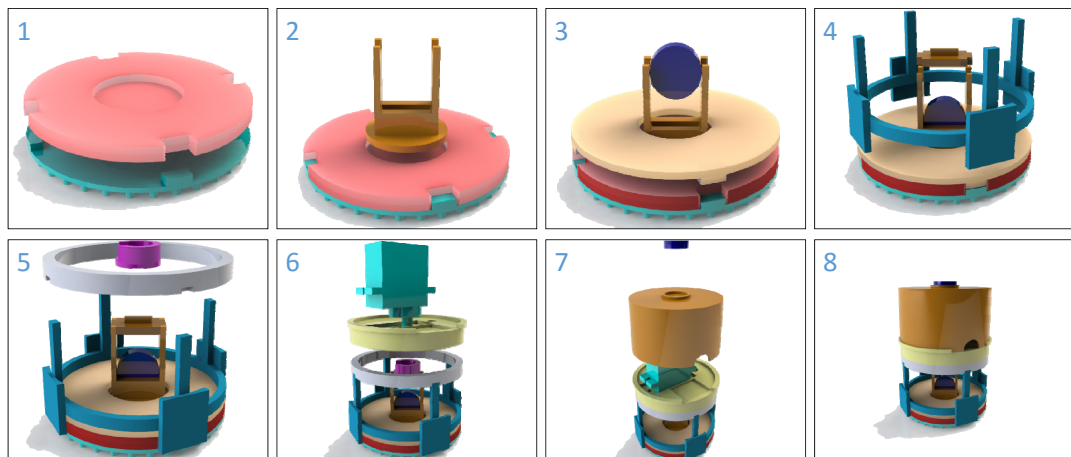


Figure 2.13: CAD drawings of mechanical components of the MechaCell and assembly

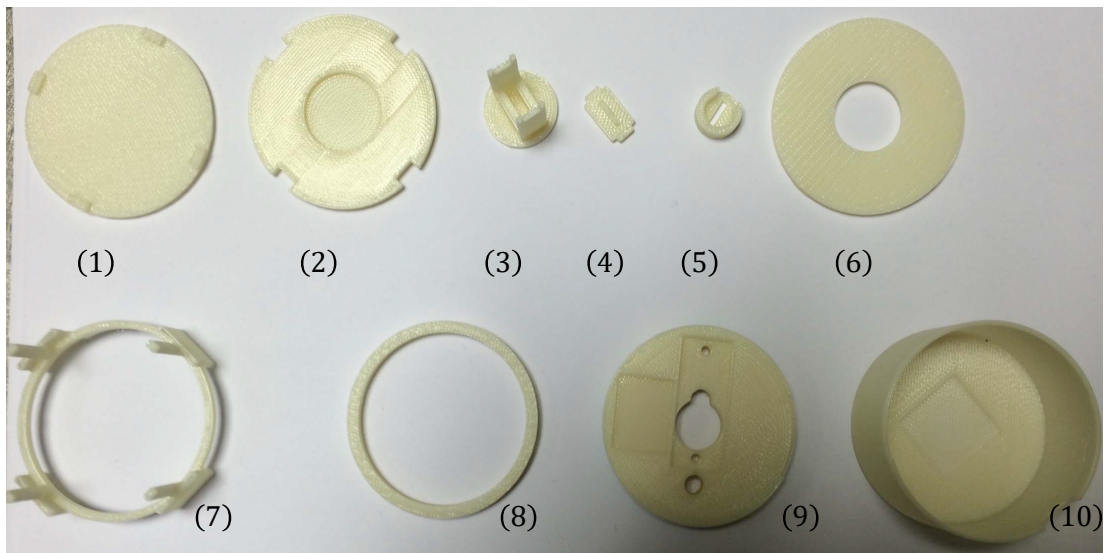


Figure 2.14: Manufactured parts of the MechaCell

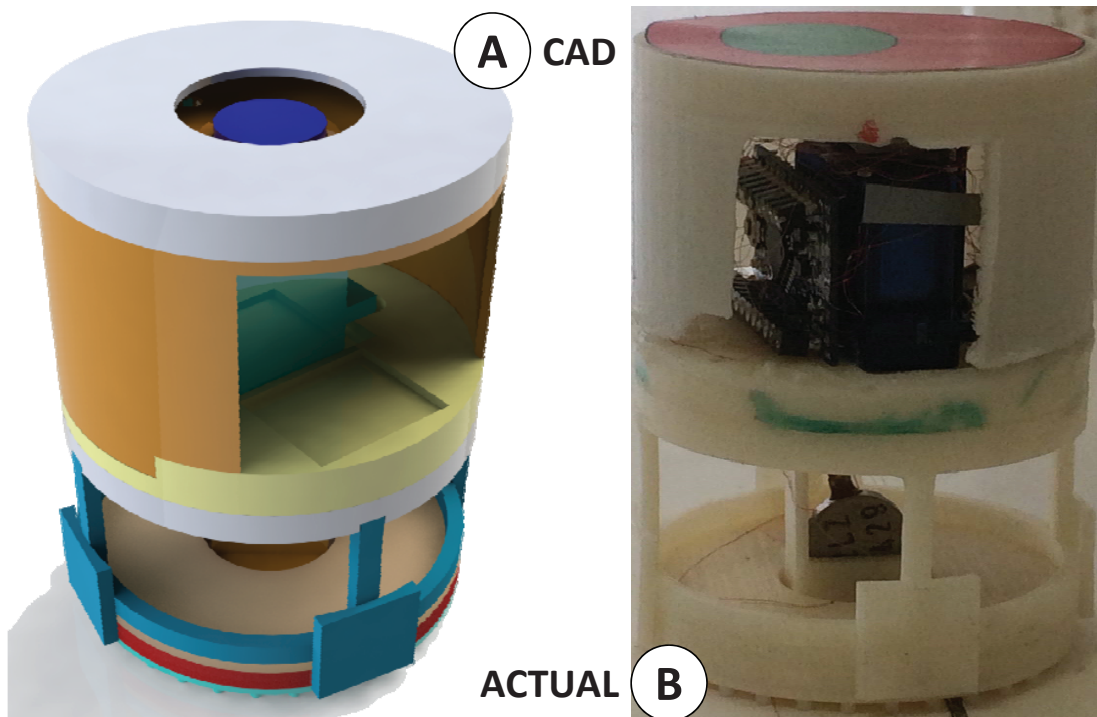


Figure 2.15: CAD and Actual MechaCell

Chapter 3

Modeling & Control

In this chapter, mathematical model of the MechaCell is presented. First, the MechaCell is assumed to be a point mass and actuator forces are assumed to be external forces. Then, a more realistic model of the MechaCell is developed in Simulink's SimMechanics environment. This model includes a 3D model of the MechaCell in which the rotating unbalanced mass modeling is also presented. MechaCell–MechaCell interaction and MechaCell–Workpiece interaction is modeled in SimMechanics environment.

Another part of this chapter is the control strategy of the MechaCell. Controller of the MechaCell is divided into two sub-controllers, a translation controller (t-controller) and a steering controller (s-controller). Control strategies are implemented into the SimMechanics model and simulations are carried out. Simulations include path following (single MechaCell simulations) and guiding a workpiece to follow a prescribed path (multi-MechaCell simulations).

3.1 Mathematical Model

In Figure (2.1) of Chapter 2, a 3D sketch explaining working principle of the MechaCell's locomotion system was presented. In the same figure, two coordinate

systems are defined: an inertial coordinate system, XYZ , and a non-inertial coordinate system, $X'Y'Z'$. The non-inertial coordinate system is attached to the steering cylinder of the MechaCell such that the unbalanced mass is rotating about Y' -axis at all times; hence, X' -axis points the direction of locomotion of the MechaCell in the inertial coordinate system and Z' -axis is coincident with the Z -axis.

Projected view of the 3D sketch in Figure (2.1) on the $X'Z'$ -plane is shown in Figure (3.1, A) and in Figure (3.1, B) a free body diagram (FBD) based on Figure (3.1, A) is shown. Based on the FBD in Figure (3.1, B) magnitude of the centripetal force, $\mathbf{F}_u$, produced by the rotating mass, m , as a function of its rotational speed, ω , is governed by Equation (3.1).

$$|\mathbf{F}_u(\omega)| = mr\omega^2 \quad (3.1)$$

where, r is the length of the rod which connects mass m to the steering cylinder shown in Figure (3.1, A).

Centripetal force, $\mathbf{F}_u$, is decomposed in the $X'Z'$ -plane, and magnitudes of the components of centripetal force in X' -direction, $F_{uX'}$, and in Z' -direction, $F_{uZ'}$, are given in Equation (3.2a) and Equation (3.2b) respectively.

$$F_{uX'}(\omega) = |\mathbf{F}_u(\omega)|\cos(\theta) \quad (3.2a)$$

$$F_{uZ'}(\omega) = |\mathbf{F}_u(\omega)|\sin(\theta) \quad (3.2b)$$

where, angle, $\theta = \omega t$, is the angle formed by the positive X' -axis and the rod of the unbalance. Steering-cylinder and base-cylinder together have mass M , as shown in Figure (3.1). Forces acting on this system (steering and base cylinder together) are: gravitational force, centripetal force (from the rotating unbalance), friction and normal force from the floor. Weight of the unbalanced mass, m , is neglected. Magnitude of the normal force is shown in Equation (3.3).

$$N(\omega) = Mg + F_{uZ'}(\omega) \quad (3.3)$$

Coulomb's friction model with static and dynamic coefficients of friction, μ_s and μ_k respectively, is used to model the friction force. Friction force is given in

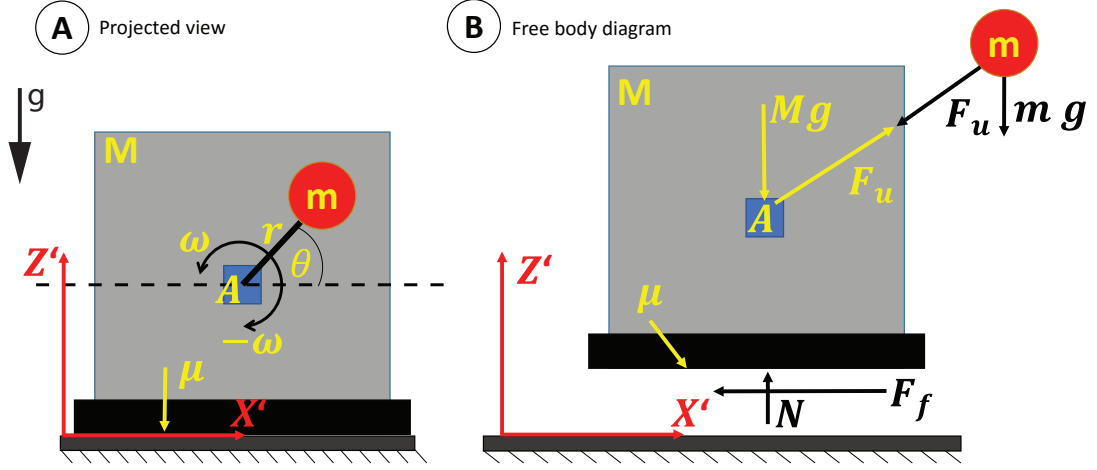


Figure 3.1: Projected view on XZ -plane of Figure (2.1) and corresponding FBD

Equation (3.4).

$$F_f(\omega) = \begin{cases} -F_{uX'} & |F_{uX'}(\omega)| \leq |\mu_s N| \\ -\mu_k N(\omega) \text{sign}(\dot{X}') & |\dot{X}'| \neq 0 \end{cases} \quad (3.4)$$

Where, $\dot{X}'$, is the velocity along X' -axis.

MechaCell represented with a point mass in the inertial coordinate system is shown in Figure (3.2). In the same figure actuating force, $F_{uX'}$, and friction force, F_f , acting on the MechaCell are shown. Total force along X' -axis acting on the MechaCell combining Equations (3.1),(3.2),(3.3) and (3.4) is represented in Equation (3.5) as a function of rotational speed ω , mass of the MechaCell M , length of the connecting rod r and mass of the rotating unbalance m .

$$F_{X'total} = mr\omega^2 \cos(\omega t) + \begin{cases} -mr\omega^2 \cos(\omega t) & |F_{uX'}(\omega)| \leq |\mu_s N| \\ -\mu_k (Mg + mr\omega^2 \sin(\omega t)) \text{sign}(\dot{X}') & |\dot{X}'| \neq 0 \end{cases} \quad (3.5)$$

As shown in Equation (3.5) total force acting on the MechaCell in the XY -plane is dependent on four parameters (M, m, r, ω), and can be manipulated by varying any of these parameters. However all parameters are fixed except for the rotational speed, ω , of the unbalance mass. Translational speed of the MechaCell

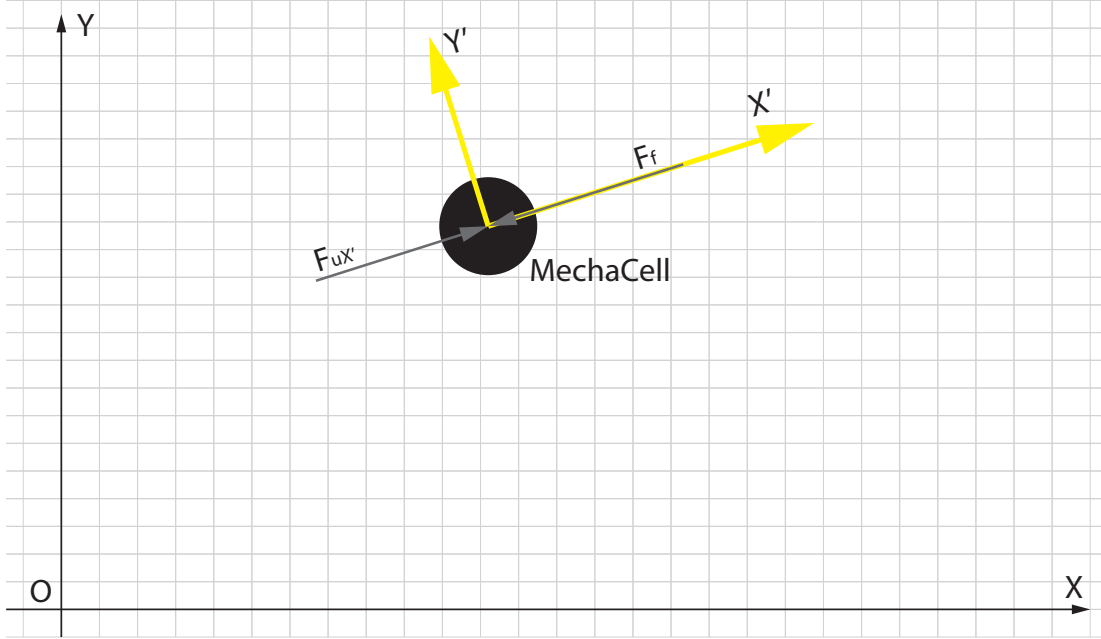


Figure 3.2: MechaCell represented as a point mass in the XY -plane of the inertial coordinate system

is directly dependent on the force presented in Equation (3.5) therefore directly dependent on the rotational speed, ω , of the unbalance mass.

The simple model developed here is used for the friction compensation algorithm in the control part.

3.2 Model of the MechaCell in SimMechanics

To capture more realistic behavior of the MechaCell's locomotion system, a model in SimMechanics is built. In the SimMechanics environment, MechaCell is no longer modeled as a point mass; instead it is modeled as a 3D body. Properties of the 3D body (moments of inertia, center of mass, etc) are automatically calculated by the software from the CAD model of the part and the mass of the part. Second generation of SimMechanics has an option to insert CAD models directly from third party CAD software like SolidWorks. In Chapter 2, CAD drawings of all parts of the MechaCell were presented; however, in order to implement that

design into the SimMechanics environment, some parts had to be joined together. Hence, in SimMechanics environment, MechaCell is constructed from three parts only, the main body part–part 1, steering cylinder part–part 2 and the rotating unbalance part–part 3, shown in Figure (3.3).

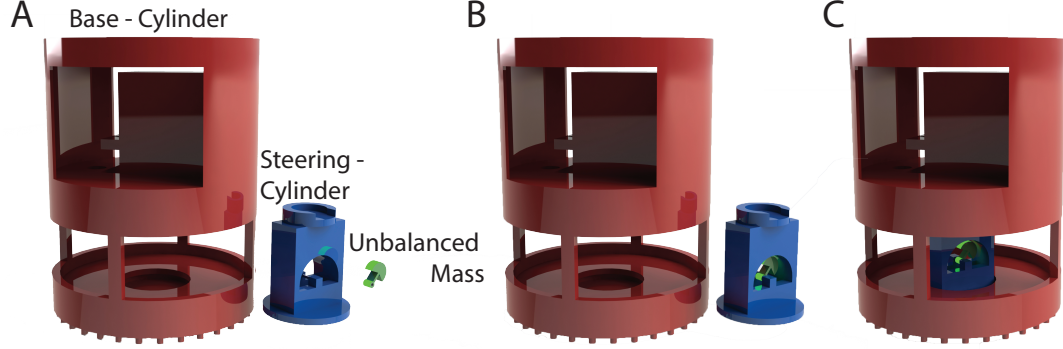


Figure 3.3: CAD drawings of the parts used in SimMechanics to model the MechaCell; A) Exploded view of all three parts; B) Rotating unbalance is fixed in the steering cylinder; C) Steering cylinder is fixed inside the main body;

Simulation environment in SimMechanics is constructed from a floor of 50×50 [cm] enclosed with walls on all four sides. MechaCell is resting on the floor and its motion is constrained by the walls. A snap shot from the SimMechanics environment where a single MechaCell is resting on the floor is shown in Figure (3.4). Three parts of the MechaCell are labeled and annunciated in this figure.

3.3 Force Models in SimMechanics

In this section, force models that are used by the SimMechanics software are presented. Two types of forces can be differentiated between two bodies in contact: normal and tangential forces. These forces are modeled in SimMechanics as contact and friction forces respectively. Contact between two bodies is shown Figure (3.5).

The diagram illustrates the contact between two rigid bodies, Body 1 and Body 2, in a 2D Cartesian coordinate system. The X-axis is horizontal and the Y-axis is vertical. Body 1 is a large, irregularly shaped object with a thick blue outline. Body 2 is a smaller, irregularly shaped object with a thick red outline. They are in contact at a point, which is highlighted by a small green circle. At this contact point, a normal vector n (black arrow) points from the contact point into Body 1, and a tangent vector t (black arrow) points along the boundary of Body 1. The contact point is also marked with a small green circle. The diagram shows the contact point and the normal and tangent vectors at the contact point. The normal vector n is perpendicular to the contact surface, and the tangent vector t is parallel to the contact surface. The contact point is marked with a small green circle. The diagram shows the contact point and the normal and tangent vectors at the contact point.

31

3.3.1 Contact Forces

Contact forces are modeled according to two laws in SimMechanics. Law 1 is contact between a circle and a line (circle–line law). Law 2 is contact between a circle and a circle (circle–circle law). Both laws are graphically shown in Figure (3.6). In the definition of contact between two bodies in SimMechanics, bodies are designated as "Base" body and "Follower" body, this designation depends on the force law.

Base body in force law 1 should be the body represented by the line. X -axis of the coordinate system associated with that body should be perpendicular to the line, follower body is represented by the circle. Equations governing this law are given in Equation (3.6).

$$F_x = \begin{cases} kx_{pen} + bv_{pen} & x_{pen} > 0, v_{pen} > 0 \\ kx_{pen} & x_{pen} > 0, v_{pen} < 0 \\ 0 & x_{pen} \leq 0 \end{cases} \quad (3.6)$$

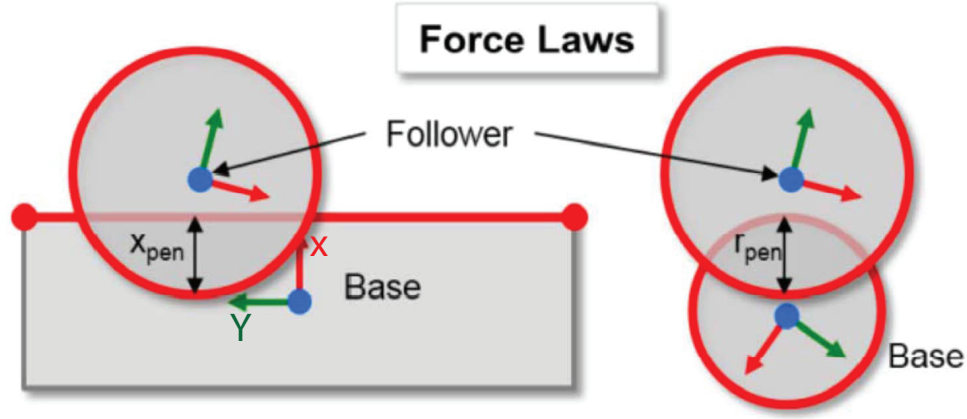
$$F_y = 0$$

where, $(x)_{pen}$ is penetration depth, b and k constants are contact damping and contact stiffness constants, respectively, and v_{pen} , is speed of penetration of the follower body with respect to the base body.

In the second force law (circle–circle law) choice of base and follower body is indifferent, both bodies can be either base body or follower body. Governing equations of the second law are shown in Equation (3.7).

$$F_r = \begin{cases} kr_{pen} + bv_{pen} & r_{pen} > 0, v_{pen} > 0 \\ kr_{pen} & r_{pen} > 0, v_{pen} < 0 \\ 0 & r_{pen} \leq 0 \end{cases} \quad (3.7)$$

where, $(r)_{pen}$ is penetration depth, b and k constants are contact damping and contact stiffness constants, respectively, and v_{pen} , is speed of penetration of the follower body with respect to the base body.



Contact Stiffness (k): Spring stiffness for force law
Contact Damping (b): Damping constant for force law

Figure 3.6: Graphical representation of circle-line and circle-circle force laws

3.3.2 Friction Forces

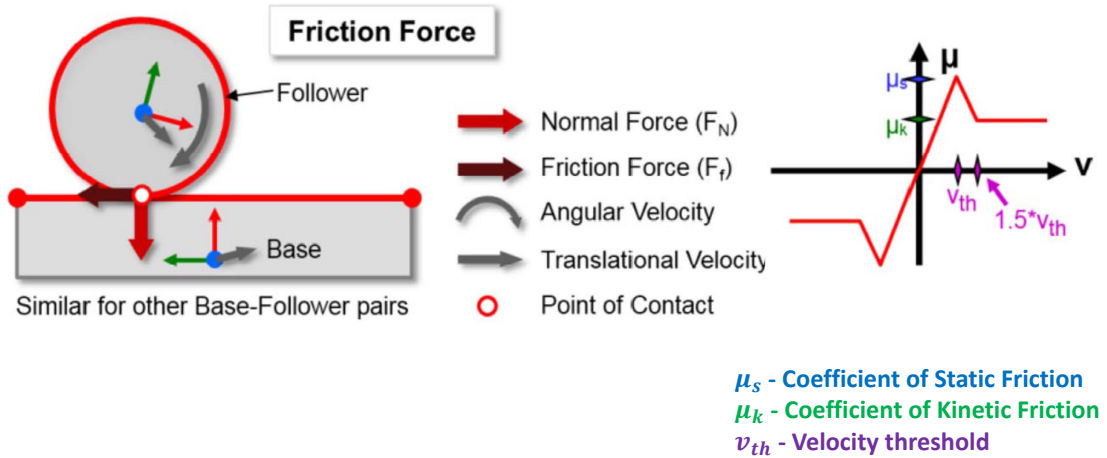


Figure 3.7: Graphical representation of friction force law

Similar to the contact laws, in the friction law, base and follower body should be defined. Both base-follower definitions for circle-line and circle-circle force laws are valid here. In Figure (3.7) circle-line contact configuration is shown. Friction force is given in Equation (3.8).

$$F_f = F_N \mu \quad (3.8)$$

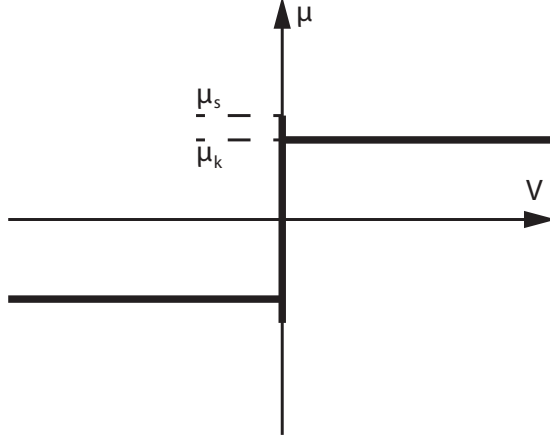


Figure 3.8: Coulomb's friction model

where, F_N is the normal force. μ is the coefficient of friction modeled as a function of the velocity of the point of contact, and is given in Equation (3.9).

$$\mu(v_{poc}) = \begin{cases} \frac{v_{poc}}{v_{th}} \mu_s & v_{poc} < v_{th} \\ \mu_s - v_{poc} \frac{(\mu_s - \mu_k)}{0.5v_{th}} & v_{th} \leq v_{poc} \leq 1.5v_{th} \\ \mu_k & v_{poc} > v_{th} \end{cases} \quad (3.9)$$

where, v_{poc} is the velocity of the contact point, μ_s and μ_k are the static and kinetic coefficients of friction, respectively and v_{th} is threshold velocity. This is an unrealistic continuous friction model used by the SimMechanics solver. Keeping the threshold velocity close to zero, here threshold velocity's value is 10^{-5} , this model converges to coulomb's friction model shown in Figure (3.8).

3.4 Control Algorithm

In this section controller of the MechaCell is discussed. Two sub-controllers make up the MechaCell's controller: a translational controller (t-controller) and a steering controller (s-controller). Working principle of the locomotion system was discussed in Chapter 2, the same chapter presented the actuators used in the mechanical design of the MechaCell (a vibration motor and a servo motor) to attain functions of the desired locomotion system; accordingly two sub-controllers, one for each actuator, construct the MechaCell's controller.

Mathematical model of the MechaCell was derived in Section 3.1 by assuming the MechaCell as a point mass. Total force locomoting the MechaCell is directly dependent on the rotational speed, ω , of the unbalance mass, m , as presented in Equation (3.5). Hence translational speed of the MechaCell is dependent on, ω , of m . Therefore, rotational speed of the unbalance mass i.e. rotational speed of the vibration motor is controlled by the t-controller.

As the name suggests angular orientation of the steering-cylinder i.e. angular orientation of the servo motor's shaft is controlled by the s-controller.

Both sub-controllers use position feedback control scheme, this control scheme is shown in Figure (3.9).

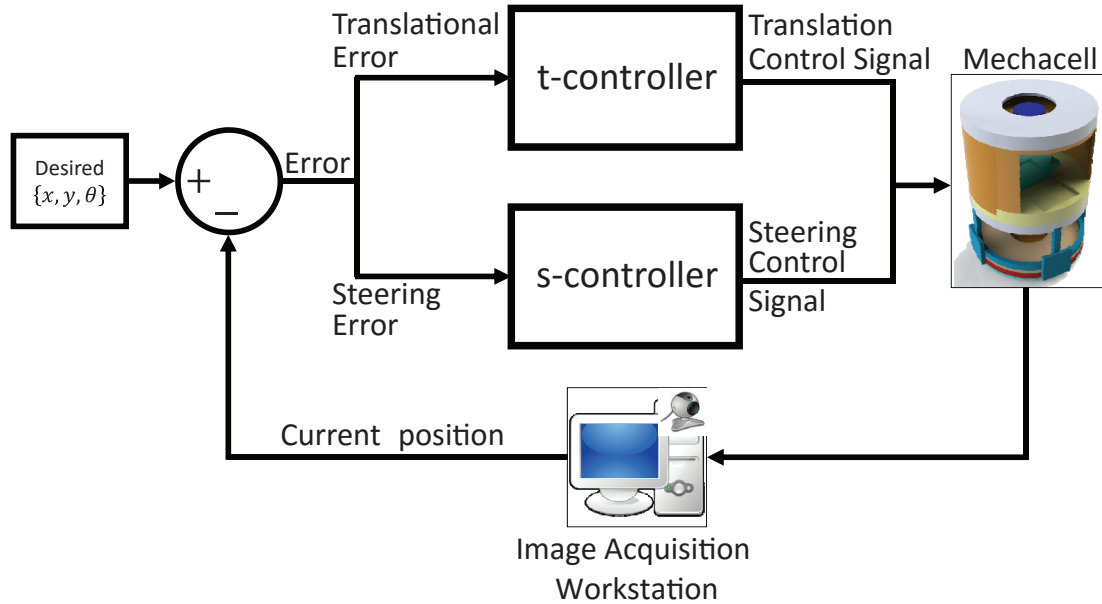


Figure 3.9: Block diagram of the control structured implemented in the MechaCell

3.4.1 Translational Control (t-controller)

Rotational speed of the unbalance mass is controlled by the translational controller (t-controller) of the MechaCell. The t-controller is comprised of two parts,

a linear controller (Proportional Integral Derivative–PID controller) supported by friction compensation algorithm as in [42]. In Figure (3.10) control scheme of the t–controller is shown. Friction observer block predicts magnitude of the frictional force using the mathematical model developed in Section 3.1.

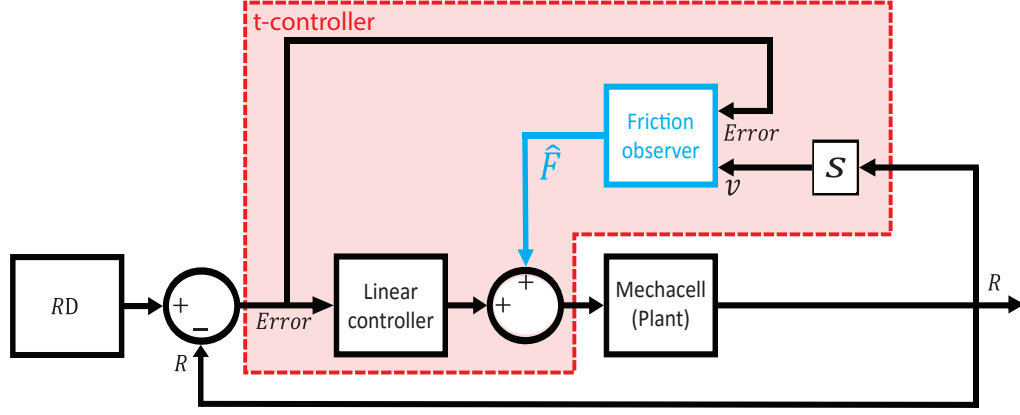


Figure 3.10: Block diagram of the t–controller consisting of a PID controller and a friction compensation algorithm

In Figure (3.11) a sketch of the top view of the MechaCell is shown. In this figure, MechaCell’s position vector and position vector designating the desired location based on the inertial coordinate system are shown. PID controller of the t–controller of the MechaCell uses a scalar error shown in Equation (3.10), whose calculation is based on the vectors shown in Figure (3.11).

$$e = |\mathbf{R}_D - \mathbf{R}_M| \quad (3.10)$$

Friction observer uses the mathematical model from Section 3.1 to estimate the magnitude of the friction force. Estimation is based on the rotational speed ω of m which is proportional to the magnitude of the position error given in Equation (3.10), and the translational speed of the MechaCell. Speed of the MechaCell is numerically calculated from the position change, its analytic formulation is given in Equation (3.11).

$$|v| = \left| \frac{d\mathbf{R}_M}{dt} \right| \quad (3.11)$$

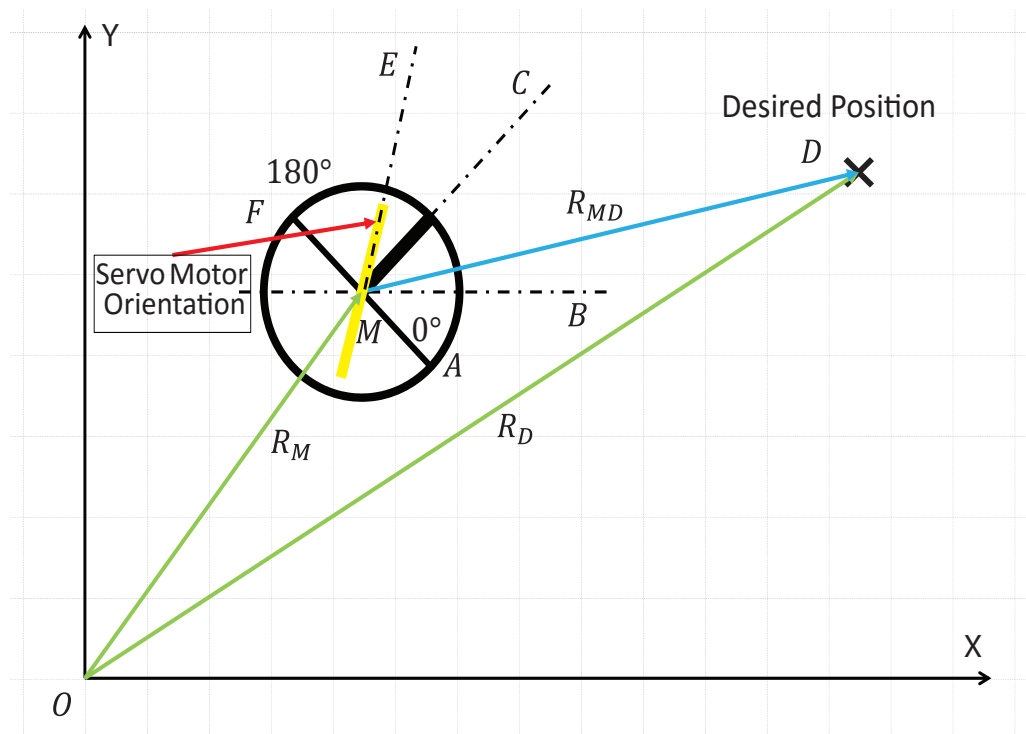


Figure 3.11: Top view of the MechaCell, its position, desired position are sketched

Table 3.1: ANGLE NOMENCLATURE FOR FIGURE (3.11)

$\angle AME \equiv \alpha_{servo}^{rel}$	Relative orientation of the servo motor measured from MA in CCW direction
$\angle BME \equiv \alpha_{servo}^{abs}$	Absolute orientation of the servo motor
$\angle BMD \equiv \alpha_{target}$	Orientation at which the servo motor should be fixed at
$\angle BMC \equiv \alpha_{MC}$	Orientation of the MechaCell
$\angle BMA \equiv \alpha_{lim1}$	This limit corresponds to $\alpha_{servo}^{rel} = 0^\circ$
$\angle BMA \equiv \alpha_{lim2}$	This limit corresponds to $\alpha_{servo}^{rel} = 180^\circ$

3.4.2 Steering Control (s-controller)

In Table (3.1) definitions based on Figure (3.11) of the parameters used by the s-controller in calculating the servo motor's angular position are listed. Design of the steering control algorithm requires that the MechaCell goes from any initial position to any desired position by following the straight line connecting the two points. This implies that the absolute orientation of the servo motor is equal to the target orientation. Using the parameters listed in Table (3.1), this requirement is shown in Equation (3.12). In Equation (3.13) relative angle of the servo motor is calculated; this angle is controlled by the s-controller.

$$\alpha_{servo}^{abs} \equiv \alpha_{target} \quad (3.12)$$

$$\alpha_{servo}^{rel} = \alpha_{MC} + \alpha_{servo}^{abs} - 90^\circ \quad (3.13)$$

In Equation (3.14), difference between the orientation of the MechaCell and the target angle is calculated. Angles are represented in the range $[-180^\circ \ 180^\circ]$. In order to conserve this convention, algorithm shown in Equation (3.15) is used. Angle of the servo motor is constrained to a specific range, $\alpha_{servo}^{rel} \in [0^\circ \ 180^\circ]$ due to physical constraints imposed by the servo motor. If condition in Equation (3.16) is satisfied, target angle is replaced by its complement angle and direction

of the rotation of the vibration motor is reversed.

$$\alpha_{diff} = \alpha_{MC} - \alpha_{target} \quad (3.14)$$

$$\mathbf{IF} \quad |\alpha_{diff}| > 180^\circ \quad (3.15)$$

$$\mathbf{THEN} \quad \alpha_{diff} = \alpha_{diff} - \text{sign}(\alpha_{diff})360^\circ$$

$$\mathbf{IF} \quad |\alpha_{diff}| > 90^\circ$$

$$\mathbf{THEN} \quad \begin{cases} \alpha_{target} = \alpha_{target} - \text{sign}(\alpha_{target})180^\circ \\ dir = -dir \end{cases} \quad (3.16)$$

In Equation (3.17), relative angle of the servo motor is calculated. Calculations may result in a numerical value that is outside the range $[0^\circ \ 180^\circ]$. Algorithms in Equation (3.18) and Equation (3.19) are used to contain the numerical value of the relative angle of the servo motor in the desired range.

$$\alpha_{servo}^{rel} = \alpha_{MC} + \alpha_{target} - 90^\circ \quad (3.17)$$

$$\mathbf{IF} \quad |\alpha_{servo}^{rel}| > 180^\circ \quad (3.18)$$

$$\mathbf{THEN} \quad \alpha_{servo}^{rel} = \alpha_{servo}^{rel} - \text{sign}(\alpha_{servo}^{rel})360^\circ$$

$$\mathbf{IF} \quad \alpha_{servo}^{rel} < 0^\circ \quad (3.19)$$

$$\mathbf{THEN} \quad \alpha_{servo}^{rel} = \alpha_{servo}^{rel} + 180^\circ$$

3.5 Single MechaCell Simulations in SimMechanics

Simulations are based on the SimMechanics setup discussed in Section 3.2. SimMechanics environment consists of a single MechaCell resting on a floor fenced with four walls.

Contact forces between the MechaCell and the floor as well as the contact forces between the MechaCell and the walls are modeled with circle–line force law that was discussed in Section 3.3.1.

Simulation goal is for the MechaCell to follow a prescribed path using the control

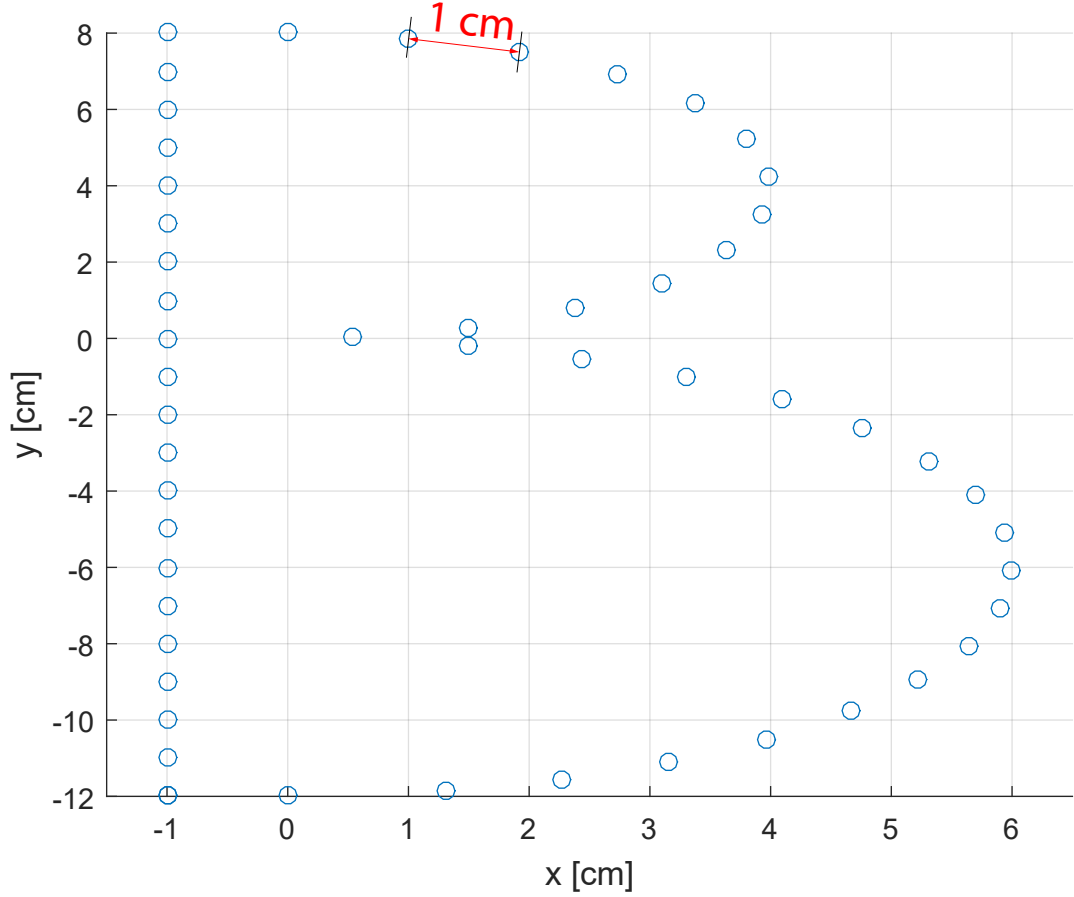


Figure 3.12: Prescribed path for single MechaCell simulations, B-shaped path

structure discussed in Section 3.4. The prescribed path is a complex path, i.e. a path that is constructed from straight line segments, curved line segments and sharp edges. Path satisfying all criteria for complexity is a B-shaped path shown in Figure (3.12). The path consists of 52 way-points that are placed 1 [cm] apart from each other.

Desired position (way-point) is changed every 1.5 [s] until the last way-point of the path corresponds with the starting point of the path, in this case the 53rd point of the B-shaped path.

Numerical values used in the simulation are presented in Table (3.2) and simulation sampling time is 1 [ms]. Coefficients of friction for Styrofoam with plastic are found in [43]. Contact stiffness and damping numerical values are found with error and trial until logical results are obtained. In Figure (3.13), results of the

Table 3.2: Numerical values of key parameters used in the SimMechanics simulation

Definition	Symbol	Numerical Values	Unit
MechaCell Mass	M	0.048	kg
Unbalance Mass	m	0.006	kg
Radius of the rod	r	0.004	m
Gravitational constant	g	9.81	m/s^2
Contact Forces			
Contact Stiffness	k	10^5	N/m
Contact Damping	b	500	Ns/m
Friction Forces			
Coefficient of static friction	μ_s	0.2	/
Coefficient of kinetic friction	μ_k	0.15	/
Threshold velocity	v_{th}	10^{-5}	m/s
Controller parameters			
Proportional	P	110	/
Integral	I	0.1	/
Derivative	D	0.05	/

simulation of the MechaCell following a B-shaped path are shown. Desired path, way-points, and MechaCell path are shown in the same figure. Snapshot of the SimMechanics environment while the MechaCell is following the B-shaped path is shown in Figure (3.14).

Effectiveness of path following of the MechaCell is demonstrated by introducing three numerical error parameters, x_{err} , y_{err} , ϵ . Desired position is updated every second and a half with the assumption that the MechaCell can cover the distance between two points in that time, hence error metrics x_{err} and y_{err} are introduced which represent the deviation of the MechaCell's position to the desired position at specific time. Third numerical error parameter is the deviation from path at any instance. This is the most important error parameter in path following.

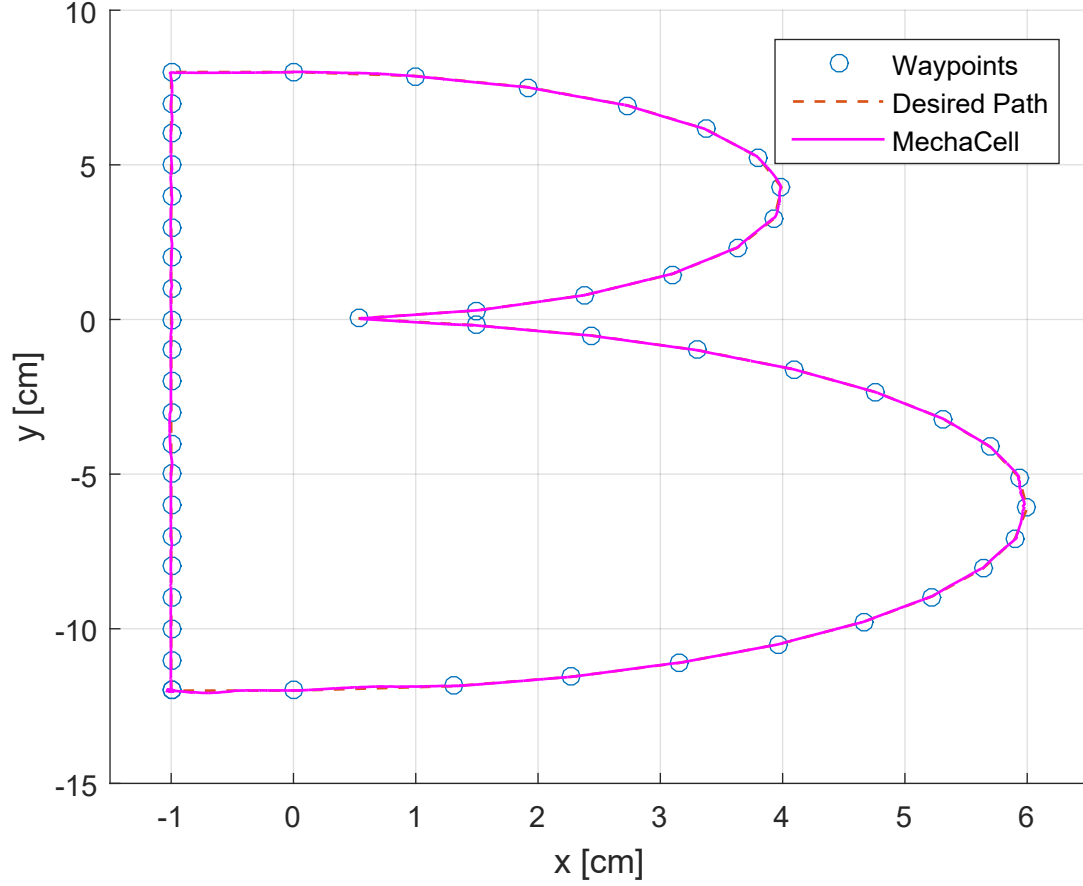


Figure 3.13: Simulation results of a MechaCell following a prescribed complex path

Between two way-points the MechaCell should travel a straight line that connects the two points. Graphical interpretation of the deviation from path numerical error parameter, ϵ , is shown in Figure (3.15).

Using Root Mean Square (RMS) averaging, error parameters for the simulation of single MechaCell following a complex path are: $\{x_{err} \ y_{err} \ \epsilon\} = \{0.03 \ 0.04 \ 0.01\}[cm]$. All error parameters have values smaller than a centimeter with deviation from path error parameter being the smallest, which, is a satisfying performance for path following task.

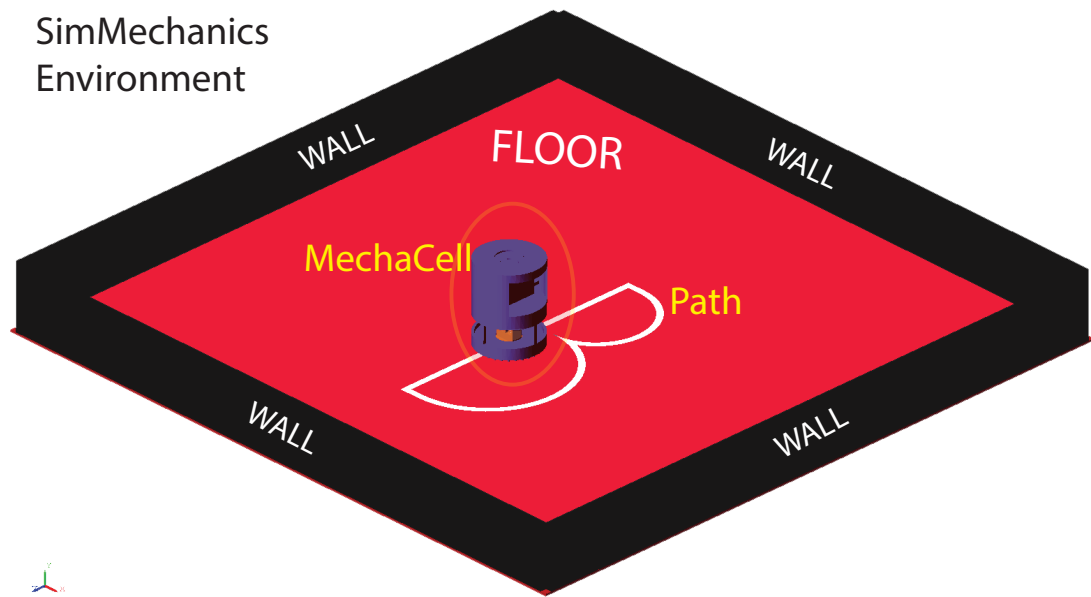


Figure 3.14: Snapshot of the SimMechanics environment: Single MechaCell following a path

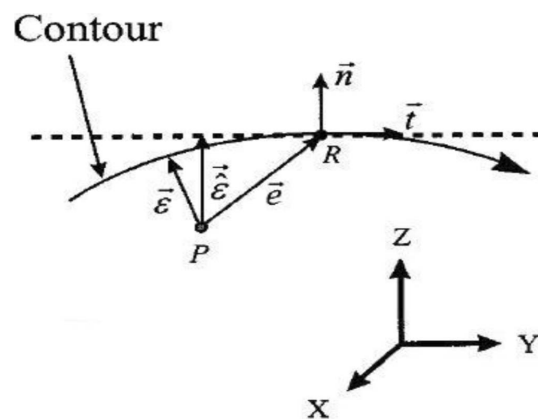


Figure 3.15: Graphical representation of the deviation from path numerical error parameter, ϵ

3.6 Cooperative Workpiece Manipulation Model in SimMechanics

Expanding on the single MechaCell model, the model in this section focuses on multiple MechaCells guiding a workpiece to follow a prescribed path, inter-MechaCell interactions, and MechaCell-workpiece interactions.

SimMechanics model that was presented in Section 3.5 is expanded such that the new expanded model contains four MechaCells and a workpiece. Before dwelling in simulation results, coordinated object manipulation strategies with multiple mobile robots are discussed.

3.6.1 Object Caging

Object caging strategy is commonly used in performing coordinated object manipulation tasks with multiple mobile robots such that there are no physical attachments to the object. Object caging definition and different types of object caging strategies such as force and form closure caging are given in [44].

Force closure caging demands all mobile robots participating in the object manipulation task to exert a force on the object's perimeter at all times; hence, forming some kind of gripping effect which traps the object. On the other hand, form closure caging does not require all mobile robots participating in the object manipulation task to exert force on the object's perimeter at all times; mobile robots that only aid the manipulation exert force on the object's perimeter. In this type of object closure, mobile robots are arranged in a way that they form a polygon which encloses the object. During the object manipulation task the form is unchanged. In Figure (3.16) two instances are presented, in part a, an object is surrounded by four mobile robots, but form closure caging condition is not realized, and in part b, the same four mobile robots are rearranged in a way that form closure caging condition is realized.

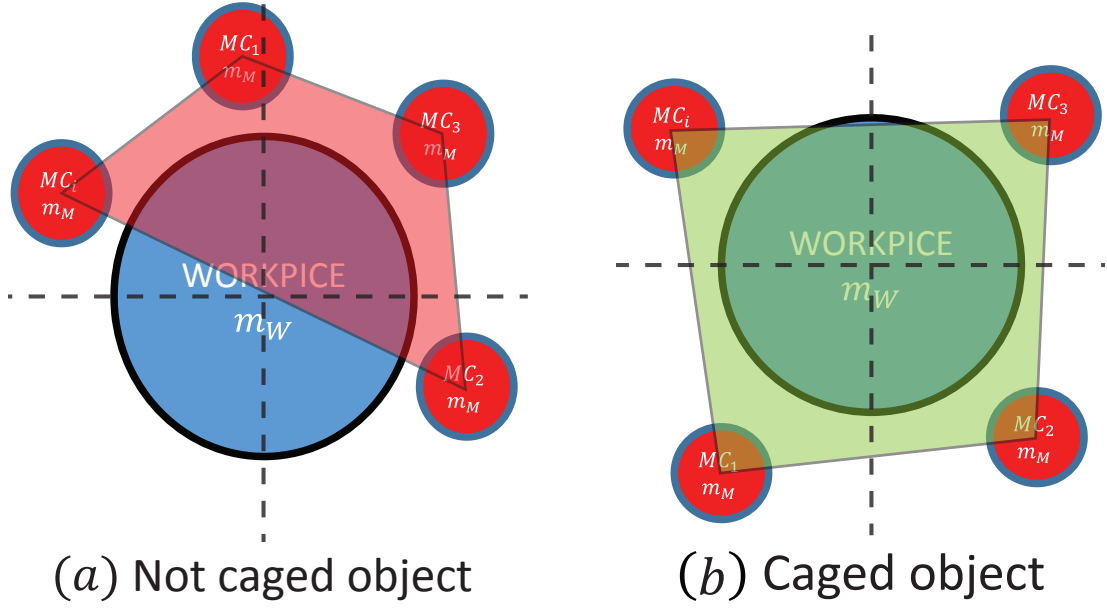


Figure 3.16: Form object closure caging example, object is not caged in part a, object is caged in part b

After rearrangement of the mobile robots, form closure caging condition has been reached Figure (3.16) part b, then the object manipulation task can be initiated.

Form closure caging condition can be realized with different number of mobile robots, minimum number however is three mobile robots since polygon cannot be formed with less than three mobile robots. In Figure (3.17) different number of disk shaped mobile robots are used to achieve form closure caging of a disk shaped object. This figure shows arrangement of three, four and five mobile robots around the disk shaped object such that form closure caging condition is realized.

Form closure caging condition: A disk shaped object is form closure caged with n mobile robots when every mobile robot remains inside its own caging region during the object manipulation task. Every mobile robot participating in the object manipulation task has its own caging region; hence, number of caging regions is equal to the number of mobile robots. Moreover, all caging regions are equal-sized. Caging regions are derived when the disk is divided into n equal sectors as shown in Figure (3.17) for 3, 4 and 5 mobile robots. In Equation (3.20)

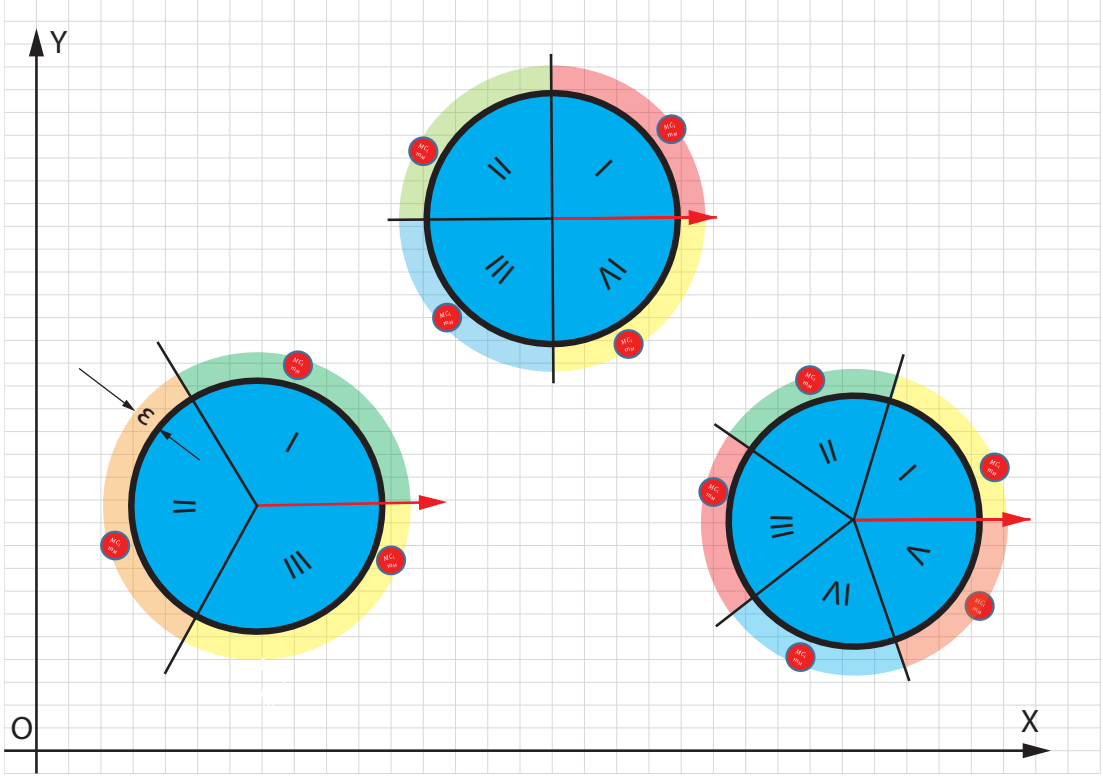


Figure 3.17: A disk shaped object is caged with form closure caging strategy by 3,4 and 5 mobile robots

caging regions for i^{th} mobile robot out of n mobile robots is defined.

$$r < |\mathbf{R}_W^i| < r + \epsilon \quad (3.20a)$$

$$\theta_1 < \theta^i < \theta_2 \quad (3.20b)$$

where, r is the radius of the disk shaped manipulated object, $\mathbf{R}_W^i$ is the position vector of the i^{th} mobile robot w.r.t. non-inertial coordinate system W with axis parallel to the inertial coordinate system O and origin at the geometric center of the disk. $\theta^i = \angle \mathbf{R}_W^i$ and ϵ is chosen such that distance between all neighboring mobile robots is smaller than the diameter of the disk, as shown in Equation (3.21). Coordinate systems and position vectors are shown in Figure (3.18).

$$|\mathbf{R}_O^i - \mathbf{R}_O^{i+1}| < 2r \quad (3.21)$$

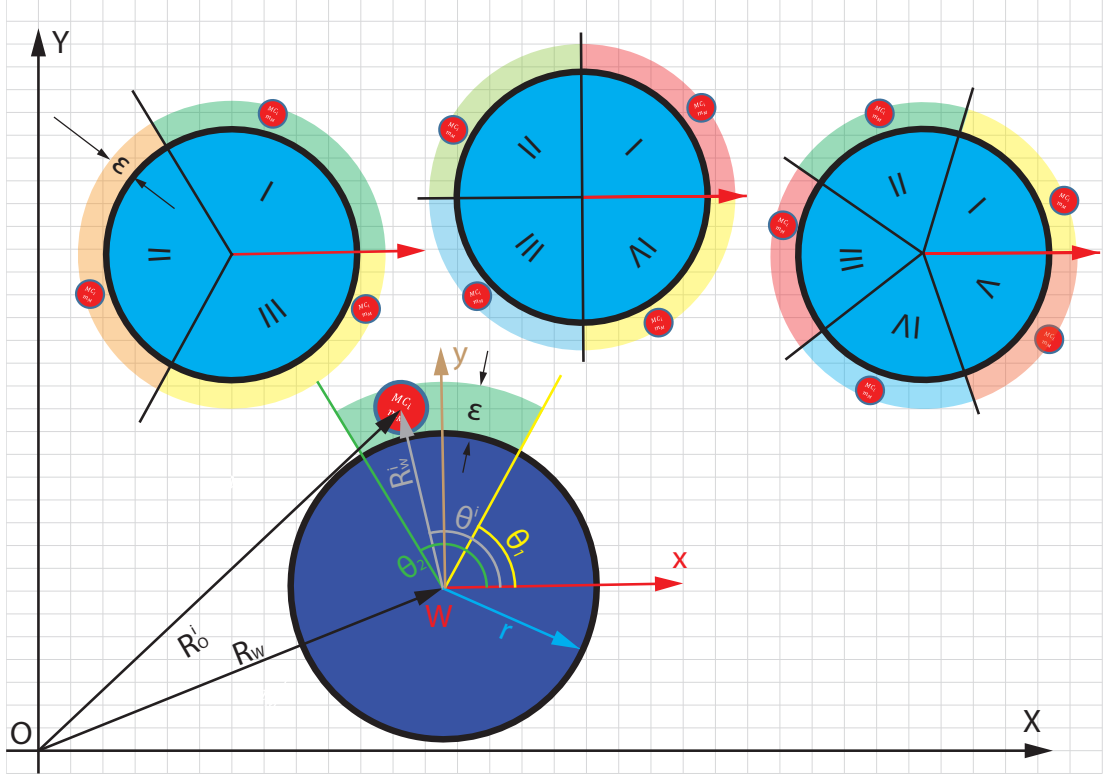


Figure 3.18: Form closure caging of an object by 3, 4 and 5 mobile robots and definitions of parameters used in Equation (3.20)

Sector division of the disk starts at the x -axis of the W coordinate system. Using this point and expressing vectors w.r.t. to the O coordinate system, Equation (3.20) for i^{th} mobile robot can be rewritten as Equation (3.22).

$$r < |\mathbf{R}_O^i - \mathbf{R}_W| < r + \epsilon \quad (3.22a)$$

$$(i-1)\frac{360^\circ}{n} < \angle(\mathbf{R}_O^i - \mathbf{R}_W) < i\frac{360^\circ}{n} \quad (3.22b)$$

where, $\mathbf{R}_O^i$ and $\mathbf{R}_W$ are position vectors of the i^{th} mobile robot and the disk shaped manipulated object w.r.t. O coordinate system respectively.

Form closure conditions above apply to objects of any shape. If the object does not have circular shape, then planar shape of the object is approximated with the smallest circle that circumscribes that shape. In Figure (3.19), objects with different shapes including objects with shape of a regular polygon, an irregular polygon, and with non-polygonal shape are presented. However, this approach

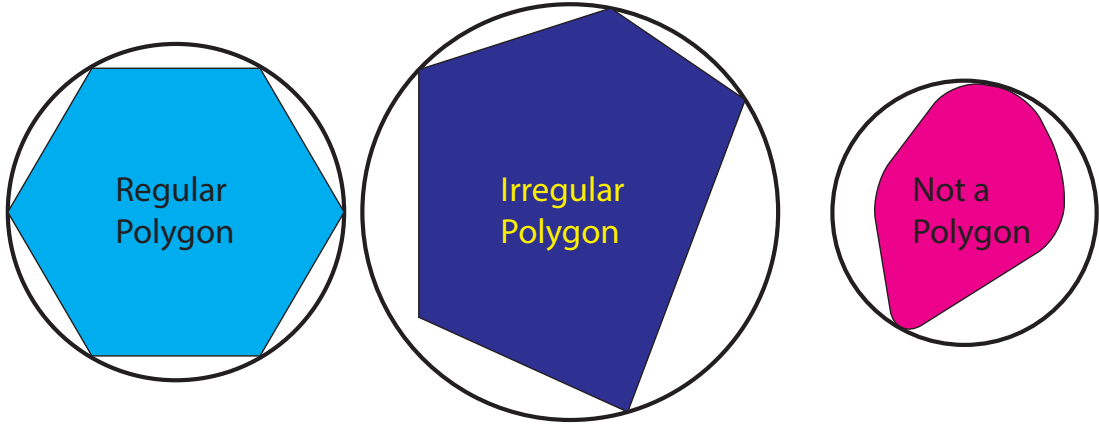


Figure 3.19: Different planar shapes: Regular polygon inscribed in a circle, Irregular polygon inscribed in the smallest circle, Non polygonal shape inscribed in the smallest circle

has limitations; a rectangle whose width is much larger than its height will not be caged successfully.

3.7 Simulations of Cooperative Workpiece Manipulation in SimMechanics Environment

Interactions between a MechaCell and the floor and walls is modeled identically as in Section 3.5, however MechaCell–MechaCell and MechaCell–workpiece interactions are modeled with circle–circle force law that was discussed in Section 3.3.1.

Goal of the simulations is to guide the workpiece to follow a prescribed path that satisfies complexity criteria described in Section 3.5, i.e. path should be constructed of straight line segments, curved line segments and contain sharp edges. In the simulations, a B-shaped path is replaced by a simpler one: deformed rotated D-shaped path. Since, this is more complicated setup a simpler path has been chosen. In Figure (3.20), the D-shaped path with a total of 30 way-points is shown. Similar to the way-points distribution in the B-shaped path, gap between any two neighboring way-points of the D-shaped path is 1 [cm].

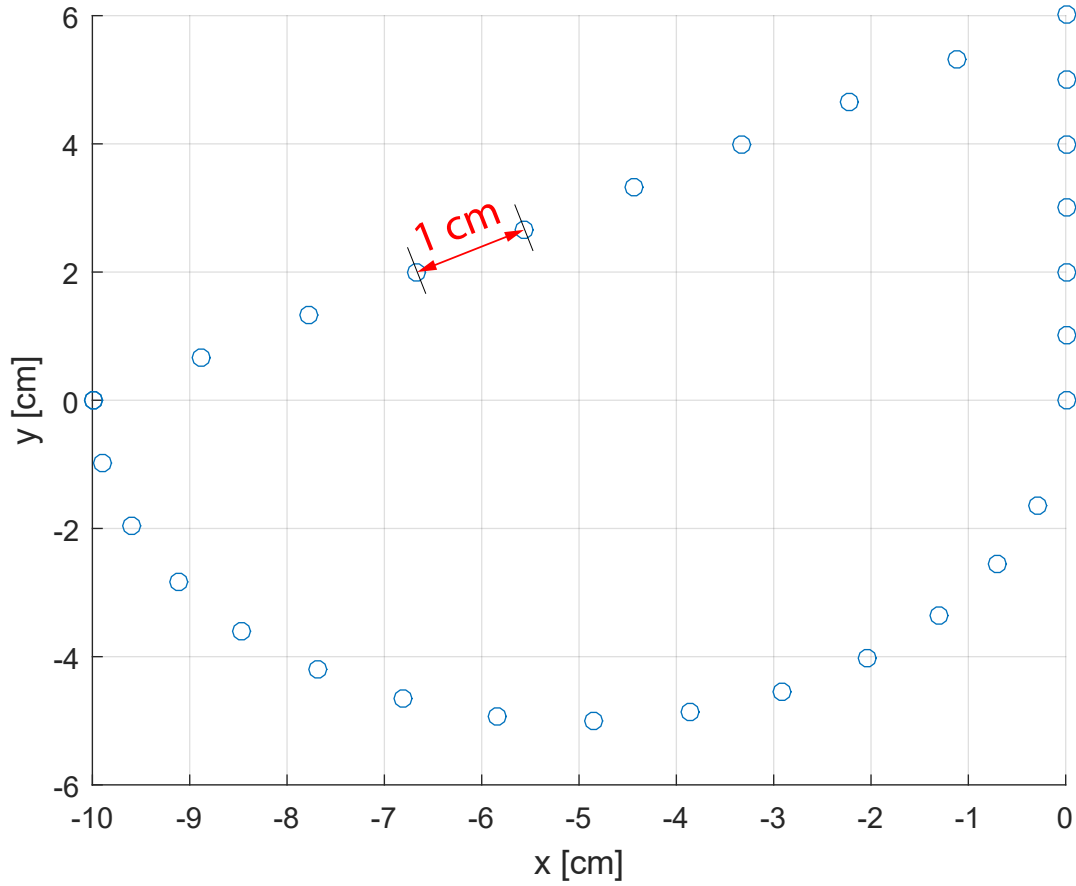


Figure 3.20: Deformed rotated D-shape path that is used in the workpiece manipulation simulations

Snapshot of the SimMechanics environment of this model where four MechaCell and a workpiece rest on a floor, and MechaCells are arranged around the workpiece such that form closure caging condition is realized is shown in Figure (3.21).

Four MechaCells are included in the workpiece guidance task to follow a prescribed path. Following the form closure caging algorithm presented in Section 3.6.1, the disk shaped workpiece is divided into four sectors each sector corresponding to a MechaCell. Moreover, MechaCells are evenly distributed around the workpiece. Diameter of the workpiece is $D = 12$ [cm] and its height is $H = 2$ [cm], radius of the cylindrical shape of the MechaCell is 2.5 [cm] as discussed in Chapter 2. The position vectors of each MechaCell w.r.t. the O

SimMechanics Environment

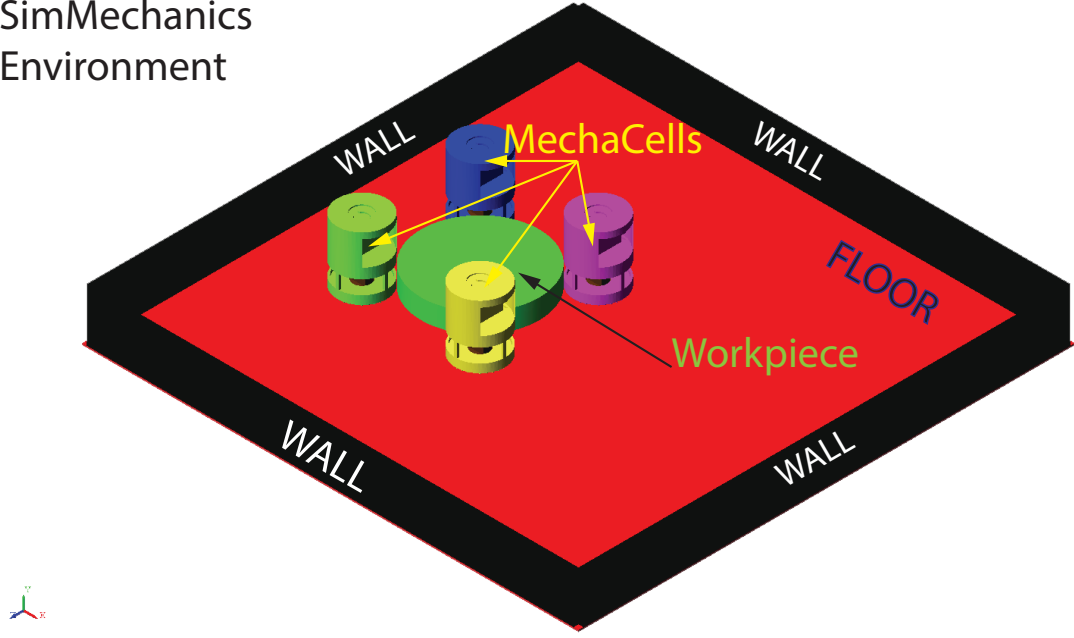


Figure 3.21: Snapshot of SimMechanics environment where four MechaCells and a workpiece rest on a floor and MechaCell are distributed around the workpiece to achieve form closure caging condition

coordinate system are given in Equation (3.23), and the top view of Figure (3.21) on which inertial coordinate system O and non-inertial coordinate system W are shown is presented in Figure (3.22).

$$\mathbf{R}_O^{\text{MC1}} = \mathbf{R}_W + [-6.01 \ 6.01]'[cm] \quad (3.23a)$$

$$\mathbf{R}_O^{\text{MC2}} = \mathbf{R}_W + [6.01 \ -6.01]'[cm] \quad (3.23b)$$

$$\mathbf{R}_O^{\text{MC3}} = \mathbf{R}_W + [6.01 \ 6.01]'[cm] \quad (3.23c)$$

$$\mathbf{R}_O^{\text{MC4}} = \mathbf{R}_W + [-6.01 \ -6.01]'[cm] \quad (3.23d)$$

As mentioned above, the goal is to guide the workpiece to follow a desired prescribed path. Hence, similar to Equation (3.23) that ensures form closure caging condition, desired location for each MechaCell is prescribed using Equation (3.24).

$$\mathbf{R}_O^{\text{MC1,D}} = \mathbf{R}_W^D + [-6.01 \ 6.01]'[cm] \quad (3.24a)$$

$$\mathbf{R}_O^{\text{MC2,D}} = \mathbf{R}_W^D + [-6.01 \ 6.01]'[cm] \quad (3.24b)$$

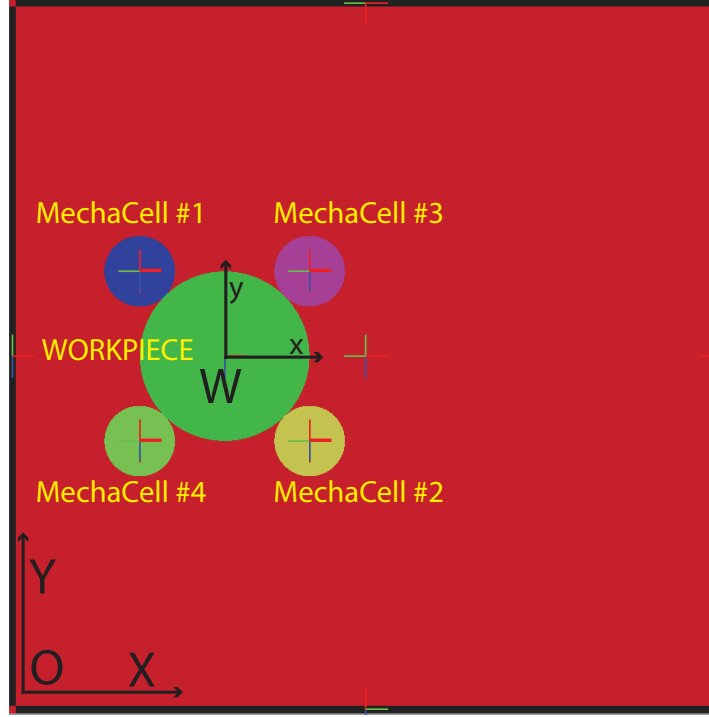


Figure 3.22: Top view of Figure (3.21) with visible coordinate systems

$$\mathbf{R}_O^{\text{MC3,D}} = \mathbf{R}_W^D + [-6.01 \ 6.01]'[cm] \quad (3.24c)$$

$$\mathbf{R}_O^{\text{MC4,D}} = \mathbf{R}_W^D + [-6.01 \ 6.01]'[cm] \quad (3.24d)$$

Desired position update (way-point update) was done every 1.5 [s] in Section 3.5. On the other hand, in this scenario, "chase the point" algorithm is executed to update the desired position of the MechaCells in order to ensure form closure caging.

"Chase the point" algorithm jumps to the next desired position (way-point) when the MechaCell comes close to the current desired position, i.e. if numerical value of the position error described in Section 3.4, Equation (3.10) drops below an assigned value. Moreover, new desired position for each MechaCell is assigned when numerical values of all MechaCells are below the threshold value. Mathematical description of the desired position update is given in Equation (3.25).

$$\begin{aligned} \text{IF } e_{MC1} \leq \tau \ \& \ e_{MC2} \leq \tau \ \& \ e_{MC3} \leq \tau \ \& \ e_{MC4} \leq \tau \\ \text{THEN } \mathbf{R}_O^{\text{MCi,Dj}} &\rightarrow \mathbf{R}_O^{\text{MCi,D(j+1)}} \end{aligned} \quad (3.25)$$

Where, $e_{MCi} = |\mathbf{R}_O^{\text{MCi,D}} - \mathbf{R}_O^{\text{MCi}}|$, i represents the i^{th} MechaCell, j represents the j^{th} way-point and τ is the threshold value whose numerical value is $\tau = 0.05 \text{ [cm]}$.

Key numerical values used in simulating the model described above where four MechaCells guide a workpiece to follow a prescribed path are listed in Table (3.3), and results are given in Figure (3.23).

Table 3.3: Numerical values of key parameters used to perform the workpiece manipulation simulations

Definition	Symbol	Numerical Values	Unit
MechaCell Properties			
MechaCell Mass	M	0.048	kg
Unbalance Mass	m	0.006	kg
Radius of the rod	r	0.004	m
Workpiece Properties			
Mass	M_w	0.1	kg
Diameter	D	0.12	m
Height	H	0.02	m
Contact Forces (Floor–MechaCell, Walls–MechaCell, Floor–Workpiece)			
Contact Stiffness	k	10^5	N/m
Contact Damping	b	500	Ns/m
Contact Forces (MechaCell–MechaCell)			
Contact Stiffness	k	$10^{1.3}$	N/m
Contact Damping	b	10	Ns/m
Contact Forces (MechaCell–Workpiece)			
Contact Stiffness	k	4.5	N/m
Contact Damping	b	4.25	Ns/m
Friction Forces			
Coefficient of static friction	μ_s	0.2	/
Coefficient of kinetic friction	μ_k	0.15	/
Threshold velocity	v_{th}	10^{-5}	m/s
Controller parameters (all MechaCells)			
Proportional	P	110	/
Integral	I	0.1	/
Derivative	D	0.05	/

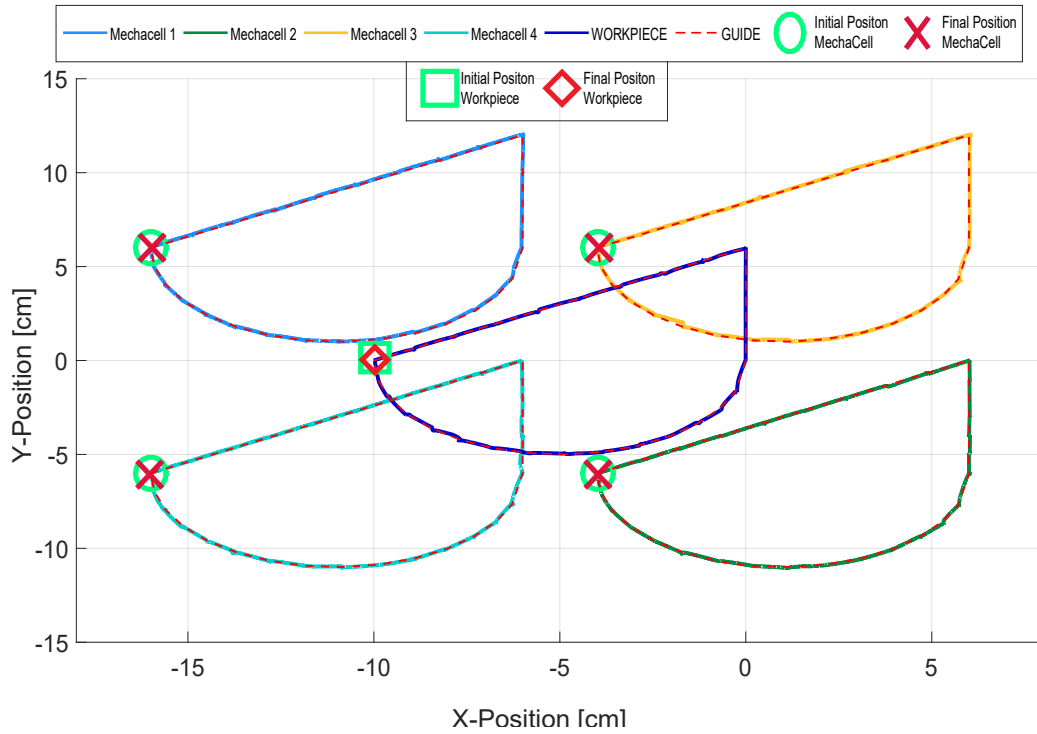


Figure 3.23: Workpiece path guiding task simulation results

Effectiveness of workpiece path guidance is expressed with the deviation from path error parameter that was defined in Section 3.5, ϵ . Rms of the numerical values of the error parameter for all four MechaCells and the workpiece are listed in Table (3.4).

Table 3.4: Numerical error parameters for four MechaCells and the workpiece handled by the MechaCells

	$\epsilon[cm]$
MechaCell 1	0.030
MechaCell 2	0.024
MechaCell 3	0.031
MechaCell 4	0.024
Workpiece	0.023

All numerical values of the error parameter for the MechaCells and the workpiece are less than 1 [cm]. Workpiece's deviation from path is less than a millimeter, which is a successfully completed path guiding task.

Chapter 4

Experimental Setup & Experiments

Mathematical model of the MechaCell was derived in Chapter 3. In the derivation of the mathematical model, the MechaCell was assumed to be point mass and the actuation force produced by the rotating unbalanced mass was assumed to be an external force acting on the point mass.

Chapter 3 carries forward with a discussion about the controller of the MechaCell. The controller consists of two independent sub-controllers, a translational controller (t-controller) and a steering controller (s-controller). Translational speed of the MechaCell is controlled by the t-controller with regulation of the rotational speed, ω of the unbalanced mass. Steering is controlled by the s-controller, that is regulating the orientation of the steering-cylinder in the inertial coordinate system. To improve performance of the t-controller, a friction compensation algorithm is added to a linear controller. Friction force predictions by the friction observer are based on the developed mathematical model.

Next, in Chapter 3, a model of the MechaCell in SimMechanics environment is presented. SimMechanics software was used to build a more realistic model of the MechaCell and simulate the MechaCell perform different tasks. Simulated

tasks performed by a MechaCell of group of MechaCells included a complex path following task and guidance of a workpiece to follow a complex path.

Discussion in this chapter goes on by presenting the experimental setup constructed in the laboratory which consists of a PC, a platform, a power source and an overhead camera for MechaCell and workpiece position tracking. This is followed by discussing the position tracking algorithm based on color tracking algorithm.

Mathematical model developed in Section 3.1 in which the MechaCell is assumed to be a point mass is experimentally verified. Step responses obtained using different PID configurations are compared, and the best one is chosen. Effect of the addition of the friction compensation algorithm is experimentally verified.

Single MechaCell path following, and multi-MechaCell workpiece path guiding experiments are performed. Experimental setup is identical to the simulation setup of Section 3.5 and Section 3.6.

4.1 Experimental Setup

Experimental setup consists of six elements: a PC with Bluetooth capabilities, a platform, an overhead camera, a power-source, four MechaCells and a workpiece, shown in Figure (4.1).

PC component of the experimental setup runs the position tracking algorithm and MechaCell's controller in Simulink and sends necessary control signals to the MechaCells via Bluetooth.

Experiments are performed on a platform over which the overhead camera is placed for position tracking. The platform is 50×50 [cm] and its floor is made of Styrofoam.

Overhead HD web-cam sends live video stream to the position tracking algorithm

running in Simulink. The position tracking algorithm sends the position data of the MechaCells and the workpiece to the controller of the MechaCell running in Simulink.

MechaCells are powered with an external power-source of 10 [V]. To eliminate the influence of a tether on the MechaCell's motion a very thin wire (magnetic wire) is used as a power-cord.

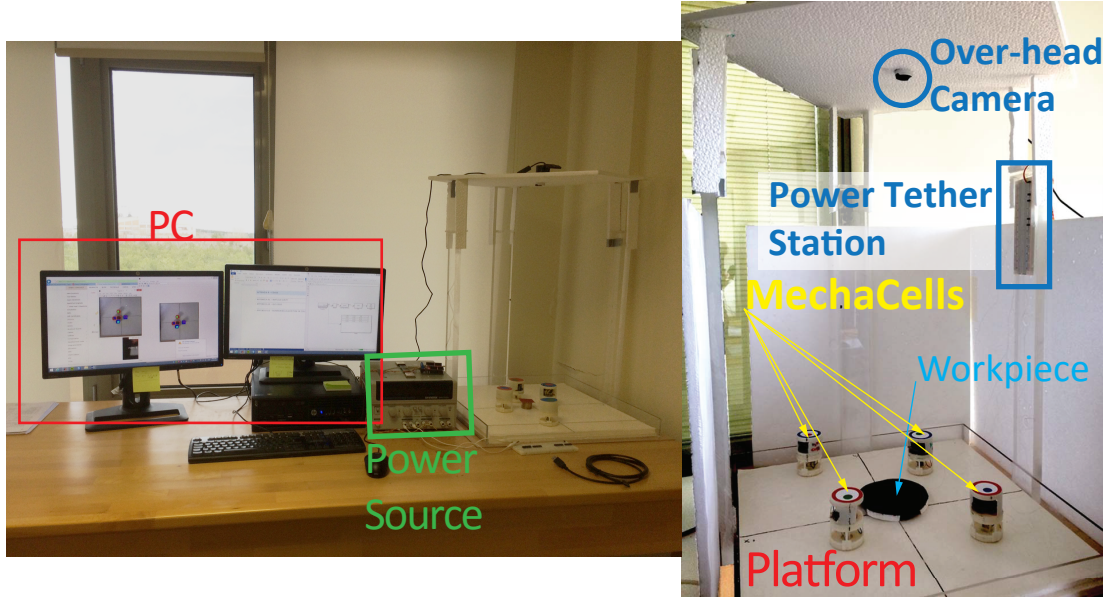


Figure 4.1: Experimental Setup

In the experiments, four MechaCells and a workpiece are used. The workpiece has cylindrical shape with diameter and height of $D = 12$ [cm], $H = 2$ [cm], and is made out of Styrofoam.

4.1.1 Position Tracking

Different methods exist to track an object in a video stream. A color-based object tracking algorithm is used in this work. To achieve tracking, top of each MechaCell is colored with a specific color.

Color tracking algorithm identifies pixels with matching colors, groups them into

regions and characterizes each region with centroid, bounding box, area, major and minor axis of a circumscribing ellipse and count. Centroid of a region in two dimensional space is defined as the arithmetic mean position of all points in the region. Bounding box of a two dimensional region is a rectangle whose vertices of its diagonal are defined in Equation (4.1).

$$v_1 = \{\min(C_x) \ \min(C_y)\} \quad (4.1a)$$

$$v_2 = \{\max(C_x) \ \max(C_y)\} \quad (4.1b)$$

where, C_x and C_y are sets of x and y coordinates of all points, $P = [x \ y]'$, that lie on the curve C defined by the perimeter of the region, respectively, is given in Equation (4.2). Examples of bounding boxes for different shapes are shown in Figure (4.2).

$$C_x = \{\forall P \in C \mid C_x = [1 \ 0]P\} \quad (4.2a)$$

$$C_y = \{\forall P \in C \mid C_y = [0 \ 1]P\} \quad (4.2b)$$

Number of regions with identical color is counted and represented with the count parameter.

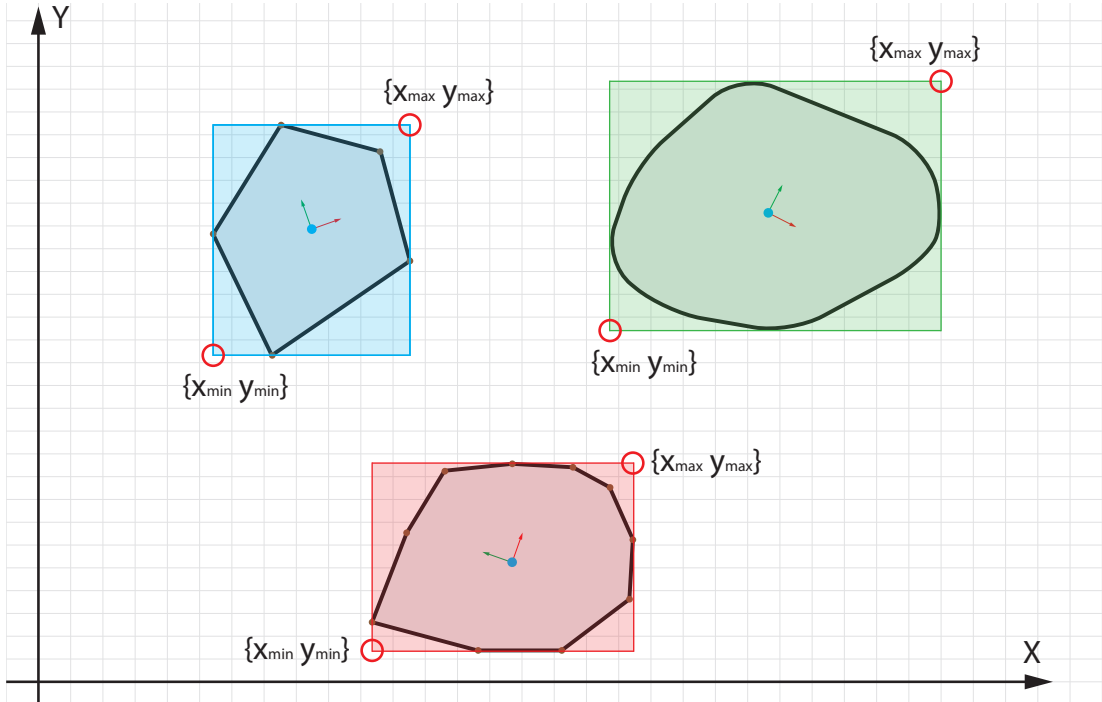


Figure 4.2: Examples of bounding boxes for different shapes

MechaCell's controller is constructed with a position feedback loop. Position and orientation data of each MechaCell is sent to the controller by the color tracking algorithm.

As previously mentioned, each MechaCell's top is colored with a specific color. Actually, each MechaCell's top is colored with two different colors, a background color and a foreground color. In Figure (4.3) coloring of the MechaCell #1 is shown. Two differently colored shapes are used in the coloring of the MechaCells.

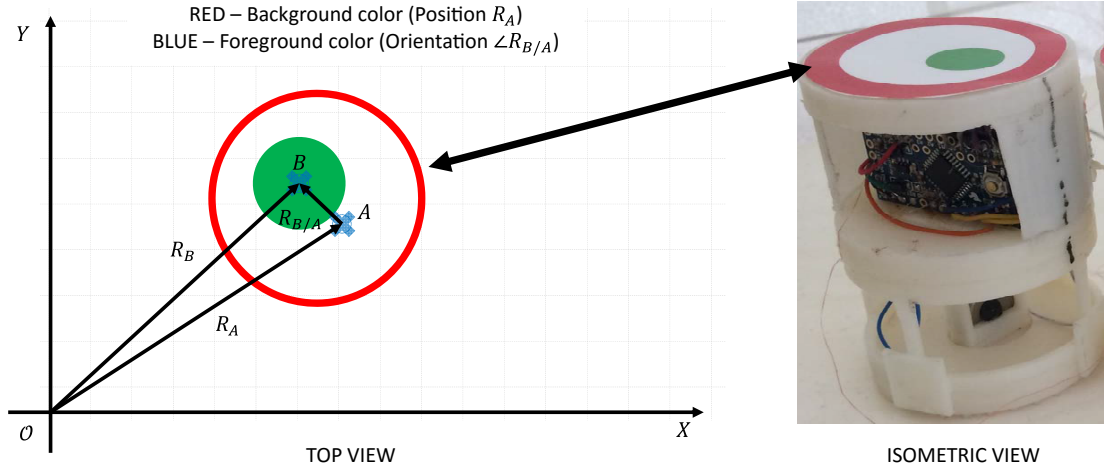


Figure 4.3: Color coding of MechaCells, with red background color and green foreground color

Background color fills a ring whose outer radius coincides with the MechaCell's rim. The foreground color fills a circle whose radius is one third of outer radius of the ring and whose center is at one fourth of the outer radius of the ring from the ring's center. Centroid of the background color is the MechaCell's position and the orientation of the vector starting at the centroid of the background color and ending at the centroid of the foreground color is the MechaCell's orientation. Based on Figure (4.3), vector $\mathbf{R}_A$ is position vector of the MechaCell, and orientation of vector $\mathbf{R}_{B/A}$ is the MechaCell's orientation and is shown in Equation (4.3).

$$\begin{aligned}\mathbf{R}_{B/A} &= \mathbf{R}_B - \mathbf{R}_A \\ \theta &= \angle(\mathbf{R}_{B/A})\end{aligned}\tag{4.3}$$

Simple color code is created to differentiate between MechaCells. In Table (4.1)

MechaCell identification corresponding to its color code is shown.

Table 4.1: MechaCell numbering according to the color codes

BACKGROUND COLORS	FOREGROUND COLORS			
		RED	GREEN	BLUE
	RED		MC 1	MC 2
	BLUE	MC 3	MC 4	
	BLACK	Workpiece		

As shown in Table (4.1) each MechaCell is colored with two colors out of three (Red, Green, Blue (RGB)) colors, however the workpiece is colored differently; its background color is black.

Identification algorithm is a sub routine of the position tracking algorithm and its task is to determine which color pair corresponds to which MechaCell/workpiece. Bounding box data from the color tracking algorithm is used to determine which color region is background color and which color region is foreground color. Using Table (4.1) as a reference, MechaCells/workpiece are identified. For region to be identified as a foreground region, its bounding box must be completely enclosed in another region's bounding box (background region). In Figure (4.4), a snapshot of a video preprocessed by the position tracking algorithm of four MechaCells arranged around a workpiece is shown. Identification algorithm is provided in Appendix B.1.

4.1.2 Camera Calibration

HD web-camera is used for video acquisition by the position tracking algorithm. Resolution of the camera is set to 360×360 [px] with 30.000 [fps] frame rate and the camera is positioned such that center of the platform is at 180×180 [px]. To convert pixels to centimeters a 10×1 [cm] green rectangle is used. This rectangle is positioned such that when identified by the color tracking algorithm, vertices of its bounding box coincide with its own vertices. Thus, comparing the length of the diagonal of the bounding box expressed in pixels to the

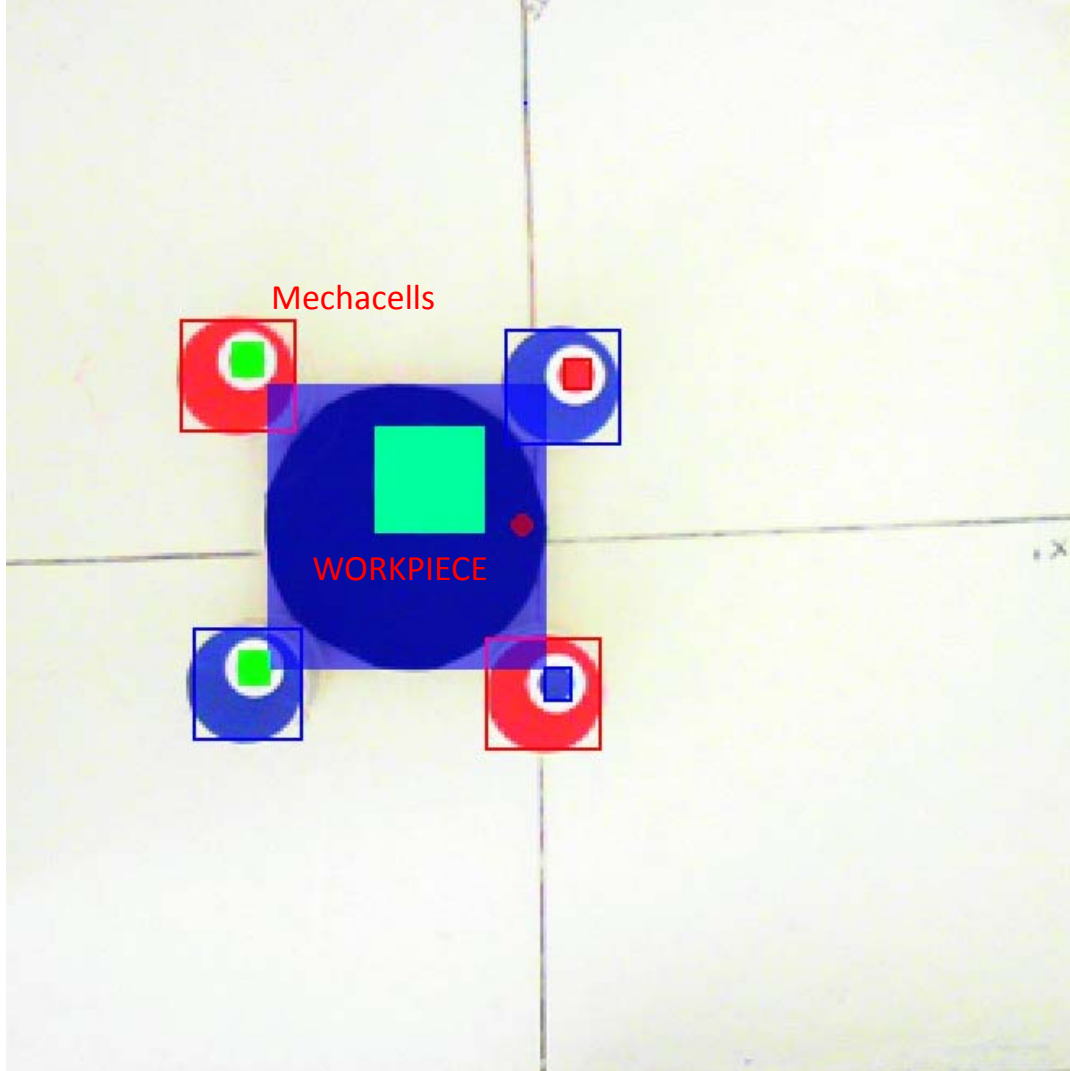


Figure 4.4: Snapshot of the color tracking algorithm

length of the diagonal of the rectangle expressed in centimeters, pixel–centimeter is derived as $1 [px] = 4/21 [cm]$.

4.2 Experiments

The Bluetooth enabled PC runs two separate sessions of Simulink (each on separate Matlab session): position tracking Simulink session and controller Simulink session. Video is streamed to the position tracking Simulink session. Position

tracking algorithm runs in this Simulink session and position data is sent to the controller Simulink session using User Datagram Protocol (UDP). Controller Simulink session contains the desired position and compares it to the current position received from position tracking Simulink session to calculate the error signal. Control signals are determined by the controller based on the value of the error signal and are transmitted over Bluetooth to the MechaCell. MechaCell's reaction to the received signals is captured by the overhead camera thus completing the loop. In Figure (4.5), graphical interpretation of the work flow between components of the experimental setup during the experiments is shown.

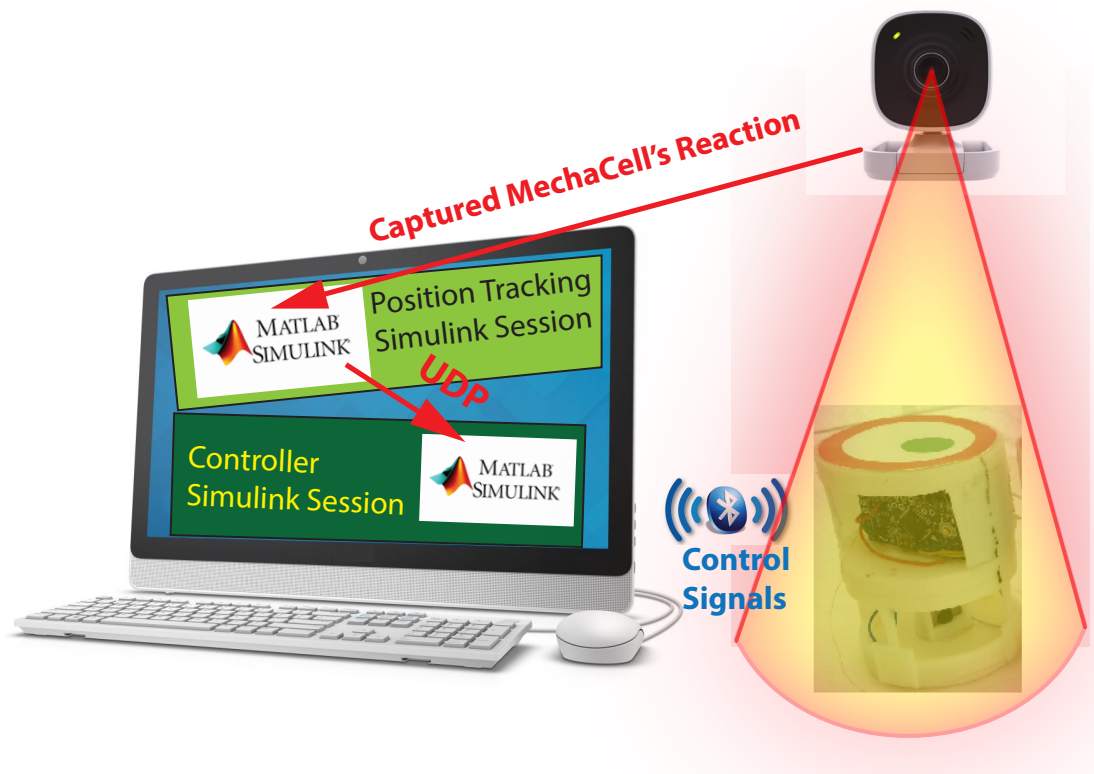


Figure 4.5: Graphical interpretation of the work flow between components of the experimental setup during experiments

4.2.1 Mathematical Model Verification

In Section 3.1, MechaCell was considered as a point mass and mathematical model was developed based on that consideration. Friction observer algorithm of the t-controller uses this mathematical model to predict value of the friction force. Therefore target in this experiment is to validate the mathematical model.

Scene setting for the model validation is as follows:

- **Mathematical Model:** Steering-cylinder is fixed at an arbitrary angle, rotational speed, ω , of the rotating unbalance mass is varied from zero to a specific value, then direction of rotation is changed, and ω is varied from the specific value back to zero. Variation of the rotational speed of the unbalance mass, ω , corresponds to PWM signal variation from zero to hundred and back to zero. MechaCell's linear displacement is determined by solving the equation of motion of the point mass.
- **Experiment:** MechaCell is left at a random position on the platform, servo motor's shaft is fixed at an arbitrary position and speed of the vibration motor is varied from zero to a specific value, then direction of rotation is changed, and speed of the vibration motor is varied from the specific value back to zero. Variation of the rotational speed of the vibration motor corresponds to PWM signal variation from zero to hundred and back to zero. MechaCell's change in linear position is tracked and recorded then normalized to start from zero.

In Figure (4.6) results of the validation of the mathematical model are shown and in Table (4.2) numerical values of key parameters used in the mathematical model are listed.

MechaCell's linear displacement in both the mathematical model and the experiment is zero, since with the reversal of rotation of the unbalance mass the MechaCell is expected to change direction of translation by 180° and return to

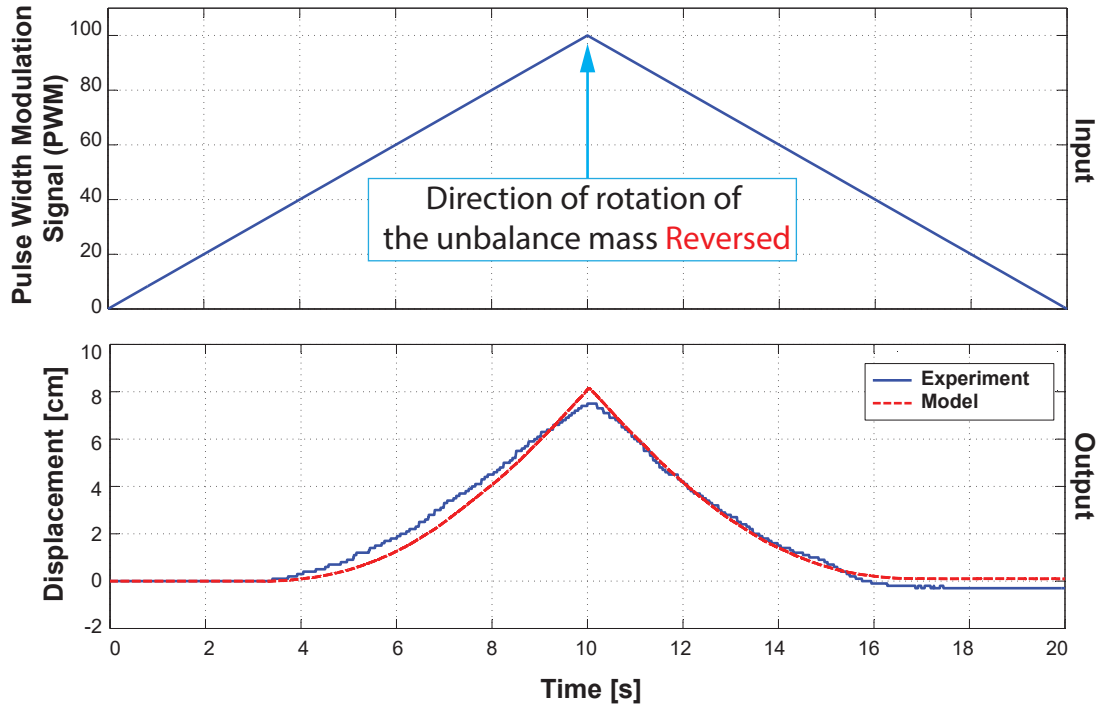


Figure 4.6: Mathematical model with point mass assumption validation

its starting point. Thus mathematical model based on the assumption of the MechaCell as a point mass is validated.

4.2.2 Linear controller (PID) tuning

MechaCell's controller contains two independent sub-controllers the t-controller and the s-controller, the t-controller is composed of two sub-systems a PID-controller and a friction compensation algorithm. PID-controller tuning process is carried out after removing all other controller components. Position feedback control structure with a PID-controller is shown in Figure (4.7).

Setting of the experiment: servo motor's shaft is fixed at an arbitrary angular position and desired position is set to 1 [cm] from the starting point along the line that the steering-mechanism is pointed at.

Table 4.2: Parameters used in the mathematical model with numerical values and definitions

Definition	Symbol	Numerical Values	Unit
Mass of MechaCell	M	0.048	kg
Mass of Unbalance	m	0.006	kg
Radius of the rod	r	0.004	m
1 PWM to rad/s	$/$	6.57	$rad/PWM\ s$
Coefficient of static friction	μ_s	0.2	$/$
Coefficient of kinetic friction	μ_k	0.15	$/$
Gravitational constant	g	9.81	m/s^2

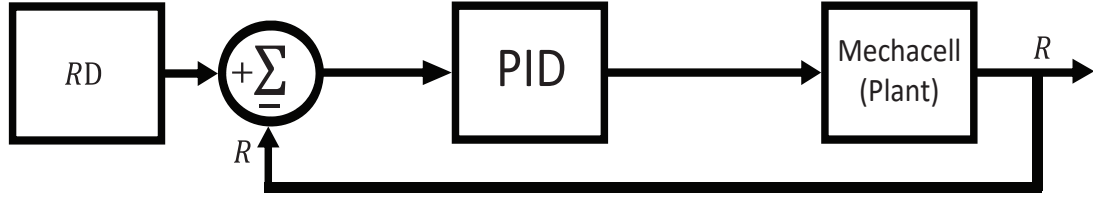


Figure 4.7: Feedback control structure of the MechaCell with only PID controller

Experimental results for 12 different parameter configurations of the PID-controller are shown in Figure (4.8).

Proportional and integral parameters of the PID-controller are changed such that proportional parameter changes with columns from 60 to 90 with increments of 10 and integral parameter changes with rows from 0.05 to 0.15 with increments of 0.05, value of the derivative parameter is 0.1 and is kept constant. Rows and columns mentioned here refer to Figure (4.8).

Fourth response of the results, presented in Figure (4.8), is considered to be the best response among the responses shown in Figure (4.8). PID-controller configuration of the fourth response is $PID = [90\ 0.05\ 0.1]$. Fast response, no overshoot and no steady state error were the criteria to why the fourth response was selected over the rest of the responses presented in Figure (4.8). In Figure (4.9), enlarged best response from Figure (4.8) is shown.

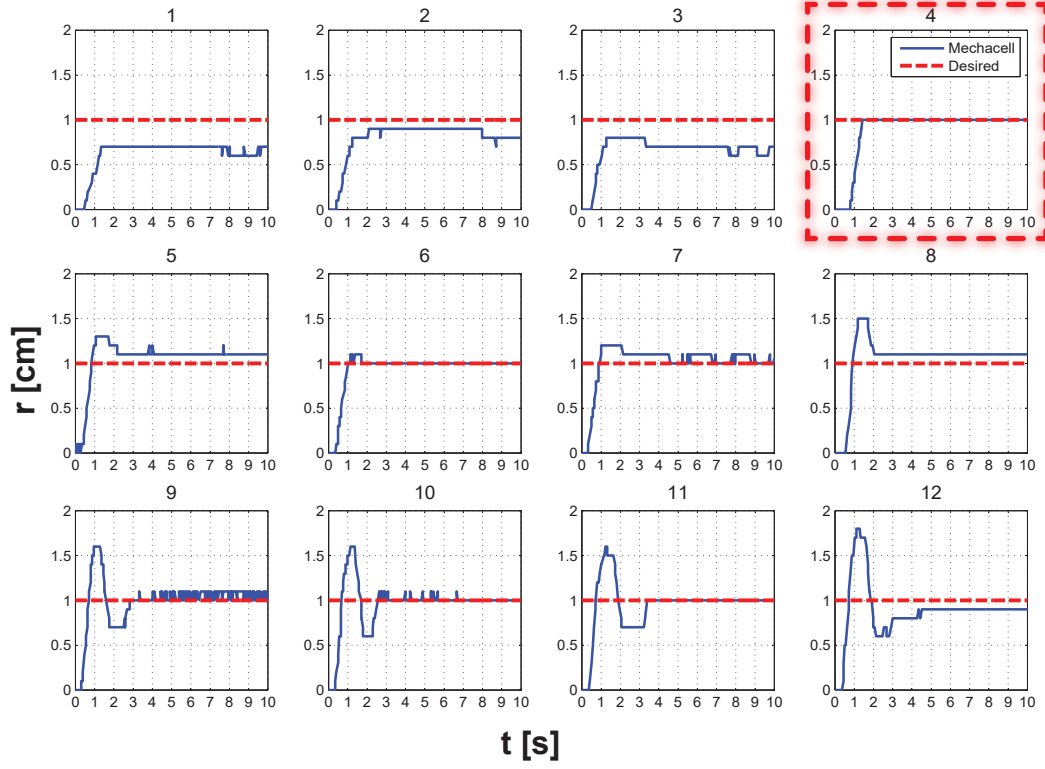


Figure 4.8: Step responses for different PID-controller configurations

Next part of this experiment is validate the improvement of MechaCell's t-controller when the friction compensation algorithm is added. As the name suggests, this algorithm should compensate for the energy lost to friction; thus, decrement of the delay witnessed in the response in Figure (4.9) is expected.

Experiment setting is identical with the previous part; servo motor's shaft angular orientation is fixed at an arbitrary angle (not necessarily the same with the previous part), desired position is 1 [cm] from the initial position in the direction to which the steering-mechanism is pointed. MechaCell's controller in this part is upgraded to full t-controller (PID-controller with friction compensation algorithm). S-controller is excluded from the control scheme.

In Figure (4.10) best response of Figure (4.8) is shown again, and in the same figure second response is shown obtained using the same PID-controller configuration with the addition of friction compensation algorithm. As expected addition of friction compensation algorithm contributed to faster response, and

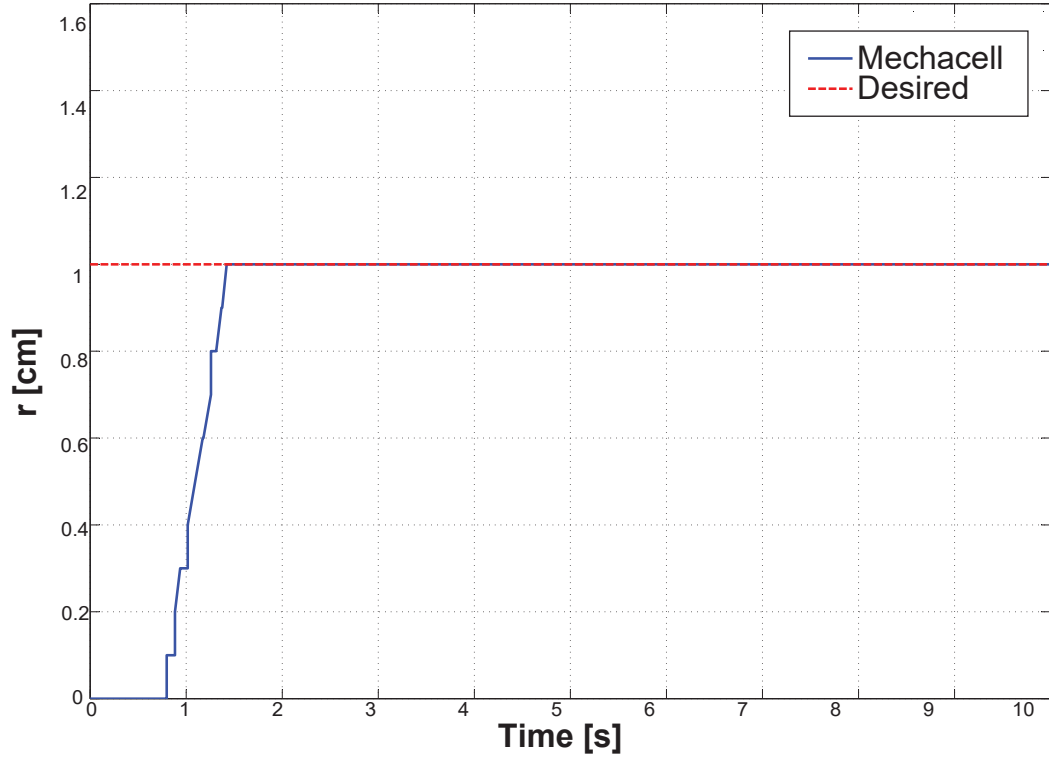


Figure 4.9: Best step response from Figure (4.8)

lesser delay.

4.2.3 Single MechaCell Path Following–Experiments

MechaCell executing the complex path following task (which in Section 3.5 was simulated in SimMechanics environment) in reality is the intent of the experiment presented in this section.

Complex path utilized in the MechaCell’s path following experiments is identical to the complex path used in Section 3.5 for simulations in SimMechanics environment. The B-shaped path is again shown in Figure (4.11). The B-shaped path consists of 52 way-points that are placed 1 [cm] apart from each other.

MechaCell’s controller used in this experiment has the PID configuration obtained in the previous section. Way-point update is performed every one and a half

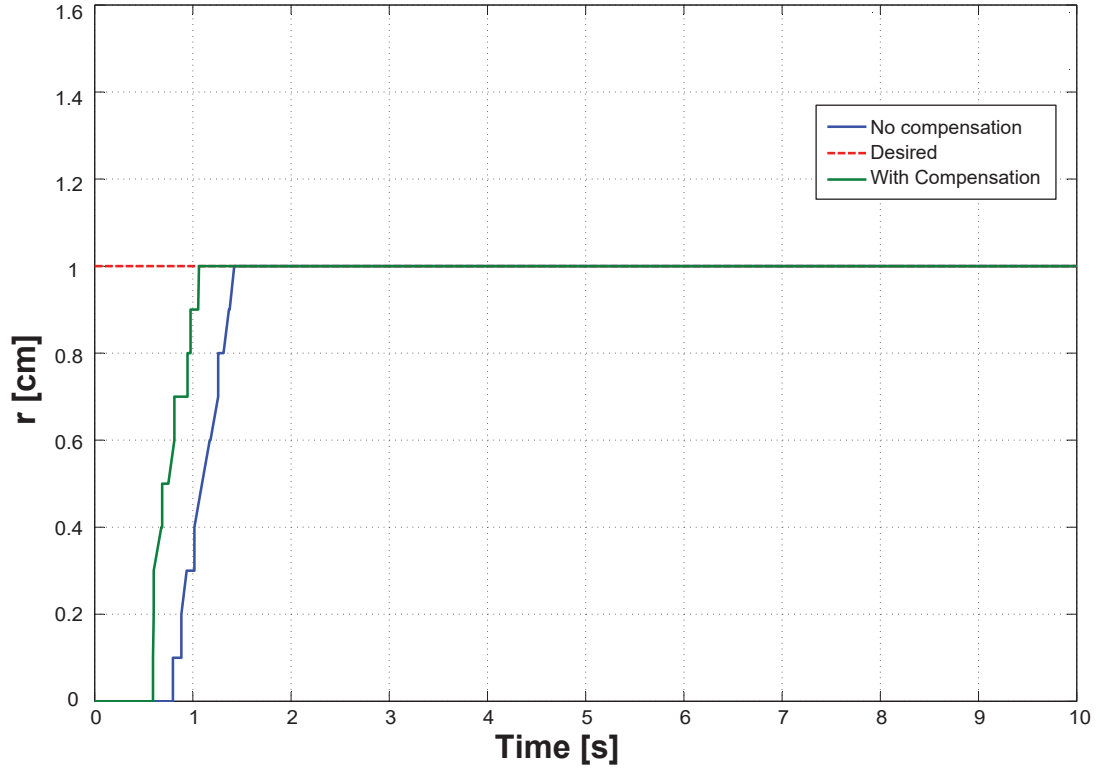


Figure 4.10: Best step response from Figure (4.8) and response improvement with after integration of friction compensation algorithm

second. Initial position of the MechaCell is random but close to the first way-point.

In Figure (4.12), results of the single MechaCell following a complex path are shown.

Performance of MechaCell's ability to follow a complex path is expressed using the same error parameters explained in Section 3.5, deviation from desired x -position, x_{err} , deviation from desired y -position, y_{err} , and deviation from path, ϵ . In Table (4.3) numerical values of the error parameters of both the experiments and the simulation from Section 3.5 are listed.

Every error parameter in Table (4.3) has value less than 1 [cm] these results conclude the task of complex path following as a success. MechaCell completed the path following tasks in 80 [s], with average speed of 0.6 [m/s].

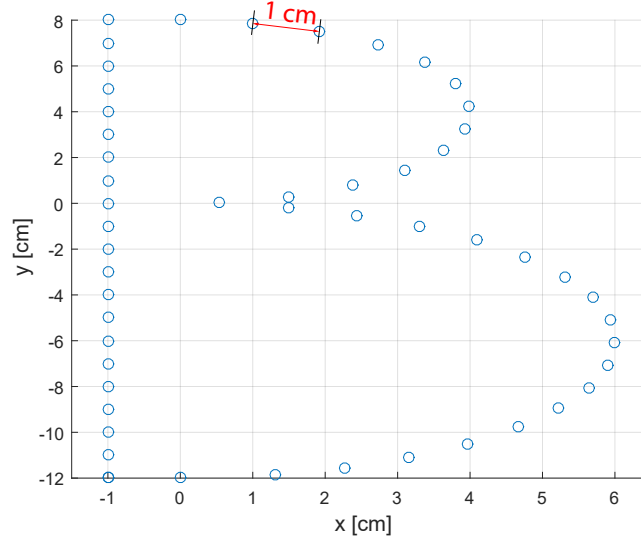


Figure 4.11: Complex path used in the path following experiments identical to the path in Section 3.5

Table 4.3: Numerical values of the error parameters from the experiment and the simulations

$x_{err}[cm]$	$y_{err}[cm]$	$\epsilon[cm]$
SimMechanics Simulation		
0.03	0.04	0.01
Experiment		
0.47	0.33	0.12

4.2.4 Cooperative Workpiece Manipulation–Experiments

Cooperative workpiece manipulation in plane is the task performed in the experiment discussed in this section. Experiment setting is identical to the setting of the simulation performed in SimMechanics for the same task in Section 3.7. Four MechaCell are arranged around a cylindrically shaped workpiece made of Styrofoam weighting $0.1 [kg]$ and $12 [cm]$ in diameter. Arrangement of the MechaCells around the workpiece assures form closure caging by the criteria discussed in Section 3.6.1. In Equation (3.23) arrangement of the MechaCells around the workpiece w.r.t. an inertial coordinate system was given, here same arrangement is used.

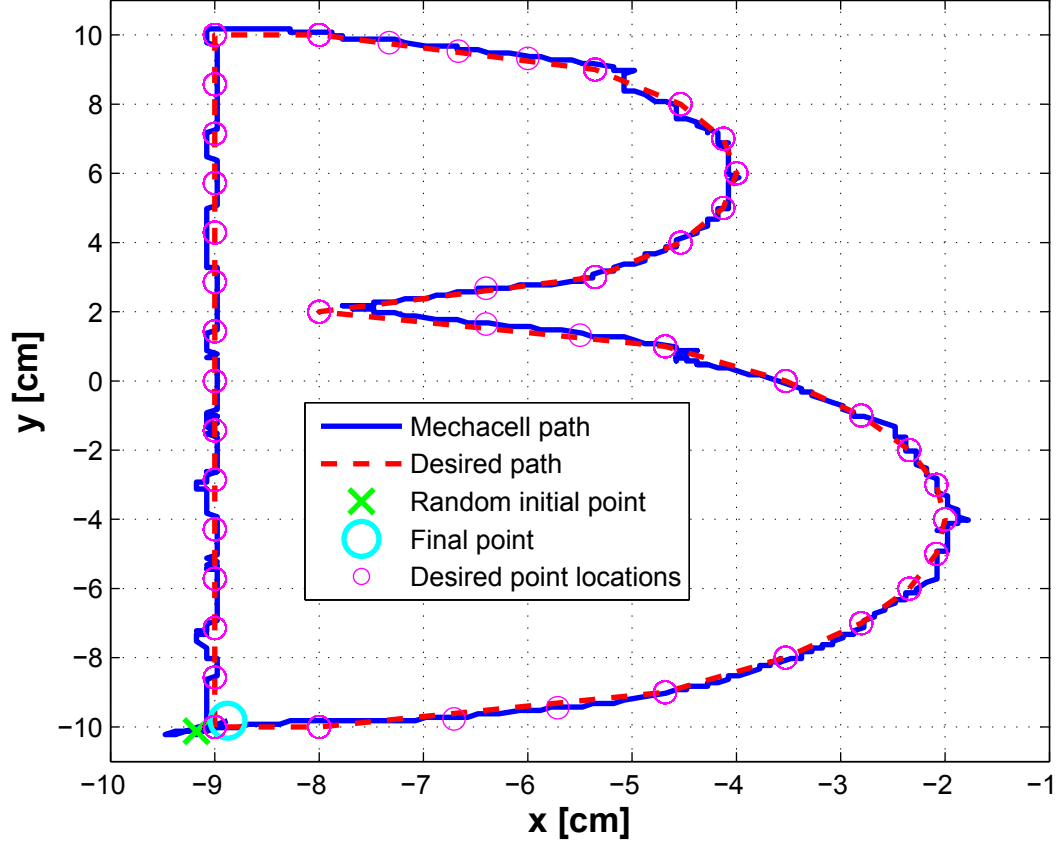


Figure 4.12: Results of the path following tasks of a single MechaCell

Aim of the experiment is to guide the workpiece to follow a complex path (path complexity criteria discussed in Section 3.5). Identical path is chosen to the one utilized in the SimMechanics simulations in Section 3.7, a deformed, rotated D-shape shown in Figure (4.13). The path consists of 30 way-points such that distance between any two neighboring way-points is 1 [cm].

Way-point update is executed using the "Chase the point" algorithm explained in Section 3.7.

In Figure (4.14), results of the workpiece guidance experiment by four MechaCells are shown. Path in the middle is the workpiece path.

Performance of the swarm of MechaCells guiding a workpiece to follow a complex

path is expressed using the deviation from path error parameters as in Section 3.7,

Numerical values of the error parameter for the experiment discussed in this section and the simulations done in SimMechanics in Section 3.7 are listed in Table (4.4).

Table 4.4: Numerical values of error parameters for the workpiece guiding tasks

	$\epsilon[cm]$	
	Sim	Exp
MechaCell 1	0.03	0.17
MechaCell 2	0.02	0.40
MechaCell 3	0.03	0.16
MechaCell 4	0.02	0.12
Workpiece	0.02	0.14

Deviation from path error parameter of the workpiece has value less than 1 [cm]. According to these results, workpiece path guiding task is successfully accomplished.

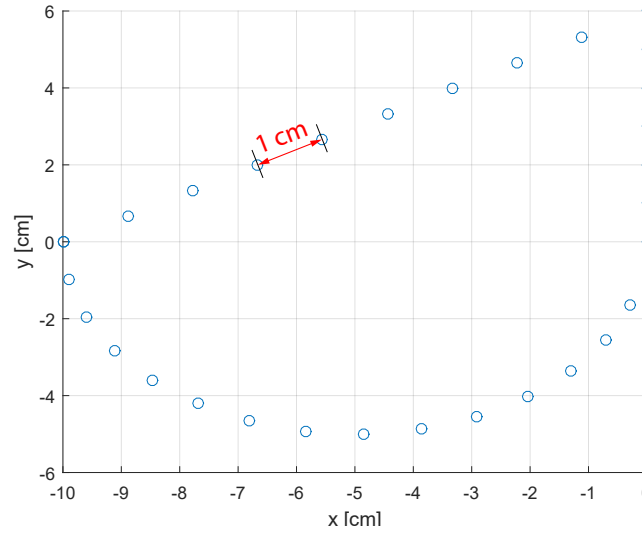


Figure 4.13: Complex path used in the path guidance experiments identical to the path in Section 3.7

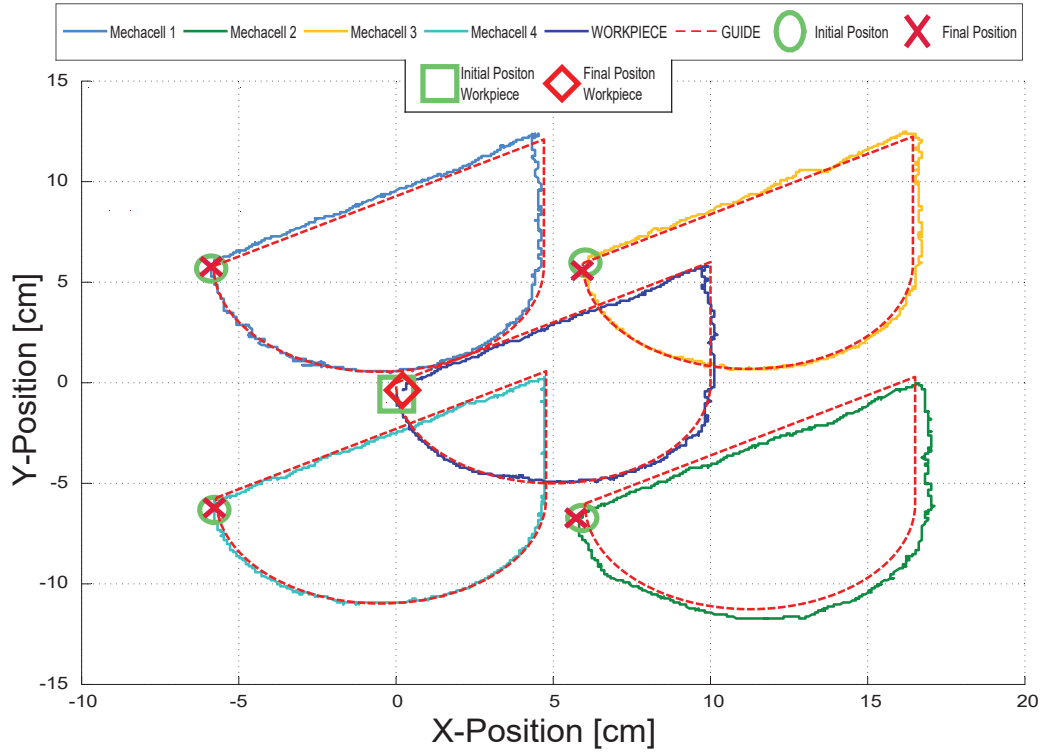


Figure 4.14: Workpiece path guidance experiment's results

Chapter 5

Conclusion & Future Work

5.1 Conclusion

Work presented in this thesis shows the design of a modular mechatronic device—The MechaCell. MechaCell is a simple module that on its own cannot accomplish much. But a pack of MechaCells are able to overcome complicated tasks such as workpiece manipulation in plane. Body of the MechaCell is a cylinder inscribed in a cube with side length $a = 5\text{ cm}$.

Novel locomotion system is integrated in the design of the MechaCell. Mechanism converting vibration motion to translational motion is implemented in the locomotion system. Our locomotion system offers an omni-directional drive. Omni-directionality differentiates our locomotion system in comparison with locomotion system using vibration motion in the literature.

A MechaCell controller is designed to perform path following and workpiece guiding tasks. Two sub-controllers are distinguished: a translational speed controller (t-controller) and a steering controller (s-controller). T-controller has a linear (PID) controller supported with a friction compensation algorithm and it regulates the translational speed of the MechaCell. Friction force magnitude is

estimated by a friction observer in the friction compensation algorithm using the developed mathematical model. S-controller regulates direction of translation.

Off-the-shelf electronics and actuators were used in the device's design. As the main processing unit Arduino ProMini board is utilized. Wireless communication to a PC is established using Bluetooth technology and a Bluetooth module is integrated. Servo motor for steering and vibrational motor for locomotion are actuators of the MechaCell. Vibrational motor is controlled by the Arduino board through a motor driver and servo motor is controlled directly through the Arduino board.

Path following experiments were carried out. Single MechaCell is shown following a complex path; complexity of a path is defined as consisting of straight and curved line segments and sharp turns. Path that is tracked by single MechaCell is chosen to be a B-shape constructed of 52 points. Path that is used in the multi-MechaCell experiments and simulations where a workpiece is guided, is a deformed and rotated D-shape.

MechaCell was able to successfully track a complex path, and four MechaCells, using form closure caging strategy, were able to guide a workpiece to track a complex path with deviation from path error parameter in sub-centimeter scale.

5.2 Future Work

Future work includes improvements of the MechaCell and implementation of sensors. Sensors that can be integrated include: accelerometers to estimate velocity and force and proximity sensors to measure contact force with the workpiece and detect close environment, respectively.

In addition, shrinking MechaCell's size is planned for the future. One pending design is presented in Figure (5.1). In this, design the servo motor is eliminated and two vibrational motors are used instead. A prototype was manufactured,

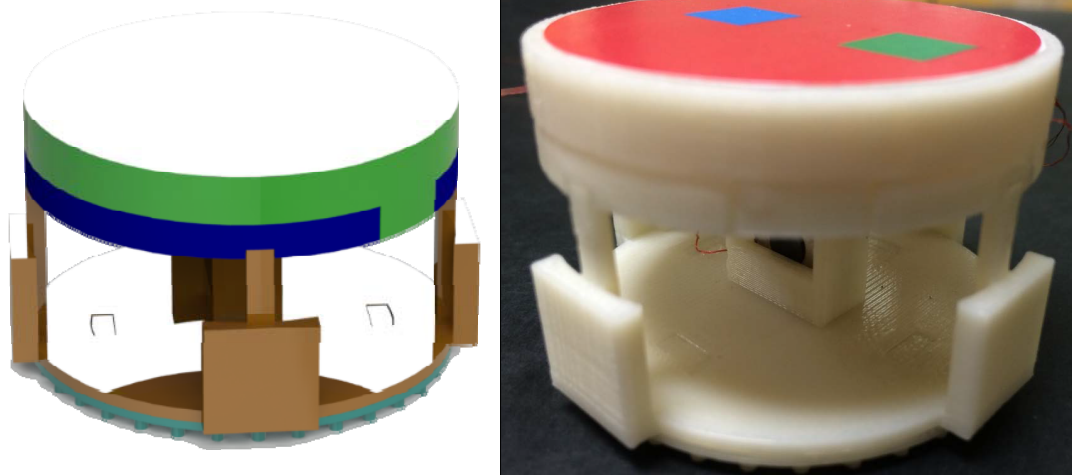


Figure 5.1: Future design of the MechaCell-mini MechaCell

extensive studies are to be conducted.

Friction has a big effect on the locomotion system of the MechaCell; hence, further studies on the shape and material of the bottom surface of the MechaCell are required.

Shrinking the MechaCell's size further is the next step, but that requires alternative actuators, which are smaller in size, since current vibration motor is the smallest one that can be purchased. Vibration motor can be replaced by two piezoelectric actuators such that one is positioned vertically and one is positioned horizontally.

Bibliography

- [1] *Oxford Dictionary of English*. Oxford University Press, 3 ed., 2010.
- [2] J. Needham, “Science and civilization in china. vol. 2 history of scientific thought,” 1956.
- [3] J. Burckhardt, O. Murray, and S. Stern, *The Greeks and Greek Civilization*. St. Martin’s Press, 1998.
- [4] E. G. Turner, “Ptolemaic egypt,” *The Cambridge Ancient History*, vol. 7, no. Part 1, 1984.
- [5] C. Huffman, *Archytas of Tarentum: Pythagorean, philosopher and mathematician king*. Cambridge University Press, 2005.
- [6] M. E. Moran, “The da vinci robot,” *Journal of endourology*, vol. 20, no. 12, pp. 986–990, 2006.
- [7] T. S. Ashton, *The industrial revolution: 1760-1830*, vol. 109. In the Hands of a Child, 1966.
- [8] W. A. Goddard III, D. Brenner, S. E. Lyshevski, and G. J. Iafrate, *Handbook of nanoscience, engineering, and technology*. CRC press, 2007.
- [9] A. M. Turing, “On computable numbers, with an application to the entscheidungsproblem,” *J. of Math*, vol. 58, no. 345-363, p. 5, 1936.

- [10] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "The intelligent asimo: System overview and integration," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 3, pp. 2478–2483, IEEE, 2002.
- [11] R. Playter, M. Buehler, and M. Raibert, "Bigdog," in *Defense and Security Symposium*, pp. 62302O–62302O, International Society for Optics and Photonics, 2006.
- [12] U. Saranli, M. Buehler, and D. E. Koditschek, "Rhex: A simple and highly mobile hexapod robot," *The International Journal of Robotics Research*, vol. 20, no. 7, pp. 616–631, 2001.
- [13] T. Sugihar, Y. Nakamura, and H. Inoue, "Real-time humanoid motion generation through zmp manipulation based on inverted pendulum control," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 2, pp. 1404–1409, IEEE, 2002.
- [14] 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 *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 2, pp. 1620–1626, IEEE, 2003.
- [15] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with non-linear dynamical systems in humanoid robots," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 2, pp. 1398–1403, IEEE, 2002.
- [16] H. Hirukawa, S. Hattori, K. Harada, S. Kajita, K. Kaneko, F. Kanehiro, K. Fujiwara, and M. Morisawa, "A universal stability criterion of the foot contact of legged robots-adios zmp," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 1976–1983, IEEE, 2006.
- [17] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.

- [18] A. F. Bourke, N. R. Franks, and L. Keller, *Social evolution in ants*. Princeton University Press Princeton, 1995.
- [19] C. D. Michener, *The social behavior of the bees: a comparative study*, vol. 73. Harvard University Press, 1974.
- [20] A. J. Sharkey, “Robots, insects and swarm intelligence,” *Artificial Intelligence Review*, vol. 26, no. 4, pp. 255–268, 2006.
- [21] A. J. Sharkey and N. Sharkey, “The application of swarm intelligence to collective robots,” *Advances in applied artificial intelligence*, p. 157, 2006.
- [22] T. Fong, I. Nourbakhsh, and K. Dautenhahn, “A survey of socially interactive robots,” *Robotics and autonomous systems*, vol. 42, no. 3, pp. 143–166, 2003.
- [23] G. Caprari, T. Estier, and R. Siegwart, “Fascination of down scaling the sugar cube robot,” *Journal of micromechatronics*, vol. 1, no. 3, pp. 177–189, 2001.
- [24] M. Rubenstein, C. Ahler, and R. Nagpal, “Kilobot: A low cost scalable robot system for collective behaviors,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 3293–3298, IEEE, 2012.
- [25] P. Raja and S. Pugazhenthir, “Optimal path planning of mobile robots: A review,” *International Journal of Physical Sciences*, vol. 7, no. 9, pp. 1314–1320, 2012.
- [26] C. A. Sierakowski and L. d. S. Coelho, “Study of two swarm intelligence techniques for path planning of mobile robots,” in *16th IFAC World Congress, Prague*, 2005.
- [27] A. Martinoli, K. Easton, and W. Agassounon, “Modeling swarm robotic systems: A case study in collaborative distributed manipulation,” *The International Journal of Robotics Research*, vol. 23, no. 4-5, pp. 415–436, 2004.
- [28] R. Gross and M. Dorigo, “Towards group transport by swarms of robots,” *International Journal of Bio-Inspired Computation*, vol. 1, no. 1-2, pp. 1–13, 2009.

- [29] S. Martel and M. Mohammadi, “Using a swarm of self-propelled natural microrobots in the form of flagellated bacteria to perform complex micro-assembly tasks,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 500–505, IEEE, 2010.
- [30] Y. Mohan and S. Ponnambalam, “An extensive review of research in swarm robotics,” in *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pp. 140–145, IEEE, 2009.
- [31] B. E. Bishop, “Swarm-based object manipulation using redundant manipulator analogs,” in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 1495–1500, IEEE, 2008.
- [32] A. T. Nguyen and S. Martel, “Locomotion of a miniature robot based on synchronized vibrating actuation mechanisms,” in *Advanced intelligent mechatronics, 2007 IEEE/ASME international conference on*, pp. 1–6, Sept 2007.
- [33] M. Aureli, V. Kopman, and M. Porfiri, “Free-locomotion of underwater vehicles actuated by ionic polymer metal composites,” *Mechatronics, IEEE/ASME Transactions on*, vol. 15, pp. 603–614, Aug 2010.
- [34] G. Cicconofri and A. DeSimone, “Motility of a model bristle-bot: A theoretical analysis,” *International Journal of Non-Linear Mechanics*, pp. 1–7, 2015.
- [35] Z. Bi and W. Zhang, “Flexible fixture design and automation: review, issues and future directions,” *International Journal of Production Research*, vol. 39, no. 13, pp. 2867–2894, 2001.
- [36] Q. Sayeed and E. D. Meter, “Machining fixture design and analysis software,” *THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH*, vol. 32, no. 7, pp. 1655–1674, 1994.
- [37] Z. Wang, Y. Hirata, and K. Kosuge, “An algorithm for testing object caging condition by multiple mobile robots,” in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 3022–3027, IEEE, 2005.

- [38] Z. Wang, Y. Hirata, and K. Kosuge, “Control a rigid caging formation for co-operative object transportation by multiple mobile robots,” in *Robotics and Automation, 2004. Proceedings. ICRA’04. 2004 IEEE International Conference on*, vol. 2, pp. 1580–1585, IEEE, 2004.
- [39] F. Vasca and L. Iannelli, *Dynamics and Control of Switched Electronic Systems*. Springer, 2012.
- [40] “Arduino board definition.” <http://www.arduino.cc/en/Main/ArduinoBoardProMini>. Accessed: July 2015.
- [41] G. Campa, “Arduino io package.” <http://www.mathworks.com/matlabcentral/fileexchange/32374-legacy-matlab-and-simulink-support-for-arduino>. Accessed: July 2015.
- [42] H. Olsson, K. Åström, C. Canudas de Wit, M. Gäfvert, and P. Lischinsky, “Friction Models and Friction Compensation,” *European Journal of Control*, vol. 4, pp. 176–195, January 1998.
- [43] M. Najm, “Experiment on friction among styrofoam, wood and carton with felt, plastic and cork.” <https://prezi.com/mus-bihg5n7x/experiment-on-friction-among-styrofoam-wood-and-carton-with-felt-plastic-and> 2012. Accessed: July 2015.
- [44] A. Sudsang and J. Ponce, “A new approach to motion planning for disc-shaped robots manipulating a polygonal object in the plane,” in *Robotics and Automation, 2000. Proceedings. ICRA ’00. IEEE International Conference on*, vol. 2, pp. 1068–1075 vol.2, 2000.

Appendix A

Electronic Components

A.1 Arduino ProMini Board

Arduino ProMini board's spec summary is presented in Table (A.1). Arduino ProMini board is powered and programmed with an FTDI cable.

Table A.1: Arduino ProMini board's spec summary

Microcontroller	ATmega328
Operating Voltage	3.3V or 5V (depending on model)
Input Voltage	3.35 -12 V (3.3V model) or 5 - 12 V (5V model)
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
Flash Memory	32 kB (of which 0.5 kB used by bootloader)
SRAM	2 kB
EEPROM	1 kB
Clock Speed	8 MHz (3.3V model) or 16 MHz (5V model)

In the MechaCell design, 3.3V version of the board is used.¹

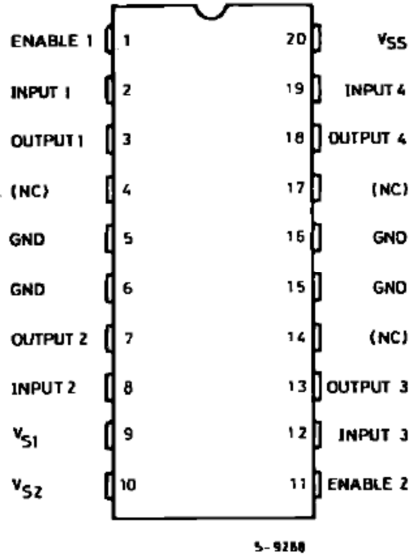


Figure A.1: L293C motor driver pin-out

A.2 Motor Driver

DESCRIPTION: *"The L293C is a monolithic high voltage, high current integrated circuit four channel driver in a 20 pin DIP. It is designed to accept standard TTL or DTL input logic levels and drive inductive loads (such as relays, solenoids, DC and stepping motors) and switching power transistors. The device may easily be used as a dual H-bridge driver: separate chip enable and high voltage power supply pins are provided for each H-bridge. In addition, a separate power supply is provided for the logic section of the device. The L293C is assembled in a 20 lead plastic package which has 4 center pins connected together and used for heat-sinking."* [Datasheet] In Figure (A.1), pin-out of the L293C motor driver is presented.

A.3 Bluetooth Module - RN42

Bluetooth RN42 module has been purchased from Sparkfun electronics, device description is given below.

DESCRIPTION: *"The RN42 is a small form factor, low power, highly economic Bluetooth radio for OEMs adding wireless capability to their products. The RN42 supports multiple interface protocols, is simple to design in and fully certified, making it a complete embedded Bluetooth solution. The RN 42 is functionally compatible with RN 41. With its high performance on chip antenna and support for Bluetooth Enhanced Data Rate (EDR), the RN42 delivers up to 3 Mbps data rate for distances to 20M. The RN-42 also comes in a package with no antenna (RN-42-N). Useful when the application requires an external antenna, the RN-42-N is shorter in length and has RF pads to route the antenna signal."*[Data-sheet]

Features:

- Fully qualified Bluetooth 2.1/2.0/1.2/1.1 module
- Bluetooth v2.0+EDR support
- Available with on board chip antenna (RN-42) and without antenna (RN-42-N)
- Postage stamp sized form factor, $13.4 \times 20 \times 2[mm]$ (RN-42-N)
- Low power (26uA sleep, 3mA connected, 30mA transmit)
- UART (SPP or HCI) and USB (HCI only) data connection interfaces.
- Sustained SPP data rates - 240Kbps (slave), 300Kbps (master)
- HCI data rates - 1.5Mbps sustained, 3.0Mbps
- Embedded Bluetooth stack profiles included (requires no host stack): GAP, SDP, RFCOMM and L2CAP protocols, with SPP and DUN profile support. burst in HCI mode
- Bluetooth SIG certified
- Castellated SMT pads for easy and reliable PCB mounting
- Certifications: FCC, ICS, CE

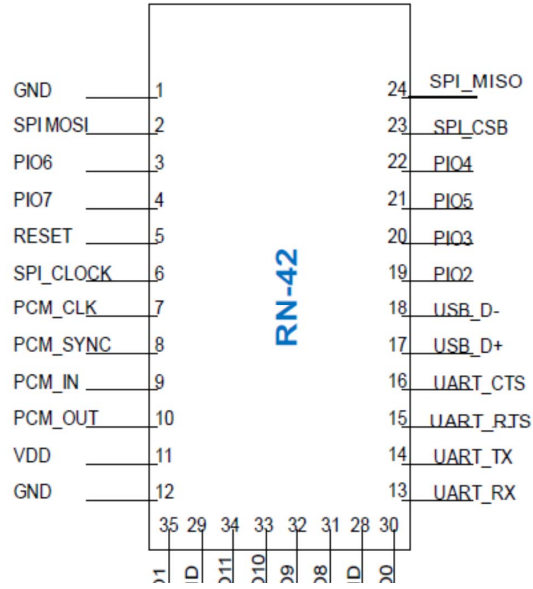


Figure A.2: RN42 Bluetooth module pin-out

- Environmentally friendly, RoHS compliant

Figure (A.2) shows the pin-out of the RN42 Bluetooth module.

A.4 Vibration Motor

Vibration motor is of a shaft-less type; basic specifications are presented in Table (A.2). In Figure (A.3), current versus rotational speed in RPM is presented.

Table A.2: Basic specifications of the shaftless vibrational motor

Specifications	
Size	10mm diameter, 2.0mm height
Weight	0.8g
Recommended operating voltage	2.53.5V
Vibration amplitude at 3V	0.75g
Speed at 3V	14500RPM (12000RPM min)
Free-run current at 3V	60mA (80mA max)
Startup current at 3V	120mA max
Motor resistance	$29 \pm 6\Omega$
Operating temperature	$-10^{\circ}C$ to $+60^{\circ}C$
Storage temperature	$-30^{\circ}C$ to $+70^{\circ}C$

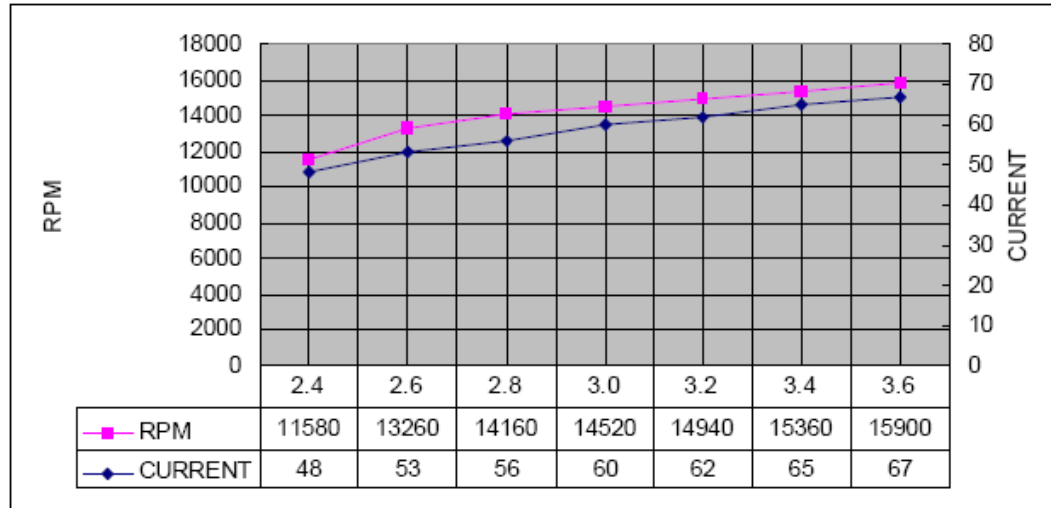


Figure A.3: Rotational speed with current variations of the shaft-less vibration motor



Figure A.4: TowerPro SG90 servo motor

A.5 Servo Motor

Servo motor used is TowerPro SG90, shown in Figure (A.4). Basic specifications are presented in Table (A.3).

Table A.3: TowerPro SG90 servo motor basic specifications

Basic Information	
Modulation:	Analog
Torque:	4.8V 25.0 oz-in ($1.80kg - cm$)
Speed:	4.8V 0.12sec/60°
Weight:	0.32 oz (9.0g)
Dimensions:	Length: 0.91 in (23.0mm)
	Width: 0.48 in (12.2mm)
	Height: 1.14 in (29.0mm)
Motor Type:	3-pole
Gear Type:	Plastic
Rotation/Support:	Bushing

Appendix B

Matlab-Code

B.1 Identification Algorithm

```
function PosMat = fcn(CenR, CenG, CenB, NumR, NumG, NumB, BboxR, BboxG, BboxB)
%#codegen
PosMat = zeros(4,4); % Position matrix [x,y,theta]
% Find Red Backgrounds
for i=1:NumR
    vecRup=BboxR(i,[1 2]);
    vecRup=vecRup';
    vecRdown=[BboxR(i,1)+BboxR(i,3);BboxR(i,2)+BboxR(i,4)];
    % Check if there are any green obj inside red
    for j=1:NumG;
        vecGu=BboxG(j,[1 2]);
        vecGu=vecGu';
        vecGd=[BboxG(j,1)+BboxG(j,3);BboxG(j,2)+BboxG(j,4)];
        logic1 = vecGu>vecRup;
        logic2 = vecGu<vecRdown;
        logic3 = vecGd>vecRup;
        logic4 = vecGd<vecRdown;
        logic = and(and(logic1,logic2),and(logic3,logic4));
        if and(logic(1),logic(2));
            ind = 1;
```

```

        diffVec = CenG(j,:)-CenR(i,:);
        diffVec = diffVec';
        theta = atan2d(diffVec(2),diffVec(1));
%         theta = single(theta);
        PosMat(ind,:) = [CenR(i,:) theta ind];
    end
end
% Check if there are any blue obj inside red
for j=1:NumB;
    vecBu=BboxB(j,[1 2]);
    vecBu=vecBu';
    vecBd=[BboxB(j,1)+BboxB(j,3);BboxB(j,2)+BboxB(j,4)];
    logic1 = vecBu>vecRup;
    logic2 = vecBu<vecRdown;
    logic3 = vecBd>vecRup;
    logic4 = vecBd<vecRdown;
    logic = and(and(logic1,logic2),and(logic3,logic4));
    if and(logic(1),logic(2));
        ind = 2;
        diffVec = CenB(j,:)-CenR(i,:);
        diffVec = diffVec';
        theta = atan2d(diffVec(2),diffVec(1));
%         theta = single(theta);
        PosMat(ind,:) = [CenR(i,:) theta ind];
    end
end
end
% Find Blue backgrounds
for i=1:NumB
    vecRup=BboxB(i,[1 2]);
    vecRup=vecRup';
    vecRdown=[BboxB(i,1)+BboxB(i,3);BboxB(i,2)+BboxB(i,4)];
    % Check if there are any green obj inside blue
    for j=1:NumG;
        vecGu=BboxG(j,[1 2]);
        vecGu=vecGu';
        vecGd=[BboxG(j,1)+BboxG(j,3);BboxG(j,2)+BboxG(j,4)];
        logic1 = vecGu>vecRup;
        logic2 = vecGu<vecRdown;
        logic3 = vecGd>vecRup;

```

```

        logic4 = vecGd<vecRdown;
        logic = and(and(logic1,logic2),and(logic3,logic4));
        if and(logic(1),logic(2));
            ind = 4;
            diffVec = CenG(j,:)-CenB(i,:);
            diffVec = diffVec';
            theta = atan2d(diffVec(2),diffVec(1));
%           theta = single(theta);
            PosMat(ind,:) = [CenB(i,:) theta ind];
        end
    end
% Check if there are any red obj inside blue
for j=1:NumR;
    vecRu=BboxR(j,[1 2]);
    vecRu=vecRu';
    vecRd=[BboxR(j,1)+BboxR(j,3);BboxR(j,2)+BboxR(j,4)];
    logic1 = vecRu>vecRup;
    logic2 = vecRu<vecRdown;
    logic3 = vecRd>vecRup;
    logic4 = vecRd<vecRdown;
    logic = and(and(logic1,logic2),and(logic3,logic4));
    if and(logic(1),logic(2));
        ind = 3;
        diffVec = CenR(j,:)-CenB(i,:);
        diffVec = diffVec';
        theta = atan2d(diffVec(2),diffVec(1));
%       theta = single(theta);
        PosMat(ind,:) = [CenB(i,:) theta ind];
    end
end
end

for i=1:4;
    pos = PosMat(i,[1 2]);
    pos = ([pos(1) pos(2)] - [240 235]/2 );
    pos = [pos(1) -pos(2)];
    PosMat(i,[1 2]) = pos*16/83.86;
    PosMat(i,3)=-PosMat(i,3);
end

```

B.2 Servo Motor Angle Calculation

```
function [ths,dir1] = fcn(thd,thm)
%#codegen

dir = false;
d1 = thm-90;
d2 = thm+90;
if abs(d1)>180;
    d1 = d1 - sign(d1)*360;
end
if abs(d2)>180;
    d2 = d2 - sign(d2)*360;
end
temp1 = d1;
temp2 = d2;
if d1>d2;
    d1 = temp2;
    d2 = temp1;
end
if thd<d1 || thd>d2;
    thd = thd-sign(thd)*180;
%     if abs(thm)<=90;
%         dir = ~dir;
%     end
end
ths = thm - 90 - thd;
if abs(ths)>180;
    ths = ths - sign(ths)*360;
%     dir = 1;
end
if ths<0;
    ths = ths + 180;
%     dir = 1;
end
if abs(thm)>90;
    dir = ~dir;
end
```

```
dir1 = double(dir);
```

B.3 IDE-Code Uploaded to the Arduino

Code written by: Giampiero Campa. This code written in the arduino IDE has to be uploaded on the Arduino board so that real time communication is established between the Arduino board and Simulink.

```
/* Analog and Digital Input and Output Server for MATLAB */
/* Giampiero Campa, Copyright 2013 The MathWorks, Inc */

/* This file is meant to be used with the MATLAB arduino IO
   package, however, it can be used from the IDE environment
   (or any other serial terminal) by typing commands like:
0e0 : assigns digital pin #4 (e) as input
0f1 : assigns digital pin #5 (f) as output
0n1 : assigns digital pin #13 (n) as output
1c : reads digital pin #2 (c)
1e : reads digital pin #4 (e)
2n0 : sets digital pin #13 (n) low
2n1 : sets digital pin #13 (n) high
2f1 : sets digital pin #5 (f) high
2f0 : sets digital pin #5 (f) low
4j2 : sets digital pin #9 (j) to 50=ascii(2) over 255
4jz : sets digital pin #9 (j) to 122=ascii(z) over 255
3a : reads analog pin #0 (a)
3f : reads analog pin #5 (f)
R0 : sets analog reference to DEFAULT
R1 : sets analog reference to INTERNAL
R2 : sets analog reference to EXTERNAL
X3 : roundtrip example case returning the input (ascii(3))
99 : returns script type (0 adio.pde ... 3 motor.pde ) */
/* define internal for the MEGA as 1.1V (as as for the 328) */
#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
#define INTERNAL INTERNAL1V1
#endif
void setup() {
```

```

    /* initialize serial */
    Serial.begin(115200);
}
void loop() {
    /* variables declaration and initialization */
    static int  s  = -1;    /* state */
    static int  pin = 13;   /* generic pin number */
    int  val = 0;           /* generic value read from serial */
    int  agv = 0;           /* generic analog value */
    int  dgv = 0;           /* generic digital value */
    /* The following instruction constantly checks if anything
       is available on the serial port. Nothing gets executed in
       the loop if nothing is available to be read, but as soon
       as anything becomes available, then the part coded after
       the if statement (that is the real stuff) gets executed */
    if (Serial.available() >0) {
        /* whatever is available from the serial is read here */
        val = Serial.read();
        /* This part basically implements a state machine that
           reads the serial port and makes just one transition
           to a new state, depending on both the previous state
           and the command that is read from the serial port.
           Some commands need additional inputs from the serial
           port, so they need 2 or 3 state transitions (each one
           happening as soon as anything new is available from
           the serial port) to be fully executed. After a command
           is fully executed the state returns to its initial
           value s=-1 */
        switch (s) {
            /* s=-1 means NOTHING RECEIVED YET ***** */
            case -1:
                /* calculate next state */
                if (val>47 && val<90) {
                    /* the first received value indicates the mode
                       49 is ascii for 1, ... 90 is ascii for Z
                       s=0 is change-pin mode;
                       s=10 is DI; s=20 is DO; s=30 is AI; s=40 is AO;
                       s=90 is query script type (1 basic, 2 motor);
                       s=340 is change analog reference;
                       s=400 example echo returning the input argument;

```



```

                                                                    */
    s=10*(val-48);
}

/* the following statements are needed to handle
   unexpected first values coming from the serial (if
   the value is unrecognized then it defaults to s=-1) */
if ((s>40 && s<90) || (s>90 && s!=340 && s!=400)) {
    s=-1;
}

/* the break statements gets out of the switch-case, so
/* we go back and wait for new serial data */
break; /* s=-1 (initial state) taken care of */

/* s=0 or 1 means CHANGE PIN MODE */

case 0:
/* the second received value indicates the pin
   from abs('c')=99, pin 2, to abs(' ') =166, pin 69 */
if (val>98 && val<167) {
    pin=val-97; /* calculate pin */
    s=1; /* next we will need to get 0 or 1 from serial */
}
else {
    s=-1; /* if value is not a pin then return to -1 */
}
break; /* s=0 taken care of */

case 1:
/* the third received value indicates the value 0 or 1 */
if (val>47 && val<50) {
    /* set pin mode */
    if (val==48) {
        pinMode(pin, INPUT);
    }
    else {

```

```

        pinMode(pin, OUTPUT);
    }
}
s=-1; /* we are done with CHANGE PIN so go to -1 */
break; /* s=1 taken care of */

/* s=10 means DIGITAL INPUT ***** */

case 10:
/* the second received value indicates the pin
   from abs('c')=99, pin 2, to abs(' ') =166, pin 69 */
if (val>98 && val<167) {
    pin=val-97; /* calculate pin */
    dgv=digitalRead(pin); /* perform Digital Input */
    Serial.println(dgv); /* send value via serial */
}
s=-1; /* we are done with DI so next state is -1 */
break; /* s=10 taken care of */

/* s=20 or 21 means DIGITAL OUTPUT ***** */

case 20:
/* the second received value indicates the pin
   from abs('c')=99, pin 2, to abs(' ') =166, pin 69 */
if (val>98 && val<167) {
    pin=val-97; /* calculate pin */
    s=21; /* next we will need to get 0 or 1 from serial */
}
else {
    s=-1; /* if value is not a pin then return to -1 */
}
break; /* s=20 taken care of */

case 21:
/* the third received value indicates the value 0 or 1 */
if (val>47 && val<50) {

```

```

        dgv=val-48;                /* calculate value          */
digitalWrite(pin,dgv);           /* perform Digital Output */
    }
    s=-1; /* we are done with DO so next state is -1          */
    break; /* s=21 taken care of                               */

/* s=30 means ANALOG INPUT ***** */

case 30:
/* the second received value indicates the pin
   from abs('a')=97, pin 0, to abs('p')=112, pin 15          */
if (val>96 && val<113) {
    pin=val-97;                /* calculate pin          */
    agv=analogRead(pin);       /* perform Analog Input   */
    Serial.println(agv);       /* send value via serial  */
}
    s=-1; /* we are done with AI so next state is -1          */
    break; /* s=30 taken care of                               */

/* s=40 or 41 means ANALOG OUTPUT ***** */

case 40:
/* the second received value indicates the pin
   from abs('c')=99, pin 2, to abs(' ')=166, pin 69          */
if (val>98 && val<167) {
    pin=val-97;                /* calculate pin          */
    s=41; /* next we will need to get value from serial      */
}
else {
    s=-1; /* if value is not a pin then return to -1          */
}
    break; /* s=40 taken care of                               */

case 41:
/* the third received value indicates the analog value */

```

```

    analogWrite(pin, val);          /* perform Analog Output */
    s=-1; /* we are done with A0 so next state is -1 */
    break; /* s=41 taken care of */

/* s=90 means Query Script Type:
   (0 adio, 1 adioenc, 2 adiosrv, 3 motor) */

case 90:
if (val==57) {
    /* if string sent is 99 send script type via serial */
    Serial.println(0);
}
s=-1; /* we are done with this so next state is -1 */
break; /* s=90 taken care of */

/* s=340 or 341 means ANALOG REFERENCE ***** */

case 340:
/* the second received value indicates the reference,
   which is encoded as is 0,1,2 for DEFAULT, INTERNAL
   and EXTERNAL, respectively. Note that this function
   is ignored for boards not featuring AVR or PIC32 */

#if defined(__AVR__) || defined(__PIC32MX__)

    switch (val) {

        case 48:
            analogReference(DEFAULT);
            break;

        case 49:
            analogReference(INTERNAL);
            break;

        case 50:

```

```

        analogReference(EXTERNAL);
        break;

        default:                /* unrecognized, no action */
        break;
    }

#endif

    s=-1; /* we are done with this so next state is -1 */
    break; /* s=341 taken care of */

    /* s=400 roundtrip example function (returns the input)*/

    case 400:
    /* the second value (val) can really be anything here */

    /* This is an auxiliary function that returns the ASCII
       value of its first argument. It is provided as an
       example for people that want to add their own code */
    /* your own code goes here instead of the serial print */
    Serial.println(val);
    s=-1; /* we are done with the aux function so -1 */
    break; /* s=400 taken care of */
    /* ***** UNRECOGNIZED STATE, go back to s=-1 ***** */
    default:
    /* we should never get here but if we do it means we
       are in an unexpected state so whatever is the second
       received value we get out of here and back to s=-1 */

    s=-1; /* go back to the initial state, break unneeded */
} /* end switch on state s
} /* end if serial available

} /* end loop statement

```