

# Kinetic Pace: Bipedal Robot

A Project Report

submitted by

*Devansh Garg*

(2019UME1883)

under the supervision of

*Dr. Dinesh Kumar*

in partial fulfillment of

the requirements for the degree of  
Bachelors of Mechanical Engineering

to the



Department of Mechanical Engineering  
Malaviya National Institute of Technology Jaipur

May 2023

MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY JAIPUR  
DEPARTMENT OF MECHANICAL ENGINEERING

## DECLARATION

I hereby declare that the work reported in the project titled “**Kinetic Pace: Bipedal Robot**” submitted for the partial fulfillment of B.Tech. degree at the Department of Mechanical Engineering, Malaviya National Institute of Technology Jaipur, is record of my work carried out under the supervision of *Dr. Dinesh Kumar*.

Devansh Garg

(B.Tech. Final)

Department of Mechanical Engineering

MNIT Jaipur

May 2023

MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY JAIPUR  
DEPARTMENT OF MECHANICAL ENGINEERING

## CERTIFICATE

It is certified that **Devansh Garg** has submitted project under my supervision for partial fulfillment of B.Tech. degree in Mechanical Engineering on the topic “**Kinetic Pace: Bipedal Robot**”. It is further certified that the above candidate has carried out the project work under my guidance during the academic session 2022-2023 at the Department of Mechanical Engineering, Malaviya National Institute of Technology Jaipur.

Dr. Dinesh Kumar

Associate Professor

Department of Mechanical Engineering

MNIT Jaipur- 302017

# ACKNOWLEDGEMENT

As a matter of first importance, I offer my genuine thanks to my supervisor Dr. Dinesh Kumar, Associate Professor, MNIT Jaipur. I appreciate his support and help during the project work.

I want to express my sincere gratitude to Prof. Andy Ruina, Professor, Sibley School of Mechanical and Aerospace Engineering, Cornell University. His enthusiasm, patience, insightful comments, helpful information, practical advice, and unceasing ideas have helped me at all times in the research of this thesis. It was indeed a rich learning experience to interact with him and his ever-supporting guidance, making it a learning conducive project. Without his constant support and guidance, this project would not have been possible.

Lastly, my family's constant encouragement, understanding, and belief in my abilities have been instrumental in my academic journey. I am grateful for their sacrifices, patience, and understanding during the challenging moments of this endeavor.

Devansh Garg  
(2019UME1883)

# ABSTRACT

This project focuses on the design and development of a bipedal robot that utilizes the natural dynamics of the Inverted Pendulum Model to optimize energy efficiency. The robot incorporates a code parser, similar to a G-Code parser in CNC machines, for precise control and motion planning.

A key feature of the robot is its serial-parallel hybrid leg mechanism, which combines serial and parallel mechanisms to optimize workspace and load distribution. The primary objective is to design and build a bipedal robot that maximizes energy efficiency using the Inverted Pendulum Model. Secondary objectives include the implementation of a code parser and an embedded system for robot control, as well as testing the robot's performance in simulated and real-world environments.

The outcomes of this project, including the architectural and mechanical designs, and the code parser, will serve as a foundation for the subsequent project, which will focus on implementing an improved controller design. This future phase aims to validate the effectiveness of the enhanced algorithm in enhancing stability and energy efficiency during practical applications.

# PLAGIARISM REPORT

## Kinetic Pace: Bipedal Robot

### ORIGINALITY REPORT

**7%** SIMILARITY INDEX    **5%** INTERNET SOURCES    **3%** PUBLICATIONS    **3%** STUDENT PAPERS

### PRIMARY SOURCES

1	v1.overleaf.com Internet Source	1 %
2	hdl.handle.net Internet Source	1 %
3	"Humanoid Robotics: A Reference", Springer Science and Business Media LLC, 2019 Publication	1 %
4	repozitorij.uni-lj.si Internet Source	1 %
5	www.overleaf.com Internet Source	<1 %
6	Submitted to Malaviya National Institute of Technology Student Paper	<1 %
7	Qi Zou, Dan Zhang, Guanyu Huang. "Dynamic performance evaluation of the parallel mechanism for a 3T2R hybrid robot", Mechanism and Machine Theory, 2022 Publication	<1 %
dspace.dtu.ac.in:8080		

17	asmedigitalcollection.asme.org Internet Source	<1 %
18	la.disneyresearch.com Internet Source	<1 %
19	Submitted to NIIT University Student Paper	<1 %
20	Kevin G. Gim, Joohyung Kim, Katsu Yamane. "Design and Fabrication of a Bipedal Robot Using Serial-Parallel Hybrid Leg Mechanism", 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018 Publication	<1 %
21	www.science.gov Internet Source	<1 %
22	"Sensors Set", Wiley, 1995 Publication	<1 %
23	Fredy Martinez, Angelica Rendon. "Autonomous Motion Planning for a Differential Robot using Particle Swarm Optimization", International Journal of Advanced Computer Science and Applications, 2023 Publication	<1 %

8	Internet Source	<1 %
9	123docz.net Internet Source	<1 %
10	"Proceedings of the International Conference on Recent Cognizance in Wireless Communication & Image Processing", Springer Science + Business Media, 2016 Publication	<1 %
11	Submitted to University of Derby Student Paper	<1 %
12	catalog.libraries.psu.edu Internet Source	<1 %
13	eprints.uthm.edu.my Internet Source	<1 %
14	ko.overleaf.com Internet Source	<1 %
15	Hao Ma, Chunhao Zhong, Bing Chen, Kai-Ming Chan, Wei-Hsin Liao. "User-adaptive Assistance of Assistive Knee Braces for Gait Rehabilitation", IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2018 Publication	<1 %
16	Submitted to University of Leicester Student Paper	<1 %

# Contents

<b>1</b>	<b>Introduction and Background</b>	<b>6</b>
1.1	Introduction . . . . .	6
1.2	Background . . . . .	7
1.3	Objectives . . . . .	7
<b>2</b>	<b>Architecture</b>	<b>9</b>
2.1	Robot Architecture . . . . .	9
2.2	Serial-Parallel Hybrid Leg Mechanism . . . . .	10
2.3	Embedded System . . . . .	10
2.4	Code Parser . . . . .	10
2.5	Manufacturing . . . . .	10
<b>3</b>	<b>Mechanical Design</b>	<b>11</b>
3.1	Components . . . . .	11
3.1.1	Feet . . . . .	11
3.1.2	Hip . . . . .	11
3.1.3	5-bar Linkage . . . . .	11
3.2	Materials . . . . .	13
3.2.1	Metal Manufacturing . . . . .	14
3.3	Mechanisms . . . . .	16
3.4	CAD Model . . . . .	16
<b>4</b>	<b>Embedded Systems</b>	<b>18</b>
4.1	Architecture . . . . .	18
4.2	Components . . . . .	18
4.3	Communication Protocols . . . . .	19

4.4	Code Parser . . . . .	19
4.5	Conclusion . . . . .	20
<b>5</b>	<b>Code Parser</b>	<b>21</b>
5.1	Design . . . . .	21
5.2	Implementation . . . . .	22
5.3	Usage (Documentation) . . . . .	28
5.3.1	General Guideline . . . . .	29
5.3.2	Sequence of Command . . . . .	29
5.3.3	Speed Commands . . . . .	29
5.3.4	Speed Commands stored as Global Variables . . . . .	30
5.3.5	Absolute Value for Positions . . . . .	30
5.3.6	More Examples: . . . . .	30
5.4	Conclusion . . . . .	31
<b>6</b>	<b>Controller</b>	<b>32</b>
6.1	Petr Zeytsev's Controller . . . . .	32
6.1.1	Extended Viable and Controllable Regions . . . . .	32
6.1.2	Inverted Pendulum Model . . . . .	33
6.1.3	Choosing a Controller . . . . .	33
6.2	Improvements . . . . .	33
6.2.1	Building Blocks . . . . .	34
6.2.2	Derivation of Better Nominal State . . . . .	34
<b>7</b>	<b>Simulation, Testing and Results</b>	<b>36</b>
7.1	Comparison of TCOT for Initial and New Nominal States . . . . .	36
7.2	Simulation . . . . .	36
7.3	Results . . . . .	37
7.4	Conclusion . . . . .	38
<b>8</b>	<b>Conclusion</b>	<b>39</b>
8.1	Mechanical Design . . . . .	39
8.2	Embedded Systems . . . . .	39
8.3	Code Parser . . . . .	40



8.4	Improved Controller . . . . .	40
8.5	Controller Implementation on Physical Robot in Subsequent Project .	40
<b>9</b>	<b>Applications and Future Scope</b>	<b>41</b>
9.1	Potential Applications . . . . .	41
9.2	Further Research . . . . .	42
9.3	Conclusion . . . . .	42
<b>10</b>	<b>Visual Documentation</b>	<b>43</b>

# List of Figures

3.1	Single Leg Schematic: Green Joints (Actively powered by Servo Motors); White Joints (Passive Joints) . . . . .	16
3.2	CAD Model View 1 . . . . .	17
3.3	CAD Model View 2 . . . . .	17
3.4	Rendered View of the Robot . . . . .	17
4.1	Embedded Systems Architecture . . . . .	19
6.1	Poincare Map Equation . . . . .	33
7.1	C1 for Nominal Angular Velocity of 0.19 . . . . .	37
7.2	C1 for Nominal Angular Velocity of 0.512 . . . . .	37
7.3	C1 for both Nominal Angular Velocities . . . . .	37
7.4	C1 for both Nominal Angular Velocities ( $\dot{\theta}_0$ vs p) . . . . .	38
7.5	C1 for both Nominal Angular Velocities ( $\dot{\theta}_0$ vs $x_{st}$ ) . . . . .	38
10.1	Initial Prototype Initial Model CAD Model View 1 . . . . .	44
10.2	Initial Prototype Initial Model CAD Model View 2 . . . . .	44
10.3	Initial Prototype Physical Model View 1 . . . . .	44
10.4	Initial Prototype Physical Model View 2 . . . . .	44
10.5	Physical Model of Kinetic Pace for Final Year Project . . . . .	45
10.6	Physical Model of Kinetic Pace for Final Year Project . . . . .	45
10.7	Guide Rod for Final Year Project Physical Model . . . . .	46
10.8	Guide Rod for Final Year Project Physical Model . . . . .	46
10.9	Meticulous Wire Management with SMPS . . . . .	47
10.10	Meticulous Wire Management with SMPS . . . . .	47

# List of Tables

3.1	Component List . . . . .	13
-----	--------------------------	----

# Chapter 1

## Introduction and Background

This chapter provides an introduction to the project, including its background and objectives. The Inverted Pendulum Model has been identified as a popular approach to designing bipedal robots, and the project aims to design a robot that optimizes on energy using this model.

### 1.1 Introduction

Bipedal robots have gained significant attention in recent years due to their potential to perform tasks in unstructured environments that are difficult or impossible for wheeled robots. In particular, the Inverted Pendulum Model has been a popular approach to designing bipedal robots due to its simplicity and energy efficiency. This model considers the robot as an inverted pendulum that is constantly falling forward and requires controlled stepping to maintain balance.

This project focuses on designing a bipedal robot that uses the natural dynamics of the Inverted Pendulum Model to optimize energy. The robot also includes a code parser that can be used similarly to a G-Code parser in CNC machines. The robot is designed to have a unique mechanism, the serial-parallel hybrid leg, which leverages both serial and parallel mechanisms to optimize workspace and load distribution.

## 1.2 Background

The idea for this project originated from a research internship application that was sent to Professor Andy Ruina at Cornell University. After several meetings and discussions, the project was focused on improving the controller for a bipedal robot designed by one of his PhD students, Petr Zeytsev. After two months of reading and understanding the thesis, minor improvements were proposed and tested using simulations. These improvements showed not only an improvement in energy but also in the controllable regions of the robot, resulting in improved robustness.

From this experience, the idea of designing a bipedal robot to implement the improved algorithm originated. Research was conducted on the mechanism and hardware to be used, and the paper by Disney Research[1] on the serial-parallel hybrid leg caught my attention. The design was perfect for the Inverted Pendulum Model as it kept most of the leg mass at the hip and leveraged both serial and parallel mechanisms for optimal workspace and load distribution.

The initial prototype took about 1.5 years to complete before I went on a semester-long internship, and the project came to a temporary halt. Upon returning to college, the project was converted into the final year project, serving as a catalyst to motivate and actively propel me toward the successful completion of the project. During the six-month internship in Bangalore, I learned new design concepts and techniques for machining, 3D printing, and GD&T, and hence, the project was redesigned to include these improvements.

## 1.3 Objectives

The primary objective of this project is to design and build a bipedal robot that uses the natural dynamics of the Inverted Pendulum Model to optimize on energy. The robot is to be designed with a unique mechanism, the serial-parallel hybrid leg, which leverages both serial and parallel mechanisms to optimize workspace and load distribution.

Secondary objectives include designing and implementing a code parser that can be used similarly to a G-Code parser in CNC machines, designing and implementing an embedded system for controlling the robot, and testing the robot's performance in

simulated and real-world environments.

The project's outcomes (the Architectural Design, the Mechanical Design, and the Code Parser) will be instrumental in the subsequent project, where the improved controller design will be implemented. This next phase aims to demonstrate the effectiveness of the enhanced algorithm in improving the robot's stability and energy efficiency in practical settings. Isolating the algorithm implementation as a separate project, the focus of this project is on the robot's design and evaluation of the algorithm.

# Chapter 2

## Architecture

This chapter presents the design of the bipedal robot, including the robot architecture, serial-parallel hybrid leg mechanism, embedded system, code parser, manufacturing process, and simulation and testing.

### 2.1 Robot Architecture

The bipedal robot is designed to use the Inverted Pendulum Model, which is a simple yet effective approach to maintaining balance. The robot consists of a torso, two legs, and two feet, with each leg comprising of a serial-parallel hybrid mechanism that is connected to the torso via a hip-roll joint. Each leg consists of 6 degrees of motion, namely: Hip Pitch, Hip Roll, Knee Pitch, Foot Pitch, Foot Roll, and Foot Yaw.

To avoid overextension of the research and ensure the quality of the results, the Hip Roll, Foot Roll, and Foot Yaw motions will be constrained as follows:

1. Electronically (by using code-Parser to prevent asymmetric motion between the 5-bar chains of a leg ), and
2. Mechanically (by using 1 DOF redundant joint on the Foot shaft instead of universal joint and ball joint on either ends).

## **2.2 Serial-Parallel Hybrid Leg Mechanism**

The serial-parallel hybrid leg mechanism is the core design element of the robot's legs. The mechanism consists of two parallel linkages connected to a serial chain of links. The parallel linkages allow for load-sharing capacity among the servo motors, while the serial chain provides additional workspace, support, and stability.

## **2.3 Embedded System**

The robot is controlled by an embedded system, which includes a microcontroller (Arduino Uno), motor driver circuitry (consisting of Adafruit's 16-Channel Servo Motor Drive- PCA9685), and an Inertial Measurement Unit (IMU-6050 to get the Angles and the Angular Velocities while free-falling).

## **2.4 Code Parser**

A code parser is designed and implemented to enable the robot to move in a controlled manner. The parser is similar to a G-Code parser used in CNC machines and converts high-level movement commands into lower-level motor control signals. The parser is designed to accept commands such as "LK" for "Left Knee" or "RF" for "Right Foot" and convert them into appropriate motor control signals for the robot to execute.

## **2.5 Manufacturing**

The robot components are manufactured using a combination of manufacturing techniques. The joints are 3D Printed in Resin and PLA, while the limbs are cut out of Pultruded Carbon Fiber Rod. The manufacturing process is optimized to ensure accurate and repeatable results, and to keep the robot limbs safe for testing various motions.



# Chapter 3

## Mechanical Design

The mechanical design of the bipedal robot is crucial for its functionality, stability, and energy efficiency. This chapter outlines the mechanical design process, including the selection of materials, components, and mechanisms.

### 3.1 Components

#### 3.1.1 Feet

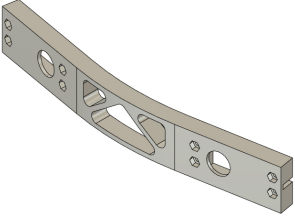
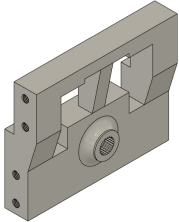
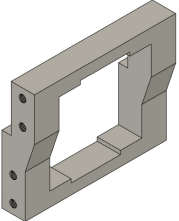
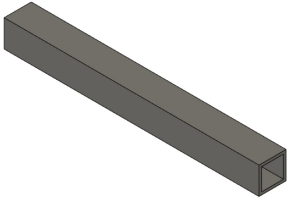
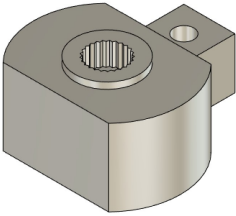
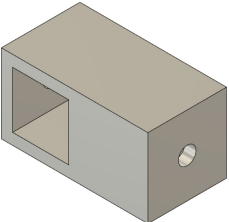
The robot's feet are designed to maximize ground contact and provide stability during movement. The feet consist of a foam layer for improved traction and prevention of damage due to impact.

#### 3.1.2 Hip

The robot's frame (the hip) is designed to provide rigidity and support for the legs and hold onto the electrical components and wiring.

#### 3.1.3 5-bar Linkage

The 5-bar linkage design offers a unique advantage by allowing the weight to be positioned at the top while maintaining control over the knee joint, which is located at a distance from the hip.

Components List		
Component Name	CAD Model	Remarks
Hip Frame		For holding both legs and the electronics assembly
Motor Back Frame		For holding 4 butt motors of a leg
Motor Front Frame		For holding 4 butt motors of a leg
Carbon Fiber Rod		Acts as limb (links) between various Joints
Joints		Various Joints derived from same basic shape
Interconnect between Carbon Fiber Rods		Needed for Links with an offset in Perpendicular Direction

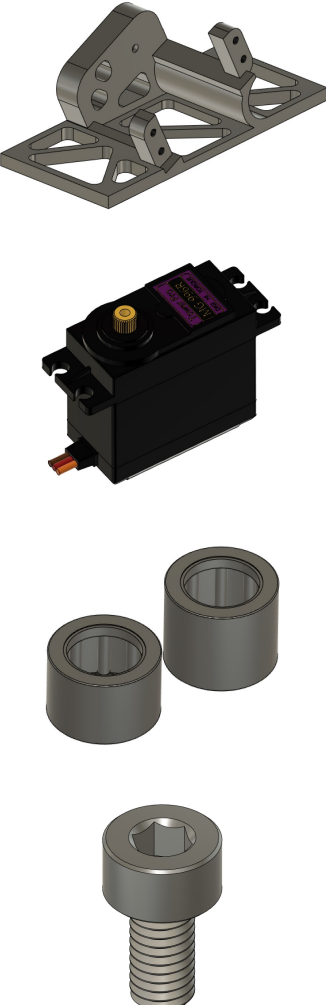
<p>Foot</p> <p>TowerPro MG996R Metal Gear 180° Servo</p> <p>HK0808 &amp; HK0810 Needle Roller Bearings</p> <p>M3 Screws</p>		<p>Foot</p> <p>For actuation of various Joints</p> <p>For smooth motion between Passive Joints</p> <p>Screws of different lengths as per requirement</p>
---	--	--

Table 3.1: Component List

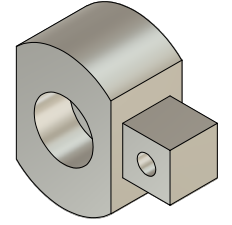
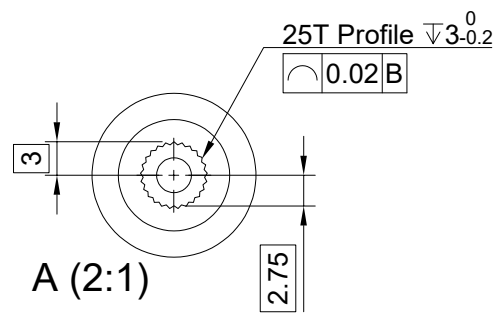
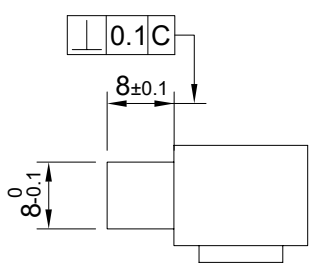
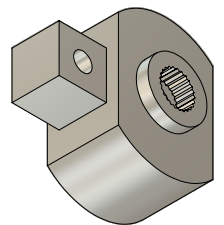
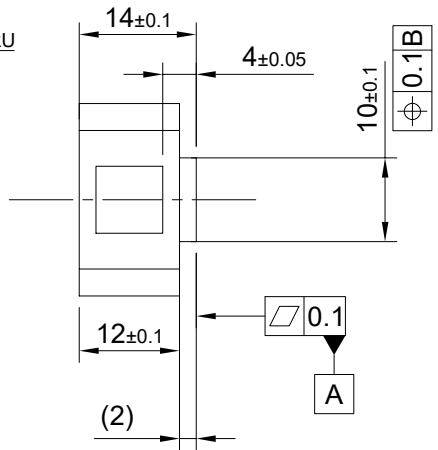
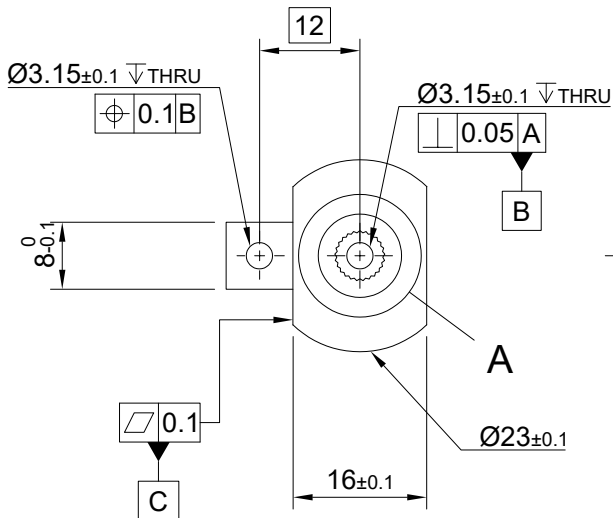
## 3.2 Materials

The materials used in the mechanical design of the robot are selected based on their strength, durability, weight, and cost.

The joints selected for this project are made from Resin SLA 3D-printing technology and are directly affixed to the servo motor. These joints possess a 25T gear profile that can only be achieved through high-resolution SLA printing, making it a suitable manufacturing technique.

### **3.2.1 Metal Manufacturing**

In the pursuit of creating a robust and functional mechanical system, I recognized the importance of designing metal-manufacturing capable joints for my project. These joints would not only provide strength and durability but also ensure compatibility with metal fabrication processes, specifically the use of a Virtual Machining Center (VMC). However, the constraints of time and budget presented significant challenges. As a result, I was unable to proceed with the actual metal manufacturing process, but the following shows the drawing files that can be used for manufacturing the joints in metal:



PART NAME: Joint_Motor v4 MATERIAL: Aluminum WEIGHT: 10.927 g	CREATED BY: Devansh Garg DATE: 09-02-2023
	SHEET NUMBER: 1/1 REVISION NUMBER:

Other Joints and robot structure are FDM 3D-printed in PLA to keep costs low.

Lastly, the limbs are made out of carbon fiber rods because of their low weight yet high strength to bare the impact forces.

### 3.3 Mechanisms

The five-bar linkage mechanism in the robot's legs is designed to concentrate the mass of motors near the hip yet provide a big work envelope. A simple schematic below shows the active and the redundant joints in mechanism.

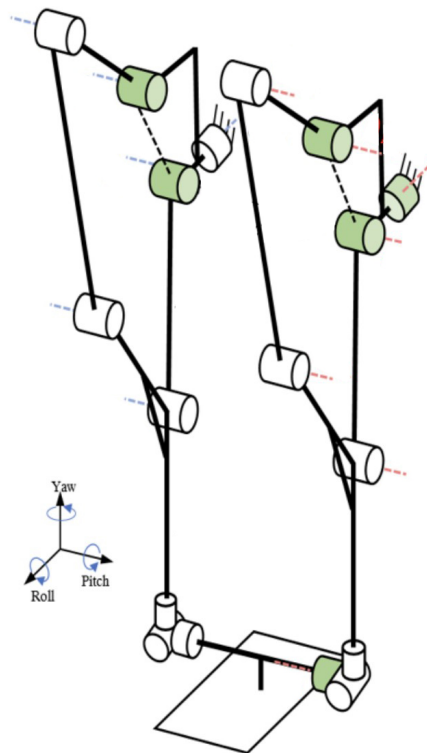


Figure 3.1: Single Leg Schematic: Green Joints (Actively powered by Servo Motors); White Joints (Passive Joints)

### 3.4 CAD Model

A detailed CAD model of the robot is created to ensure the mechanical design is feasible and could be assembled correctly. The CAD model is used to verify that the robot's components would fit together and operate as intended. It is also used to simulate the robot's movements and validate its stability.

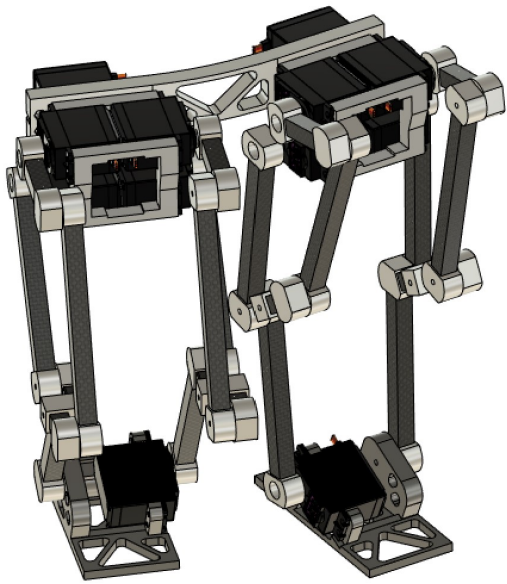


Figure 3.2: CAD Model View 1

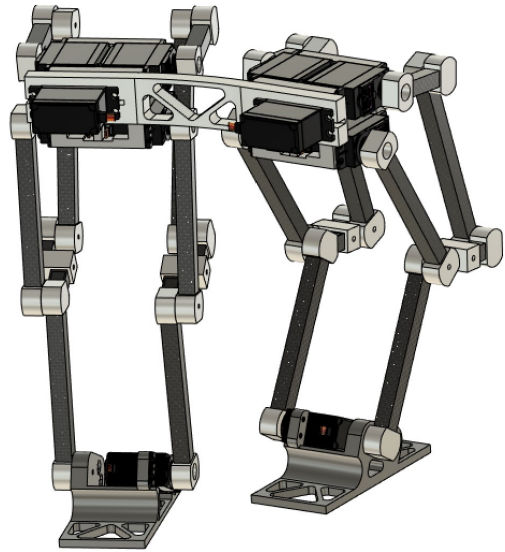


Figure 3.3: CAD Model View 2



Figure 3.4: Rendered View of the Robot

# Chapter 4

## Embedded Systems

Embedded systems are a vital component of Kinetic Pace, enabling its real-time performance and energy efficiency. This chapter explores the microcontrollers, sensors, and control algorithms that power the robot's natural movement and code parsing capabilities.

### 4.1 Architecture

The embedded system used in this project comprises two main components, an Arduino and a Raspberry Pi. The Arduino is responsible for controlling the servo motors using the PCA9685 driver. The Raspberry Pi serves as the main controller and is responsible for communicating with the Arduino and processing sensor data from the IMU.

### 4.2 Components

The main components used in the embedded system are the Arduino Uno, Raspberry Pi 4, PCA9685 PWM Driver, and the MPU6050 IMU. The servos are powered by a 5V, 21Amps SMPS through the PCA9865. The PCA9865 and the MPU6050's ICs are powered by the Arduino. The servo motors used in the project are the MG996R with a 180-degree rotation capability. A total of 12 motors are used, each connected to the PCA9685 driver.



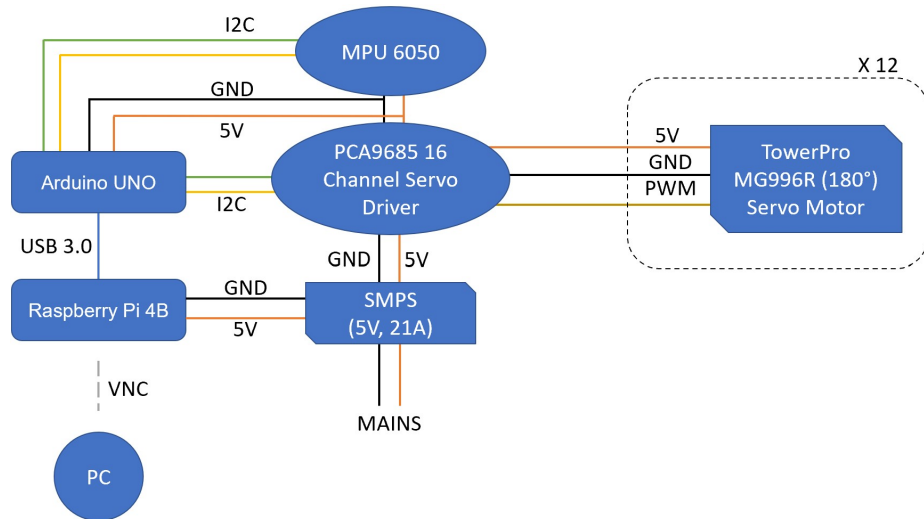


Figure 4.1: Embedded Systems Architecture

### 4.3 Communication Protocols

The PCA9685 PWM driver is connected to the Arduino using the I2C protocol. The IMU is also connected to the Arduino using the I2C protocol and is powered by the Arduino. The Arduino is programmed to receive data from the IMU and send the motor control signals to the PCA9685 driver using the I2C protocol. The Raspberry Pi is connected to the Arduino via the USB 3.0 port and is programmed to receive data, run the simulations and do High Processing Tasks.

The communication between the Raspberry Pi and the Arduino is achieved using the PySerial library in Python. The Raspberry Pi sends control signals to the Arduino in the form of bytes. The Arduino receives the bytes and processes them to generate the required motor control signals.

### 4.4 Code Parser

The code parser is designed to run on the Arduino Uno and enables users to send commands to the board through the serial monitor. These commands allow users to either request data or direct the robot to perform a prescribed motion. By using the serial monitor, the Raspberry Pi can send data outputted from the algorithm to the Arduino and enable the robot to walk more intelligently. Essentially, this system provides the robot with a “brain.”

## 4.5 Conclusion

Overall, the embedded system architecture is designed to provide a reliable and efficient control mechanism for the robot. The combination of the Raspberry Pi and the Arduino provides a flexible and scalable platform for future developments in the project. The use of standard communication protocols such as I2C and USB ensures compatibility with a wide range of sensors and other devices, allowing for easy integration with the system.

# Chapter 5

## Code Parser

The code parser is a crucial component of the bipedal robot control system as it serves as an interface between the user and the robot. The purpose of the code parser is to interpret user commands and convert them into executable instructions for the robot.

The code parser allows the user to send commands to the Arduino board via the serial monitor. The commands can be used to request data or make the robot perform a prescribed motion. The use of the serial monitor allows for real-time communication between the user and the robot, enabling the user to monitor the robot's performance and adjust its behavior as needed. The user here can be Raspi or a Human directly commanding data over Serial

The code parser is designed to recognize several different types of commands, including motion commands, data requests, and calibration commands. The motion commands are used to control the robot's movements, while the data requests are used to request sensor data from the robot. The calibration commands are used to calibrate the sensors and actuators to ensure accurate measurements and precise movements.

### 5.1 Design

The code Parser is designed such that it can implement motions of several kind together and each completing the motion with unique speeds. This additional feature of motions being completed in different time using different speeds is not normally

seen in G-Code Parsers used in CNC machines.

## 5.2 Implementation

```
// Defining All Motors. In terms of index. Not channels
#define LFoot 0 //Foot
#define LL2 1 //Main
#define LL1 2 //Knee
#define LHip 3 //Hip
#define LR1 4 //Knee
#define LR2 5 //Main
#define RL2 6 //Main
#define RL1 7 //Knee
#define RHip 9 //Hip
#define RR1 9 //Knee
#define RR2 10 //Main
#define RFoot 11 //Foot

uint8_t MOTOR_PIN_ASSIGNMENT[12] = {0,2,3,4,6,7,8,9,11,12,13,15};

//COMMAND SELECTOR Groups parsed in commands_bytearray
//[gs,lk,lh,lf,rk,rh,rf,empty]
#define COMMAND_SELECTOR_G bit(7) // For Global Speed
#define COMMAND_SELECTOR_LK bit(6) // For Left Knee
#define COMMAND_SELECTOR_LH bit(5) // For Left Hip
#define COMMAND_SELECTOR_LF bit(4) // For Left Foot
#define COMMAND_SELECTOR_RK bit(3) // For Right Knee
#define COMMAND_SELECTOR_RH bit(2) // For Right Hip
#define COMMAND_SELECTOR_RF bit(1) // For Right Foot

#define MAX_SPEED 1 // In Microseconds
#define MIN_SPEED 16384 // In Microseconds
#define DEFAULT_SPEED 50 // In Microseconds

uint16_t CURRENT_POSITION[12] = {};
// Tracks the 12 Values of Motors in Present

uint16_t GLOBAL_SPEED = DEFAULT_SPEED;
```

```

bool INVALID_COMMAND = false;

struct command {
    uint16_t value = 0;
    uint16_t speed = DEFAULT_SPEED;
} lk, lh, lf, rk, rh, rf;

void motion(uint8_t commands_bytearray) {
    if(INVALID_COMMAND) {
        Serial.println("INVALID_COMMAND");
        return;
    }
    uint8_t desired_reached = 1;
    desired_reached |= ~commands_bytearray;
    // Uncommanded bits already at position

    desired_reached |= COMMAND_SELECTOR_G;
    // GS bit, Useless

    uint32_t time_counter = 0;
    while(desired_reached != 255) {
        time_counter++;

        // Left Knee
        if (time_counter%lk.speed == 0 && !((desired_reached >> 6)&1)) {
            if (CURRENT_POSITION[LL1] > lk.value) {
                CURRENT_POSITION[LL1] = CURRENT_POSITION[LL1] -1;
                pwm.setPWM(MOTOR_PIN_ASSIGNMENT[LL1], 0,
                    max(SERVOMIN,HOMING_POSITION+HOMING_OFFSETS[LL1]
                    -2048+CURRENT_POSITION[LL1]));
                CURRENT_POSITION[LR1] = CURRENT_POSITION[LR1] -1;
                pwm.setPWM(MOTOR_PIN_ASSIGNMENT[LR1], 0,
                    max(SERVOMIN,HOMING_POSITION+HOMING_OFFSETS[LR1]
                    +2048-CURRENT_POSITION[LR1]));
            }
            else {
                if (CURRENT_POSITION[LL1] < lk.value) {
                    CURRENT_POSITION[LL1] = CURRENT_POSITION[LL1] +1;
                    pwm.setPWM(MOTOR_PIN_ASSIGNMENT[LL1], 0,

```

```

        min(SERVOMAX,HOMING_POSITION+HOMING_OFFSETS[LL1]
        -2048+CURRENT_POSITION[LL1]));
        CURRENT_POSITION[LR1] = CURRENT_POSITION[LR1] +1;
        pwm.setPWM(MOTOR_PIN_ASSIGNMENT[LR1], 0,
        min(SERVOMAX,HOMING_POSITION+HOMING_OFFSETS[LR1]
        +2048-CURRENT_POSITION[LR1]));
    }
    else {desired_reached |= COMMAND_SELECTOR_LK;}
}
}

```

.  
.  
.

Similarly for

Left Hip  
Left Foot  
Right Knee  
Right Hip  
Right Foot

.  
.  
.

```

        delayMicroseconds(1);
    }
    return;
}

```

// Simplifies Command to only contain Alphanumeric\_characters that are  
// returned in UpperCase

```

void command_interpreter(char *command_simplified, int max_len) {
    int char_counter = 0;
    char c;
    while(Serial.available()) {
        delay(10);
        c = Serial.read();
        if(c == '\n') { //End of line reached

```

```

        command_simplified[char_counter++] = '\0';
        return;
    }
    else {
        if(!isAlphaNumeric(c)) { //Ignore Non-alphanumeric characters
        }
        else { //Append c in UpperCase Format
            command_simplified[char_counter++] = toupper(c);
        }
    }
}
command_simplified[char_counter] = '\0';
return;
}

```

```

// Execute the Simplified Commands received
void execute_command(char *line) {
    uint8_t char_counter = 0;
    char first_letter;
    char second_letter;
    uint8_t commands_bytearray = 0;
    while (line[char_counter] != '\0'){
        bool speedset = false;
        String value = "";
        String speed = "";
        char first_letter = line[char_counter];
        char_counter++;
        char second_letter = line[char_counter];
        char_counter++;
        switch (first_letter) {
            case 'G':
                switch (second_letter) {
                    case 'S':
                        commands_bytearray |= COMMAND_SELECTOR_G;
                        break;
                    default:
                        INVALID_COMMAND = true;
                }
            }
        break;
    }
}

```

```

case 'L':
    switch (second_letter) {
        case 'K':
            commands_bytearray |= COMMAND_SELECTOR_LK;
            break;
        case 'H':
            commands_bytearray |= COMMAND_SELECTOR_LH;
            break;
        case 'F':
            commands_bytearray |= COMMAND_SELECTOR_LF;
            break;
        default:
            INVALID_COMMAND = true;
    }
    break;
case 'R':
    switch (second_letter) {
        case 'K':
            commands_bytearray |= COMMAND_SELECTOR_RK;
            break;
        case 'H':
            commands_bytearray |= COMMAND_SELECTOR_RH;
            break;
        case 'F':
            commands_bytearray |= COMMAND_SELECTOR_RF;
            break;
        default:
            INVALID_COMMAND = true;
    }
    break;
default:
    INVALID_COMMAND = true;
}
for(uint16_t value_index =0; value_index <5; value_index++){
// 5 because max servo value can be of 4 digits + 1 to initiate
// char_counter++
    if (line[char_counter] >= '0' && line[char_counter] <= '9') {
        value.concat(line[char_counter]);
        char_counter++;
    }
}

```



```

}
else {
    if (line[char_counter] == 'S') {
        speedset = true;
        char_counter++;
        for(uint16_t speed_index =0; speed_index <4; speed_index++){
            // 4 because max speed can be of 4 digits
            if (line[char_counter] >= '0' && line[char_counter] <= '9') {
                speed.concat(line[char_counter]);
                char_counter++;
            }
            else {
                goto exitbothloops;
            }
        }
    }
}
}
exitbothloops: ;
switch(first_letter) {
    case 'G':
        lk.speed = value.toInt();
        lh.speed = value.toInt();
        lf.speed = value.toInt();
        rk.speed = value.toInt();
        rh.speed = value.toInt();
        rf.speed = value.toInt();
        break;
    case 'L':
        switch(second_letter) {
            case 'K':
                lk.value = value.toInt();
                if (speedset) { lk.speed = speed.toInt(); }
                break;
            case 'H':
                lh.value = value.toInt();
                if (speedset) { lh.speed = speed.toInt(); }
                break;
            case 'F':

```

```

        lf.value = value.toInt();
        if (speedset) { lf.speed = speed.toInt(); }
        break;
    }
    break;
case 'R':
    switch(second_letter) {
        case 'K':
            rk.value = value.toInt();
            if (speedset) { rk.speed = speed.toInt(); }
            break;
        case 'H':
            rh.value = value.toInt();
            if (speedset) { rh.speed = speed.toInt(); }
            break;
        case 'F':
            rf.value = value.toInt();
            if (speedset) { rf.speed = speed.toInt(); }
            break;
    }
    break;
    }
}

#ifdef CHECK_COMMAND_VALUES
check_values();
#endif
motion(commands_bytearray);
}

```

### 5.3 Usage (Documentation)

The Code Parser is a program that allows the user to send commands to the Arduino Board via the Serial Monitor running on Baud Rate of 115200. The program works by parsing the user's input, ignoring all characters that are not alphanumeric, and removing any whitespaces.

The following are the only valid commands that the Code Parser recognizes:

- LK: Left Knee Pitch
- LF: Left Foot Pitch
- LH: Left Hip Pitch
- RK: Right Knee Pitch
- RF: Right Foot Pitch
- RH: Right Hip Pitch
- GS: Set Global Speed
- S: Set Motion Speed

The L stands for Left Leg, while the R stands for Right Leg as seen from front of the Robot. The K stands for Knee Pitch, F stands for Foot Pitch, and H stands for Hip Pitch.

### **5.3.1 General Guideline**

After homing, the default current positions are set to 2048 for each motor. To send a valid command, it must be followed by numerical values that have a maximum value of 4096. The Code Parser will invalidate any commands that do not follow this rule.

### **5.3.2 Sequence of Command**

Commands can be entered in the Serial Monitor in any sequence. The only exception is that the S command for a motion must be immediately preceded by the corresponding motion command. However, motion commands need not necessarily be followed by S command and the motion command takes the Global Speed Value if set, or the default Speed Value.

### **5.3.3 Speed Commands**

The Speed commands (both GS and S) are defined in terms of Time Delays i.e. The larger the value, the slower the motions, and vice versa.

The GS command sets the global speed for all the motions. The GS command if used, assigns speed to all the motion commands except the ones followed by S command. However, an exception to this is that if the motion command has been assigned a S command, but the motion command precedes the GS command, it won't take the S value but the GS value.

Example: LK2060 S20 LF2040 GS30 RH2012 RK2048S10 RF2005

Result: LK, LF, RH and RF commands will have speed

= Value of GS i.e. 30.

(Note that the LK command even though is followed by S takes the GS value as it precedes the GS command)

RK command will have a speed = Value of S i.e. 10.

### 5.3.4 Speed Commands stored as Global Variables

If a motion is already set to a specific speed in a previous command entry instance, the speed does not change unless specified while sending commands the next time. This is because the value of speeds remains stored in global variables.

### 5.3.5 Absolute Value for Positions

All positions sent using the motion commands take absolute values, not relative. Therefore, it is important to remember that the homing position is 2048.

### 5.3.6 More Examples:

- To set the left knee pitch to 3000, the user can enter the command “LK 3000” in the Serial Monitor.
- To set the global speed to 500, the user can enter the command “GS 500”.
- To set the speed of the left foot pitch to 1000, the user can enter the command “LF 2048 S 1000”

## 5.4 Conclusion

In this project, the code parser plays a significant role in enabling the Raspberry Pi to send commands to the Arduino Uno board via the Serial Monitor. The user can request for data or make the robot perform a prescribed motion. The Serial Monitor allows for the Raspi to send the data outputted from the algorithm and make the robot walk smarter.

Future improvements can be made to the code parser to enhance its capabilities. One improvement may be to enable the parser to accept more complex commands. Additionally, improving the user interface by designing a more user-friendly Serial Monitor may enhance the user experience.

# Chapter 6

## Controller

This chapter provides an overview of the controller designed by Petr Zeytsev [2], as discussed in his dissertation. It summarizes the key components and principles of the original controller design, outlining its objectives and goals. Additionally, it presents the improvements made to the controller, highlighting the enhancements and modifications implemented to address any limitations or challenges. The chapter concludes with a discussion of the results and performance evaluation of the improved controller, along with the implications and potential future directions for further development.

### 6.1 Petr Zeytsev's Controller

The controller designed by Petr Zeytsev [2] leverages the two models: Pratt's Capture Region and the Viability Theory, and make Extended Viable and Controllable Regions for an Inverted Pendulum Model.

#### 6.1.1 Extended Viable and Controllable Regions

The concept of the Viable and Controllable Regions, which describes the state of the robot, has been expanded to include an extended definition incorporating the controls available to the robot to reach the target. This extended state is represented as (State, Controls), where State refers to the robot's current state, and Controls refers to the available control inputs that can be utilized to navigate toward the

desired target. This expanded definition provides a comprehensive understanding of the robot's capabilities and the range of actions it can take within its operational space.

### 6.1.2 Inverted Pendulum Model

By applying the basic principles of kinematics and dynamics to the Inverted Pendulum model, the following equation was derived:

$$m \left( \sqrt{\frac{2g}{l} (1 - \cos \theta_{sw}) + \dot{\theta}_1^2} - \sqrt{\frac{2g}{l} (1 - \cos \theta_{sw}) + \dot{\theta}_0^2 \cos 2\theta_{sw}} \right) = \frac{-p \sin 2\theta_{sw}}{l}$$

establishes Poincaré map or relation between  $\dot{\theta}_0$  and  $\dot{\theta}_1$  in terms of  $\theta_{sw}/x_{st}$  and  $p$

Figure 6.1: Poincare Map Equation

The above equation is a relation between

1. the robot's angular velocity at present state ( $\dot{\theta}_0$ )
2. the robot's angular velocity in next state ( $\dot{\theta}_1$ )
3. the push-off value i.e. the ground reaction on foot ( $p$ )
4. the leg stance either in terms of angle ( $\theta_0$ ) or stepping length ( $x_{st}$ )

### 6.1.3 Choosing a Controller

Upon further analysis of the constraints involved in walking, the initially derived controllable regions were found to be more limited. As a result, a controller was needed to be selected from the remaining restricted workspace of  $[\dot{\theta}_0, p, x_{st}]$ . This selection process involved considering the available options within this limited range and choosing a controller that best suited the desired walking behavior and objectives.

## 6.2 Improvements

The improvements made to the controller are built upon the foundational concepts outlined in the paper. These key building blocks include TCOT (Total Cost of Transport), the Poincare Map, and the Nominal State. By leveraging these concepts, the

controller design is refined to achieve an improved Nominal State. This leads to expanded controllable regions and a reduction in the total cost of transport.

### 6.2.1 Building Blocks

#### Total Cost of Transport (TCOT)

The paper describes the equation for TCOT (Total Cost of Transport) of the Cornell Robot (Ranger), which is defined as the total energy used divided by the product of the Weight of the robot and the Distance traveled.

$$TCOT = \frac{TotalEnergyUtilised}{Weight * Distance} \quad (6.1)$$

Specifically, for one step, the equation is given as follows

$$TCOT = \frac{0.011 + 0.017T_{st} + 0.5p^2}{x_{st}} \quad (6.2)$$

where  $T_{st}$  = Duration of Step.

#### Nominal State

The nominal state for the Cornell Robot Ranger was selected randomly. The nominal state represents the state that the robot strives to reach if it were to step infinitely. It can also be referred to as the ultimate target or final state. Once the nominal state is achieved, the push-off and stepping length are chosen in such a way that the angular velocity remains constant, thereby keeping the robot in the target state.

### 6.2.2 Derivation of Better Nominal State

The Poincare Map is used such that  $\dot{\theta}_1 = \dot{\theta}_0$  reduces it in the Nominal State, thereby leaving the equation in 3 variables:  $p$ ,  $x_{st}$  and  $\dot{\theta}_0$ .

A Python script is coded to evolve the equation of Theta with Time using the “odeint” module in the Scipy Library. The continuous state  $[0 < p < 1, 0 < \theta_0 < \pi/2, 0 < \dot{\theta}_0 < 1]$  is discretized and checked if each nominal state satisfied the constraints of walking, and the one which resulted in the least TCOT is chosen as the Best Nominal State.



## Procedure

This is done as follows:

1. The Continuous Angular Velocity ' $\dot{\theta}_0$ ' is discretized into steps of 1/1000 varying from 0 to 1
2. The Angular Velocity ' $\dot{\theta}_0$ ' is evolved with Time in steps of 0.001 seconds varying from 0 to 5 seconds
3. The values of the Stance Angle ' $\theta_0$ ' obtained are stored
4. The corresponding Push-off Values 'p' are found such that the Poincare Map results in Nominal States.
5. Now for all the states  $[p, x_{st}, \dot{\theta}_0]$  that resulted in Nominal State and satisfied the constraints of walking, TCOT is calculated and stored in a new variable called 'Least\_TCOT'
6. The 'Least\_TCOT' variable and its corresponding state  $[p, x_{st}, \dot{\theta}_0]$  are stored as the best Nominal State

The testing takes 48 Hours on an RTX-3090 compute-enabled graphics card and it is found that the Nominal State for the least TCOT is

$$[p, x_{st}, \dot{\theta}_0] = [0.127, 0.445, 0.512] \quad (6.3)$$

having the TCOT value equal to 0.0754.

# Chapter 7

## Simulation, Testing and Results

Simulation and testing are conducted to evaluate its performance in different scenarios. The simulation is carried out to evolve the controller equation with time and conclude its theoretical performance.

### 7.1 Comparison of TCOT for Initial and New Nominal States

The initial Nominal State for the Cornell Robot Ranger was  $[p, x_{st}, \dot{\theta}_0] = [0.046, 0.35, 0.19]$  having the TCOT value equal to 0.28, and the new derived Nominal State for the robot is  $[p, x_{st}, \dot{\theta}_0] = [0.127, 0.445, 0.512]$  having the TCOT value equal to 0.075. This is an improvement of 73% in terms of the Total Cost of Travel.

### 7.2 Simulation

The simulation is conducted using a Python script to find out the Controllable region C1 for New and Old Nominal States and compare the two.

The Python script involves making a work-space of all the states in the region  $[0 < p < 1, 0 < x_{st} < 2, 0 < \dot{\theta}_0 < 1]$ , and finding the ones that satisfy the constraints of walking and lead to the Nominal State in just one step. The computation is carried out for 200 discretized points in  $x_{st}$  and 100 discretized points in  $\dot{\theta}_0$ . The graphs for

the same are plotted.

## 7.3 Results

The following are the graphs obtained for Controllable Region 1 for the initial Nominal Angular Velocity (0.19), and the derived-improved Nominal Angular Velocity (0.512). The red Circle indicates the Nominal State of the Robot.

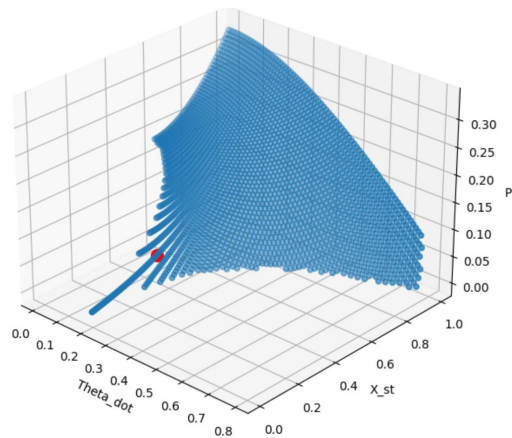


Figure 7.1: C1 for Nominal Angular Velocity of 0.19

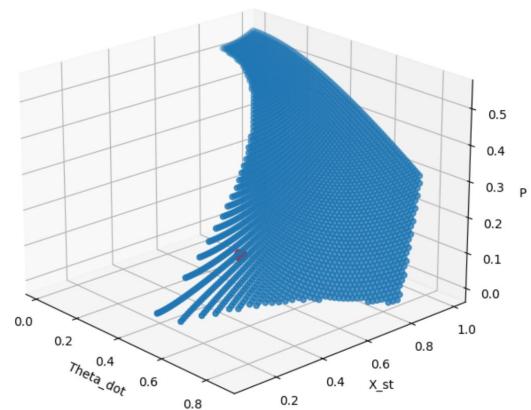


Figure 7.2: C1 for Nominal Angular Velocity of 0.512

The following graphs are combined for both the Nominal Angular Velocities. The Green is for the initial angular velocity and the Blue is for the new-derived angular velocity.

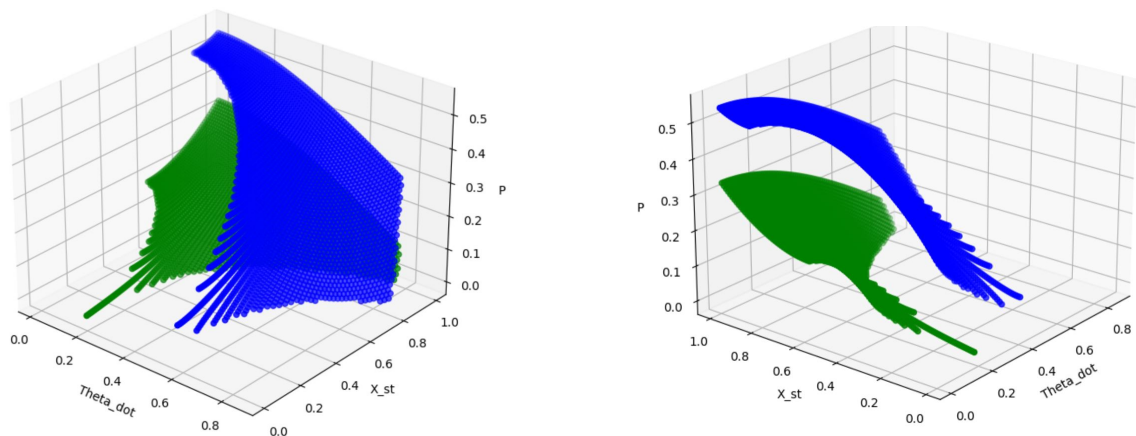


Figure 7.3: C1 for both Nominal Angular Velocities

The following graphs are combined for both the Nominal Angular Velocities as seen from the given different views.

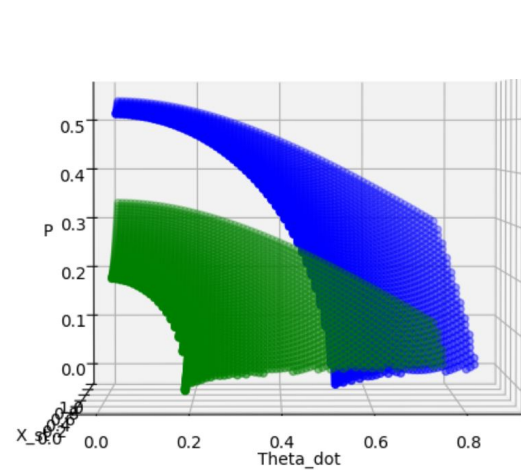


Figure 7.4: C1 for both Nominal Angular Velocities ( $\theta_0$  vs  $p$ )

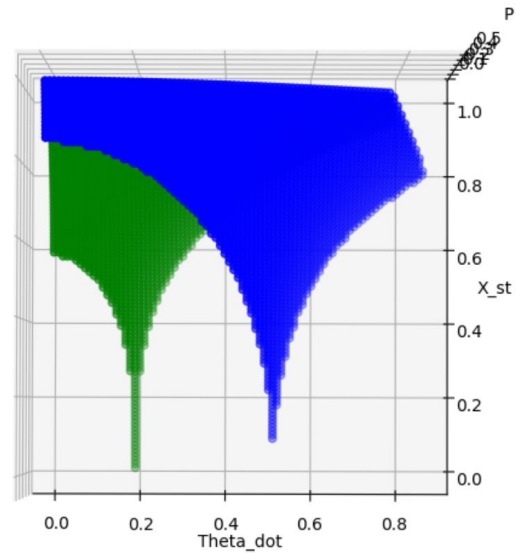


Figure 7.5: C1 for both Nominal Angular Velocities ( $\theta_0$  vs  $x_{st}$ )

## 7.4 Conclusion

The graphs derived from the data together indicate that there is a 17% increase in the area of C1. This, therefore, increases the state-space and provides more options for the robot to correct itself, if deviated from the target. Thus, it increases the robustness of the robot by 17%.

# Chapter 8

## Conclusion

The project has successfully accomplished its objectives, with the completion of the mechanical design, an appropriate and minimal embedded system, and a functional code parser. The improvements to the controller have elevated the robot's capabilities, resulting in an increased robustness and a significant reduction in the total cost of transport. These achievements pave the way for future advancements in the implementation of advanced algorithms, propelling the field of bipedal robotics forward.

### 8.1 Mechanical Design

In this project, I have successfully completed the mechanical design of the bipedal robot, Kinetic Pace. The design incorporates the principles of the Inverted Pendulum Model, leveraging its natural dynamics to optimize energy efficiency and stability. The mechanical components have been carefully engineered to ensure robustness and effective locomotion across various terrains. Through rigorous testing and refinement, I have achieved a reliable and functional mechanical system that meets the objectives of the project.

### 8.2 Embedded Systems

The embedded system implemented in Kinetic Pace has proven to be appropriate and minimal for the current objectives. The selection of micro-controllers, sensors, and

control algorithms allows for a real-time performance and reliable operation. The embedded system works seamlessly with the mechanical design, providing precise control and responsiveness to the robot's movements.

### **8.3 Code Parser**

Furthermore, the addition of the Code Parser has significantly improved the user experience. Drawing inspiration from G-Code Parsers used in CNC Machines, the Code Parser allows for versatile programming and control of Kinetic Pace. Users can easily input commands and customize the robot's behavior, enhancing its adaptability for various applications.

### **8.4 Improved Controller**

The improvements made to the controller have played a vital role in enhancing the robot's overall performance. Through careful analysis and leveraging concepts such as TCOT, the Poincare Map, and the Nominal State, the controller has become more robust by 17%. This improvement ensures better stability, control and energy efficiency during the robot's locomotion. Additionally, the optimized Nominal State has contributed to a remarkable 73% decrease in the total cost of transport, further improving the robot's efficiency.

### **8.5 Controller Implementation on Physical Robot in Subsequent Project**

With the successful achievement of the project's objectives, I am confident that the outcomes will have a profound impact on subsequent projects. The improved controller design and optimized Nominal State lay the foundation for the implementation of advanced algorithms on the physical robot, promising even greater stability, energy efficiency and performance in future iterations.

# Chapter 9

## Applications and Future Scope

Bipedal robots have a wide range of potential applications in various fields, including robotics, manufacturing, and military operations. In this section, I will discuss some of the potential applications of the bipedal robot, Kinetic Pace, as well as areas for further research and examples of bipedal robots in other applications.

### 9.1 Potential Applications

1. Robotics: Kinetic Pace can be used in robotics applications that require a robot to navigate uneven or unpredictable terrain, such as disaster response or exploration missions. Its natural dynamic movements would allow it to move efficiently over challenging surfaces while conserving energy.
2. Manufacturing: Kinetic Pace can also be used in manufacturing, where it can be programmed to perform repetitive tasks that require mobility, such as moving parts or materials between workstations.
3. Military: Kinetic Pace can potentially be used in military operations to assist soldiers in carrying heavy loads over rough terrain, or in reconnaissance missions where a stealthy and agile robot is required.

## 9.2 Further Research

1. Efficiency: One area for further research is improving the efficiency of Kinetic Pace's movement. This could involve optimizing the control algorithms further, to reduce energy consumption or exploring different gait patterns that could improve its overall efficiency.
2. Range of motion: Expanding Kinetic Pace's range of motion may involve exploring different joint designs that would allow the robot to move in more varied and complex ways, such as climbing stairs or crawling through tight spaces.
3. Other research: For example, researchers are exploring the use of bipedal robots for search and rescue missions, where they could navigate difficult terrain to reach victims in need of assistance.

## 9.3 Conclusion

In conclusion, Kinetic Pace has a wide range of potential applications in various fields. There are many areas for further research that could improve its functionality and expand its capabilities. By exploring other examples of bipedal robots in other applications, we gain valuable insights that complement my work on Kinetic Pace.



# Chapter 10

## Visual Documentation

The “Visual Documentation” chapter provides a visual representation of the physical robot designed and constructed for this project. Through a series of photographs, readers can gain a comprehensive understanding of the robot’s mechanical design, components, and overall structure.

It not only presents the physical aspects of the robot but also serves as a visual chronicle of my journey of development. It showcases the progress made from the initial makeshift setup at home to the refined and completed design. These visual representations capture the evolution of the project, highlighting the determination, resourcefulness, and commitment that went into bringing the robot to life.



Figure 10.1: Initial Prototype Initial Model CAD Model View 1

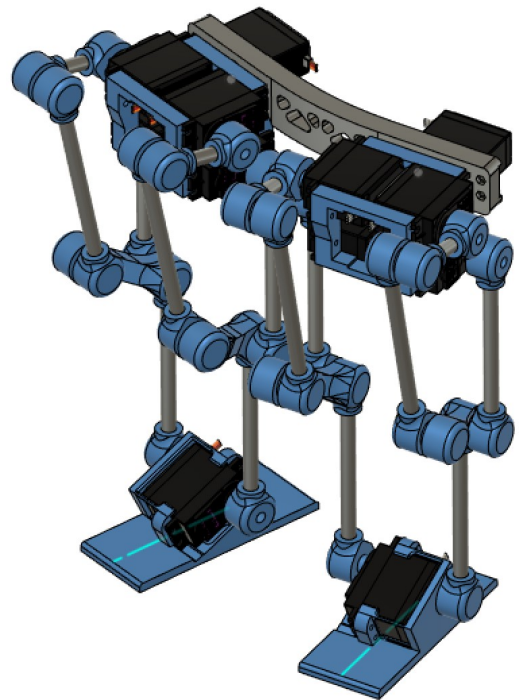


Figure 10.2: Initial Prototype Initial Model CAD Model View 2

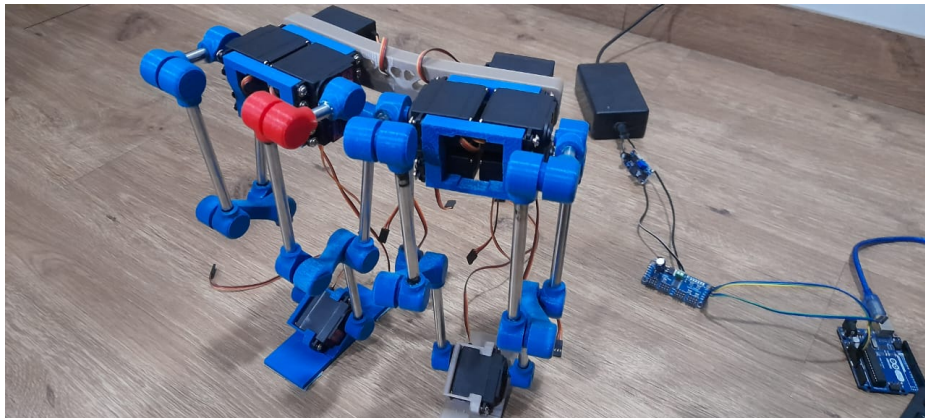


Figure 10.3: Initial Prototype Physical Model View 1

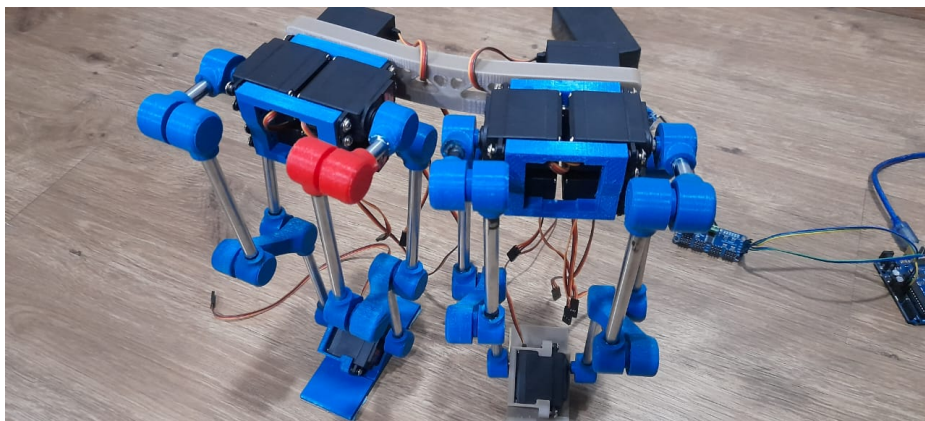


Figure 10.4: Initial Prototype Physical Model View 2

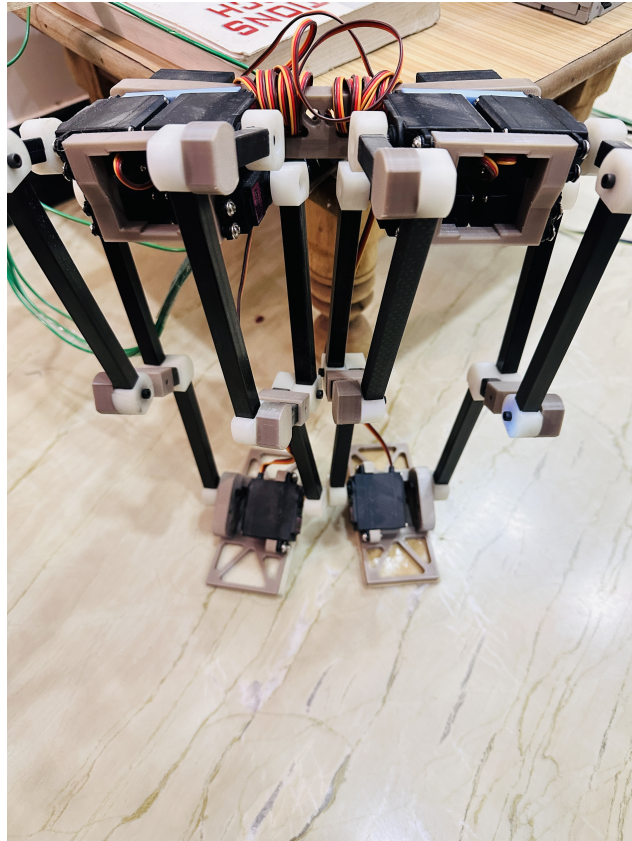


Figure 10.5: Physical Model of Kinetic Pace for Final Year Project



Figure 10.6: Physical Model of Kinetic Pace for Final Year Project



Figure 10.7: Guide Rod for Final Year Project Physical Model



Figure 10.8: Guide Rod for Final Year Project Physical Model



Figure 10.9: Meticulous Wire Management with SMPS

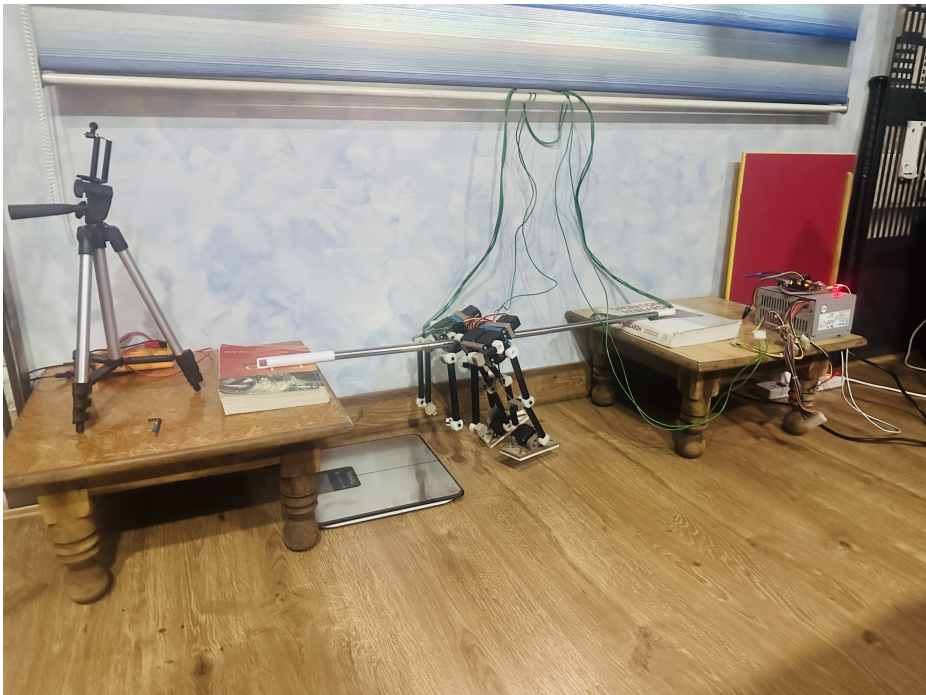


Figure 10.10: Meticulous Wire Management with SMPS

# References

- [1] Kevin G. Gim, Joohyung Kim, and Katsu Yamane, *Design and Fabrication of a Bipedal Robot using Serial-Parallel Hybrid Leg Mechanism*, Disney Research, (October 1, 2018). <https://la.disneyresearch.com/publication/design-and-fabrication-of-a-bipedal-robot-using-serial-parallel-hybrid-leg-mechanism/>
  
- [2] Petr Victorovich Zaytsev *Using Controllability Of Simple Models To Generate Maximally Robust Walking-robot Controllers*, Cornell University, (January 2015). [http://ruina.tam.cornell.edu/research/topics/locomotion\\_and\\_robotics/TikTok/Zaytsev2015\\_Dissertation.pdf](http://ruina.tam.cornell.edu/research/topics/locomotion_and_robotics/TikTok/Zaytsev2015_Dissertation.pdf)