Mechanical Design, Analysis and Simulation of a Quadruped Robot

An Undergraduate Honors Thesis
Submitted to the Department of Mechanical Engineering
The Ohio State University
In Partial Fulfillment of the Requirements
For Graduation with Distinction in Mechanical Engineering

Zhixin Li
April 2019

Advisor: Haijun Su, Ph.D.
Committee: Ayonga Hereid, Ph.D.

Abstract

Wheeled and legged robots are two essential kinds of locomotion robots that have different limitations on speed and mobility. The wheeled robot can move faster on the well-structured, while legged robots which have high mobility can move more smoothly on the rough terrains. The previous researcher in the Design, innovation and simulation lab at the Ohio State University have designed and prototyped a transformative robotic leg that combined these two features to obtain both high moving speeds on the flat surface and high mobility on irregular terrains. However, the previous design of having the servo motor directly connect the lower and upper limb failed to consider the motor weight can cause great undesired inertia to another motor at the shoulder, which can lead to waste of power, high expense, and increase of the potential risk of breaking the motors. This research has been focusing on designing the quadruped robot legs using the 5-bar linkage mechanism to reduce the inertia and simulating the robot model under different scenarios, such as walking, trotting, pacing, and bounding. The final design was capable to walk and has small body size (0.4m*0.342m*0.24m). The components used in this design were either commercially available or 3D printed resulting in a low-cost and replicable platform so that it will be good for commercial and educational use.

Acknowledgement

I would first like to thank my advisor, Dr. Haijun Su. He provided a very clear guide on how to complete each step of the project. Also, he was always willing to explain and teach the knowledge that I don't understand. While doing this research project I learned a great deal from him.

Also, I would like to thank Dr. Ayonga Hereid. He provided a lot of ideas and suggestions on the project.

Lastly, I would like to thank John Ouyang and Noah Limes. They listened to my research progress presentation every week and asked many valuable questions which helped me to learn the part that I did not explain very well and be better prepared for my oral defense.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Wheeled Robots vs. Legged Robots

In the past few years, people have been devoted to designing different kinds of robots to make life easier, faster and safer. One of the most critical function that the robots were hoped to achieve is locomotion which allows the robots to carry cargos to and work at places that human not be able to reach. The two major locomotion robots on grounds are wheeled robots and legged robots.

Wheeled robots such as vehicles and motorcycles have many advantages such as simpler structure, easier to control and more robust. They can achieve high speed and transport efficiently on flat terrains. Wheeled robots also become the most common mobile robot because of their high maneuverability and stability. The limitation of wheeled robots is their low mobility and only being able to navigate on relative well-structured environment (Bai et al., 2018). However, in some situations, robots are required to across obstacles and gaps. Compare with wheeled robots, legged robots are more agile and can move on more kinds of terrains, such as mountains, hills, swaps, and valleys, but they also have many limitations. Designing a legged robot have more stringent requirements on choosing actuators and batteries, designing platform and complex control system and simulating different scenarios.

1.2 Quadruped Robots

There are three major types of legged robot, biped robot, quadruped robot, and hexapod robot. Considering the robots with the same robotic leg design, the robots with more legs can have larger load capacity, so the load capacity of the quadruped robots is the midpoint in these three major types of robot. Therefore, the quadruped robot can be a good choice with a certain load, since the control system of the quadruped robots is relatively simple. Among these three

types of robots biped robots are the most challenging from stability view (Jin et al., 2010). Compare with biped robot, hexapod and quadruped robot have built-in static stability. The coordination and synchronization of the legs of the quadruped robot are simpler than the hexapod robot. Therefore, in these three major types of legged robot, the morphology of the quadruped robot is the least complex, which results in a less complicated control system.

1.3 Previous Work

The previous student worked in the Design, innovation and simulation lab at the Ohio State University, Yupeng Cheng, designed and prototyped a transformable quadruped robotic leg (2018) which can successfully transform between leg and wheel. The transformable robot (wheel-legged robot), which combined the advantages of the high mobility and agility of the legged robot and the high maneuverability and stability of the wheeled robot, which can be suitable in conditions where the robot need to navigate on a large amount of well-structured ground and a small number of obstacles. The robot can transform between two forms to achieve more efficient use of time and power on different terrains.

The final designed robot has degree a of freedom on the roll, pitch, and yaw. There are three actuators on each leg used for turning the forearm, the main arm and the shoulder which works like the three major joints on the leg of typical four-legged animals, such as dogs (Figure 1).

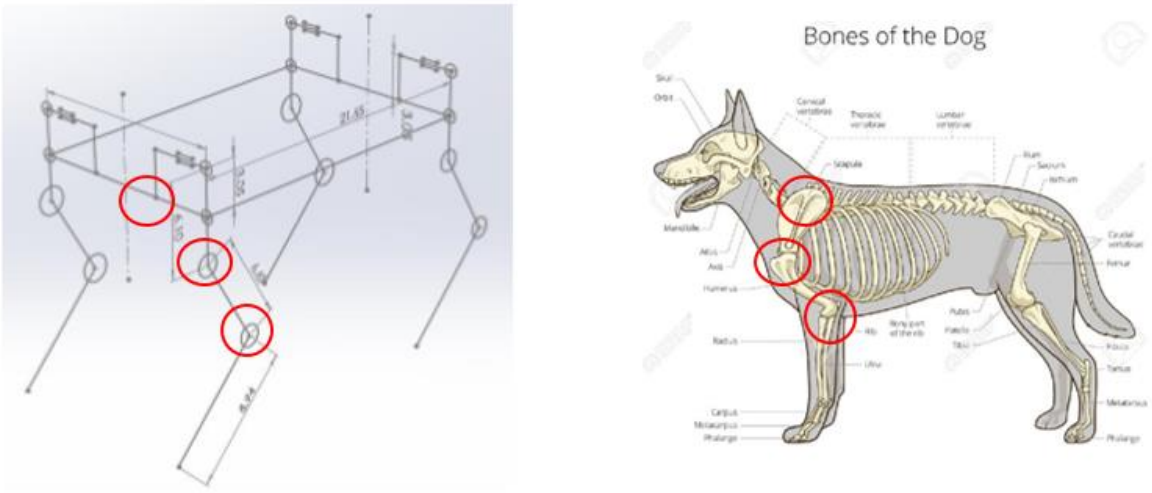


Figure 1: Comparison of the position of three actuators on the robot leg and the joints on the dog leg.

In this design, the servo motor at the elbow helps to manage the transformation process. When the robot needs to move on the rough terrain, the wheel on the leg can be disabled and the robot can navigate using legs. When there is a large amount of flat surface, the leg can be folded and reveal the wheel to move (Figure 2).

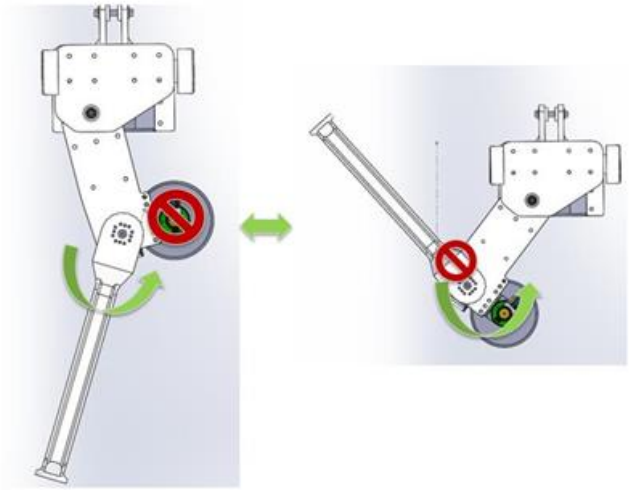


Figure 2: Transformation process of transformable quadruped robot.

1.4 Objectives of this research

There is one remained problem of the transformable robotics leg designed by previous students in the DISL lab. Assuming a situation of lifting the thigh to across some obstacles (Figure 3), the weight of the servo motor at the elbow that rotates the forearm can create a large inertia to the motor at the shoulder, which did not conform to minimize mass and inertia principle of design a leg (McKenzie, 2012). Also, the high inertia caused by the weight of the motor can lead to high cost of buying the motor and increase the potential risk of breaking the motors.

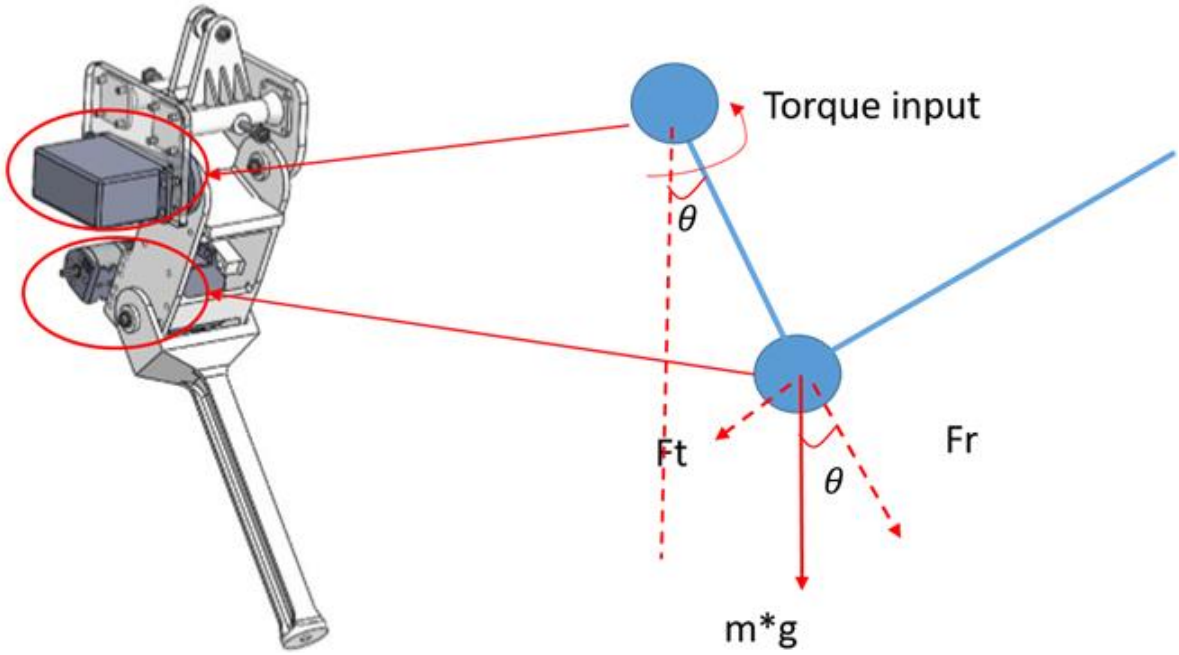


Figure 3: Complete assembly and conceptual figure of the transformable robotics leg.

The required input torque to lift the leg can be written as:

$$\tau_{in} = \tau_{out} + rmgsin\theta \quad (1)$$

Where

τ_{out}=Output torque, torque required to lift up the linkages

r=Length of the thigh

m=Mass of the motor at the elbow

g=gravitational constant, 9.8m/s^2

θ=angle between vertical line and the thigh

Therefore, the main purpose of this research is redesigning the leg of the quadruped robot using a different mechanism to reduce the inertia caused by the weight of motor at the elbow to the motor at the shoulder. More specific objectives are finding a mechanism that can be used to reduce the inertia caused by the weight of the motor, redesigning the quadruped robot leg using that mechanism, creating the Solidworks and Simscape Multibody models of the whole robot and simulating the scenarios of the robot walking, trotting, bounding, and pacing.

1.5 Overview of the thesis

This thesis focuses on discussing the design process of the platform, the mechanism used to achieve the design requirement, how the design been analyzed and simulated, and the final results of the simulations. Chapter 2 discusses the platform design of the quadruped robot. Chapter 3 discusses the analysis methods of the design. Chapter 4 focuses on discussing the trajectory planning for the quadruped robot. Chapter 5 discusses the Simscape Multibody of the design. Chapter 6 talks about the simulation scenarios and results of the design. Lastly, chapter 7 concludes the research progress and future works. The standard 3 views of the Solidworks model of the platform components, the Simscape multibody block diagrams, and the MATLAB codes are included in the appendix as references.

Chapter 2: Platform Design

2.1 Design Constraints

The design constraints of the quadruped robot are as follows:

- The overall maximum dimension should be 400mm long, 300mm wide and 240mm high.
- The motors should be chosen from ROBOTIS DYNAMIXEL MX series.
- Considering the compatibility between the motors and the linkages on the leg, the brackets sold by the ROBOTIS company should be used to directly connect the motor shafts.
- The designed robot should have 3 degrees of freedom for each leg and a total of 12 degrees of freedom for 4 legs. The whole robot should have a degree of freedom on the roll, yaw and pitch axis.
- Conceptually, the upper limb should be able to swing about -60 degree to +60 degree to the vertical line and the lower limb should be able to swing about -30 degree to +90 degree relative to the horizontal line (Figure 4).

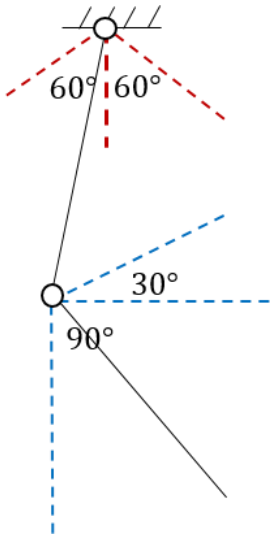


Figure 4: Workspace constrain of the limb.

2.2 5-bar Linkage Mechanism

The 5-bar linkage mechanism, the closed chain of 5 linkages, has two degrees of freedom which is the same as the 2 linkages open chain of the previous transformable leg. One motor can directly drive the upper limb. Another motor can indirectly drive the lower (Figure 5). Also, designing the robotics leg using the 5-bar linkage mechanism can reduce the inertia caused by the weight of the motor, since the two actuators can be put close together on the same ground axis.

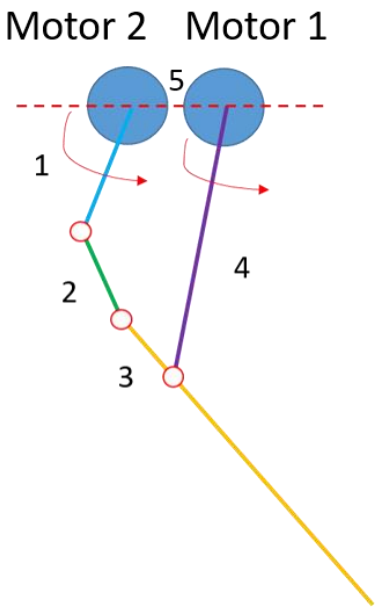


Figure 5: Simplified five bar linkage robotic leg.

2.3 Arrangement of the Motors

The arrangement of the motors was determined using Solidworks and based on the dimension constraint of the robot. The model of the largest motor in the DROBOTIS DYNAMIXEL MX series, the MX64R was downloaded from the DYNAMIXEL Company and used to determine the motor arrangement as well. Placing two motors along lateral axis was tried

(Figure 6). However, because of the limitation of 300mm wide of the robot, there is not enough space for the third motor.

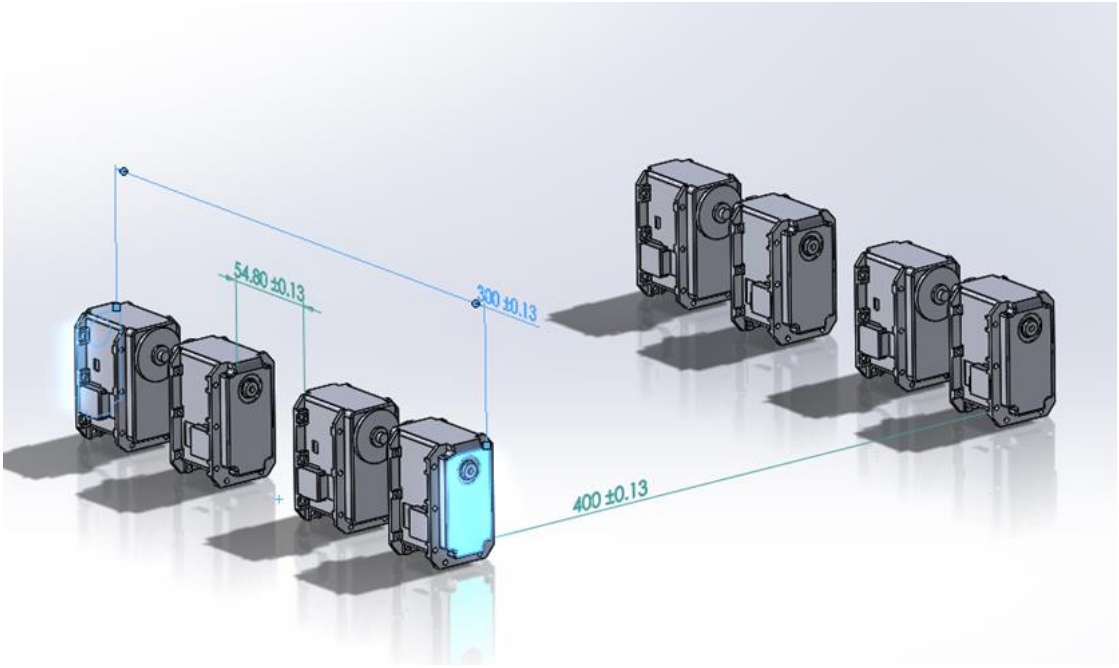


Figure 6: Arrangement of the 8 leg motors along the lateral axis.

After that, placing two motors along longitudinal axis was tried (Figure 7). There is enough space (199mm) between each limb to place the third motor to achieve the degree of freedom on the roll axis. Therefore, the arrangement of placing two motors along longitudinal axis can be used to design the 5-bar linkage leg.

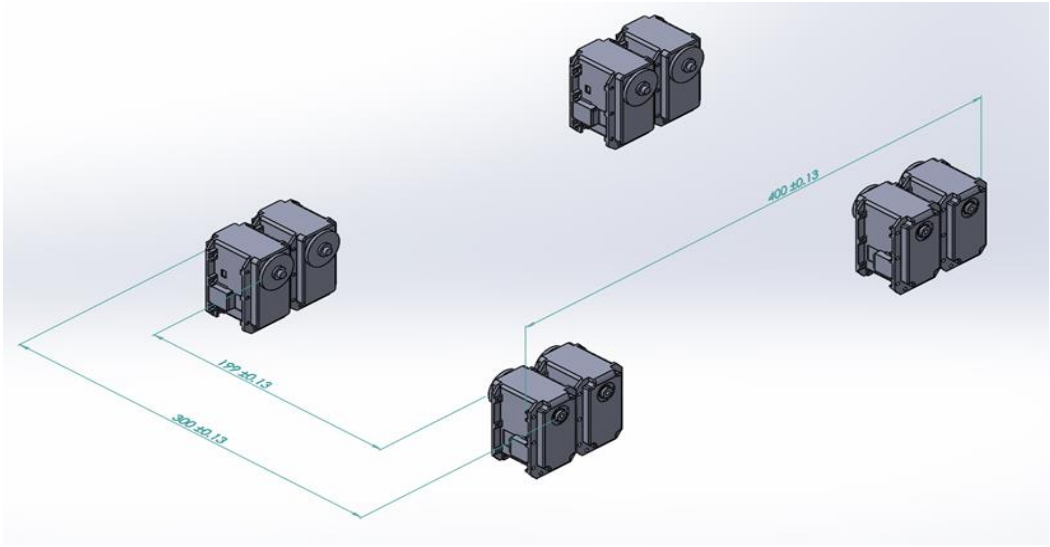


Figure 7: Arrangement of the 8 leg motors along the longitudinal axis.

Then, the same arrangement (Figure 8) as the previous design of placing the third motor horizontally was checked using the SOLIDWORKS. However, there is limited space (46.50mm) for the linkages and thickness of the limb.

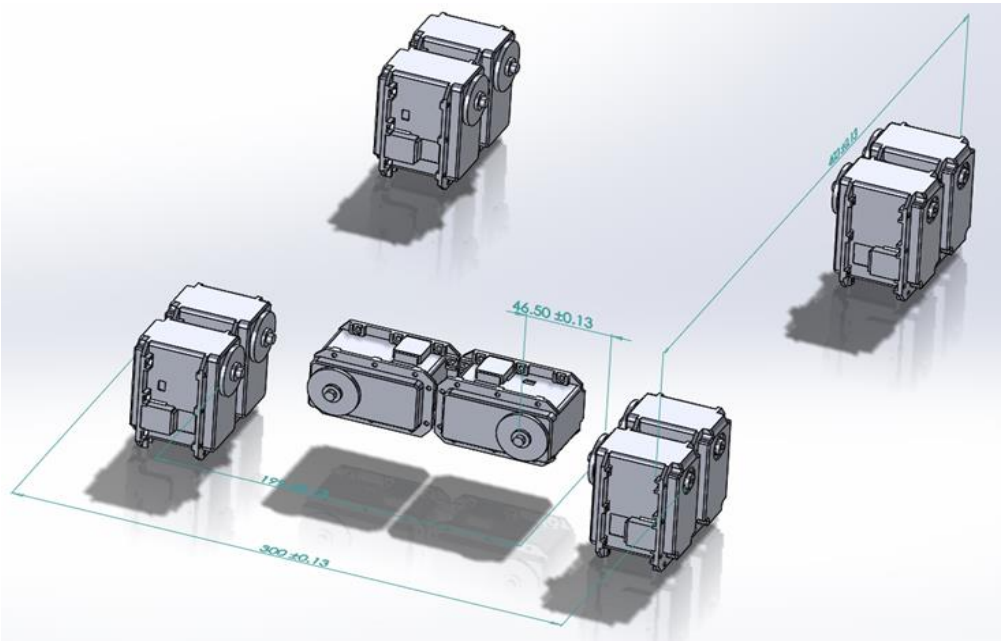


Figure 8: Arrangement of the 8 leg motors along the longitudinal axis and the shoulder motors horizontally.

The final arrangement of the motors is that place two motors along the longitudinal axis for the 5-bar linkage mechanism and the third motor vertically aside of the other two motors. Also, a conceptual SOLIDWORKS model of the quadruped robot (Figure 9) was created to check the scale of the model. According to the conceptual model, there is enough space for the linkages at the shoulder and the main body, so this arrangment can be used to do further design.

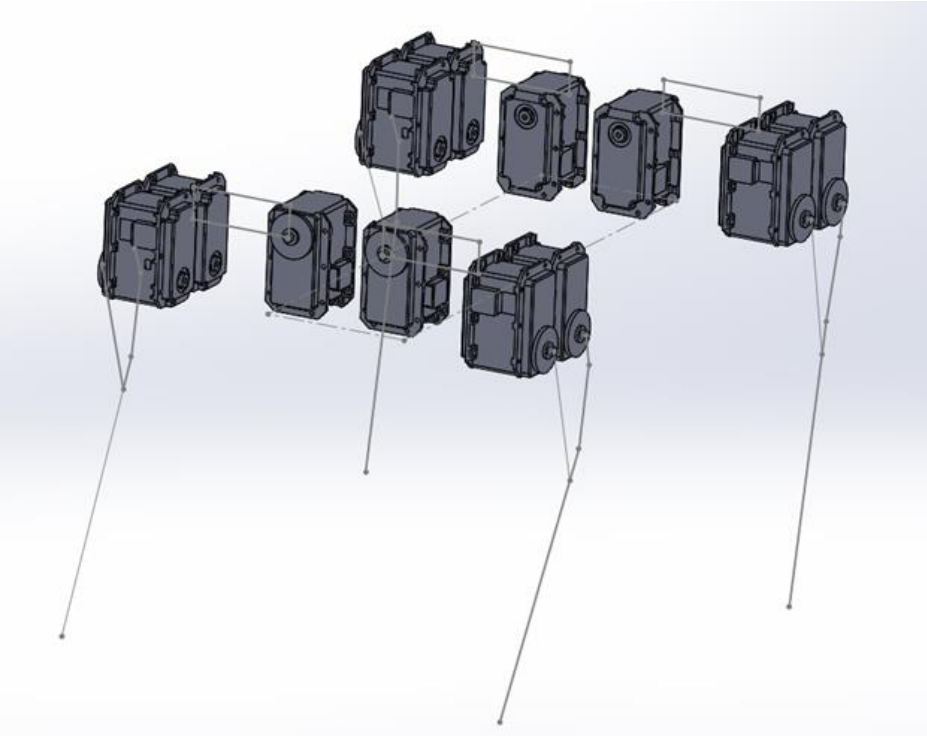


Figure 9: Conceptual Solidworks model of the quadruped robot.

2.4 Mechanical Design of the Platform

The length of each bar was first determined using Solidworks sketch. The constraints of linkage length were first the ground bar larger or equal to 40.2 mm, which is the width of two motor, the length of bar 1 longer or equal to 24.08 mm, which is the minimum length of the bracket (Figure 10) modified from the commercial bracket FR05-H101 without collision with the bottom of the motor. In addition, the length of each should also satisfy the requirement that,

without considering the collision, the upper limb should be able to swing about -60 degree to +60 degree to the vertical line and the lower limb should be able to swing about -30 degree to +90 degree relative to the horizontal line. To decide the specific length of the linkages, a closed frame with 5 binary links was draw using Solidworks sketches that the ground bar was 60.2 mm and fixed in order to leave enough space between the two motors for bar 1 to turn. Then multiple sets of the length of the other 4 links were tried to achieve the constraints of the workspace. Table x shows the specific length of each link for this design.

Table 1: Link length of the robotic limb

Link 1	**Link 2**	**Link 3**	**Link 4**	**Link 5**
32mm	60mm	40.11mm	89.07mm	60.2mm

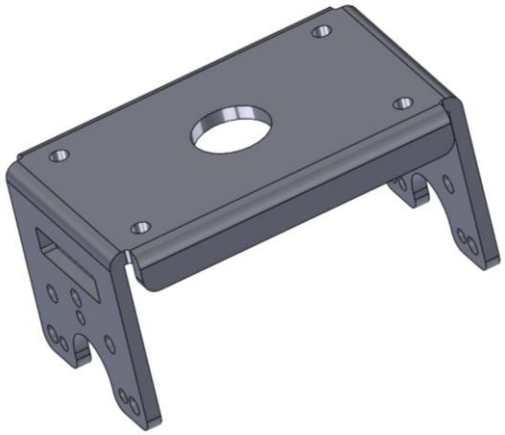


Figure 10: Solidworks model of the modified bracket.

The mechanical model of the quadruped robot platform (Figure 11) has a dimension of 400 mm long, 342 mm wide and 240 mm high, which exceed the 300 mm wide requirement in order to leave enough space for the main body of the robot.

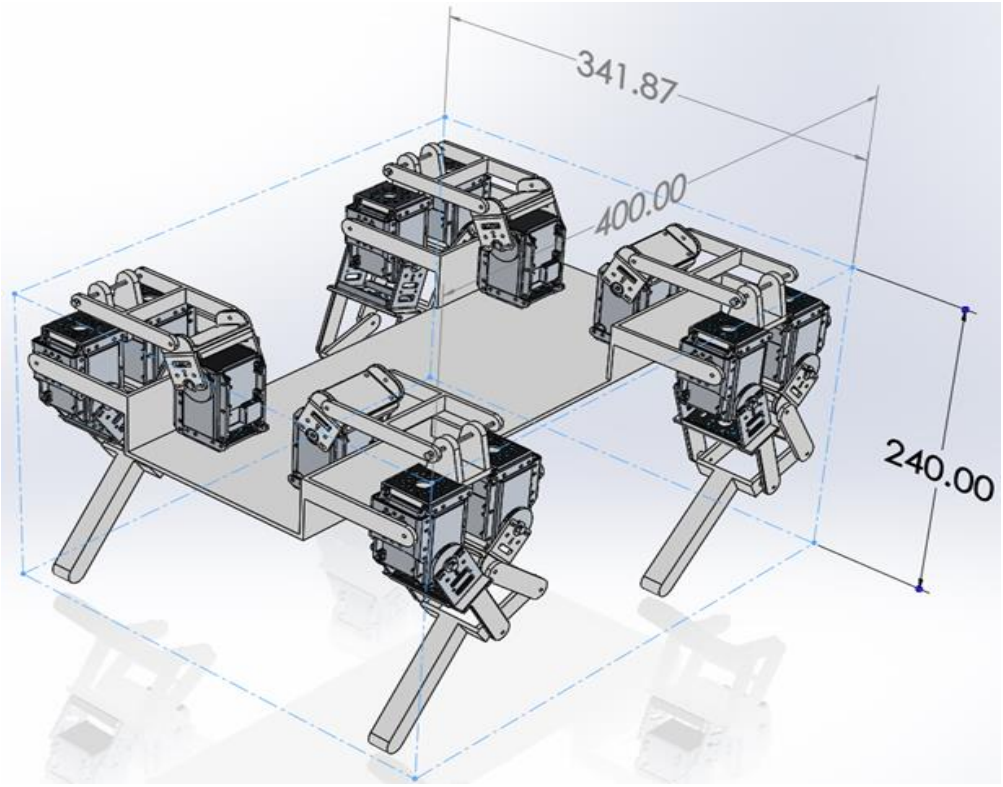


Figure 11: Final Solidworks model of the quadruped robot platform.

Chapter 3 Kinematics Analysis

3.1 Forward kinematics of five-bar linkage

Forward kinematics is the process of using the forward kinematics equation to calculate the end-effector of the robot given the joint parameters. In this case, the forward kinematics equation can be used to find out the foot tip position of the limb, if the angles of the two motors are known. Figure 12 below shows the model of one robotic leg and Figure 13 shows the schematic of the upside down 5-bar linkage robotic leg. The reason for creating the configuration of the leg upside down is that the coordination of the foot tip can always be positive, which was easy to analyze. In the schematic figure of the 5-bar linkage limb, l_1, l_2, l_3, l_4, l_5 and l are the length of the links and θ_1, θ_2, θ_3, θ_4, θ_5 and θ are the angle of the links. The final forward kinematic equations have two inputs, θ_4 and θ_1, which are the angles of motor 1 and motor 2, and two outputs, the x and y coordinates of the foot tip.

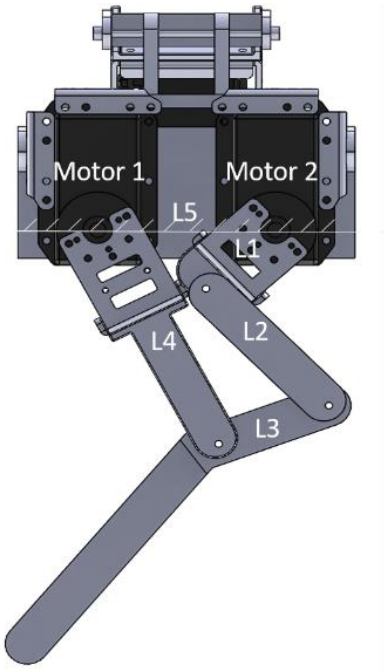


Figure 12: Solidworks model of one leg.

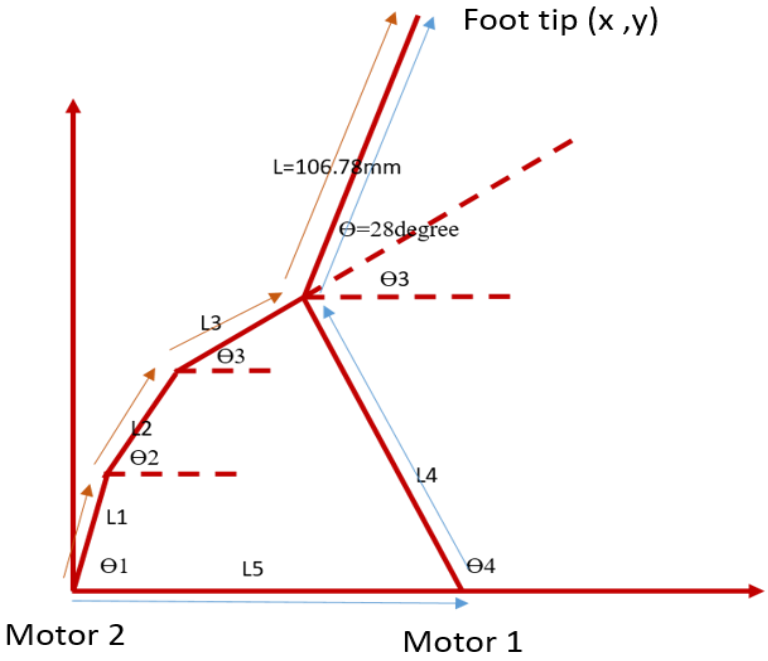


Figure 13: Schematic figure of the 5-bar linkage limb.

The position of foot tip can be found by adding up all orange vectors or all blue vector. The foot tip position equations of adding up the blue vectors from motor 2 to the foot tip going through l_1, l_2, l_3 and l, in x and y direction:

$$l_1 * cos\theta_1 + l_2 * cos\theta_2 + l_3 * cos\theta_3 + l * \cos(\theta + \theta_3) = x \quad (2)$$

$$l_1 * sin\theta_1 + l_2 * sin\theta_2 + l_3 * sin\theta_3 + l * \sin(\theta + \theta_3) = y \quad (3)$$

The foot tip position equations of adding up the blue vectors from motor 2 to the foot tip going through l_5, l_4 and l, in x and y direction:

$$l_5 + l_4 * \cos\theta_4 + l * \cos(\theta + \theta_3) = x \quad (4)$$

$$l_4 * sin\theta_4 + l * \sin(\theta + \theta_3) = y \quad (5)$$

Rearrange equations (2)-(5) and eliminate the unknown variables θ_2 and $\theta_3$to obtain the forward kinematic equations such that x and y coordinates of the foot tip are the outputs and the actuators' angle, θ_4 and θ_1, are the inputs:

$$x = l_5 + l_4 * \cos\theta_4 + l * \cos(\theta + 2\arctan(x_1)) \quad (5)$$

$$y = l_4 * sin\theta_4 + l * \sin(\theta + 2\arctan(x_1)) \quad (6)$$

Where,

$$x_1 = \frac{D - \sqrt{D^2 - (A+B+C)(A+B-C)}}{A+B+C}$$

$$A = l_1^2 + l_3^2 + l_4^2 + l_5^2 - l_2^2$$

$$B = 2l_5 l_4 cos\theta_4 - 2l_1 l_4 \cos(\theta_4 - \theta_1) - 2l_5 l_1 cos\theta_1$$

$$C = 2l_3 l_4 cos\theta_4 + 2l_5 l_3 - 2l_3 l_1 cos\theta_1$$

$$D = 2l_3 l_4 sin\theta_4 - 2l_3 l_1 sin\theta_1$$

l_1= length of link 1, 32.14 mm.

l_2= length of link 2, 60 mm.

l_3= length of link 3, 40.11 mm.

l_4= length of link 4, 89.2 mm.

l_5= length of link 5, 60.2 mm.

l= length of extended link 1, 106.78 mm.

θ= Angle between link 3 and extended link l, 28°.

3.2 Inverse kinematics equations of 5-bar linkage

Inverse kinematics is the process using the inverse kinematics to calculate the joint parameters given the end-effector of the robot. Rearrange equations (2)-(5) and eliminate the

unknown variables θ_2 and $\theta_3$to obtain the inverse kinematic equations such that the actuators' angle, θ_4 and θ_1, are the outputs and the x and y coordinates of the foot tip are the inputs:

$$\theta_4 = \arctan\left(\frac{y - l*\sin(\theta + 2*\arctan(a) - \theta)}{x - l*\cos(\theta + 2*\arctan(a) - \theta) - l_5}\right) \quad (7)$$

$$\theta_1 = 2 * \arctan(b) \quad (8)$$

Where,

$$a = \frac{K - \sqrt{K^2 - (M^2 - N^2)}}{M + N}$$

$$M = y^2 + (x - l_5)^2 + l^2 - l_4^2$$

$$K = 2 * y * l$$

$$N = 2 * (x - l_5) * l$$

$$b = \frac{P - \sqrt{P^2 - (O^2 - Q^2)}}{O + Q}$$

$$O = (x - l_3 * cos\theta_3 - l * \cos(\theta + \theta_3))^2 + (y - l_3 * sin\theta_3 - l * \sin(\theta + \theta_3))^2 + l_1^2 - l_2^2$$

$$P = 2 * l_1 * (y - l_3 * sin\theta_3 - l * \sin(\theta + \theta_3))$$

$$Q = 2 * l_1 * (x - l_3 * cos\theta_3 - l * \cos(\theta + \theta_3))$$

3.3. Forward velocity kinematic of 5-bar linkage

Forward velocity kinematics equations can be used to calculate the velocity of the foot tip in x and y direction given the angular velocities of the two motors and angles of all links. To derive the forward velocity kinematic equations of the robotic leg first is to differentiate equation (2)-(5). The differentiated velocity equations:

$$-l_1 * sin\theta_1 * \dot{\theta}_1 - l_2 * sin\theta_2 * \dot{\theta}_2 - (l_3 * sin\theta_3 + l * \sin(\theta + \theta_3)) * \dot{\theta}_3 = \dot{x} \quad (9)$$

$$l_1 * cos\theta_1 * \dot{\theta}_1 + l_2 * cos\theta_2 * \dot{\theta}_2 + (l_3 * cos\theta_3 + l * \cos(\theta + \theta_3)) * \dot{\theta}_3 = \dot{y} \quad (10)$$

$$-l_4 * sin\theta_4 * \dot{\theta}_4 - l * \sin(\theta + \theta_3) * \dot{\theta}_3 = \dot{x} \quad (11)$$

$$l_4 * cos\theta_4 * \dot{\theta}_4 + l * \cos(\theta + \theta_3) * \dot{\theta}_3 = \dot{y} \quad (12)$$

Rearrange equation (9)-(12) and eliminate the unknown variables $\dot{\theta}_2$ and $\dot{\theta}_3$to obtain the forward velocity kinematic equations such that the foot tip velocity, $\dot{x}$ and $\dot{y}$, are the outputs and the actuators' angular velocity, $\dot{\theta}_4$ and $\dot{\theta}_1$, are the inputs:

$$\dot{x} = (-\tan(\theta + \theta_3)\,\alpha)\dot{\theta}_1 + (-l_4 sin\theta_4 + \tan(\theta + \theta_3)\,l_4 cos\theta_4 - \tan(\theta + \theta_3)\beta)\dot{\theta}_4 \quad (13)$$

$$\dot{y} = \alpha\dot{\theta}_1 + \beta\dot{\theta}_4 \quad (14)$$

Where,

$$\alpha = \left(\frac{(-l_1 sin\theta_1 + tan\theta_2 l_1 cos\theta_1)}{(tan\theta_2\left(-\frac{l_3 cos\theta_3}{l\cos(\theta + \theta_3)}\right) + \frac{l_3 sin\theta_3}{l\cos(\theta + \theta_3)})}\right)$$

$$\beta = \left(\frac{(-tan\theta_2\left(\frac{l_3 cos\theta_3}{l\cos(\theta + \theta_3)} + 1\right) l_4 cos\theta_4 + l_3 sin\theta_3 l_4 cos\theta_4 + l_4 sin\theta_4)}{(tan\theta_2\left(-\frac{l_3 cos\theta_3}{l\cos(\theta + \theta_3)}\right) + \frac{l_3 sin\theta_3}{l\cos(\theta + \theta_3)})}\right)$$

3.4. Inverse velocity kinematic of 5-bar linkage

Inverse velocity kinematic equations can be used to calculate angular velocities of the two motors given the velocity of the foot tip in x and y direction and angles of all links. Rearrange equation (9)-(12) and eliminate the unknown variables $\dot{\theta}_2$ and $\dot{\theta}_3$ to obtain the inverse velocity kinematic equations such that the actuators' angular velocity, $\dot{\theta}_4$ and $\dot{\theta}_1$, are the outputs and the foot tip velocity, $\dot{x}$ and $\dot{y}$, are the inputs:

$$\dot{\theta_1} = \frac{\dot{x} - \frac{(-l_4 sin\theta_4 + \tan(\theta + \theta_3)\, l_4 cos\theta_4 - \tan(\theta + \theta_3)\beta)}{\beta}\dot{y}}{(-\tan(\theta + \theta_3)\,\alpha) + (l_4 sin\theta_4 + \tan(\theta + \theta_3)\, l_4 cos\theta_4 + \tan(\theta + \theta_3)\beta)\alpha/\beta} \qquad (15)$$

$$\dot{\theta_4} = (\dot{y} - \alpha * \dot{\theta_1})/\beta \qquad (16)$$

Where, α and β are the same equations as the ones under section 3.3.

3.5. The torque equations of the 5-bar linkage

In order to further analyze the actuator torque required to achieve the designed motion, torque equations of the two motors were derived. The derivation was based on the principle of virtue work that assumes there is no power loss on the limb. The power input of the two motors should be equal to the power output at the foot tip. The power conservation equations of the limb in matrix form:

$$[F_x \quad F_y]\begin{Bmatrix}\dot{x}\\ \dot{y}\end{Bmatrix} = [\tau_1 \quad \tau_4]\begin{Bmatrix}\dot{\theta_1}\\ \dot{\theta_4}\end{Bmatrix} \qquad (17)$$

Where,

F_x= The ground force act on the foot tip in x direction, N.

F_y= The ground force act on the foot tip in y direction, N.

$\dot{x}$= The linear velocity of the foot tip in x direction, m/s.

$\dot{y}$= The linear velocity of the foot tip in y direction, m/s.

τ_4= Actuation torque of motor 1, Nm.

τ_1= Actuation torque of motor 2, Nm.

Plugging the inverse velocity kinematic equations, the equation 13 and 14, into the power conservation equations to cancel out the angular velocity of the two motors, $\dot{\theta}_1$and $\dot{\theta}_{41}$, the actuation torque equation was obtained:

$$\tau_1 = F_x * (-\tan(\theta + \theta_3) * \alpha) + F_y * \alpha \quad (18)$$

$$\tau_4 = F_x * (-l_4 * sin\theta_4 + \tan(\theta + \theta_3) * l_4 * cos\theta_4 - \tan(\theta + \theta_3) * \beta) + F_y * \beta \quad (19)$$

Where, α and β are the same equations as the ones under section 3.3.

3.6. Workspace analysis of the robotic leg

To avoid the collision between the components might either snap the linkages or damage the motors, the workspace of the robotic leg needs to be found, before planning the trajectory. In this research, the interference detection function in the Solidworks was used to find the range of motor angles without collision. This function can be found under “Evaluate” in the Solidwork assembly interface. It can help to detect if there is an overlap between the chosen components and the volume of the interference.

To evaluate the workspace of the designed quadruped robotic leg, first, the minimum angular position of motor 1 without collision was found. Then the corresponding minimum and maximum position of motor 2 were measured using the interference detection function. After that, the previous step with 2-degree increment of the angular position of motor 1 were repeated until motor 1 reached its maximum angle. The angular workspace results of the two motors were shown in Figure 14, where the x-axis is the angle of motor 2 and y-axis is the angle of motor 1.

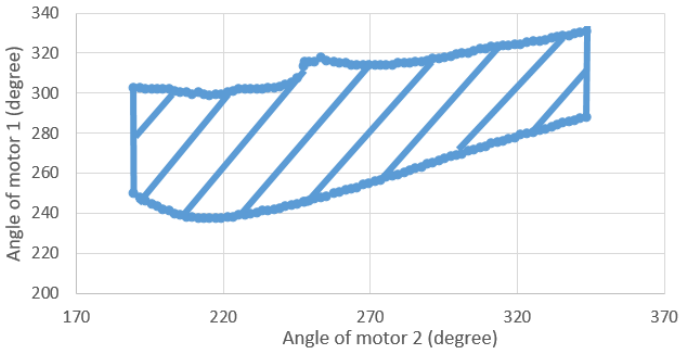


Figure 14: Angular workspace results of the two motors.

The workspace of the limb tip was calculated by plugging the angle workspace of the two motors into the forward kinematic equation. The shaded area in Figure 15 below shows workspace of the limb tip. The position of two motors are also displayed in the figure, where the center of motor 2 was set as the origin.

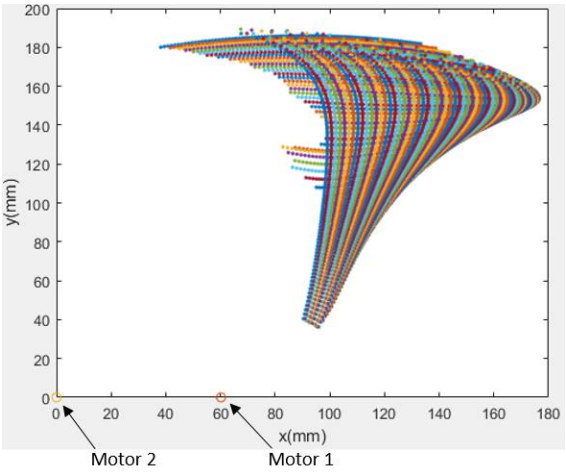


Figure 15: Workspace of the foot tip of the robotic leg.

Chapter 4. Trajectory Planning

4.1. Trajectory Planning of One Leg

The trajectory of each foot tip of the robot move on the flat surface was designed to be a "D" shape, where the straight line represents the stance phase and the curve line represents the flight phase. The foot tip trajectory does not have to be a perfectly smooth "D" shape curve as long as inside the workspace of the limb. However, considering the work required of one gait cycle, the length of the curve should be as short as possible. Also, considering the performance of the motor the sharp turns should be avoided.

The curve below shows the upside-down "D" shape the trajectory of the robot walking, trotting, pacing, and bounding. L1 is the length of the straight portion on the ground. L2 is the maximum length of the trajectory. H1 is the maximum height of the curve, also being considered as the height that the foot tip will lift. H2 is the height from the ground to one point on the side curve. X and y are the distances in the horizontal and vertical direction from the center point of the straight line to the position of the motor 2. The parameters of the trajectory can be changed in the MATLAB to easily modify the size of the gait.

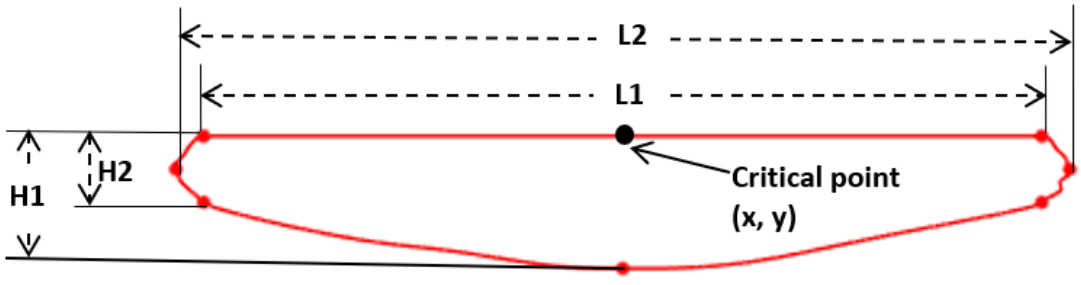


Figure 16: "D" shape trajectory of the robot.

For a quadruped robot, the stance phase takes three fourths of one period and the flight takes one-fourth of one period (Zhou et al., 2016). To achieve this, the straight portion of the trajectory were divided into 36 equally spaced pieces and the curve was divided into 12 pieces, where the time duration of each piece is same so that the total time spent on the straight line will be three times as the time spends on the curve portion. Figure 17 below shows the distribution of the critical points and Table 2 shows the coordinates of the critical points used to fit the trajectory.

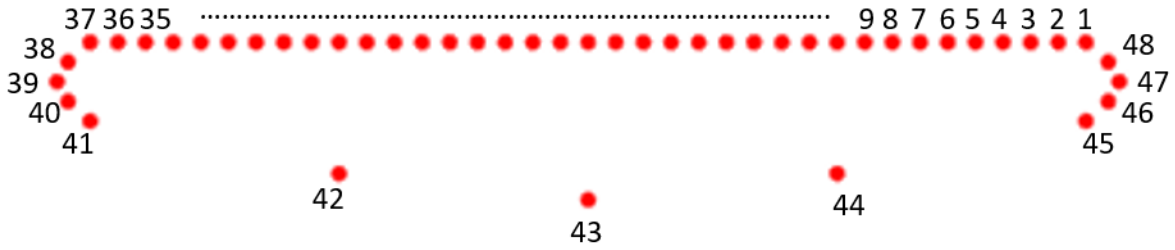


Figure 17: Critical points distribution of the trajectory.

Table 2: Coordination of the trajectory critical points.

$P_1\ (x+\frac{l_1}{2}, y)$	$P_2\ (x+\frac{17l_1}{36}, y)$	$P_3\ (x+\frac{16l_1}{36}, y)$	$P_4\ (x+\frac{15l_1}{36}, y)$
$P_5\ (x+\frac{14l_1}{36}, y)$	$P_6\ (x+\frac{13l_1}{36}, y)$	$P_7\ (x+\frac{12l_1}{36}, y)$	$P_8\ (x+\frac{11l_1}{36}, y)$
$P_9\ (x+\frac{10l_1}{36}, y)$	$P_{10}\ (x+\frac{9l_1}{36}, y)$	$P_{11}\ (x+\frac{8l_1}{36}, y)$	$P_{12}\ (x+\frac{7l_1}{36}, y)$
$P_{13}\ (x+\frac{6l_1}{36}, y)$	$P_{14}\ (x+\frac{5l_1}{36}, y)$	$P_{15}\ (x+\frac{4l_1}{36}, y)$	$P_{16}\ (x+\frac{3l_1}{36}, y)$
$P_{17}\ (x+\frac{2l_1}{36}, y)$	$P_{18}\ (x+\frac{l_1}{36}, y)$	$P_{19}\ (x, y)$	$P_{20}\ (x-\frac{l_1}{36}, y)$
$P_{21}\ (x-\frac{2l_1}{36}, y)$	$P_{22}\ (x-\frac{3l_1}{36}, y)$	$P_{23}\ (x-\frac{4l_1}{36}, y)$	$P_{24}\ (x-\frac{5l_1}{36}, y)$
$P_{25}\ (x-\frac{6l_1}{36}, y)$	$P_{26}\ (x-\frac{7l_1}{36}, y)$	$P_{27}\ (x-\frac{8l_1}{36}, y)$	$P_{28}\ (x-\frac{9l_1}{36}, y)$
$P_{29}\ (x-\frac{10l_1}{36}, y)$	$P_{30}\ (x-\frac{11l_1}{36}, y)$	$P_{31}\ (x-\frac{12l_1}{36}, y)$	$P_{32}\ (x-\frac{13l_1}{36}, y)$
$P_{33}\ (x-\frac{14l_1}{36}, y)$	$P_{34}\ (x-\frac{15l_1}{36}, y)$	$P_{35}\ (x-\frac{16l_1}{36}, y)$	$P_{36}\ (x-\frac{17l_1}{36}, y)$
$P_{37}\ (x-\frac{l_1}{2}, y)$	$P_{38}\ (x-\frac{l_1}{2}-\frac{l_2-l_1}{3}, y-\frac{h_2}{4})$	$P_{39}\ (x-\frac{l_2}{2}, y-\frac{h_2}{2})$	$P_{40}\ (x-\frac{l_1}{2}-\frac{l_2-l_1}{3}, y-\frac{3h_2}{4})$
$P_{41}\ (x-\frac{l_2}{2}, y-h_2)$	$P_{42}\ (x-\frac{l_1}{4}, y-\frac{5h_1}{6})$	$P_{43}\ (x, y-h_1)$	$P_{44}\ (x+\frac{l_1}{4}, y-\frac{5h_1}{6})$
$P_{45}\ (x+\frac{l_1}{2}, y-h_2)$	$P_{46}\ (x+\frac{l_1}{2}+\frac{l_2-l_1}{3}, y-\frac{3h_2}{4})$	$P_{47}\ (x+\frac{l_2}{2}, y-\frac{h_2}{2})$	$P_{48}\ (x+\frac{l_1}{2}+\frac{l_2-l_1}{3}, y-\frac{h_2}{4})$

Fifth order polynomial was used to fit the trajectory, so the velocity and acceleration of the curve are continuous (Biagiotti and Melchiorri, 2010). Also, the values of the velocity and acceleration at the critical point can be manually assigned. In order to obtain a smooth curve, the vertical velocity at the critical number 43 and 37 was assigned as zero, the overall velocity at critical number 40 and 41 was same and the overall velocity at critical number 46 and 47 was same as well. Figure 18 shows one sample fitted trajectory on the workspace graph. The trajectory is very smooth and passes through all critical points assigned. Also, it falls inside the range of the foot tip workspace of the design quadruped robot, which means the designed trajectory is achievable.

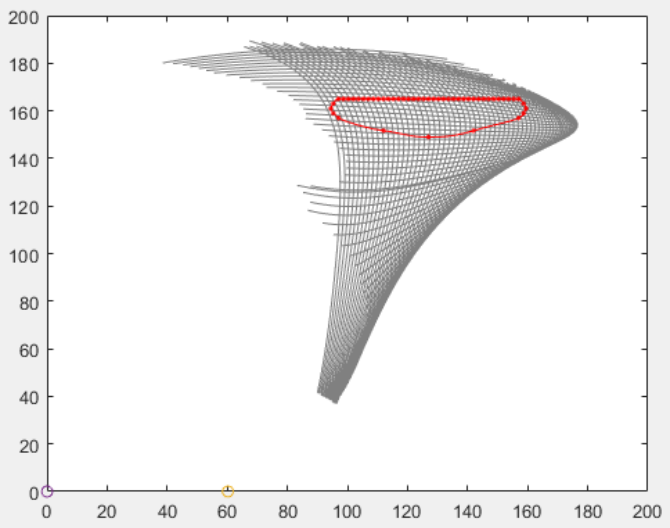


Figure 18: Fitted trajectory on the workspace graph.

4.2. Trajectory Planning of Four Legs

The primary gaits of walk, trot, pace, and bound for the quadruped are showing in Table 3 below (Zhou et al., 2016). The p_i in the table indicates the phase shift with respect to the front left leg, where 2π means one period. Considering the primary gaits, stance phase and flight phase, the walk, trot, pace, and bound gaits were founded. Figure 19 shows the configuration of

the different gaits of the robot. The types of the gait for quadruped can be easily modified in the MATLAB code "motor_angle" by changing the constants that determine the phase shift with respect to the front left leg. To obtain the motor angle signal of different gaits, the inverse kinematic equation was applied.

Table 3. Primary gaits for quadrupeds.

	Front left (p_{LF})	Front right (p_{RF})	Hind left (p_{LH})	Hind right (p_{RH})
Walk	0	π	3π/2	π/2
Trot	0	π	π	0
Pace	0	π	0	π
Bound	0	0	π	π

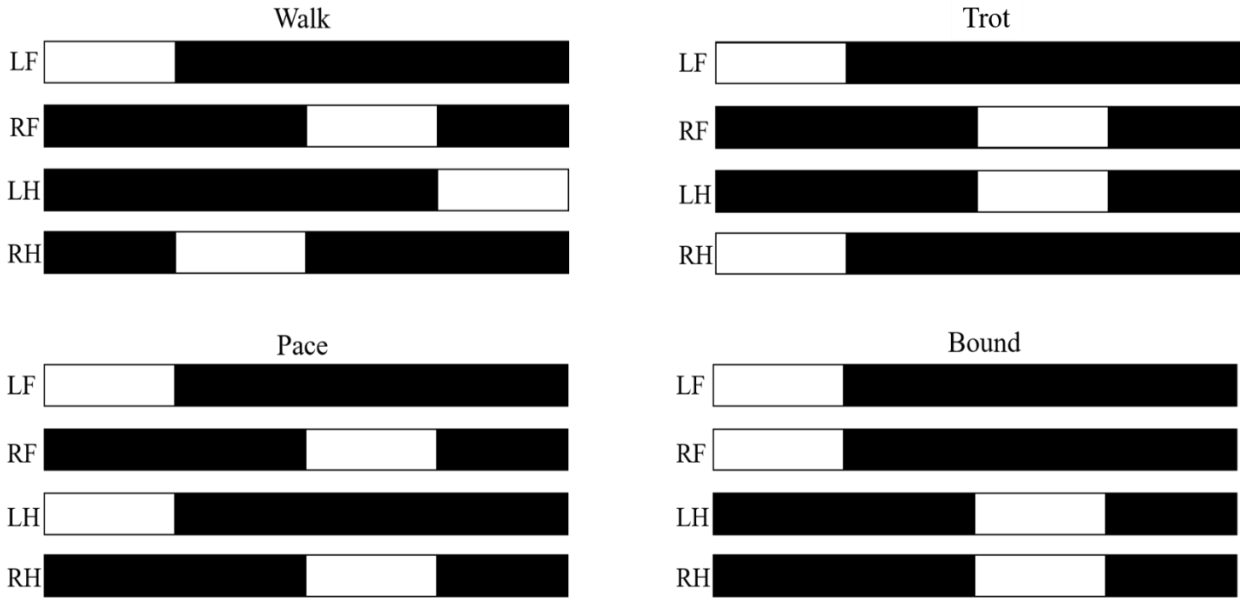


Figure 19: Configurations of the gaits for quadruped robot (The black portion represents the stance phase and the white portion represents the flight phase.)

Chapter 5. Simscape Multibody Model

5.1. Introduce of Simscape Multibody

In order to check the analysis results and simulate different gaits of the robot moving on the flat surface, a Simscape Multibody model was created. The Simscape Multibody can simulate different kinds of 3D mechanical systems that include components such as bodies, joints, constraints, forces, and sensors. Simscape Multibody is a product of Mathwork, so it is an efficient tool to simulate the quadruped robot. The model can be easily parameterize using MATLAB. Also, the control system can be designed and added to the model using Simulink. In addition, it allows the user to integrate the designed mechanical system with hydraulic, electrical and other physical systems. Simscape Multibody accept the .xml file, so the model can be imported from Solidworks. The moment of inertia and center of mass can be calculated automatically by the software based on the geometry designed and material assigned.

5.2. Simscape Multibody Model of the Platform

The .xml file of the quadruped can be obtained using the Solidworks add-in, the "Simscape Multibody Link". In order to obtain a correct Simscape Multibody model from Solidworks, the mates used to assemble the parts should be very careful. Table 4 shows the pairs of mates to get the expected joints ("Mates and Joints- MATLAB & Simulink", 2019). One major type of Simscape joint used in the quadruped robot model is the revolute joint which is equivalent to the combination of the concentric mate of two cylinders and coincides mates of two surface in Solidworks.

Table 4. Mate combinations map of joint blocks.

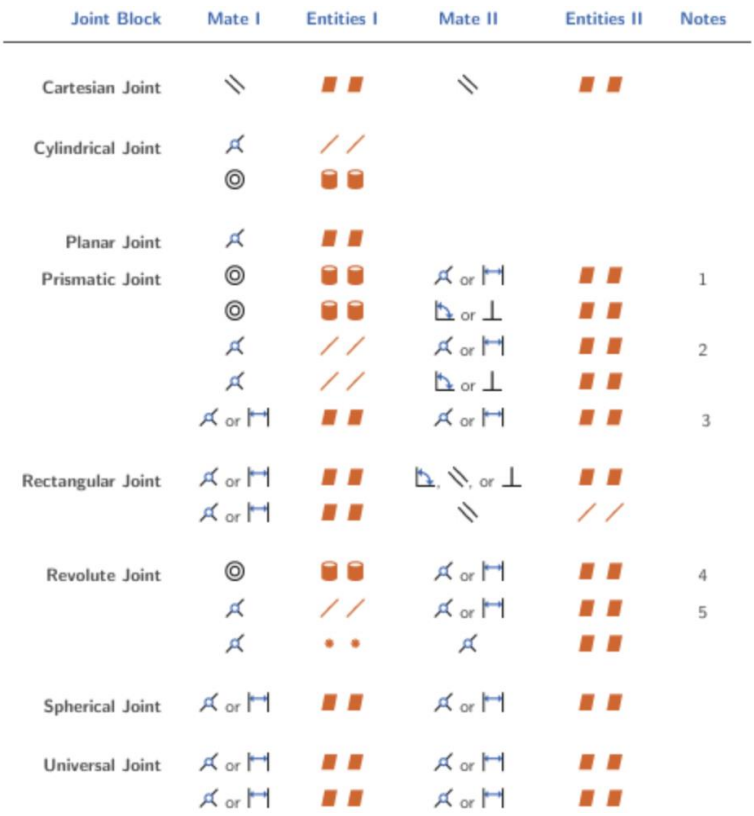


Figure 20 shows the final Simscape Multibody model of the quadruped robot on the ground. The whole model includes a total of 6 subsystems. Those subsystems are the input signals, the quadruped robot model imported from the Solidworks, the solver of the model, the contact force blocks, the scopes that display the sensed actuator angular velocity and the actuation torque required to achieve the designed motion, and the sensors used to measure the leg tip positions and velocities.

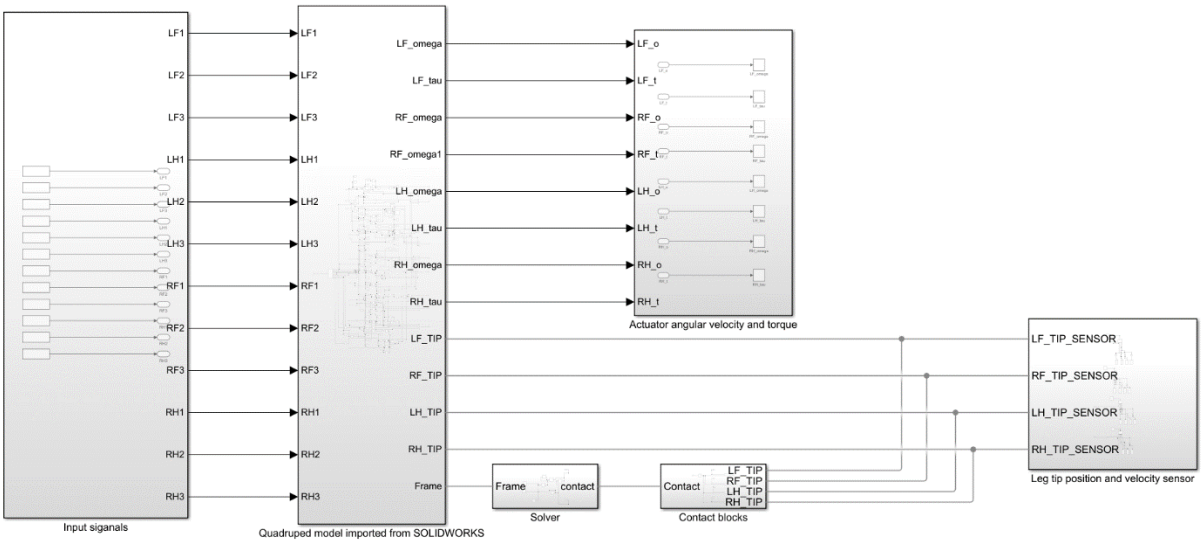


Figure 20. Final Simscape Multibody model used to simulate different scenarios for the quadruped robot.

For this model, the signals inputted are the angles of the motor with respect to time obtained in the trajectory planning step. The signals can be automatically modified in the MATLAB code by changing the shape of the trajectory, the velocity of the walking speed, and the number of periods of the simulation. Also, the signals can be automatically imported from the MATLAB workspace once the Simscape Multibody model is run. Therefore, unless a different control method wants to be used, the input signals subsystem does not need to be changed.

The subsystem of the quadruped model imported form Solidworks contains the body blocks of all components of the robot, the joint blocks that used to determine the constraints between each component, and the transform blocks that used to define the 3-D rigid transformation between two frames. Figure 21 on the next page shows a sample assembly of the quadruped robot model in the mechanic explorer obtained using Simscape Multibody, which looks the same as the Solidworks model.

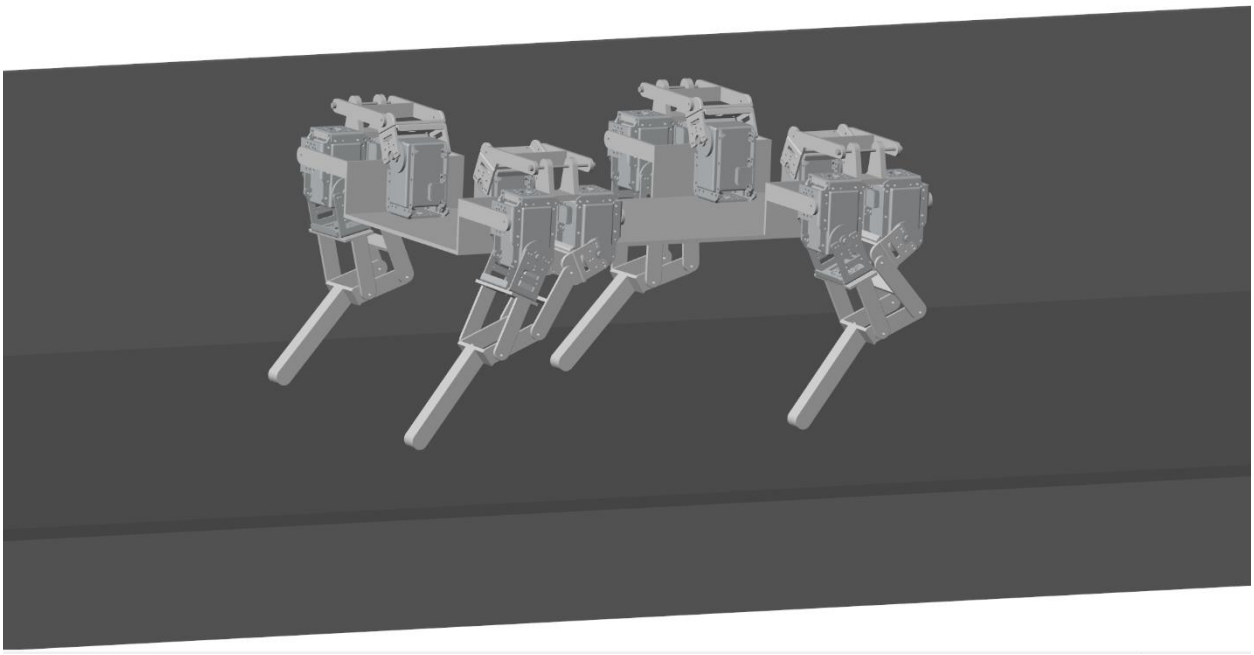


Figure 21. Quadruped robot simulated using Simscape Multibody.

The solver subsystem includes the mechanism configuration block where the gravity constant and the linearization resolution can be defined. It also includes the solver configuration block where the solver settings to use for the simulations can be defined, such as the consistency tolerance, memory budget and filtering time constant. The solver configuration is the main block that calculate the simulation results. The number assigned in these two blocks are defaulted number. In addition, the world frame, the ground blocks, the 6 DOF joint that used to define the distance between the robot body and the ground, and the angle constraint block that used to fix the side movement of the robot are included in the solver subsystem as well.

5.3. Sensors and contact force

The sensors used to measure the actuation torque and the angular position, velocity, and acceleration can be added in the revolute joint block. Figure 22 below the properties of the revolute joint which is the motor 1 of the left front leg. For the actuation of this motor, the

motion of the motor, the angular position of the motor, was provided, and the actuation torque required for this model to achieve that motion was calculated. The motor motion inputs were calculated based on the trajectory planned. Since the data that inputs were the unitless Simulink input signal and the Simscape only accept physical signal, a Simulink-PS converter needs to be added. Also, the Simulink-PS converter can filter and automatically calculate the derivatives of the signal. For instance, if the "filter input, derivatives calculated" and the "second-order filtering" were chosen, only the discrete position signal need to be provided. The converter can smooth out the curve that the first and second derivative of the signal is continuous as well. In the sensing section, the velocity and the actuator torque were checked. A Simulink-SP converter was added to convert the physical signal to Simulink signal so that the results can be read using scope blocks or save to the MATLAB workspace. In addition, the unit of the output signal can be chosen in the Simulink-SP converter as needed.

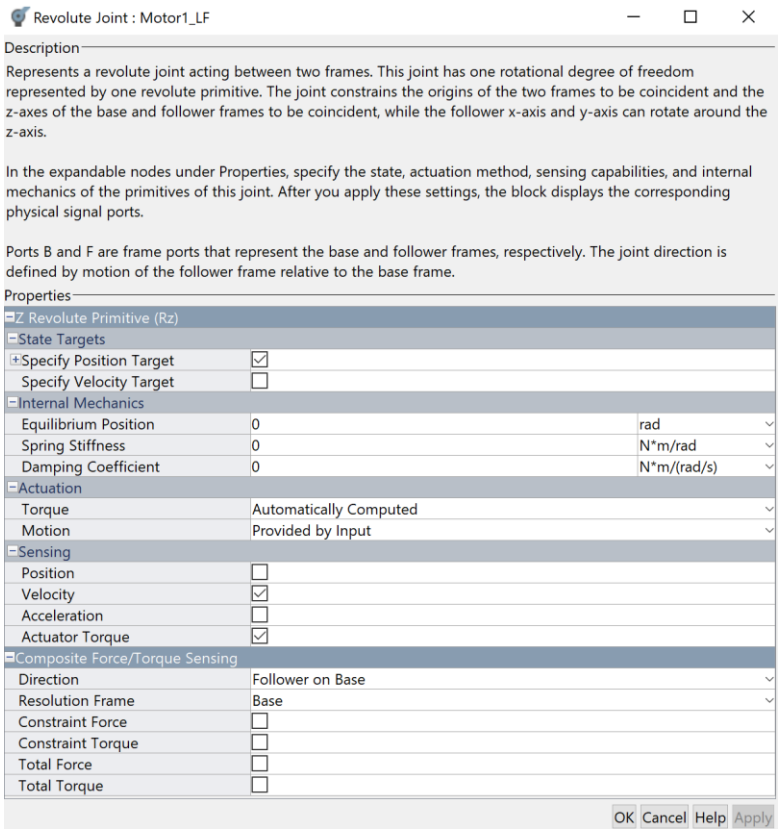


Figure 22. Properties of the revolute joint that represent the motor 1 of left front leg.

The contact force library contains different contact force blocks. The contact force blocks can be downloaded from the Mathwork website ("Simscape Multibody Contact Forces Library - File Exchange - MATLAB Central", 2019). The calculation of the contact force blocks is based on the contact mechanism. For simulation of the quadruped robot, the sphere to plane 3D contact block was used to simulate the contact condition between the foot tip and the ground, where the sphere side connects with the frame of the foot tip and the plane side connect with the frame of the ground. The radius of the foot tip and the depth of the floor can be defined in the contact force block. Also, the contact stiffness, contact damping coefficient, kinetic friction coefficient, and static friction stiffness between the foot tip and the floor can also be changed based on the simulation scenarios.

Chapter 6. Simulation

6.1. One leg simulation

The one leg simulation was done in order to check the Simscape Multibody result with the analysis result. In this scenario, the motors were fixed and controlled the foot tip move in the air following a certain trajectory. A 25N normal force and a 12.5 N friction force were applied to the foot tip. Control the foot tip move with a speed of 75mm/s following the trajectory. The motor angle signals were calculated using the inverse kinematic equations and the input into the Simscape Multibody model. Figure 23 below shows the block diagram of the Simscape Multibody of the one robotic leg.

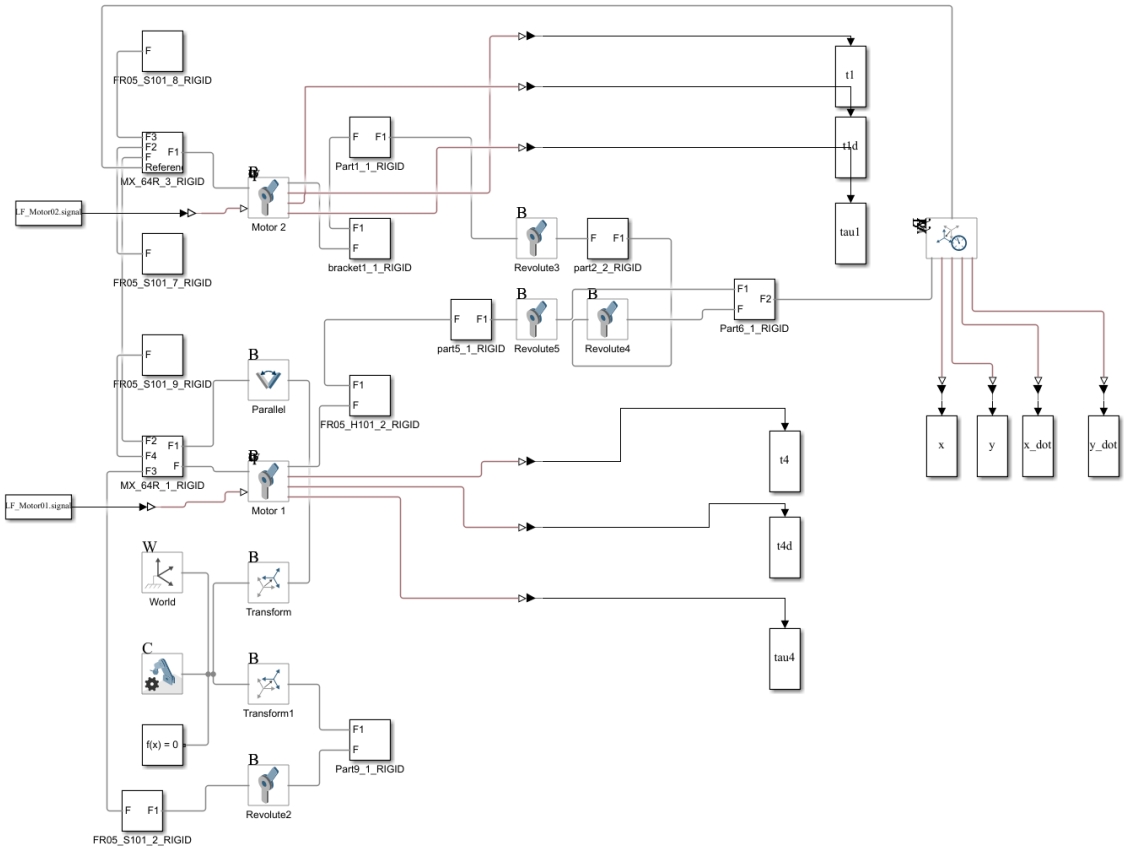


Figure 23: Simscape Multibody block diagram of one leg.

The angular velocities, actuator torques, foot tip positions were measured by adding sensors to the motor and the foot tip. Figure 24. below displays the analysis results and the simulation results of the motor angular velocities. The analysis results and the simulation results of the motor angular velocities are the same, which means the forward and inverse kinematics equations are correct. Figure 25. below shows the planned foot tip trajectory and the sensed simulation foot tip trajectory. The two trajectories overlap, which means the Simscape Multibody model is correct and can be used to do the kinematic analysis of the design.

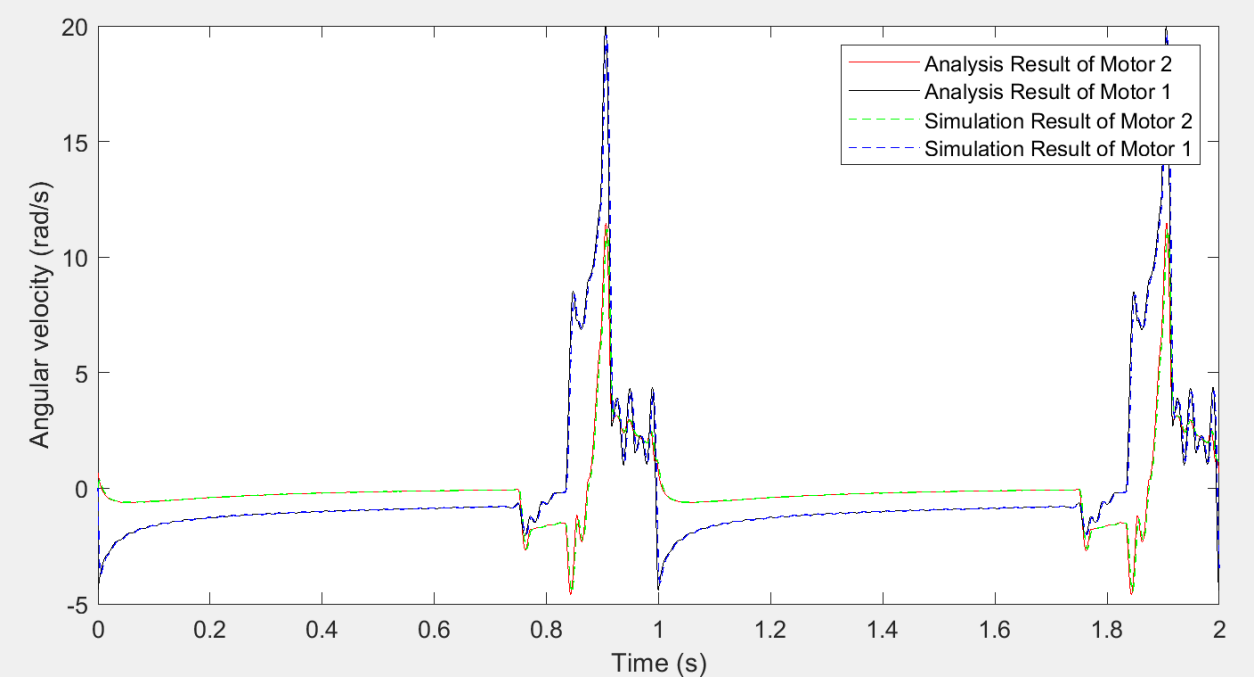


Figure 24: The analysis results and the simulation results of the motor angular velocities.

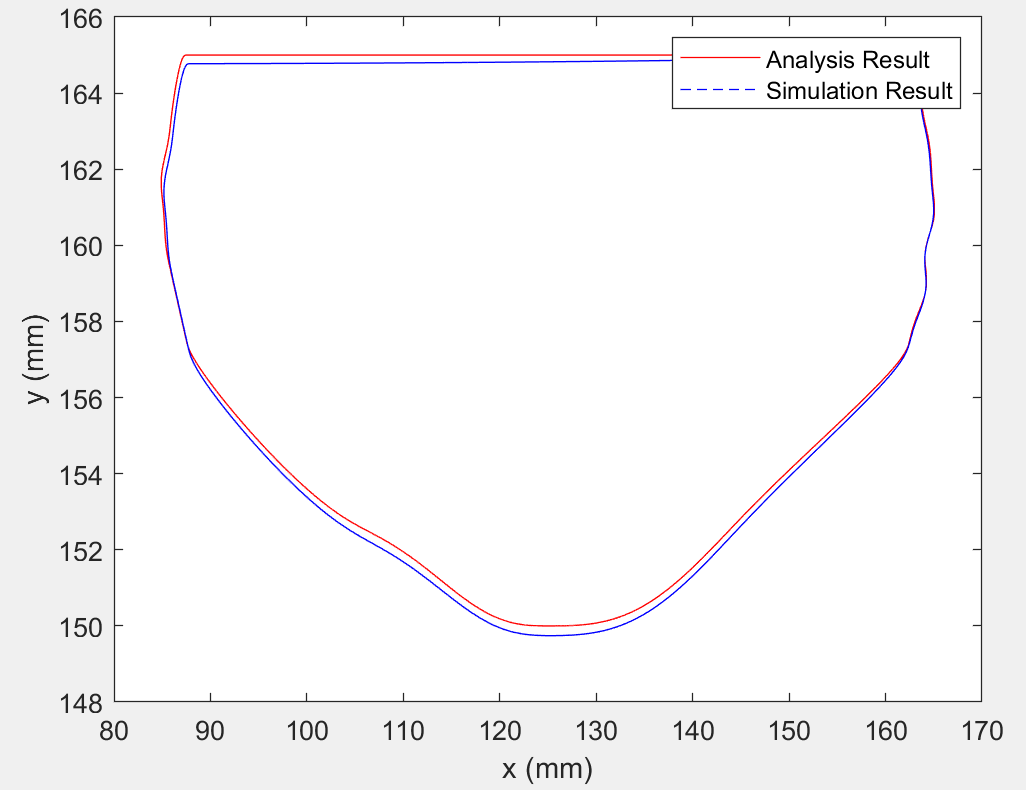


Figure 25: The planned foot tip trajectory and the sensed simulation foot tip trajectory.

Figure 26 displays the analysis results and the simulation results of the actuator torque required for the robotic leg to achieve expected motion. The calculated actuator torques are the same as the simulation torque results read from the torque sensor in Simscape, which means the derived torque equations are correct. Also, since the Simscape Multibody considers the inertia of the components, but the analysis results assume there is no power loss in the system, the similarity of the analysis and simulation results of the actuator torques indicates that there is little power loss in the system. Therefore, the quadruped robot designed using the 5-bar linkage mechanism can reduce the inertia and power loss caused by the weight of the motor.

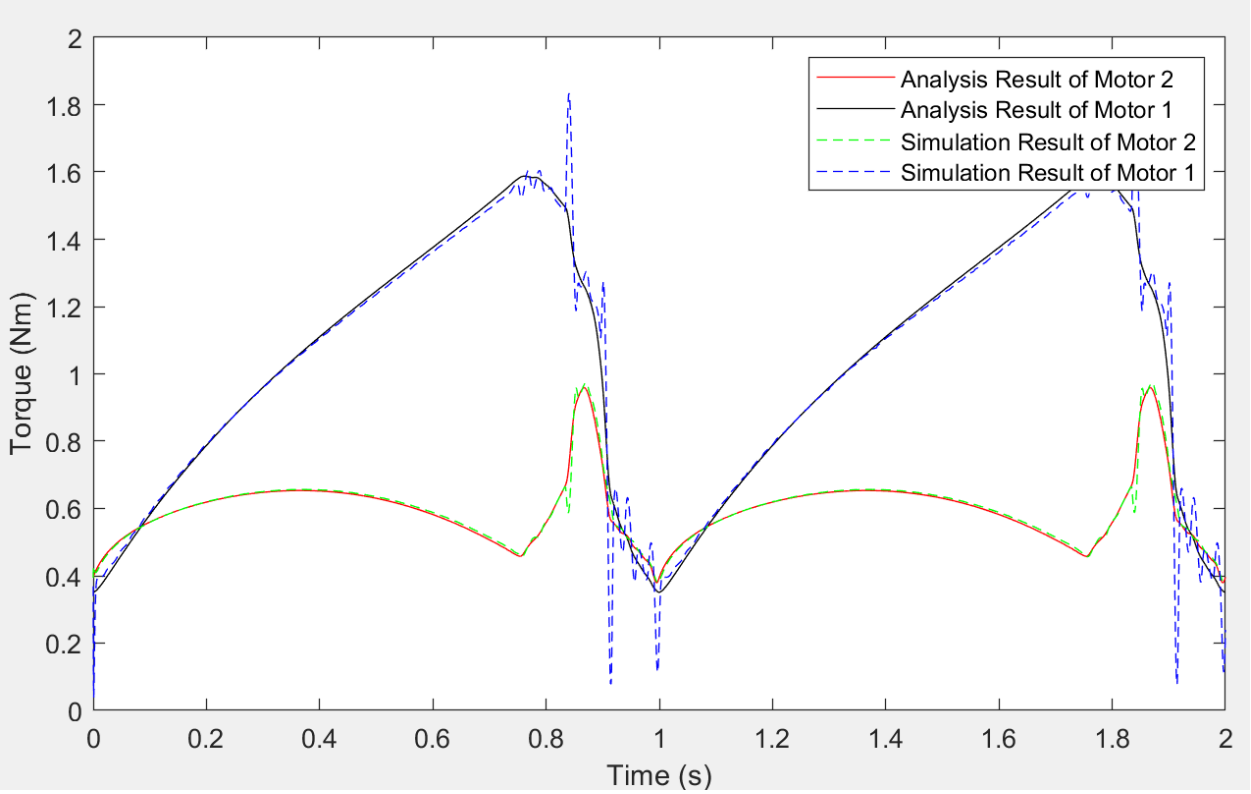


Figure 26: The analysis results and the simulation results of the actuator torques.

6.2. Simulation of walking

The scenario of the robot walking on the ground was simulated to understand the performance of the robot. In this case, the side plane of the robot was fixed in order to simplify the simulation condition, so that the robot can only do the forward motion. The data of motor angles change with time of walking situation was calculated using MATLAB function. The contact force between the foot tip and the ground was applied, where the contact stiffness used is 10,000 N/m, the contact damping is 100 Ns/m, the coefficient of kinetic friction is 0.5, the coefficient of static friction is 0.7, and the velocity threshold is 0.001 m/s. Figure 27 on next page shows the motor torque versus angular velocity of motor 1 and 2 of the left front leg and left hind leg on the performance plot of the motor MX64R ("ROBOTIS e-Manual", 2019). The

torque versus angular velocity graphs of motors on left front leg are the same as the graph of the right hind leg. Also, the torque verses angular velocity graphs of motors on the left hind leg are the same as the graph of the right front leg. The black line shows the maximum achievable performance of the motor. As a result, if the robot walks with a speed of 75mm/s, the torque versus angular velocity graph of the motor 2 of left hind leg almost lay beyond the performance of the motor, which means the maximum walking speed of the robot is 75mm/s.

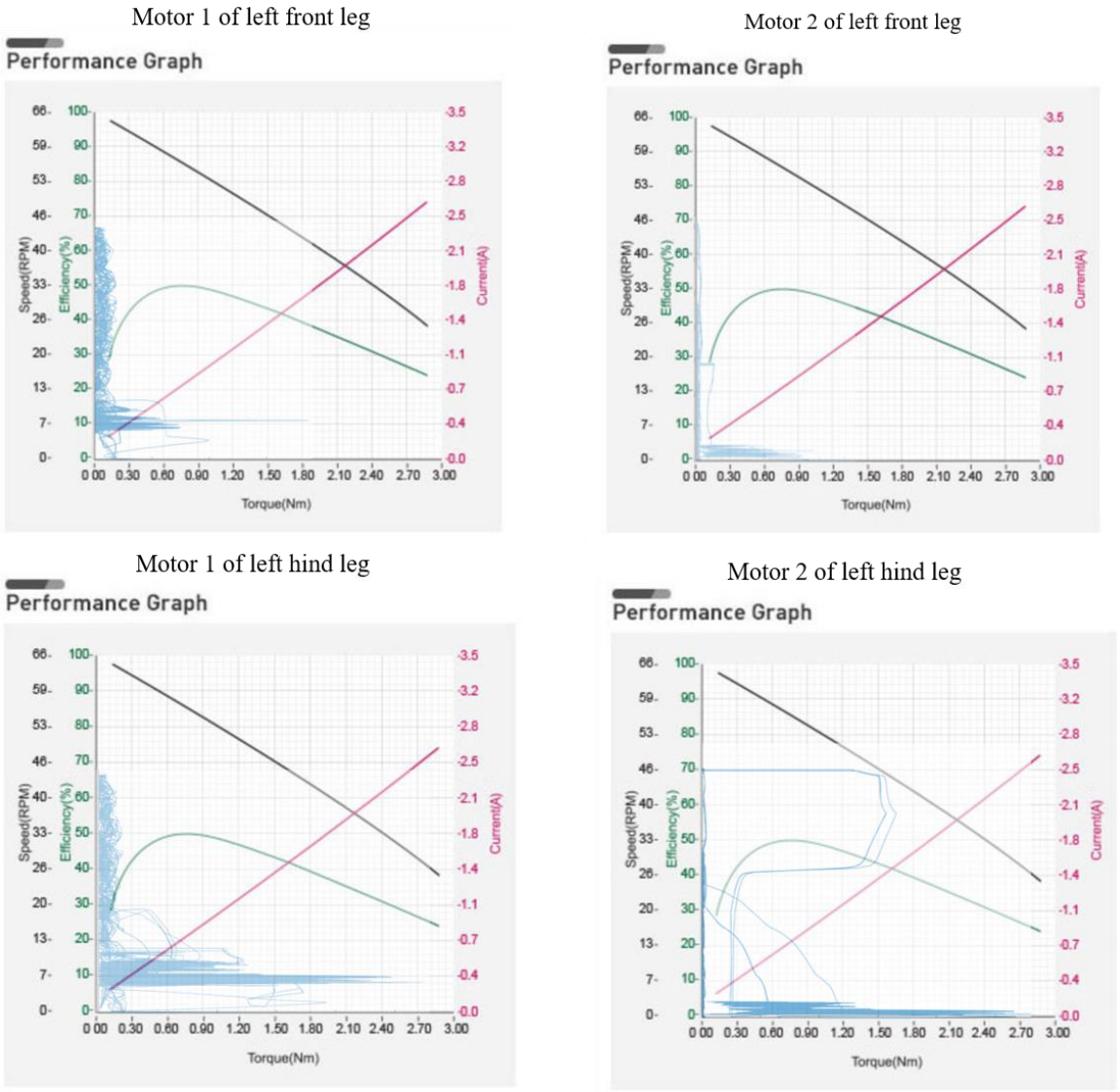


Figure 27: Torque verses angular speed of motors of left front and hind leg on the performance graph of the motor.

Chapter 7. Conclusion

7.1 Contributions

In this research, a quadruped robot was designed using the 5-bar linkage mechanism, where the commercial actuators, brackets, and frames were applied on the design. The commercial actuators and brackets are less expensive. Also, time and cost of designing, testing, calibrating customized motors can be saved. Connecting the motors using the commercial brackets and frames sell by the same company as the actuators, the sizing problem between the motors and the brackets and frames can be avoided. In addition, the platform can be easily duplicate because of the use of commercial components.

Also, this research approved that the 5-bar linkage mechanism is one effective method to reduce the inertia caused by the motor weight. According to the little difference between the simulation results and the analysis results that assume no power waste weight in the system, the weight of the linkages can create almost zero power loss. The designed quadruped robot has a small size. Therefore, it can be suitable for light load and small space applications, such as small range searching, transporting light cargos and entertainment. In addition, the quadruped robot can be good for educational use.

7.2 Future Work

7.2.1 Simulation of Other Scenarios

There are many other simulations can be done to better analyze the model. For example, the further students can simulate the walking, trotting, bounding, and pacing scenarios of the robot with the side plane removed, which are more realistic. In that case, more trajectory planning needs to be done. In addition, the scenarios of the robot turning and climbing stairs can

also be done. For the turning scenarios, the trajectory planning of the third motor at the shoulder and more researches will be needed.

7.2.2 Prototyping and Testing

Once all simulations are done, the future students can work on building the prototype of the quadruped robot. The Solidworks model of the quadruped robot has already built. The future students can modify the model to fit the bearings, bolts, screws, and controllers, etc. In additions, more functionalities can be added to the prototype based on the application. If needed, the designed quadruped robot can also be scaled for heavy load applications. Future students can also test the prototype can compare the experimental results with the simulation and analysis results that were done in this research.

7.3 Summary

In this research, the mechanical design, analysis, and simulation of a 5-bar linkage quadruped robot were done. The final design platform has a maximum dimension of 400 mm long, 342 mm wide and 240 mm high. As a result, the commercial DYNAMIXIAL MX64R motors were chosen to actuate the legs, which is powerful enough for the robot to talk. The forward and inverse kinematic equations and the torque equations were derived. In addition, the MATLAB codes and Simscape Multibody model of the robot were created for the simulations. Based on the analysis and simulation results, the maximum speed of walking is 75mm/s.

References

Bai, L., Guan, J., Chen, X., Hou, J., Duan, W., 2018. An optional passive/active transformable wheel-legged mobility concept for search and rescue robots. Robotics and Autonomous Systems 107, 145–155. doi:10.1016/j.robot.2018.06.005

Biagiotti, L., Melchiorri, C., 2010. Trajectory Planning. Trajectory Planning for Automatic Machines and Robots 1–12. doi:10.1007/978-3-540-85629-0_1

Cheng, Y., 2018. Design and Prototyping of a Transformable Quadruped Robot. (Doctoral dissertation, The Ohio State University)

Jin, M., Lei, X., Du, J., 2010. Evolutionary Game Theory in Multi-Objective Optimization Problem. International Journal of Computational Intelligence Systems 3, 74–87. doi:10.1080/18756891.2010.9727754

Mates and Joints - MATLAB & Simulink. [WWW Document]. URL https://www.mathworks.com/help/physmod/smlink/ref/mates-and-joints.html (accessed 4.5.19).

McKenzie, J. E., 2012. Design of robotic quadruped legs. (Doctoral dissertation, Massachusetts Institute of Technology).

ROBOTIS e [WWW Document]. Manual. URL http://emanual.robotis.com/docs/en/dxl/mx/mx-64-2/ (accessed 4.5.19).

Simscape Multibody Contact Forces Library - File Exchange - MATLAB Central [WWW Document], n.d. [WWW Document]. Simscape Multibody Contact Forces Library - File Exchange - MATLAB Central. URL

https://www.mathworks.com/matlabcentral/fileexchange/47417-simscape-multibody-contact-forces-library (accessed 4.5.19).

Zhou, C., Wang, B., Zhu, Q., Wu, J., 2016. An online gait generator for quadruped walking using motor primitives. International Journal of Advanced Robotic Systems 13, 172988141665796. doi:10.1177/1729881416657960

Appendix A: MATLAB code

Forward kinematic equation function: forward

```
function [x,y,xa,xb,xc,xd,xe,ya,yb,yc,yd,ye] = forward(theta1,theta4)
theta=27.98/180*pi; %theta, angle between link 3 and the extended link
l1=32.14; %Length of link 1, in mm.
l2=60;    %Length of link 2, in mm.
l3=40.11; %Length of link 3, in mm.
l4=89.2;  %Length of link 4, in mm.
l5=60.2;  %Length of link 5, in mm.
l=106.78; %Length of extended portion of link 3, in mm.
%Calculation below were based on the derived forward kinematic equations.
A=l1.^2+l3.^2+l4.^2+l5.^2-l2.^2;
B=2.*l5.*l4.*cos(theta4)-2.*l1.*l4.*cos(theta4-theta1)-
2.*l5.*l1.*cos(theta1);
C=2.*l3.*l4.*cos(theta4)+2.*l5.*l3-2.*l3.*l1.*cos(theta1);
D=2.*l3.*l4.*sin(theta4)-2.*l3.*l1.*sin(theta1);
% x1=(D+sqrt(D.^2-(A+B+C).*(A+B-C))./(A+B+C);
% theta3_1=2.*atan(x1);
% x_1=l5+l4.*cos(theta4)+l.*cos(theta+theta3_1)
% y_1=l4.*sin(theta4)+l.*cos(theta+theta3_1) %Incorrect assembly mode
x2=(D-sqrt(D.^2-(A+B+C).*(A+B-C)))./(A+B+C);
theta3_2=2.*atan(x2);
x=l5+l4.*cos(theta4)+l.*cos(theta+theta3_2); %foot tip position in x
direction
y=l4.*sin(theta4)+l.*sin(theta+theta3_2);    %foot tip position in y
direction
xa=0;                   %joint 1 (motor 2)position in x direction
ya=0;                   %joint 1 (motor 2)position in y direction
xe=60.2;                %joint 5 (motor 1)position in x direction
ye=0;                   %joint 5 (motor 1)position in y direction
xd=xe+l4.*cos(theta4);  %joint 4 position in x direction
yd=ye+l4.*sin(theta4);  %joint 4 position in y direction
xb=xa+l1.*cos(theta1);  %joint 2 position in x direction
yb=xa+l1.*sin(theta1);  %joint 2 position in y direction
xc=xd-l3.*cos(theta3_2);%joint 3 position in x direction
yc=yd-l3.*sin(theta3_2);%joint 3 position in y direction
end
```

Forward velocity kinematic equation function:

```
function [xd,yd] = forward_v(t1d,t4d,t1,t2,t3,t4)
%xd:x_dot yd:y_dot t1d:theta1_dot t4d:theta4_dot t1:theta1 t2:theta2
%t3:theta3 t4:theta4
theta=27.98/180*pi;  %theta, angle between link 3 and the extended link
l1=32.14;            %Length of link 1, in mm.
l2=60;               %Length of link 2, in mm.
l3=40.11;            %Length of link 3, in mm.
l4=89.2;             %Length of link 4, in mm.
l5=60.2;             %Length of link 5, in mm.
l=106.78;            %Length of extended portion of link 3, in mm.
%Calculation below were based on the derived forward velocity kinematic
equations.
A=(-l1.*sin(t1)+tan(t2).*l1.*cos(t1))./(tan(t2).*(-l3.*cos(t3))./(l.*...
cos(theta+t3))+(l3.*sin(t3)./(l.*cos(theta+t3))));
```

```
B=(-tan(t2).*((l3.*cos(t3))./(l.*cos(theta+t3))+1).*l4.*cos(t4)+(l3.*...
sin(t3).*l4.*cos(t4))./(l.*cos(theta+t3))+l4.*sin(t4))./(tan(t2).*(-l3.*...
cos(t3))./(l.*cos(theta+t3))+(l3.*sin(t3)./(l.*cos(theta+t3))));
xd=(-tan(theta+t3).*A.*t1d)+(-l4.*sin(t4)+tan(theta+t3)*l4.*cos(t4)-...
tan(theta+t3).*B).*t4d;  %foot tip position in y direction
yd=A.*t1d+B.*t4d;        %foot tip velocity in y direction
end
```

Inverse kinematic equation function:

```
function [theta1,theta2,theta3,theta4] = inverse(x,y)
l1=32.14;           %Length of link 1, in mm.
l2=60;              %Length of link 2, in mm.
l3=40.11;           %Length of link 3, in mm.
l4=89.2;            %Length of link 4, in mm.
l5=60.2;            %Length of link 5, in mm.
l=106.78;           %Length of extended portion of link 3, in mm.
theta=27.98/180*pi; %theta, angle between link 3 and the extended link
%Calculation below were based on the derived inverse kinematic equations.
K=2.*l.*y;
N=2.*l.*(x-l5);
M=y.^2+l.^2-l4.^2+(x-l5).^2;
theta3=pi/180*2.*atan2d(K-sqrt(K.^2-(M.^2-N.^2)),(M+N))-theta;
theta4=pi/180*atan2d((y-l.*sin(theta+theta3)),(x-l.*cos(theta+theta3)-l5));
%theta 4, angle of motor 1, in rad
O=(y-l3.*sin(theta3)-l.*sin(theta3+theta)).^2+(x-l3.*cos(theta3)-l.*...
cos(theta3+theta)).^2+l1.^2-l2.^2;
P=2.*(y-l3.*sin(theta3)-l.*sin(theta3+theta)).*l1;
Q=2.*(x-l3.*cos(theta3)-l.*cos(theta3+theta)).*l1;
theta1=pi/180*2.*atan2d((P-sqrt(P.^2-(O.^2-Q.^2))),(O+Q));
%theta 1, angle of motor 2, in rad
theta2=pi/180*atan2d((y-l.*sin(theta+theta3)-l3.*sin(theta3)-l1.*...
sin(theta1)),(x-l.*cos(theta+theta3)-l3.*cos(theta3)-l1.*cos(theta1)));
end
```

Inverse kinematic velocity equation function:

```
function [t1d,t4d] = inverse_v(xd,yd,t1,t2,t3,t4)
theta=27.98/180*pi; %theta, angle between link 3 and the extended link
l1=32.14;           %Length of link 1, in mm.
l2=60;              %Length of link 1, in mm.
l3=40.11;           %Length of link 1, in mm.
l4=89.2;            %Length of link 1, in mm.
l5=60.2;            %Length of link 1, in mm.
l=106.78;           %Length of extended portion of link 3, in mm.
%Calculation below were based on the derived inverse kinematic velocity
equations.
A=(-l1.*sin(t1)+tan(t2).*l1.*cos(t1))./(tan(t2).*(-l3.*cos(t3))./(l.*...
cos(theta+t3))+(l3.*sin(t3)./(l.*cos(theta+t3))));
B=(-tan(t2).*((l3.*cos(t3))./(l.*cos(theta+t3))+1).*l4.*cos(t4)+(l3.*...
sin(t3).*l4.*cos(t4))./(l.*cos(theta+t3))+l4.*sin(t4))./(tan(t2).*(-l3.*...
cos(t3))./(l.*cos(theta+t3))+(l3.*sin(t3)./(l.*cos(theta+t3))));
t1d=(xd-(-l4.*sin(t4)+tan(theta+t3)*l4.*cos(t4)-tan(theta+t3).*B).*yd./...
B)./(-tan(theta+t3).*A+(-l4.*sin(t4)+tan(theta+t3)*l4.*cos(t4)-...
tan(theta+t3).*B).*(-A)./B); %angular velocity of motor 2, in rad/s
```

```
t4d=(yd-A.*t1d)./B;          %angular velocity of motor 1, in rad/s
end
```

Foot workspace function:

```
function [x_2,y_2]=workspace(theta4_min,theta4_max,c)
a=theta4_min'; %theta4_min, in rad
b=theta4_max'; %theta4_max, in rad
theta1=c';     %theta1, in rad
d=(b-a)./79;
theta=27.98/180*pi; %theta
for j=1:79
    theta4(j,:)=a+d.*(j-1);
end
theta4;   %Angle of motor 1 theta4, in rad
l1=32.14; %Length of link 1, in mm.
l2=60;    %Length of link 2, in mm.
l3=40.11; %Length of link 3, in mm.
l4=89.2;  %Length of link 4, in mm.
l5=60.2;  %Length of link 5, in mm.
l=106.78; %Length of extended portion of link 3, in mm.
[x_2,y_2,xa,xb,xc,xd,xe,ya,yb,yc,yd,ye] = forward(theta1,theta4);
end
```

"D" shape gait function:

```
function [x_cpoint,y_cpoint,bx_new,by_new,T]=gait_walking(x,y,l1,l2,h1,h2)
%Coordination of all critical points (D shape)

%1:(x+l1/2,y)%2:(x+l1*17/36,y)%3:(x+l1*16/36,y)%4:(x+l1*15/36,y)
%5:(x+l1*14/36,y)%6:(x+l1*13/36,y)%7:(x+l1*12/36,y)%8:(x+l1*11/36,y)
%9:(x+l1*10/36,y)%10:(x+l1*9/36,y)%11:(x+l1*8/36,y)%12:(x+l1*7/36,y)
%13:(x+l1*6/36,y)%14:(x+l1*5/36,y)%15:(x+l1*4/36,y)%16:(x+l1*3/36,y)
%17:(x+l1*2/36,y)%18:(x+l1/36,y)%19:(x,y)%20:(x-l1/36,y)%21:(x-l1*2/36,y)
%22:(x-l1*3/36,y)%23:(x-l1*4/36,y)%24:(x-l1*5/36,y)%25:(x-l1*6/36,y)
%26:(x-l1*7/36,y)%27:(x-l1*8/36,y)%28:(x-l1*9/36,y)%29:(x-l1*10/36,y)
%30:(x-l1*11/36,y)%31:(x-l1*12/36,y)%32:(x-l1*13/36,y)%33:(x-l1*14/36,y)
%34:(x-l1*15/36,y)%35:(x-l1*16/36,y)%36:(x-l1*17/36,y)%37:(x-l1/2,y)
%38:(x-l1/2-(l2-l1)/3,y-h2/4)%39:(x-l2/2,y-h2/2)%40:(x-l1/2-(l2-l1)/3,y-
3*h2/4)
%41:(x-l1/2,y-h2)%42:(x-l1/4,y-5*h1/6)%43:(x,y-h1)%44:(x+l1/4,y-5*h1/6)
%45:(x+l1/2,y-h2)%46:(x+l1/2+(l2-l1)/3,y-3*h2/4)%47:(x+l2/2,y-h2/2)
%48:(x+l1/2+(l2-l1)/3,y-h2/4)

%Write the coordinate of the critical points into code:
for i=1:18
x_cpoint(i)=x+l1*(19-i)/36;
x_cpoint(i+19)=x-l1*i/36;
y_cpoint(i)=y;
y_cpoint(i+19)=y;
end
x_cpoint(19)=x;
y_cpoint(19)=y;
x_cpoint(38:50)=[x-l1/2-(l2-l1)/3,x-l2/2,x-l1/2-(l2-l1)/3,x-l1/2,x-l1/4,...
```

```
    x,x+l1/4,x+l1/2,x+l1/2+(l2-l1)/3,x+l2/2,x+l1/2+(l2-
l1)/3,x+l1/2,x+l1*17/36];
y_cpoint(38:50)=[y-h2/4,y-h2/2,y-3*h2/4,y-h2,y-5*h1/6,y-h1,y-5*h1/6,y-h2,...
    y-3*h2/4,y-h2/2,y-h2/4,y,y];
delta_t=1/48; % Time duration between each point, if assume the time
%duration of one cycle is 1 second.
%Assign the velocity at the critical point
for i=1:49
x_dot(i)=49*(x_cpoint(i+1)-x_cpoint(i));
y_dot(i)=49*(y_cpoint(i+1)-y_cpoint(i));
end
y_dot(43)=0;
y_dot(37)=0;
y_dot(41)=y_dot(40);
x_dot(41)=x_dot(40);
y_dot(46)=y_dot(47);
x_dot(46)=x_dot(47);
x_dot(50)=48*(x_cpoint(48)-x_cpoint(1));
y_dot(50)=48*(y_cpoint(48)-y_cpoint(1));
%Assign the acceleration at the critical point
x_2dot=linspace(0,0,50);
y_2dot=linspace(0,0,50);
% Fit x using 5th order polynomial
bx=x_cpoint(1:49);
bx_dot=x_dot(1:49);
bx_2dot=x_2dot(1:49);
bx1=x_cpoint(2:50);
bx1_dot=x_dot(2:50);
bx1_2dot=x_2dot(2:50);
ax0=bx;
ax1=bx_dot;
ax2=1/2.*bx_2dot;
ax3=(20.*(bx1-bx)-(8.*bx1_dot+12.*bx_dot).*delta_t-(3.*bx_2dot-
bx1_2dot).*delta_t.^2)./(2.*(delta_t^3));
ax4=(-30.*(bx1-bx)+(14.*bx1_dot+16.*bx_dot).*delta_t+(3.*bx_2dot-
2.*bx1_2dot).*delta_t.^2)./(2.*(delta_t^4));
ax5=(12.*(bx1-bx)-(6.*bx1_dot+6.*bx_dot).*delta_t+(bx_2dot-
bx1_2dot).*delta_t.^2)./(2.*(delta_t^5));

% Fit y using 5th order polynomial
dt=0:1/48/10000:(1/48-1/48/10000); %Time vector between two critical point
by=y_cpoint(1:49);
by_dot=y_dot(1:49);
by_2dot=x_2dot(1:49);
by1=y_cpoint(2:50);
by1_dot=y_dot(2:50);
by1_2dot=y_2dot(2:50);
ay0=by;
ay1=by_dot;
ay2=1/2.*bx_2dot;
ay3=(20.*(by1-by)-(8.*by1_dot+12.*by_dot).*delta_t-(3.*by_2dot-by1_2dot).*...
delta_t.^2)./(2.*(delta_t^3));
ay4=(-30.*(by1-by)+(14.*by1_dot+16.*by_dot).*delta_t+(3.*by_2dot-2.*...
by1_2dot).*delta_t.^2)./(2.*(delta_t^4));
ay5=(12.*(by1-by)-(6.*by1_dot+6.*by_dot).*delta_t+(by_2dot-by1_2dot).*...
delta_t.^2)./(2.*(delta_t^5));
%create 2 empty matrix for fitted x and y value of the trajectory
```

```
bx_new=[];
by_new=[];
%Fit the curve
for i=1:48
m=(i-1).*1/48;
bx_new((i-1)*10000+1:i*10000)=ax0(i)+ax1(i).*dt+ax2(i).*dt.^2+ax3(i).*...
dt.^3+ax4(i).*dt.^4+ax5(i).*dt.^5; %Fitted x value, in mm
by_new((i-1)*10000+1:i*10000)=ay0(i)+ay1(i).*dt+ay2(i).*dt.^2+ay3(i).*...
dt.^3+ay4(i).*dt.^4+ay5(i).*dt.^5; %Fitted y value, in mm
T((i-1)*10000+1:i*10000)=dt+m;     %Time, in s
end
end
```

Random gait function (Template):

```
function [bx_new,by_new,T]=gait_random()

%Coordination of all critical points:
%x1(,) x2(,)...
%y1(,) y2(,)...

N=; %Number of points, constant

%Write the coordinate of the critical points into code:
x_cpoint=[] %fill this row vector, repeat the x position of the
%first two point at the end of the velocity row vector. The dimension of
%this matrix: (1,N+2)
y_cpoint=[] %fill this row vector, repeat the y position of the
%first two point at the end of the velocity row vector. The dimension of
%this matrix: (1,N+2)

delta_t=1/N; % Time duration between each point, if assume the time
%duration of one cycle is 1 second.

%Assign the velocity at the critical point, repeat the velocity of the
%first two point at the end of the velocity row vector
x_dot(i)=[]%The dimension of this matrix: (1,N+2)
y_dot(i)=[]%The dimension of this matrix: (1,N+2)

%Assign the acceleration at the critical point, repeat the acceleration of
the
%first two point at the end of the velocity row vector
x_2dot=[]  %The dimension of this matrix: (1,N+2)
y_2dot=[]  %The dimension of this matrix: (1,N+2)

% Fit x using 5th order polynomial
bx=x_cpoint(1:N+1);
bx_dot=x_dot(1:N+1);
bx_2dot=x_2dot(1:N+1);
bx1=x_cpoint(2:N+2);
bx1_dot=x_dot(2:N+2);
bx1_2dot=x_2dot(2:N+2);
ax0=bx;
ax1=bx_dot;
```

```
ax2=1/2.*bx_2dot;
ax3=(20.*(bx1-bx)-(8.*bx1_dot+12.*bx_dot).*delta_t-(3.*bx_2dot-
bx1_2dot).*delta_t.^2)./(2.*(delta_t^3));
ax4=(-30.*(bx1-bx)+(14.*bx1_dot+16.*bx_dot).*delta_t+(3.*bx_2dot-
2.*bx1_2dot).*delta_t.^2)./(2.*(delta_t^4));
ax5=(12.*(bx1-bx)-(6.*bx1_dot+6.*bx_dot).*delta_t+(bx_2dot-
bx1_2dot).*delta_t.^2)./(2.*(delta_t^5));

% Fit y using 5th order polynomial
dt=0:1/N/10000:(1/N-1/N/10000); %Time vector between two critical point
by=y_cpoint(1:N+1);
by_dot=y_dot(1:N+1);
by_2dot=x_2dot(1:N+1);
by1=y_cpoint(2:N+2);
by1_dot=y_dot(2:N+2);
by1_2dot=y_2dot(2:N+2);
ay0=by;
ay1=by_dot;
ay2=1/2.*bx_2dot;
ay3=(20.*(by1-by)-(8.*by1_dot+12.*by_dot).*delta_t-(3.*by_2dot-by1_2dot).*...
delta_t.^2)./(2.*(delta_t^3));
ay4=(-30.*(by1-by)+(14.*by1_dot+16.*by_dot).*delta_t+(3.*by_2dot-2.*...
by1_2dot).*delta_t.^2)./(2.*(delta_t^4));
ay5=(12.*(by1-by)-(6.*by1_dot+6.*by_dot).*delta_t+(by_2dot-by1_2dot).*...
delta_t.^2)./(2.*(delta_t^5));
%create 2 empty matrix for fitted x and y value of the trajectory
bx_new=[];
by_new=[];

%Fit the curve
for i=1:N
m=(i-1).*1/N;
bx_new((i-1)*10000+1:i*10000)=ax0(i)+ax1(i).*dt+ax2(i).*dt.^2+ax3(i).*...
dt.^3+ax4(i).*dt.^4+ax5(i).*dt.^5; %Fitted x value, in mm
by_new((i-1)*10000+1:i*10000)=ay0(i)+ay1(i).*dt+ay2(i).*dt.^2+ay3(i).*...
dt.^3+ay4(i).*dt.^4+ay5(i).*dt.^5; %Fitted y value, in mm
T((i-1)*10000+1:i*10000)=dt+m;     %Time, in s
end
end
```

Motor angle signal function for quadruped of different gaits:

```
function
[Motor1_LF,Motor2_LF,Motor1_RF,Motor2_RF,Motor1_LH,Motor2_LH,Motor1_RH,Motor2
_RH]=motor_angle(bx_new,by_new)
[t1,t2,t3,t4] = inverse(bx_new,by_new); %Using inverse kinematic equation to
find the angle of motors
t=[linspace(0,1,480000)]';%Time duration. Don't change it
N=length(t1);%Length of the angle matrix
%Primary gaits for quadruped robot
         %Front left(P_LF)  %Front right(P_RF)    %Hind left(P_LH)     %Hind
right(P_RH)
%Walk        0                    pi                3*pi/2
pi/2
%Trot        0                    pi                  pi                  0
```

```
%Pace        0                       pi                  0
pi
%Bound       0                       0                   pi
pi
%Phase increases from 0 to 2*pi during one perios and P_i represents the
%phase shift with repect to front left leg.
%The number in the equation below (0.5,0.75,0.25) can be changed based on
%the primary gaits provided in the chart. c=P_i/2/pi. For example, for the
%front right leg of trot gait c=pi.pi/2=0.5
t1_RF=[t1(0.5*N+1:N)';t1(1:0.5*N)'];  %theta_1 (angle of motor 2)of right
front leg
t4_RF=[t4(0.5*N+1:N)';t4(1:0.5*N)'];  %theta_4 (angle of motor 1)of right
front leg
t1_LH=[t1(0.75*N+1:N)';t1(1:0.75*N)'];%theta_1 (angle of motor 2)of left hind
leg
t4_LH=[t4(0.75*N+1:N)';t4(1:0.75*N)'];%theta_4 (angle of motor 1)of left hind
leg
t1_RH=[t1(0.25*N+1:N)';t1(1:0.25*N)'];%theta_1 (angle of motor 2)of right
hind leg
t4_RH=[t4(0.25*N+1:N)';t4(1:0.25*N)'];%theta_4 (angle of motor 1)of right
hind leg
Motor1_LF=[t,[2*pi-t4]'];    %Motor 1 signal of left front leg.
Motor2_LF=[t,[pi-t1]'];      %Motor 2 signal of left front leg.
Motor1_RF=[t,[2*pi-t4_RF]];  %Motor 1 signal of right front leg.
Motor2_RF=[t,[pi-t1_RF]];    %Motor 2 signal of right front leg.
Motor1_LH=[t,[2*pi-t4_LH]];  %Motor 1 signal of left hind leg.
Motor2_LH=[t,[t1_LH-pi]];    %Motor 2 signal of left hind leg.
Motor1_RH=[t,[t4_RH-2*pi]];  %Motor 1 signal of right hind leg.
Motor2_RH=[t,[pi-t1_RH]];    %Motor 2 signal of right hind leg.
end
```

Main code for the SIMSCAPE Multibody:

```
% Simscape(TM) Multibody(TM) version: 6.0
clc, clear all, close all
% This is a model data file derived from a Simscape Multibody Import XML file
using the smimport function.
% The data in this file sets the block parameter values in an imported
Simscape Multibody model.
% For more information on this file, see the smimport function help page in
the Simscape Multibody documentation.
% You can modify numerical values, but avoid any other changes to this file.
% Do not edit the physical units shown in comments.

%SPEED 75mm/s
V=100;   %Change Speed (mm/s)
cycle=6; %Change # of cycle(number of stpes)

%Draw the workspace of the foot tip
w=xlsread('workspace.xlsx');          %Read excel file
theta1=[(w(2:80,4)-180)/180*pi];      %theta1 (motor 2 angles)
theta4_min=[(w(2:80,5)-180)/180*pi]; %theta4_min (minimum motor 1 angles)
theta4_max=[(w(2:80,6)-180)/180*pi]; %theta4_max (maximum motor 1 angles)
[x_2,y_2]=workspace(theta4_min,theta4_max,theta1); %Find foot tip workspace
%The workspace of the foot tip was calculated using forward kinematics
```

```
%equations.
x_m1=60.2; %x of motor 1 (mm)
y_m1=0;    %y of motor 1 (mm)
x_m2=0;    %x of motor 2 (mm)
y_m2=0;    %y of motor 2 (mm)
figure(1)
plot(x_2,y_2,'color',[0.5 0.5 0.5])%Plot the workspace
hold on
plot(x_m1,y_m1,'o',x_m2,y_m1,'o')  %Plot the motor position

%Change trajectory of walking
l1=60;   %gait size, mm
y=165;   %reference point y, mm
x=127;   %reference point x, mm
l2=65;   %maximum length of gait, mm
h1=16;   %gait height, mm
h2=h1/2; %height of a critical point, mm
h=y*0.001-0.005; %Initial height of the robot (In the simulation, the robot
%will drop from the hight you assigned)
[x_cpoint,y_cpoint,bx_new,by_new]=gait_walking(x,y,l1,l2,h1,h2);%Find
critical points
%[critical point x,critical point y,fitted trajectory_x,fitted trajectory_x]
plot(x_cpoint,y_cpoint,'.r') %Plot critical point
plot(bx_new,by_new,'r')      %Plot fitted trajectory
hold off
axis([0 200 0 200])          %Range of the axis

%If you want to design a new trajectory other than D shape, you can
%Complete the gait random code and call the function below
%[bx_new,by_new,T]=gait_random()

%Motor angle of 8 motors
[Motor1_LF,Motor2_LF,Motor1_RF,Motor2_RF,Motor1_LH,Motor2_LH,Motor1_RH,Motor2
_RH]=motor_angle(bx_new,by_new);
%[motor 1 angle of left front leg, motor 2 angle of left front leg,...]
%These angles were calculated using inverse kinematic equations
t=l1/V;     %Time duration of one cycle
ts=t*cycle; %stop time of the simulation
time=linspace(0,cycle*t,480000*cycle); %Time vector
%Motor time signals of left front leg
LF_Motor01.signal(:,1)=time';
LF_Motor02.signal(:,1)=time';
LF_Motor03.signal(:,1)=time';
%Motor time signals of left hind leg
LH_Motor01.signal(:,1)=time';
LH_Motor02.signal(:,1)=time';
LH_Motor03.signal(:,1)=time';
%Motor time signals of right front leg
RF_Motor01.signal(:,1)=time';
RF_Motor02.signal(:,1)=time';
RF_Motor03.signal(:,1)=time';
%Motor time signals of right hind leg
RH_Motor01.signal(:,1)=time';
RH_Motor02.signal(:,1)=time';
RH_Motor03.signal(:,1)=time';
%Motor 3 signal of four legs (stationary)
```

```
lf_motor03=xlsread('Motor3_LF');
lh_motor03=xlsread('Motor3_LH');
rf_motor03=xlsread('Motor3_RF');
rh_motor03=xlsread('Motor3_RH');
%Motor angle signals of left front leg
LF_Motor01.signal(:,2)=repmat(Motor1_LF(:,2)',1,cycle)';
LF_Motor02.signal(:,2)=repmat(Motor2_LF(:,2)',1,cycle)';
LF_Motor03.signal(:,2)=repmat(lf_motor03(1,2),1,480000*cycle)';
%Motor angle signals of left hind leg
LH_Motor01.signal(:,2)=repmat(Motor1_LH(:,2)',1,cycle)';
LH_Motor02.signal(:,2)=repmat(Motor2_LH(:,2)',1,cycle)';
LH_Motor03.signal(:,2)=repmat(lh_motor03(1,2),1,480000*cycle)';
%Motor angle signals of right front leg
RF_Motor01.signal(:,2)=repmat(Motor1_RF(:,2)',1,cycle)';
RF_Motor02.signal(:,2)=repmat(Motor2_RF(:,2)',1,cycle)';
RF_Motor03.signal(:,2)=repmat(rf_motor03(1,2),1,480000*cycle)';
%Motor angle signals of right hind leg
RH_Motor01.signal(:,2)=repmat(Motor1_RH(:,2)',1,cycle)';
RH_Motor02.signal(:,2)=repmat(Motor2_RH(:,2)',1,cycle)';
RH_Motor03.signal(:,2)=repmat(rh_motor03(1,2),1,480000*cycle)';
```

Main code for one leg simulation:

```
% Simscape(TM) Multibody(TM) version: 6.0
clc, clear all, close all
% This is a model data file derived from a Simscape Multibody Import XML file
using the smimport function.
% The data in this file sets the block parameter values in an imported
Simscape Multibody model.
% For more information on this file, see the smimport function help page in
the Simscape Multibody documentation.
% You can modify numerical values, but avoid any other changes to this file.
% Do not add code to this file. Do not edit the physical units shown in
comments.
%SPEED 75mm/s
V=75;   %Change Speed (mm/s)
cycle=2; %Change # of cycle(number of stpes)

%Draw the workspace of the foot tip
w=xlsread('workspace.xlsx');          %Read excel file
theta1=[(w(2:80,4)-180)/180*pi];      %theta1 (motor 2 angles)
theta4_min=[(w(2:80,5)-180)/180*pi]; %theta4_min (minimum motor 1 angles)
theta4_max=[(w(2:80,6)-180)/180*pi]; %theta4_max (maximum motor 1 angles)
[x_2,y_2]=workspace(theta4_min,theta4_max,theta1); %Find foot tip workspace
%The workspace of the foot tip was calculated using forward kinematics
%equations.
x_m1=60.2; %x of motor 1 (mm)
y_m1=0;    %y of motor 1 (mm)
x_m2=0;    %x of motor 2 (mm)
y_m2=0;    %y of motor 2 (mm)
figure(1)
plot(x_2,y_2,'color',[0.5 0.5 0.5])%Plot the workspace
hold on
plot(x_m1,y_m1,'o',x_m2,y_m1,'o')  %Plot the motor position

%Change trajectory of walking
```

```
l1=75;   %gait size, mm
y=165;   %reference point y, mm
x=125;   %reference point x, mm
l2=80;   %maximum length of gait, mm
h1=15;   %gait height, mm
h2=h1/2; %height of a critical point, mm
h=y*0.001-0.005; %Initial height of the robot (In the simulation, the robot
%will drop from the hight you assigned)
[x_cpoint,y_cpoint,bx_new,by_new]=gait_walking(x,y,l1,l2,h1,h2);%Find
critical points
%[critical point x,critical point y,fitted trajectory_x,fitted trajectory_x]
plot(x_cpoint,y_cpoint,'.r') %Plot critical point
plot(bx_new,by_new,'r')      %Plot fitted trajectory
hold off
axis([0 200 0 200])          %Range of the axis
xlabel('x (mm)')
ylabel('y (mm)')
%If you want to design a new trajectory other than D shape, you can
%Complete the gait random code and call the function below
%[bx_new,by_new,T]=gait_random()
%Motor angle of 8 motors
[Motor1_LF,Motor2_LF,Motor1_RF,Motor2_RF,Motor1_LH,Motor2_LH,Motor1_RH,Motor2
_RH]=motor_angle(bx_new,by_new);
%[motor 1 angle of left front leg, motor 2 angle of left front leg,...]
%These angles were calculated using inverse kinematic equations
t=l1/V;     %Time duration of one cycle
ts=t*cycle; %stop time of the simulation
time=linspace(0,cycle*t,480000*cycle); %Time vector
%Motor time signals of left front leg
LF_Motor01.signal(:,1)=time';
LF_Motor02.signal(:,1)=time';
LF_Motor03.signal(:,1)=time';
%Motor 3 signal of four legs (stationary)
lf_motor03=xlsread('Motor3_LF');
%Motor angle signals of left front leg
LF_Motor01.signal(:,2)=repmat(Motor1_LF(:,2)',1,cycle)';
LF_Motor02.signal(:,2)=repmat(Motor2_LF(:,2)',1,cycle)';
LF_Motor03.signal(:,2)=repmat(lf_motor03(1,2),1,480000*cycle)';

%%%VariableName:smiData


%============= RigidTransform =============%

%Initialize the RigidTransform structure array by filling in null values.
smiData.RigidTransform(29).translation = [0.0 0.0 0.0];
smiData.RigidTransform(29).angle = 0.0;
smiData.RigidTransform(29).axis = [0.0 0.0 0.0];
smiData.RigidTransform(29).ID = '';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(1).translation = [0 0 28.374999999999996];  % mm
smiData.RigidTransform(1).angle = 3.1415926535897931;  % rad
smiData.RigidTransform(1).axis = [1 0 0];
smiData.RigidTransform(1).ID = 'B[MX-64R-1:-:FR05-H101-2]';
```

```
%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(2).translation = [28.374999999999986 -
6.865619184281968e-13 -9.2370555648813024e-14];  % mm
smiData.RigidTransform(2).angle = 2.0943951023931953;  % rad
smiData.RigidTransform(2).axis = [-0.57735026918962584 -0.57735026918962584
0.57735026918962584];
smiData.RigidTransform(2).ID = 'F[MX-64R-1:-:FR05-H101-2]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(3).translation = [-23.5 36.000000000000028
13.999999999999984];  % mm
smiData.RigidTransform(3).angle = 2.0943951023931953;  % rad
smiData.RigidTransform(3).axis = [-0.57735026918962584 -0.57735026918962584 -
0.57735026918962584];
smiData.RigidTransform(3).ID = 'B[FR05-H101-2:-:part5-1]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(4).translation = [-18.681451612902556
2.3980817331903381e-14 17.040322580645096];  % mm
smiData.RigidTransform(4).angle = 2.0943951023931948;  % rad
smiData.RigidTransform(4).axis = [-0.57735026918962673 -0.57735026918962551 -
0.57735026918962506];
smiData.RigidTransform(4).ID = 'F[FR05-H101-2:-:part5-1]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(5).translation = [0 0 20.500000000000004];  % mm
smiData.RigidTransform(5).angle = 0;  % rad
smiData.RigidTransform(5).axis = [0 0 0];
smiData.RigidTransform(5).ID = 'B[MX-64R-1:-:]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(6).translation = [248.52361797549509 -
45.000000000000071 59.595078331327215];  % mm
smiData.RigidTransform(6).angle = 3.1415926535897931;  % rad
smiData.RigidTransform(6).axis = [-0 -0 1];
smiData.RigidTransform(6).ID = 'F[MX-64R-1:-:]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(7).translation = [14.500000000000012
13.000000000000005 -20.5];  % mm
smiData.RigidTransform(7).angle = 2.0943951023931953;  % rad
smiData.RigidTransform(7).axis = [-0.57735026918962584 -0.57735026918962584 -
0.57735026918962584];
smiData.RigidTransform(7).ID = 'B[MX-64R-1:-:MX-64R-3]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(8).translation = [74.700000000002291 13.00000000000022
-20.500000000000135];  % mm
smiData.RigidTransform(8).angle = 2.0943951023931935;  % rad
```

```
smiData.RigidTransform(8).axis = [-0.57735026918962618 -0.57735026918962518 -
0.57735026918962595];
smiData.RigidTransform(8).ID = 'F[MX-64R-1:-:MX-64R-3]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(9).translation = [0 0 28.375000000000011];  % mm
smiData.RigidTransform(9).angle = 3.1415926535897931;  % rad
smiData.RigidTransform(9).axis = [1 0 0];
smiData.RigidTransform(9).ID = 'B[MX-64R-3:-:bracket1-1]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(10).translation = [-30.498418506574069
24.000000000000043 -2.5161870503596901];  % mm
smiData.RigidTransform(10).angle = 2.0943951023931953;  % rad
smiData.RigidTransform(10).axis = [0.57735026918962662 0.57735026918962562
0.57735026918962506];
smiData.RigidTransform(10).ID = 'F[MX-64R-3:-:bracket1-1]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(11).translation = [17.818548387096762
2.9999999999999751 -7.959677419354855];  % mm
smiData.RigidTransform(11).angle = 2.0943951023931953;  % rad
smiData.RigidTransform(11).axis = [-0.57735026918962584 -0.57735026918962584
-0.57735026918962584];
smiData.RigidTransform(11).ID = 'B[Part1-1:-:bracket1-1]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(12).translation = [16.876581493425938 -
2.9999999999998925 8.483812949640317];  % mm
smiData.RigidTransform(12).angle = 2.0943951023931953;  % rad
smiData.RigidTransform(12).axis = [0.57735026918962595 0.57735026918962573 -
0.57735026918962584];
smiData.RigidTransform(12).ID = 'F[Part1-1:-:bracket1-1]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(13).translation = [10.999999999999982 -
45.300000000000011 17];  % mm
smiData.RigidTransform(13).angle = 3.1415926535897931;  % rad
smiData.RigidTransform(13).axis = [1 0 0];
smiData.RigidTransform(13).ID = 'B[MX-64R-1:-:FR05-S101-2]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(14).translation = [10.999999999999972
6.7000000000000011 16.999999999999986];  % mm
smiData.RigidTransform(14).angle = 3.1415926535897931;  % rad
smiData.RigidTransform(14).axis = [-1 0 0];
smiData.RigidTransform(14).ID = 'F[MX-64R-1:-:FR05-S101-2]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
```

```
smiData.RigidTransform(15).translation = [-7.778174593051995
2.0000000000000009 -7.7781745930520021];  % mm
smiData.RigidTransform(15).angle = 2.0943951023931953;  % rad
smiData.RigidTransform(15).axis = [0.57735026918962584 -0.57735026918962584
0.57735026918962584];
smiData.RigidTransform(15).ID = 'B[FR05-S101-2:-:Part9-1]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(16).translation = [-10.242108726948683
4.9999999999999378 -4.9701525638185515];  % mm
smiData.RigidTransform(16).angle = 2.0943951023931953;  % rad
smiData.RigidTransform(16).axis = [-0.57735026918962584 -0.57735026918962584
-0.57735026918962584];
smiData.RigidTransform(16).ID = 'F[FR05-S101-2:-:Part9-1]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(17).translation = [0 0 -16.999999999999996];  % mm
smiData.RigidTransform(17).angle = 3.1415926535897931;  % rad
smiData.RigidTransform(17).axis = [1 0 0];
smiData.RigidTransform(17).ID = 'B[MX-64R-3:-:FR05-S101-7]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(18).translation = [-29.000000000000774
24.000000000000085 -16.999999999999993];  % mm
smiData.RigidTransform(18).angle = 3.1415926535897931;  % rad
smiData.RigidTransform(18).axis = [-0.70710678118654835 -0.70710678118654668
4.9065389333854562e-18];
smiData.RigidTransform(18).ID = 'F[MX-64R-3:-:FR05-S101-7]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(19).translation = [0 0 17];  % mm
smiData.RigidTransform(19).angle = 0;  % rad
smiData.RigidTransform(19).axis = [0 0 0];
smiData.RigidTransform(19).ID = 'B[MX-64R-3:-:FR05-S101-8]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(20).translation = [-3.6680071654985777e-14 52 -
17.000000000000007];  % mm
smiData.RigidTransform(20).angle = 3.1415926535897931;  % rad
smiData.RigidTransform(20).axis = [-6.1062266354383629e-16 1 -
6.9388939039072808e-18];
smiData.RigidTransform(20).ID = 'F[MX-64R-3:-:FR05-S101-8]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(21).translation = [0 0 17];  % mm
smiData.RigidTransform(21).angle = 0;  % rad
smiData.RigidTransform(21).axis = [0 0 0];
smiData.RigidTransform(21).ID = 'B[MX-64R-1:-:FR05-S101-9]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
```

```
smiData.RigidTransform(22).translation = [-29.000000000000561
23.999999999999972 -16.999999999999993];  % mm
smiData.RigidTransform(22).angle = 3.1415926535897927;  % rad
smiData.RigidTransform(22).axis = [-0.70710678118654757 0.70710678118654746
3.9194107536161842e-17];
smiData.RigidTransform(22).ID = 'F[MX-64R-1:-:FR05-S101-9]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(23).translation = [-26.681451612903238
7.9999999999999796 3.0403225806451548];  % mm
smiData.RigidTransform(23).angle = 2.0943951023931953;  % rad
smiData.RigidTransform(23).axis = [-0.57735026918962584 -0.57735026918962584
0.57735026918962584];
smiData.RigidTransform(23).ID = 'B[Part1-1:-:part2-2]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(24).translation = [32.787041156840971
32.999999999975913 7.3387096774141973];  % mm
smiData.RigidTransform(24).angle = 2.0943951023931953;  % rad
smiData.RigidTransform(24).axis = [0.57735026918962629 0.57735026918962584
0.57735026918962529];
smiData.RigidTransform(24).ID = 'F[Part1-1:-:part2-2]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(25).translation = [29.787041156840928 -
27.000000000000011 7.3387096774193967];  % mm
smiData.RigidTransform(25).angle = 2.0943951023931953;  % rad
smiData.RigidTransform(25).axis = [-0.57735026918962584 -0.57735026918962584
0.57735026918962584];
smiData.RigidTransform(25).ID = 'B[part2-2:-:Part6-1]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(26).translation = [26.766293677505416 45
20.518245517974584];  % mm
smiData.RigidTransform(26).angle = 2.0943951023931948;  % rad
smiData.RigidTransform(26).axis = [0.57735026918962551 -0.5773502691896254
0.5773502691896264];
smiData.RigidTransform(26).ID = 'F[part2-2:-:Part6-1]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(27).translation = [27.318548387096762
53.000000000000028 3.0403225806451548];  % mm
smiData.RigidTransform(27).angle = 2.0943951023931953;  % rad
smiData.RigidTransform(27).axis = [-0.57735026918962584 -0.57735026918962584
0.57735026918962584];
smiData.RigidTransform(27).ID = 'B[part5-1:-:Part6-1]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(28).translation = [-13.233706322511019
44.999999999999375 20.518245517990618];  % mm
smiData.RigidTransform(28).angle = 2.0943951023931944;  % rad
```

```
smiData.RigidTransform(28).axis = [0.57735026918962551 -0.5773502691896264
0.5773502691896254];
smiData.RigidTransform(28).ID = 'F[part5-1:-:Part6-1]';

%Translation Method - Cartesian
%Rotation Method - Arbitrary Axis
smiData.RigidTransform(29).translation = [266.54390129854579 0
36.287056295808462];  % mm
smiData.RigidTransform(29).angle = 6.13673604360966e-10;  % rad
smiData.RigidTransform(29).axis = [0 1 0];
smiData.RigidTransform(29).ID = 'RootGround[Part9-1]';


%============= Solid =============%
%Center of Mass (CoM) %Moments of Inertia (MoI) %Product of Inertia (PoI)

%Initialize the Solid structure array by filling in null values.
smiData.Solid(9).mass = 0.0;
smiData.Solid(9).CoM = [0.0 0.0 0.0];
smiData.Solid(9).MoI = [0.0 0.0 0.0];
smiData.Solid(9).PoI = [0.0 0.0 0.0];
smiData.Solid(9).color = [0.0 0.0 0.0];
smiData.Solid(9).opacity = 0.0;
smiData.Solid(9).ID = '';

%Inertia Type - Custom
%Visual Properties - Simple
smiData.Solid(1).mass = 0.0085543077551081606;  % kg
smiData.Solid(1).CoM = [7.2870411568409423 2.088029852136827
7.3387096774193505];  % mm
smiData.Solid(1).MoI = [2.5242026322026345 5.1738690322597813
7.3898251507786341];  % kg*mm^2
smiData.Solid(1).PoI = [0 0 0];  % kg*mm^2
smiData.Solid(1).color = [1 1 1];
smiData.Solid(1).opacity = 1;
smiData.Solid(1).ID = 'part2*:*Default';

%Inertia Type - Custom
%Visual Properties - Simple
smiData.Solid(2).mass = 0.19554537644067652;  % kg
smiData.Solid(2).CoM = [-1.0090375674947288e-05 -15.207757813191309
0.45162537664038044];  % mm
smiData.Solid(2).MoI = [80.571685715754839 45.810332031966553
67.262710515108211];  % kg*mm^2
smiData.Solid(2).PoI = [-2.3792415260747326 3.8538684563667216e-05 -
7.72279719439599e-07];  % kg*mm^2
smiData.Solid(2).color = [0.89803921568627454 0.91764705882352937
0.92941176470588238];
smiData.Solid(2).opacity = 1;
smiData.Solid(2).ID = 'MX-64R*:*Default';

%Inertia Type - Custom
%Visual Properties - Simple
smiData.Solid(3).mass = 0.0073139198480387203;  % kg
smiData.Solid(3).CoM = [-1.1403613477939691 2.6024144394746092
3.650613341111712e-06];  % mm
```

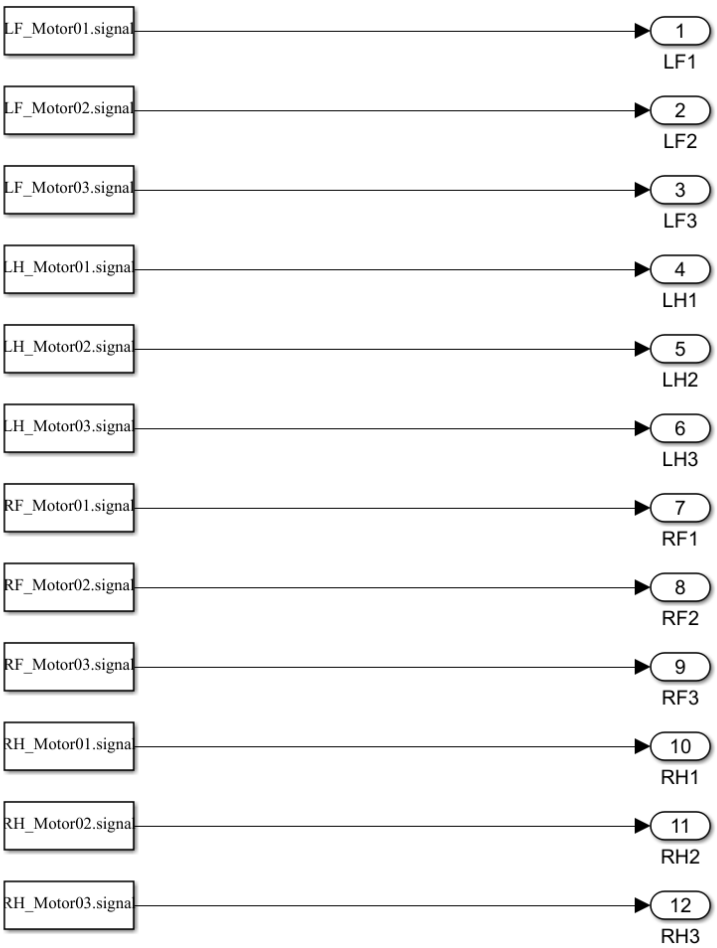
```
smiData.Solid(3).MoI = [1.4884079291232113 2.0810855752391628
0.68819276239930571];  % kg*mm^2
smiData.Solid(3).PoI = [4.9660899241797659e-08 5.5519095607826391e-06
0.02495920614506578];  % kg*mm^2
smiData.Solid(3).color = [0.89803921568627454 0.91764705882352937
0.92941176470588238];
smiData.Solid(3).opacity = 1;
smiData.Solid(3).ID = 'FR05-S101*:*Default';

%Inertia Type - Custom
%Visual Properties - Simple
smiData.Solid(4).mass = 0.017881331911097097;  % kg
smiData.Solid(4).CoM = [0 25.631314647856914 0];  % mm
smiData.Solid(4).MoI = [4.2998641225714822 8.953534226808717
9.4245628914003259];  % kg*mm^2
smiData.Solid(4).PoI = [0 0 0];  % kg*mm^2
smiData.Solid(4).color = [0.89803921568627454 0.91764705882352937
0.92941176470588238];
smiData.Solid(4).opacity = 1;
smiData.Solid(4).ID = 'FR05-H101*:*Default';

%Inertia Type - Custom
%Visual Properties - Simple
smiData.Solid(5).mass = 0.0059776231818871525;  % kg
smiData.Solid(5).CoM = [-1.1814516129032298 3.0481087480343043
3.0403225806451593];  % mm
smiData.Solid(5).MoI = [0.45421415045625479 2.2320299662204635
1.8917473171685539];  % kg*mm^2
smiData.Solid(5).PoI = [0 0 0];  % kg*mm^2
smiData.Solid(5).color = [1 1 1];
smiData.Solid(5).opacity = 1;
smiData.Solid(5).ID = 'Part1*:*Default';

%Inertia Type - Custom
%Visual Properties - Simple
smiData.Solid(6).mass = 0.028708719018235306;  % kg
smiData.Solid(6).CoM = [-49.093221738900404 22.464732915990901
39.66809112533447];  % mm
smiData.Solid(6).MoI = [10.753324289867129 42.433997440745564
37.46474298825747];  % kg*mm^2
smiData.Solid(6).PoI = [-0.016324261760793549 15.063917671532389
0.026827961542742248];  % kg*mm^2
smiData.Solid(6).color = [1 1 1];
smiData.Solid(6).opacity = 1;
smiData.Solid(6).ID = 'Part6*:*Default';

%Inertia Type - Custom
%Visual Properties - Simple
smiData.Solid(7).mass = 0.0059193023604727467;  % kg
smiData.Solid(7).CoM = [-2.1234185065740547 6.0210860988061814 -
2.5161870503597079];  % mm
smiData.Solid(7).MoI = [0.92412489094012373 2.6789110006490677
2.3801621501865102];  % kg*mm^2
smiData.Solid(7).PoI = [0 0 0];  % kg*mm^2
smiData.Solid(7).color = [1 1 1];
smiData.Solid(7).opacity = 1;
smiData.Solid(7).ID = 'bracket1*:*Default';
```

```
%Inertia Type - Custom
%Visual Properties - Simple
smiData.Solid(8).mass = 0.010645177326560149;  % kg
smiData.Solid(8).CoM = [4.8185483870967731 15.657594562637378
3.0403225806451588];  % mm
smiData.Solid(8).MoI = [4.0627966693272155 6.8408978425120033
9.9635988777203739];  % kg*mm^2
smiData.Solid(8).PoI = [0 0 0];  % kg*mm^2
smiData.Solid(8).color = [1 1 1];
smiData.Solid(8).opacity = 1;
smiData.Solid(8).ID = 'part5*:*Default';

%Inertia Type - Custom
%Visual Properties - Simple
smiData.Solid(9).mass = 0.016654938626886683;  % kg
smiData.Solid(9).CoM = [12.077891273051348 7.5249500464547392
2.8098474361814425];  % mm
smiData.Solid(9).MoI = [2.0070612580952285 8.9069644490136213
9.3963220749164709];  % kg*mm^2
smiData.Solid(9).PoI = [0 0 0];  % kg*mm^2
smiData.Solid(9).color = [1 1 1];
smiData.Solid(9).opacity = 1;
smiData.Solid(9).ID = 'Part9*:*Default';


%============= Joint =============%
%X Revolute Primitive (Rx) %Y Revolute Primitive (Ry) %Z Revolute Primitive
(Rz)
%X Prismatic Primitive (Px) %Y Prismatic Primitive (Py) %Z Prismatic
Primitive (Pz) %Spherical Primitive (S)
%Constant Velocity Primitive (CV) %Lead Screw Primitive (LS)
%Position Target (Pos)

%Initialize the RevoluteJoint structure array by filling in null values.
smiData.RevoluteJoint(6).Rz.Pos = 0.0;
smiData.RevoluteJoint(6).ID = '';

smiData.RevoluteJoint(1).Rz.Pos = -70.570600000000141;  % deg
smiData.RevoluteJoint(1).ID = '[MX-64R-1:-:FR05-H101-2]';

smiData.RevoluteJoint(2).Rz.Pos = 147.09159999999952;  % deg
smiData.RevoluteJoint(2).ID = '[MX-64R-3:-:bracket1-1]';

smiData.RevoluteJoint(3).Rz.Pos = 3.5160904781391919e-08;  % deg
smiData.RevoluteJoint(3).ID = '[FR05-S101-2:-:Part9-1]';

smiData.RevoluteJoint(4).Rz.Pos = -147.26759612467723;  % deg
smiData.RevoluteJoint(4).ID = '[Part1-1:-:part2-2]';

%This joint has been chosen as a cut joint. Simscape Multibody treats cut
joints as algebraic constraints to solve closed kinematic loops. The imported
model does not use the state target data for this joint.
smiData.RevoluteJoint(5).Rz.Pos = 138.35211185930379;  % deg
smiData.RevoluteJoint(5).ID = '[part2-2:-:Part6-1]';

smiData.RevoluteJoint(6).Rz.Pos = -36.718092016019305;  % deg
```

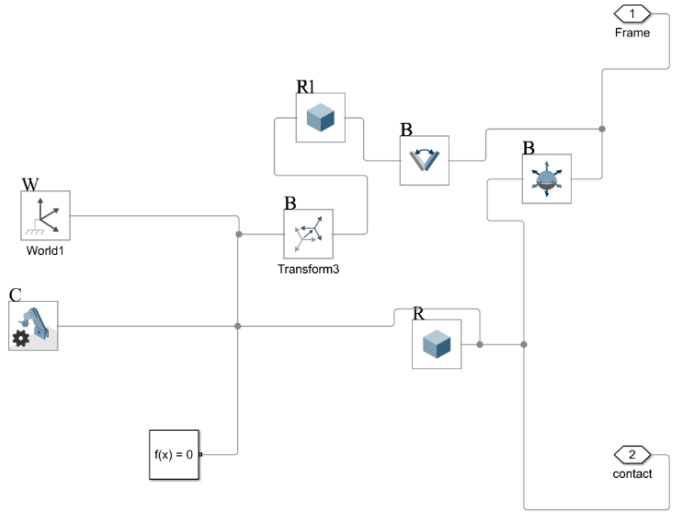
```
smiData.RevoluteJoint(6).ID = '[part5-1:-:Part6-1]';

sim('Assem')
figure(2)
plot(bx_new,by_new,'r')       %Plot fitted trajectory
hold on
plot(-x*1000,-y*1000,'--b')   %Plot the simulated trajectory
hold off
xlabel('x (mm)')
ylabel('y (mm)')
legend('Analysis Result','Simulation Result')
[theta_1,theta_2,theta_3,theta_4] = inverse(bx_new,by_new);
figure(3)
[torque1,torque4] = torque(12.5,25,theta_1,theta_2,theta_3,theta_4);
plot(time,-repmat(torque1,1,cycle)/1000,'r',time,-
repmat(torque4,1,cycle)/1000,'k')%Plot analysis torque
hold on
plot(tau1,'--g')
plot(tau4,'--b') %Plot the simulated torque
hold off
xlabel('Time (s)')
ylabel('Torque (Nm)')
legend('Analysis Result of Motor 2','Analysis Result of Motor 1',...
    'Simulation Result of Motor 2','Simulation Result of Motor 1')
theta_1_dot=[]
theta_4_dot=[]
for i=1:479999
    theta_1_dot(i)=(theta_1(i+1)-theta_1(i))*480000;
    theta_4_dot(i)=(theta_4(i+1)-theta_4(i))*480000;
end
theta_1_dot(480000)=(theta_1(1)-theta_1(480000))*480000;
theta_4_dot(480000)=(theta_4(1)-theta_4(480000))*480000;
figure(4)
plot(time,-repmat(theta_1_dot,1,cycle),'r',time,-
repmat(theta_4_dot,1,cycle),'k')%Plot analysis angular velocity
hold on
plot(t1d,'--g')
plot(t4d,'--b') %Plot the simulated angular velocity
hold off
hold off
xlabel('Time (s)')
ylabel('Angular velocity (Nm)')
ylabel('Torque (Nm)')
legend('Analysis Result of Motor 2','Analysis Result of Motor 1',...
    'Simulation Result of Motor 2','Simulation Result of Motor 1')
```

Appendix B: Simscape Multibody model

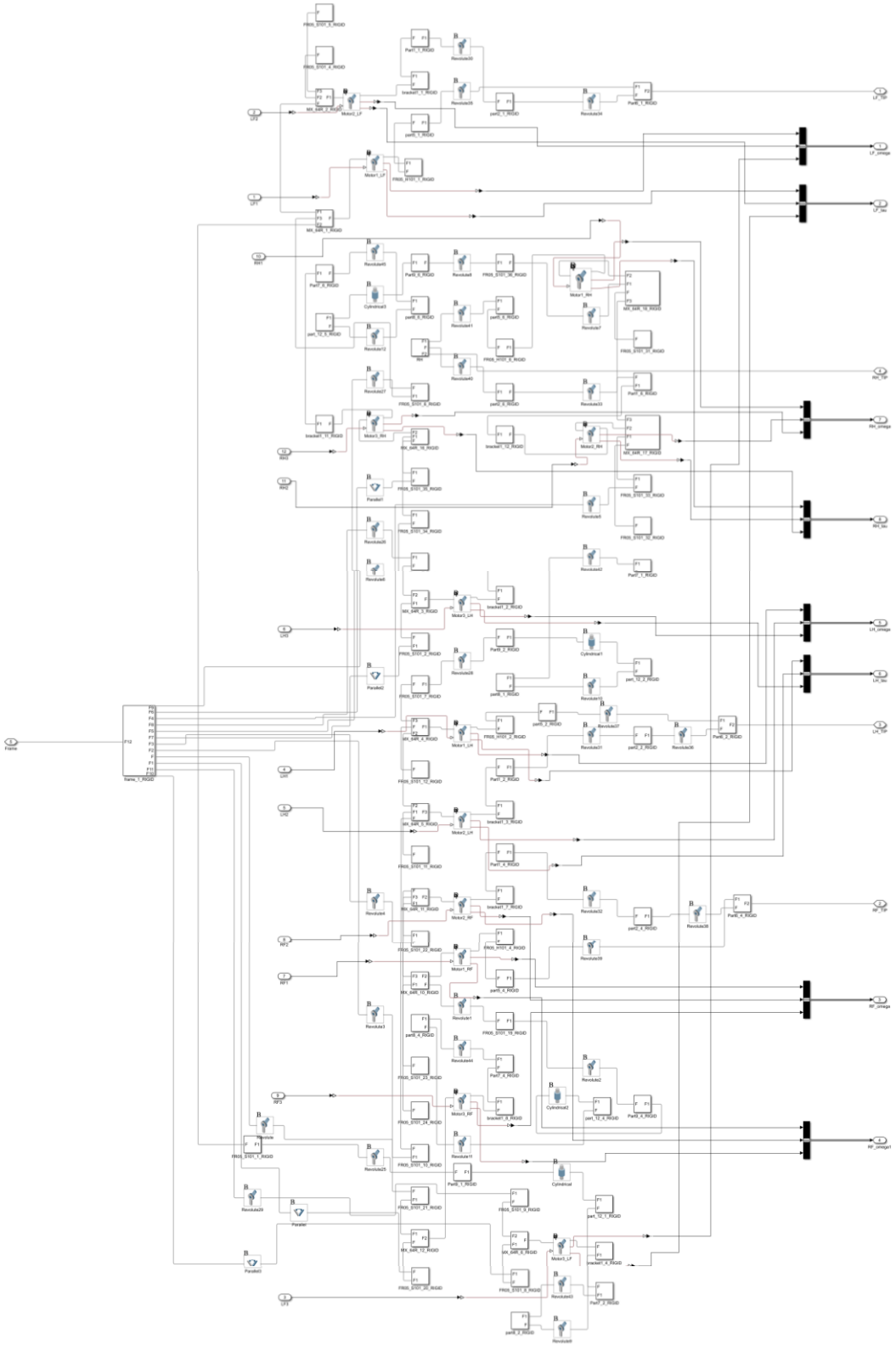
Input signal subsystem



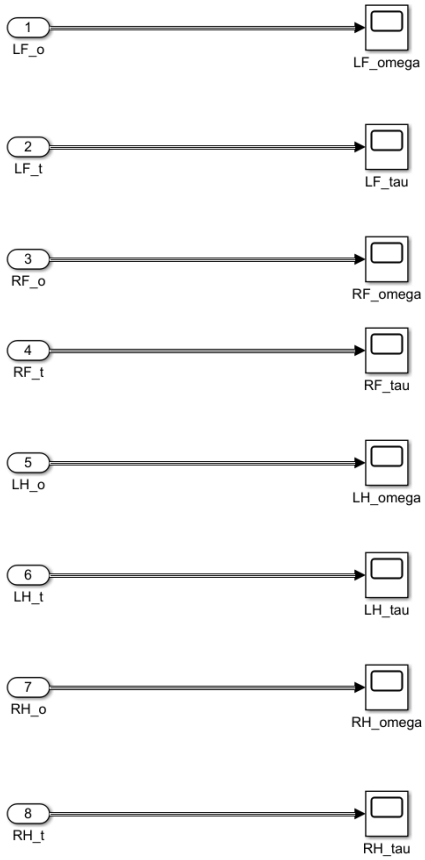
Solver subsystem



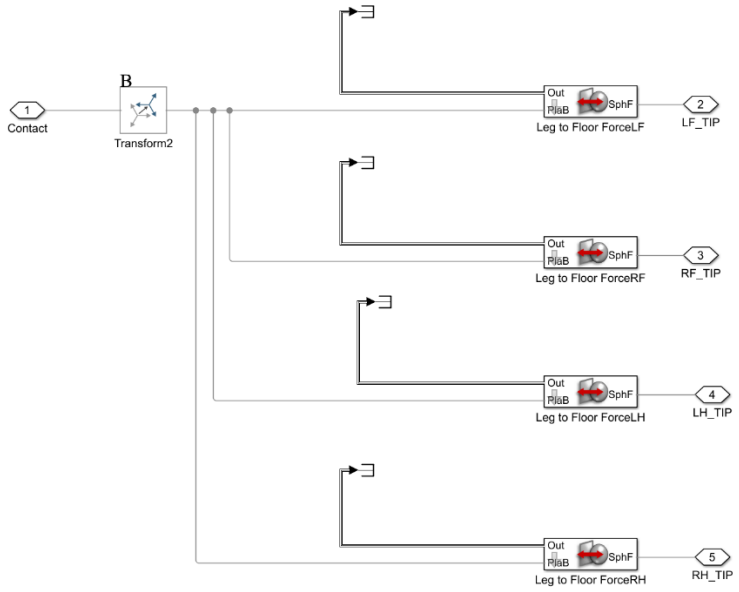
Quadruped robot model imported from SOLIDWORKS



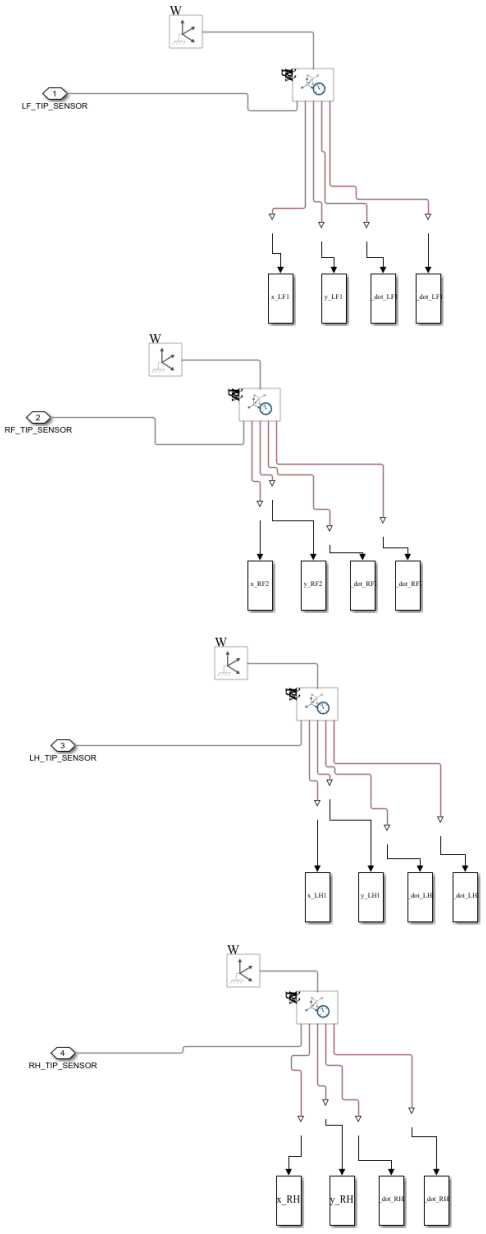
Actuator angular velocity and torque subsystem



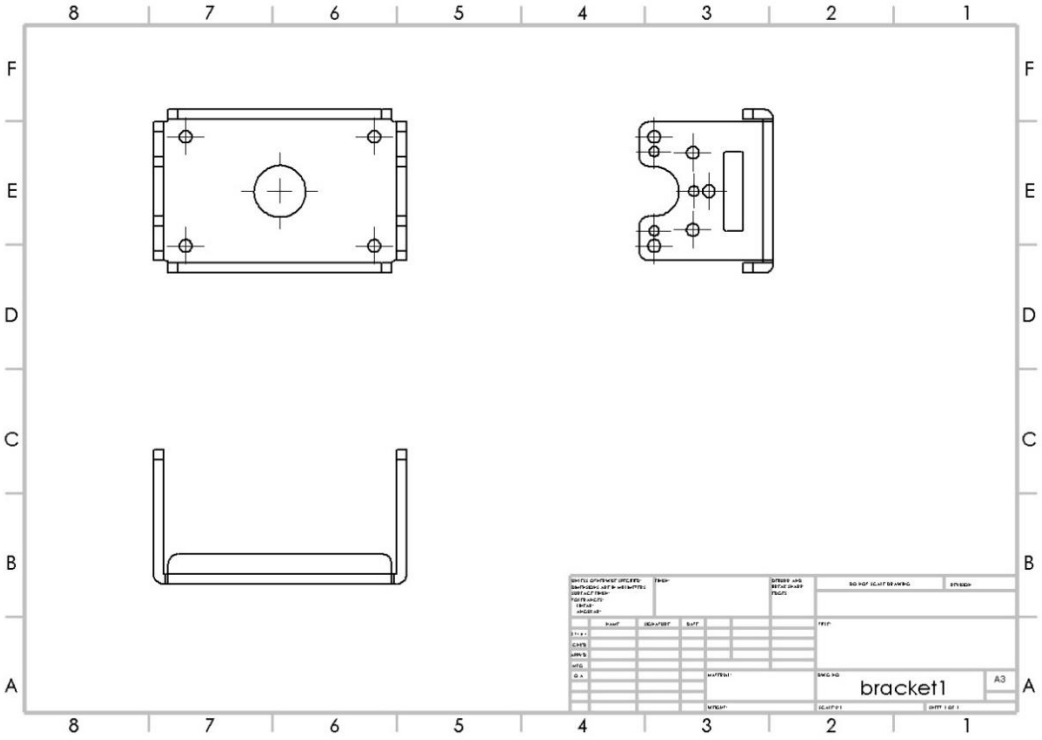
Contact force subsystem

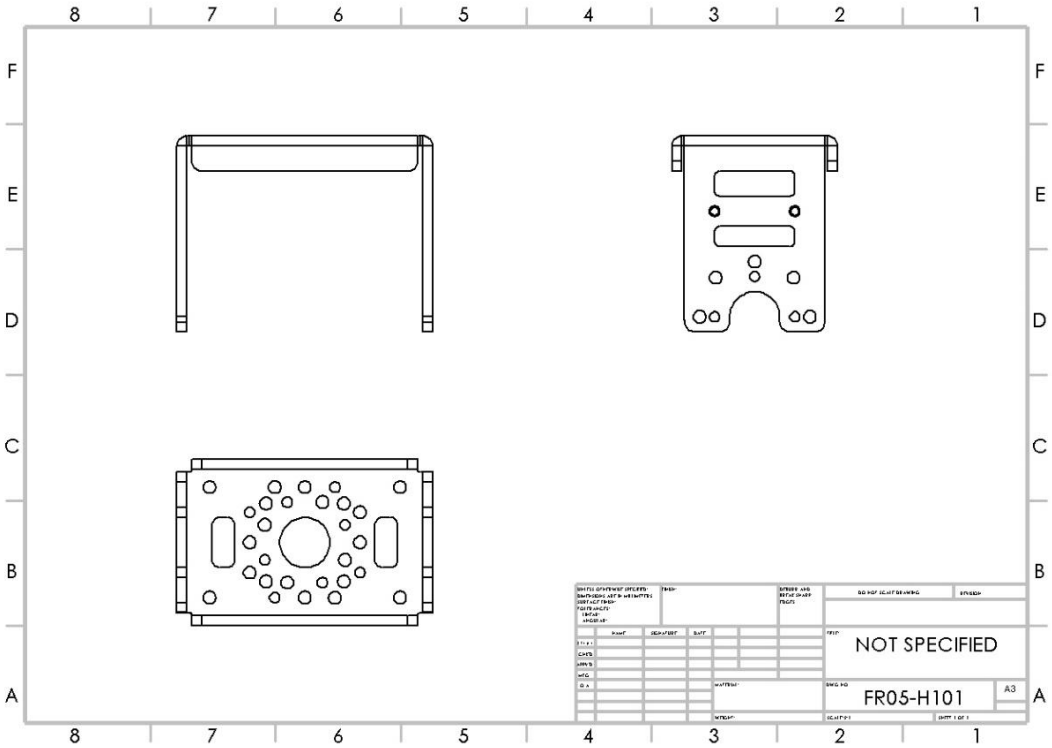


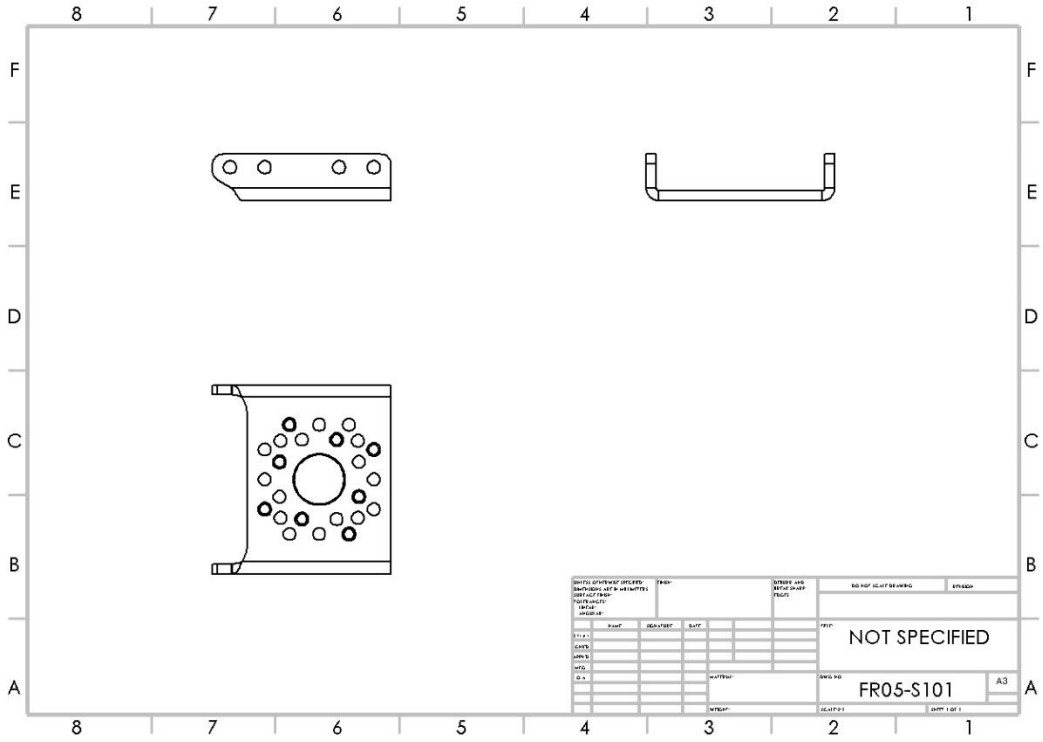
Leg tip position and velocity sensor subsystem

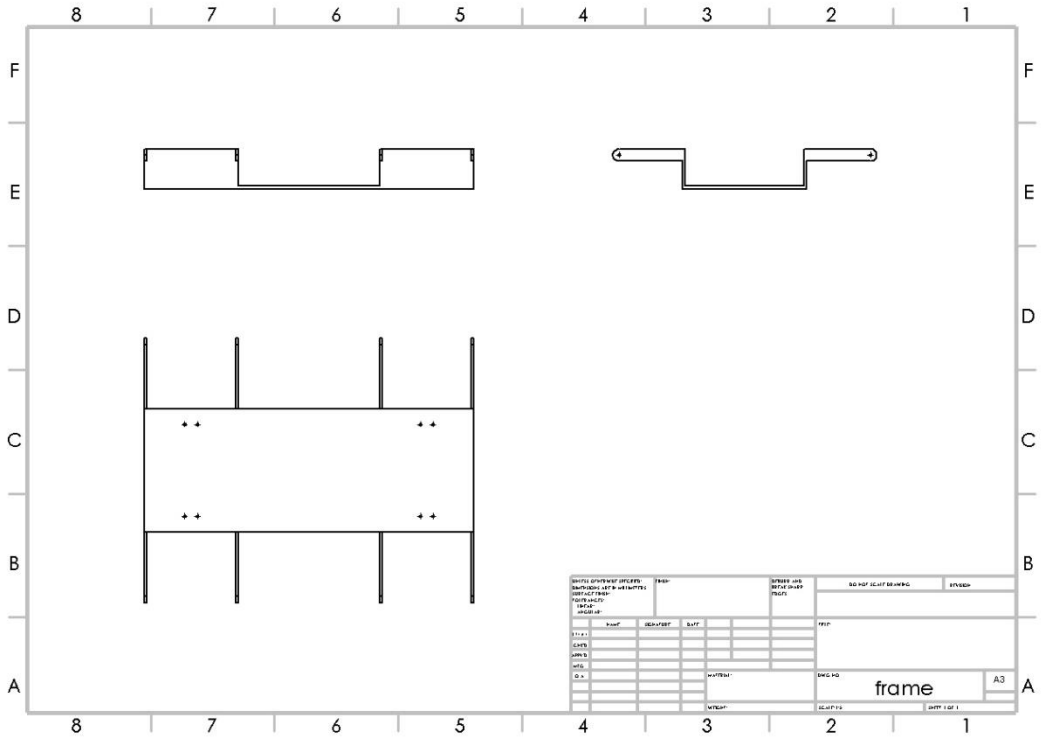


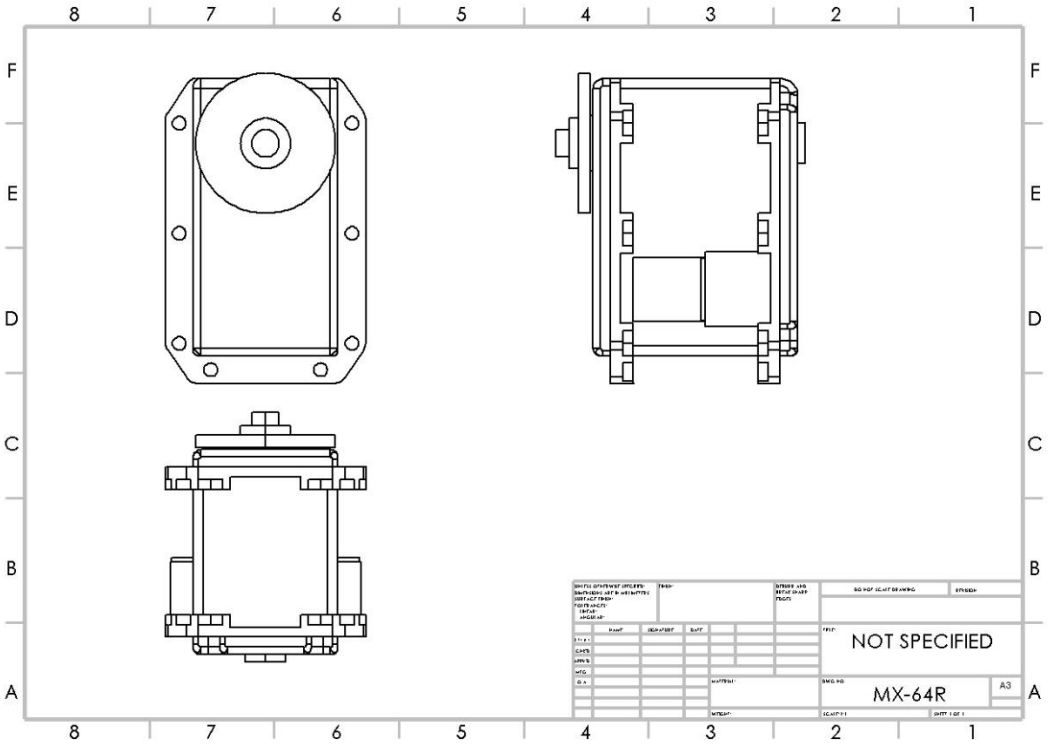
Appendix C: Solidworks model

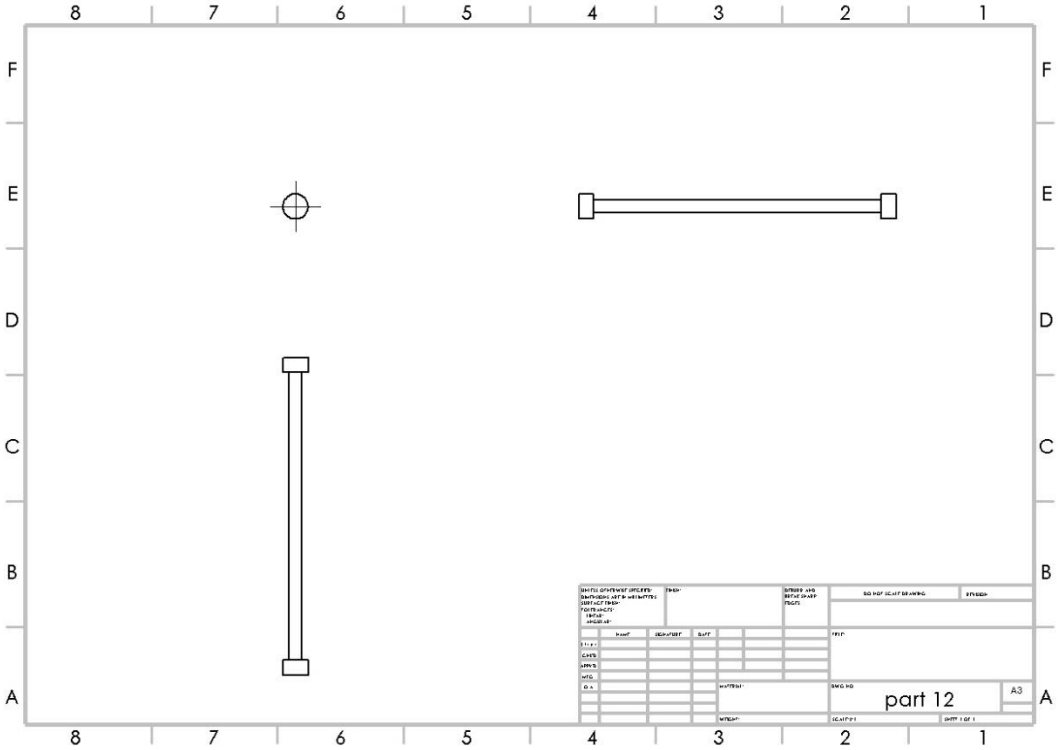


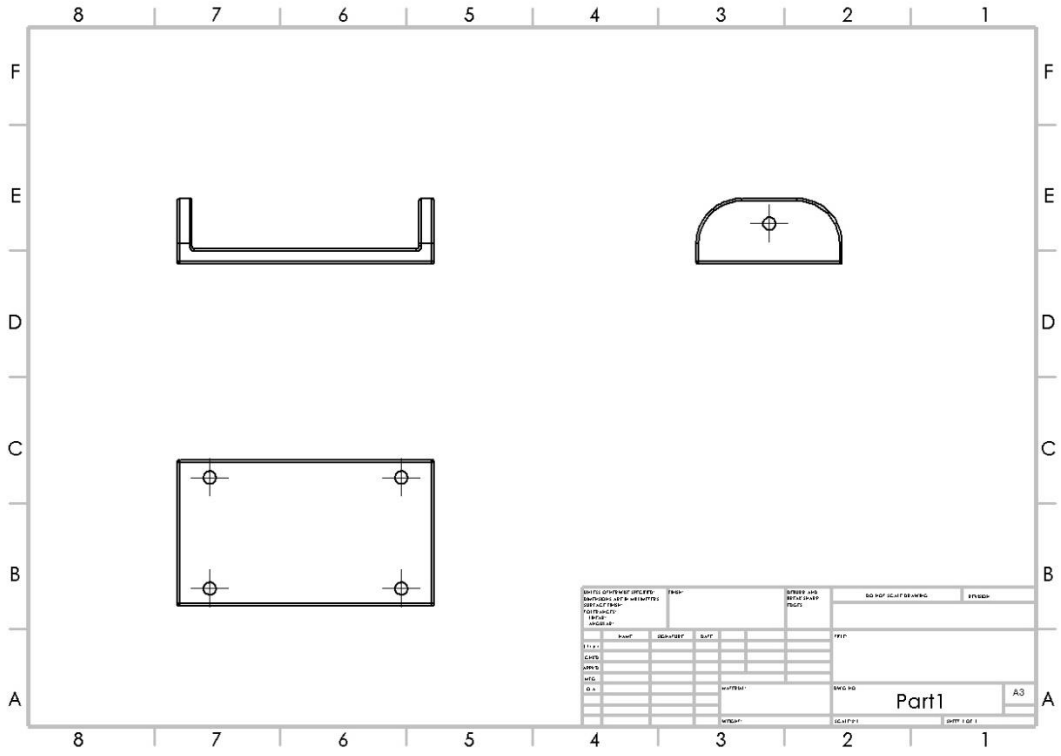


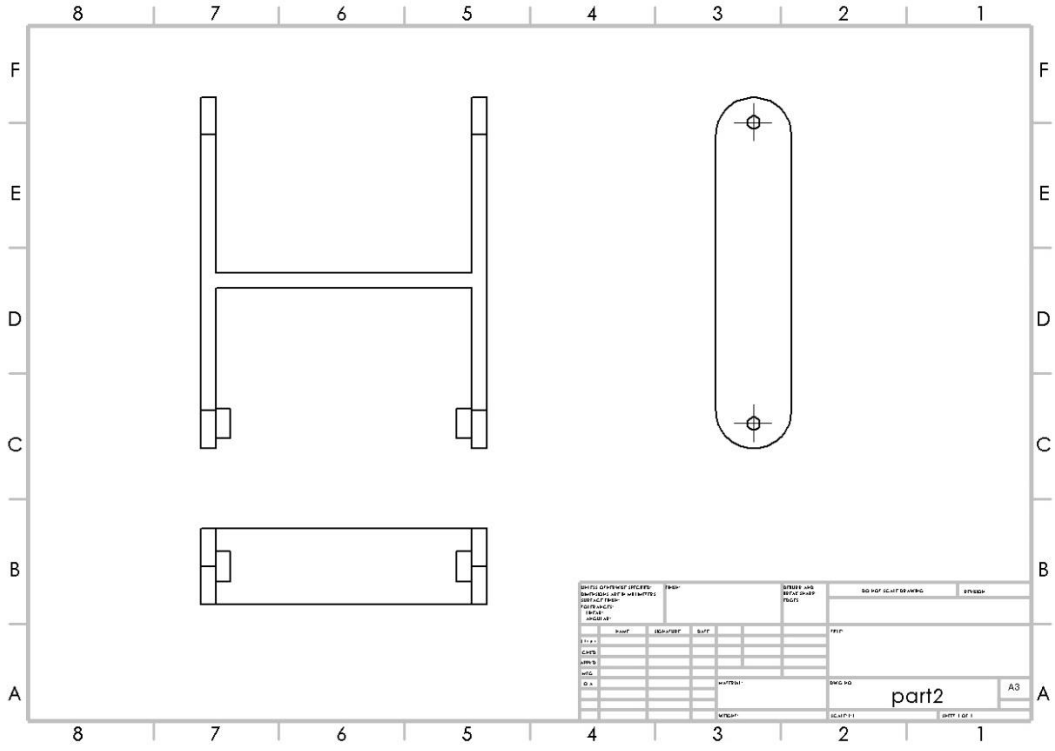

NOT SPECIFIED
FR05-S101
A3

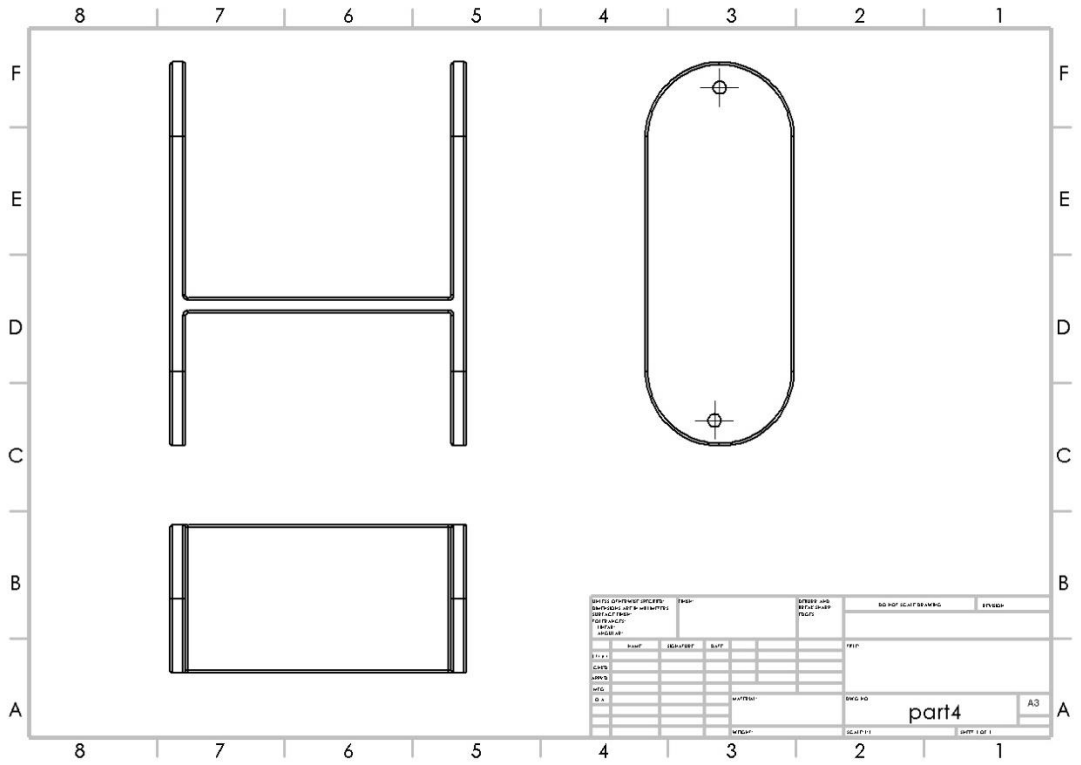

frame
A3

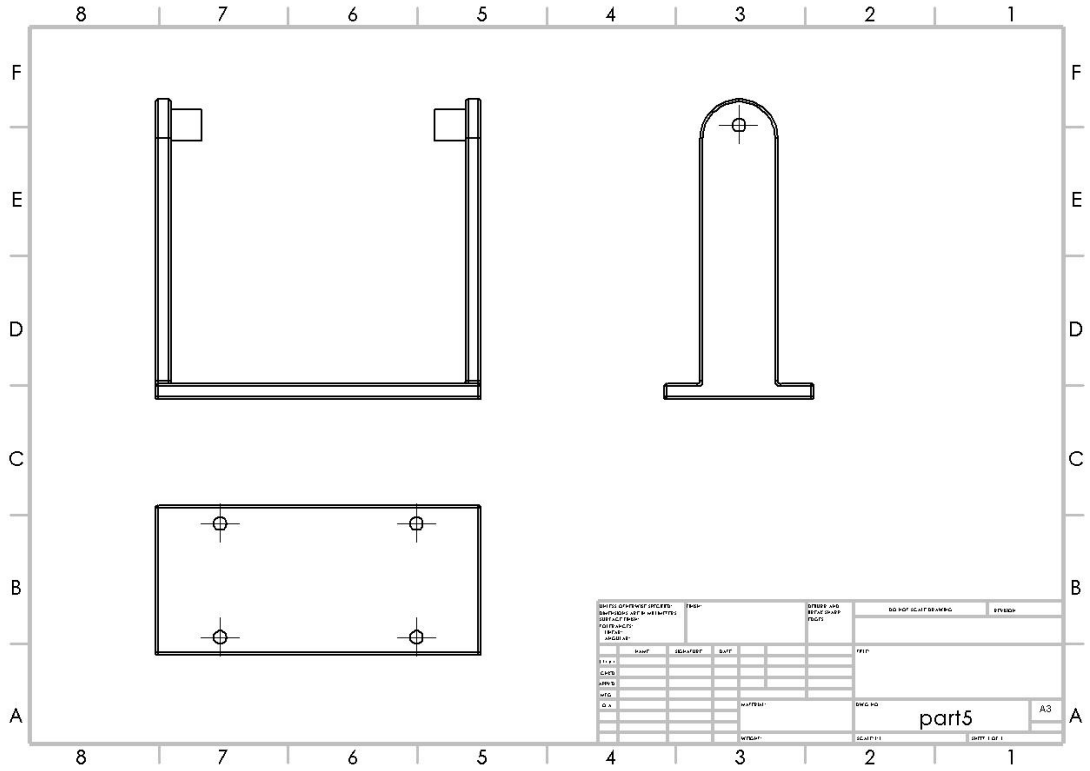

NOT SPECIFIED
MX-64R
A3

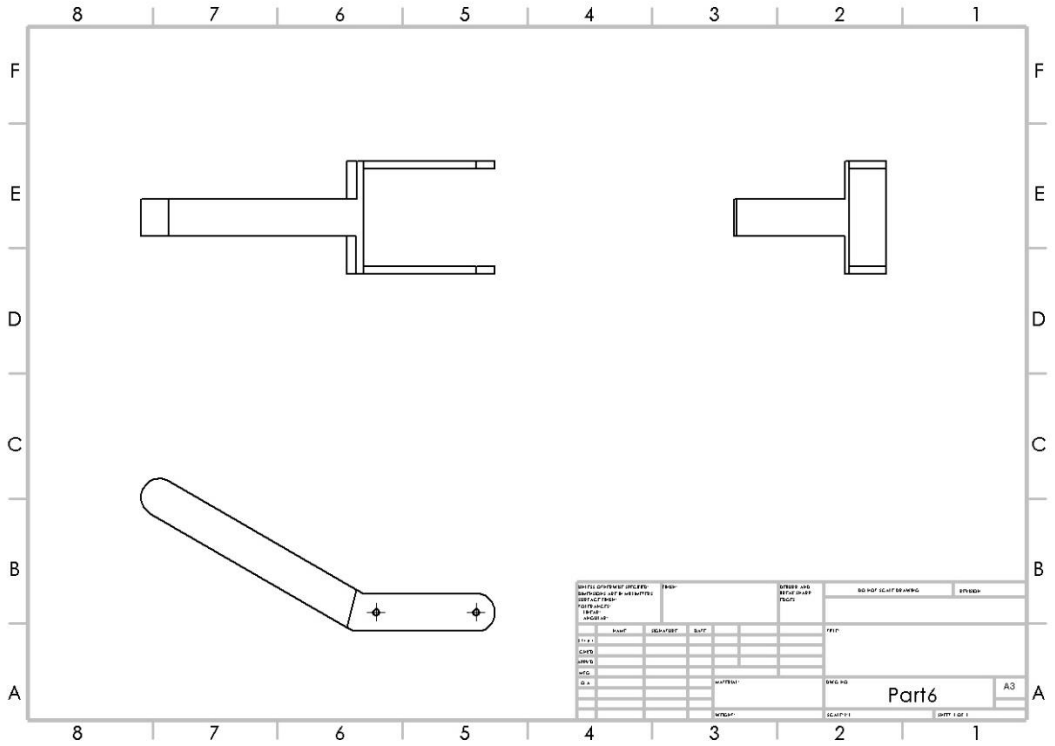

part 12
A3

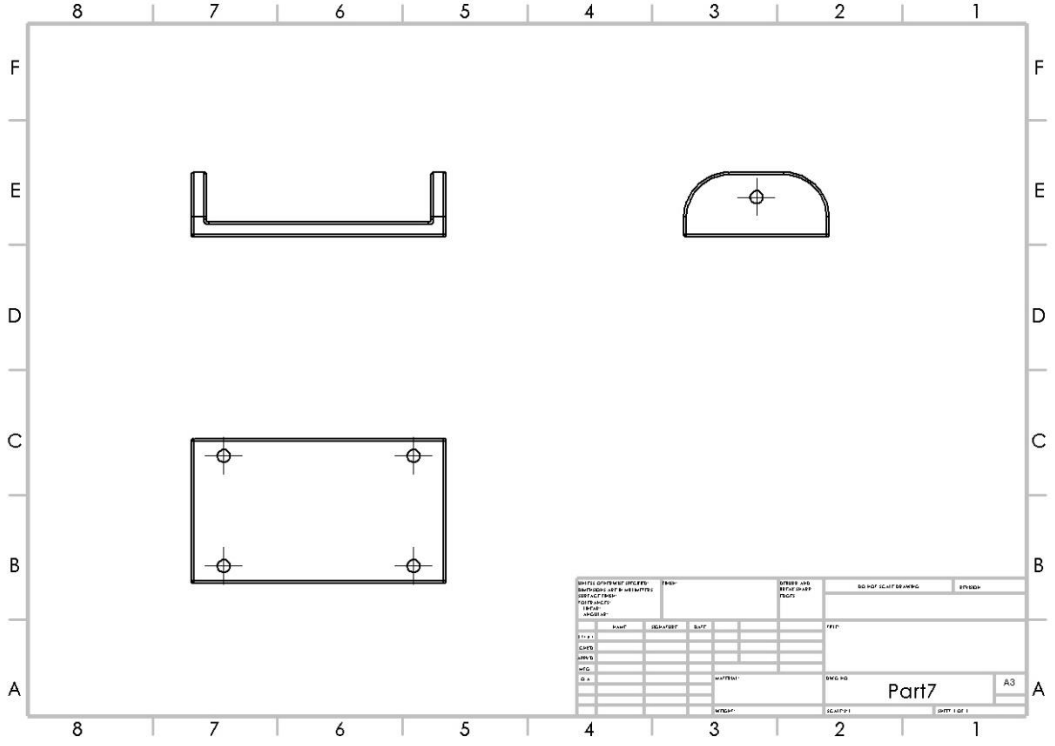

Part1
A3

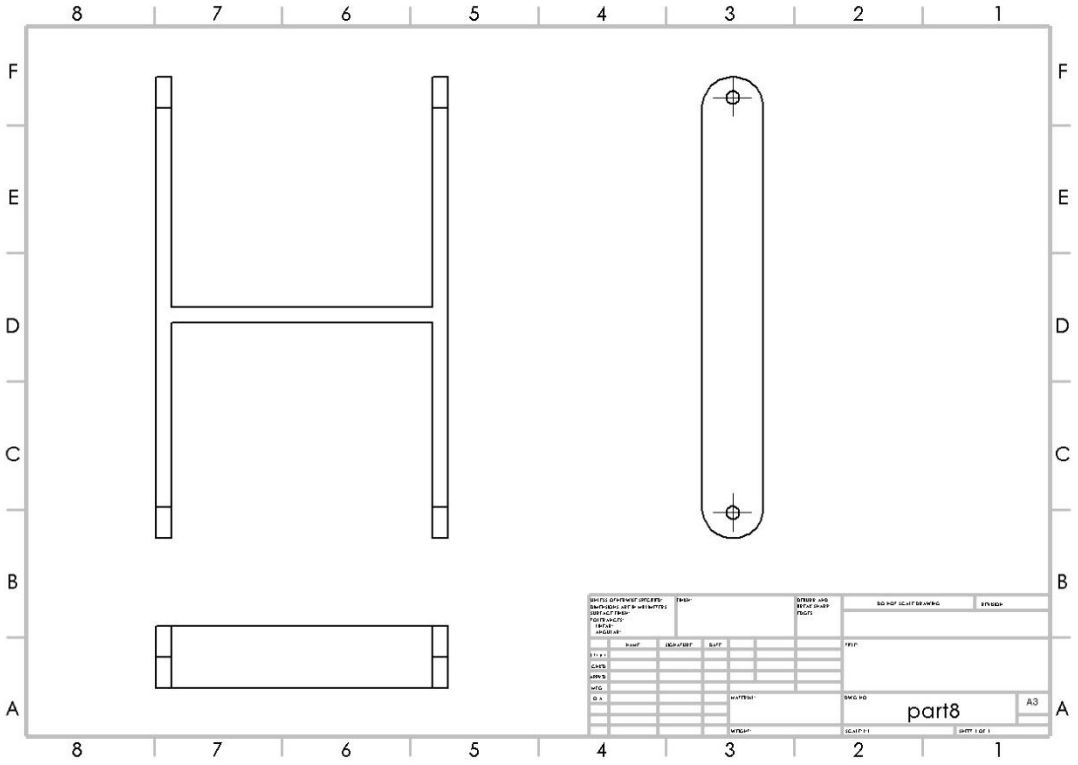

part2
A3

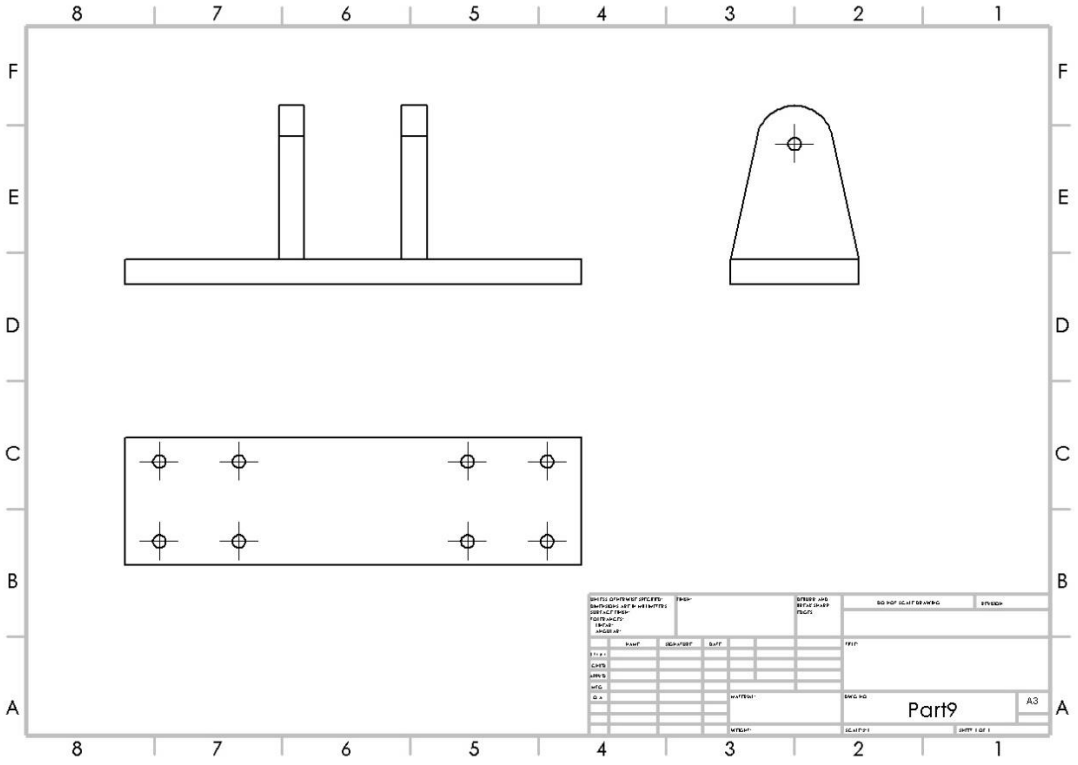

8 7 6 5 4 3 2 1
F E D C B A
part4
A3

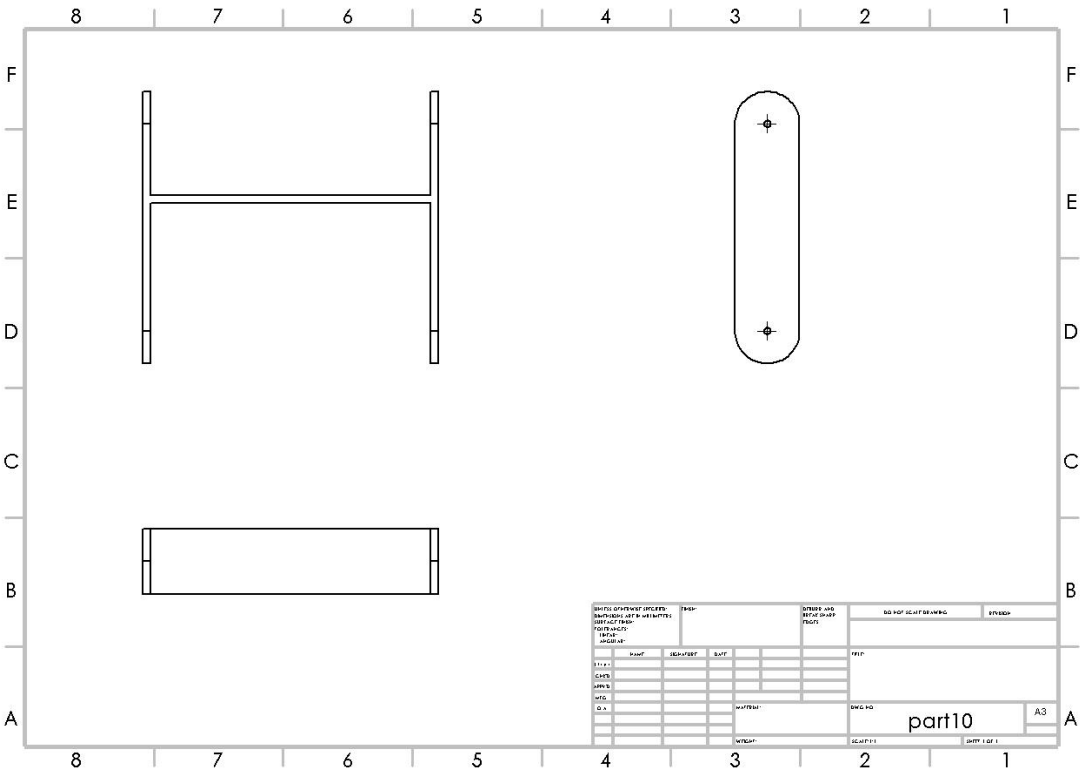

8 7 6 5 4 3 2 1
F E D C B A
part5
A3

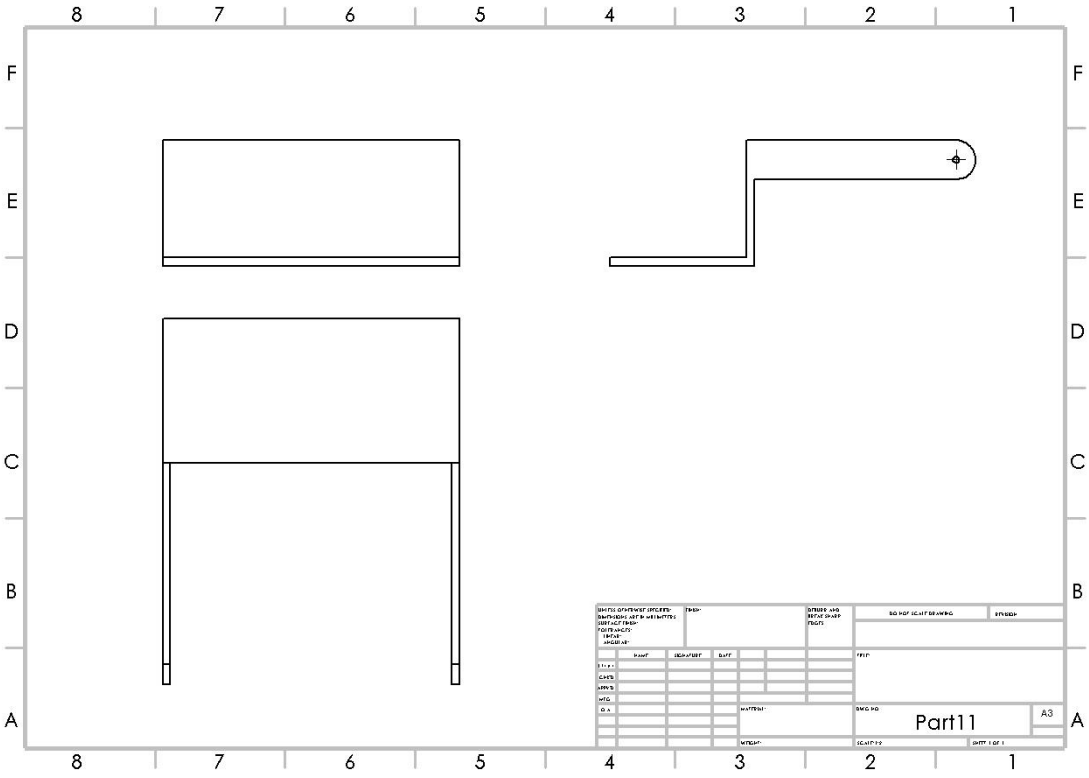

8 7 6 5 4 3 2 1
F E D C B A
Part6
A3


8 7 6 5 4 3 2 1
F E D C B A
Part7
A3


part8
A3


Part9
A3


8 7 6 5 4 3 2 1
F E D C B A
part10
A3


8 7 6 5 4 3 2 1
F E D C B A
Part11
A3