DOKUZ EYLÜL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DESIGN AND ANALYSIS OF A SPIDER ROBOT

by

Ali ŞENOL

August, 2019

İZMİR

DESIGN AND ANALYSIS OF A SPIDER ROBOT

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science
in Mechatronics Engineering

by

Ali ŞENOL

August, 2019
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"DESIGN AND ANALYSIS OF A SPIDER ROBOT"** completed by **ALİ ŞENOL** under supervision of **PROF. DR. ZEKİ KIRAL** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Zeki KIRAL

Supervisor

Dr. Öğr. Üyesi Şahin YAVUZ

(Jury Member)

Dr. Öğr. Üyesi Özgün BAŞER

(Jury Member)

Prof.Dr. Kadriye ERTEKİN
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

First of all, I wish to express my sincere regards to my supervisor, Prof. Dr. Zeki KIRAL for his guidance, moral and material supports, his suggestions and his patience. It was a privilege and honor to me to study with his supervision for my master program. I wish to express my sincere regards to Assist. Prof. Dr. Taner AKKAN for his valuable discussions and guidance about the program codes.

Additionally, I dedicate this thesis to my family for their unconditional love and supports.

Ali ŞENOL

DESIGN AND ANALYSIS OF A SPIDER ROBOT

ABSTRACT

Wheel or palette type robots are usually used for explorations on flat surfaces because of simplicity of their mechanical, electronic and software implementations. But these robots are not quite applicable on rough terrains. Legged type robots are not fast as wheel types but have abilities to be operated successfully in rough terrains such as rocky or sandy areas, narrow paths, etc. Biped robots have relatively simple structures but they are hard to be balanced. On the other hand, as the number of legs increases the mechanical complexity also increases for multi-legged robots.

In this study, it is aimed to develop an eight-legged robot with minimal mechanical and cabling complexity. Eight-legged explorer robot (ELER) was designed in SolidWorks and Matlab Simmechanics softwares and then succesfully be implemented using smart servo motors and 3D printing technique.

Keywords: Explorer robots, multi-legged robot, human-robot interface, smart servo motor

BİR ÖRÜMCEK ROBOTUN TASARIMI VE ANALİZİ

ÖZ

Tekerlek veya palet tipi robotlar genellikle mekanik, elektronik ve yazılım uygulamalarının basitliği nedeniyle düz yüzeylerdeki araştırmalarda kullanılır. Ancak bu robotlar engebeli arazilerde kolaylıkla uygulanabilir değildir. Bacaklı tip robotlar tekerlek tipi kadar hızlı değildir, ancak kayalık ya da kumlu alanlar, dar yollar vb. gibi zorlu arazilerde başarılı bir şekilde çalıştırılabilmeleri için gerekli yeteneklere sahiptir. Biped robotlar göreceli olarak basit yapılara sahiptir fakat dengelenmeleri güçtür. Diğer taraftan çok bacaklı robotlarda bacakların sayısı arttıkça, mekanik karmaşıklık da artmaktadır.

Bu çalışmada, en az mekanik ve kablolama karmaşıklığına sahip sekiz bacaklı bir robotun geliştirilmesi amaçlanmıştır. Sekiz bacaklı kaşif robotu (ELER) SolidWorks ve Matlab Simmechanics programları ile tasarlanmış ve akıllı servo motorlar ve üç boyutlu yazdırma tekniği kullanılarak başarıyla uygulanmıştır.

Anahtar Kelimeler: Kaşif robotlar, çok bacaklı robot, insan-robot arayüzü, akıllı servo motor

CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER ONE
INTRODUCTION

Exploring rough terrains and hazardous areas is a challenging issue for mobile explorer robotic designs (Billah, Ahmed & Farhana, 2008). Wheel or palette type robots are usually used for explorations on flat surfaces because of simplicity of their mechanical, electronic and software implementations. But these robots are not quite applicable on rough terrains. Legged style robots are used to cope with these challenges because of their climbing and movement capabilities (Rohmer, 2009).

In this study, an **E**ight-**L**egged **E**xplorer **R**obot (ELER) was mechanically designed and implemented. ELER's goal is to achieve to help search and rescue missions where natural disasters happen or general map exploring tasks. Eight-legged design was selected to mimic the spider's ability of advancing on nearly every terrain and overcoming surrounding obstacles. Multi-legged robotic systems are hard to implement because of the mechanics, electronics and software concerns (Tadeschi & Carbone, 2014). Therefore, this study is a very good optimization problem and didactic process for a mechatronics project.

Intelligent servo motors were used to simplify the mechanical design and ease the power and signal cabling. These intelligent motors also provide sensory feedback such as voltage, current, power, torque and motor temperature values to the electronic control unit (Slatour, 2016). 24 smart servo motors were used for the mechanical leg movements. There are 3 servo motors for each leg and the motors are placed in a way to imitate the real life spider. Using smart servos enables the cables to go through one servo to another, thus it makes cabling process and servo control from one link easy. HC-06 Bluetooth 2.0 module (Bluetooth Transceiver Module HC-06, 2017) was used to communicate with the robot. Arduino Mega platform was used to control the mechanical system, the battery system and the Bluetooth communication control system.

First, the mechanical body and the leg parts are designed using SolidWorks and kinematic analysis was performed in Matlab Simulink. Then the mechanical parts were printed using a 3D printer. Parts and the motors were assembled together. The cabling between motors, motor controller interface, Arduino Mega, SD card module, LCD module, Bluetooth module and the battery were performed maintaining minimum amount of cables. Software was initially built for each component separately, then integrated in a one robust software using Arduino C coding. PC serial channel is used for software uploading, testing and debugging purposes.

The basic leg and robot movements are developed using this PC communication interface. For enabling mobile control, a human machine interface (HMI) was built on Android mobile phone by using "MIT App Inventor 2". The robot's three dimensional movement (x, y, z and rotational movements) can be controlled with touch buttons on the screen. Some basic initial robot poses, robot base vertical ascending and descending codes are loaded using simple known robotic patterns. The other complex movements are first build with position teaching with moving the legs like a puppet. Simple human robot cooperative interaction for movement teaching was resulted an easy and effective way of robot movement programming. It is seen that teaching method eases the software development and the resultant robotic movements are satisfactory. As a result, eight-legged explorer robot electronic control software and human-robot interface software were developed. A simplified mechanical robotic system, efficient electronic system and control software were implemented successfully. Below, a brief information was given for the robotic systems.

1.1 Robots

A robot is a mechatronic system, which has mechanical and electronic parts, sensors, and a software runs on a specifically designed computer (Angeles, 2014). This system carries out pre-programmed complex time based actions considering the environmental factors using latest technological smart sensors. Robot operation and guidance can be done manually by human operator or automatically by robot itself.

The robot and human interaction usually performed by using input and output devices such as keyboard and mouse, joysticks, various controllers, touchscreens etc. Today robots even can be controlled like human-human interface such as auditory and visual communications. Robots can be in various constructions from basic manipulators to animal and human appearance.

1.2 Explorer Robots

Explorer robots are sophisticated machines that have sensory systems and are remotely controlled or controlled by preprogrammed onboard computers. (Miller & Miller, 2017). Explorer robots are used to go where humans cannot go or fear to tread. For instance, they are used in outer space probes, to explore caves and to dive far deeper underwater than humans can, and they can be used to rescue people in sunken ships. An example is Thai cave rescue operation in 2018, Elon Musk tried to help the rescue mission by bringing child-sized robot-like submarine, even though the submarine was not needed the will to help is appreciated.

1.3 Multi-legged Robots

Multi-legged robots are robots with human or animal like legs, which uses those legs for movement or other objectives. Most common multi-legged robots are two legged (biped) robot, four-legged (quadrupedal) robot, six-legged (hexapod) robot and eight-legged (octopod) robots (Machado & Silva, 2006).

Wheeled systems go very fast, but speed is not always the metric in measuring the performance. Wheels don't always go where feet do. And speed in linear travel does not translate to speed in change of pose/principle orientation.

Multi-leg robots are flexible and can travel where wheels or tracks cannot, it can even climb to a certain degree while wheel designs fail (Wettergreen & Thorpe, 1992).

1.3.1 Eight-Legged (Octopod) Robots

Eight-legged designs are generally inspired from spiders and arachnids and in this study, ELER is inspired from spiders. Eight-legged design was selected to mimic the spider's ability of advancing on nearly every terrain and overcoming surrounding obstacles. Eight-legged design is easier to balance than two or four-legged robots due to the number of legs it has. There are more than one walking pattern for multi-legged robots as shown in Figure 1.1 and Figure 1.2 and they are called as gaits.

1.3.1.1 Wave Gait

In wave gait the legs move forward one at a time, starting from the rear leg from one of the each side. Movement starts from the rear leg and finishing at front leg, after one side finished moving the other side repeats the pattern.

Since only one leg is active at any given time while the rest of the legs stay down, it is a very stable motion but lacks in speed (Inagaki & Kobayashi, 1994). This gait can be applied on both six or eight-legged designs.

1.3.1.2 Tripod Gait

It is one of the most used gaits in hexapod (6-leg) design and it can also be used at octopod (8-leg) design with little tweaks to it. The movement finishes in two sequences, front and rear leg on one side and middle leg at the other side moves forward at the same time, after the first sequence finishes the remaining legs (opposite of the moved legs) moves forward. Since while moving, three legs are always remains contact with the ground the gaits name is called tripod. While moving the center of gravity is always at the middle and the system is statically and dynamically stable (Hasnaa & Mohammed, 2017).

1.3.1.3 Ripple Gait

More complicated than wave and tripod gait but similar moving pattern to tripod gait (Pabbu, 2017). Legs are moving one by one in this gait, and different from previous gaits there is a phase angle of 180 degree instead of 360 degree. While one leg is in the middle of its forward action the second leg starts to move and while second leg is in the middle of its forward motion first one finishes its movement while third one starts. Leg pattern is similar to tripod but instead of two there is six sequences, if the movement starts from right side moving sequence is like that; right front leg, left middle leg, right rear leg, left front leg, right middle leg and lastly left rear leg.

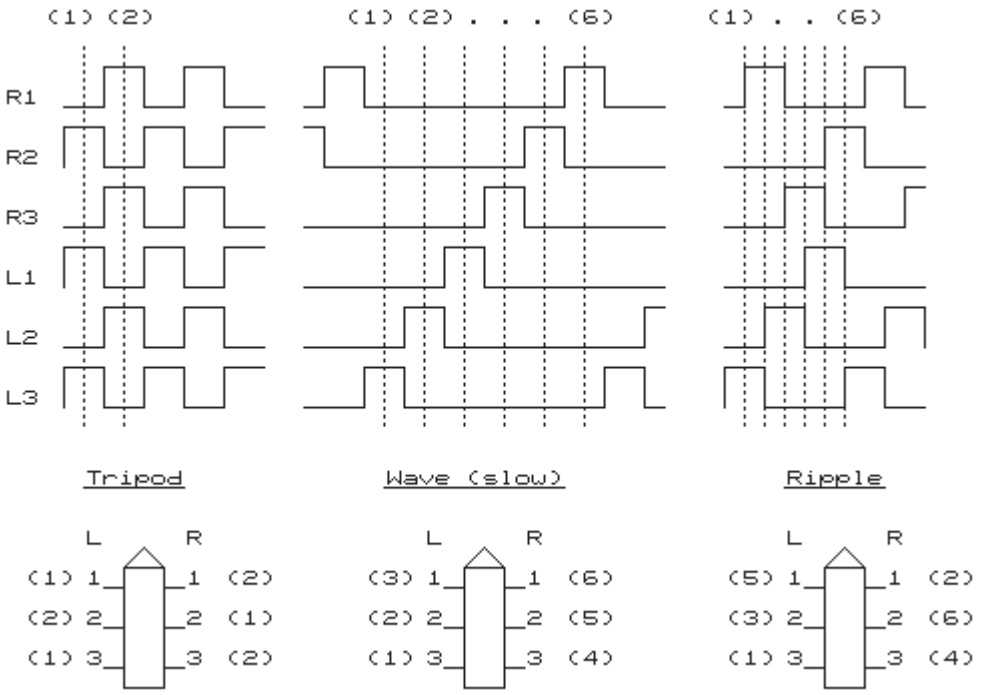


Figure 1.1 Tripod, wave and ripple gaits (Zak & Rozman, 2015)

1.3.1.4 Scorpion Gait

Living scorpions' gait has always an alternating tetrapod scheme (Figure 1.2, left). Legs, which belong to a tetrapod, beginning with the posterior leg, step in an anterior direction and each leg lags the preceding one by ten percent of the whole cycle

(phase). The other tetrapod starts the wave of motions when the first one is at the middle of its cycle. A cycle is a combination of a leg promotion period (swing) and a leg redemption period (stance). Spider robot's motion includes phase, swing and stance parameters (Figure 1.2, right) together with the step size of a leg (Klaasen, Linnemann, Spenneberg & Kirchner 2002).

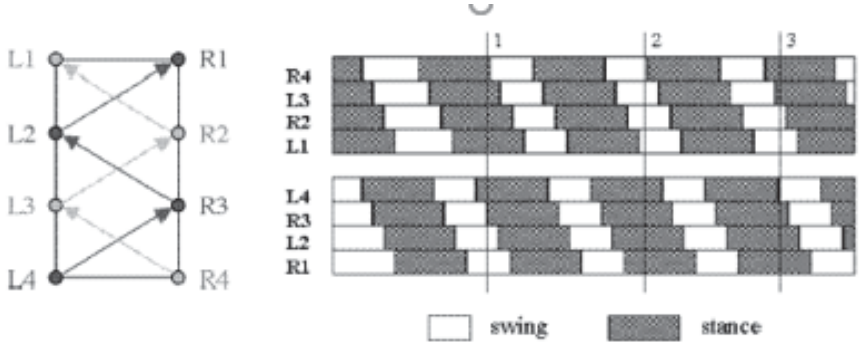


Figure 1.2 Scorpion gait (Klaassen, Linnemann, Spenneberg & Kirchner, 2003)

1.3.1.5 ELER's Gait

ELER's gait is unique, it uses a walking method of combination of tripod and scorpion walking gaits. As in the scorpion gait leg sequence is R1-L2-R3-L4 and then L1-R2-L3-R4 but as in tripod gait 4 legs move at the same time and after those 4 legs finish their movement other 4 legs start to move. The gait of the ELER can be drawn as shown in Figure 1.3.

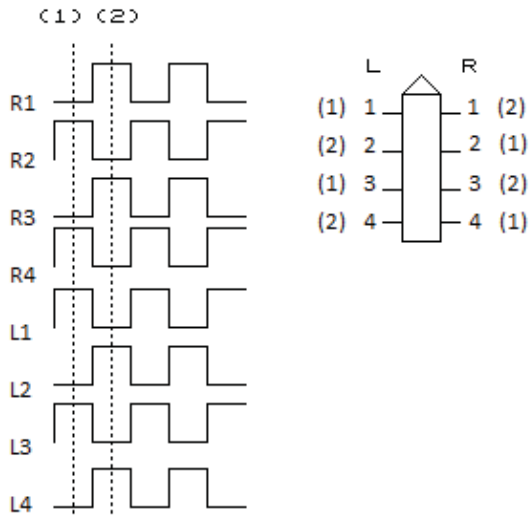


Figure 1.3 ELER's gait

1.4 Thesis Outline

In the scope of this thesis, a spider like robot was designed and realized. The mechanical design of the robot is based on a previous study presented in the literarure (Zhang, Liu, Zhao, Chen & Yan 2014) and new original design was created.

In chapter two, mechanical and electrical design of the spider robot is discussed, the modelling of the parts, placements of the sensors and motors is shown and their purposes in the design are described.

In chapter three, the simulink model of the ELER is given, the basic elements in the program are described and the results are shown.

In chapter four, the electronics design of the ELER is given.

In chapter five, the control software of the robot is described. Walking algorithms of the robot are described and the parts of the software are shown as well as the IMU sensor and the motor information.

In chapter six, mechanical and electronical outputs are given with some discussions.

Chapter seven is the closing section of the thesis and the unique properties of the robot are described and some suggestions for the future improvements are recommended.

CHAPTER TWO
MECHANICAL DESIGN

In this chapter ELER's mechanical design were explained in details. After giving some details of the mechanical parts used in the robot, the overall mechanical design was explained. The mechanical design was performed in SolidWorks and the parts were manufactured using a 3D printer. The main important part of the robot is the smart RC servo motor and the mechanical design of the robot main body and the legs were built with reference to the motor dimensions. Placement of the electronic system to the main body was also considered at the mechanical design stage.

2.1 Mechanical Parts

2.1.1 Servo Motors

RC servo motors are used for the joint movements of spider robots. The standart servo motor controls require three cables connection from the servo control circuitry. In this study, 24 RC servo motors are needed because eight legs have three joints. 24x4 different PWM signal cables are needed even if the power and ground lines are connected to each other. This action results too messy cable attachments. Moreover, the control circuitry has 24 different PWM signal generation. To cope with this problem, a smart servo motor was selected. All signal lines are connected to each other to form a network. In this configuration, we don't need 24 different PWM signal generation.

2.1.1.1 Standart Servo Motor

The design of the model started evolving after concluding which servo motor has to be used. At first Futaba S3004 standart servo was chosen (Figure 2.1) and design is started get a shape. But the servo motor had some problems.

These servo motors could not be connected serially since it creates cable cluttering. Since the 24-channel PWM cannot be received from the Arduino mega outputs, it requires a servo driver board for 24 channels. There is also a need for distributor electronic card design.

Figure 2.1 Futuba servo and the basic dimensions (Futuba specs, n.d.)

2.1.1.2 Smart Servo Motor

Finally, Feetech SCS15 servo motor (Figure 2.2) is choosen to actuate the spider robot. Since there are 8 legs and each leg has 3 servo motors as joints, 24 servo motor are used and controlled at the same time.

Figure 2.2 SCS-15 Servo Motor, cable and fixing apparatus (Feetech SCS15, n.d.)

It’s dimensions are 40.0×20.0×40.5 (in mm) and the mass of it is 56 grams. Servo motors are purchased with their own gears and aluminum brackets.

2.1.2 Servo Mount Brackets

Servo motor brackets are used for connecting servos and other parts. They are made of aluminum and tough. Techincal drawing of the bracekts are given in Figure 2.3 and Figure 2.4.

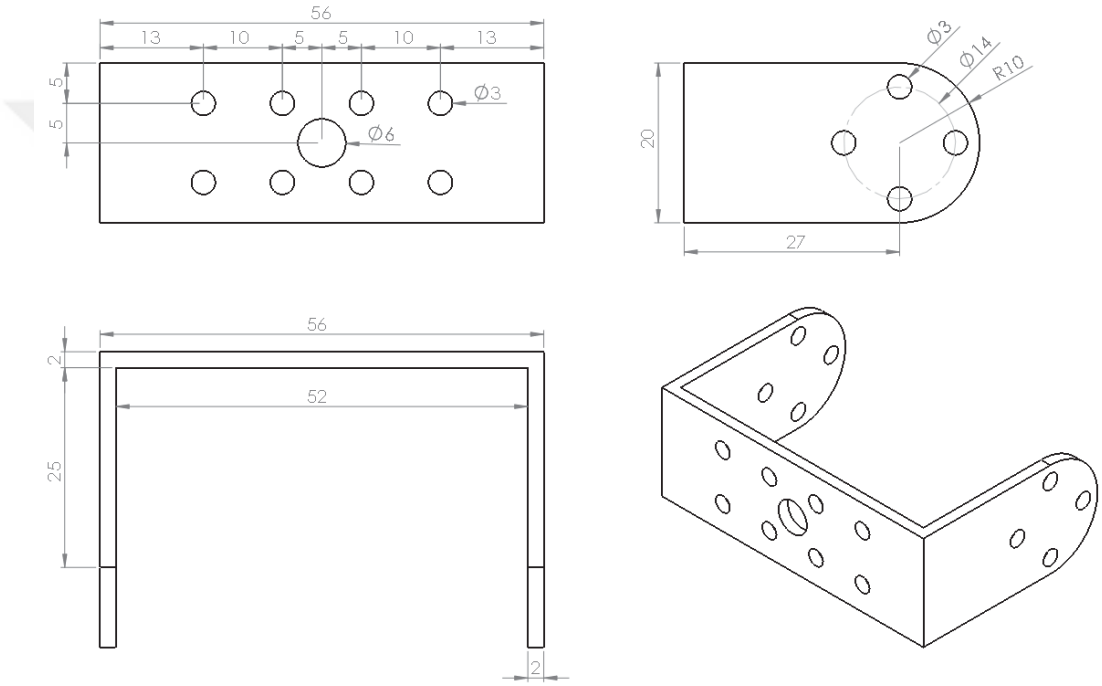


Figure 2.3 Servo mounts and the basic dimensions

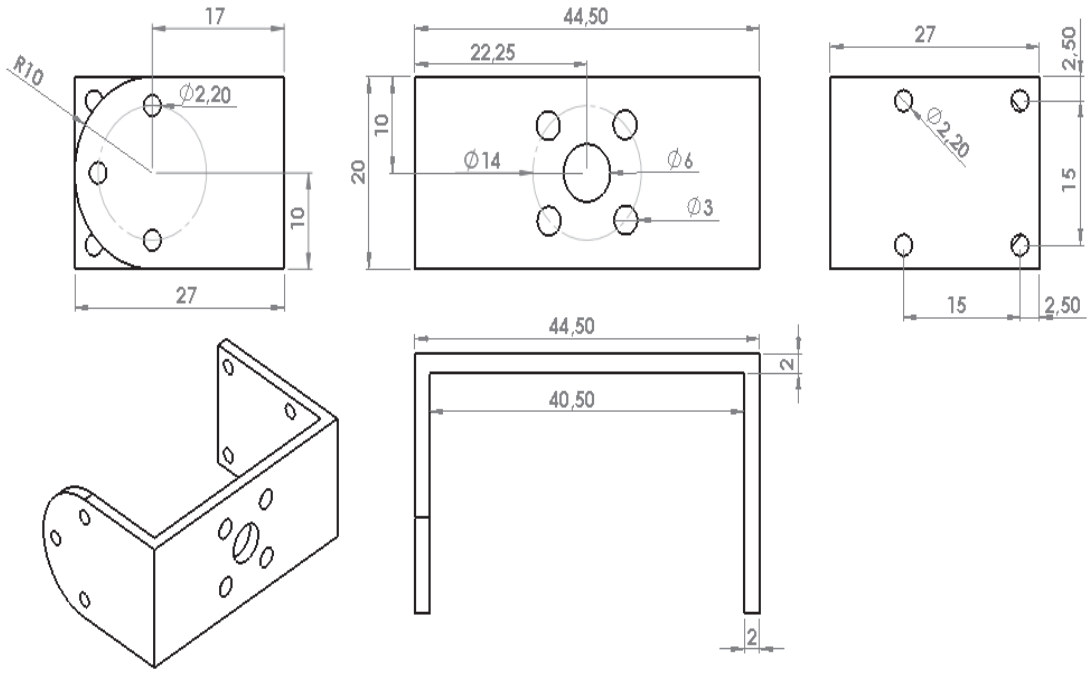


Figure 2.4 Servo mounts and basic dimensions

2.1.3 Servo Motor Disc Horns

Disc horns are shown in Figure 2.5 and they are used for connecting servo motors to brackets or other parts.

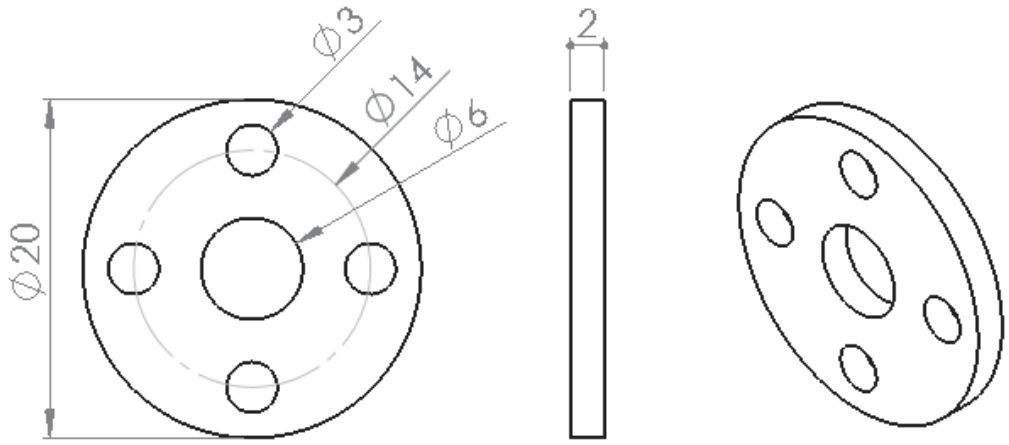


Figure 2.5 Servo discs with basic dimensions

2.2 Mechanical Design

Three-joint design is selected for more smooth design instead of real spider leg, since it can be really challenging to imitate 4-joint leg of a spider especially in software development. Also the 3-joint design is enough to provide satisfactory movement.

2.1.1 3D Printed Parts

Two different leg designs were made with Futuba S3004 servo motor as shown in Figure 2.6 and Figure 2.7.

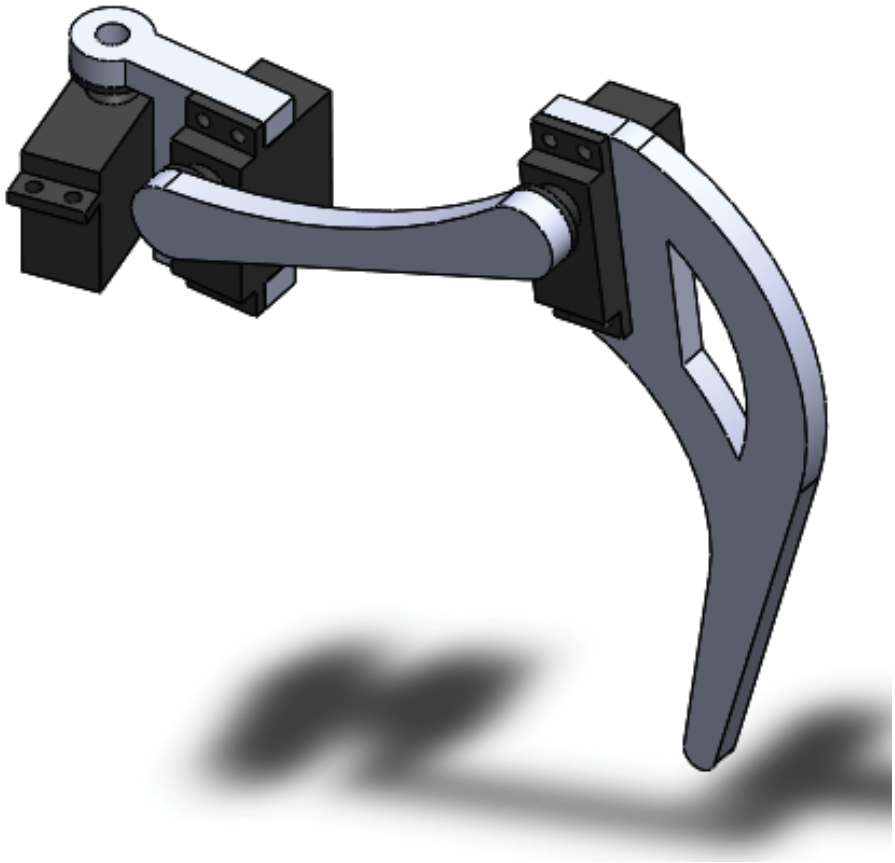


Figure 2.6 Early model of the leg, Type 1

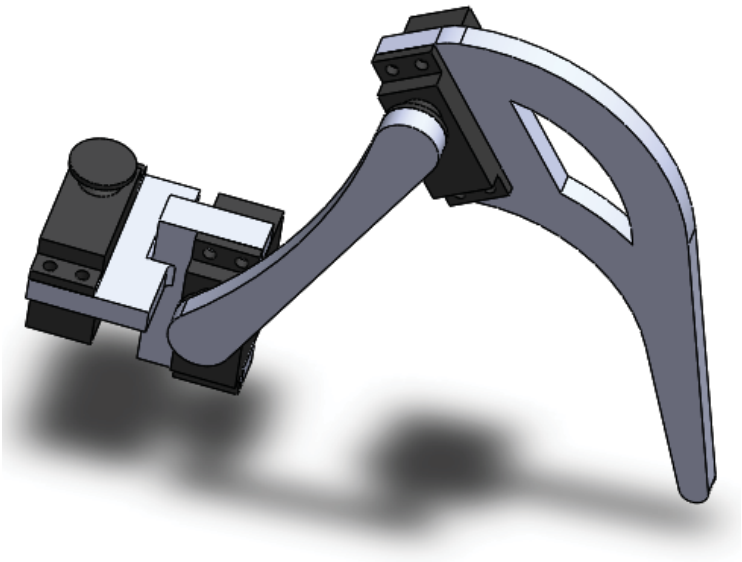


Figure 2.7 Early model of the leg, Type 2

While finishing the leg design other servo motors were checked to see if there are better options. After some investigations, in accordance with the purpose of the robot and considering future works it was concluded that the smart servo with feedback mechanism will be a suitable selection and then Feetech SCS15 servo motor was chosen and new leg designs were made according to new servo motor dimensions.

At first it was uncertain to what kind of design is going to be used so multiple leg design were made by using SCS15 as servo and the design which is more simple to manufacture and most durable is choosen.

Since the new servo has its own aluminum brackets the design changed drastically and became much more reliable (Figure 2.8). But there was one mistake, the material used in 3D printer has lower durability than expected and the parts were a bit long so after some time they began to buckle and started to shatter from their weak points and third change was made to solve this problem.

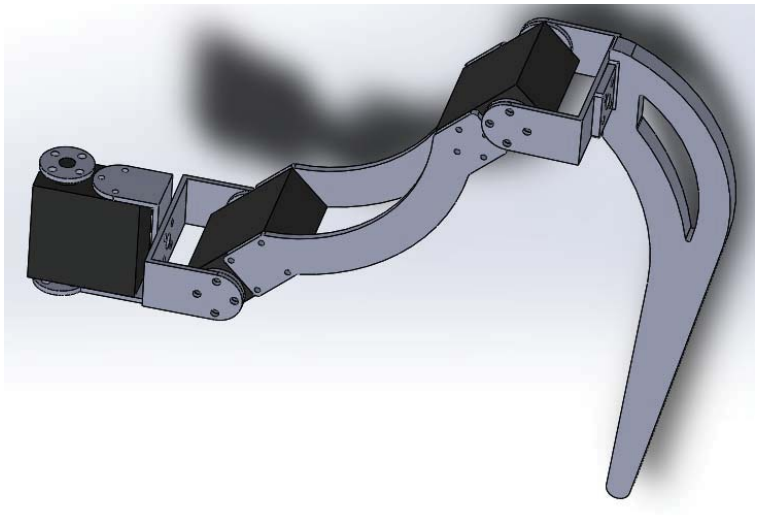


Figure 2.8 Early model of the leg with smart servo motor

The new leg should be much shorter; when the legs become shorter less stress was experienced on the plastic parts. Besides, the servo motors needed less torque to operate which translates into less energy usage and less motor wear. At the end of the evaluation process, the model shown in Figure 2.9 is obtained.

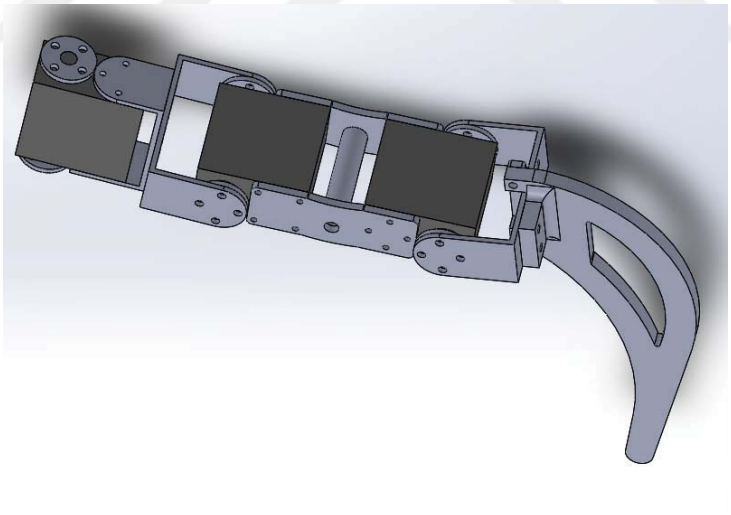


Figure 2.9 Final leg model

The basic dimensions for the previous design are given in Figure 2.10 and Figure 2.11 (in mm);

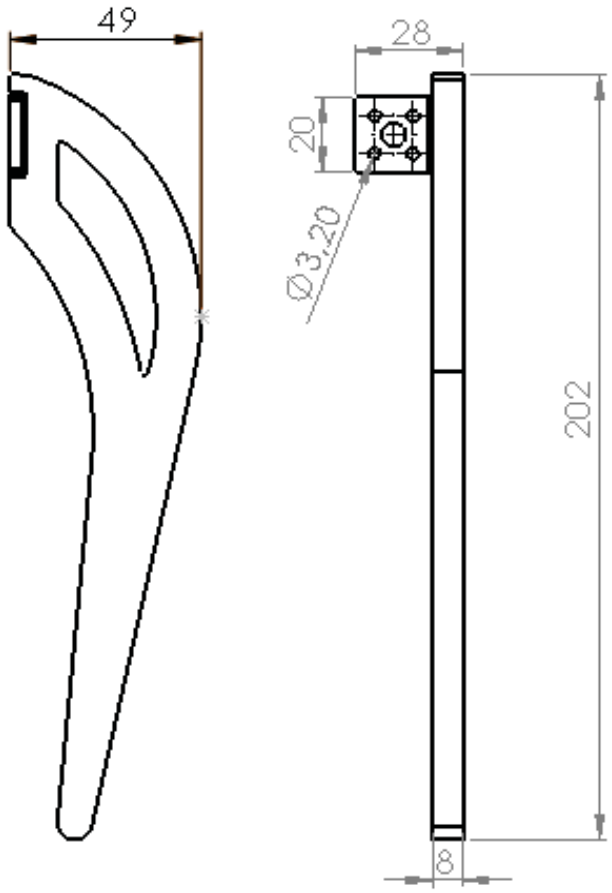


Figure 2.10 Dimensions of previous leg design

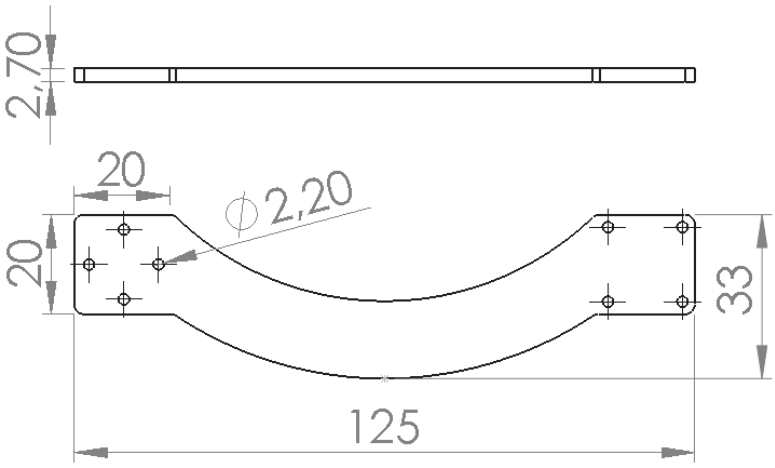


Figure 2.11 Dimensions of previous leg design

After reducing the dimensions and strengthening the weak regions, the new dimensions are obtained as given in Figure 2.12 and Figure 2.13;

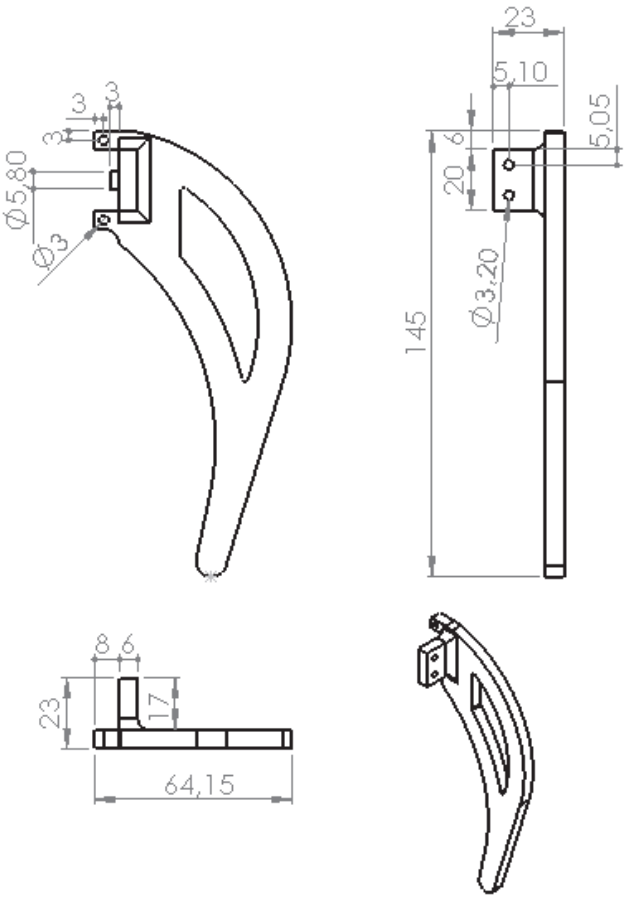


Figure 2.12 Dimensions of new leg design

The holes were designed in accordance with the mounting holes of the servo motor.

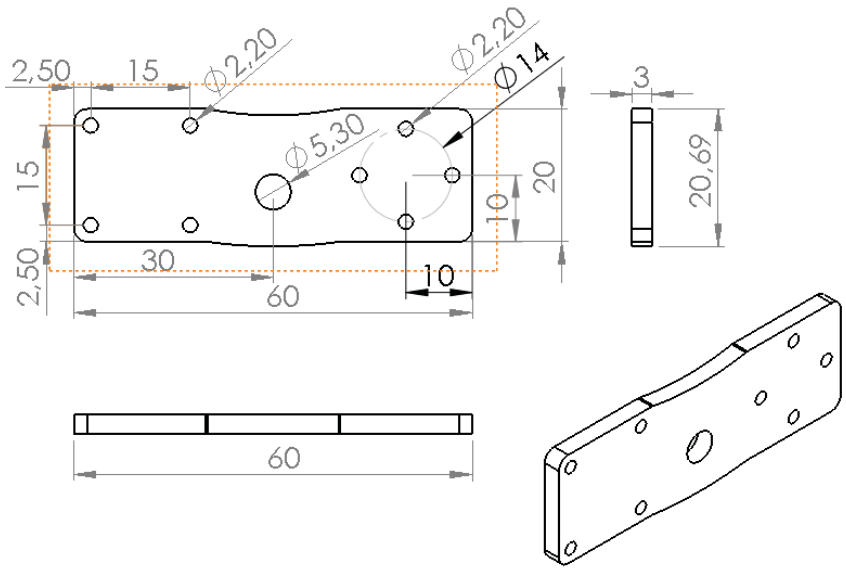


Figure 2.13 Dimensions of new leg design

For the base part, it was thought that it would be better to align all the legs symmetrically considering the software development and aesthetics purposes and the base was designed as shown in Figure 2.14;

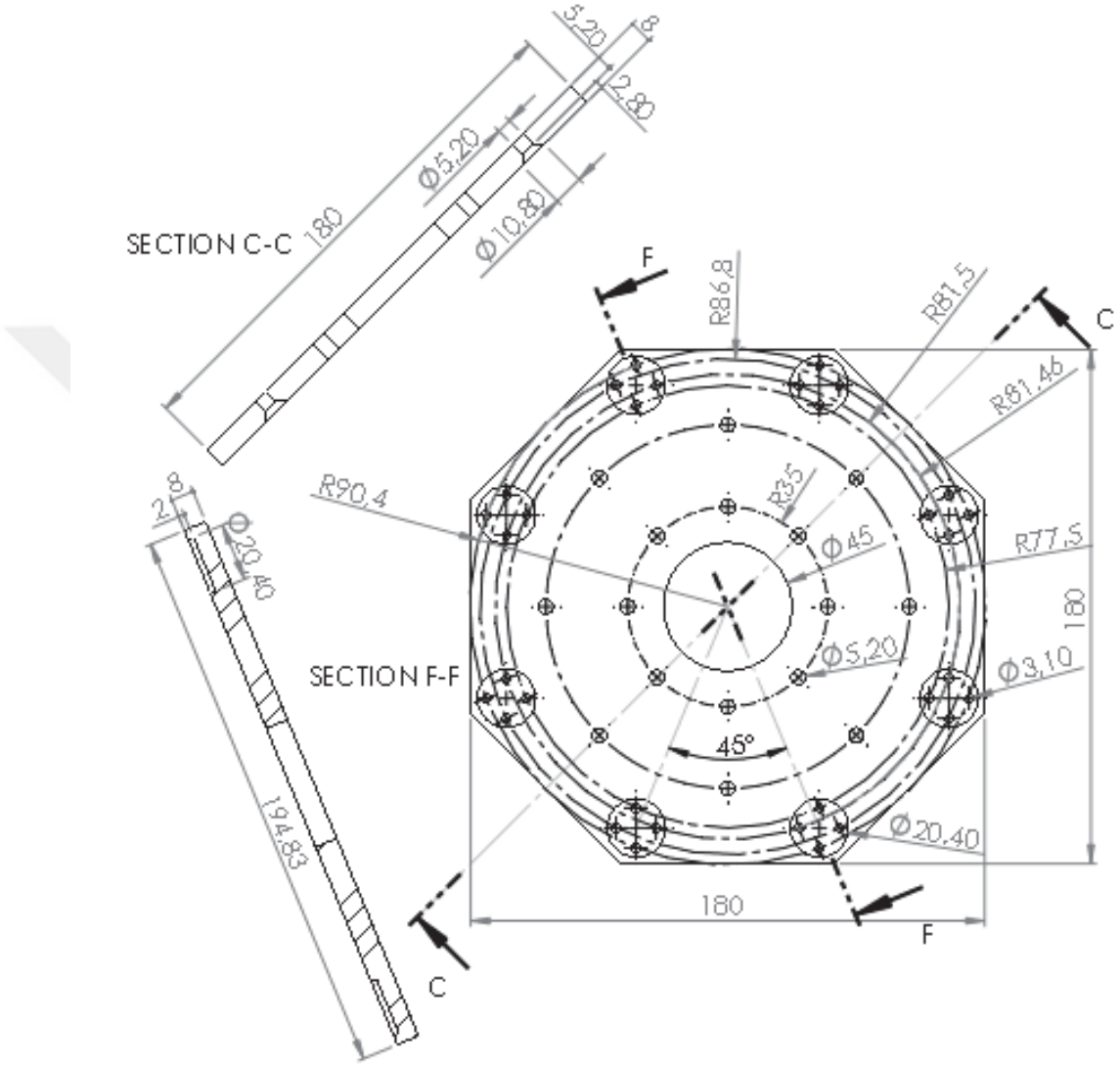


Figure 2.14 Dimensions of upper base

The initial dimensions of the base were 220x220 mm but it was impossible to print with the available 3D printer since the widest allowed dimensions are 200x200 mm. Considering the manufacturing facilities, the dimensions for the base parts are selected as 180x180mm. Ø3.1 holes are created for the screws which connects legs to the lower and upper base. Ø5.2 holes on the outer diameter are used for the support

units that connect lower and upper base parts. The others holes are used for connecting electronic parts to the base.

Support units (Figure 2.15, 2.16 and 2.17) connect the bases and add durability to the robot. There are 6 support units and 6 flat-head-countersunk-socket screws. The screws go through upper base to lower base and there are support units between them. Hexagon nuts makes sure nothing is loose and support units make sure that upper and lower base is not squeezes into each other so that the main servo motors that rotate legs are not jammed.

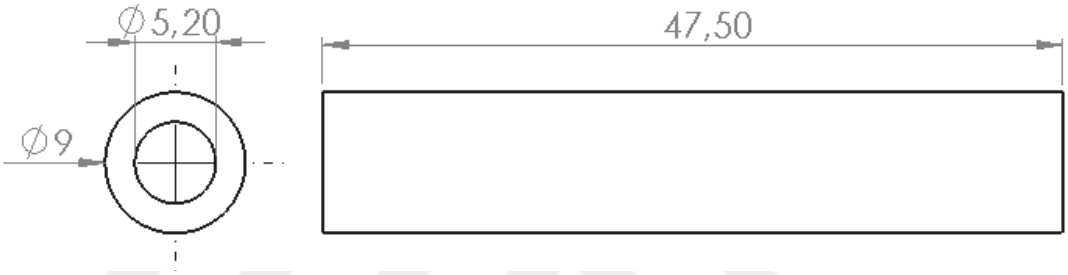


Figure 2.15 Dimensions of support unit

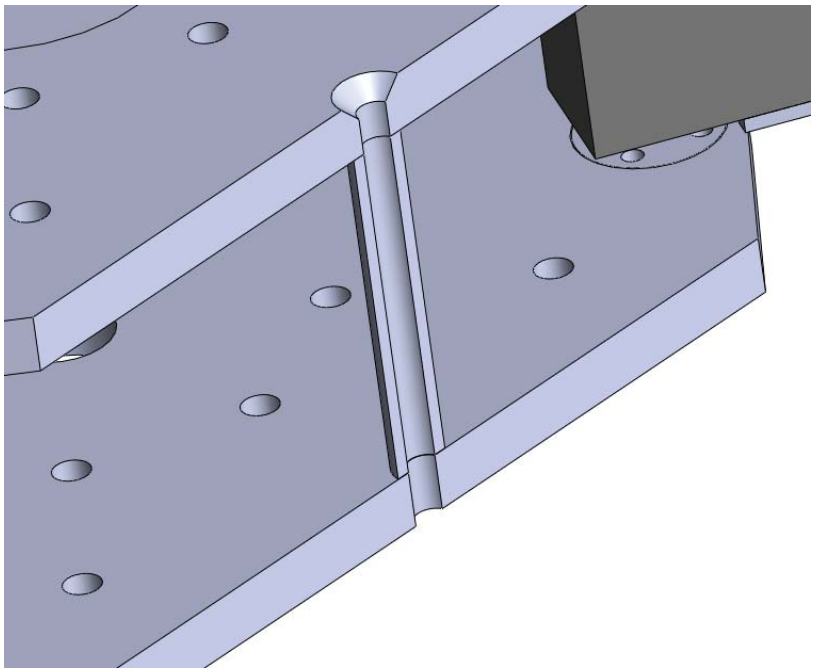


Figure 2.16 Section view of support unit (Isometric)

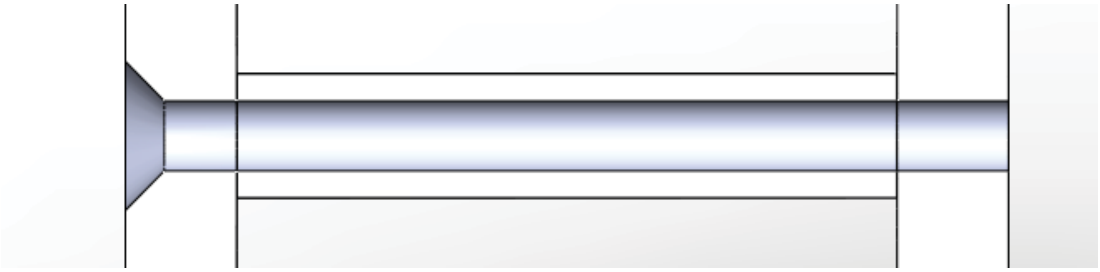
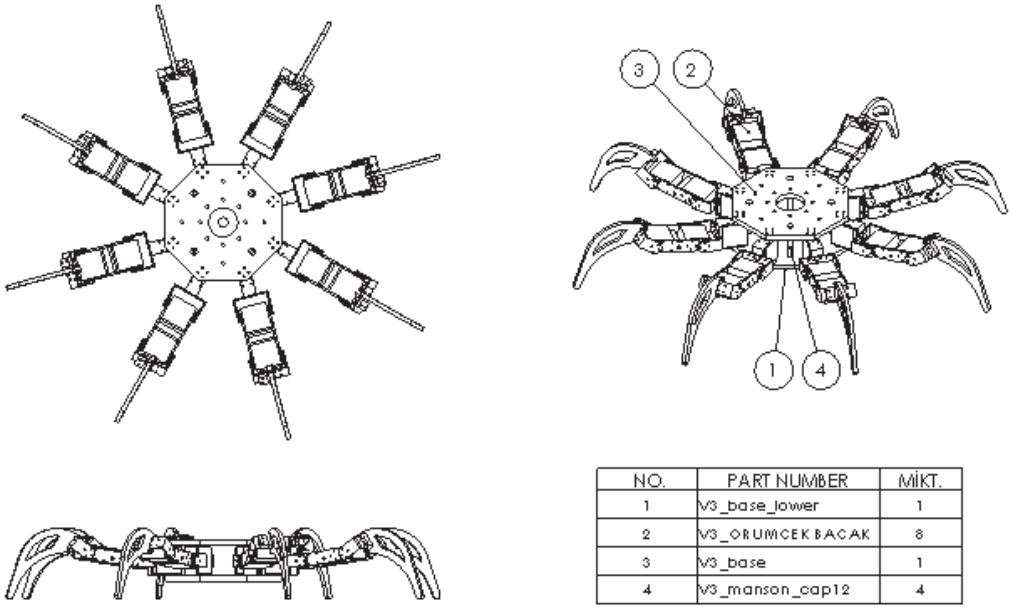


Figure 2.17 Section view of support unit

Final design of ELER as in the technical drawing can be seen in Figure 2.18. General parts and their quantities are also given in the figure.



NO.	PART NUMBER	MİKT.
1	V3_base_lower	1
2	V3_ORUMCEK BACAK	8
3	V3_base	1
4	V3_manson_cap12	4

Figure 2.18 Technical Drawing of the spider robot assembly

The leg design is done according to the Figure 2.19, which shows a real life equivalent of an insect.

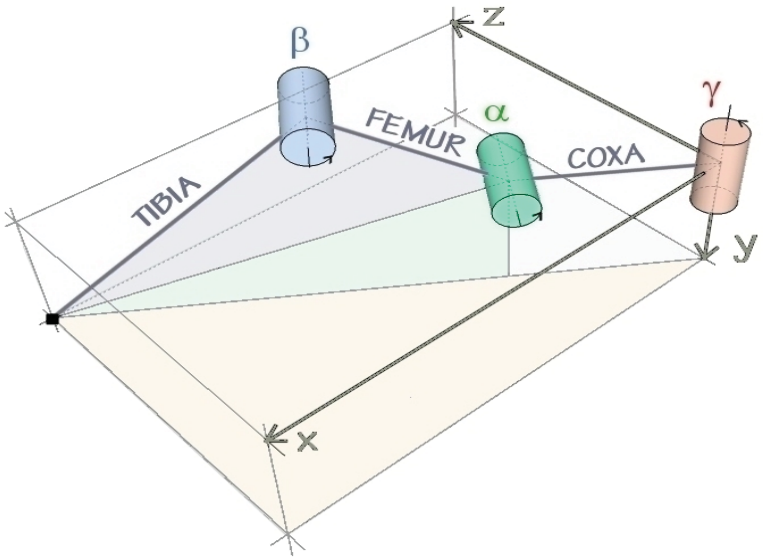


Figure 2.19 Three-joint leg anatomy (Liang, 2018)

In Figure 2.20, the model shown in Figure 2.19 is applied to ELER and the joint movements are described schematically.

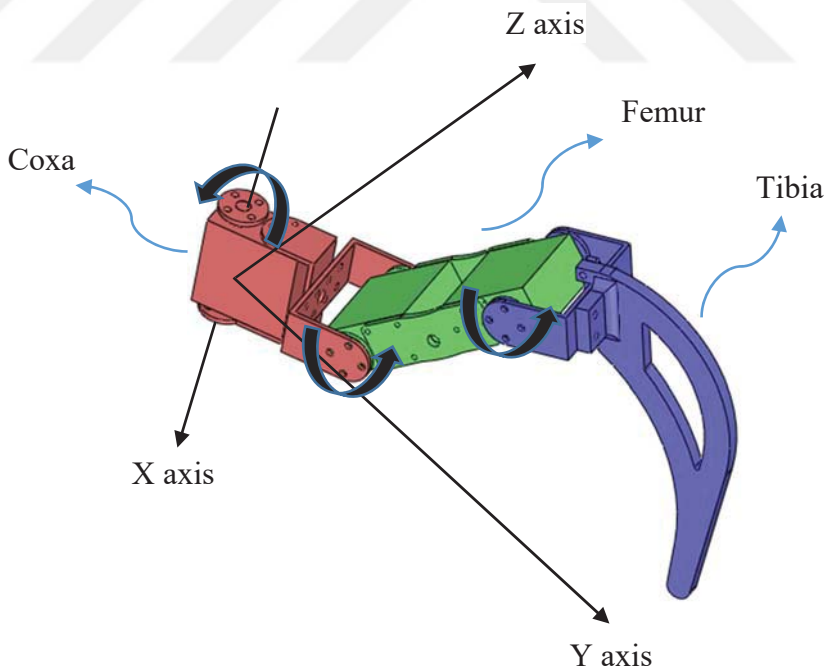


Figure 2.20 Leg anatomy of ELER

CHAPTER THREE
SIMULINK MODEL

3.1 Matlab Simulink Model

Modelling in simulation environment and testing it before designing the model in real-world could have lots of advantages since we could see the errors which we cannot foresee, resulting in the time and cost savings (Defense Acquisition University Press, 2001).

Using Matlab Simulink and SimMechanics, the design of the robot can be done virtually. We will be able to iterate on physical robot design and the parameters in a virtual system. Feasibility of the design can be checked beforehand like weight and speed of the robot, how much torque or power does the joints and other critical parts be subject of, etc. Algorithms can also be verified preliminary for the robot.

Matlab also helps us designing without harming the robot. Countless and repeated tests can be done without stressing robot hardware. We can recreate unsafe scenarios to see the boundaries and limits of the robot. Also bugs or unforeseen issues about the robot and environment can be checked.

In the mathematical modelling phase, Matlab Simulink is used. Using SolidWorks solid model, mathematical model of the spider robot is created. The size and the weight of the mathematical model in Simulink environment are nearly identical to the one in real life design and in SolidWorks model.

The model designed in SolidWorks is rebuilt in Matlab Simulink using Matlab SimMechanics blocks as shown in Figure 3.1.

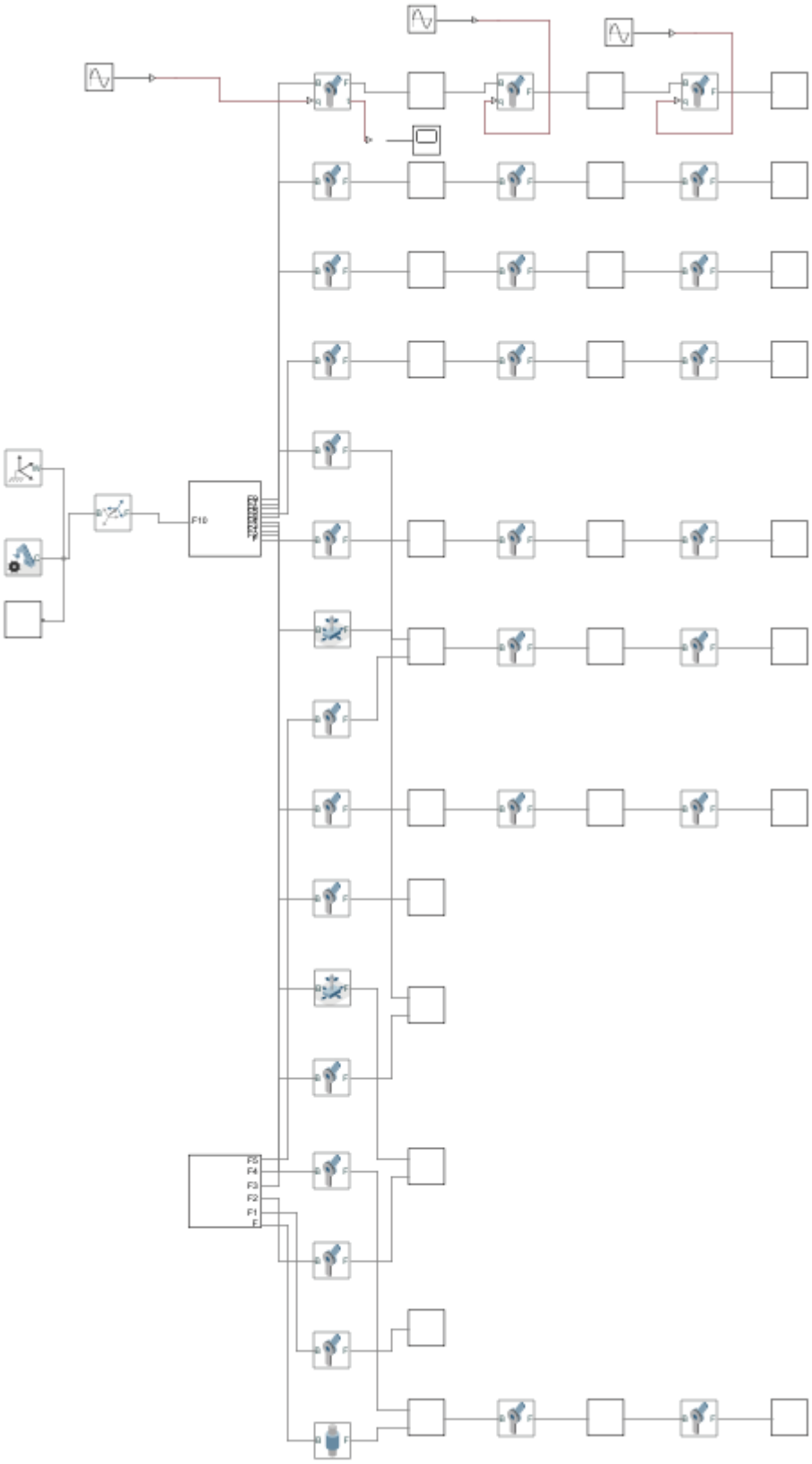


Figure 3.1 Simulink model overview of the blocks

A leg of the spider robot consists of 3 solid parts and 3 joints, servo motors are shown as joints (Figure 3.2) and body parts are blocks consisting of solid blocks and reference frames as can be seen from Figure 3.3.

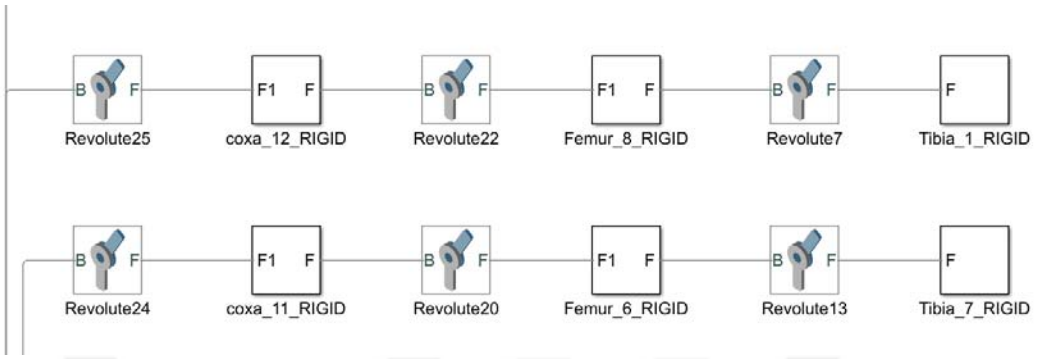


Figure 3.2 Example of leg blocks

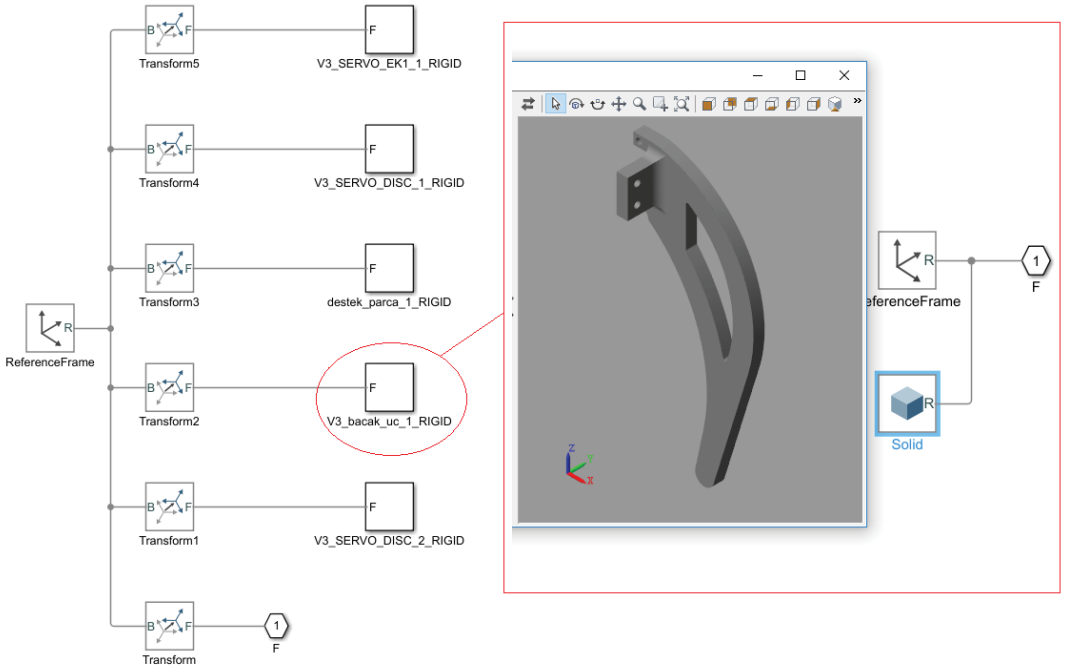


Figure 3.3 Closer look at solid blocks

After the setup of the initial robot diagram, some changes are needed to be made so the robot can move freely on the surface. Without the changes, the robot is just hanging from its main body part and its legs are dangling without any motion.

Firstly, a road like plane shown in Figure 3.4 is needed to be created so that we can simulate the various movements of the spider robot.

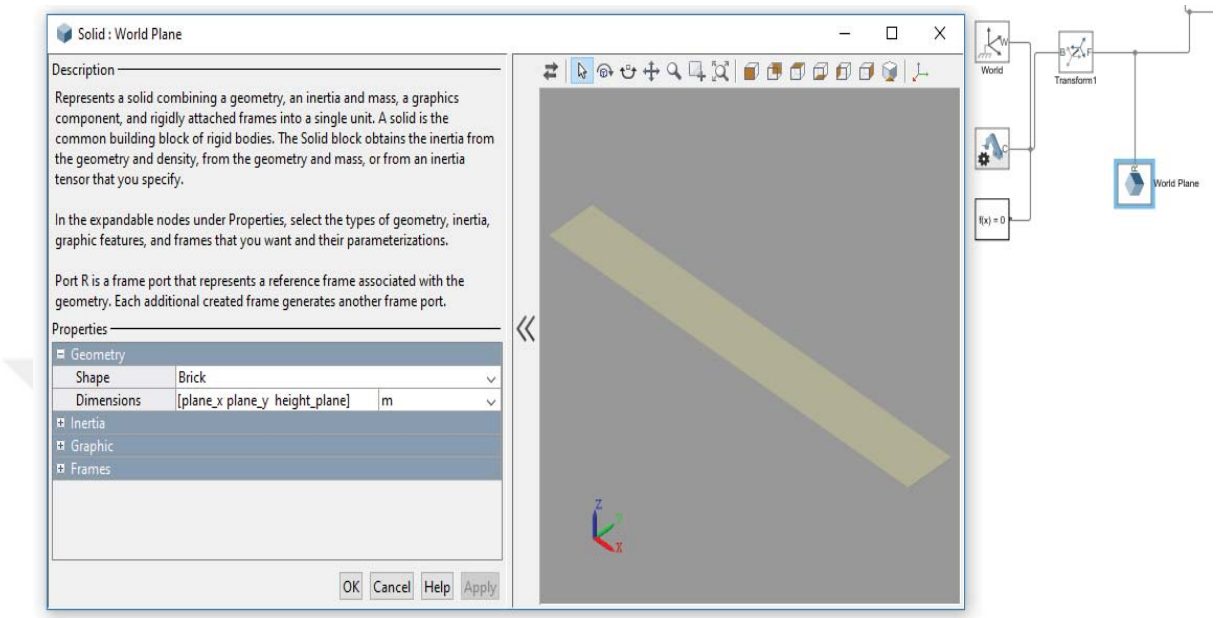


Figure 3.4 Road plane

After setting the road plane, tips of the legs (tibia) need to be in contact with the ground plane and there should be a correlation between the contact forces. To make robot stand up when the motion is given to the joints and setting a proper relation with tibia and plane, sphere to plane contact force is used as shown in Figure 3.5. This force can be found in Simscape Multibody “Contact Forces Library”.

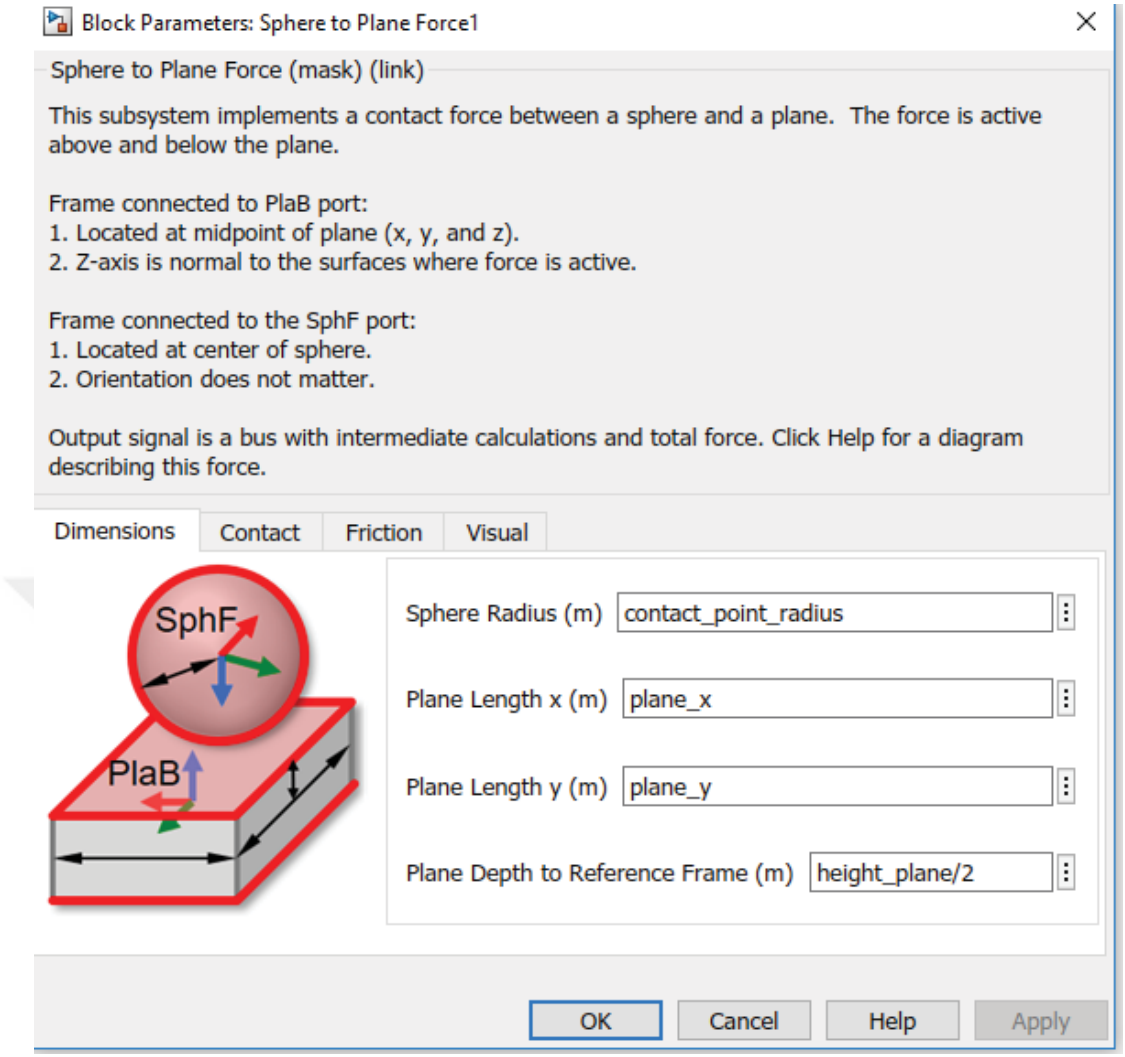


Figure 3.5 Sphere to plane contact force

Four different frame ports are introduced on the tip of the tibia as shown in Figure 3.6 and they are connected to the ground plane via sphere to plane contact force block as shown in Figure 3.7.

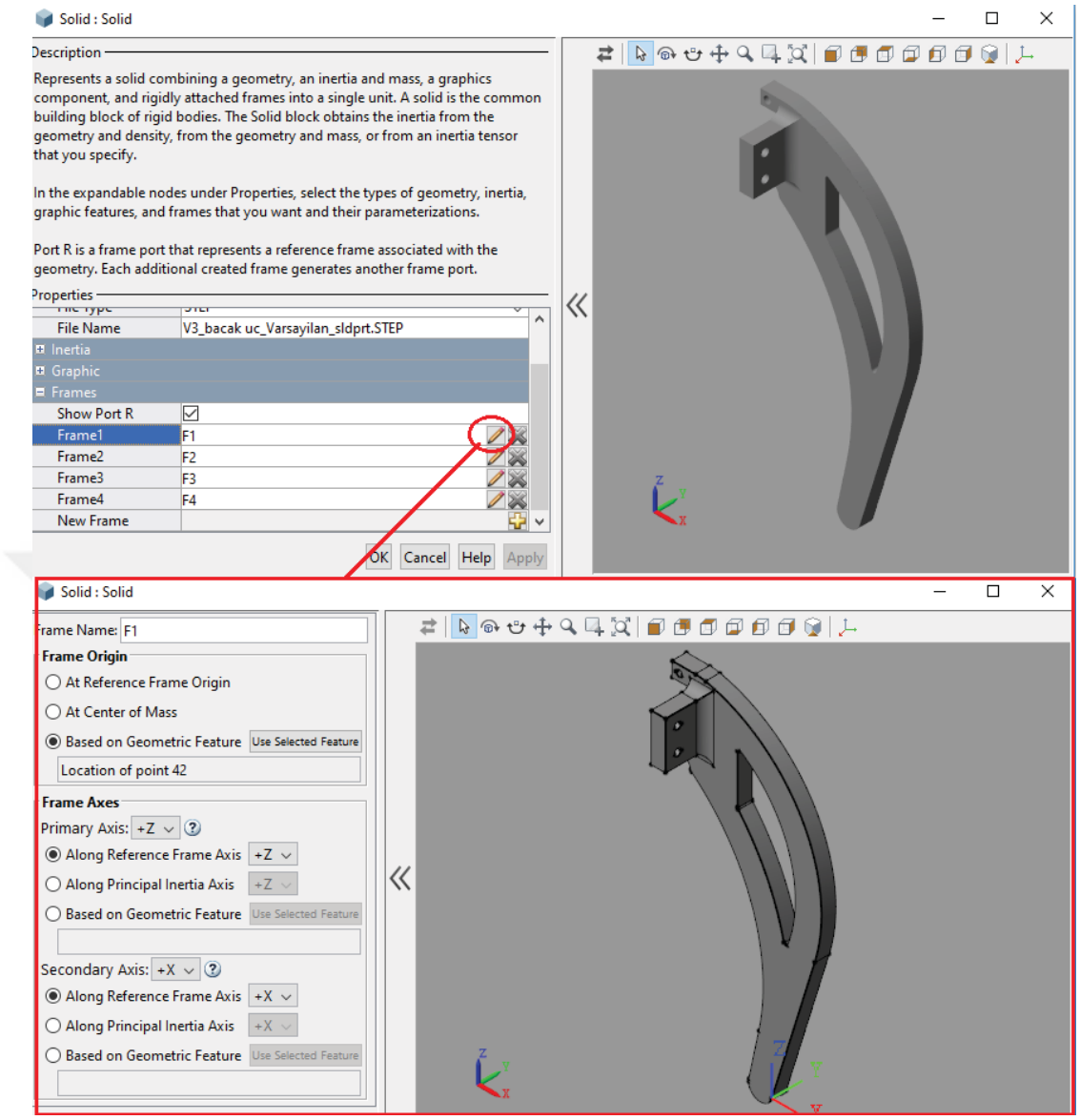


Figure 3.6 Frame ports

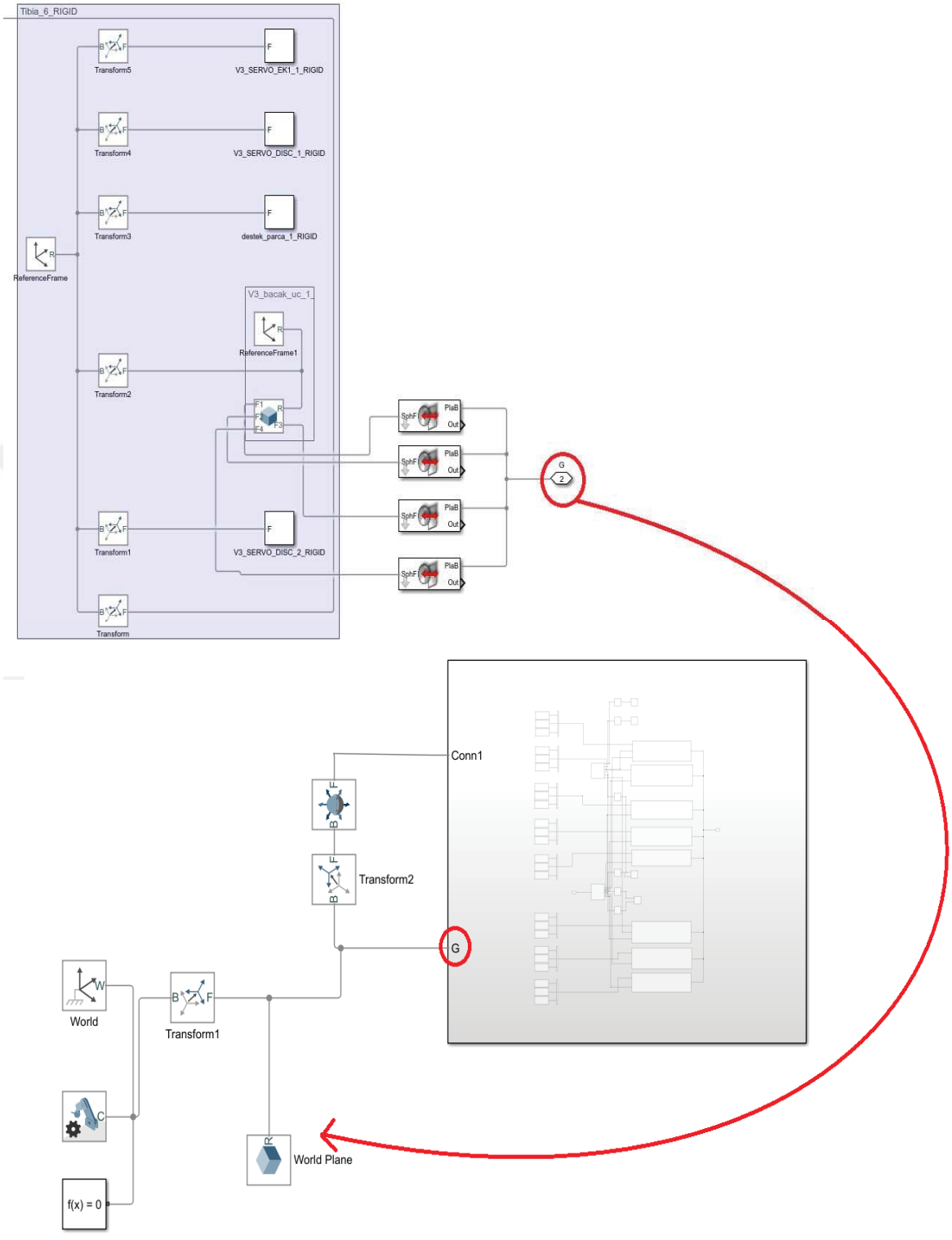


Figure 3.7 Application of the sphere to plane contact force

After setting up the ground-leg contact forces, leg joint actuation need to be defined (Figure 3.8) to give the desired motion to the legs.

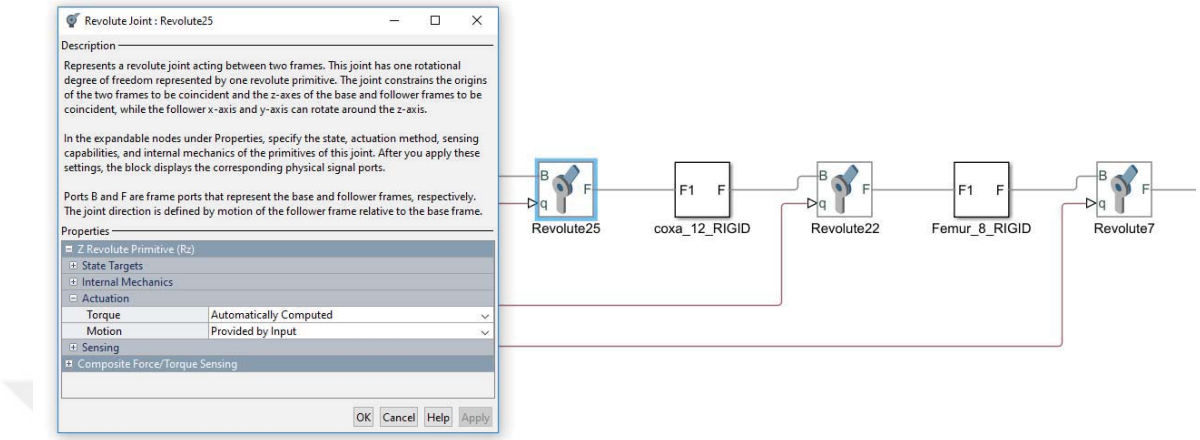


Figure 3.8 Application of the sphere to plane contact force

For the joint actuation, motion provided by input is selected using “mux” and “demux” blocks. Motion is defined by using signal building block and signal is created by hand to imitate the spider’s motion. The signal is directed to joint actuation port by using Simulink-PS converter block as seen in Figure 3.9 and Figure 3.10.

One signal block is used for each joint actuation. Motion signal is created according to the desired motion, which is the elevation motion of the spider robot. The signals for the required motion for the one leg can be seen in Figure 3.11. These signals determine the speed and direction of rotation of the joints. The same signal can create different motions on different joints because of joints position in world so each signal is unique to its own joint for the specified motion.

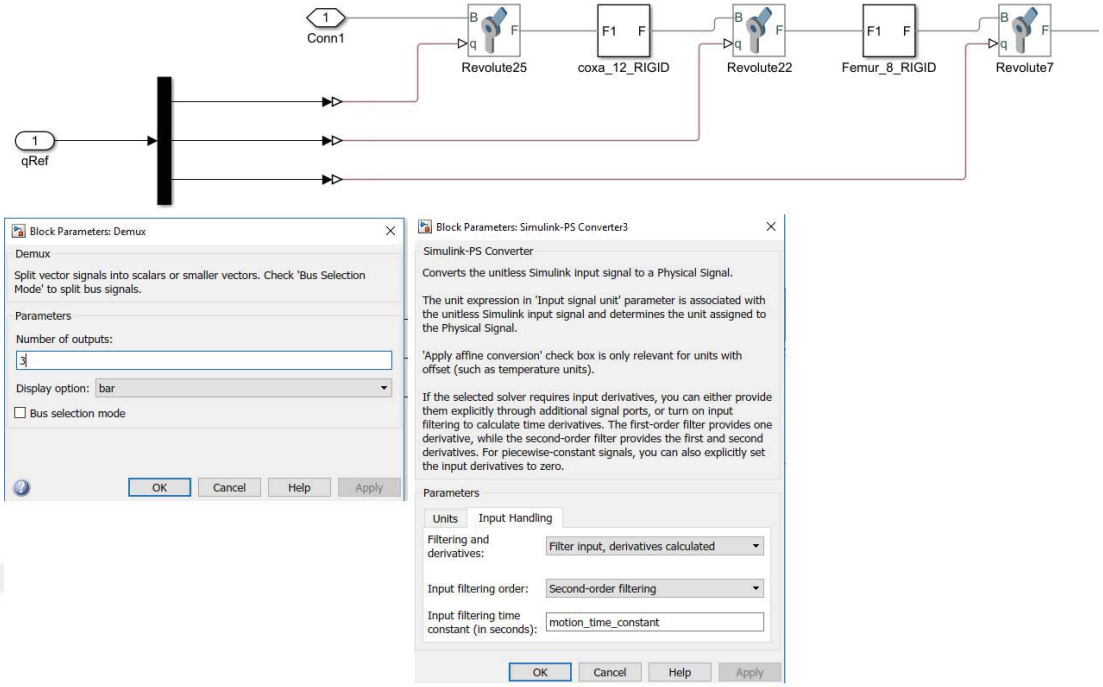


Figure 3.9 Blocks of the Demux and Simulink-PS converter.

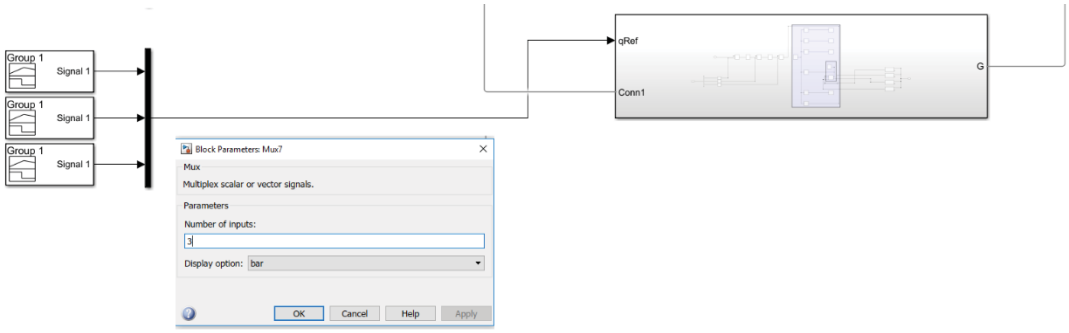


Figure 3.10 Mux block

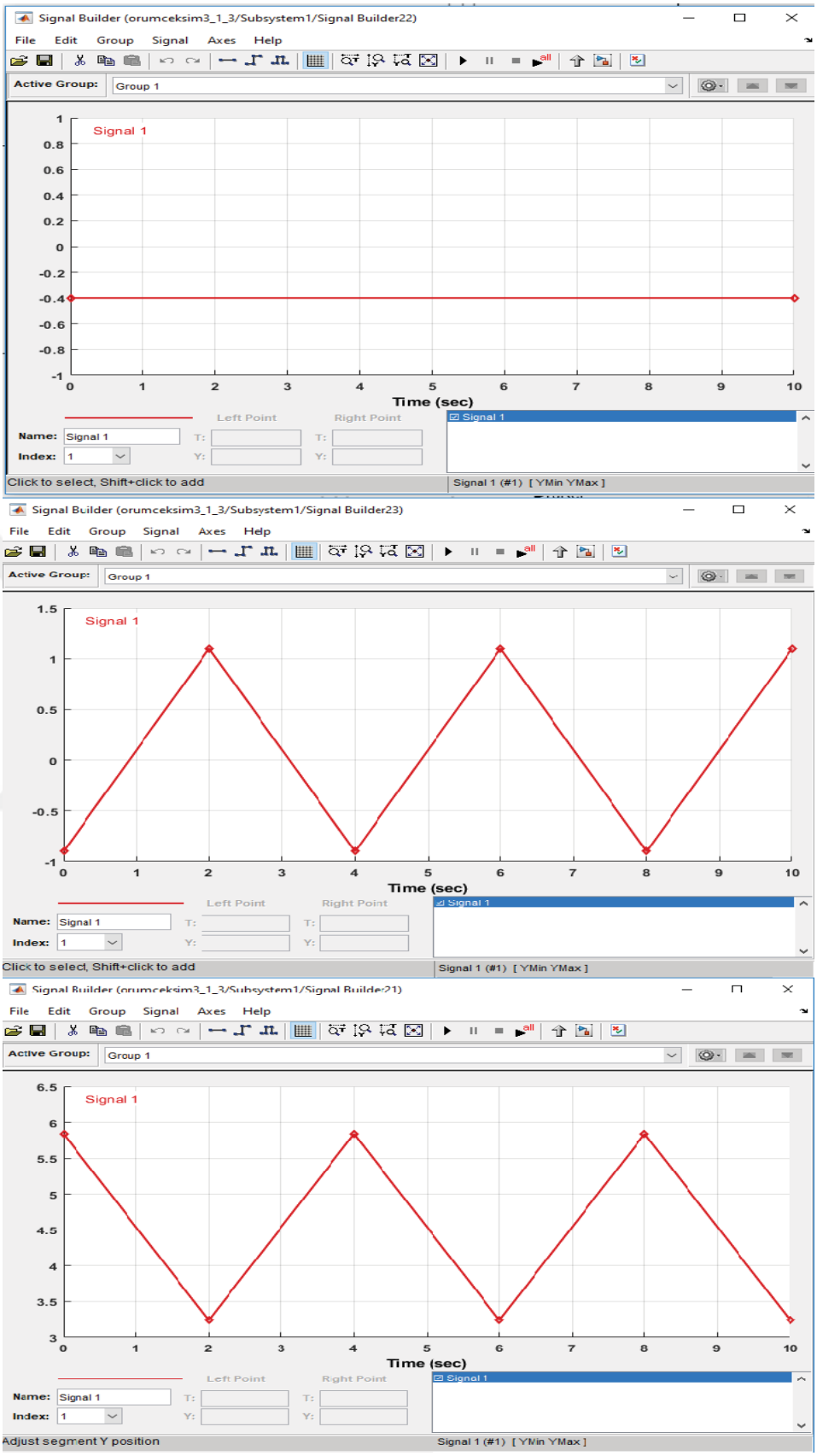


Figure 3.11 Signals for a single leg

Having established every joint with their proper motion signals shown in Figure 3.12, the simulation gets started.

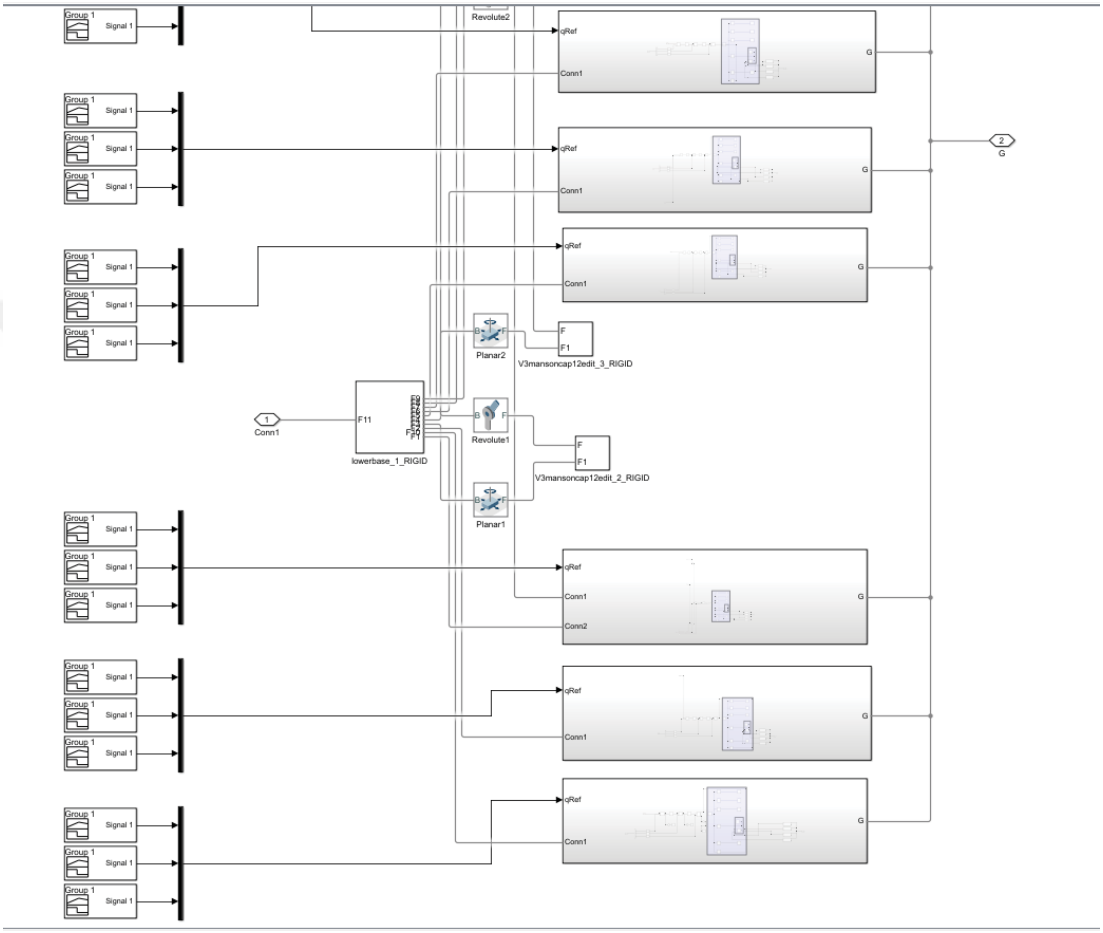


Figure 3.12 Input signals

After running the simulation, robot's up and down movement can be seen in the Matlab output window (Figure 3.13), which is identical to the real world model's up and down movement.

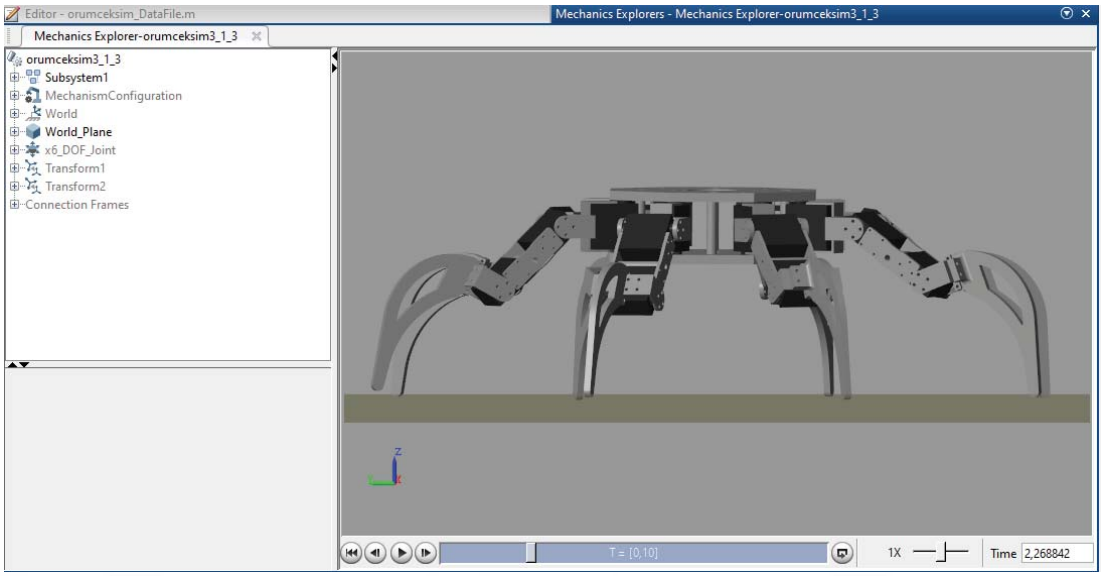


Figure 3.13 Movement in simulation environment

Using the sensing ports of the joint we can read position, velocity, acceleration and actuator torques of the joints. Sensing the torques are important for deciding if the servo motors have enough power or not. Position, velocity and acceleration values are important for mathematical model of the robot.

The data from the joints can be read by using PS-Simulink converter and scope blocks which as seen from Figure 3.14 and Figure 3.15.

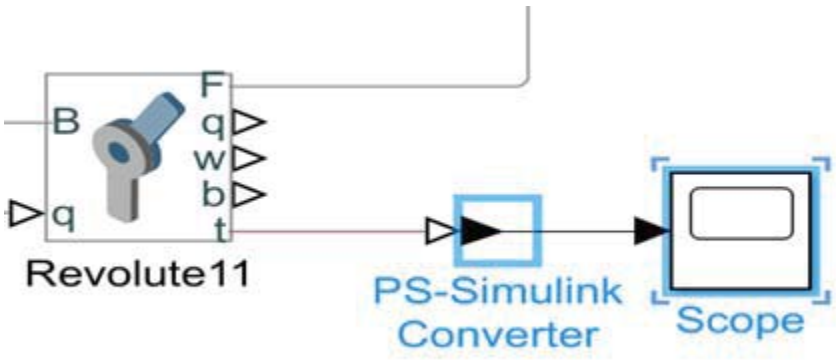


Figure 3.14 Reading data

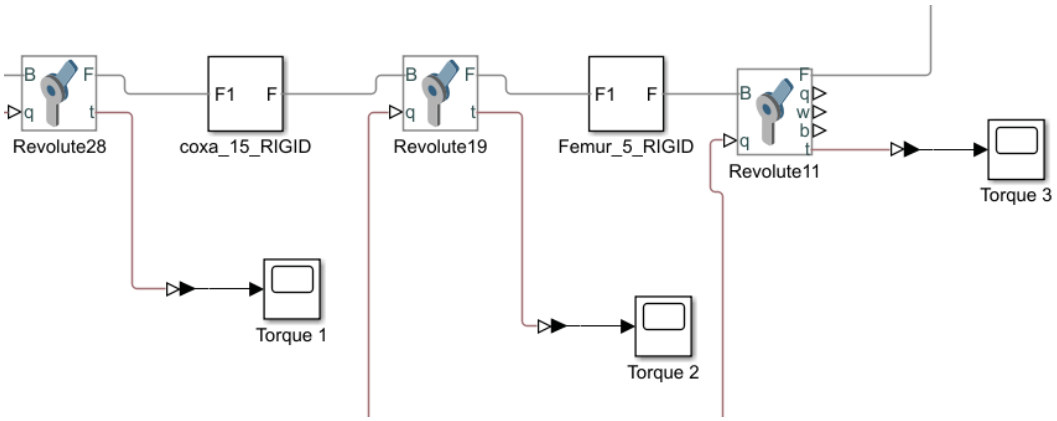


Figure 3.15 Reading torque data from joints

The results (Figure 3.16-3.18) can be seen as unit-time curve and then discussed accordingly. Torque unit is Newton-meter and time unit is second.

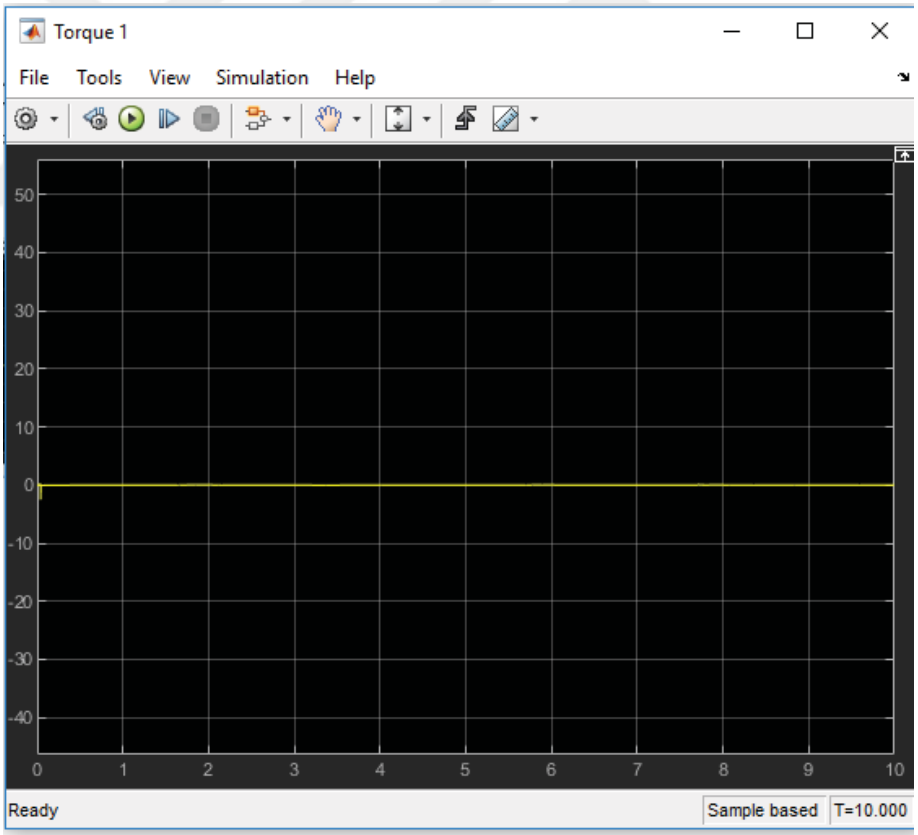


Figure 3.16 Torque data 1

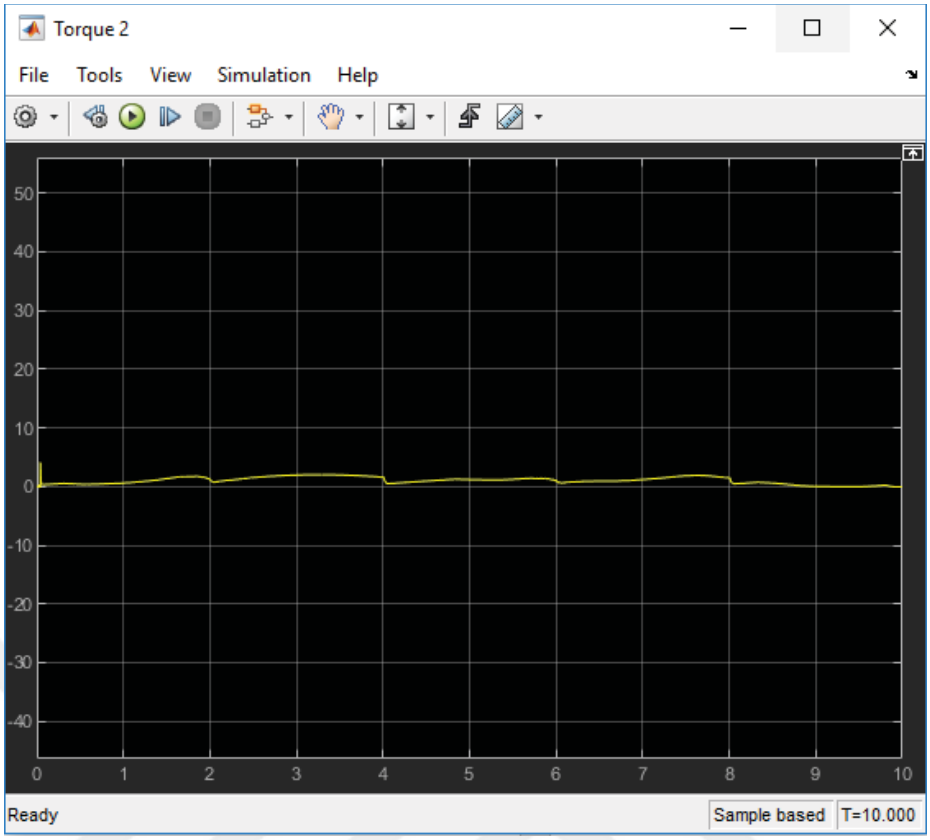


Figure 3.17 Torque data 2

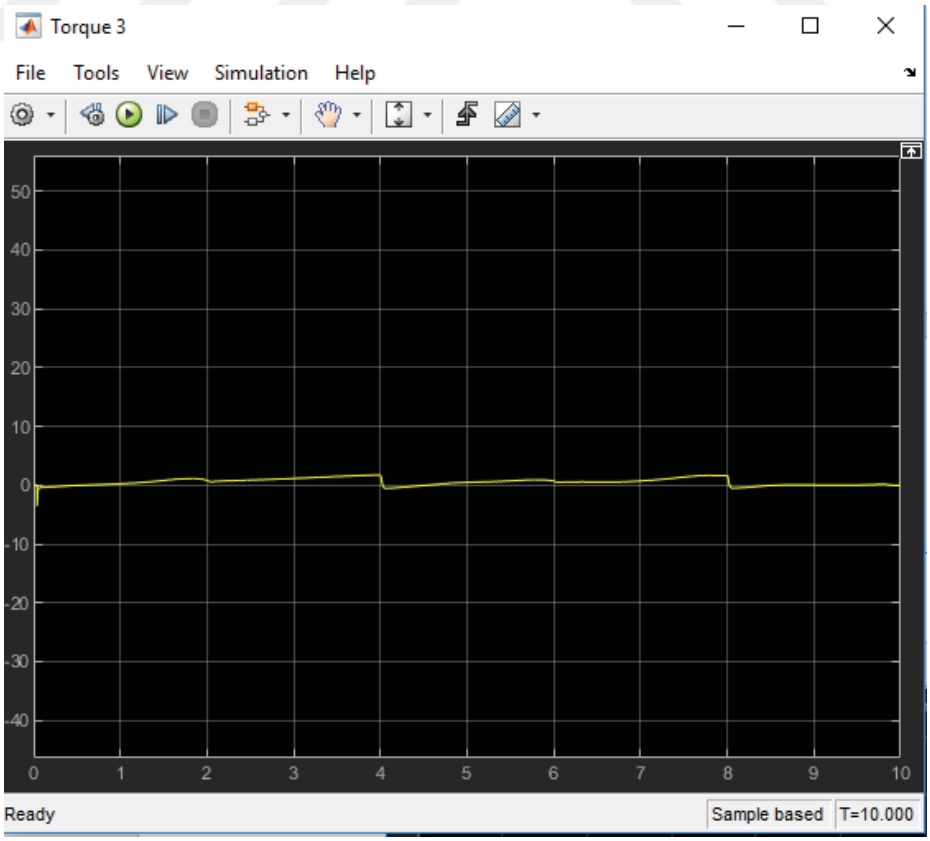


Figure 3.18 Torque data 3

Torque 1 belongs to the base joint, since it doesn't move in up and down movement its value is very close to zero. Torque 2 and Torque 3 reflects the middle and end joint torques, respectively. The robot starts its movement just a little bit above the ground so the torque values have some spike at the start to demonstrate the falling action, than the values get normal.

Using these torque values, we can merge them together to see their variation with time as shown in Figure 3.19.

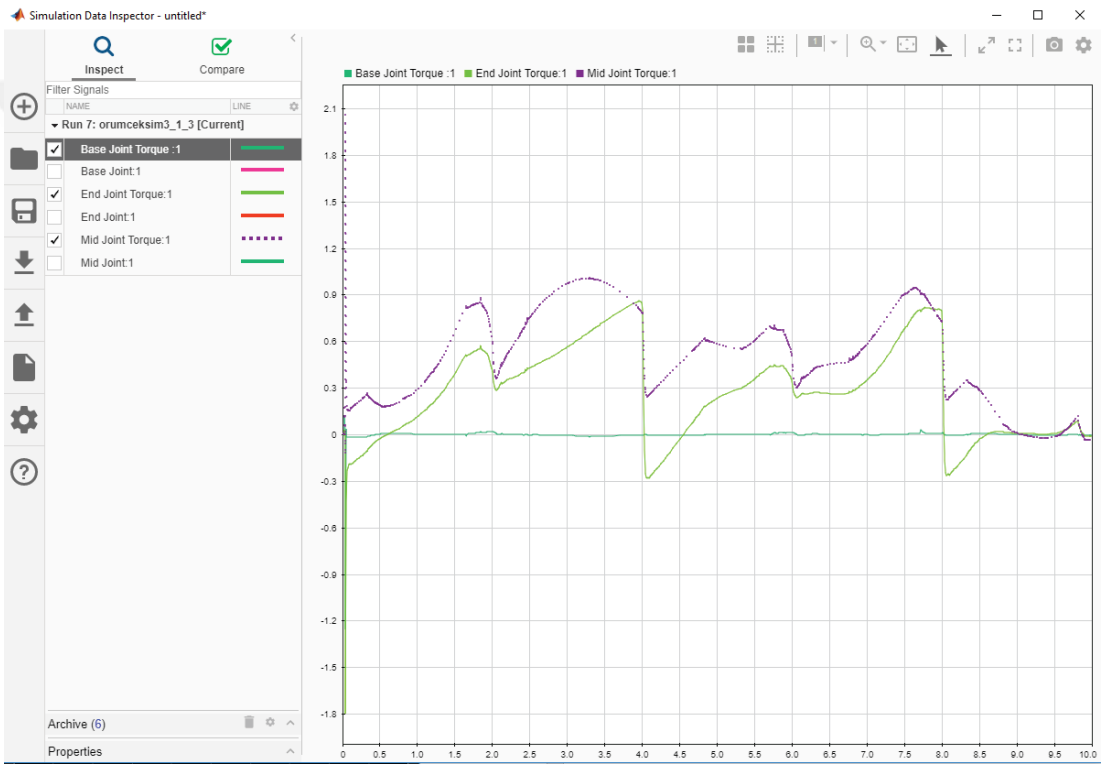


Figure 3.19 Merged torque data

CHAPTER FOUR
ELECTRONICS DESIGN

In this chapter, ELER's electronics design was explained in details. After giving some properties of the electronic parts, the electronics design was explained. The design was performed around the Arduino Mega 2560 microcontroller platform and the cabling network of the smart servo motors.

4.3 Electronic Parts

4.3.1 Arduino Mega 2560 Microcontroller Platform

Arduino Mega 2560 platform shown in Figure 4.1 was used to control the whole system.

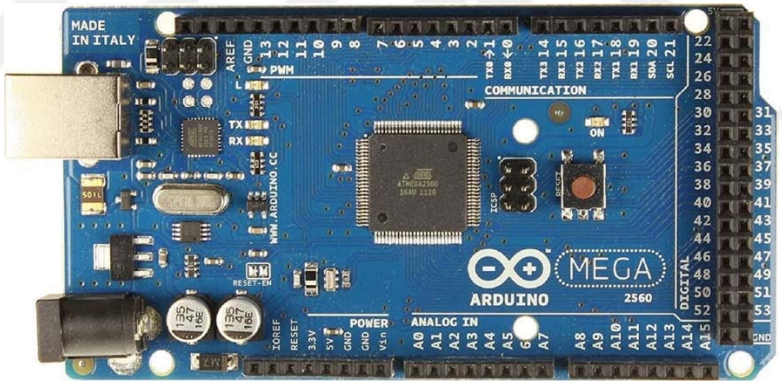


Figure 4.1 Arduino Mega 2560 (Arduino MEGA 2560 Board, n.d.)

Arduino was chosen because it is an open-source electronics platform, software and hardware is extremely accessible and very flexible to be customized. It is very flexible and offers a variety of digital and analog inputs. It is easy to use, connects to computer via USB and communicates using standard serial protocol, runs in standalone mode and has and interface connected to PC/Macintosh computers. A growing online community backs up Arduino and lots of source code is already available.

4.3.2 SCS15 Smart Servo Motor Electronic System

SCS15 is a smart servo motor and it has electronic control unit with sensors. The electronic control is different from the standard servo motors. TTL level multi drop bus interface is chosen to control multiple servo motors via the half duplex TTL serial communication. Figure 4.2 shows the TTLinker internal circuit. The baud rate from 38400 bps to 1 Mbps can be used for the communication, but the factory default communication speed is 1 Mbps. Usage of multiple servo motors in the same communication link is possible by giving them a fixed identity number (ID) individually. For this purpose, the range from 0 to 253 (0xFD) can be used (The factory default setting is ID 1), and, especially 254(0xFE) is used as the broadcast ID. If the broadcast ID is used to transmit instruction packet, we can control the servo motors at the same time. To give the IDs for servo motors, a PC software mentioned later was developed. IDs can be assigned to the servos one by one using the developed software.

The servo electronic system can be powered by 6V~8.4V power source. The upper 8.4V limit is very import for the ELER design. Thus, there is no need to use a voltage regulator to reduce the Li-Ion battery voltage to ease the electronic design.

The servo electronic has position, temperature, load, speed and input voltage sensors. These sensors are very important to make smart actions and differ from the standard servo motors, which have only position sensor.

Since each servo has a unique ID number and can be connected serially, cabling and servo control becomes much easier and smooth. These servo motors can feedback lots of variables such as the value of angular position, temperature, load, torque, speed and input voltage. Load feedback allows us to determine whether the legs are touching a surface or not while input voltage lets us to check which walking pattern is more cost efficient while travelling the same path. In the future works these feedback mechanisms can be used to implement self-learning and self-correction actions. For example, the robot can alter its motor position values every time it walks

a certain path and see for itself to use which moving pattern or which motor positions are best for energy or time saving. It can be easily controlled via Arduino with the help of TTLinker_mini between Arduino and servo connection.

4.3.3 TT Linker

TT linker is the converter between the microcontroller and the servo electronics. The servo has three inputs. One of them is the serial control input called as signal input. The others are the voltage and the ground inputs. All servo motors' signal inputs are connected to the same signal cable. Using the same cable, all servo motors can receive commands and send their status and sensor information. Figure 4.2 explains the electronic schematics and the connection of the TT linker. Figure 4.3 shows the TT linker photo.

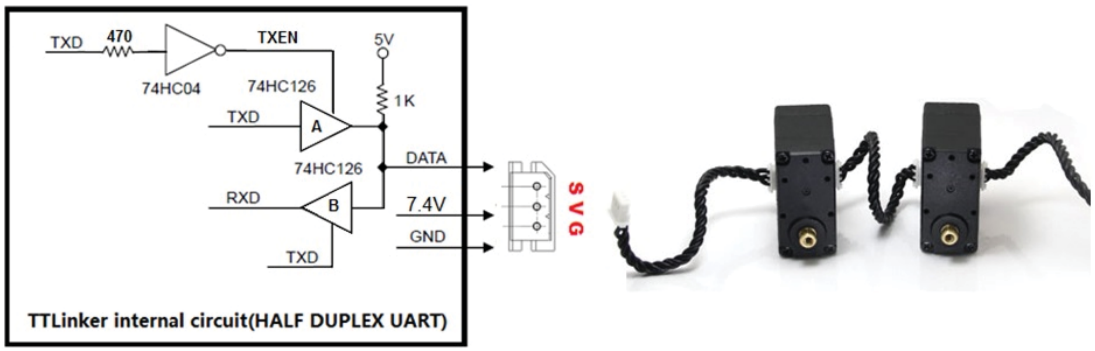


Figure 4.2 TT Linker schematics and the connection (TTLinker schematics, n.d.)

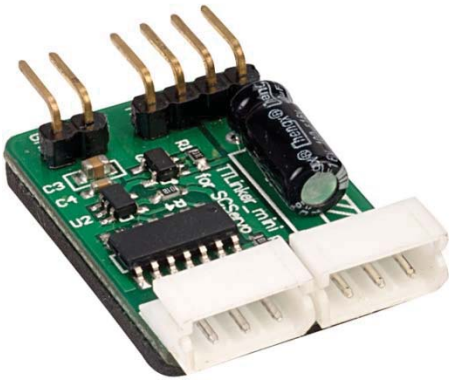


Figure 4.3 TT Linker (TTlinker, n.d.)

4.3.4 Bluetooth Module

Bluetooth is a technology that exists in almost all devices capable of wireless communication, from our mobile phones to our headsets. For Arduino communication HC-06 bluetooth module (Figure 4.4) is one of the easier to use and simple bluetooth module that we can use.

Figure 4.4 HC-06 Bluetooth modüle (HC-06, n.d.)

4.3.5 Batteries

Almost 40% of the world's batteries are lithium-ion. Nowadays, the 18650 battery (Figure 4.5) is the most used Lithium Ion battery. Especially it is used in laptop computers, electric powered tools, robots, powerbanks, flashlights, even in electric cars. For example Tesla cars use 7180 of it to form a battery pack. The battery name shows the dimensions. For 18650, the size of the battery is 18x65 millimeters. The last character shows that it is a cylinder type battery. The battery voltage is nominal voltage 3.7 V, the charging cut-off voltage is 4.2V, and the common capacity is 2200 mAh-3500 mAh. Protected and unprotected types are available, but especially for safety precautions protected ones should be used. Average charge time of the battery is about 4 hours but this can vary with the charger specifications.

Figure 4.5 The 18650 battery (NCR18650, n.d.)

4.3.6 Battery Holder

The battery holder with two serially connected compartments is shown in Figure 4.6. Two set of battery holders were used on two sides of the robot's main platform.

Figure 4.6 Battery holder (Battery holder, n.d.)

4.3.7 Battery Charger

Four set of 18650 batteries are charged with an external Li-Ion battery charger (Figure 4.7)

Figure 4.7 XTAR VC4 Li-Ion batttery charger (Li-ion charger, n.d.)

4.3.8 LCD Module

2x16 LCD module was selected for debugging and showing the current status of the ELER (Figure 4.8).

Figure 4.8 2x16 LCD module (2x16 LCD modüle, n.d.)

4.3.9 Remote Control Mobile Device

Remote control of the robot is done by an Android mobile device. The software was developed on MIT App Inventor program. Interface can be seen in Figure 4.9.

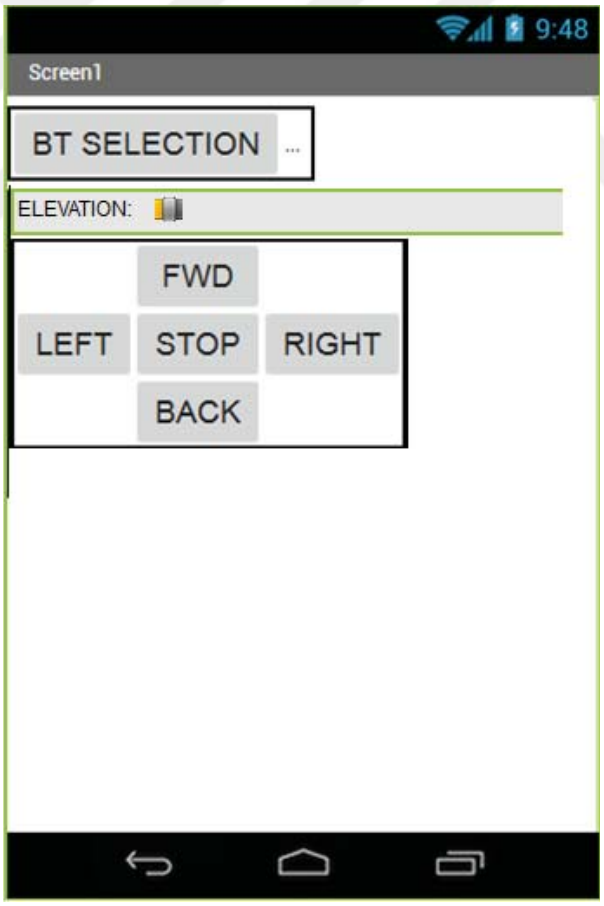


Figure 4.9 App inventor remote control interface screen

4.3.10 IMU Sensor

BNO055 is an intelligent 9-axis absolute orientation sensor (Figure 4.10). Using this sensor we can detect the position of our robot in 3-dimensional space without doing extra calculations. Thanks to the ARM Cortex-M0 microprocessor on it, it is possible to obtain the absolute position of the objects as quaternion or Euler vector in addition to the normal IMU card form. It is very suitable for use on cards like Arduino. It uses the I2C interface and it can be used with logic levels of 3.3 V and 5 V.

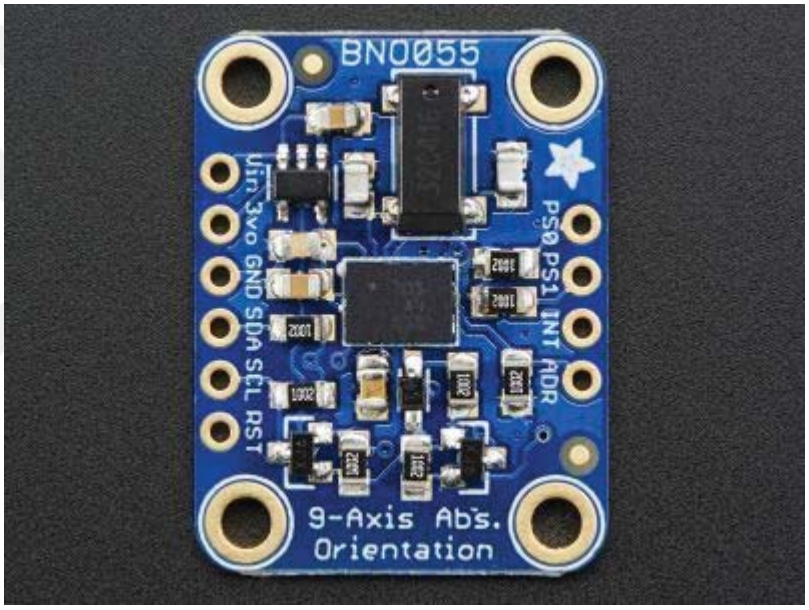


Figure 4.10 BNO055 IMU sensor (BNO055, n.d.)

4.4 Electronic Design

Electronic parts were attached to the upper base part of the robot through various holes designed in the base parts. The main important part is Arduino Mega 2560 microcontroller platform. The Arduino Mega is powered by two parallel 7.4V battery modules. The main reason of separating two battery modules into two parts is to balance the weight on the main body. The block scheme of the electronic design is shown in Figure 4.11.

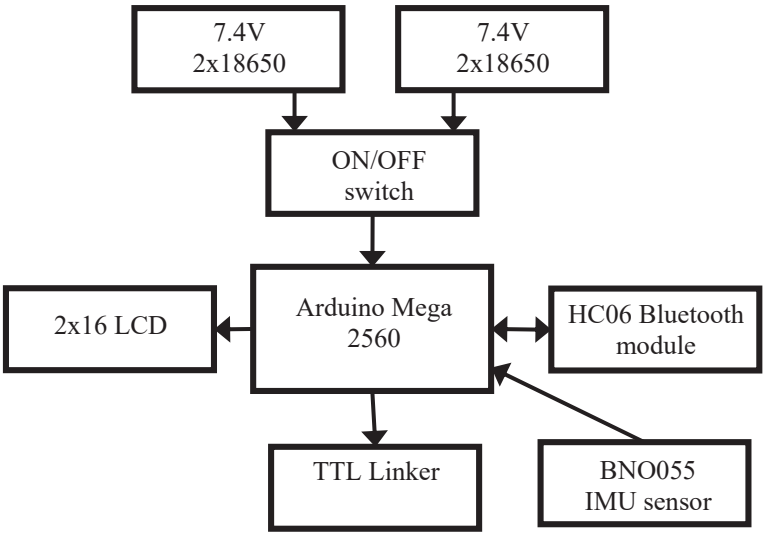


Figure 4.11 Block scheme of the electronic design

The photo of the electronic systems mounted on the robot main body is shown in Figure 4.12.

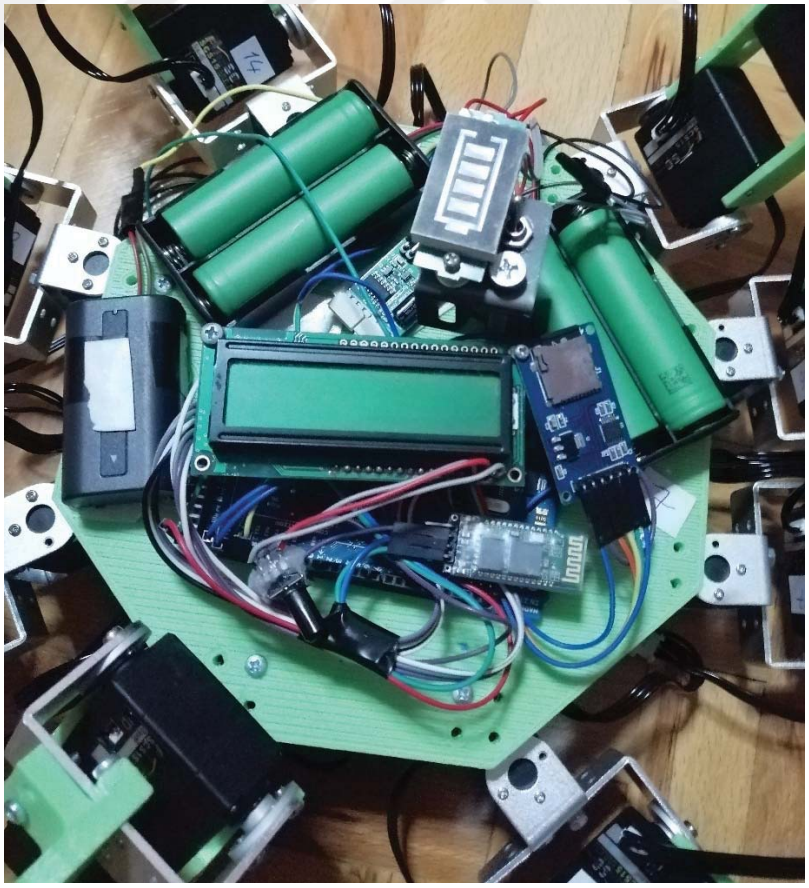


Figure 4.12 Electronical parts (Personal archive, 2019)

CHAPTER FIVE
SOFTWARE OF THE ROBOT

In this chapter, serial communication, servo serial communication protocol, bluetooth communication basics, App Inventor android software algorithm and software algorithms for the cards that will manage the robot are explained.

5.1 Servo Serial Communication

Servo serial communication protocol is a standard asynchronous serial communication. The available Baud rates in "bps" are 38400, 57600, 76800, 115200, 128000, 250000, 500000, 1000000. The factory default setting is 1000000 bps.

Servo motors have unique ID numbers. The range from 0 to 253 (0xFD) can be used (The factory default setting is ID 1), and, especially, 254(0xFE) is used as the Broadcast ID. If the Broadcast ID is used to transmit Instruction Packet, we can command to all SCServo. For servo ID definition, servo motor ID change algorithm is given in Figure 5.1.

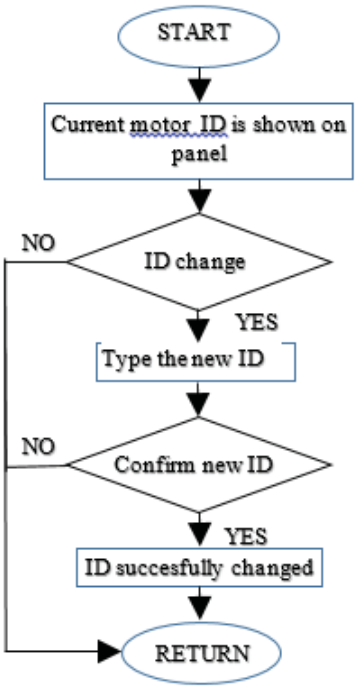


Figure 5.1 Servo motor ID change algorithm

5.2 ELER's Software

In this chapter, some details of the ELER's software are described.

5.2.1 ELER Robot Algorithm

The main algorithm of ELER developed on Arduino Mega can be analyzed in some parts. The main algorithm part is given in Figure 5.2.

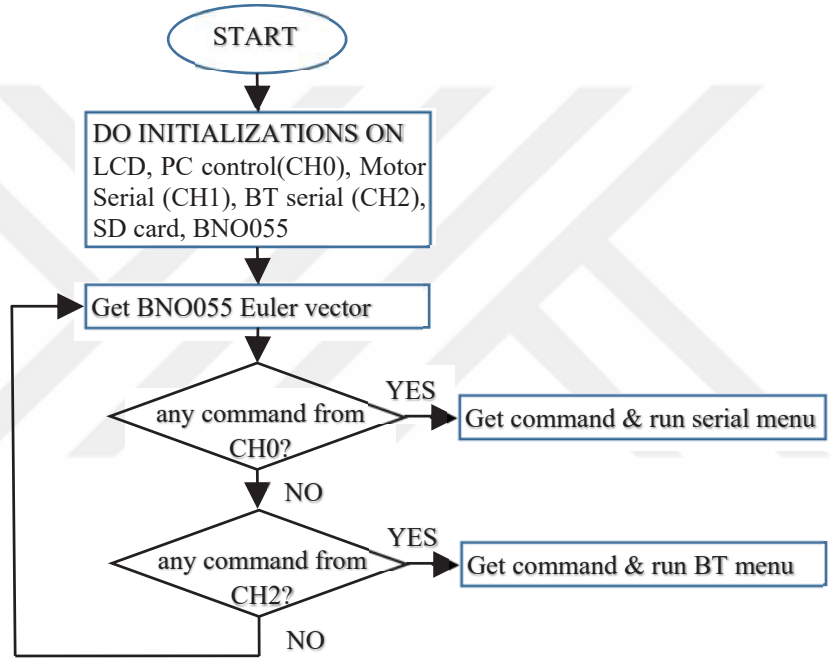


Figure 5.2 ELER main algorithm

ELER robot can be controlled using PC serial monitor. Here you can send some pre-defined commands from USB cable attached to ELER for testing purposes. Also the same commands can be send from a mobile device such as smartphones or tablets via Bluetooth. The list of commands is given in Table 5.1.

Table 5.1 ELER PC and bluetooth commands list

Character Received	**Command Name**	**Command function**
C	COMMANDS HELP	Print this command list on screen
Z	ZERO	Go to position 1
P <pos no>	POSITION	Go to <pos no> position
A	ADAPT THE CURRENT POSITION	Learn the current position
B	GO BACKWARD	Play backward position list
F	GO FORWARD	Play forward position list
L	GO LEFT	Play turning left position list
R	GO RIGHT	Play turning right position list
H	ADJUST HEIGHT	Adjust robot legs height
D	HEIGHT DEMO	Adjust robot height from bottom to top and the top to bottom for demo purpose
I <on/off>	INTERRUPT ON/OFF	BNO055 balancing action turn on and off

The algorithm of ELER serial menu subroutine is given in Figure 5.3. The ELER Bluetooth menu subroutine is the same except some extra commands. The numbers from 1 to 9 control the ELER pre-defined robot elevations.

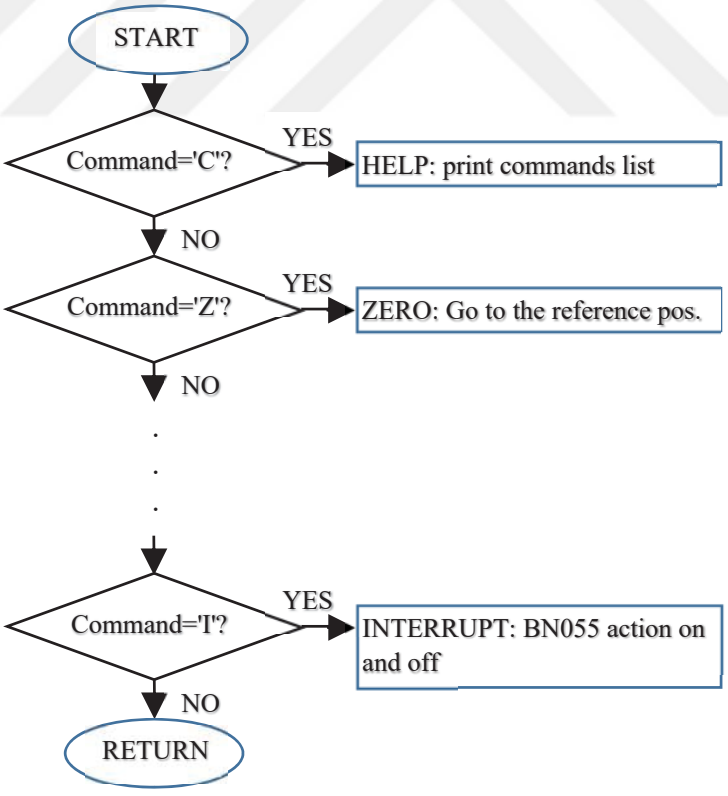


Figure 5.3 The algorithm of ELER serial and bluetooth menu

ELER's learning mode algorithm is shown in Figure 5.4. When the learning mode is initiated, the torques on the servo motors gets activated. In this mode we can give motion to the legs with our hands and the legs stays in the last position where they were last moved. After the torques are activated we give motion to the legs one by one with hand and these motions are saved in sd card. After giving the whole motion sequence the sequence gets saved to sd card. Then the recorded walk data is played by from sd card to perform the robot walk.

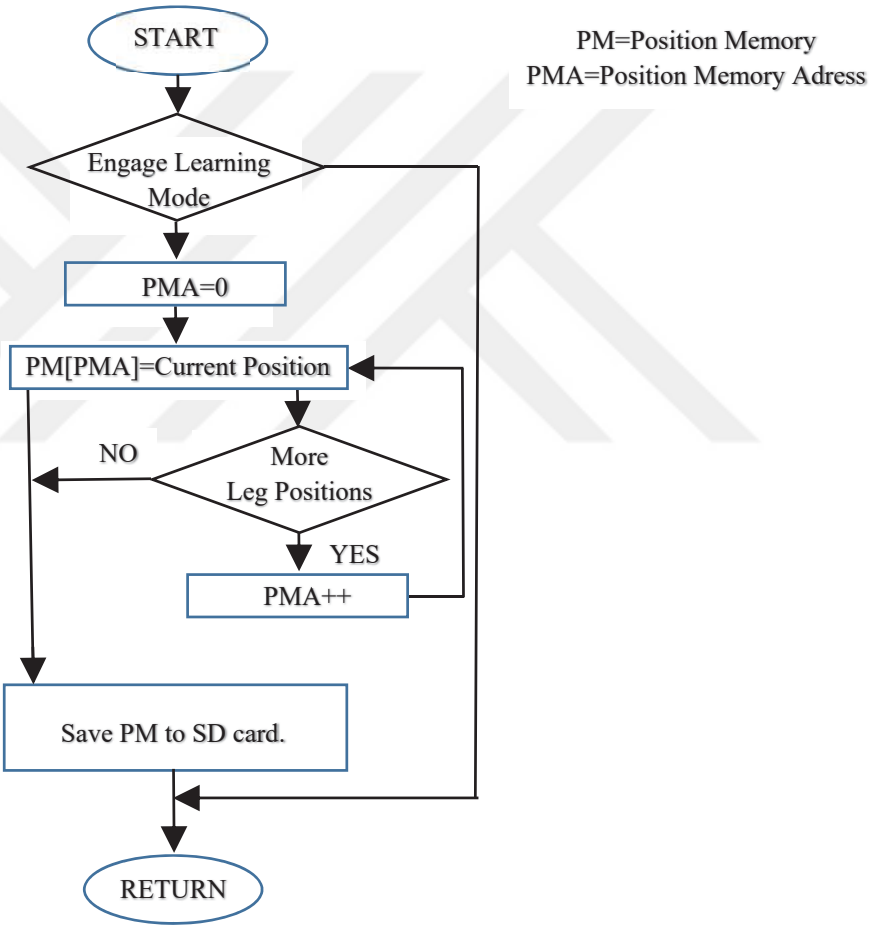


Figure 5.4 Learning mode algorithm

5.3. Mobile Device Android Software Algorithm

The robot performs the actions assigned to each character by defining the characters “A, a, B, b, …”, which are transmitted via Bluetooth as seen in Table 5.2

Table 5.2 Mobile device control characters

Character Sent	**Command Name**	**Command function**
W	INTERRUPT ON	BNO055 balancing action turn on
U	INTERRUPT OFF	BNO055 balancing action turn off
V	POSITION 1	Go to position 1
B	GO BACKWARD	Play backward position list
F	GO FORWARD	Play forward position list
L	GO LEFT	Play turning left position list
R	GO RIGHT	Play turning right position list
X	HEIGHT DEMO	Adjust robot height from bottom to top and the top to bottom for demo purpose
0…9	PRE DEFINED HEIGHTS	0: bottom height to 9: top height ELER robot elevations.

Android mobile software was written using App Inventor 2, which is developed by MIT (Massachusetts Technology Institute). Here the main logic is coding with visual program blocks and connect them each other in appropriate way. This makes the programming very easy.

The mobile android application codes are given here as program blocks. The first initialization part of the code arranges the bluetooth connection between the android mobile device and the ELER robot. A listpicker object named BT_list holds the bluetooth devices list paired with the android mobile device. Beforepicking method assigns the Bluetooth client address and names to the elements of the BT_list listpicker object. After the use picks the right bluetooth HC-06 slave device on the ELER, bluetooth client object makes a connection to this selection. After this, we can control the robot using our mobile device via bluetooth. The codes are given in Figure 5.5.

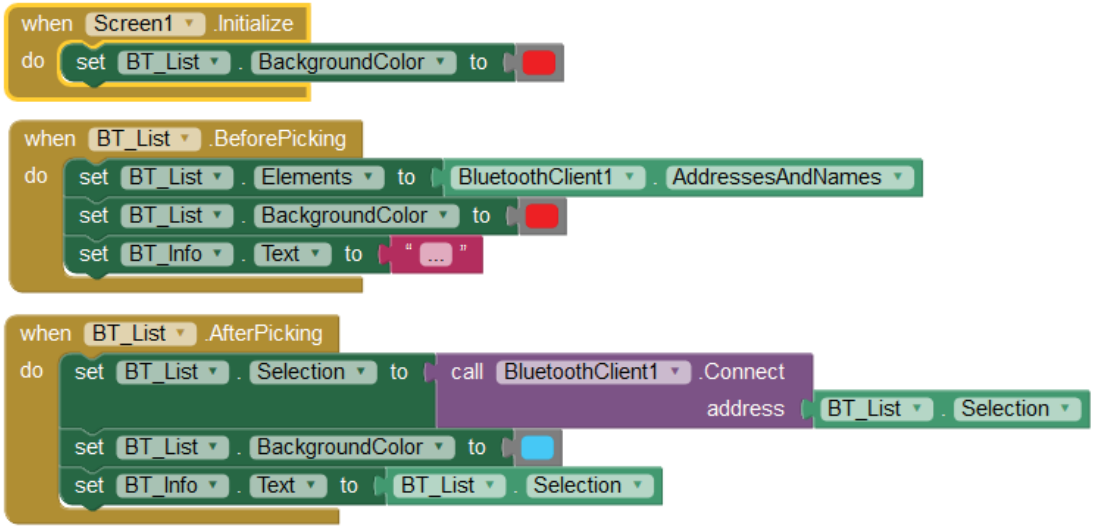


Figure 5.5 Codes of establishing bluetooth connection

After mobile device bluetooth was connected to the robot's HC-06 bluetooth slave module, the button click methods assigned to the movements send pre-defined alphabet letters to the robot. The block codes are given in Figure 5.6.

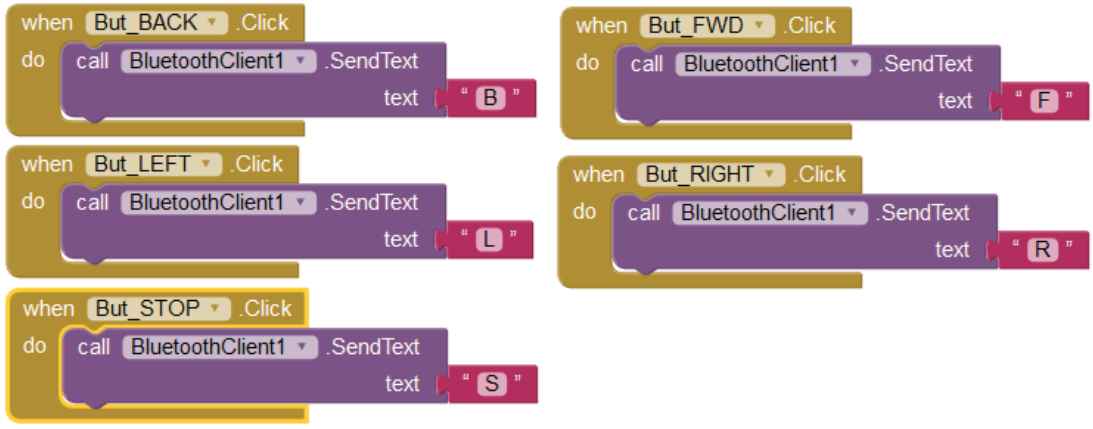


Figure 5.6 Robot movement codes

Robot elevation is adjusted by a slider. Slider left and right movement changes the value of the variable "val_elevation". The minimum value is "0" and the maximum value is "9" for the slider. If the slider value was changed, the updated value is sent as an ASCII code of the related number. The block codes are given in Figure 5.7.

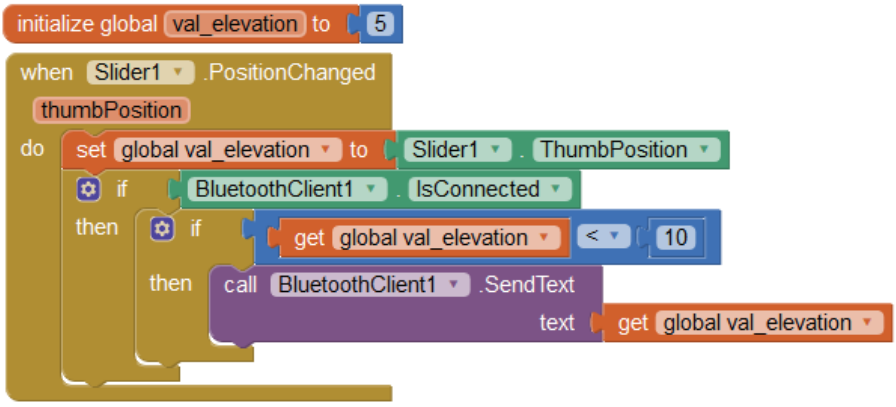


Figure 5.7 Robot elevation adjustments

The main screen of the user interface on mobile device is shown in Figure 4.9. Here BT selection button is used to select the robot HC-06 bluetooth from the mobil device paired bluetooth device list. Elevation slider adjusts the elevation level. Default is “5” and it can be changed between “0” and “9”. Movement control block has LEFT, FWD, STOP, RIGHT and BACK buttons that control the robot’s movements.

CHAPTER SIX
RESULTS

ELER has been developed according to the principles of mechanical and electronic design and the main steps of the control algorithm are explained in previous chapters. The mass of the robot without the electronical parts is about 2.3 kg.

When fully extended, the distance between the tips of the legs is 732 mm (Figure 6.1) and when fully ascended the height is 236.6 mm.

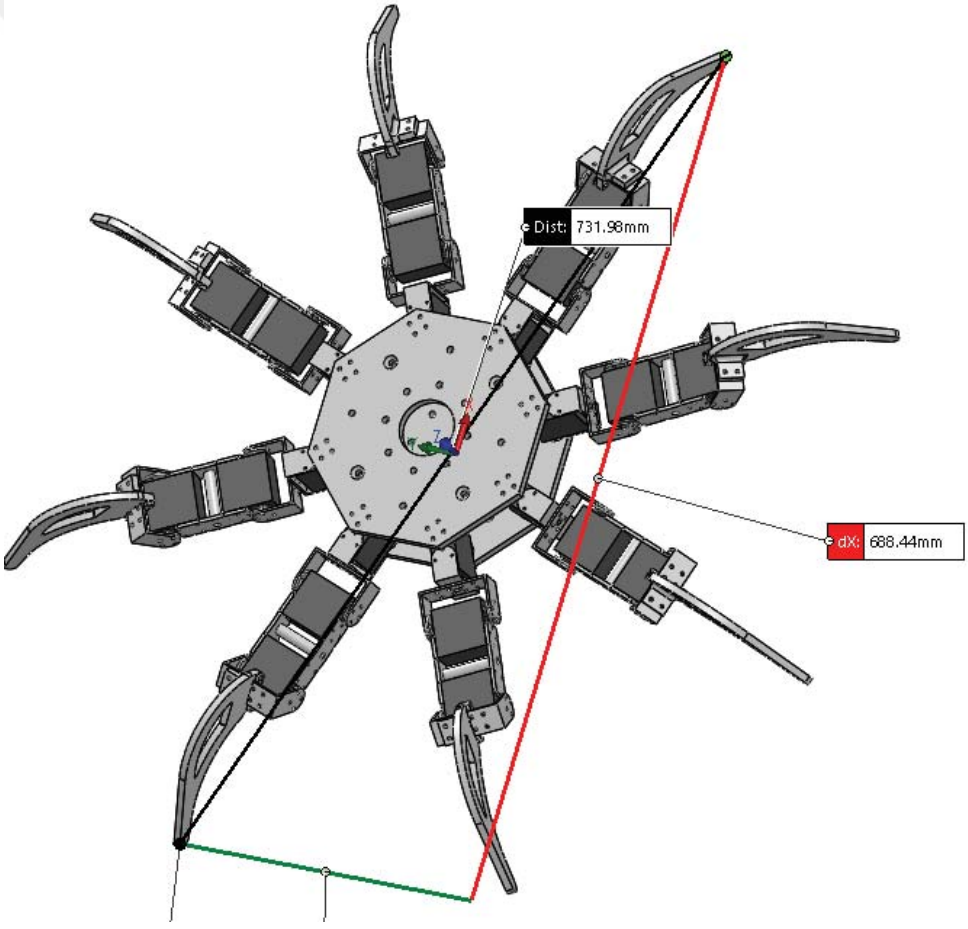


Figure 6.1 Fully extended dimensions of the robot

As a result of this study, the spider robot ELER, which can be controlled remotely on an android device, has been successfully manufactured.

Figure 6.2 shows the early design of the ELER. The leg lengths are longer than the current design, which causes mechanical problems and torque overloads on the joint motors.

Figure 6.2 Early design of ELER (Personal archive, 2018)

After strengthening the weak points, more rigid and durable design has been obtained as shown in Figure 6.3.

Figure 6.3 Final design of ELER (Personal archive, 2018)

The spider robot can ascend (Figure 6.3 and Figure 6.4) and descend to various heights and it can maintain its height while moving or rotating.

Figure 6.4 ELER in elevated position (Personal archive, 2018)

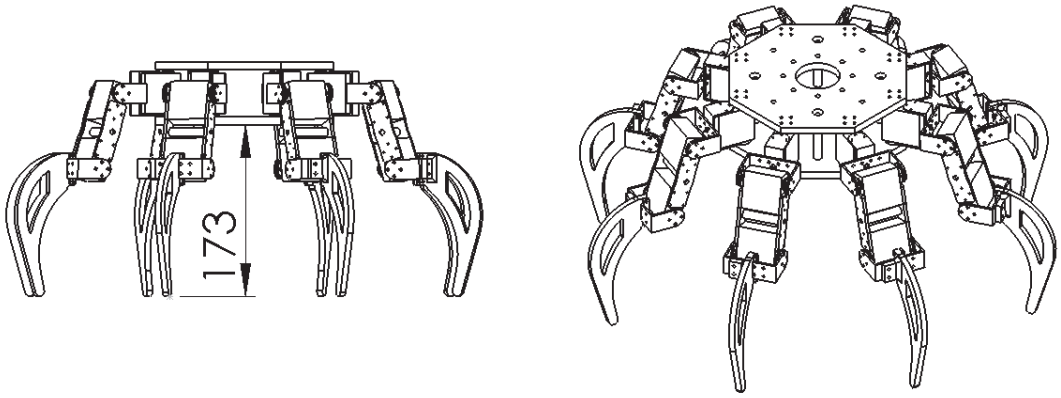


Figure 6.5 Technical drawing in elevated pose

ELER can be elevated up to 173 mm from the ground (Figure 6.5), which helps in overcoming the obstacles existing on the road. And it can maintain any height while moving or rotating around itself.

Robot control is achieved via bluetooth and mobile phone application. The Robot can be controlled via any mobile phone as long as it has a working Bluetooth and remote control application. Various position and angle data can be received from the IMU sensor attached on it.

Actions of the ELER are shown on the LCD display module while the robot is performing its motion, so we can check whether the robot is doing the desired motion or not.

Battery level can be checked from the battery display which is mounted on the ELER.

Matlab-Simulink simulation model was build by using the SolidWorks assembly model. Ascending and descending motion is the one of the most torque wise demanding motions of the robot and results were satisfying. Maximum allowed torques for the servo motors are 1.7 N.m and the simulation results give us the maximum needed torque for this movement is nearly 1 N.m which can be seen in Figure 6.6.

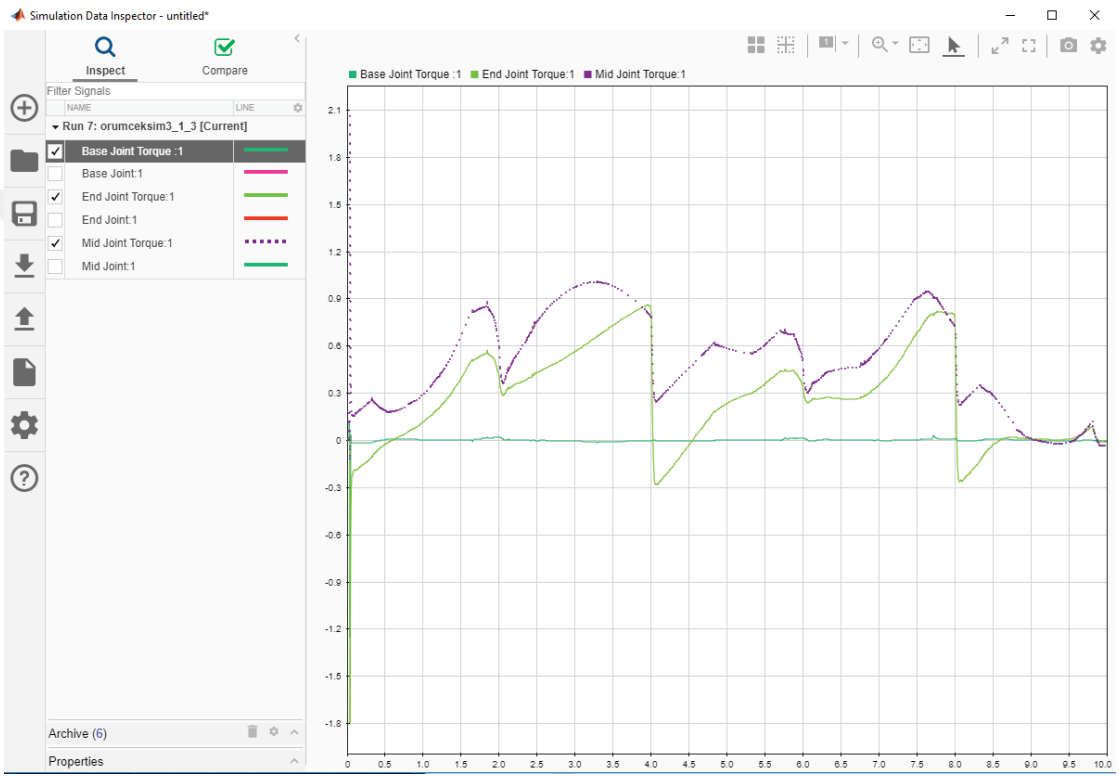


Figure 6.6 Matlab retrieved torque values from servo joints

To test the maximum load which the robot can carry, extra weight added to the main frame of the robot till the maximum measured torque is 1.7 N.m. After adding 1.5 kg extra load to the main frame the maximum torque is 1.7 N.m which can be seen in Figure 6.7.

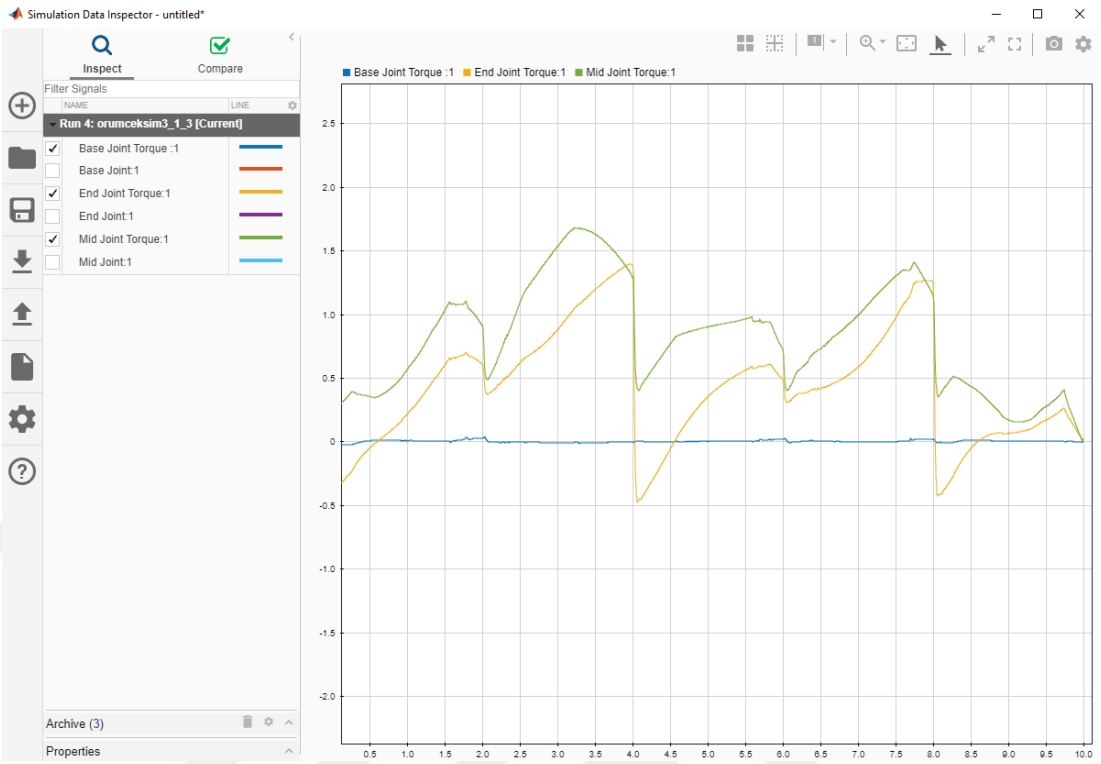


Figure 6.7 Matlab retrieved torque values from servo joints at max. load

When the batteries are fully charged the forward and backward motion has the speed of 0.3 m/s (1.08 km/h).

CHAPTER SEVEN
CONCLUSIONS

In the scope of this thesis, mechanical and electronic design of an eight-legged spider robot including its control software were considered. ELER's learning mode is unique, when the learning mode is enabled, servo motor torques get turned on just so the legs stay in their position, but the legs can still be moved by hand just with a little force. In the learning mode, ELER saves the rotation information of every servo motor and writes them on a SD card. By observing real life arachnids robot is taught to move like them. Just learning to move forward, we can make the robot go backwards by reversing the forward code and by learning one rotation, ELER can rotate in other direction just by reversing the servo motor sequence. Each motor position can be seen thanks to feedback mechanism of the servo motors and we can manipulate the code by hand so that ELER's movement is smooth.

ELER has so much to be improved further. Because of its learning mode any kind of movement can easily be uploaded to the robot. Since it has smart servo motors, the robot can feedback the value of position, temperature, load, speed and input voltage. We can see which walking gaits or patterns are much more sufficient and we can also determine which walking pattern or gait are more resourceful when the robot needs to speed up or slow down.

While moving from point A to point B we can use different routes and movement gaits to see which path and gait combination is the most efficient one thanks to smart servo motors and IMU sensor.

In the future works, if programmed the robot also can alter its movement infinite times to see which walking pattern is most efficient to use. Then it can advance on that walking pattern for the specified path. ELER can capitalize on its IMU sensor to find balance on uneven surfaces and rough terrains. This self learning combined with the IMU sensor enables lots of possibilities for the ELER.

REFERENCES

2x16 LCD module (n.d.) Retrieved June 14, 2019, from https://www.direnc.net/2x16-lcd-display-sol-ust-mavi-qapass-2x16-lcd-display-qapass-32859-45-B.jpg

Angeles J. (2014) *Fundamentals of robotic mechanical Systems* (4th ed.). Swizterland: Springer International Publishing.

Arduino MEGA 2560 Board (n.d.) Retrieved June 24, 2018, from https://images-na.ssl-images-amazon.com/images/I/61cCxvD3iLL._SL1000_.jpg

Battery Holder (n.d.) retrieved June 14, 2019, from https://www.robotshop.com/ca/en/dfrobot-2-18650-battery-holder.html

Billah M., Ahmed M. & Farhana S. (2008) Walking hexapod robot in disaster recovery: developing algorithm for terrain negotiation and navigation. *World Academy of Science, Engineering and Technology International Journal of Mechanical and Mechatronics Engineering, 2* (4), 795-800.

Bluetooth Transceiver Module HC-06, 2017. Retrieved June 14, 2019, from http://wiki.sunfounder.cc/index.php?title=Bluetooth_Transceiver_Module_HC-06

BNO055, (n.d.) Retrieved June 14, 2019, from https://www.robotistan.com/adafruit-bno055-9-dof-mutlak-oryantasyon-imu-6315-24-O.jpg

Defense Acquisition University Press (2001) Modeling and Simulation. *Systems Engineering Fundementals* (117-125). Fort Belvoir, Virginia

Feetech SCS15, (n.d.) Retrieved June 24, 2018, from https://ae01.alicdn.com/kf/HTB1kvC5XTHuK1RkSndVq6xVwpXaA.jpg

Futuba specs, (n.d.) Retrieved June 24, 2018, from https://www.futabarc.com/servos/specs-lineart/specs-futm0260.jpg

Hasnaa E. H. & Mohammed B. (2017) Planning tripod gait of an hexapod robot. 14th *International Multi-Conference on Systems, Signals & Devices (SSD),* 163-168

HC-06 (n.d.) Retrieved June 25, 2018, from https://img.dxcdn.com/productimages/sku_382686_1.jpg

Inagaki K. & Kobayashi H. (1992) Adaptive wave gait for hexapod synchronized walking. *Proceedings of the 1994 IEEE International Conference on Robotics and Automation, 2,* 1326-1331

Klaassen B., Linnemann R., Spenneberg D. & Kirchner F. (2002) Robotics an autonomous systems. *Ninth International Symposium on Intelligent Robotic Systems, 41,* 65-164

Klaassen B., Linnemann R., Spenneberg D. & Kirchner F. (2003) Biologically inspired robot design and modeling. *11th International Conference on Advanced Robotics (ICAR),* 576-581

Liang O. (2018) *Inverse kinematics implementation for hexapod robots and quadruped robots.* Retrieved June 23, 2018, from https://oscarliang.com/inverse-kinematics-implementation-hexapod-robots/

Li-Ion charger (n.d.) Retrieved June 14, 2019, from https://www.liteshop.com.au/media/catalog/product/cache/1/image/9df78eab33525d08d6e5fb8d27136e95/v/c/vc4-battery-charger-xtar__60979.1492717694.jpg

Miller M. & Miller R. (2017) *Robots and robotics: principles, systems, and industrial applications* (1st ed.). United States of America: McGraw-Hill Education

NCR18650 (n.d.) Retrieved June 14, 2019, from https://www.batteryspace.com/images/products/detail/8678.png

Pabbu A. S. (2017) Incorporating passive compliance for reduced motor loading during legged walking. B.Tech, Jawaharlal Nehru Technological University, India

Rohmer E. (2009) Retrieved June 9, 2018, Lunar Exploration Omnidirectional Netbot (LEON), from http://www.astro.mech.tohoku.ac.jp/~eric/sitePages/leon.html

Slatour (2016) *Smart Servo: The Difference Between Smart and Regular Servos,* Retrieved June 16, 1995, from https://www.robotshop.com/community/blog/show/smart-servo-the-difference-between-smart-and-regular-servos.

Tedeschi F. & Carbone G. (2014) Design issues for hexapod walking robot. *Robotics*, *3*, 181-206.

TTlinker schematics (n.d.) Retrieved June 14, 2019, from https://sc02.alicdn.com/kf/HTB13rGhMVXXXXbAXFXXq6xXFXXXy/220726101/HTB13rGhMVXXXXbAXFXXq6xXFXXXy.jpg

TTlinker (n.d.) Retrieved June 14, 2019, from https://static.rapidonline.com/catalogueimages/product/s37-1333p01wm.jpg

Wettergreen . & Thorpe C. (1992) Gait Generation for Legged Robots. *Proceedings of the IEEE International Conference on Intelligent Robots and Systems, 2,* 1413-1420

Zak M. & Rozman J. (2015) Design, construction and control of hexapod walking robot. *2015 IEEE 13th International Scientific Conference on Informatics,* 302-307.

Zhang H., Liu Y., Zhao J., Chen J. & Yan J. (2017) Development of a bionic hexapod robot for walking on unstructured terrain. *Journal of Bionic Engineering, 11*, 176-187.

APPENDICES

APPENDIX 1: Arduino Codes For the Servo ID Change

```
/* MOTOR PROFILER AND ID CHANGER */
#include "SCServo.h"
SCServo motor;

// info form
#define NO_INFO  0
#define LO_INFO  1
#define HI_INFO  2

int LED=13;

u8  no,sonuc,i,newno,junk,oku,idchange;
u8  motorid,baud,sicak,voltaj;
s16 retdeltime, modelno,
versiyon,minanglim,maxanglim,maxtork,prepos,prespeed,punc
h;
s16 pvalue,ivalue,dvalue,preload;

void setup()
{ pinMode(LED,OUTPUT);
  Serial.begin(115200); Serial1.begin(1000000);
delay(500);

  no=0xfe;
  Serial.println("----- OLD MOTOR PARAMETERS -----");
InfoShow(no,HI_INFO);

  Serial.println("Enter new motor id : ");
```

```
  while (Serial.available() == 0) ;  // Wait here until
input buffer has a character
  { newno = Serial.parseInt();  Serial.print(newno, DEC);
    while (Serial.available()>0) junk = Serial.read(); //
clear the keyboard buffer
  }
 Serial.println("");
   Serial.println("ID CHANGE OR NOT  (Yes:1 No: 0)");
  while (Serial.available() == 0) ;  // Wait here until
input buffer has a character
  { idchange = Serial.parseInt();
    while (Serial.available()>0) junk = Serial.read(); //
clear the keyboard buffer
  }

  if (idchange==1)
  {
    Serial.println("ID CHANGING NOW");
  motor.LockEeprom(no,0); delay(1000);

  sonuc=motor.WriteID(no,newno); delay(4000);
Serial.print(" with ERROR: "); Serial.println(sonuc);
  //sonuc=motor.WriteBaund(newno,6); delay(3000);
Serial.print(" with ERROR: "); Serial.println(sonuc);

  motor.LockEeprom(no,1); delay(1000);
  Serial.println("-------------------");
  }

  // TESTING THE SERVO MOTION
  sonuc=motor.EnableTorque(0xfe,1); // torkları ac
  Serial.print(" TORQUE ON with ERROR: ");
Serial.println(sonuc);  delay(500);
```

```
  no=newno;
  motor.WritePos(newno,512, 10); delay(3000);
  prepos=motor.ReadReg16(no,P_PRESENT_POSITION );
  Serial.println("------ SERVO TEST -----");
  Serial.print("PRESENT POS   : ");
Serial.println(prepos);
    Serial.println("----- NEW MOTOR PARAMETERS -----");
InfoShow(newno,HI_INFO);
 motor.WritePos(newno, 0, 300); delay(1000);
/*
  motor.WriteLimitAngle(newno, 0, 0);
  Serial.println("------ WHEEL TEST -----");
  motor.WheelDir(newno, 0);
  motor.WritePos(newno, 0, 512);
  // motor.WriteSpeed(newno, 512);
  prespeed=motor.ReadReg16(no,P_PRESENT_SPEED  );
  Serial.print("PRESENT SPEED (CCW) : ");
Serial.println(prespeed);

  delay(2000);
  motor.WheelDir(newno, 1);
  motor.WritePos(newno, 512, 512);
  //motor.WriteSpeed(newno, 2000);
  prespeed=motor.ReadReg16(no,P_PRESENT_SPEED);
  Serial.print("PRESENT SPEED (CW) : ");
Serial.println(prespeed);
  delay(2000);
  motor.WriteLimitAngle(newno, minanglim, maxanglim);


  delay(1000);
  motor.WritePos(newno, 0, 100); delay(1000);
  prepos=motor.ReadReg16(no,P_PRESENT_POSITION );
```

```
  Serial.println("------ SERVO TEST BACK -----");
  Serial.print("PRESENT POS   : ");
Serial.println(prepos);
  motor.WritePos(newno, 512, 100); delay(1000);

  Serial.println("----- NEW MOTOR PARAMETERS -----");
InfoShow(newno,HI_INFO);
  motorid=motor.ReadReg(newno,P_ID);
  Serial.print("ID  : "); Serial.println(motorid);

*/

  sonuc=motor.EnableTorque(0xfe,0); // torkları kapat
  Serial.print(" TORQUE OFF with ERROR: ");
Serial.println(sonuc);  delay(500);

}

void loop() {}

//-----------------------------------------
SUBROUTINES
void InfoShow(u8 ID,int infoform)
{
  if ((infoform==LO_INFO) || (infoform=HI_INFO))
  {
  motorid=motor.ReadReg(no,P_ID);
  baud=motor.ReadReg(no,P_BAUD_RATE);
  Serial.print("ID            : ");
Serial.print(motorid);
  Serial.print("\t\tBAUD ( "); Serial.print(baud);
Serial.print(" )    : ");
Serial.println(1000000/(baud+1));
```

```
  }

  if (infoform==HI_INFO)
  {
  sicak=motor.ReadReg(no,P_PRESENT_TEMPERATURE);
  voltaj=motor.ReadReg(no,P_PRESENT_VOLTAGE);
  retdeltime=motor.ReadReg(no,P_RETURN_DELAY_TIME);
  modelno   =motor.ReadReg16(no,P_MODEL_NUMBER);
  versiyon =motor.ReadReg16(no,P_VERSION);
  minanglim=motor.ReadReg16(no,P_MIN_ANGLE_LIMIT);
  maxanglim=motor.ReadReg16(no,P_MAX_ANGLE_LIMIT);
  maxtork=motor.ReadReg16(no,P_MAX_TORQUE );
  prepos=motor.ReadReg16(no,P_PRESENT_POSITION );
  prespeed=motor.ReadReg16(no,P_PRESENT_SPEED  );
  punch=motor.ReadReg16(no, P_PUNCH );
  pvalue=motor.ReadReg16(no, P_COMPLIANCE_P  );
  ivalue=motor.ReadReg16(no, P_COMPLIANCE_I  );
  dvalue=motor.ReadReg16(no, P_COMPLIANCE_D  );
  preload=motor.ReadReg16(no, P_PRESENT_LOAD);

  Serial.print("TEMPERATURE   : "); Serial.print(sicak);
  Serial.print("\t\tVOLTAGE       : ");
Serial.print(voltaj);
  Serial.print("\t\tRET DELAY TIME: ");
Serial.println(retdeltime);
  Serial.print("MDL           : ");
Serial.print(modelno);
  Serial.print("\t\tVER           : ");
Serial.println(versiyon);
  Serial.print("MIN ANG LIMIT : ");
Serial.print(minanglim);
  Serial.print("\t\tMAX ANG LIMIT : ");
Serial.print(maxanglim);
```

```
  Serial.print("\t\tMAX TORQUE    : ");
Serial.println(maxtork);
  Serial.print("PRESENT POS   : "); Serial.print(prepos);
  Serial.print("\t\tPRESENT SPEED : ");
Serial.print(prespeed);
  Serial.print("\t\tPRESENT LOAD  : ");
Serial.println(preload);
  Serial.print("P VALUE       : "); Serial.print(pvalue);
  Serial.print("\t\tI VALUE       : ");
Serial.print(ivalue);
  Serial.print("\t\tD VALUE       : ");
Serial.println(dvalue);
  Serial.print("PUNCH         : ");
Serial.println(punch);
  Serial.println("-----------------------");
  }
}
```