FACULTY OF ENGINEERING AND ARCHITECTURE
DEPARTMENT OF MECHATRONICS ENGINEERING

DESIGN AND MANUFACTURING PROTOTYPE WALKING ROBOT WITH TWO DOF CLOSED KINEMATICS CHAIN LEGS

AND

A PIPELINE INSPECTION ROBOT

GRADUATION PROJECT

ÖMER BURAK BAKAR 140412011
BARAN GÖZETİR 150412037
MEHMET ÖZALP 140412041

SUPERVISOR: Assist. Prof. Dr. Fatih Cemal Can

JUNE 2020

DESIGN AND MANUFACTURING PROTOTYPE WALKING ROBOT WITH TWO DOF CLOSED KINEMATICS CHAIN LEGS

AND

A PIPELINE INSPECTION ROBOT

A GRADUATION PROJECT

Approved by:

Assist. Prof. Dr. Fatih Cemal Can

(Signature)

OMER BURAK BAKAR	MEHMET ÖZALP
BARAN GÖZETİR	

JUNE 2020

ABSTRACT

Our Project contains two different sections. In section one our aim is to design and manufacture prototype a walking robot with two dof closed kinematic chain legs. However, we designed and manufactured a pipeline inspection robot also by using same kinematic chain legs with a different design in second section. Most of the walking robot prototypes are made by one degree of freedom with complex leg mechanisms to obtain walking trajectory. In one dof systems, you need to change links lengths to change trajectory.In our Projects, we use two motors with fivebar mechanism. Thanks to two degree of freedom mechanism, we can arrange any trajectory within our workspace.We are using six legs in our designs, three legs are working simultaneously to obtain smooth walking in our first robot(centipede), for our second robot(pipelinebot) two legs are working simultaneously in order to squeeze itself to pipe.Also each servo motor are working independently, so we can arrange several movements besides walking.

Keywords: Two degrees of freedom, chain legs, fivebar mechanism, pipeline inspection, trajectory

ÖZET

Projemiz iki farklı bölümden oluşmaktadır. İlk bölümde iki serbestlik dereceli kapalı kinematik zincir ayaklar ile yürüyücü robot tasarımı ve üretimi yapılmıştır. Ayrıca, ikinci bölümde boru içerisinde inceleme yapabilen robotumuz tasarlanmış ve montajı yapılmıştır. Yürüme yörüngesini elde etmek için yürüyücü robot prototiplerinin çoğu tek serbestlik dereceli ve karmaşık ayak mekanizmaları ile tasarlanmaktadır.Bu tip sistemlerde yörüngeyi değiştirmek için ayak parça boylarının değiştirilmesi gerekmektedir.Biz projelerimizde her ayak için beş bar mekanizması ile iki motor kullanıyoruz.İki serbestlik derecesine sahip olduğumuz için çalışma alanı içerisinde istediğimiz yörüngeyi çizebiliyoruz.Sistemimizde toplamda altı ayak kullanıyoruz, böylece ilk robotumuzda(centipede) üç ayak simultane hareket edip lokomosyonu sağlıyor, ikinci robotumuzda(pipelinebot) iki ayak simultane hareket edip robotumuzun kendisi boruya sıkıştırmasını sağlıyor.Ayrıca bütün motorlar birbirinden bağımsız çalıştığı için yürüme hareketinin yanında daha çeşitli hareketler elde edebiliyoruz.

Anahtar kelimeler: İki serbestlik dereceli, zincir ayaklar, beş bar mekanizması, boru hattı inceleme, yörünge

ACKNOWLEDGMENTS

We would like to thank our supervisor Assist. Prof. Dr. Fatih Cemal Can for his support during the project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SYMBOLS

θ theta

V volt

mm millimeter

N newton

W watt

kg kilogram

ACRONYMS

dof degree of Freedom

3D three dimensional

sec second

li-po lithium polymer

SECTION 1

DESIGN AND MANUFACTURING PROTOTYPE WALKING ROBOT WITH TWO DOF CLOSED KINEMATICS CHAIN LEGS CENTIPEDE

I. INTRODUCTION

1.1 Most Common Leg Mechanisms

1.1.1 Jansen's Linkage

Jansen's linkage is a planar leg mechanism designed by Theo Jansen[1].This mechanism has 1 dof and can be controlled by single actuator.

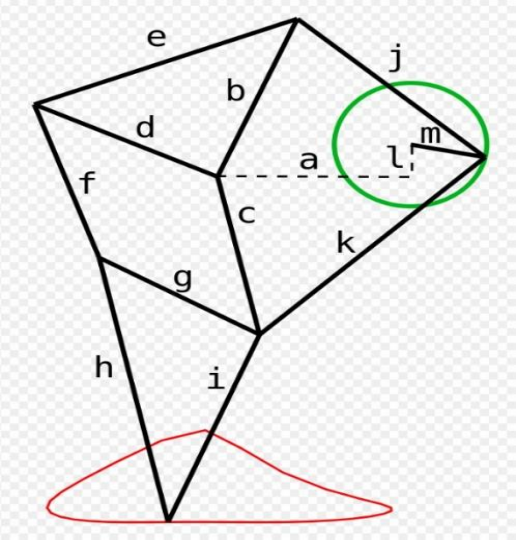


Figure I-1 *Jansen's mechanism***[1]**

1.1.2 Klann linkage

Klann's linkage is designed by Joe Klann to mimic animals leg movements[2].Different than Jansen's linkage, Klann's linkage can be used on rough terrain.

The Klann mechanism uses six links per leg, when the Jansen's linkage uses eight links per leg.

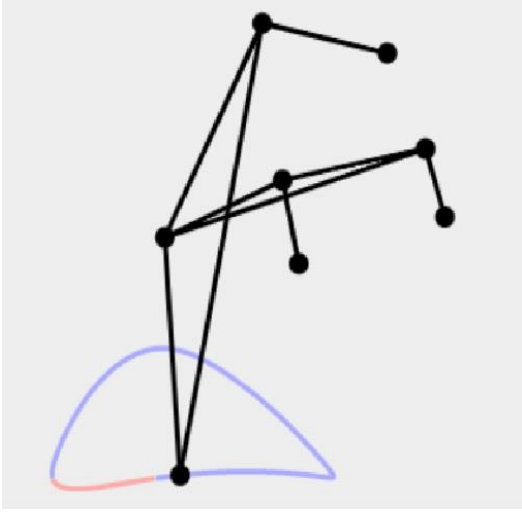


Figure I-2 *One degree of freedom klann mechanism***[2]**

1.1.3 Two dof pantograph leg mechanism

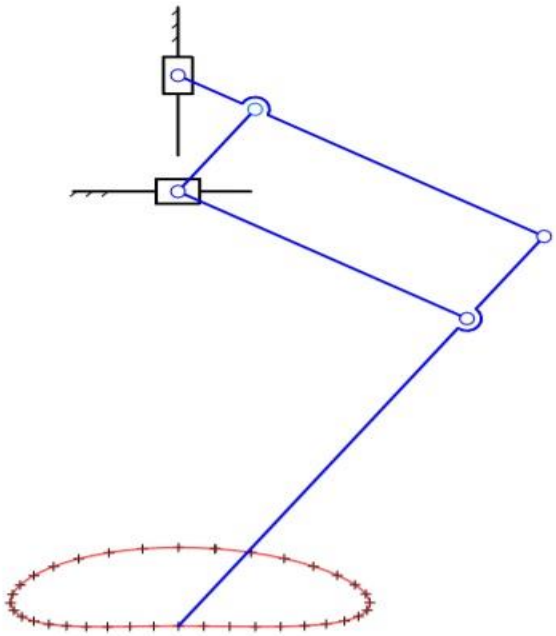


Figure I-3 *Two dof pantograph leg mechanism* **[3]**

An example of a mechanism driven by two sliders is given in figure Figure I-3.Two sliders are actuated by linear motors which are used to obtain periodic motion.

1.1.4 Chebyshev linkage

Chebyshev linkage is used to convert rotational motion to linear motion by using fourbar linkage.

Also, chebyshev lambda mechanism is used to obtain leg locomotion.We can obtain this motion by using single actuator.

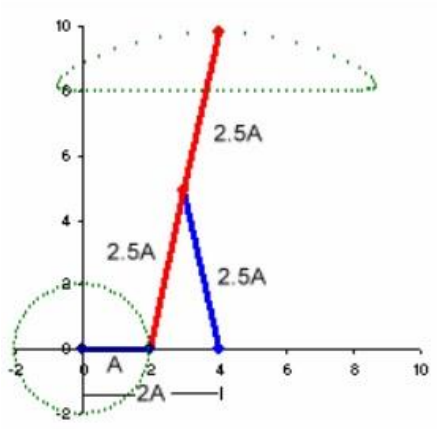


Figure I-4 *Chebyshev linkage* **[4]**

1.1.5 Peaucellier Mechanism

Peaucellier mechanism is constructed to have exact straight line.Our linkage is similar to this linkage, we don't use BA and BC links in order to make several paths and at O point we seperate links and put two servo motors[6].

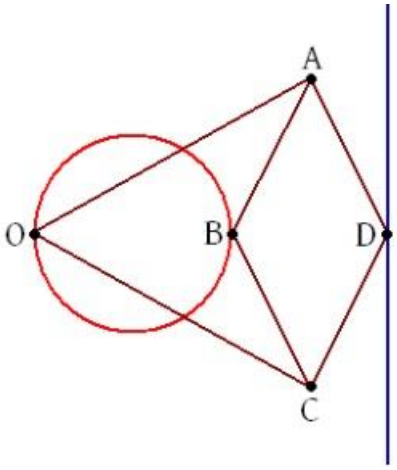


Figure I-5 *Peaucellier Mechanism* **[6]**

1.2 A recent research about new walking mechanism

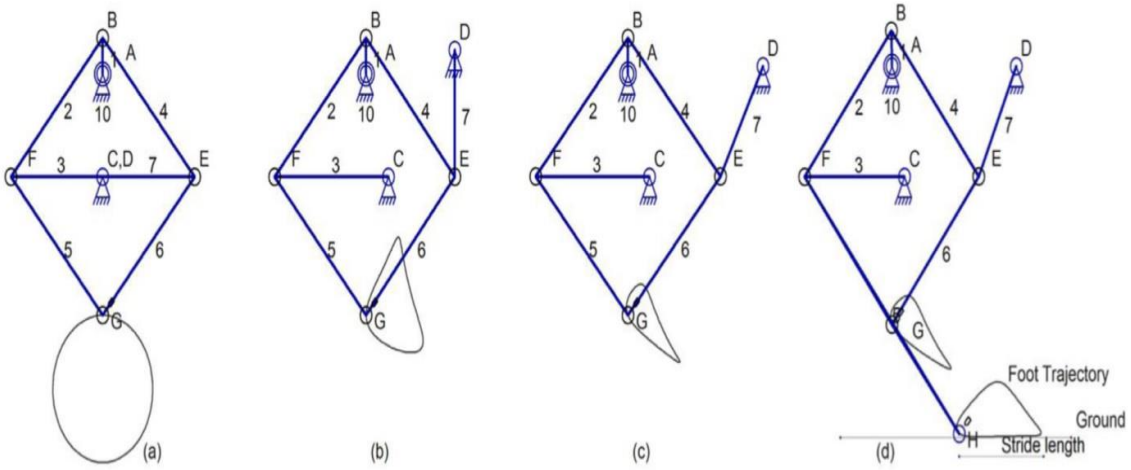


Figure I-6 *New walking mechanisms presented in a recent research***[5]**

This work is done by changing peaucellier mechanism.

Here, in figure 6.a by rotating link 1 a circular motion is obtained.After that D point is seperated and fixed 115 degree to x axis in figure 6.c.Finally,link FG is extended.

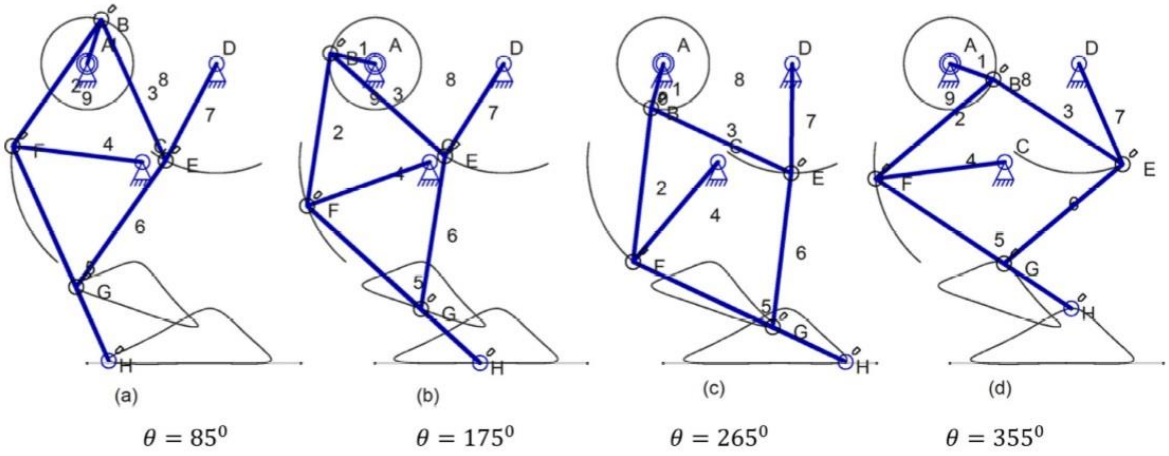


Figure I-7 Extended walking mechanism**[5]**

Here, in figure 7, 4 different position is shown according to angles. Authors show that walking trajectories can be obtained using closed kinematic chains.

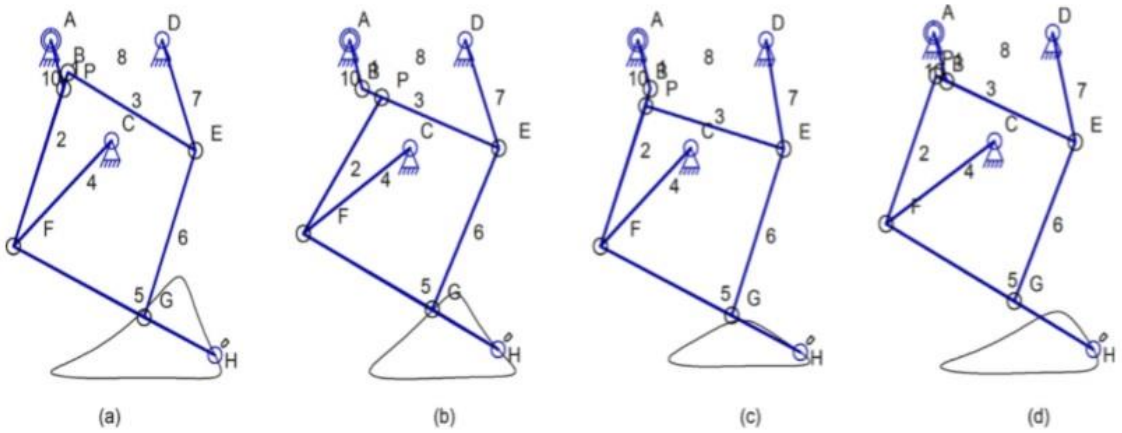


Figure I-8 *Different trajectories from some closed kinematic chains***[5]**

Here, in Fig.8 several foot trajectories are shown.By changing lenghts or positions of links different trajectories are obtained.In our Project we don't need to change parameters,we are going to obtain different trajectories by solving inverse kinematic problem of two dof fivebar mechanism.

What is walking robot and why we need these type of robots?

Walking robots are commonly used on rough surfaces where wheeled robots can not operate.Walking robots can operate on hills,caves,jungles,desert,mountains etc.Obstacles are main problem for wheeled robots.

Due to their mechanical structures links can transmit force, so required power and energy consumption decreases,this decreases the cost of project.Walkers are generally operated with smaller motors.

1.3 Studies about leg locomotion

First studies about leg locomotion are made by Eadweard Muybridge[7].Eadweard Muybridge has taken several photos of multi legged animals.When he added these photos he obtained a trajectory.

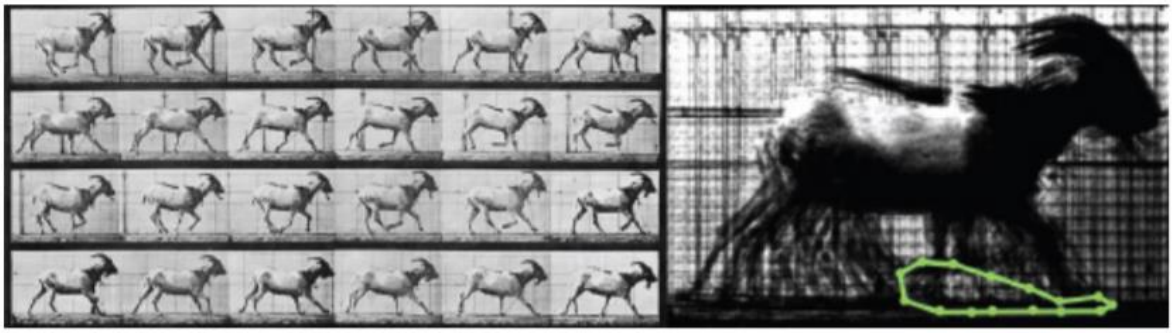


Figure I-9 *Walking trajectory of an animal***[7]**

At the end and beginning of this aerial phase there is occuring instant force and torque changing.In order to reduce this effect trajectory should be defined properly.Well defined trajectories reduces this effect, so instant torque and force distribution to legs decreases ,also energy consumption decreases and system becomes stable.For our square trajectory this effect will be bigger, however, for smooth trajectory this effect will be much more less.(see Figure IV-2 and Figure IV-4)[7]

In the field of manipulation Yoshikawa has described a manipulation measure of the robot's workspace. Transferring this insight to the kinematics of a leg, there are better regions for motion trajectories.By using dynamic equations for legged-locomotion better trajectories can be obtained. Energy-optimized trajectories are used in robotic manufacturing projects, but have rarely been used in leg trajectories for multi-legged walking robots.[7]

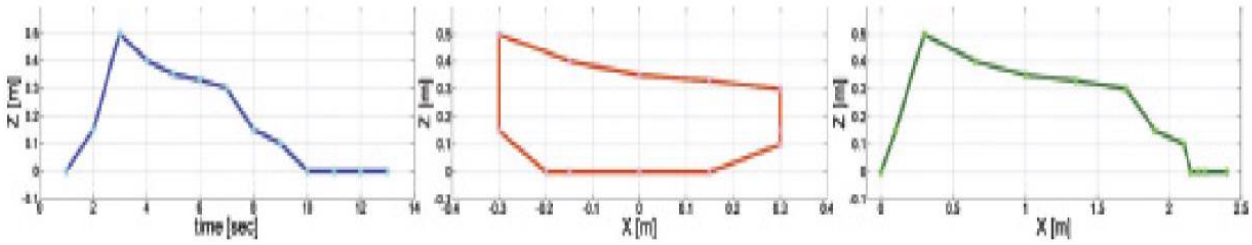


Figure I-10 *Cartesian representations of leg trajectories time based(left),coordinate based no velocity(middle),coordinate based including system velocity(right)[7]*

1.4 Comparison of previous Projects and our Project

There are lots of walking robot Projects, some of them are;

- Posture Control of 6-leg Walking Robot investigated by ,Toshio Fukuda', Yuji Adachi', Haruo Hoshino2, Kazuhiro Kosuge' , Isao Matsunaga2 and Fumihito Arai'[10]
- Analysis and Evaluation of the Stability of a Biologically Inspired, Leg Loss Tolerant Gait for Six- and Eight-Legged Walking Robots presented by ,Martin G¨orner and Gerd Hirzinger[8]
- Design and prototype of a six-legged walking insect robot, by Servet Soyguder and Hasan Alli Mechanical Engineering Department, Firat University, Elazig, Turkey[9]

Our Project is similar to six legged walking insect robot projects. The differences are;

- We do not use rotary cam shaft in order to control each leg independently.
- Our leg mechanism is fivebar and has 2 servo motors each leg, so we can arrange path for each leg independently.
- Our system has 2 dof,so we can arrange different paths by giving end coordinate of robot, by using inverse kinematic equations our microcontroller is solving these equations and gives motors actuator angles.
- Another advantage of 2 dof system is we able to move forward,backward,turn right and left with ease.
- Also, our walker can operate on rough terrain,climb the obstacles etc. thanks to 2 dof system.
- Since,we can draw any trajectory within our workspace, our Project contains detailed trajectory researches and implementations.

II. DESIGN AND SIMULATION RESULTS

2.1 Conceptual Design

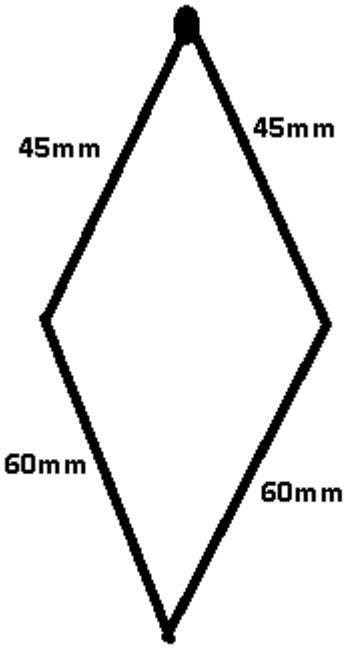


Figure II-1 *Conceptual design of leg*

One of our main purposes is to find most efficient trajectory, so we wanted a small system for easy implementation. We decided link lenghts to be as 45mm and 60mm.

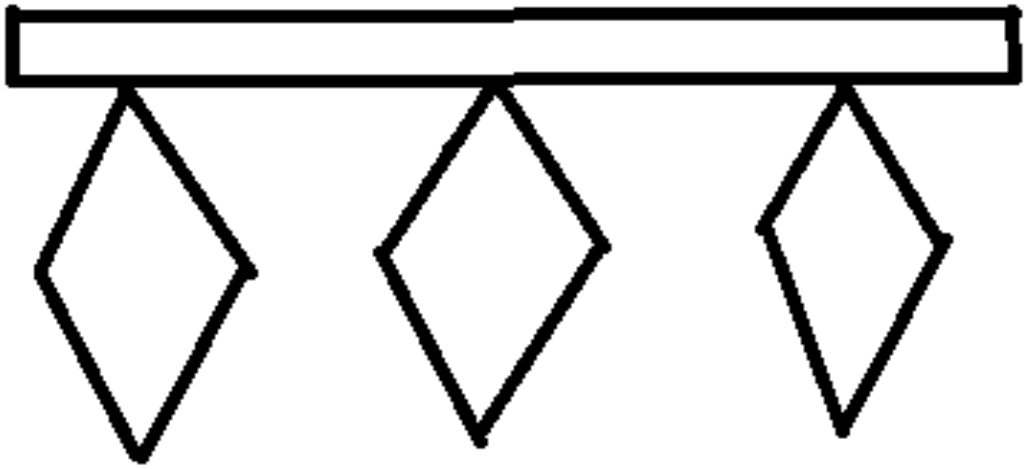


Figure II-2 *Conceptual design of walker*

Because of locomotion, we have to use at least three legs in one side. So,we decided number of legs to be six.

2.2 Detailed design

2.2.1 Technical Drawings

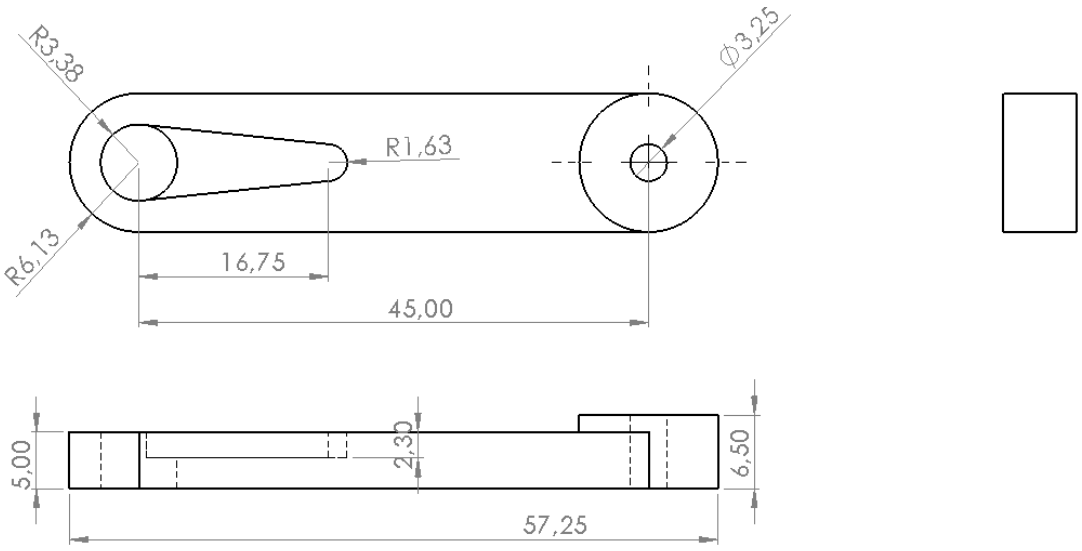


Figure II-3 *Connection rod*

We are connecting motor and small link by using servo arm (see figure II-7).In order to do that we draw a pocket with 2.30mm depth.

We used a steel bar with 3mm diameter to assemble links.It’s connection is shrink fit.We are not using bearing in small link.

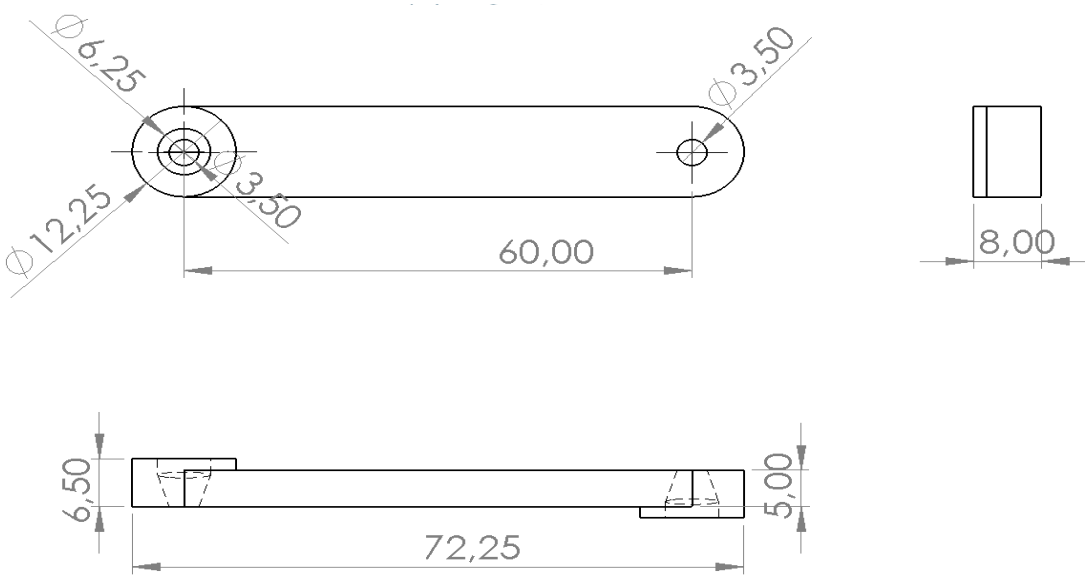


Figure II-4 *Long link*

In long links we are using bearings in order to decrease friction effect.Our bearings are shrink fit to long link.Also bearing is shrink fit to steel bar.We draw conic shape to squeeze bearing.(see figure II-15 for exploded view)

This link contains two bearings which are shrink fit to steel bar.

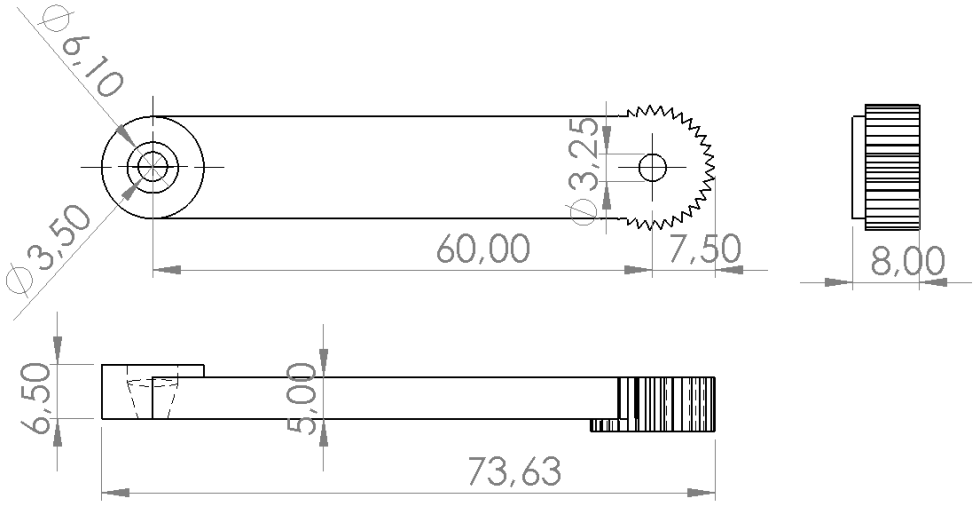


Figure II-5 *Second link*

Our second long link is contacting ground, in order to avoid slipping we used intended shape.This link contains one bearing at left side which is shrink fit to steel bar and one steel bar at right side.

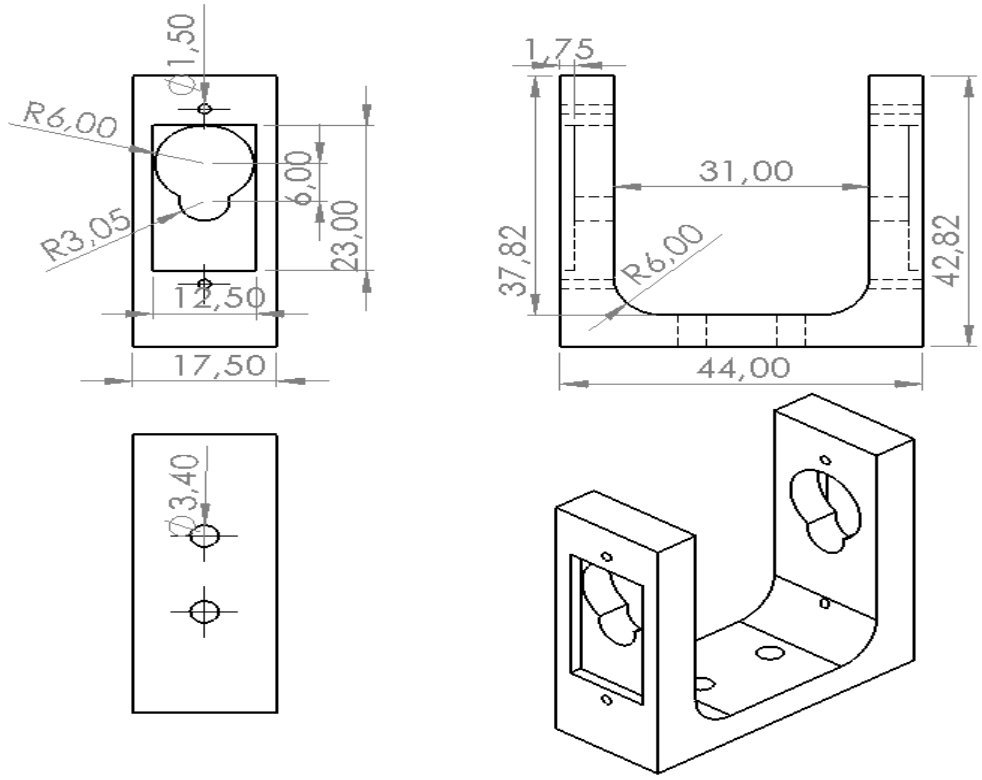


Figure II-6 *Servo holder*

In servo holder design we draw 3.40mm holes to connect our chasis.We used 1.75mm depth pockets to shrink fit servo motors.

Also we draw 1.50mm holes to screw motors to servo holder easily.We give 6mm radius to reduce shear stress at corners.

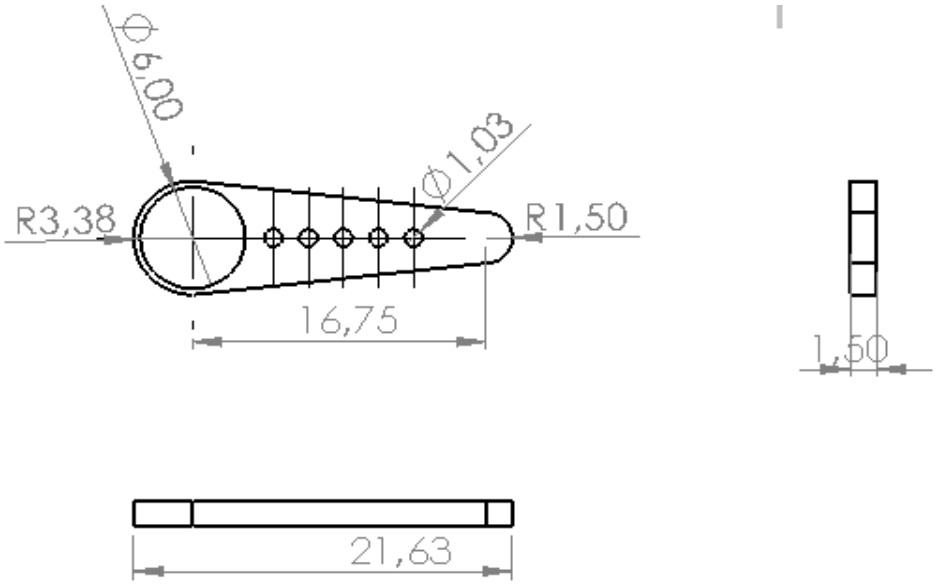


Figure II-7 *Servo arm*

We used this component for connecting motor and small link.

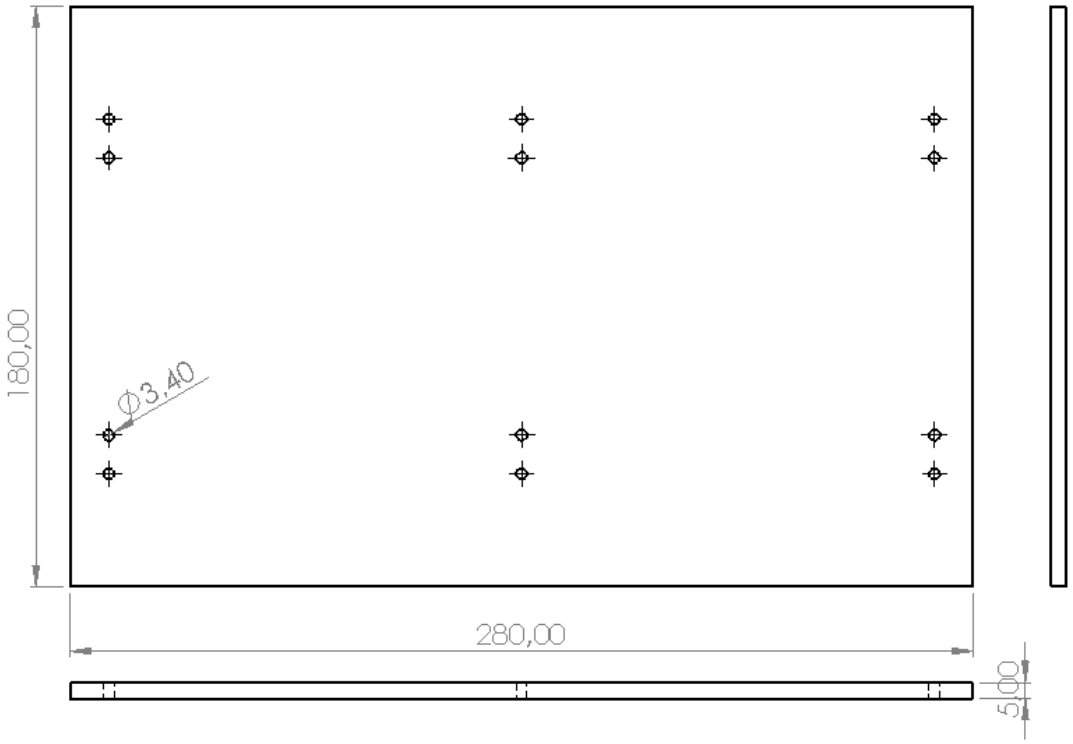


Figure II-8 *Chasis*

Chasis is designed with holes to screw servo holders to chasis.

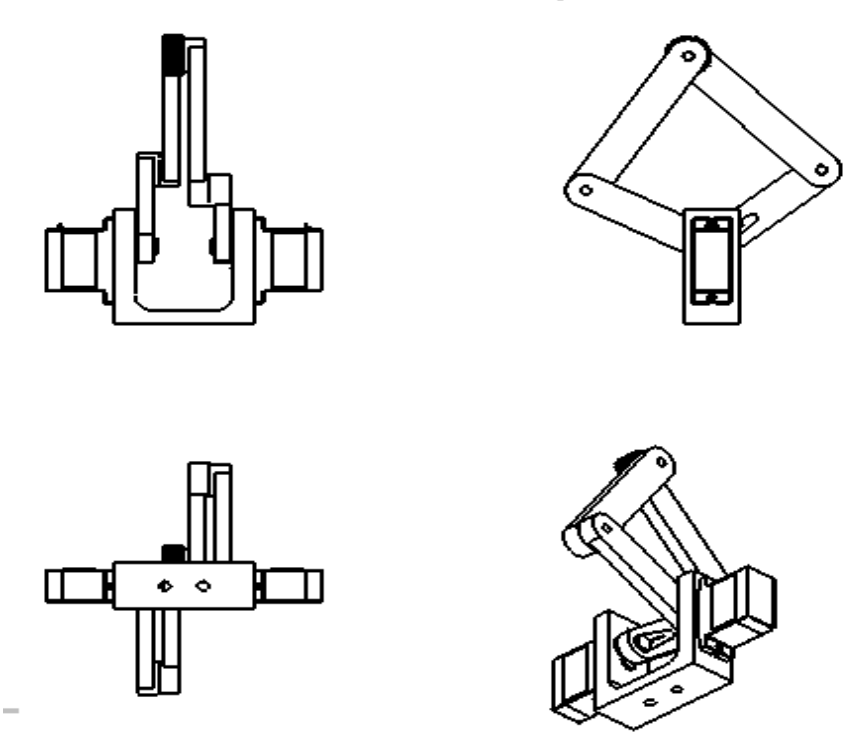


Figure II-9 *Assembled leg mechanism*

Assembled leg mechanism is seen in Figure II-9 with four views.

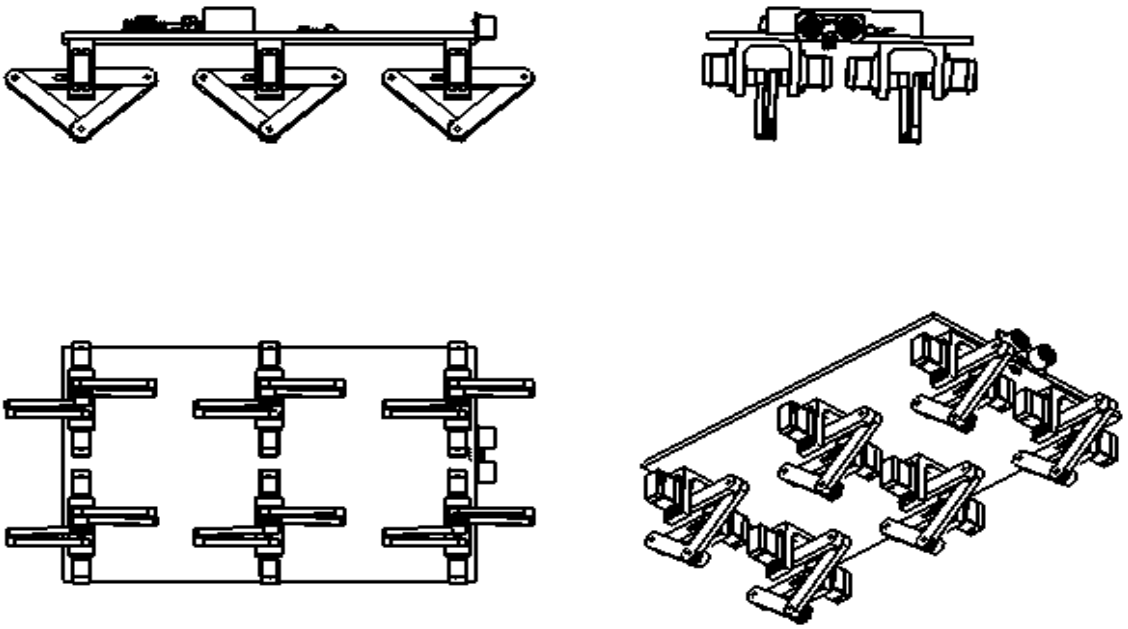


Figure II-10 *Assembled walker*

Assembled walker is shown in Figure II-10 with four views.

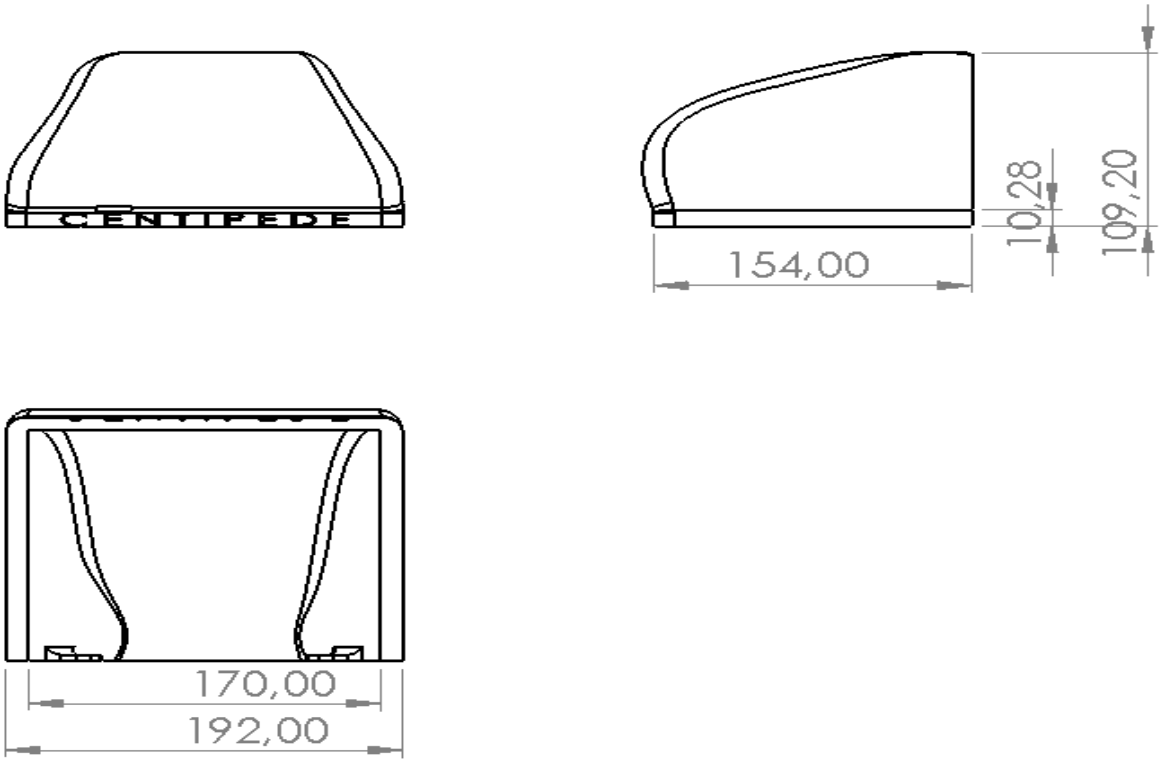


Figure II-11 *Coveringone*

First component of covering is coveringone.

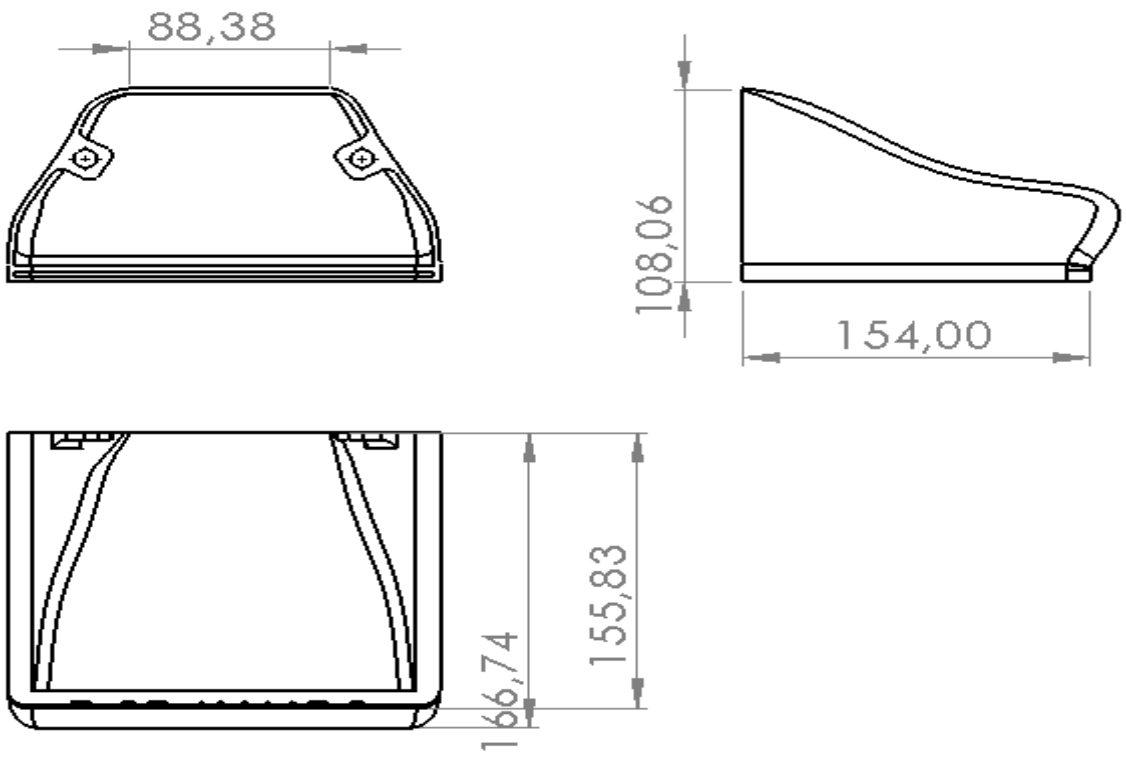


Figure II-12 *Coveringtwo*

Second component of covering is covering two.

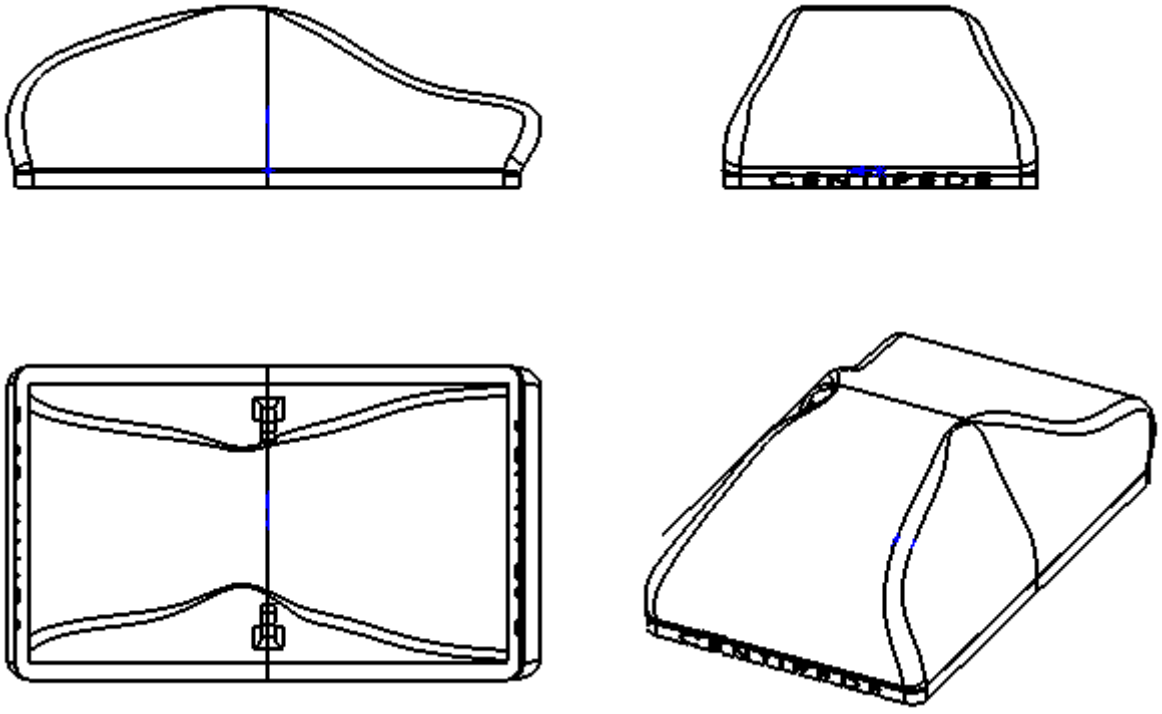


Figure II-13 *Covering*

Assebled covering is seen in Figure II-13.

2.2.2 3D views of leg and walker

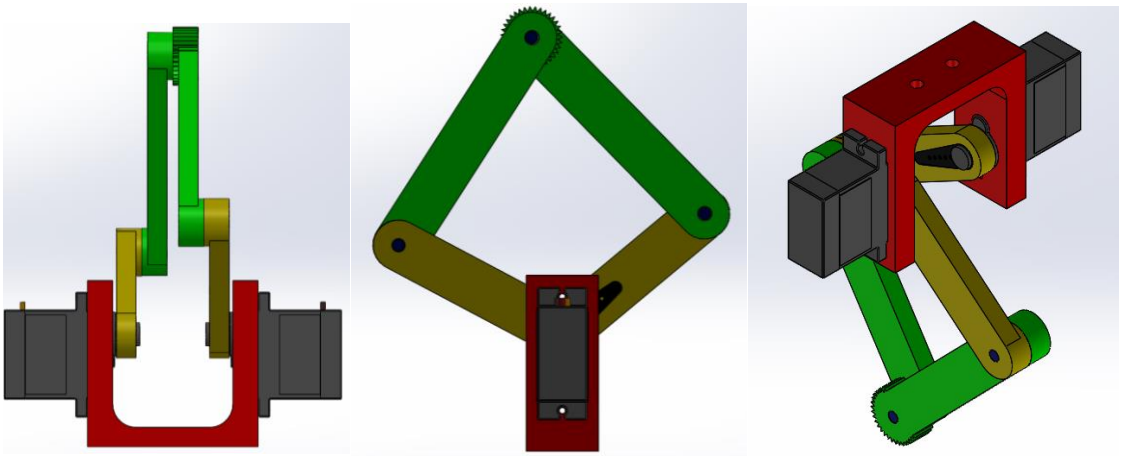


Figure II-14 *3D views of leg mechanism*

Leg mechanism is shown with colors. Same colored links have same lenghts.

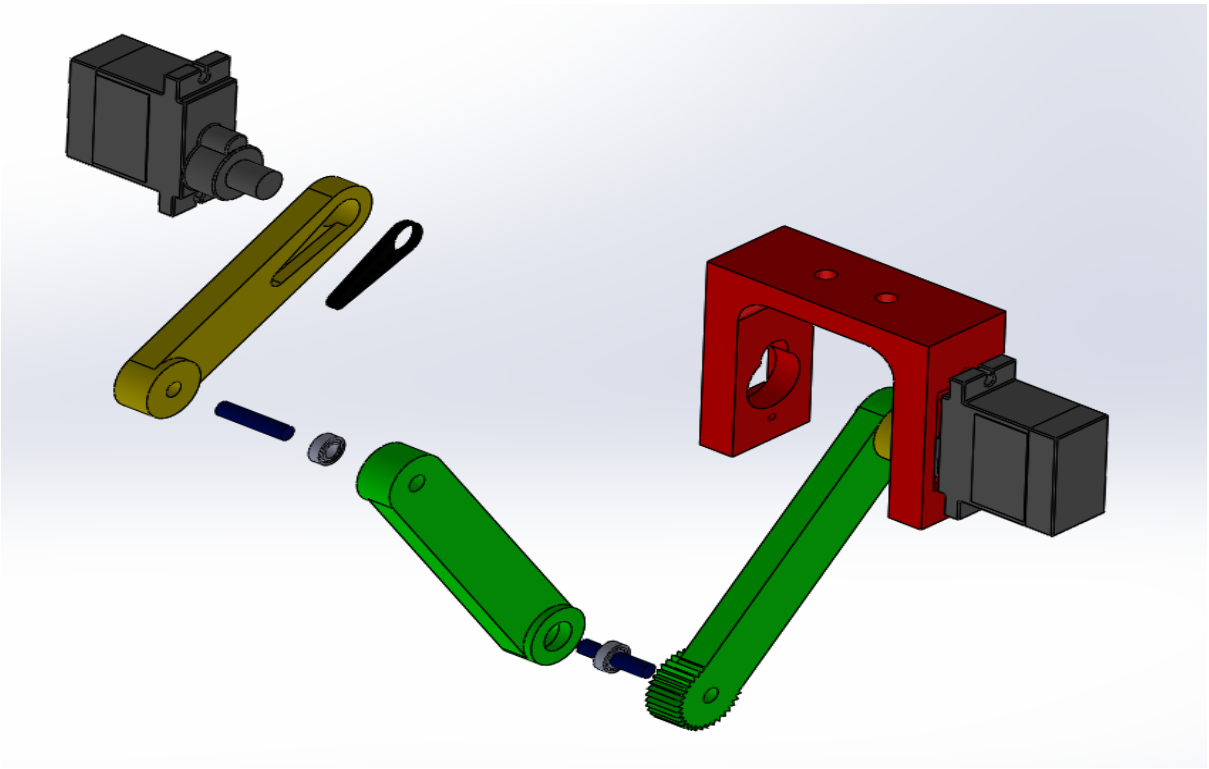


Figure II-15 *Exploded view of leg mechanism*

Exploded view of leg mechanism is obtained with Solidworks.

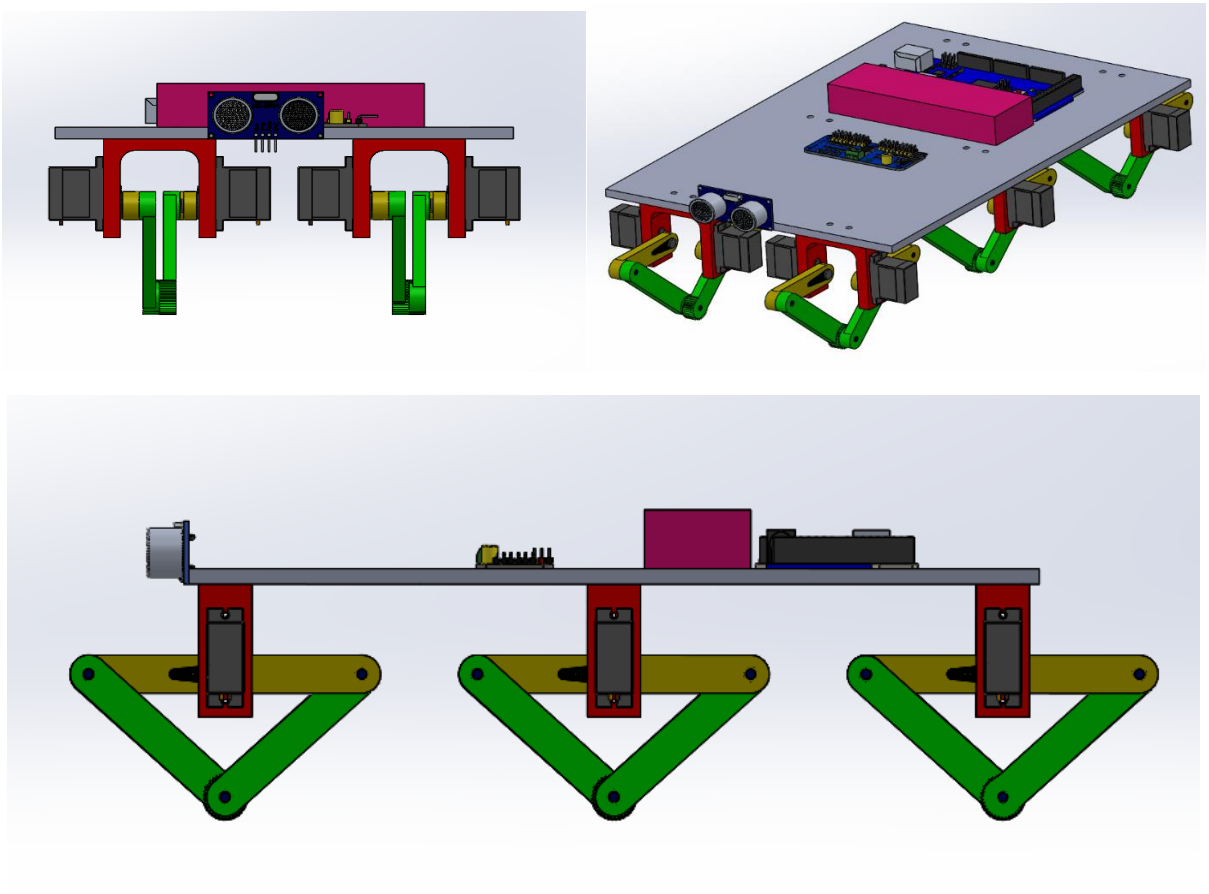


Figure II-16 *3D views of walker*

Finally, walker is shown with colors.

2.2.3 Work space

2.2.3.1 Vertical work space

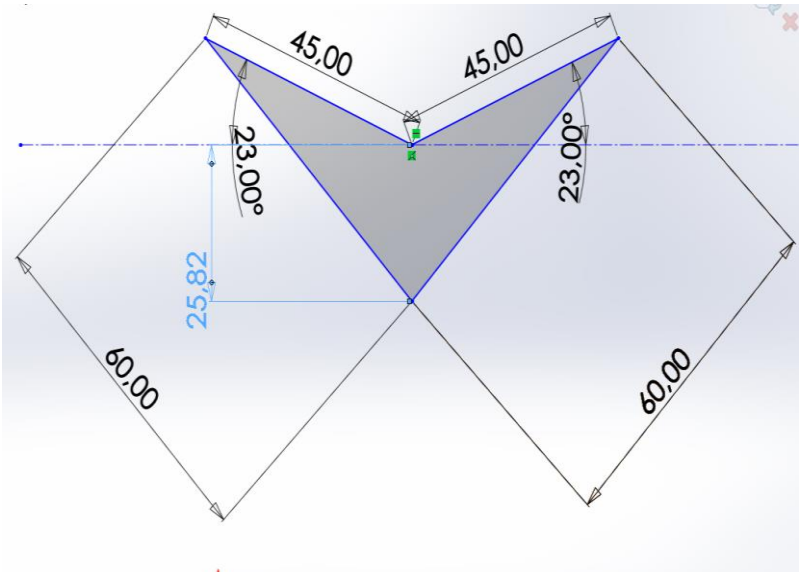


Figure II-17 *Robot's lowest leg height*

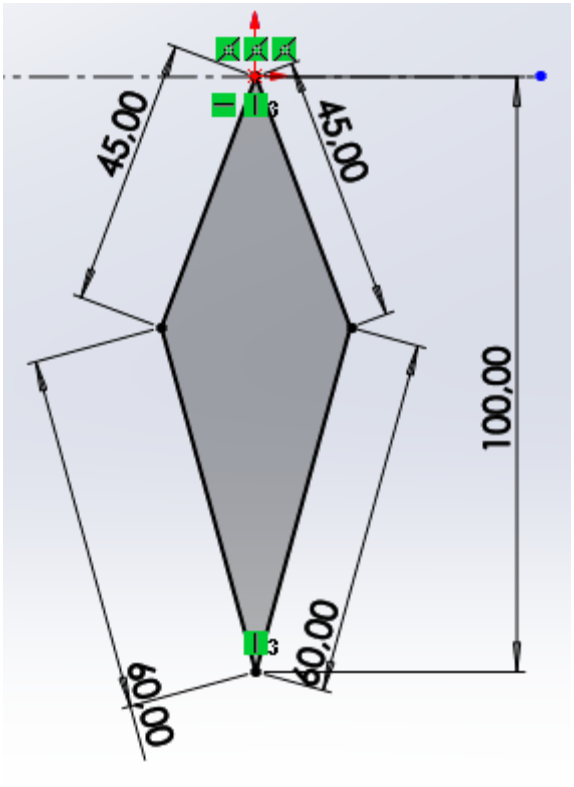


Figure II-18 *Robot's highest leg height*

2.2.3.2 Trajectory work space

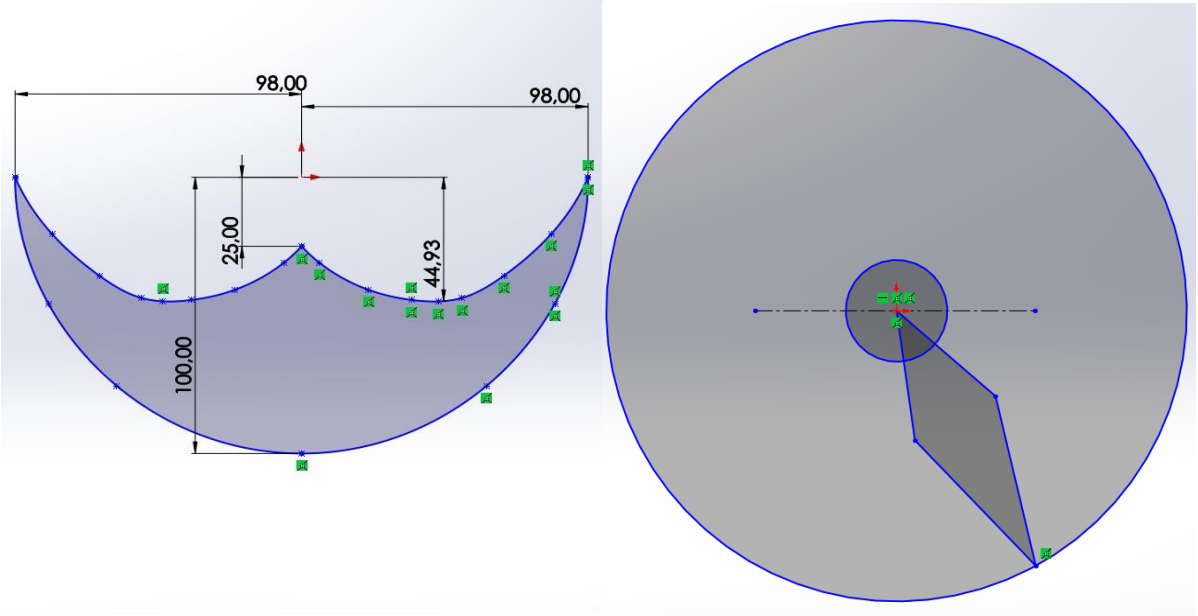


Figure II-19 *(a)Trajectory work space, (b)Fivebar workspace*

2.2.4 Assembly of prototype

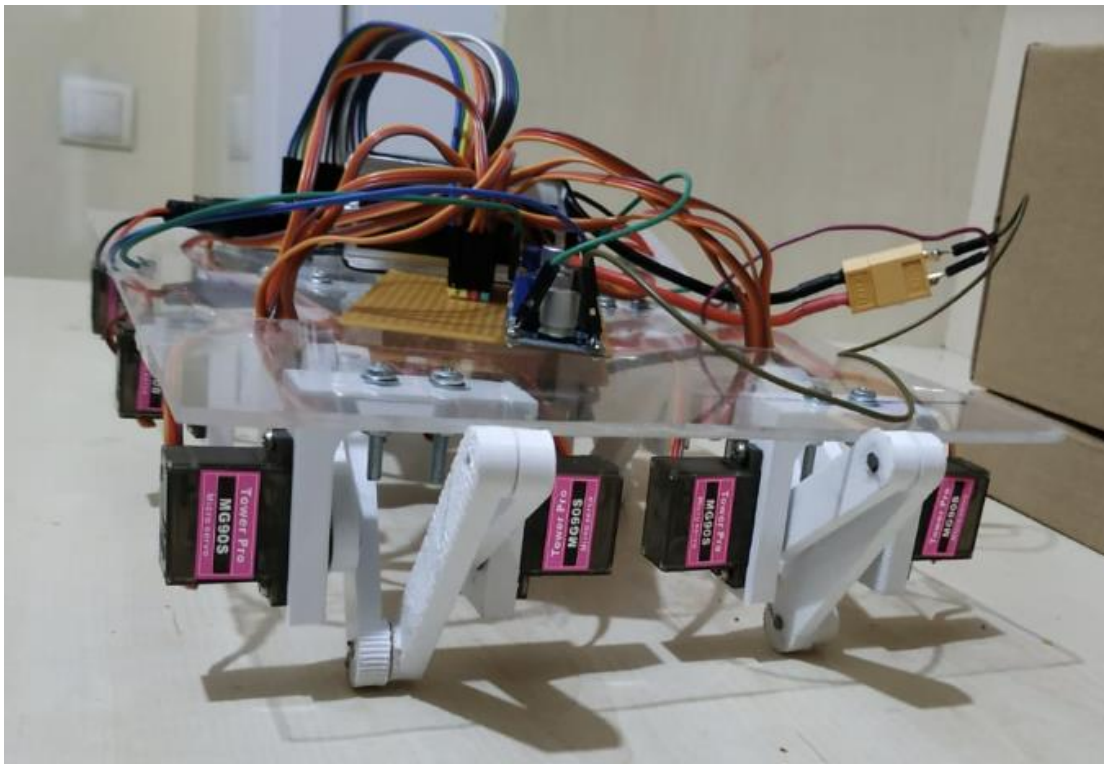


Figure II-20 *Mechanical prototype of robot from front*

Physical assembly of centipede is shown from front without covering.

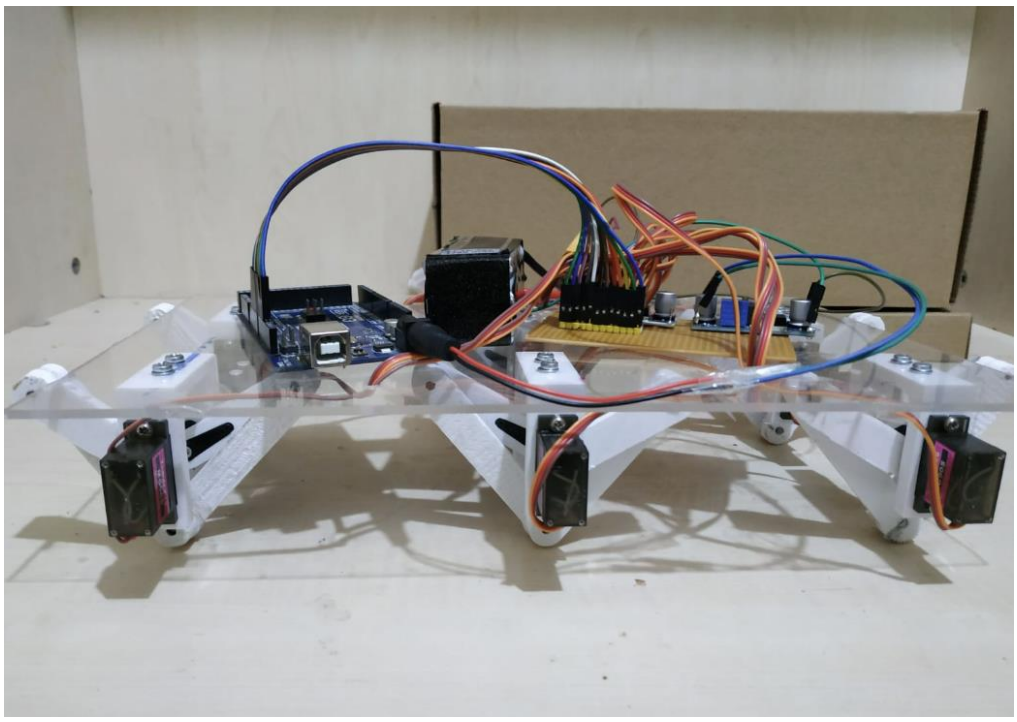


Figure II-21 *Mechanical prototype of robot from side*

Physical assembly of centipede is shown from side without covering.

Figure II-22 *Walker with covering from side*

Final Picture of centipede from side with covering .

Figure II-23 *Walker with covering from front*

Final Picture of centipede from front with covering.

2.2.5 Numbering servo motors

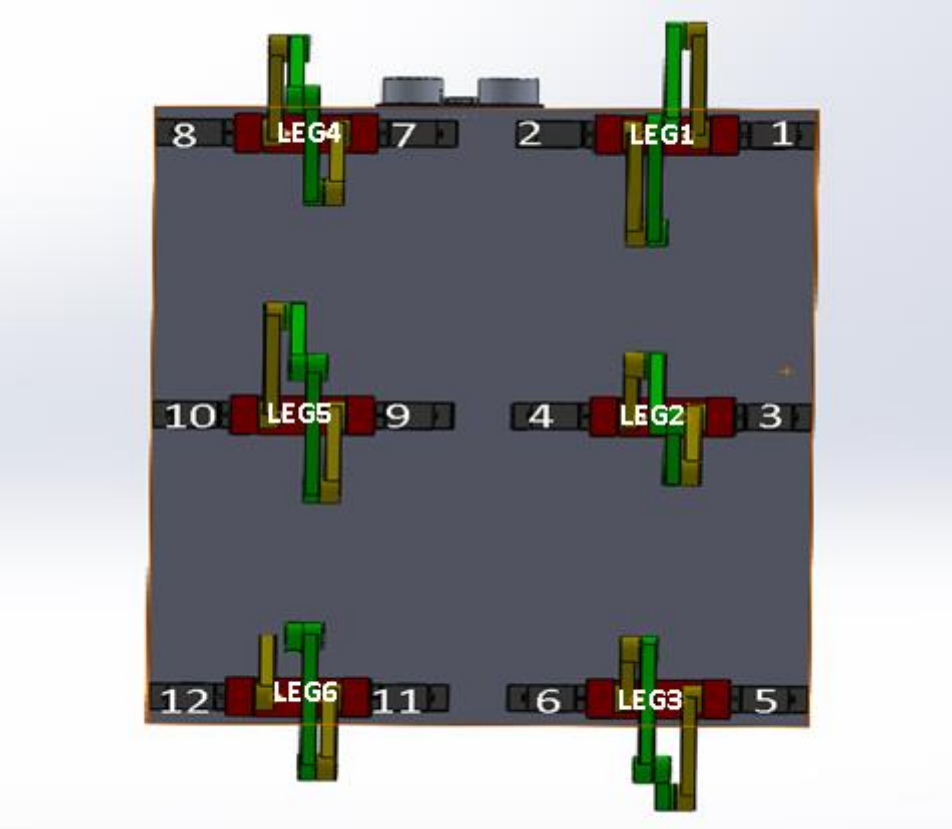


Figure II-24 *Numbers of servo motors*

2.3 Simulation results

Simulation is done by using square trajectory. Torque and energy analysis suggested us required motor and battery.

2.3.1 Unloaded simulation results

2.3.1.1 Unloaded torque analysis

Unloaded torque analysis is done without microcontroller and batteries. Our aim is to compare unloaded torques and loaded torques.

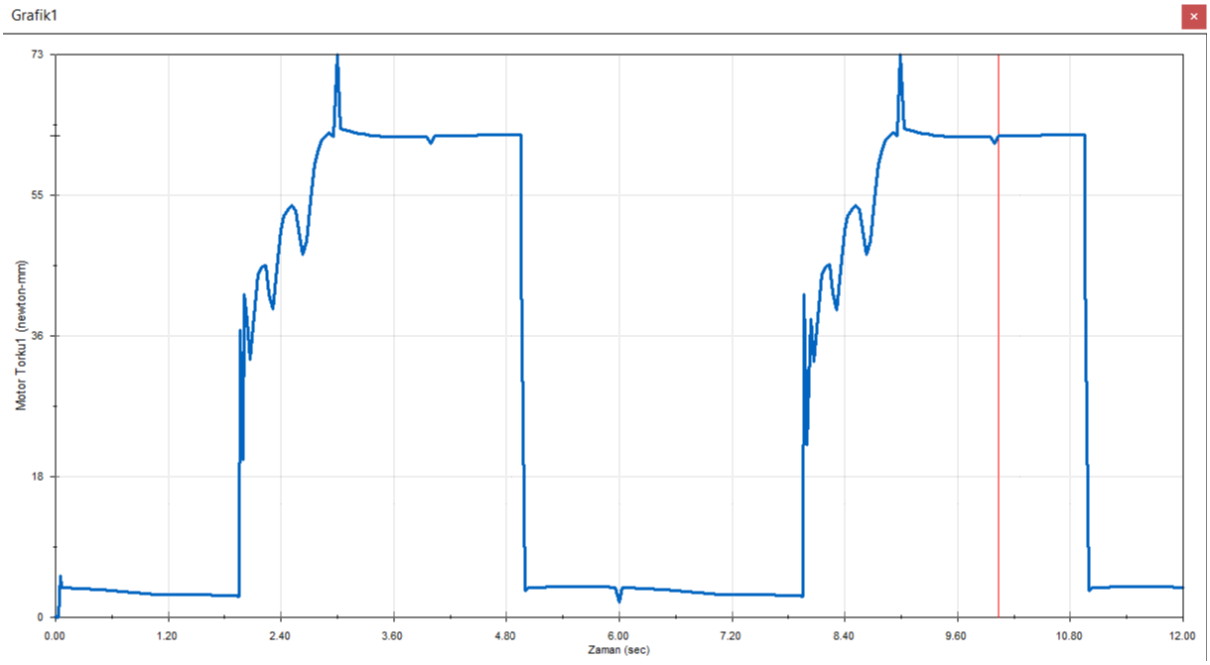


Figure II-25 *Torque analysis of motor1*

We numbered servo motors to comment and compare easily(see figure Figure II-24).In figure Figure II-25 you can see motor torque change according to time. At time 1.8s torque increases instantly, at this time leg is contacting ground and causes an instant torque increase.At time 2.8s torque reaches maximum value at point 4(see figure IV-2).When other 3 leg contacts ground at time 5s torque instantly reduces and leg leaves ground at this point.

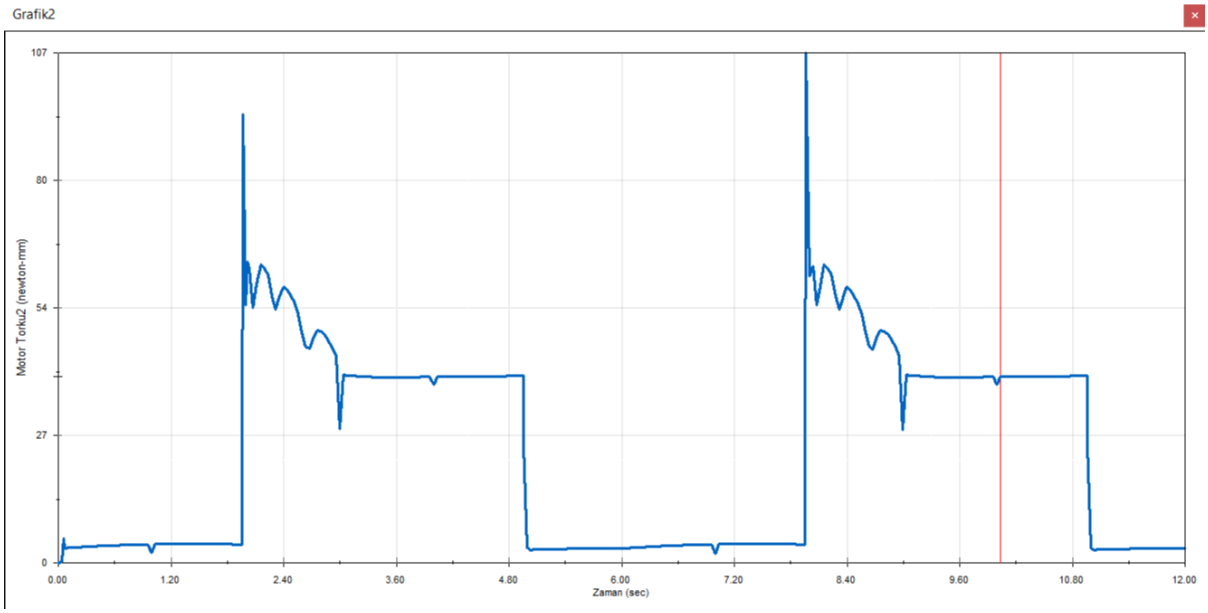


Figure II-26 *Torque analysis of motor2*

In figure II-26 torque analysis of motor2 is seen.Motor1 and motor2 are in the same leg. At time 1.8s leg contacting ground and causes instant torque increase.However,at exact contact time motor2 carries more torque than motor1.During contact period, when motor1's torque is

increasing,motor2's torque is decreasing.However,total torque is same along conctact period.At time 5s leg leaves ground and motor2's torque is decreased instantly as motor1.

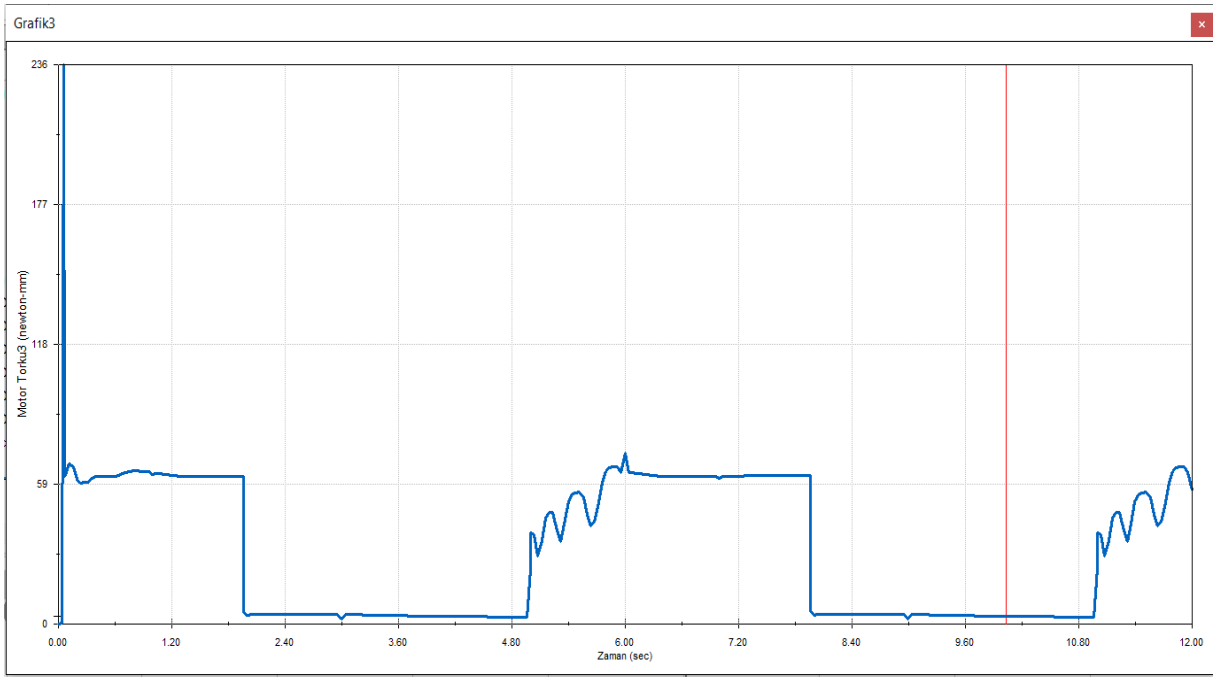


Figure II-27 *Torque analysis of motor3*

Motor3 is in leg2 which is contacting ground after leg1(see figure Figure II-24). So, we expect same graph with time delayed version. At time 5s leg2 contacting ground and causes instant torque increase. At time 8s,leg2 leaves ground and leg1 is contacting.

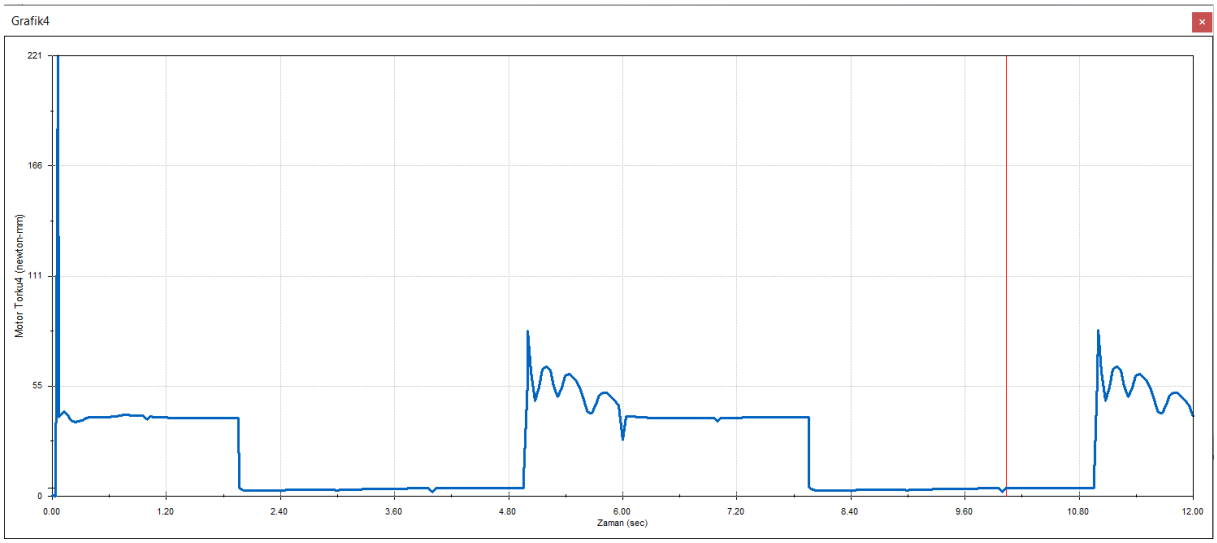


Figure II-28 *Torque analysis of motor4*

We expected total torque is to be conserved between motor3 and motor4 along contact period. We verified this comment with Figure II-28.

2.3.1.2 Unloaded energy analysis

Unloaded energy analysis is done without microcontroller and batteries. Our aim is to compare unloaded energies and loaded energies.

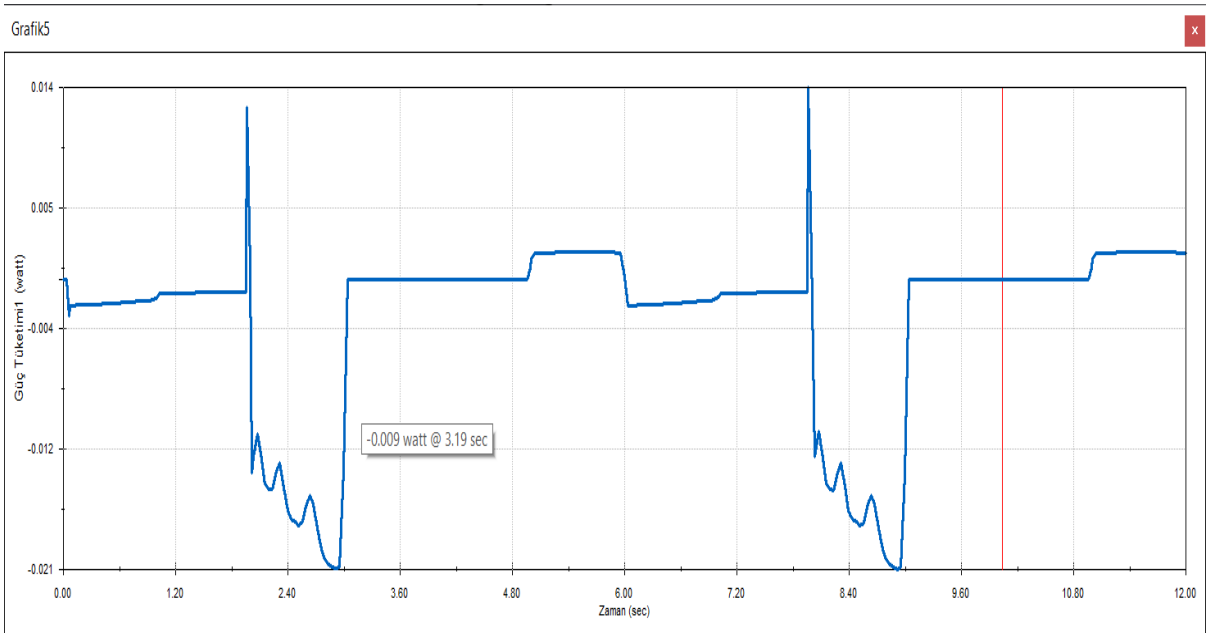


Figure II-29 *Energy consumption graph of motor1*

In figure II-29 you can see energy consumption of motor1 according to time. We are expecting energy graphs to be similar to the torque graphs. In figure II-29 we can see instant increase in energy consumption, this time is contacting time. After time 2.8 energy consumption is decreased instantly because leg doesn't move. In non contacting period energy consumption is very low because motors on air and they don't encounter any firiction.

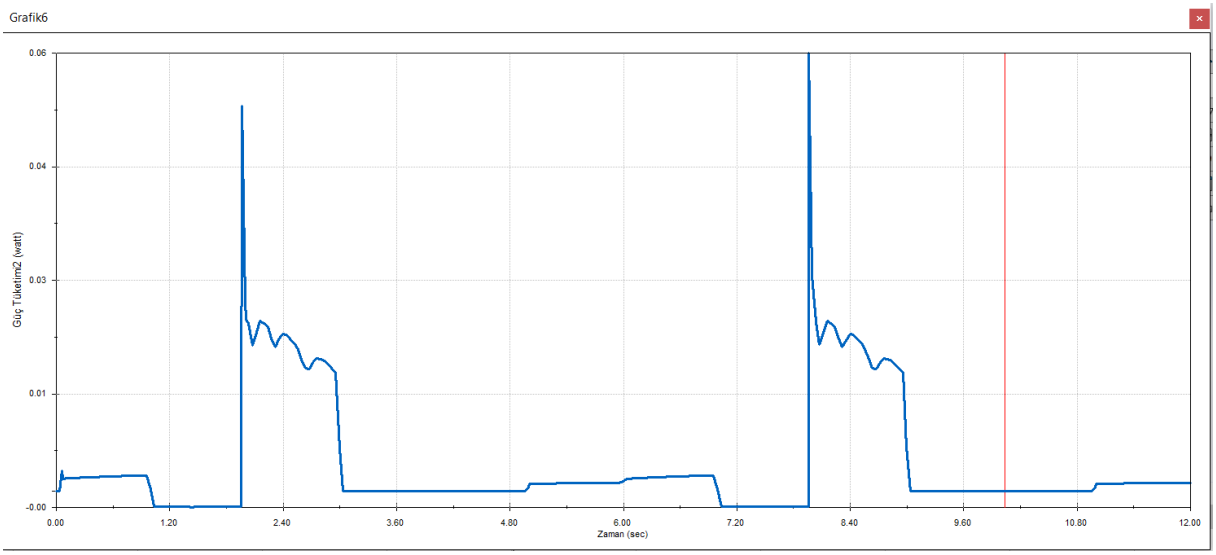


Figure II-30 *Energy consumption graph of motor2*

Total energy consumption should be same along contacting period. We verified this comment with comparing figure II-29 and figure II-30.

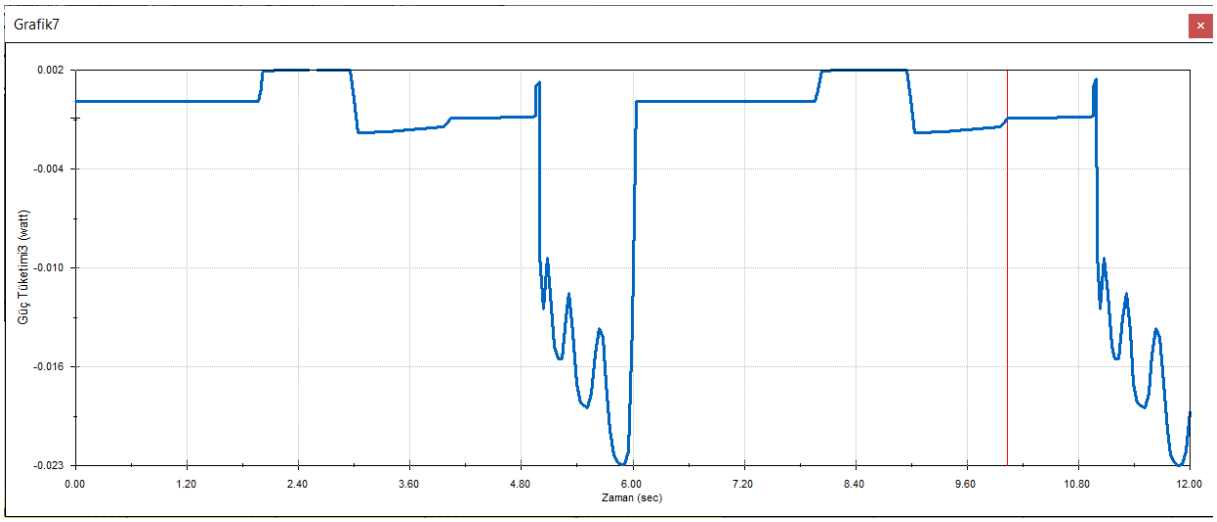


Figure II-31 *Energy consumption graph of motor3*

Because of leg1 and leg2 contacting after each other. We obtained time delayed version of motor1's graph in figure II-31.

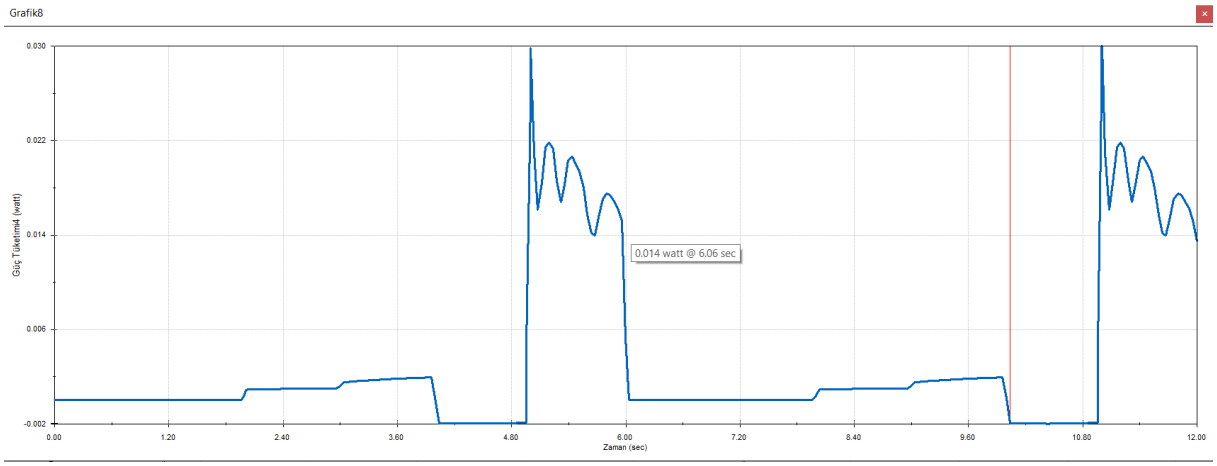


Figure II-32 *Energy consumption graph of motor4*

Because of leg1 and leg2 contacting after each other. We obtained time delayed version of motor2's graph in figure II-32.

2.3.2 Loaded simulation results

2.3.2.1 Loaded torque analysis

For loaded torque analysis we take into account masses on chasis. Total mass is 0.85kg of all components.

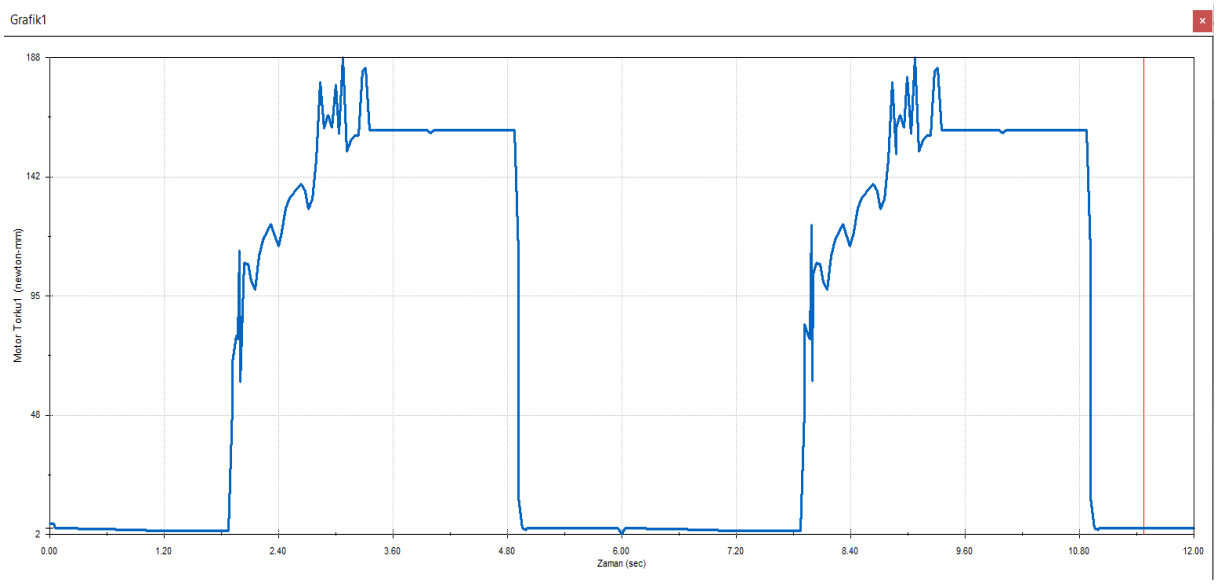


Figure II-33 *Torque analysis of motor1*

If we compare loaded and unloaded torque analysis of motor1, we can see shape of graph is very similar. However, in loaded case there are more oscillations on contacting period, also maximum torque value is increased to 188Nmm, whereas maximum unloaded torque value of motor1 was 73Nmm.

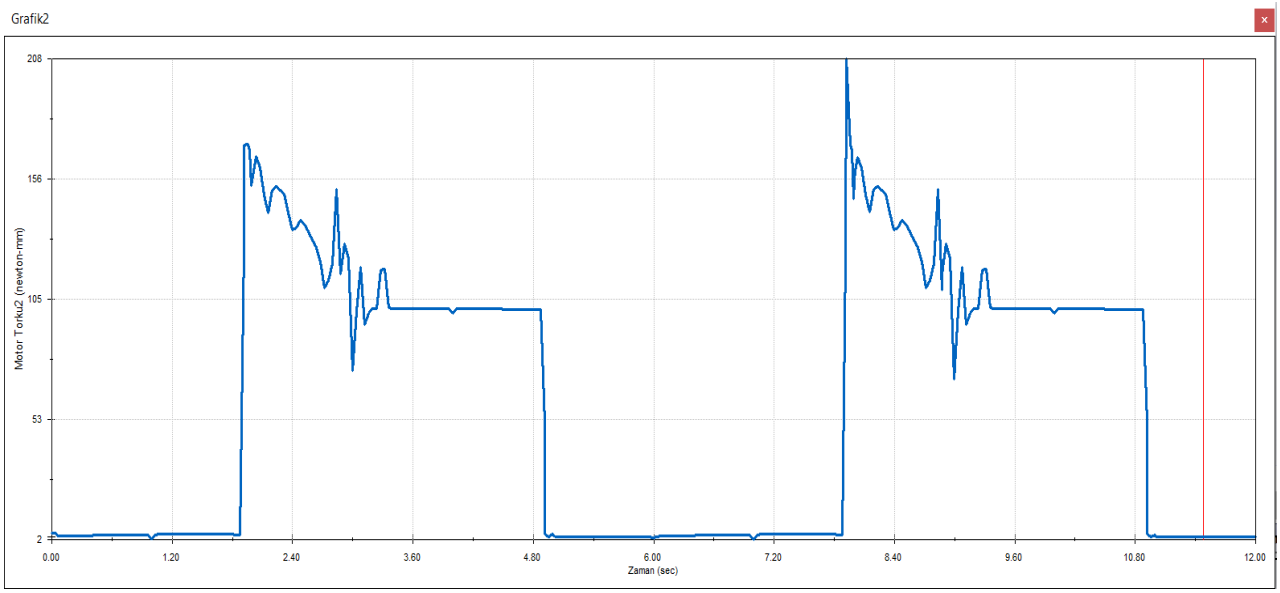


Figure II-34 *Torque analysis of motor2*

We expect total torque value is to be conserved on contacting period, when we compared motor1's graph and motor2's graph. We verified this comment with figure II-34.

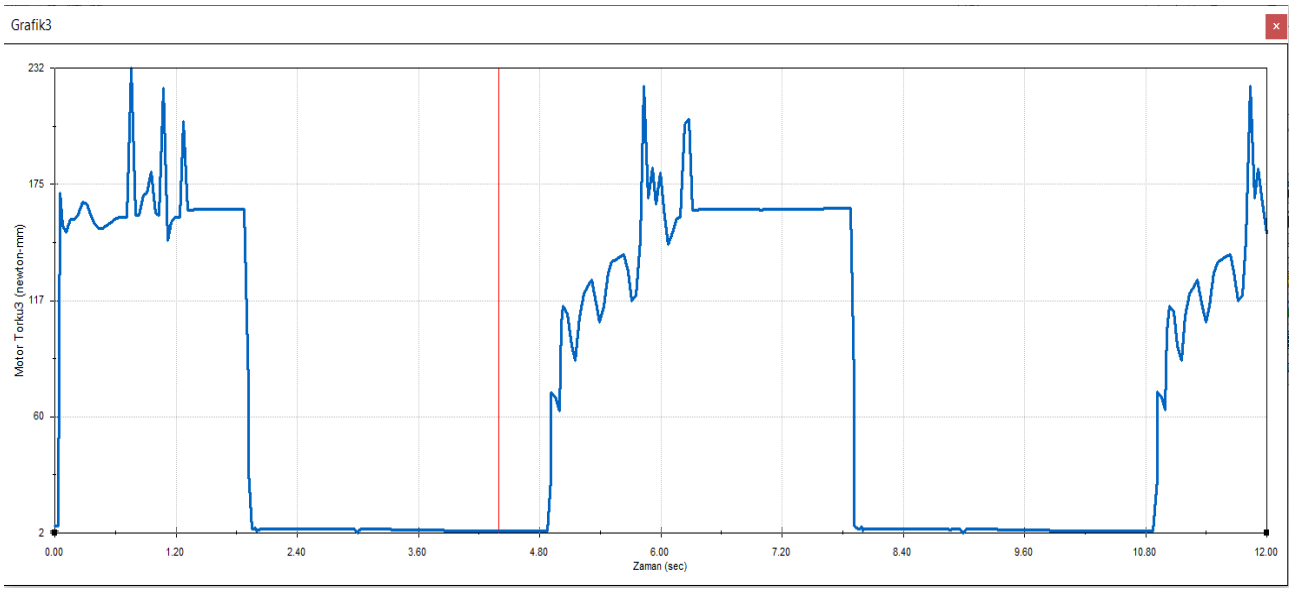


Figure II-35 *Torque analysis of motor3*

We obtained time delayed version of motor1's graph.

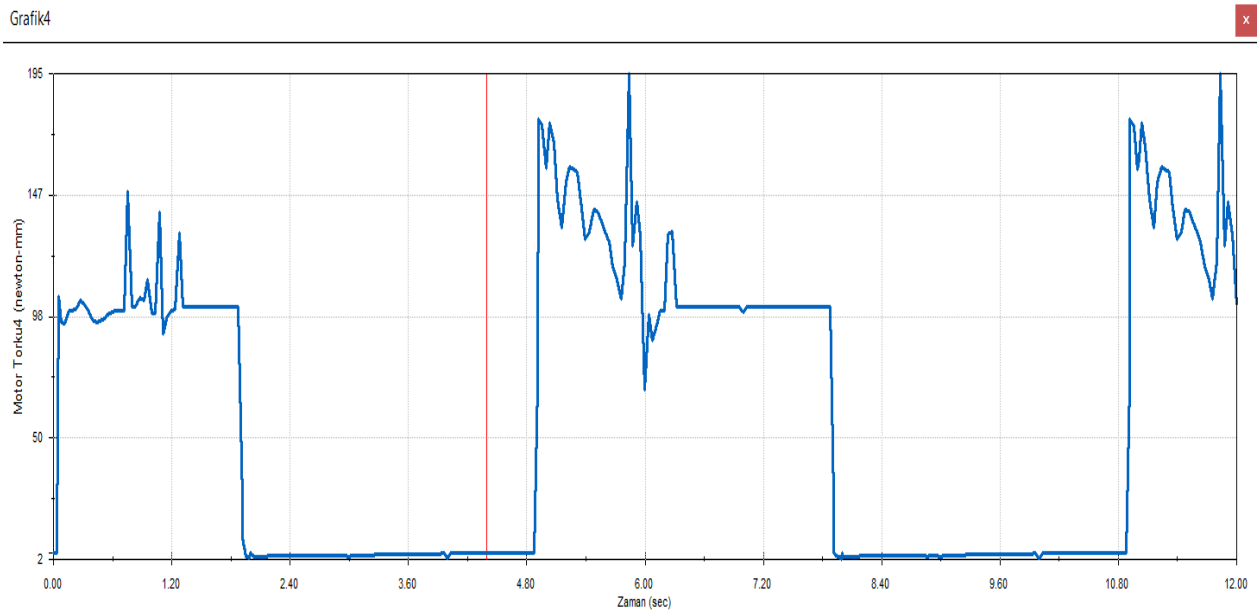


Figure II-36 *Torque analysis of motor4*

We obtained time delayed version of motor2's graph.

2.3.2.2 Loaded energy analysis

For loaded energy analysis we take into account masses on chasis. Total mass is 0.85kg of all components.

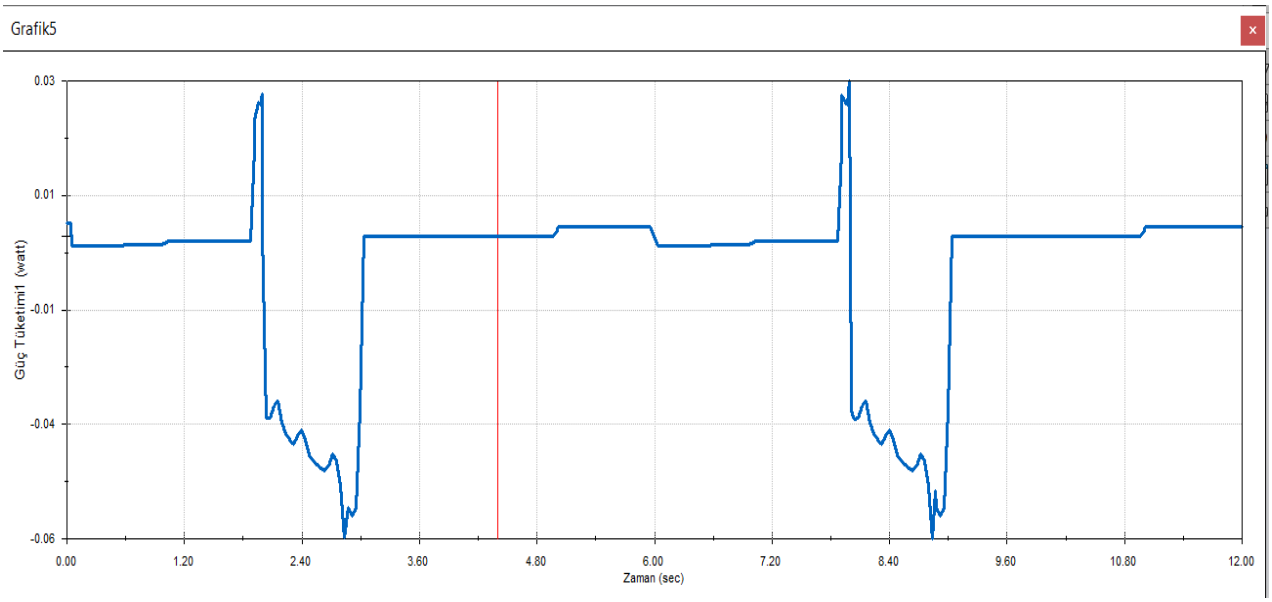


Figure II-37 *Energy analysis of motor1*

If we compare loaded and unloaded energy analysis of motor1, we can see shape of graph is very similar. However, in loaded case maximum energy value is increased to -0.06W, whereas maximum unloaded energy value of motor1 was -0.021W.

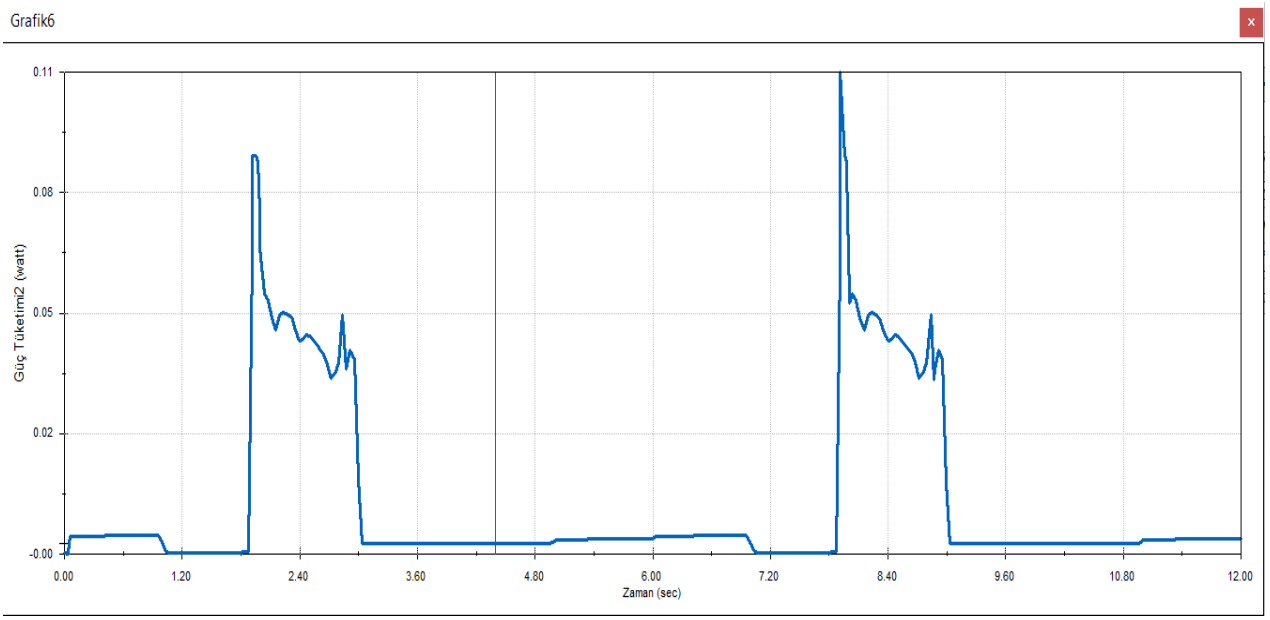


Figure II-38 *Enegy analysis of motor2*

Total energy on contacting period is conserved. When leg1 stepping, motor1 is drawing more current than motor2 as time value increases. However, at exact contact time motor2 is drawing more current.

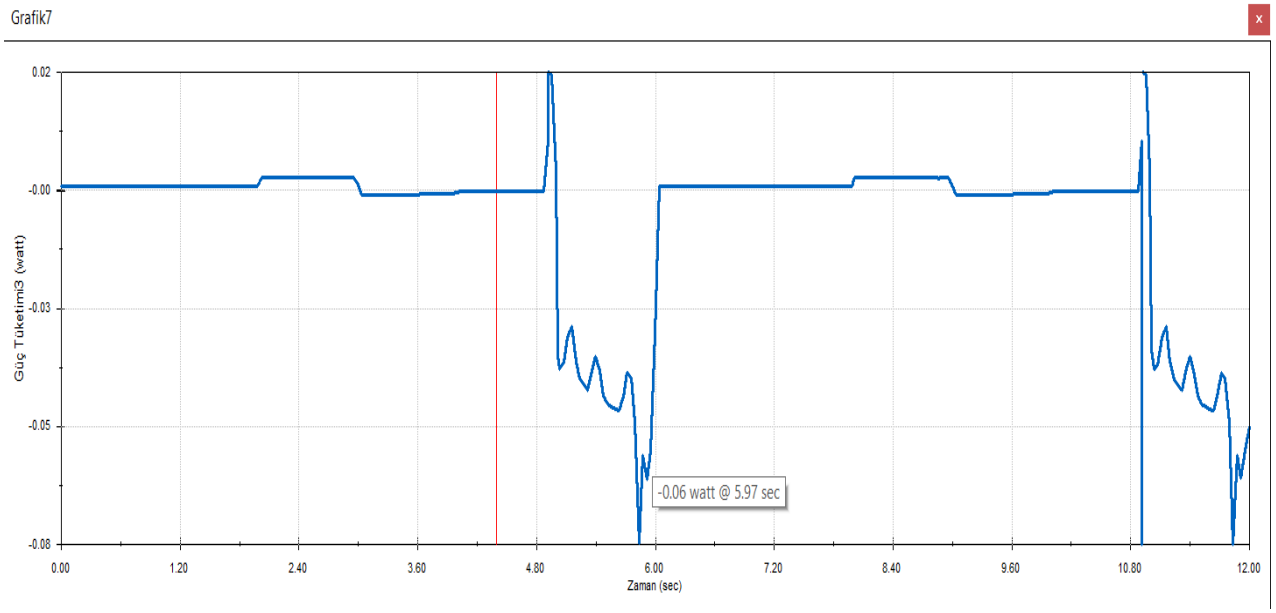


Figure II-39 *Enegy analysis of motor3*

We obtained time delayed version of motor1's graph.

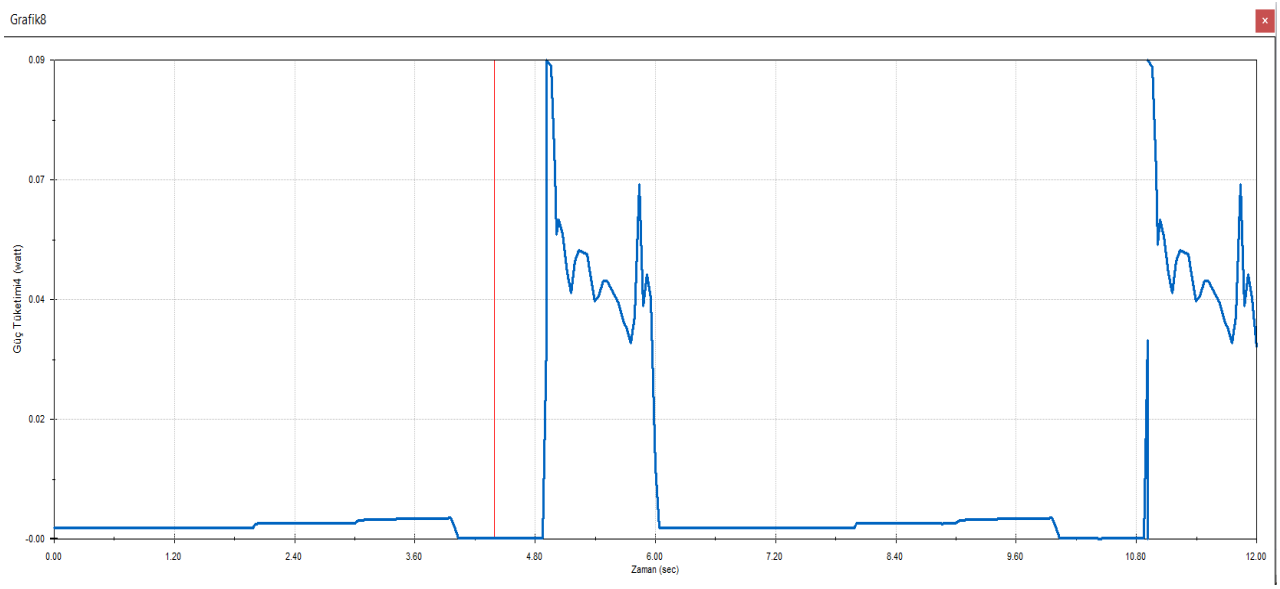


Figure II-40 *Enegy analysis of motor4*

We obtained time delayed version of motor2's graph.

If we make general conclusion ;

- Motors of leg1,leg5 and leg3 have same results, because they are contacting ground at the same time.

- Motors of leg2,leg4 and leg6 have same results, because they are also contacting ground at the same time.
- Motor1,motor5 and motor10 have same results, because they all outer motors and their legs are contacting ground at the same time.
- Motor2,motor6 and motor9 have same results, because they all inner motors and their legs are contacting ground at the same time.
- Motor3,motor8 and motor12 have same results with time delayed version of motor1, because they all outer motors and their legs are contacting ground at the same time.
- Motor4,motor7 and motor11 have same results with time delayed version of motor2, because they all inner motors and their legs are contacting ground at the same time.

III. CIRCUIT COMPONENTS AND CIRCUITRY

3.1 Circuit components

Battery: We used two li-po battery to supply our circuit. We used 11.1V li-po battery to supply servo motors .Another 7.4V li-po battery is used for supplying microcontroller.

Figure III-1 *Li-po battery*

Voltage regulator module: We used voltage regulator to reduce voltage value to 6V, so we can *supply our servo motors.*

Figure III-2 *Voltage regulator module*

Bluetooth module: Bluetooth module is used for remote control of walker.

Figure III-3 *Bluetooth module*

Microcontroller: Microcontroller is used to control motors. It includes software and solves inverse kinematic problem.

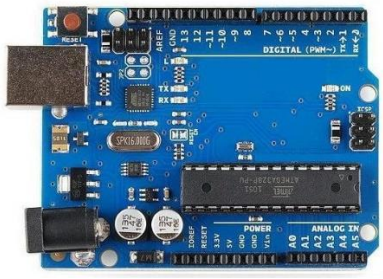


Figure III-4 *Microcontroller*

Servo motors: We used mg90s servo motors because they are light and has workspace 180 degrees.

Figure III-5 *Servo motor*

3.2 Circuitry

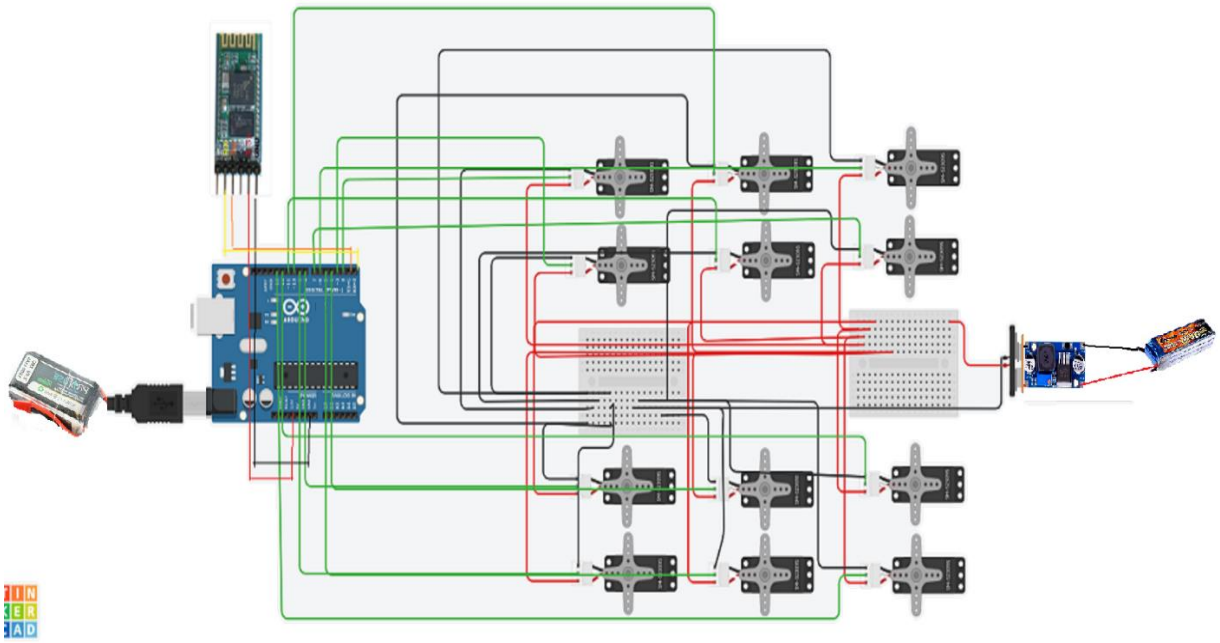


Figure III-6 *Circuitry*

We have twelve servo motors, we connected their pwm's to microcontroller, and motors connected to outputs of voltage regulator. We connected bluetooth module to 5V, ground, rx and tx on microcontroller.

IV. THREE TRAJECTORIES

Three trajectoris are used in this project.They are square,triangle and smooth trajectories.

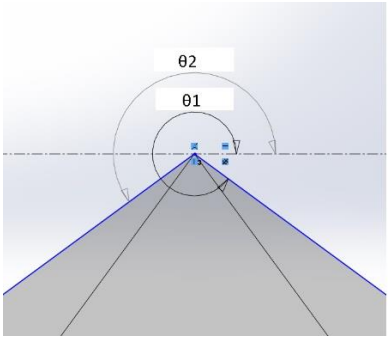


Figure IV-1 *Angles of motors and legs*

4.1 Square trajectory

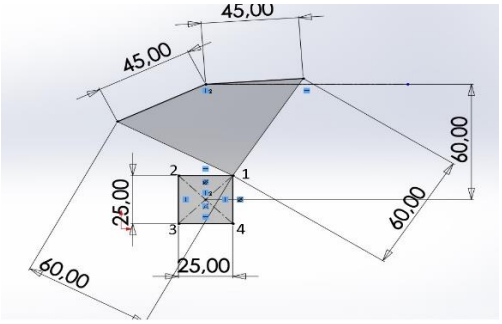


Figure IV-2 *Square trajectory*

Position	θ1	θ2
0	0°	180°
1	3,67°	205,71°
2	334,34°	176,18°
3	314,8°	205,64°
4	334,36	225,2°

Table IV-1 *Square trajectory position table*

In Figure III-2 we defined corner points of square trajectory.For each position we write actuator values.By using these tables we verified inverse kinematic formulas(see chapter IV).And then inside code, we introduced interpolation formula, so our motors move point to point without leeping.

4.2 Triangular trajectory

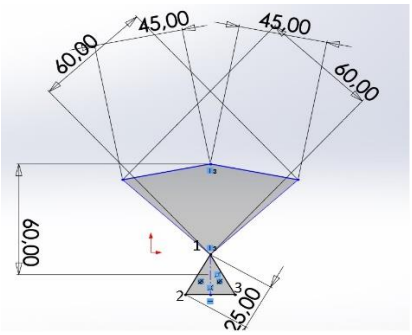


Figure IV-3 T*riangular trajectory*

Position	θ1	θ2
0	0°	180°
1	7,02°	187,02°
2	319,22°	199,71°
3	340,29°	220,78°

Table IV-2 T*riangular trajectory position table*

We defined three positions for triangular trajectory and then measured θ1 and θ2 angles for each positon.

4.3 Smooth trajectory

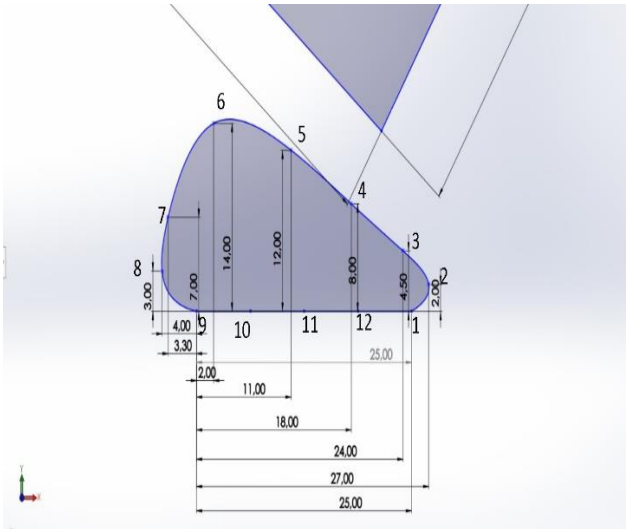


Figure IV-4 *Smooth trajectory*

Position	θ1	θ2
0	0°	180°
1	337,17°	223,08°
2	340,65°	223,43°
3	341,55°	218,37°
4	340,84°	209,3°
5	338,46°	198,58°
6	330,36°	188,40°
7	319,03°	192,81°
8	315,30°	197,03°
9	316,92°	202,83°
10	322,23°	207,03°
11	328,14°	211,86°
12	332,97°	217,23°

Table IV-3 *Smooth trajectory position table*

Smooth trajectory consists of twelve points.And this is most complex trajectory.

4.4 Comparison of energy consumption of three trajectories

Energy efficiency test is done in same surface.For each trajectory one step is arranged as 25 mm and walker operated fifteen minutes.Initial and final voltages are noted in table IV-4 **.**

Trajectory	Initial Voltage	Final voltage
Square	12.54V	12.03V
Triangular	12.54V	12.15V
Smooth	12.54V	12.17V

Table IV-4 *Energy consumption table*

As you can see in table IV-4, most efficient trajectory is smooth trajectory.As we mentioned in introduction chapter, well defined trajectories reduces instant torque and force distribution.So,energy consumption decreases.In this experiment, even in such short period as fifteen minutes ,we can see there is a significant difference between energy consumption.For long period applications, using well defined trajectories are very important.

V. INVERSE KINEMATIC ANALYSIS

This inverse method is new method.Loop closure is one possible way to solve inverse kinematics.However, this method is easier than loop closure method.

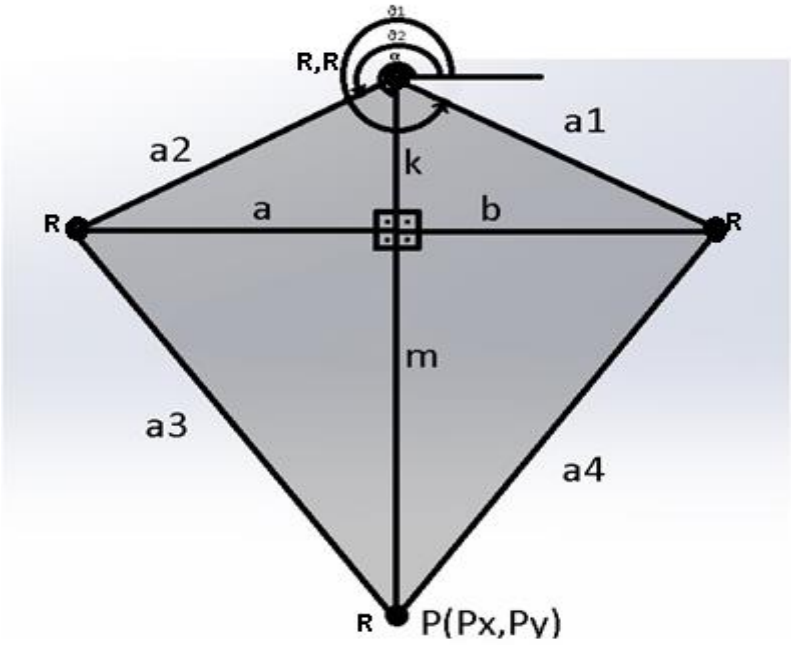


Figure V-1 *Shematic diagram of one leg chain*

a1=a2=45mm **a3=a4=60mm** **a=b** **s=k+m…..(1)**

s=$\sqrt{Px^2 + Py^2}$..(2) **$a^2+k^2=a2^2$…(3)** **$k^2+b^2=a1^2$….(4)** **$a^2+m^2=a3^2$…(5)**

$m^2+b^2=a4^2$…(6)

By substracting equation 4 and 5:

$k^2-m^2=a1^2-a3^2$ **(k-m)(k+m)= $a1^2-a3^2$**

From 1:

k-m=($a1^2-a3^2$)/s

k+m=s

2k=($a1^2-a3^2+s^2$)/s **k=($a1^2-a3^2+s^2$)/2s** **m=($s^2-a1^2+a3^2$)/2s…(7)**

By substituting k into equation 3:

a=$\sqrt[2]{a_2^2 - ((a_1^2 - a_3^2 + s^2)/2s)^2}$

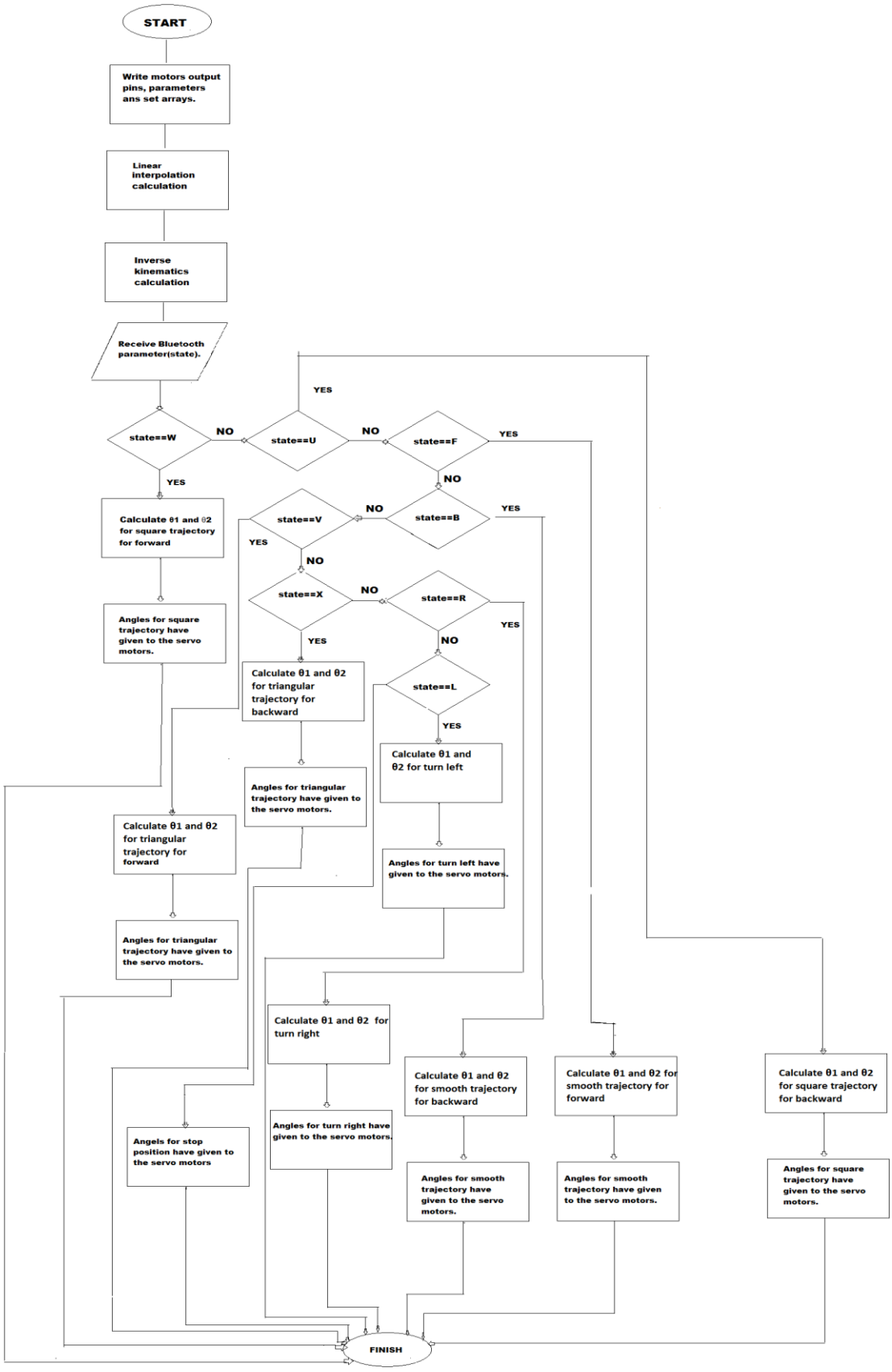
By substituting m into equation 6:

b=$\sqrt[2]{a_4^2 - ((s^2 - a_1^2 - a_3^2+)/2s)^2}$

α=arctan(Py/Px) **θ1= α+arcsin(b/a1)** **θ2= α-arcsin(a/a2)**

VI. SOFTWARE

6.1 Flow chart



6.2 Software

Software of microcontroller is explained in Appendix A.

VII. CONCLUSION AND FUTURE WORK

7.1 Conclusion

In this project we examined three different trajectories.

We compared three trajectories and we proved that well defined trajectories consumes lesser energy.

We implemented corner positions of trajectories to sofware and used interpolation to obtain smooth walking.

We simulated walker for loaded and unloaded cases.We compared the results and noted differences between required torques and energies.

We used a new inverse kinematic analysis method which is easier than loop closure method and we used these equations inside software.So, we can find actuator angles by giving end coordinate of robot to microcontroller,

7.2 Future work

In future work we may add some sensors to walker.So, it can avoid obstacles by itself.

Now we change trajectory remotely.In future robot itself may change it’s trajectory according to input data from sensors.

We are going to examine a few more trajectories and compare their efficiency.

We are going to add a raspberry pi because of memory issues of our microcontroller.

We may add camera module, so walker can process image,It may track something it saw or avoid obstacles.

SECTION 2

A PIPELINE INSPECTION ROBOT

PIPELINEBOT

I. INTRODUCTION

1.1 What is a pipeline inspection robot and why we need these robots?

Pipelines are used to transport water, oil or gas. Pipelines are generally located underground or seabed, so their maintenance and cleaning procedures are challenging. Currently intelligent pig systems are used to inspect and clean pipelines. However, cost of this process is very high. So, at this point we need pipeline inspection robots. These robots can inspect inside pipeline by using sensors, cameras etc., also some inspection robots can do cleaning and repair. Pipeline inspection robots are generally used in areas where we can not use pig systems.

The advantages of pipeline inspection robots are:

- We can reduce cost,
- We can reduce required time,
- We can reduce required manpower of cleaning and maintenance procedures.

1.2 A literature review about pipeline inspection robots

There are lots of works about pipeline inspection robots , some of them are:

- An autonomous pipeline inspection robot called FAMPER is described in Jong(2010). This robot can work inside a fixed diameter 150mm. **[11]**
- Also, a robot called MRINSPECT V can operate a fixed 8 inch diameter Se (2008). In this robot a differential driving structure is used to adapt robot to different pipeline shapes. This robot had different driving methods for saving energy. **[11]**
- An innovative pipeline inspection gauge is designed for small diameters Firas(2015). The functions were cleaning and inspecting pipelines. PIG could adapt pipelines between diameters 6 inch and 14 inch. Ultrasonic sensors and arduino software is used to measure diameter of pipeline. This robot is validated through Solidworks simulation. There was no physical model constructed. **[11]**

1.3 Two different pipeline inspection robot and our differences

1.3.1 Smart-Spider: Autonomous Self-driven In-line Robot for Versatile Pipeline Inspection

Ying Qu, Petar Durdevic, Zhenyu Yang are designed an inline robot called Smart-Spider. This robot can detect obstacles and react themautomatically with its MCU controller. In this robot there was no cable to communicate with robot. Wireless communication is used to send commands start and stop via GUI, also to send real time information to workstation.**[11]**

Smart-Spider consisted two different parts. One is the tube body, other part is flexible clutch mechanism. Fourbar mechanism Is used to adapt robot to different diameters(see figure I-1).

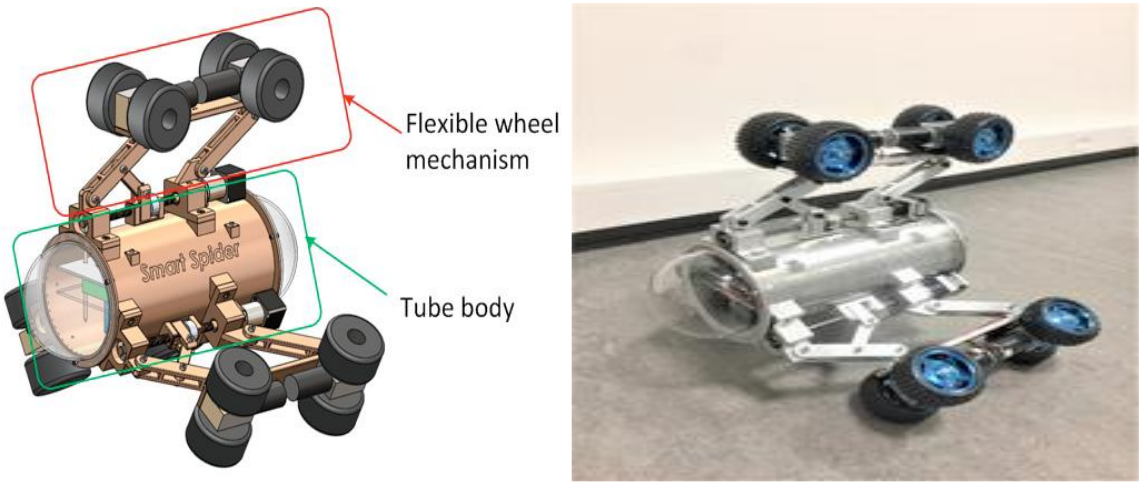


Figure I-1 *Smart-Spider*[11]

Smart-Spider can get pressure data from three flexible mechanisms. So, according to this data it can squeeze itself to pipe. In figure I-3 you can see pressure graphs of three mechanisms.

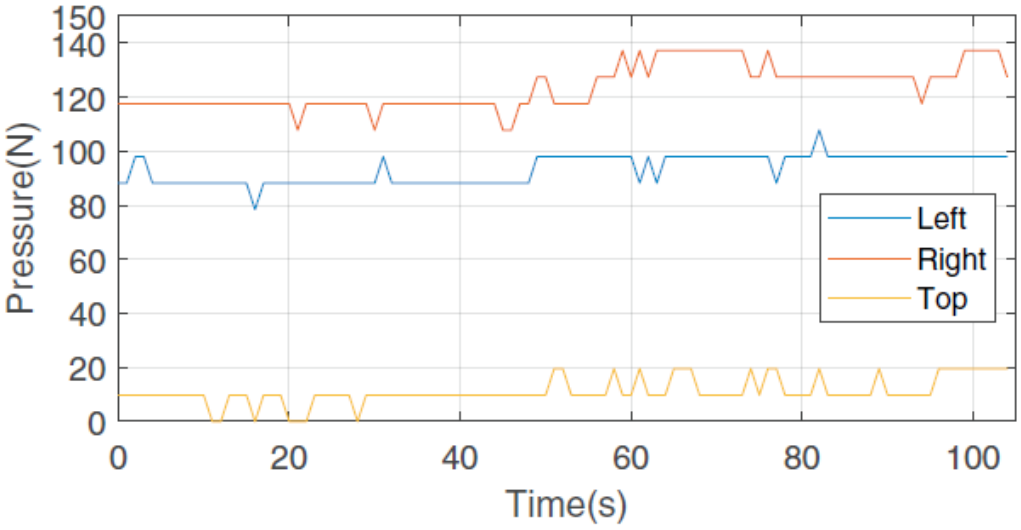


Figure I-2 *Pressure values of three mechanism*[11]

So, smart spider is a inline robot, it can handle with obstacles, it can be used different in different diameters. There is no problem about tethers or cables, so it can operate in complex industrial pipelines.

1.3.2 Laboratory-Scale pipeline inspection robot

Lee Vuen Nee, I. Elamvazuthi, Timothy Ganesan,M.K.A. Ahamed Khan and S. Parasuraman are introduced a Laboratory-Scale pipeline inspection robot. This robot is developed in order to be used at undergraduate level teaching. It consists of three motor. Two motors are used to move robot forward in straight line. A color sensor is attached to other motor. So, robot can detect cracks which are represented in different colors. Data of determined cracks are sent to master with Bluetooth connection. You can see slave and master in Figure I-3. **[12]**

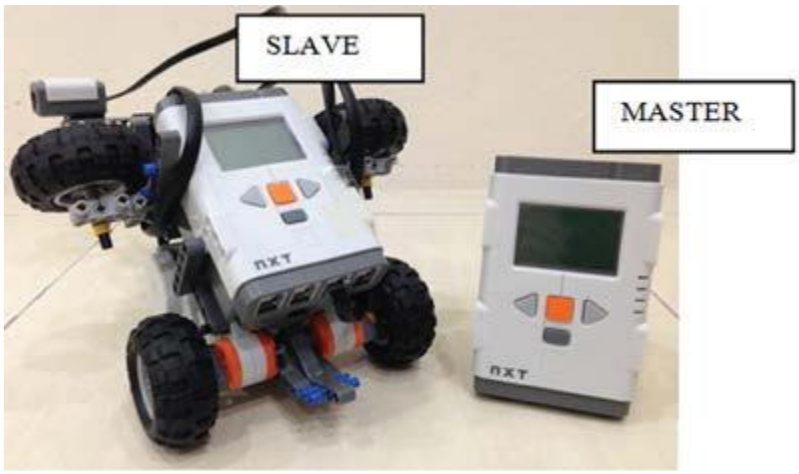


Figure I-3 *Slave and master***[12]**

Accuracy of crack detection graph is obtained for three different conditions. Which are robot speed, scanning degree(180° or 360°) and inclination amount of pipe. **[12]**

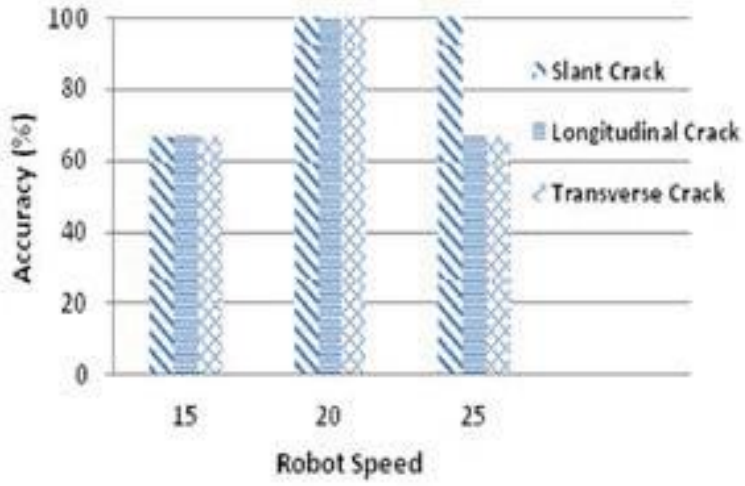


Figure I-4 *Robot speed vs. Accuracy graph***[12]**

In Figure I-4 accuracy is shown with percentage according to speed of robot for three different crack types.

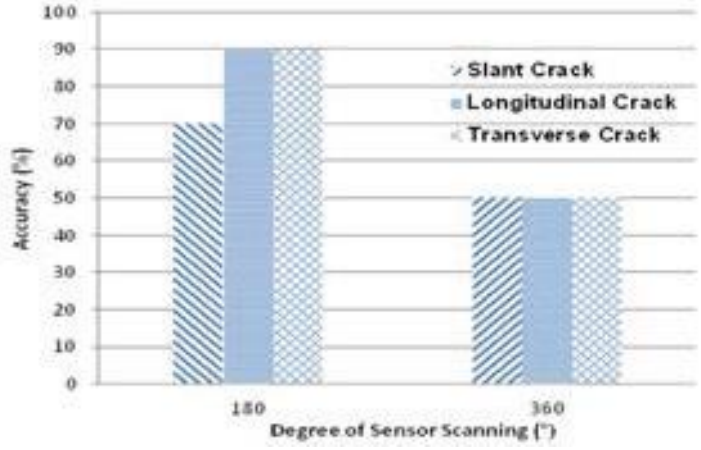


Figure I-5 *Degree vs. Accuracy graph***[12]**

In Figure I-5 accuracy is shown with percentage according to degree of sensor scanning for three different crack types.

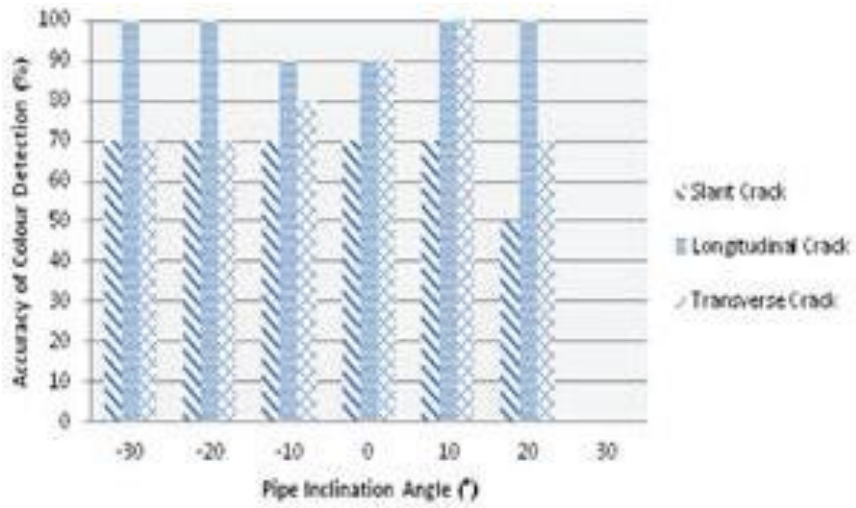


Figure I-6 *Inclination angle vs. Accuracy of Colour Detection graph***[12]**

In Figure I-6 accuracy is shown with percentage according to inclination amount of pipe.

As conclusion, "Colour Sensor is selected to be attached to the robot for the simulation of crack detection where Transverse crack, longitudinal crack and Slant crack are represented by RED, YELLOW and BLUE colour respectively. From the experiments, this PIR is proven to be able to detect crack at high accuracy of 90% (Transverse crack), 90% (Longitudinal crack) and 70% (Slant crack). It as found that this pipeline inspection robot is useful for educational purposes at undergraduate level." **[12]**

So, there are lots of pipeline inspection robot types which are used to be for different applications or experiments. In this introduction, we introduced two different robots to explain usage areas of these robot types briefly.

1.4 Our pipeline inspection robot pipelinebot

After we made literature review about pipeline robots. We see that all of the pipeline robot are wheeled. So, we decided to design and manufacture a pipeline inspection robot with legs. We, developed our design of centipede to be able to use it inside a pipeline. After that, we designed our pipeline inspection robot called pipelinebot. Thanks to multi degree of freedom design, we have several advantages according to wheeled robots. Our advantages according to wheeled robots are:

- Our leg mechanism is fivebar and has two servo motors each leg, so we can arrange path for each leg independently.
- Our system has two dof,so we can arrange different paths by giving end coordinate of robot, by using inverse kinematic equations our microcontroller is solving these equations and gives motors actuator angles.
- We have two extra motors on one side of holder and one in middle. Mid motor produces us turning motions, and holder motor rotates our one holder. So, we can turn right and left, up and down. Thanks to our holder motor we can rotate to touch every point inside pipe.

- If we add some sensors to our rotating holder, we can detect cracks, corrosion etc., so we can avoid pipe burst or pipe fracture.

- Since we can draw any trajectory within our workspace, different trajectory types can be used for different pipe types. Also, if robot encounters some obstacles we can change walking trajectory to pass obstacle.
- We can develop our design to have gripper hands. However, we can carry unwanted objects in pipelines. İf we add necessery tools to this hands, we can do processes such as; cleaning, welding, washing, dyeing etc.
- Finally, thanks to our multi degree of freedom system, our robot can be developed for several applications. According to wheeled robots, we can also do several works like cleaning and repairing.

II. DESIGN AND SIMULATION RESULTS

2.1 Conceptual Design

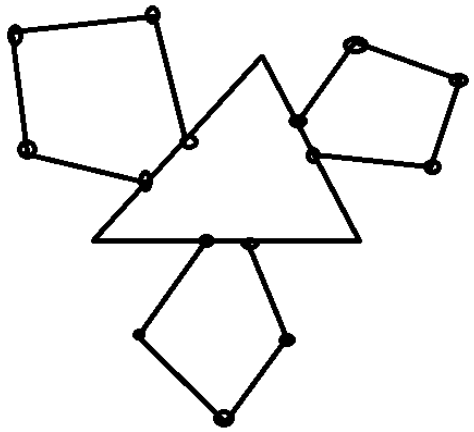


Figure II-1 *Conceptual design of leg mechanism*

First design of our leg mechanism is made with three legs and a holder.

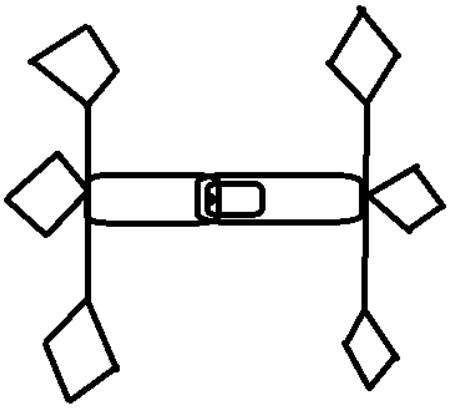


Figure II-2 *Conceptual design of pipelinebot*

Our conceptual design of robot is made with two holder and six legs. Also one motor in middle is used to obtain turning motions.

2.2 Detailed Design

2.2.1 Technical Drawings

Technical drawings are made with Solidworks. Later, these parts are printed with 3D printer.

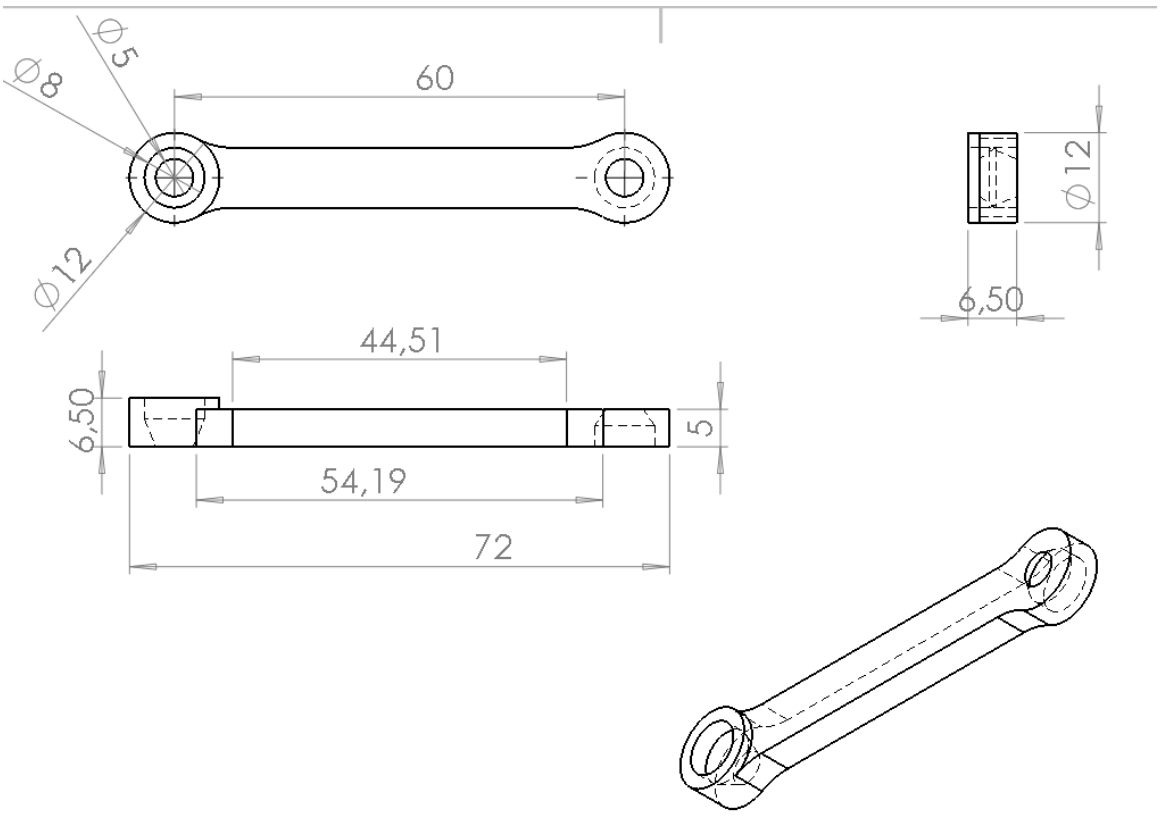


Figure II-3 *Technical drawing of mid link*

In mid link we use two bearing which is Mr85. Its lenght is defined as 60mm between bearings. Also, width of mid link is defined as 6,50mm. Its protruding is made in order to avoid friction between links. Finally, height is defined as 12mm.

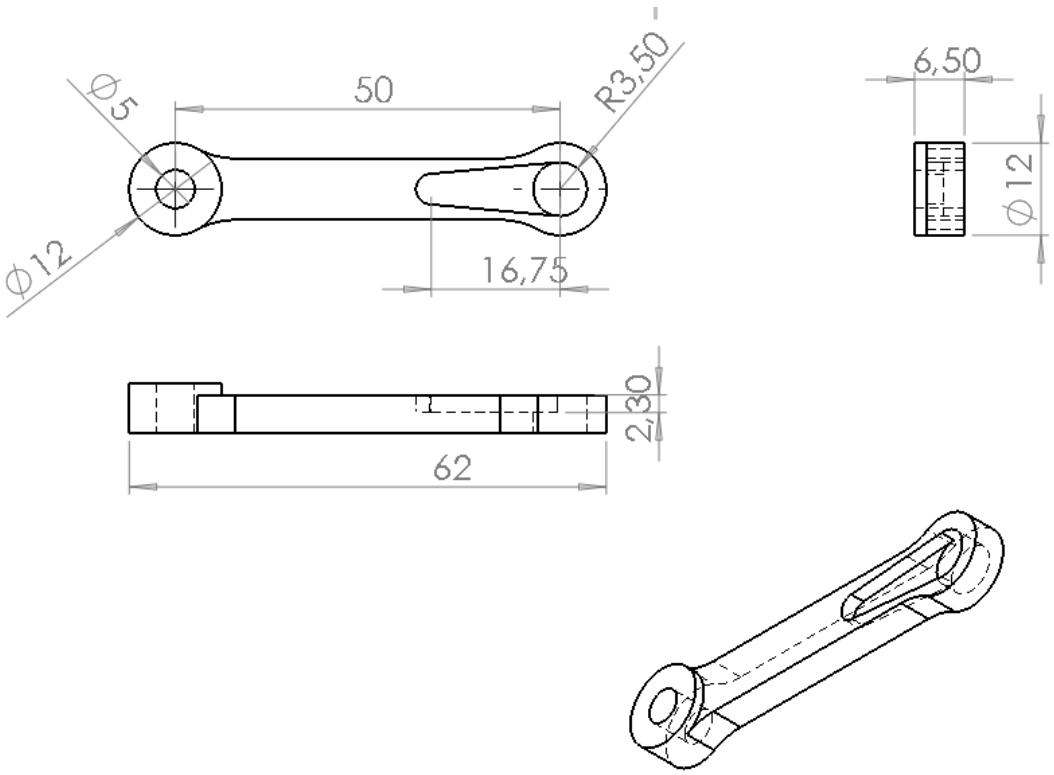


Figure II-4 *Technical drawing of connection link*

We use connection link to connect motors to legs. In order to do this, we use servo arm(see figure II-6). So, we obtained a pocket to put servo arm with depth 2.30mm. Lenght is defined as 50mm between holes. One hole is used to shrink fit steel bar which is 5mm, to the other hole motor is to be connected.

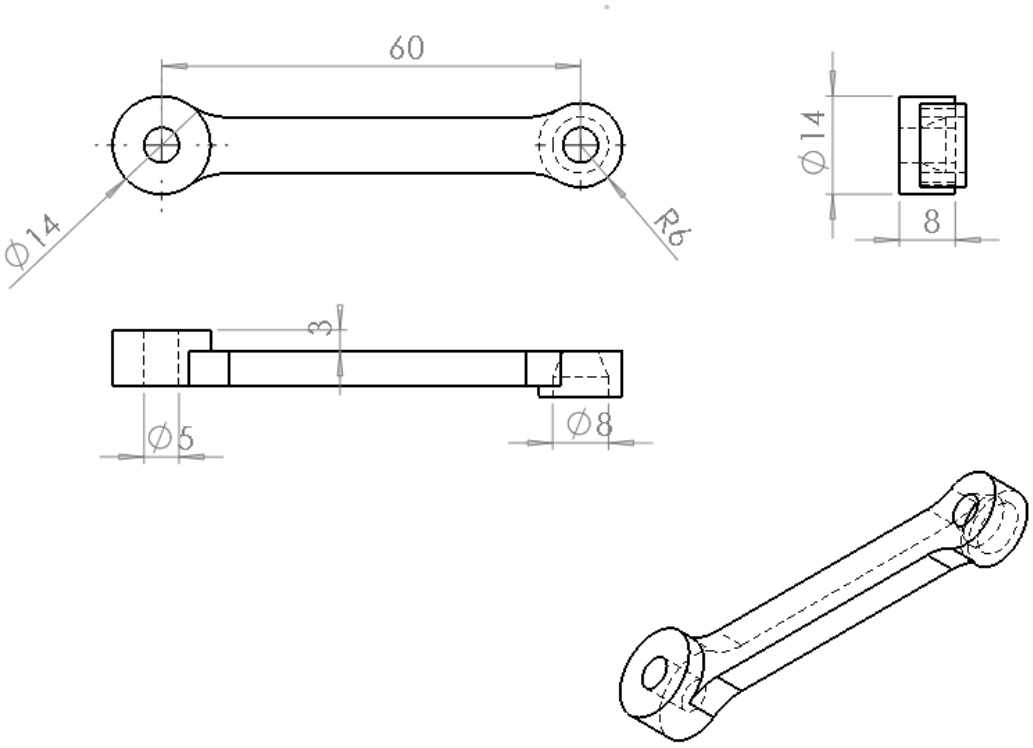


Figure II-5 *Technical drawing of g-link*

Our g-link which is touching to inside of pipelines. So, its diameter one side is arranged as 14mm. Lenght is same as mid link 60mm. It's one hole is used to shrink fit steel bar, and other hole is used to shrink fit the bearing with diameter 8mm.

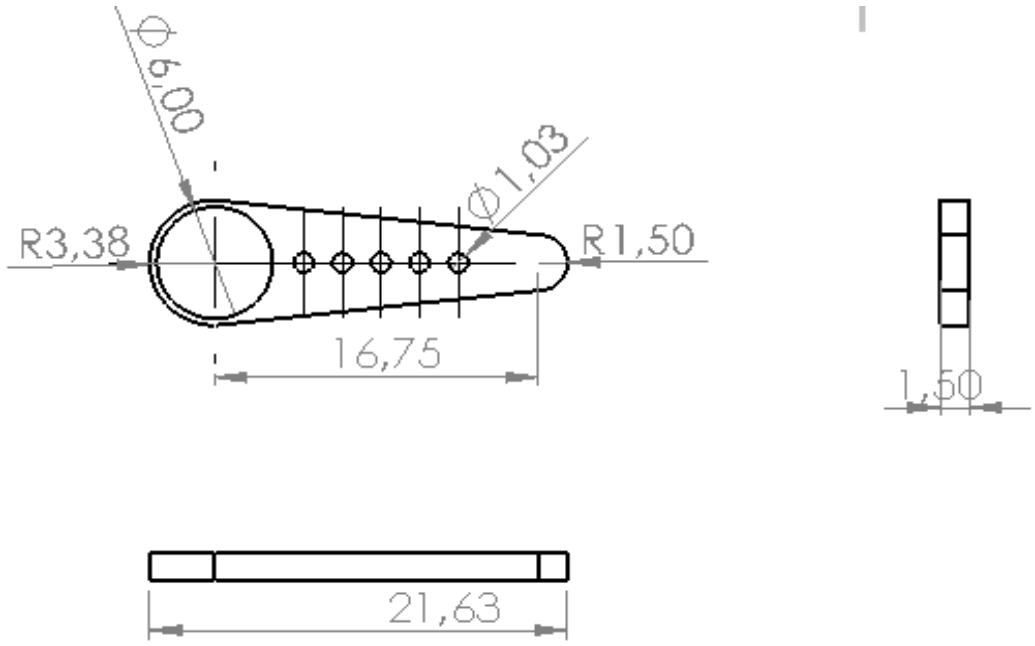


Figure II-6 *Technical drawing of servo arm*

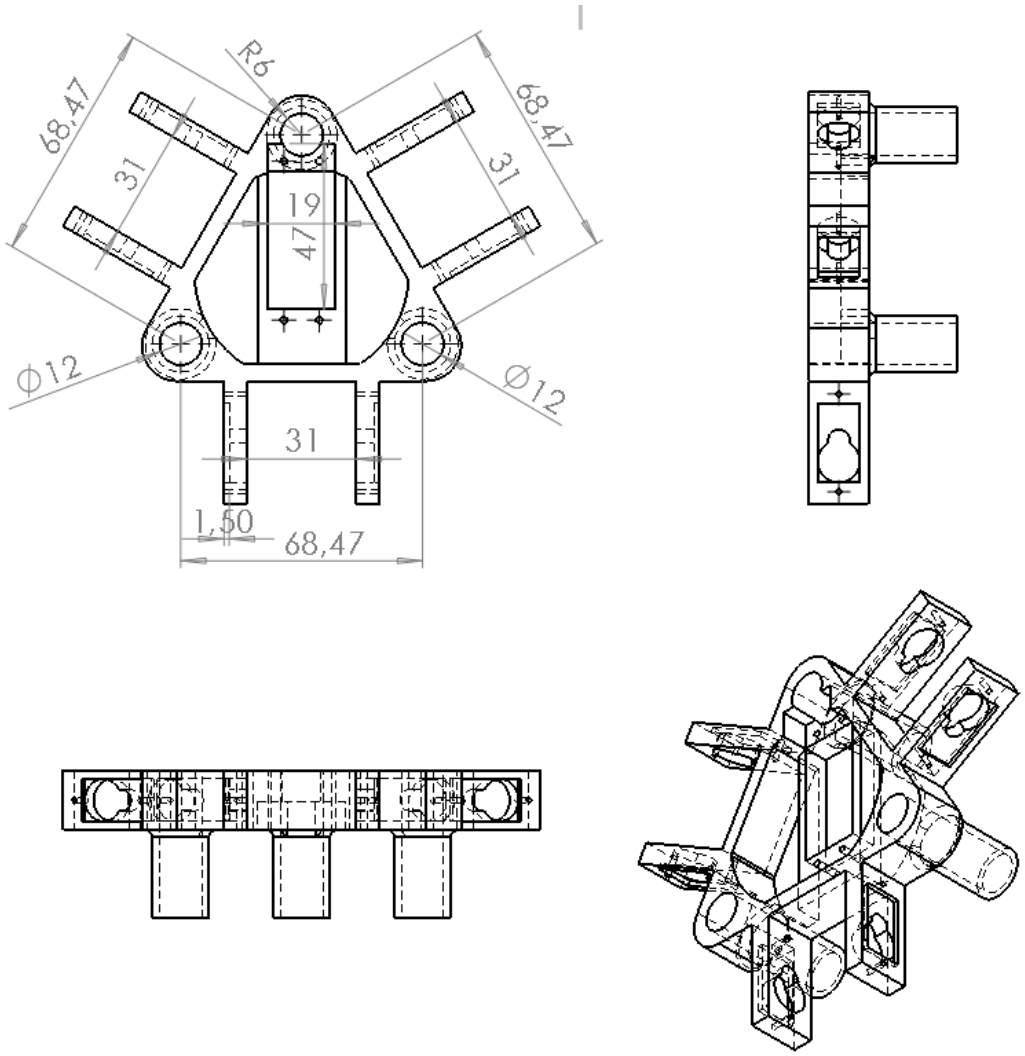


Figure II-7 *Technical drawing of holder-one*

Our holders are designed to hold three legs. 31mm width is designed to have exact touch to sides. 12mm holes are used to obtain a way for cables, so cables can't touch legs. A hole with rectangular shape is designed to middle to hold our motor es3005. By utilizing this motor we can rotate our holder, so we can touch every point inside of pipe. 1,50 mm depth is designed to shrink fit our mg90s motors.

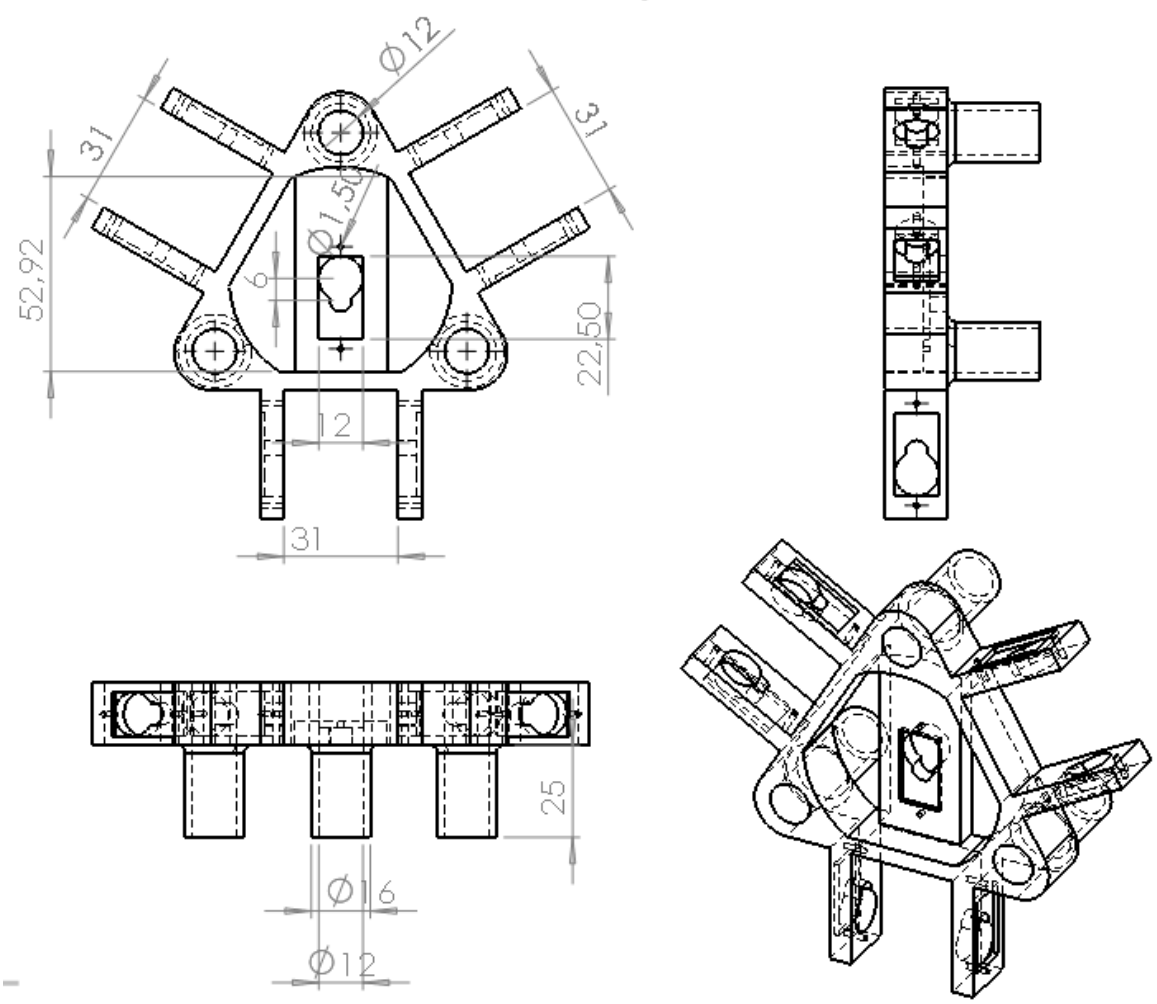


Figure II-8 *Technical drawing of holder-two*

Holder two is to be used on other side of robot. Middle hole designed firstly for mg90s. But, since its torque wasn't enough, we shrink fit here our turn-one component.(see figure II-9). Other properties of this part are same as holder-one.

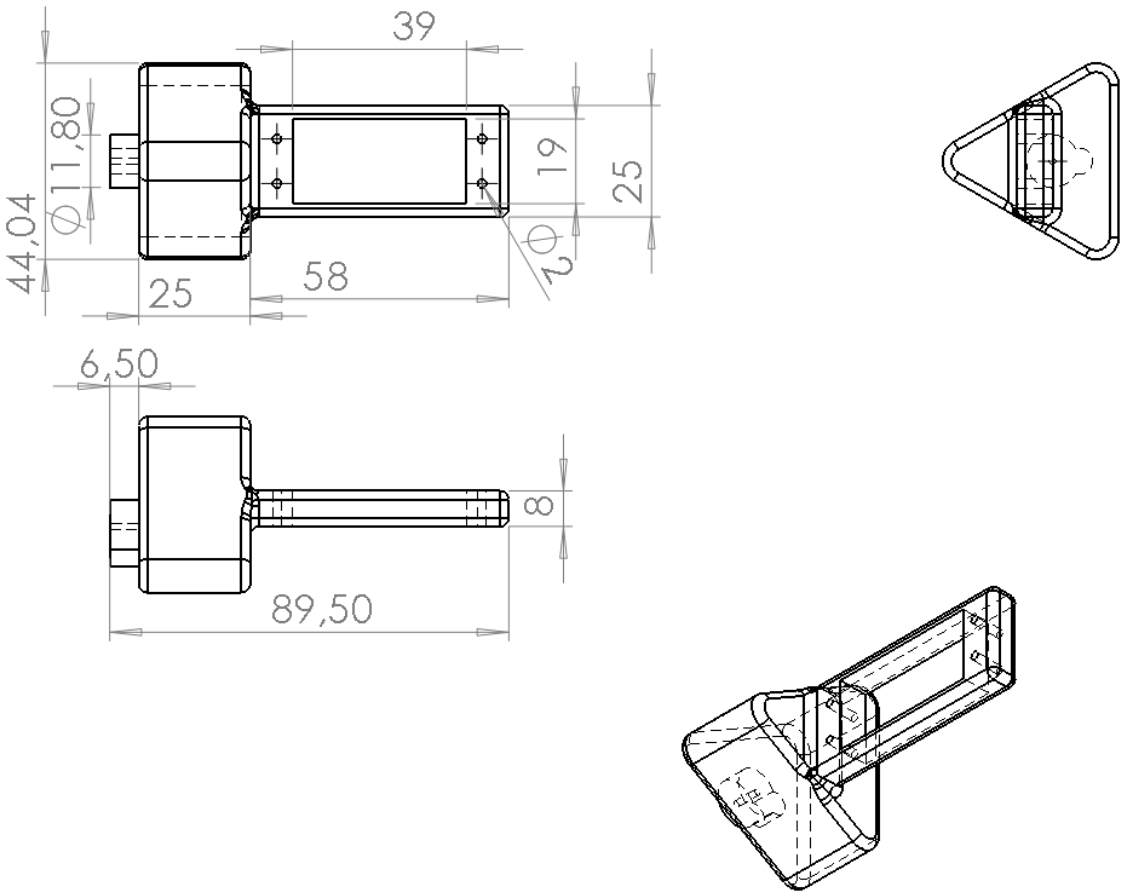


Figure II-9 *Technical drawing of turn-one*

We designed two parts to connect out two holders. One is turn-one and other one is turn-two. Turn one contains our motor es3005, so we designed rectangular hole 39mm to 19mm. The raise in left part is connected to holder-two. Thanks to our turning part on middle of our design, we can turn our robot left and right, or up and down.

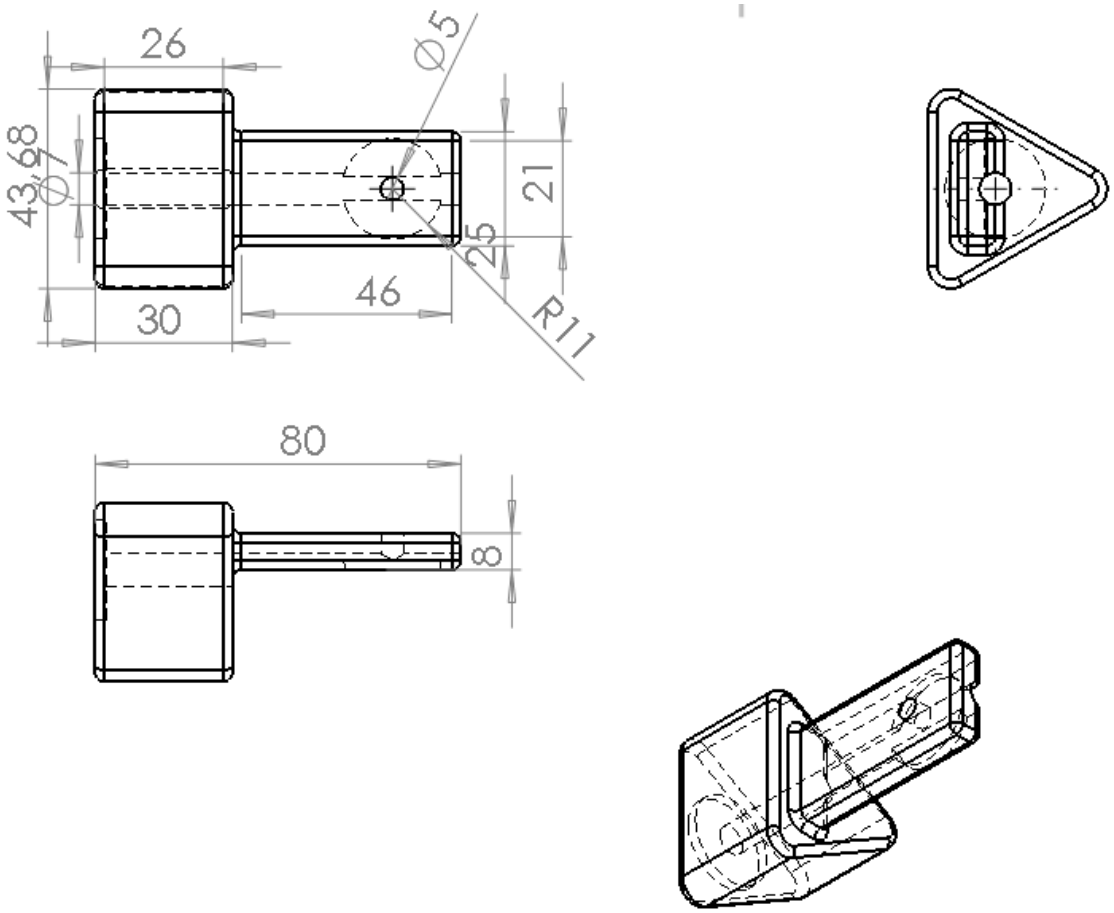


Figure II-10 *Technical drawing of turn-two*

Our turn two part is turning part. 5mm hole is designed two take mile of the es3005 motor. Left side of turn-two takes our other es3005 motor, which is attached to holder-one component. Length of this part is designed as 80mm and width is designed as 43.68mm.

2.2.2 3D views of assembled parts

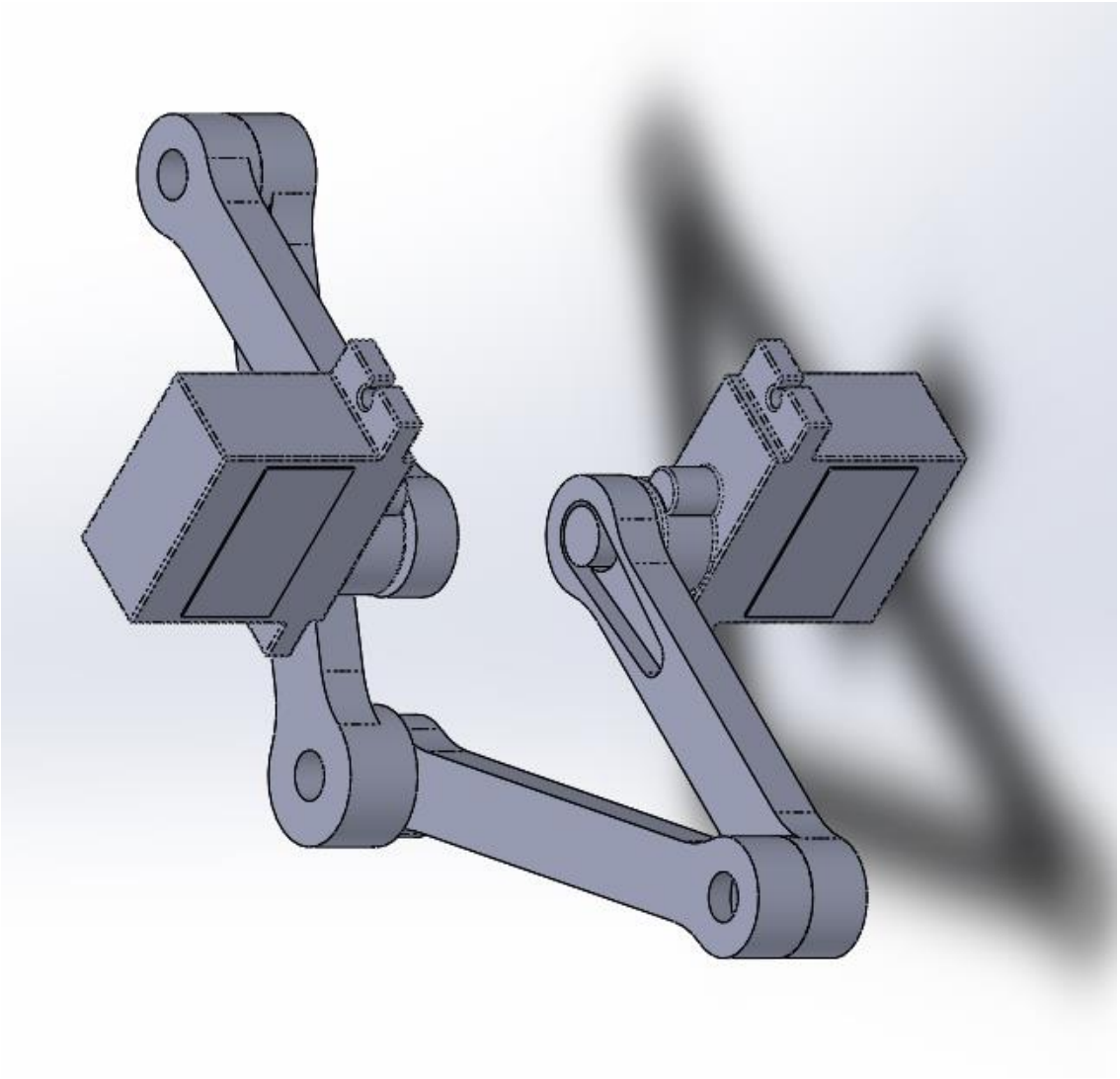


Figure II-11 *3D view of assembled leg*

We used Mr85 bearing between legs to reduce friction. A steel bar with 5mm is shrink fit one leg and to other leg bearing is used, steel bar also shrink fit to this bearing.

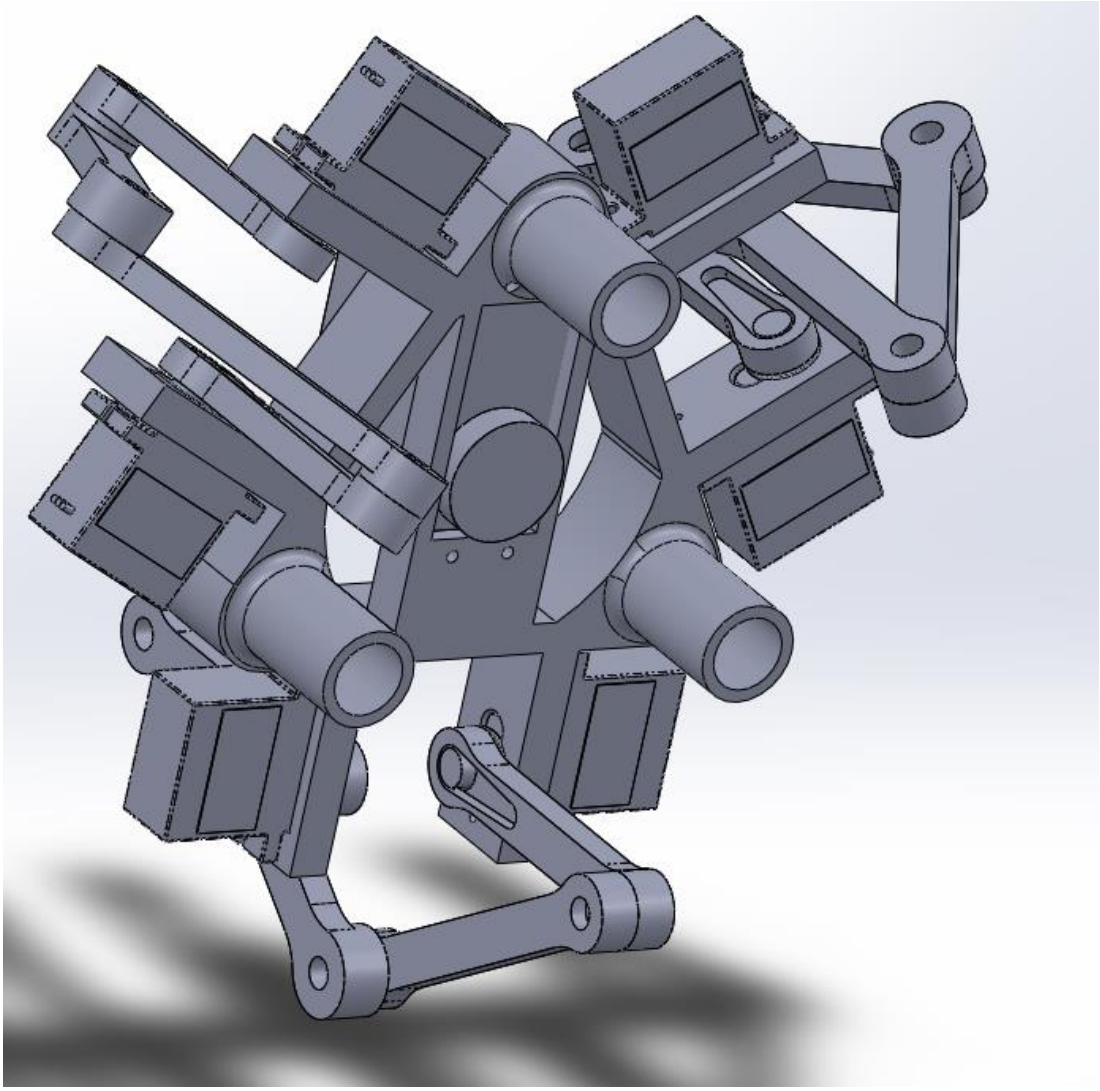


Figure II-12 *3D view of assembled leg and holder-one*

We used three legs on one holder to obtain mass balance. Also with three legs we can squeeze out robot to pipeline properly.

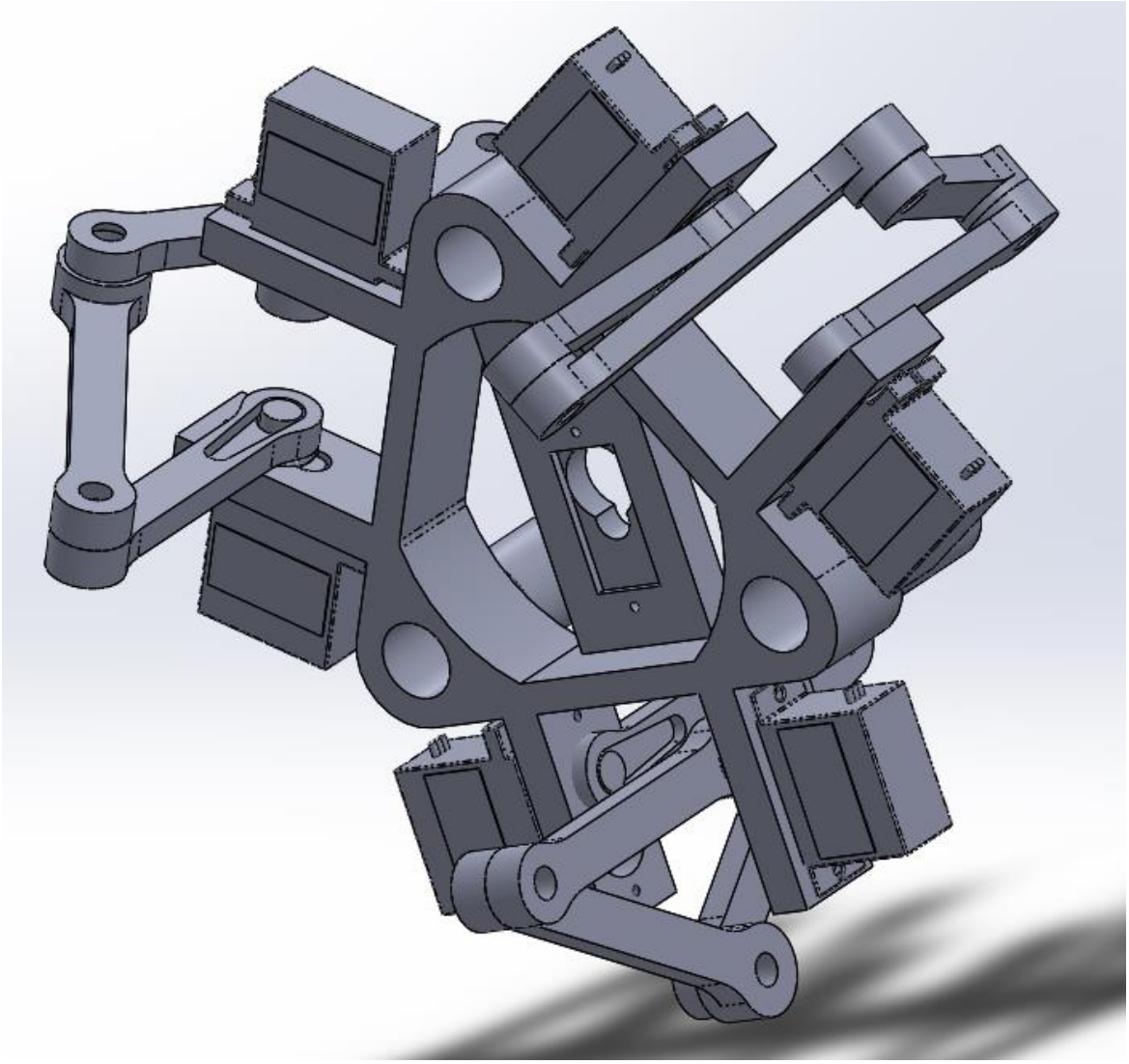


Figure II-13 *3D view of assembled leg and holder-two*

We don’t use a motor in holder two. Middle hole is used to shrink fit our turn-one component.(see figure II-9)

2.2.3 Final assembly and Pipelinebot

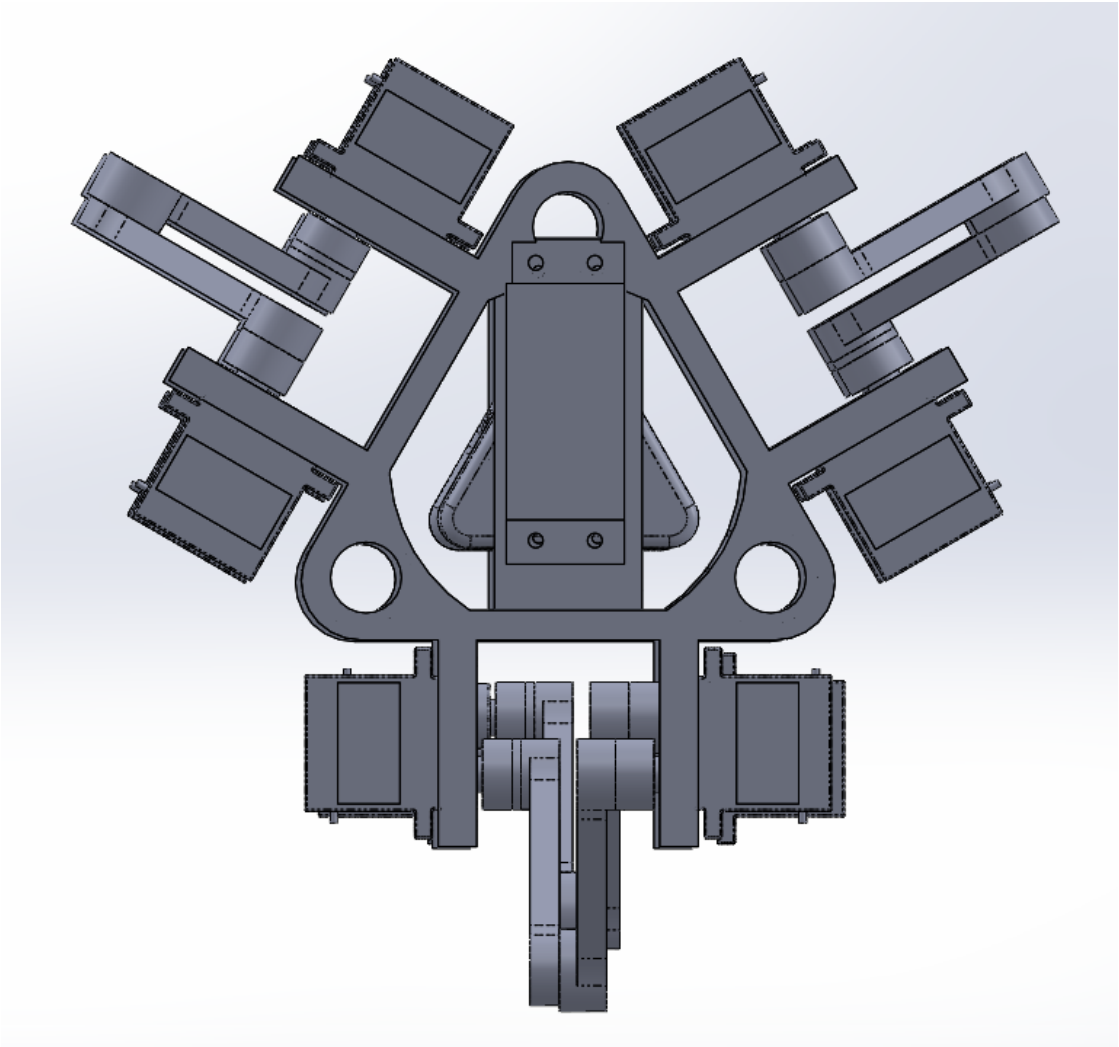


Figure II-14 *Side view of final assembly*

Side view of pipelinebot is shown in figure II-14.

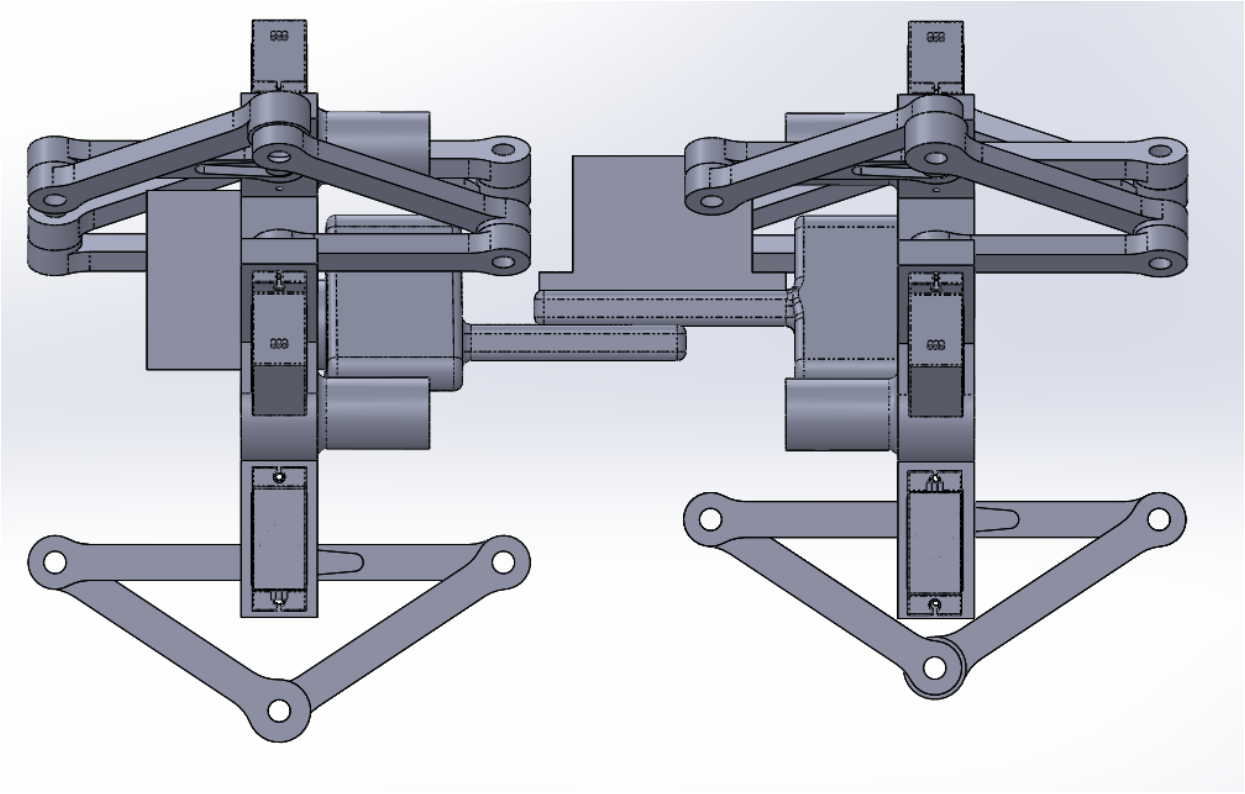


Figure II-15 *Top view of final assembly*

Top view of pipelinebot is shown in figure II-15.

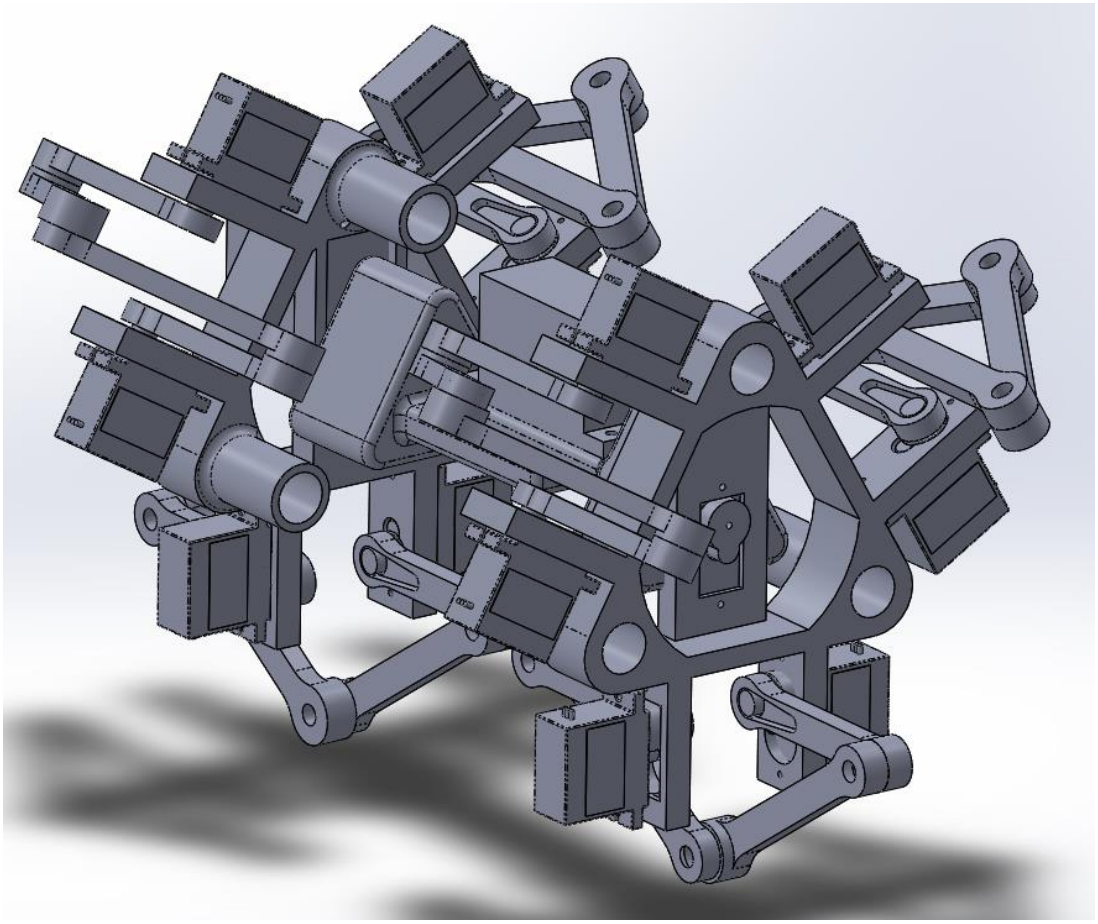


Figure II-16 *3D view of final assembly*

3D view of pipelinebot is shown in figure II-16.

Figure II-17 *Physical model of pipelinebot*

A physical model is constructed of pipelinebot to see it's behaviour in real life. We are going to validate pipelinebot in real life experimentally.

2.2.4 Work Space

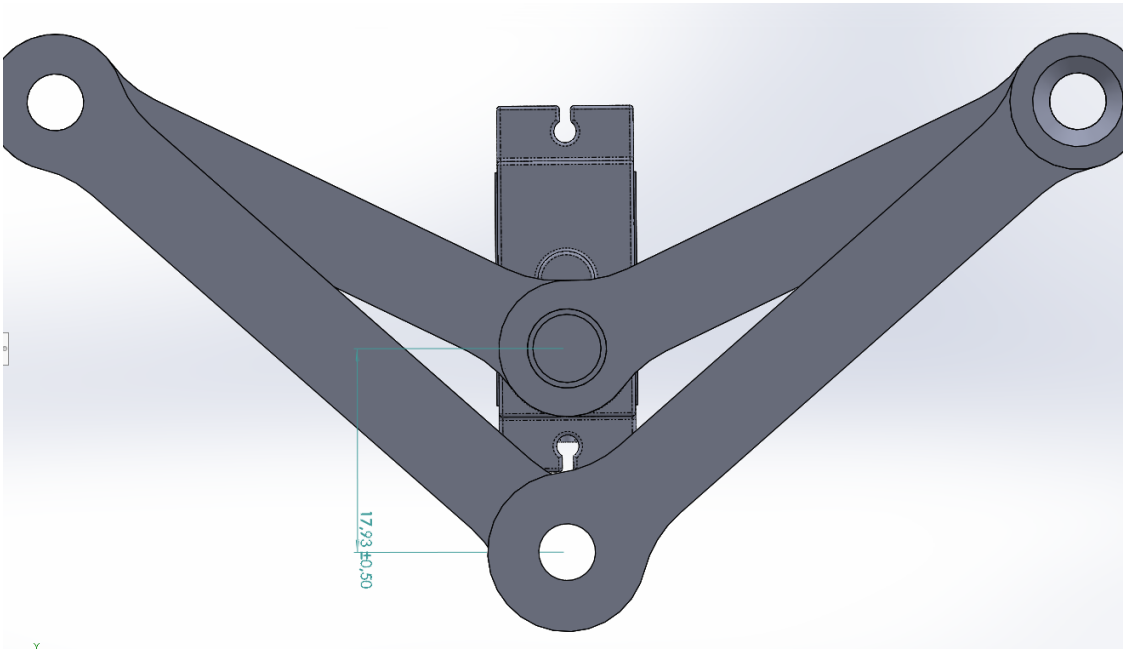


Figure II-18 *Lowest leg height*

Lowest leg height is designed as 17.93mm.

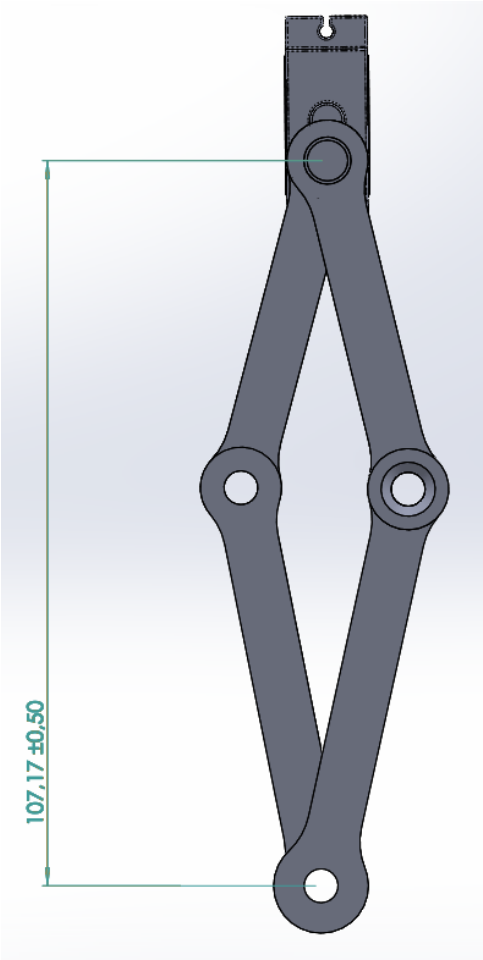


Figure II-19 *Highest leg height*

Highest leg height is designed as 107.17mm. So, we can work inside pipelines which diatemer is between 17.93mm and 107.17mm.

2.2.5 Simulation

2.2.5.1 Simulation Snapshots

Simulation process is carried out with Solidworks in order to validate our design. We proved that pipelinebot can go forward and backward inside a pipeline.

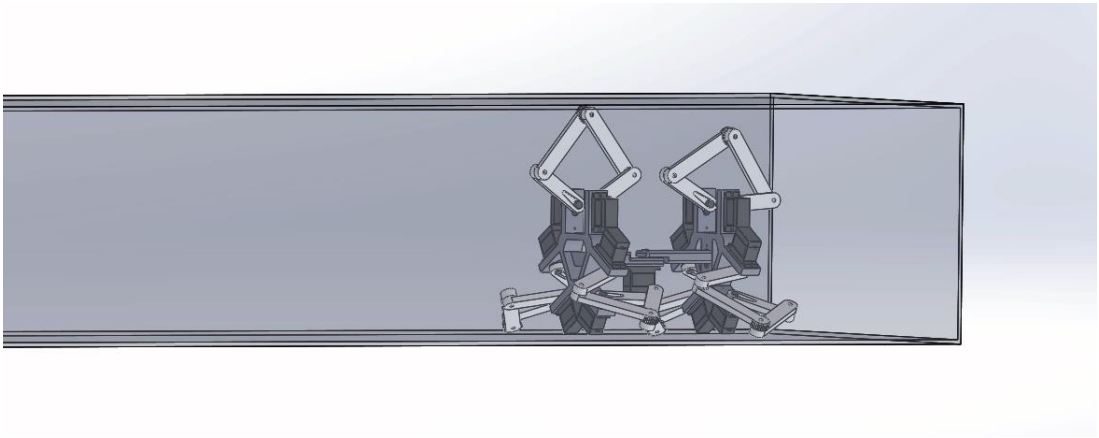


Figure II-20 *Pipelinebot inside a square pipeline*

By using Solidworks simulation we proved that pipelinebot can go forward and backward inside a square pipeline.

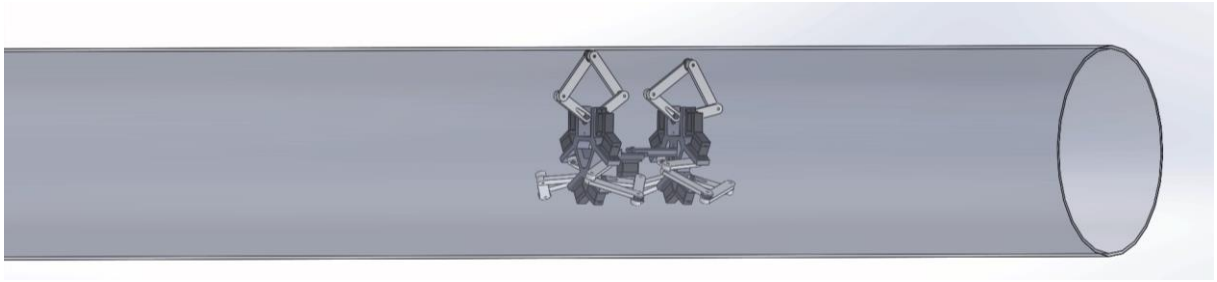


Figure II-21 *Pipelinebot inside a circular pipeline*

Besides square pipeline, our robot can move inside a circular pipeline.

2.2.5.2 Simulation Results

We made energy consumption analysis through Solidworks to obtain required battery for our application.

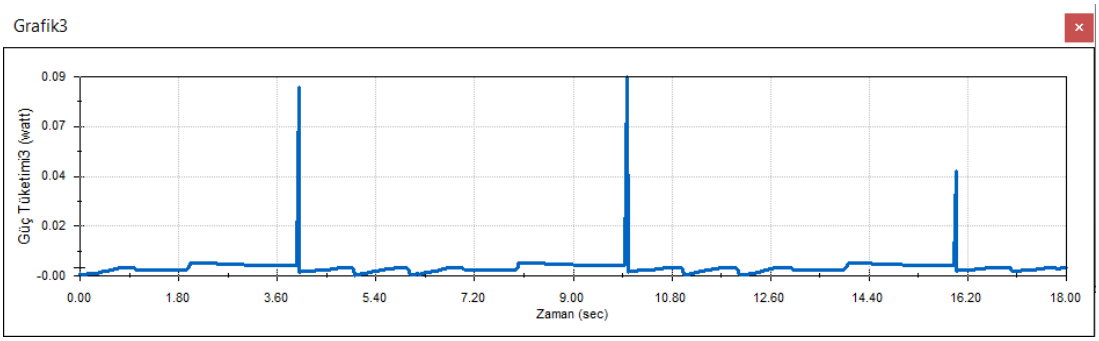


Figure II-22 *Unloaded motor1 energy consumption graph*

We carried out simulations for two situations which are unloaded and loaded cases. For motor1, unloaded simulation analysis can be seen in Figure II-22. There exists peeks at contacting phases, besides these energy consumption almost zero for air phase.

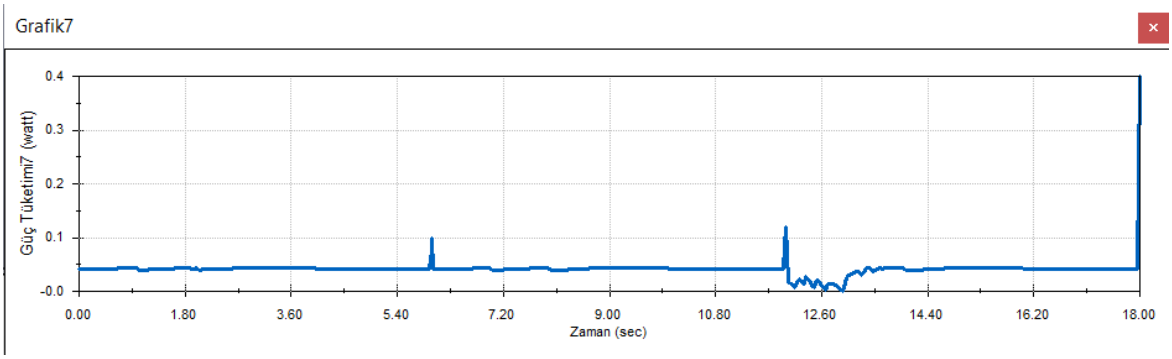


Figure II-23 *Loaded motor1 energy consumption graph*

In figure II-23 loaded analysis can be seen. So, since gravitational effect pipelinebot can rotate. This situation causes error and we can see time shift between two graphs. Also, energy

consumption increases 0.07 to 1 watt. We can see an oscillation at time 12.60 seconds, this is caused by unwanted rotation of robot inside of pipeline.

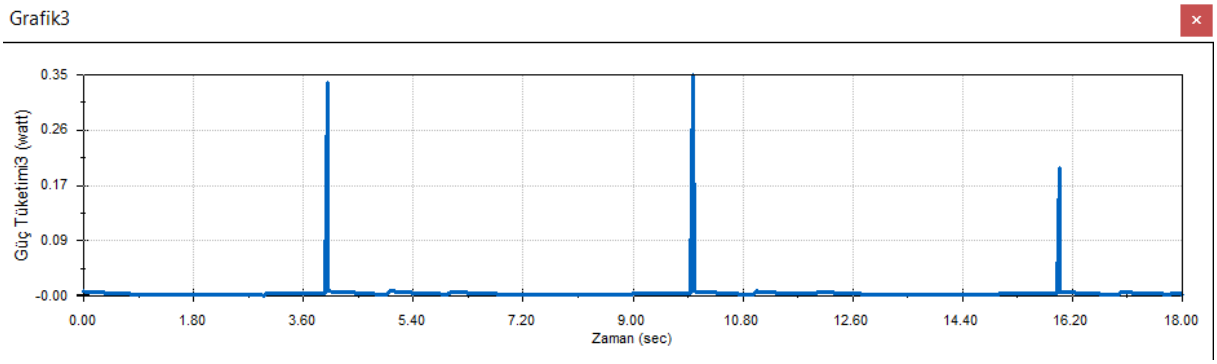


Figure II-24 *Unloaded motor2 energy consumption graph*

Motor2 is in same leg with motor1. For unloaded energy analysis, we can see peek power is 0.35 watt.

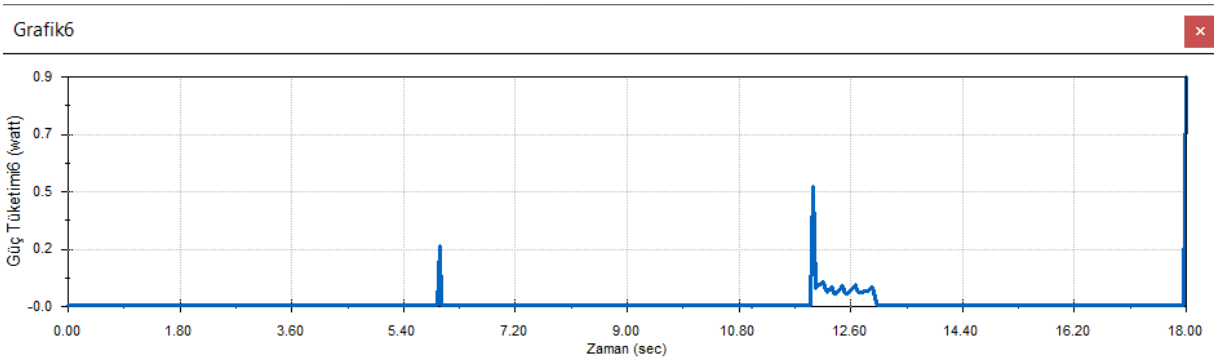


Figure II-25 *Loaded motor2 energy consumption graph*

For loaded case of motor2 we can see peek voltage as 0.5 watt.

In order to see shifted time, we also simulated other leg which contacts after this leg.

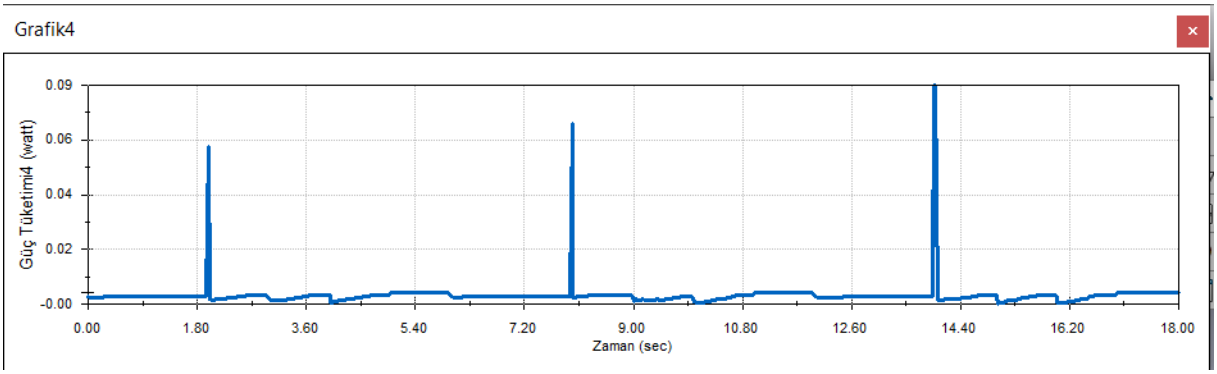


Figure II-26 *Unloaded motor3 energy consumption graph*

We can obtain a time shifted graph according to motor1 and motor2.

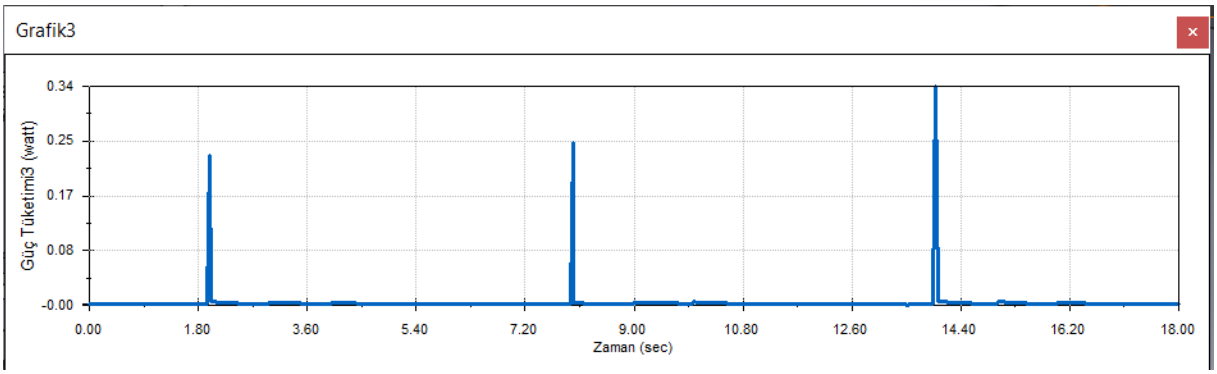


Figure II-27 *Unloaded motor4 energy consumption graph*

Motor4 in in same leg with motor3. So their peek times are same, but we can see difference in peek watts.

III. CIRCUIT COMPONENTS AND CIRCUITRY

3.1 Circuit components

Battery: We are going to use two li-po battery to supply our circuit. We plan to use 11.1V li-po battery to supply servo motors .Another 7.4V li-po battery is going to be used for supplying microcontroller.

Figure III-1 *Li-po battery*

Voltage regulator module: We will use voltage regulator to reduce voltage value to 6V, so we can supply our servo motors.

Figure III-2 *Voltage regulator module*

Bluetooth module: Bluetooth module is used for remote control of walker.

Figure III-3 *Bluetooth module*

Microcontroller: Microcontroller will be used to control motors. It includes software and solves inverse kinematic problem.

Figure III-4 *Microcontroller*

Servo motor mg90s: We used mg90s servo motors because they are light and has workspace 180 degrees.

Figure III-5 *Servo motor mg90s*

Servo motor es3005: We needed more torque to rotate our holder and obtain rotation motions. So, we used es3005 servo motors for these operations.

Figure III-6 *Servo motor es3005*

3.2 Circuitry

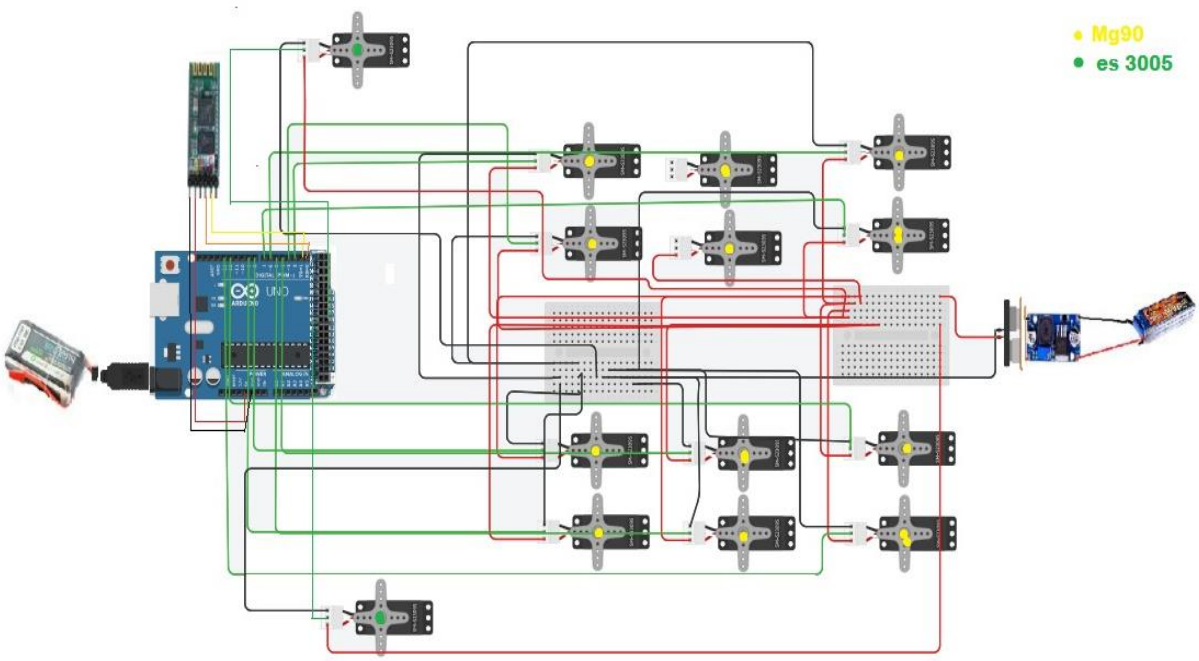


Figure III-7 *Circuitry*

Our assembled circuitry is seen in Figure III-7. Our mid and rotation motors are left side of graph with shown green dot. And motors which are on legs are seen with yellow dot. We use one arduino with 14 pwm outputs to control motors.

IV. TRAJECTORY AND INVERSE KINEMATIC ANALYSIS

For walking trajectory of pipelinebot, we used our smooth trajectory since it has the least energy consumption. Also same inverse kinematic analysis is used to obtain end effector point .

4.1 Trajectory

We used our smooth trajectory for pipelinebot.

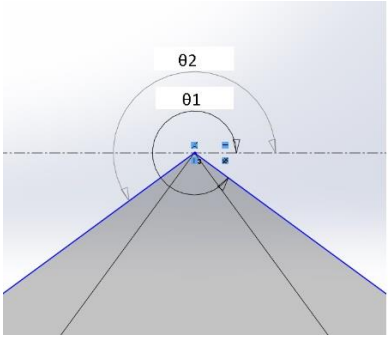


Figure IV-1 *Angles of motors and legs*

In figure IV-1 angles of motors a re defined as θ1 and θ2.

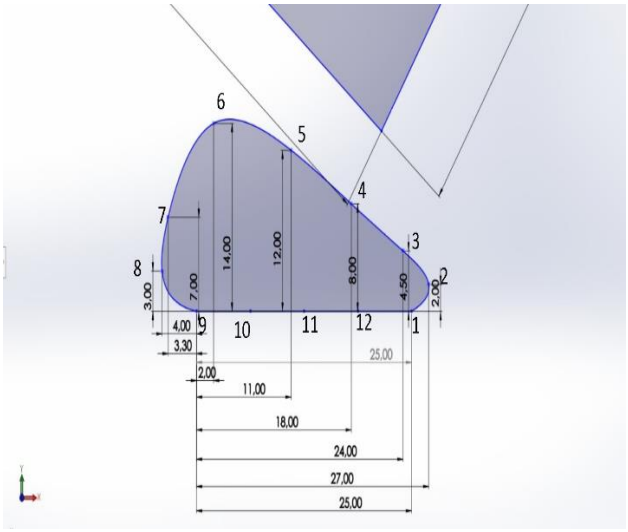


Figure IV-2 *Smooth trajectory*

Position	θ1	θ2
0	0°	180°
1	337,17°	223,08°
2	340,65°	223,43°
3	341,55°	218,37°
4	340,84°	209,3°
5	338,46°	198,58°
6	330,36°	188,40°
7	319,03°	192,81°
8	315,30°	197,03°
9	316,92°	202,83°
10	322,23°	207,03°
11	328,14°	211,86°
12	332,97°	217,23°

Table IV-1 *Smooth trajectory position table*

You can see in figure IV-2 detailed dimensions and points of smooth trajectory. Also, in table IV-1 angles of each motors according to precision points are shown. You can find detailed explanation in section one.

4.2 Inverse Kinematic Analysis

When we are solving inverse kinematic analysis to obtain end effector point of leg, we utilized our new method which we used with our centipede robot. Since, leg designs are same for both robot, we can use this method again. So, we repeated the procedure once again below.

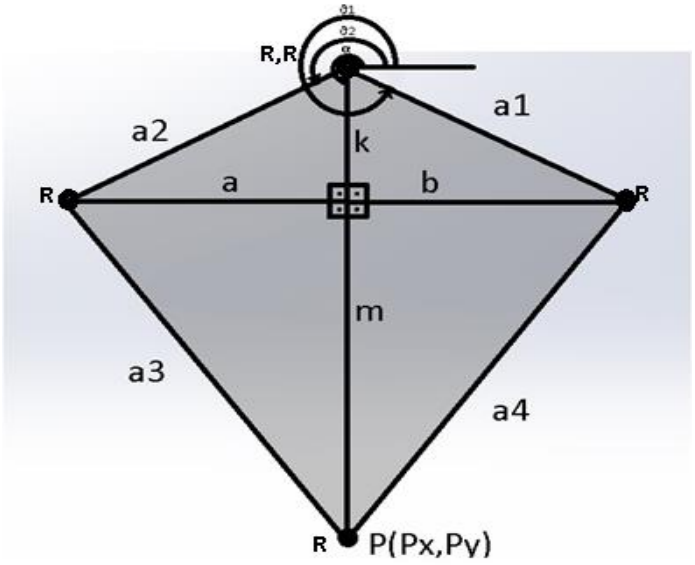


Figure IV-3 *Shematic diagram of one leg chain*

a1=a2=45mm a3=a4=60mm a=b s=k+m…..(1)

s=$\sqrt{Px^2 + Py^2}$..(2) $a^2+k^2=a2^2$…(3) $k^2+b^2=a1^2$….(4) $a^2+m^2=a3^2$…(5)

$m^2+b^2=a4^2$…(6)

By substracting equation 4 and 5:

$k^2-m^2=a1^2-a3^2$ $(k-m)(k+m)= a1^2-a3^2$

From 1:

$k-m=(a1^2-a3^2)/s$

k+m=s

$2k=(a1^2-a3^2+s^2)/s$ $k=(a1^2-a3^2+s^2)/2s$ $m=(s^2-a1^2+a3^2)/2s$…(7)

By substituting k into equation 3:

$a=\sqrt[2]{a_2^2 - ((a_1^2 - a_3^2 + s^2)/2s)^2}$

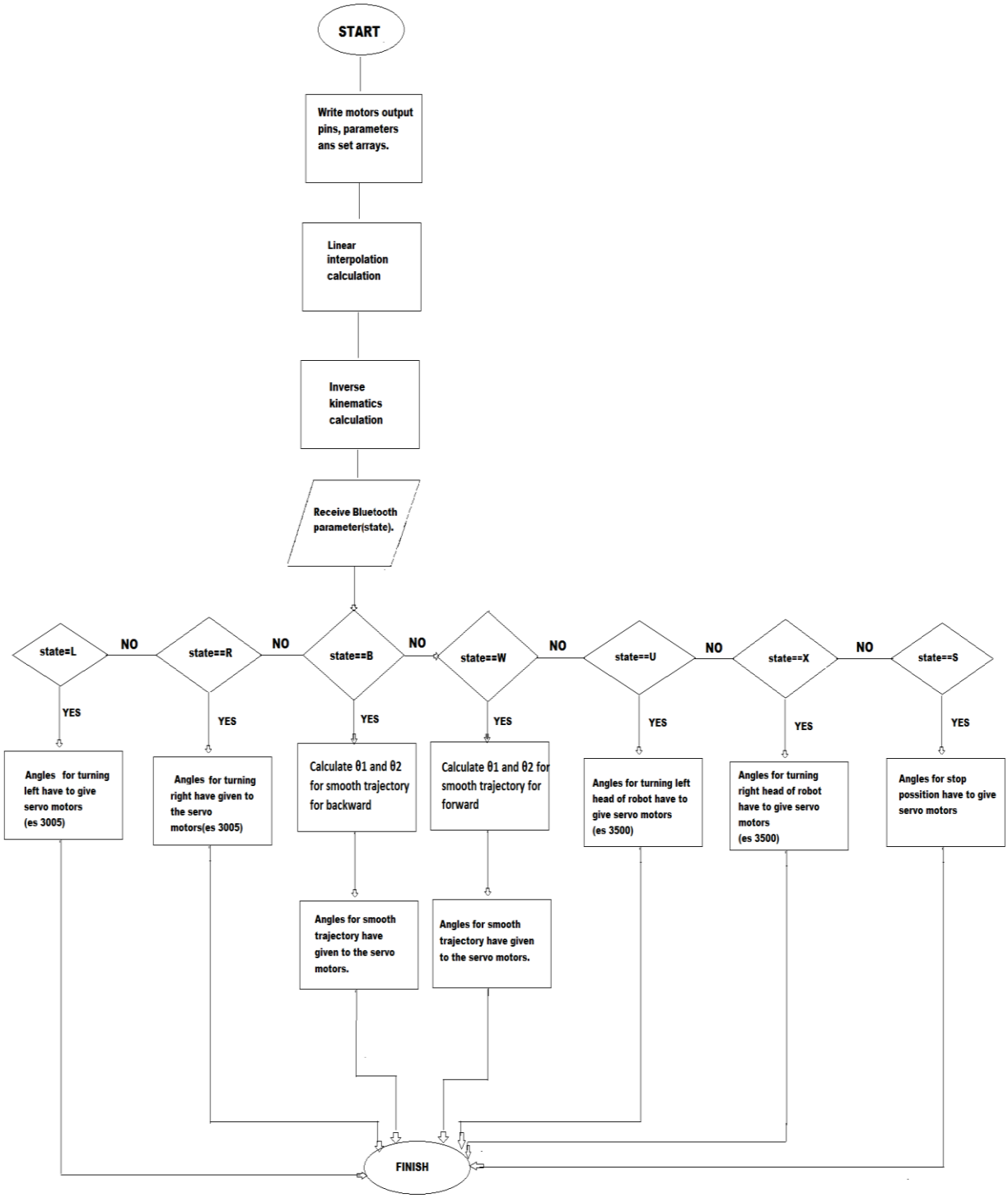
By substituting m into equation 6:

$b=\sqrt[2]{a_4^2 - ((s^2 - a_1^2 - a_3^2+)/2s)^2}$

α=arctan(Py/Px) θ1= α+arcsin(b/a1) θ2= α-arcsin(a/a2)

V. SOFTWARE

5.1 Flowchart



5.2 Software

Software is explained in APPENDIX B.

VI. CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this project we designed and manufactured a pipeline inspection robot.

We used smooth trajectory, which we defined in section one to reduce energy consumption.

We used new inverse kinematic analysis method which is easier than loop closure method. Thanks to this method our calculation procedure gets easier and system works faster.

We validated our inspection robot with Solidworks simulation. We validated only forward and backward motions so far.

We manufactured our robot and assembled, but we could not prove that our physical model works so far.

Finally, with pipeline robot project we proposed a remote controlled robot which can operate inside pipelines. These operations involves procedures like; cleaning, welding, dyeing, carrying objects etc.

6.2 Future work

In future work several sensors can be added to our robot. For instance by using temperature sensors we can detect anomalies inside pipelines.

We may add several trajectories to be used in different conditions. For example, if robot encounters with a obstacle it can change its trajectory to pass through obstacle.

We may add a raspberry pi to process image easily, or avoid memory problems about our microcontroller, when we want to use several trajectories.

We may add camera module, so robot can process image, and detailed data may send to work station.

In later our robot can be autonomous, it can process data from several sensors and can decide its walkins trajectory. Also, can do repairing procedures by itself inside pipelines.

References

[1]https://en.wikipedia.org/wiki/Jansen%27s_linkage (Date of availability 2.1.2020)

[2]https://dogfeatherdesign.com/engineering-projects/mechanisms-mechanical-walker/(Date of availability 2.1.2020)

[3] Simionescu, P. A. (2016, August). MeKin2D: Suite for planar mechanism kinematics. In ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers Digital Collection.

[4]https://en.wikipedia.org/wiki/Chebyshev%27s_Lambda_Mechanism(Date of availability 2.1.2020)

[5] Desai, S. G., Annigeri, A. R., & TimmanaGouda, A. (2019). Analysis of a new single degree-of-freedom eight link leg mechanism for walking machine. *Mechanism and Machine Theory*, *140*, 747-764.

[6]https://en.wikipedia.org/wiki/Peaucellier%E2%80%93Lipkin_linkage(Date of availability 2.1.2020)

[7] Roennau, A., Heppner, G., & Dillmann, R. (2014). ONLINE ADAPTIVE LEG TRAJECTORIES FOR MULTI-LEGGED WALKING ROBOTS. In *Mobile Service Robotics* (pp. 369-376).

[8]GÖRNER, Martin; HIRZINGER, Gerd. Analysis and evaluation of the stability of a biologically inspired, leg loss tolerant gait for six-and eight-legged walking robots. In: 2010 IEEE International Conference on Robotics and Automation. IEEE, 2010. p. 4728-4735.

[9]SOYGUDER, Servet; ALLI, Hasan. Design and prototype of a six-legged walking insect robot. Industrial Robot: An International Journal, 2007, 34.5: 412-422.

[10] FUKUDA, Toshio, et al. Posture control of 6-leg walking robot. In: Proceedings of 1995 IEEE International Conference on Robotics and Automation. IEEE, 1995. p. 1006-1011.

[11] Qu, Y., Durdevic, P., & Yang, Z. (2018). Smart-spider: Autonomous self-driven in-line robot for versatile pipeline inspection. *Ifac-papersonline*, *51*(8), 251-256.

[12] Nee, L. V., Elamvazuthi, I., Ganesan, T., Khan, M. A., & Parasuraman, S. (2015). Development of a laboratory-scale pipeline inspection robot. *Procedia Computer Science*, *76*, 9-14.

A. APPENDIX

SOFTWARE OF CENTIPEDE

```
float s,m,b,a,k,Px,Py,ac;
const int a1=45;
const int a2=45;
const int a3=60;
const int a4=60;
float out1,out2;
const float pi=3.14159;
//turn left
 int hl=0;
 float teta1l,teta2l;
 int gl;
 float teta11l,teta12l,teta11gl,teta12gl;

float P0xl[2]={-12.5,12.5};
float P0yl[2]={-57.5,-57.5};
float P1xl[20];
float P11xl[20];
float P1yl[20];
float t1l[20];
float t2l[20];
float t1gl[20];
float t2gl[20];
float teta1gl,teta2gl;
int il,jl;
//turn right
```

```
float teta11r,teta12r,teta11gr,teta12gr;


 int hr=0;
 float teta1r,teta2r;
 int gr;


float P0xr[2]={-12.5,12.5};
float P0yr[2]={-57.5,-57.5};
float P1xr[20];
float P11xr[20];
float P1yr[20];
float t1r[20];
float t2r[20];
float t1gr[20];
float t2gr[20];
float teta1gr,teta2gr;
int ir,jr;
//square
float teta11k,teta12k;
 int hk=0;
 float teta1k,teta2k;
 int gk;
float P0xk[5]={12.5,12.5,-12.5,-12.5,12.5};
float P0yk[5]={-72.5,-47.5,-47.5,-72.5,-72.5};
float P1xk[60];
float P1yk[60];
float t1k[60];
float t2k[60];
69n tik,jk;


//square backwards
```

```
float teta11kg,teta12kg;

 int hkg=0;
 float teta1kg,teta2kg;
 int gkg;

float P0xkg[5]={-12.5,-12.5,12.5,12.5,-12.5};
float P0ykg[5]={-72.5,-47.5,-47.5,-72.5,-72.5};
float P1xkg[60];
float P1ykg[60];
float t1kg[60];
float t2kg[60];
int ikg,jkg;


//triangle
float teta11u,teta12u;

 int hu=0;
 float teta1u,teta2u;
 int gu;

float P0xu[4]={12.5,0,-12.5,12.5};
float P0yu[4]={-70.83,-49.18,-70.83,-70.83};
float P1xu[40];
float P1yu[40];
float t1u[40];
float t2u[40];
int iu,ju;

//triangle backwards
```

```
float teta11ug,teta12ug;

 int hug=0;
 float teta1ug,teta2ug;
 int gug;

float P0xug[4]={-12.5,0,12.5,-12.5};
float P0yug[4]={-70.83,-49.18,-70.83,-70.83};
float P1xug[40];
float P1yug[40];
float t1ug[40];
float t2ug[40];
int iug,jug;
//Smooth

float teta11o,teta12o;

 int ho=0;
 float teta1o,teta2o;
 int go;

float P0xo[10]={12.5,16.5,15.8,10.5,1.5,-5.5,-11.5,-14.5,-12.5,12.5};
float P0yo[10]={-70,-67,-63,-56,-58,-62,-65.5,-68,-70,-70};
float P1xo[80];
float P1yo[80];
float t1o[80];
float t2o[80];
int io,jo;
//Smooth backwards
```

```
float teta11og,teta12og;


 int hog=0;
 float teta1og,teta2og;
 int gog;

float P0xog[10]={-12.5,-16.5,-15.8,-10.5,-1.5,5.5,11.5,14.5,12.5,-12.5};
float P0yog[10]={-70,-67,-63,-56,-58,-62,-65.5,-68,-70,-70};
float P1xog[80];
float P1yog[80];
float t1og[80];
float t2og[80];
int iog,jog;

#include<Servo.h>
Servo motor1;
Servo motor2;
Servo motor3;
Servo motor4;
Servo motor5;
Servo motor6;
Servo motor7;
Servo motor8;
Servo motor9;
Servo motor10;
Servo motor11;
Servo motor12;
  int state;

void setup() {
    motor1.attach(2);
```

```
   motor2.attach(3);
   motor3.attach(4);
   motor4.attach(5);
   motor5.attach(6);
   motor6.attach(7);
   motor7.attach(8);
   motor8.attach(9);
   motor9.attach(10);
   motor10.attach(11);
   motor11.attach(12);
   motor12.attach(13);


Serial.begin(9600);



//turn left
 for(il = 0; il<20; il++)
  {
   P1xl[hl]=P0xl[0]+(((P0xl[1]-P0xl[0]))/20*il);
   P1yl[hl]=P0yl[0]+(((P0yl[1]-P0yl[0]))/20*il);
   P11xl[hl]=P0xl[1]+(((P0xl[0]-P0xl[1]))/20*il);
   hl=hl+1;
 }


  for(gl = 0; gl<20; gl++)
{
  inverse(P1xl[gl],P1yl[gl]);
  teta1l=out1;
```

```
   teta2l=out2;
if(teta1l<0)
{
  teta1l=360+teta1l;
  }
  if(teta2l<0)
{
  teta2l=360+teta2l;

  }

if (teta1l>180)
{teta11l=360-teta1l+23;}
if (teta1l<180)
{teta11l=23-teta1l;}

if (teta2l>180)
{teta12l=teta2l-180+23;}
if (teta2l<=180)
{teta12l=23-(180-teta2l);
}
t1l[gl]=teta11l;
t2l[gl]=teta12l;
}


  for(gl = 0; gl<20; gl++)
{
   inverse(P11xl[gl],P1yl[gl]);
   teta1gl=out1;
   teta2gl=out2;
```

```
if(teta1gl<0)
{
  teta1gl=360+teta1gl;
  }
  if(teta2gl<0)
{
  teta2gl=360+teta2gl;

  }

if (teta1gl>180)
{teta11gl=360-teta1gl+23;}
if (teta1gl<180)
{teta11gl=23-teta1gl;}

if (teta2gl>180)
{teta12gl=teta2gl-180+23;}
if (teta2gl<=180)
{teta12gl=23-(180-teta2gl);
}
t1gl[gl]=teta11gl;
t2gl[gl]=teta12gl;
}



//turn right

for(ir = 0; ir<20; ir++)
  {
    P1xr[hr]=P0xr[0]+(((P0xr[1]-P0xr[0]))/20*ir);
    P1yr[hr]=P0yr[0]+(((P0yr[1]-P0yr[0]))/20*ir);
```

```
    P11xr[hr]=P0xr[1]+(((P0xr[0]-P0xr[1]))/20*ir);
    hr=hr+1;
 }


  for(gr = 0; gr<20; gr++)
{
   inverse(P1xr[gr],P1yr[gr]);
   teta1r=out1;
   teta2r=out2;
if(teta1r<0)
{
  teta1r=360+teta1r;
  }
  if(teta2r<0)
{
  teta2r=360+teta2r;

  }

if (teta1r>180)
{teta11r=360-teta1r+23;}
if (teta1r<180)
{teta11r=23-teta1r;}

if (teta2r>180)
{teta12r=teta2r-180+23;}
if (teta2r<=180)
{teta12r=23-(180-teta2r);
}
t1r[gr]=teta11r;
```

```
t2r[gr]=teta12r;
}


  for(gr = 0; gr<20; gr++)
{
   inverse(P11xr[gr],P1yr[gr]);
   teta1gr=out1;
   teta2gr=out2;
if(teta1gr<0)
{
  teta1gr=360+teta1gr;
  }
  if(teta2gr<0)
{
  teta2gr=360+teta2gr;

  }

if (teta1gr>180)
{teta11gr=360-teta1gr+23;}
if (teta1gr<180)
{teta11gr=23-teta1gr;}

if (teta2gr>180)
{teta12gr=teta2gr-180+23;}
if (teta2gr<=180)
{teta12gr=23-(180-teta2gr);
}
t1gr[gr]=teta11gr;
t2gr[gr]=teta12gr;
```

```
}


//square

for(jk = 0; jk<3; jk++)
{
 for(ik = 0; ik<10; ik++)
  {
    P1xk[hk]=P0xk[jk]+(((P0xk[jk+1]-P0xk[jk]))/10*ik);
    P1yk[hk]=P0yk[jk]+(((P0yk[jk+1]-P0yk[jk]))/10*ik);
    hk=hk+1;
 }



 }
 for(int yk=0;yk<30;yk++)
 { P1xk[hk]=P0xk[3]+(((P0xk[4]-P0xk[3]))/30*yk);
    P1yk[hk]=P0yk[3]+(((P0yk[4]-P0yk[3]))/30*yk);
  hk=hk+1;

  }

 for(gk = 0; gk<60; gk++)
{
   inverse(P1xk[gk],P1yk[gk]);
   teta1k=out1;
   teta2k=out2;
if(teta1k<0)
{
```

```
 teta1k=360+teta1k;
 }
 if(teta2k<0)
{
 teta2k=360+teta2k;

 }

if (teta1k>180)
{teta11k=360-teta1k+23;}
if (teta1k<180)
{teta11k=23-teta1k;}

if (teta2k>180)
{teta12k=teta2k-180+23;}
if (teta2k<=180)
{teta12k=23-(180-teta2k);
}
t1k[gk]=teta11k;
t2k[gk]=teta12k;
}

//square backwards
for(jkg = 0; jkg<3; jkg++)
{
 for(ikg = 0; ikg<10; ikg++)
 {
   P1xkg[hkg]=P0xkg[jkg]+(((P0xkg[jkg+1]-P0xkg[jkg]))/10*ikg);
   P1ykg[hkg]=P0ykg[jkg]+(((P0ykg[jkg+1]-P0ykg[jkg]))/10*ikg);
   hkg=hkg+1;
 }
```

```
 }
 for(int ykg=0;ykg<30;ykg++)
 { P1xkg[hkg]=P0xkg[3]+(((P0xkg[4]-P0xkg[3]))/30*ykg);
    P1ykg[hkg]=P0ykg[3]+(((P0ykg[4]-P0ykg[3]))/30*ykg);
  hkg=hkg+1;

  }

 for(gkg = 0; gkg<60; gkg++)
{
   inverse(P1xkg[gkg],P1ykg[gkg]);
   teta1kg=out1;
   teta2kg=out2;
if(teta1kg<0)
{
  teta1kg=360+teta1kg;
  }
  if(teta2kg<0)
{
  teta2kg=360+teta2kg;

  }

if (teta1kg>180)
{teta11kg=360-teta1kg+23;}
if (teta1kg<180)
{teta11kg=23-teta1kg;}
```

```
if (teta2kg>180)
{teta12kg=teta2kg-180+23;}
if (teta2kg<=180)
{teta12kg=23-(180-teta2kg);
}
t1kg[gkg]=teta11kg;
t2kg[gkg]=teta12kg;
}



//triangle
for(ju = 0; ju<2; ju++)
{
 for(iu = 0; iu<10; iu++)
  {
    P1xu[hu]=P0xu[ju]+(((P0xu[ju+1]-P0xu[ju]))/10*iu);
    P1yu[hu]=P0yu[ju]+(((P0yu[ju+1]-P0yu[ju]))/10*iu);
    hu=hu+1;
 }



 }
 for(int yu=0;yu<20;yu++)
 { P1xu[hu]=P0xu[2]+(((P0xu[3]-P0xu[2]))/20*yu);
    P1yu[hu]=P0yu[2]+(((P0yu[3]-P0yu[2]))/20*yu);
  hu=hu+1;

  }
```

```
 for(gu = 0; gu<40; gu++)
{
   inverse(P1xu[gu],P1yu[gu]);
   teta1u=out1;
   teta2u=out2;
if(teta1u<0)
{
  teta1u=360+teta1u;
  }
  if(teta2u<0)
{
  teta2u=360+teta2u;

  }

if (teta1u>180)
{teta11u=360-teta1u+23;}
if (teta1u<180)
{teta11u=23-teta1u;}

if (teta2u>180)
{teta12u=teta2u-180+23;}
if (teta2u<=180)
{teta12u=23-(180-teta2u);
}
t1u[gu]=teta11u;
t2u[gu]=teta12u;
}
//triangle backwards
for(jug = 0; jug<2; jug++)
{
```

```
 for(iug = 0; iug<10; iug++)
  {
    P1xug[hug]=P0xug[jug]+(((P0xug[jug+1]-P0xug[jug]))/10*iug);
    P1yug[hug]=P0yug[jug]+(((P0yug[jug+1]-P0yug[jug]))/10*iug);
    hug=hug+1;
  }



  }
  for(int yug=0;yug<20;yug++)
  { P1xug[hug]=P0xug[2]+(((P0xug[3]-P0xug[2]))/20*yug);
    P1yug[hug]=P0yug[2]+(((P0yug[3]-P0yug[2]))/20*yug);
   hug=hug+1;

   }

  for(gug = 0; gug<40; gug++)
 {
    inverse(P1xug[gug],P1yug[gug]);
    teta1ug=out1;
    teta2ug=out2;
 if(teta1ug<0)
 {
   teta1ug=360+teta1ug;
   }
   if(teta2ug<0)
 {
   teta2ug=360+teta2ug;

   }
```

```
if (teta1ug>180)
{teta11ug=360-teta1ug+23;}
if (teta1ug<180)
{teta11ug=23-teta1ug;}

if (teta2ug>180)
{teta12ug=teta2ug-180+23;}
if (teta2ug<=180)
{teta12ug=23-(180-teta2ug);
}
t1ug[gug]=teta11ug;
t2ug[gug]=teta12ug;
}



//Smooth
for(jo = 0; jo<8; jo++)
{
 for(io = 0; io<5; io++)
  {
    P1xo[ho]=P0xo[jo]+(((P0xo[jo+1]-P0xo[jo]))/10*io);
    P1yo[ho]=P0yo[jo]+(((P0yo[jo+1]-P0yo[jo]))/10*io);
    ho=ho+1;
 }



 }
 for(int yo=0;yo<40;yo++)
 { P1xo[ho]=P0xo[8]+(((P0xo[9]-P0xo[8]))/40*yo);
```

```
   P1yo[ho]=P0yo[8]+(((P0yo[9]-P0yo[8]))/40*yo);
  ho=ho+1;

  }

 for(go = 0; go<80; go++)
{
   inverse(P1xo[go],P1yo[go]);
   teta1o=out1;
   teta2o=out2;
if(teta1o<0)
{
  teta1o=360+teta1o;
  }
  if(teta2o<0)
{
  teta2o=360+teta2o;

  }

if (teta1o>180)
{teta11o=360-teta1o+23;}
if (teta1o<180)
{teta11o=23-teta1o;}

if (teta2o>180)
{teta12o=teta2o-180+23;}
if (teta2o<=180)
{teta12o=23-(180-teta2o);
}
t1o[go]=teta11o;
```

```
t2o[go]=teta12o;
}
 //Smooth backwards
for(jog = 0; jog<8; jog++)
{
 for(iog = 0; iog<5; iog++)
  {
    P1xog[hog]=P0xog[jog]+(((P0xog[jog+1]-P0xog[jog]))/10*iog);
    P1yog[hog]=P0yog[jog]+(((P0yog[jog+1]-P0yog[jog]))/10*iog);
    hog=hog+1;
 }




 }
 for(int yog=0;yog<40;yog++)
 { P1xog[hog]=P0xog[8]+(((P0xog[9]-P0xog[8]))/40*yog);
    P1yog[hog]=P0yog[8]+(((P0yog[9]-P0yog[8]))/40*yog);
  hog=hog+1;

  }

 for(gog = 0; gog<80; gog++)
{
   inverse(P1xog[gog],P1yog[gog]);
   teta1og=out1;
   teta2og=out2;
if(teta1og<0)
{
  teta1og=360+teta1og;
  }
```

```
 if(teta2og<0)
{
 teta2og=360+teta2og;

 }

if (teta1og>180)
{teta11og=360-teta1og+23;}
if (teta1og<180)
{teta11og=23-teta1og;}

if (teta2og>180)
{teta12og=teta2og-180+23;}
if (teta2og<=180)
{teta12og=23-(180-teta2og);
}
t1og[gog]=teta11og;
t2og[gog]=teta12og;
}




    Serial.begin(9600);

 }
void loop() {

     if(Serial.available() > 0){
     state = Serial.read();
   }
```

```
   else{motor1.write(23);
motor2.write(23);
motor3.write(23);
motor4.write(23);
motor5.write(23);
motor6.write(23);
motor7.write(23);
motor8.write(23);
motor9.write(23);
motor10.write(23);
motor11.write(23);
motor12.write(23);}



//turn left
if(state=='L')
{
  for(gl = 0; gl<20; gl++)
{
motor1.write(t2l[gl]);
motor2.write(t1l[gl]);
motor3.write(t2gl[gl]);
motor4.write(t1gl[gl]);
motor5.write(t2l[gl]);
motor6.write(t1l[gl]);
motor7.write(t2gl[gl]);
motor8.write(t1gl[gl]);
motor9.write(t2l[gl]);
motor10.write(t1l[gl]);
motor11.write(t2gl[gl]);
```

```
motor12.write(t1gl[gl]);
delay(10);

}

  }
  //turn right
  if(state==’R’)
    {
for(gr = 0; gr<20; gr++)
{
motor1.write(t2gr[gr]);
motor2.write(t1gr[gr]);
motor3.write(t2r[gr]);
motor4.write(t1r[gr]);
motor5.write(t2gr[gr]);
motor6.write(t1gr[gr]);
motor7.write(t2r[gr]);
motor8.write(t1r[gr]);
motor9.write(t2gr[gr]);
motor10.write(t1gr[gr]);
motor11.write(t2r[gr]);
motor12.write(t1r[gr]);
delay(10);

}

    }

//triangle
if (state ==’V’){
```

```
for(gu = 0; gu<40; gu++)
{




if(gu>=20){
motor1.write(t2u[gu]);
motor2.write(t1u[gu]);
motor3.write(t2u[gu]);
motor4.write(t1u[gu]);
motor5.write(t2u[gu]);
motor6.write(t1u[gu]);
motor7.write(t2u[gu-20]);
motor8.write(t1u[gu-20]);
motor9.write(t2u[gu-20]);
motor10.write(t1u[gu-20]);
motor11.write(t2u[gu-20]);
motor12.write(t1u[gu-20]);
delay(20);

}
if(gu<20){
motor1.write(t2u[gu]);
motor2.write(t1u[gu]);
motor3.write(t2u[gu]);
motor4.write(t1u[gu]);
motor5.write(t2u[gu]);
motor6.write(t1u[gu]);
```

```
motor7.write(t2u[gu+20]);
motor8.write(t1u[gu+20]);
motor9.write(t2u[gu+20]);
motor10.write(t1u[gu+20]);
motor11.write(t2u[gu+20]);
motor12.write(t1u[gu+20]);
delay(20);
 }



}


  }

//triangle backwards

if(state=='X')
{
  for(gug = 0; gug<40; gug++)
{







if(gug>=20){
motor1.write(t2ug[gug]);
motor2.write(t1ug[gug]);
motor3.write(t2ug[gug]);
motor4.write(t1ug[gug]);
motor5.write(t2ug[gug]);
```

```
motor6.write(t1ug[gug]);
motor7.write(t2ug[gug-20]);
motor8.write(t1ug[gug-20]);
motor9.write(t2ug[gug-20]);
motor10.write(t1ug[gug-20]);
motor11.write(t2ug[gug-20]);
motor12.write(t1ug[gug-20]);
delay(20);

}
if(gug<20){
motor1.write(t2ug[gug]);
motor2.write(t1ug[gug]);
motor3.write(t2ug[gug]);
motor4.write(t1ug[gug]);
motor5.write(t2ug[gug]);
motor6.write(t1ug[gug]);
motor7.write(t2ug[gug+20]);
motor8.write(t1ug[gug+20]);
motor9.write(t2ug[gug+20]);
motor10.write(t1ug[gug+20]);
motor11.write(t2ug[gug+20]);
motor12.write(t1ug[gug+20]);
delay(20);
 }


}

  }
```

```
 //square
   if(state=='W'){


for(gk = 0; gk<60; gk++)
{









if(gk>=30){
motor1.write(t2k[gk]);
motor2.write(t1k[gk]);
motor3.write(t2k[gk]);
motor4.write(t1k[gk]);
motor5.write(t2k[gk]);
motor6.write(t1k[gk]);
motor7.write(t2k[gk-30]);
motor8.write(t1k[gk-30]);
motor9.write(t2k[gk-30]);
motor10.write(t1k[gk-30]);
motor11.write(t2k[gk-30]);
motor12.write(t1k[gk-30]);
delay(20);

}
if(gk<30){
motor1.write(t2k[gk]);
motor2.write(t1k[gk]);
```

```
motor3.write(t2k[gk]);
motor4.write(t1k[gk]);
motor5.write(t2k[gk]);
motor6.write(t1k[gk]);
motor7.write(t2k[gk+30]);
motor8.write(t1k[gk+30]);
motor9.write(t2k[gk+30]);
motor10.write(t1k[gk+30]);
motor11.write(t2k[gk+30]);
motor12.write(t1k[gk+30]);
delay(20);
 }


}

}



//square backwards
if(state=='U')
{

for(gkg = 0; gkg<60; gkg++)
{
```

```
if(gkg>=30){
motor1.write(t2kg[gkg]);
motor2.write(t1kg[gkg]);
motor3.write(t2kg[gkg]);
motor4.write(t1kg[gkg]);
motor5.write(t2kg[gkg]);
motor6.write(t1kg[gkg]);
motor7.write(t2kg[gkg-30]);
motor8.write(t1kg[gkg-30]);
motor9.write(t2kg[gkg-30]);
motor10.write(t1kg[gkg-30]);
motor11.write(t2kg[gkg-30]);
motor12.write(t1kg[gkg-30]);
delay(20);

}
if(gkg<30){
motor1.write(t2kg[gkg]);
motor2.write(t1kg[gkg]);
motor3.write(t2kg[gkg]);
motor4.write(t1kg[gkg]);
motor5.write(t2kg[gkg]);
motor6.write(t1kg[gkg]);
motor7.write(t2kg[gkg+30]);
motor8.write(t1kg[gkg+30]);
motor9.write(t2kg[gkg+30]);
motor10.write(t1kg[gkg+30]);
motor11.write(t2kg[gkg+30]);
motor12.write(t1kg[gkg+30]);
delay(20);
 }
```

```
}
 }

//Smooth

  if (state == 'F')
  {
   for(go = 0; go<80; go++)
{




if(go>=40){
motor1.write(t2o[go]);
motor2.write(t1o[go]);
motor3.write(t2o[go]);
motor4.write(t1o[go]);
motor5.write(t2o[go]);
motor6.write(t1o[go]);
motor7.write(t2o[go-40]);
motor8.write(t1o[go-40]);
motor9.write(t2o[go-40]);
motor10.write(t1o[go-40]);
motor11.write(t2o[go-40]);
motor12.write(t1o[go-40]);
delay(20);
```

```
}
if(go<40){
motor1.write(t2o[go]);
motor2.write(t1o[go]);
motor3.write(t2o[go]);
motor4.write(t1o[go]);
motor5.write(t2o[go]);
motor6.write(t1o[go]);
motor7.write(t2o[go+40]);
motor8.write(t1o[go+40]);
motor9.write(t2o[go+40]);
motor10.write(t1o[go+40]);
motor11.write(t2o[go+40]);
motor12.write(t1o[go+40]);
delay(20);
 }


} }
//Smooth backwards
if(state==’B’)
{
  for(gog = 0; gog<80; gog++)
{


if(gog>=40){
motor1.write(t2og[gog]);
motor2.write(t1og[gog]);
motor3.write(t2og[gog]);
motor4.write(t1og[gog]);
```

```
motor5.write(t2og[gog]);
motor6.write(t1og[gog]);
motor7.write(t2og[gog-40]);
motor8.write(t1og[gog-40]);
motor9.write(t2og[gog-40]);
motor10.write(t1og[gog-40]);
motor11.write(t2og[gog-40]);
motor12.write(t1og[gog-40]);
delay(20);

}
if(gog<40){
motor1.write(t2og[gog]);
motor2.write(t1og[gog]);
motor3.write(t2og[gog]);
motor4.write(t1og[gog]);
motor5.write(t2og[gog]);
motor6.write(t1og[gog]);
motor7.write(t2og[gog+40]);
motor8.write(t1og[gog+40]);
motor9.write(t2og[gog+40]);
motor10.write(t1og[gog+40]);
motor11.write(t2og[gog+40]);
motor12.write(t1og[gog+40]);
delay(20);
 }


}

  }
```

```
  if (state == 'S'){
motor1.write(23);
motor2.write(23);
motor3.write(23);
motor4.write(23);
motor5.write(23);
motor6.write(23);
motor7.write(23);
motor8.write(23);
motor9.write(23);
motor10.write(23);
motor11.write(23);
motor12.write(23);
   }



}
void inverse(float PX, float PY)
{
 s=sqrt(PX*PX+PY*PY);
 a=sqrt((a2*a2)-(pow(((a1*a1-a3*a3+s*s)/(2*s)),2)));
 b=sqrt((a4*a4)-(pow(((s*s-a1*a1+a3*a3)/(2*s)),2)));
 k=((a1*a1-a3*a3+s*s)/(2*s));
 m=((s*s-a1*a1+a3*a3)/(2*s));
 ac=atan2(PY,PX);
 out1=((ac+asin(b/a1))*180/pi);
 out2=((ac-asin(a/a2))*180/pi);
```

```
}
```

B. APPENDIX

SOFTWARE OF PIPELINEBOT

```
float s,m,b,a,k,Px,Py,ac;
const int a1=50;
const int a2=50;
const int a3=60;
const int a4=60;
float out1,out2;
const float pi=3.14159;
//Forward
float teta11k,teta12k;
 int hk=0;
 float teta1k,teta2k;
 int gk;
float P0xk[5]={12.5,12.5,-12.5,-12.5,12.5};
float P0yk[5]={-72.5,-47.5,-47.5,-72.5,-72.5};
float P1xk[60];
float P1yk[60];
float t1k[60];
float t2k[60];
int ik,jk;

//Backward
float teta11kg,teta12kg;

 int hkg=0;
 float teta1kg,teta2kg;
 int gkg;

float P0xkg[5]={-12.5,-12.5,12.5,12.5,-12.5};
```

```
float P0ykg[5]={-72.5,-47.5,-47.5,-72.5,-72.5};
float P1xkg[60];
float P1ykg[60];
float t1kg[60];
float t2kg[60];
int ikg,jkg;



#include<Servo.h>
Servo motor1;
Servo motor2;
Servo motor3;
Servo motor4;
Servo motor5;
Servo motor6;
Servo motor7;
Servo motor8;
Servo motor9;
Servo motor10;
Servo motor11;
Servo motor12;
Servo motor13;
Servo motor14;
 int state;

void setup() {
   motor1.attach(2);
   motor2.attach(3);
   motor3.attach(4);
   motor4.attach(5);
```

```
   motor5.attach(6);
   motor6.attach(7);
   motor7.attach(8);
   motor8.attach(9);
   motor9.attach(10);
   motor10.attach(11);
   motor11.attach(12);
   motor12.attach(13);
   motor13.attach(44);
   motor14.attach(45);

Serial.begin(9600);
//Forward
for(jk = 0; jk<3; jk++)
{
 for(ik = 0; ik<10; ik++)
  {
   P1xk[hk]=P0xk[jk]+(((P0xk[jk+1]-P0xk[jk]))/10*ik);
   P1yk[hk]=P0yk[jk]+(((P0yk[jk+1]-P0yk[jk]))/10*ik);
   hk=hk+1;
 }
 }
 for(int yk=0;yk<30;yk++)
 { P1xk[hk]=P0xk[3]+(((P0xk[4]-P0xk[3]))/30*yk);
   P1yk[hk]=P0yk[3]+(((P0yk[4]-P0yk[3]))/30*yk);
  hk=hk+1;

  }

 for(gk = 0; gk<60; gk++)
{
```

```
   inverse(P1xk[gk],P1yk[gk]);
   teta1k=out1;
   teta2k=out2;
if(teta1k<0)
{
  teta1k=360+teta1k;
  }
  if(teta2k<0)
{
  teta2k=360+teta2k;

  }

if (teta1k>180)
{teta11k=360-teta1k+23;}
if (teta1k<180)
{teta11k=23-teta1k;}

if (teta2k>180)
{teta12k=teta2k-180+23;}
if (teta2k<=180)
{teta12k=23-(180-teta2k);
}
t1k[gk]=teta11k;
t2k[gk]=teta12k;
}
//geri
for(jkg = 0; jkg<3; jkg++)
{
 for(ikg = 0; ikg<10; ikg++)
  {
```

```
   P1xkg[hkg]=P0xkg[jkg]+(((P0xkg[jkg+1]-P0xkg[jkg]))/10*ikg);
   P1ykg[hkg]=P0ykg[jkg]+(((P0ykg[jkg+1]-P0ykg[jkg]))/10*ikg);
   hkg=hkg+1;
 }
 }
 for(int ykg=0;ykg<30;ykg++)
 { P1xkg[hkg]=P0xkg[3]+(((P0xkg[4]-P0xkg[3]))/30*ykg);
   P1ykg[hkg]=P0ykg[3]+(((P0ykg[4]-P0ykg[3]))/30*ykg);
  hkg=hkg+1;

  }

 for(gkg = 0; gkg<60; gkg++)
{
   inverse(P1xkg[gkg],P1ykg[gkg]);
   teta1kg=out1;
   teta2kg=out2;
if(teta1kg<0)
{
  teta1kg=360+teta1kg;
  }
  if(teta2kg<0)
{
  teta2kg=360+teta2kg;

  }

if (teta1kg>180)
{teta11kg=360-teta1kg+23;}
if (teta1kg<180)
{teta11kg=23-teta1kg;}
```

```
if (teta2kg>180)
{teta12kg=teta2kg-180+23;}
if (teta2kg<=180)
{teta12kg=23-(180-teta2kg);
}
t1kg[gkg]=teta11kg;
t2kg[gkg]=teta12kg;
}

    Serial.begin(9600);

 }
void loop() {

     if(Serial.available() > 0){
     state = Serial.read();
   }
   else{motor1.write(23);
motor2.write(23);
motor3.write(23);
motor4.write(23);
motor5.write(23);
motor6.write(23);
motor7.write(23);
motor8.write(23);
motor9.write(23);
motor10.write(23);
motor11.write(23);
motor12.write(23);}
```

```
 //Forward
   if(state=='W'){
 motor13.write(90);
  motor14.write(90);
for(gk = 0; gk<60; gk++)
{
if(gk>=30){
motor1.write(t2k[gk]);
motor2.write(t1k[gk]);
motor3.write(t2k[gk]);
motor4.write(t1k[gk]);
motor5.write(t2k[gk]);
motor6.write(t1k[gk]);
motor7.write(t2k[gk-30]);
motor8.write(t1k[gk-30]);
motor9.write(t2k[gk-30]);
motor10.write(t1k[gk-30]);
motor11.write(t2k[gk-30]);
motor12.write(t1k[gk-30]);
delay(20);
}
if(gk<30){
motor1.write(t2k[gk]);
motor2.write(t1k[gk]);
motor3.write(t2k[gk]);
motor4.write(t1k[gk]);
motor5.write(t2k[gk]);
motor6.write(t1k[gk]);
```

```
motor7.write(t2k[gk+30]);
motor8.write(t1k[gk+30]);
motor9.write(t2k[gk+30]);
motor10.write(t1k[gk+30]);
motor11.write(t2k[gk+30]);
motor12.write(t1k[gk+30]);
delay(20);
 }
}
}
//Backward
if(state=='B')
{
   motor13.write(90);
      motor14.write(90);
for(gkg = 0; gkg<60; gkg++)
{
if(gkg>=30){
motor1.write(t2kg[gkg]);
motor2.write(t1kg[gkg]);
motor3.write(t2kg[gkg]);
motor4.write(t1kg[gkg]);
motor5.write(t2kg[gkg]);
motor6.write(t1kg[gkg]);
motor7.write(t2kg[gkg-30]);
motor8.write(t1kg[gkg-30]);
motor9.write(t2kg[gkg-30]);
motor10.write(t1kg[gkg-30]);
motor11.write(t2kg[gkg-30]);
motor12.write(t1kg[gkg-30]);
delay(20);
```

```
}
if(gkg<30){
motor1.write(t2kg[gkg]);
motor2.write(t1kg[gkg]);
motor3.write(t2kg[gkg]);
motor4.write(t1kg[gkg]);
motor5.write(t2kg[gkg]);
motor6.write(t1kg[gkg]);
motor7.write(t2kg[gkg+30]);
motor8.write(t1kg[gkg+30]);
motor9.write(t2kg[gkg+30]);
motor10.write(t1kg[gkg+30]);
motor11.write(t2kg[gkg+30]);
motor12.write(t1kg[gkg+30]);
delay(20);
 }
}
  }
if (state == 'S'){
motor1.write(23);
motor2.write(23);
motor3.write(23);
motor4.write(23);
motor5.write(23);
motor6.write(23);
motor7.write(23);
motor8.write(23);
motor9.write(23);
motor10.write(23);
motor11.write(23);
```

```
motor12.write(23);
motor13.write(90);
motor14.write(90);
   }
   if(state == 'R')
   {for(int r=90;r++;r<180)
   {
     motor13.write(r);
      if(state != 'R')
      {
       r=180;
       }
     }

     }
       if(state == 'L')
   {for(int l=90;l--;l<0)
   {
     motor13.write(l);
      if(state != 'L')
      {
       l=0;
       }
     }

     }

     if(state == 'U')
   {for(int u=90;u++;u<180)
   {
     motor14.write(u);
```

```
      if(state != 'U')
      {
       u=180;
       }
     }

     }
      if(state == 'X')
   {for(int q=90;q--;q<0)
   {
     motor14.write(q);
      if(state != 'X')
      {
       q=0;
       }
     }

     }
}
void inverse(float PX, float PY)// Inverse kinematic analysis to find end effector points.
{
  s=sqrt(PX*PX+PY*PY);
  a=sqrt((a2*a2)-(pow(((a1*a1-a3*a3+s*s)/(2*s)),2)));
  b=sqrt((a4*a4)-(pow(((s*s-a1*a1+a3*a3)/(2*s)),2)));
  k=((a1*a1-a3*a3+s*s)/(2*s));
  m=((s*s-a1*a1+a3*a3)/(2*s));
  ac=atan2(PY,PX);
  out1=((ac+asin(b/a1))*180/pi);
  out2=((ac-asin(a/a2))*180/pi);
}
```