T.C.
BAHÇEŞEHİR UNIVERSITY

FACULTY OF ENGINEERING AND NATURAL SCIENCES

CAPSTONE FINAL REPORT

AUTONOMOUS WARRIOR SUMO ROBOT WITH IMAGE PROCESSING

Students: Canberk Bagana MCH
Ardahan Dikmedaş CMP
Yunus Elçi CMP
Ömer Yiğit Gevrek MCH
Adem Arda Günay MCH
Berkay Emre Yoran CMP

Advisors: Doç. Dr. Yalçın ÇEKİÇ
Assist. Prof. Tarkan Aydın

ISTANBUL, June 2022

STUDENT DECLARATION

By submitting this report, as partial fulfillment of the requirements of the Capstone course, the students promise on penalty of failure of the course that

- they have given credit to and declared (by citation), any work that is not their own (e.g. parts of the report that is copied/pasted from the Internet, design or construction performed by another person, etc.);
- they have not received unpermitted aid for the project design, construction, report or presentation;
- they have not falsely assigned credit for work to another student in the group, and not take credit for work done by another student in the group.

ABSTRACT

AUTONOMOUS WARRIOR SUMO ROBOT WITH IMAGE PROCESSING

Students: Canberk Bagana MCH

Ardahan Dikmedaş CMP

Yunus Elçi CMP

Ömer Yiğit Gevrek MCH

Adem Arda Günay MCH

Berkay Emre Yoran CMP

Faculty of Engineering and Natural Sciences

Advisors: Doç. Dr. Yalçın ÇEKİÇ

Assist. Prof. Tarkan Aydın

June 2022

In this project, our team's responsibility is to design a warrior sumo robot. The goal of the project is to build an autonomous warrior sumo robot with image processing to win the competition against the opponents without harming allies in a dohyo. With a camera attached to the designed robot, it is automatically determined whether it is an enemy or friend, depending on the shapes and colors on the objects, and the coordinates of the enemies are transmitted to the microcontrollers. According to the incoming coordinates, while the robot tries to keep the friends on the dohyo, it tries to push the enemies out of the dohyo.

Key Words: Sumo robot, raspberry pi, sensors, image processing , mdf , camera,motor mount,dc motor

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ML	Machine Learning
OD	Object Detection
IP	Image Processing
TL	Turkish Lira
STP	Stepper Motor
CMP	Computer Engineering
RASP	Raspberry Pi

1. OVERVIEW

1.1. Identification of the need

In today's world, visual processing is used in subjects such as autonomous driving and digital camera tracking. Visual processing is important to switch from high-cost human resource to low-cost machine resource. Machines that will replace many professions in the world of the future will use visual processing and machine learning techniques. The aim of the team is to make a sum robot based on visual processing, which is the technology of the future, and to present a demo about visual processing.

1.2. Definition of the problem

In this project, there are enemies that we do not want on dohyo. At the same time, there are friendly objects that we can define on dohyo. The problem is to keep the friendly objects on dohyo and push the enemy objects out of the dohyo.

1.2.1. Functional requirements

The robot must detect objects using image processing. Accordingly, objects should be detected with a camera attached to the robot or which will monitor the dohyo from the outside, and the coordinates of the unwanted objects should be sent to the robot.

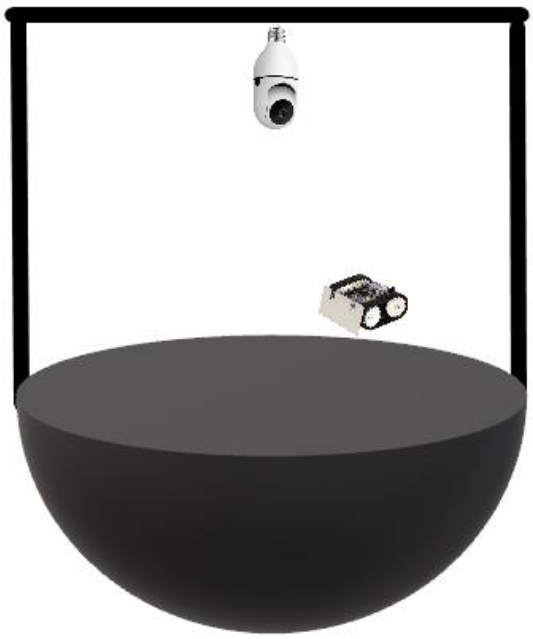
1.3. Conceptual solutions

In this section, some possible system design solutions will be evaluated.

1.3.1 CMP Conceptual Solutions

Concept 1:

In this solution, monitoring the dohyo with an external camera and detecting enemy objects, then transmitting the coordinates to the sumo robot.



The sumo robot, which is on this dohyo with a radius of 50 cm, receives the coordinates of the enemy objects from the camera connected to it from above with a cable.

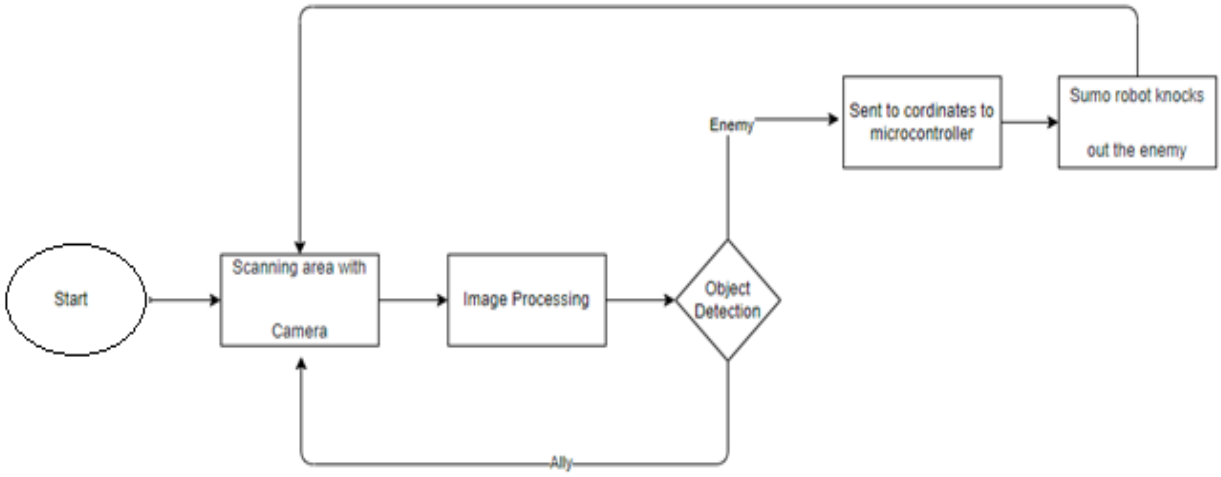


Figure 1. CMP Conpept 1 Algorithm Chart.

1.3.1.1 CMP Conceptual Solution 1 Potential Problems

Problem 1:

When we wanted to implement this concept, we had to wirelessly communicate the arduino inside the robot and the raspberry pi computer connected to the camera, since the camera had to watch the robot from above.

Before looking at the necessary bluetooth devices for this, we tried serial communication between raspberry pi and arduino. Even wired communication lagged far behind image processing. We abandoned this concept because the resulting slowness was too much.

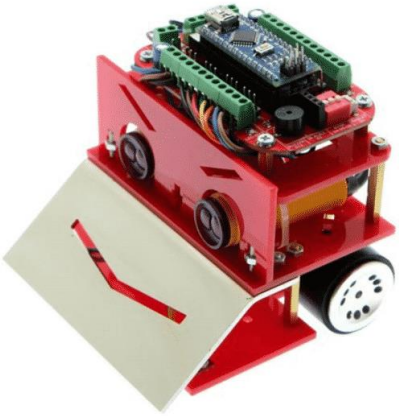
Problem 2:

This concept had to include a route planning algorithm. Otherwise, when the robot tries to advance to the enemy object, it will also crash into the friendly objects between the enemy object and the robot, causing undesirable results.

We can say that even our image processing algorithm slows down our raspberry pi device a lot. Running an extra route planning algorithm on it would cause the robot to wait for minutes.

Concept 2:

In this solution, the robot detects the location of the enemies with the camera on it and tries to throw the enemy objects out. Afterwards, if there is no enemy object in sight, it slowly turns around and looks for a new enemy object, and when it finds it, dohyo makes a move to throw the enemy object outside. By repeating this cycle, it will autonomously remove the enemy objects inside the dohyo.



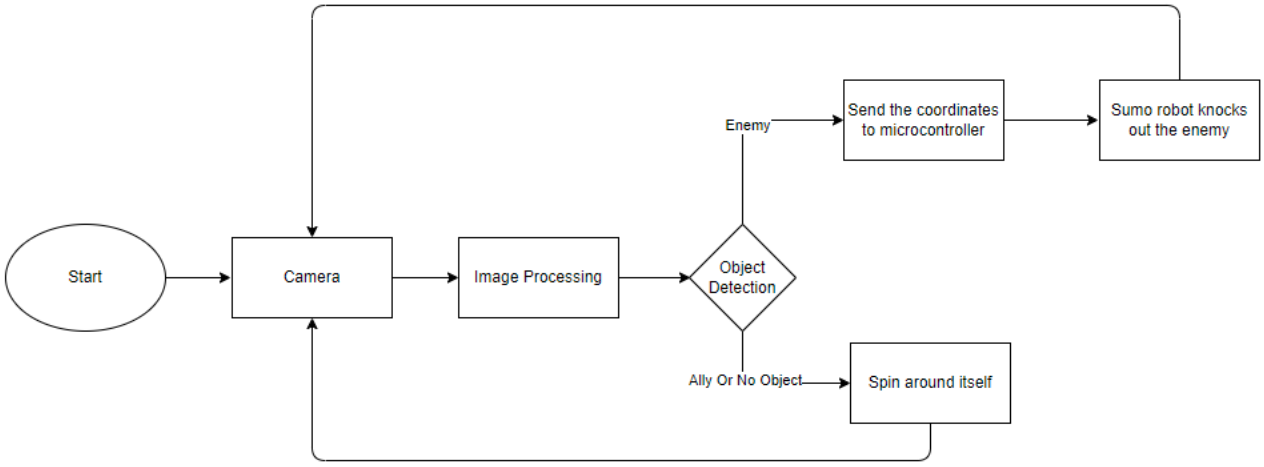


Figure 2. CMP Concept 2 Algorithm Chart.

1.3.1.2 CMP Conceptual Solution 2 Potential Problems And Solutions

Problem 1:

Since the camera will always be fixed on the robot, the image processing should work properly even though the robot's movements shake the camera. Since we do not have a very powerful computer, the code running on the raspberry pi will not be able to process such a rapidly changing image that fast.

Solution 1:

To solve this problem, we chose to give the raspberry pi some time to process the image. To do this, we decided to restrict the robot's movements.

Although we tried to slow down the robot before restricting its movements, the L298N motors we had did not respond to slow operation according to the power provided by our battery. That's why we gave the robot a high and a low power over python, so that it can move limitedly.

Code Example:

```
if state == 'scan':
    print 'scan'
    p.ChangeDutyCycle(43)
    p2.ChangeDutyCycle(43)
    GPIO.output(in1, GPIO.HIGH)
    GPIO.output(in2, GPIO.LOW)
    GPIO.output(in3, GPIO.LOW)
    GPIO.output(in4, GPIO.HIGH)
    p.ChangeDutyCycle(20)
    p2.ChangeDutyCycle(20)
    state = 'z'
```

Here, our p and p2 values represent motor 1 and motor 2. With the ChangeDutyCycle() method, we give high power at first, but then slow it down and make it move piece by piece.

Problem 2:

Let's assume that our robot takes action to find the enemy object and throw it out of the dohyo. In this case, the robot will see that the center of the enemy object, which it has found and determined before, changes as it approaches the enemy object. In this context, before the robot touches the object, it will make a right or left orientation according to the determining shape on the object.

Solution 2:

Our robot, which sees the object from afar and determines it as an enemy, finds the center of the object and aligns itself with this center before moving towards the object. It has a sensitivity value when performing this operation. We increase this sensitivity value according to the increasing area of the object as we approach the object, and we tolerate the margin of error formed by the sliding center of the object. In this way, as the robot approaches the object, it successfully pushes the enemy object out of the dohyo, even if it moves away from the center of the previously determined object.

Problem 3 :

As Computer Engineering students, we wanted to design the raspberry pi device as the decision brain and the arduino uno device as the action brain for this solution. When we wanted to make this design, we received the positions of the enemy information that we sent over the raspberry pi too late via the arduino uno device.

In this case, we were faced with fast image processing and slow engine movements. Serial communication between Raspberry pi and arduino was so slow that we couldn't get any feedback from the engines.

Solution 3 :

For this problem, we first tried an optimization such as slowing down the image processing. But in this case, we would see a robot image without taking action on dohyo for minutes. We did not want this situation, so we removed the arduino uno device. Now our Raspberry Pi 4 device is the main brain of the robot. We designed it as the only robot brain that manages the motor driver, camera, cooling fans and image processing from the ports on it.

Comparison of the Concepts:

Table 1:

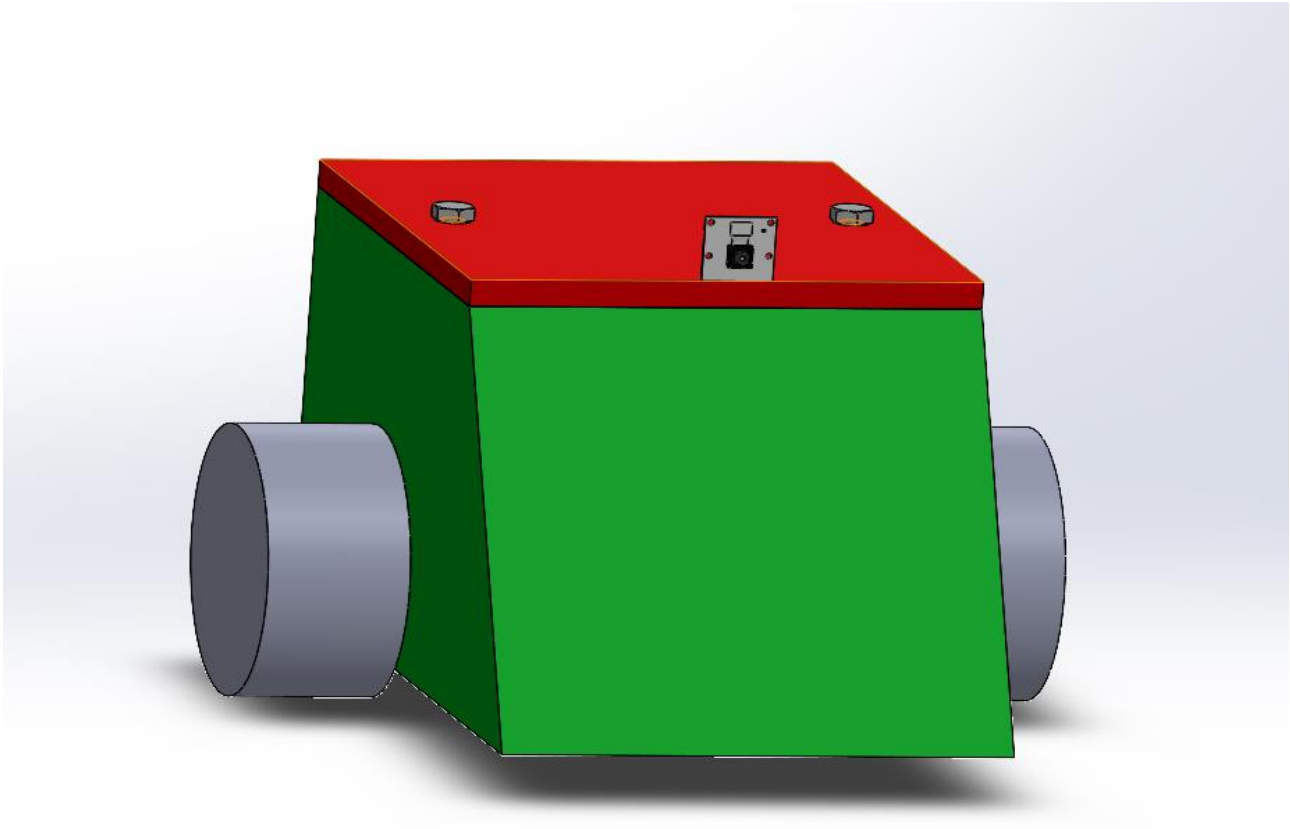
Category/Concept	Concept 1	Concept 2
Accuracy	+ + + + +	+ +
Cost	+ + +	+ + + +
Complexity	+	+ + + +
Overall	9	10

As a result of the general evaluation, it was decided to make the second concept.

1.3.2 MCH Conceptual Solutions

Concept 1: In this solution, Sumo robot will be controlled with Raspberry Pi 4. Two DC motors are connected to it.Data will be received from CMP will lead robot to its enemies.

Robot's Design will act like natural bulldoser that will throw enemies out the dohyo.



Concept 2: In this solution robot will use extra one subsystem that will push enemy. Robot will use its bulldoser to push enemy and only Raspberry Pi system will be implemented to the robot with two DC motors.

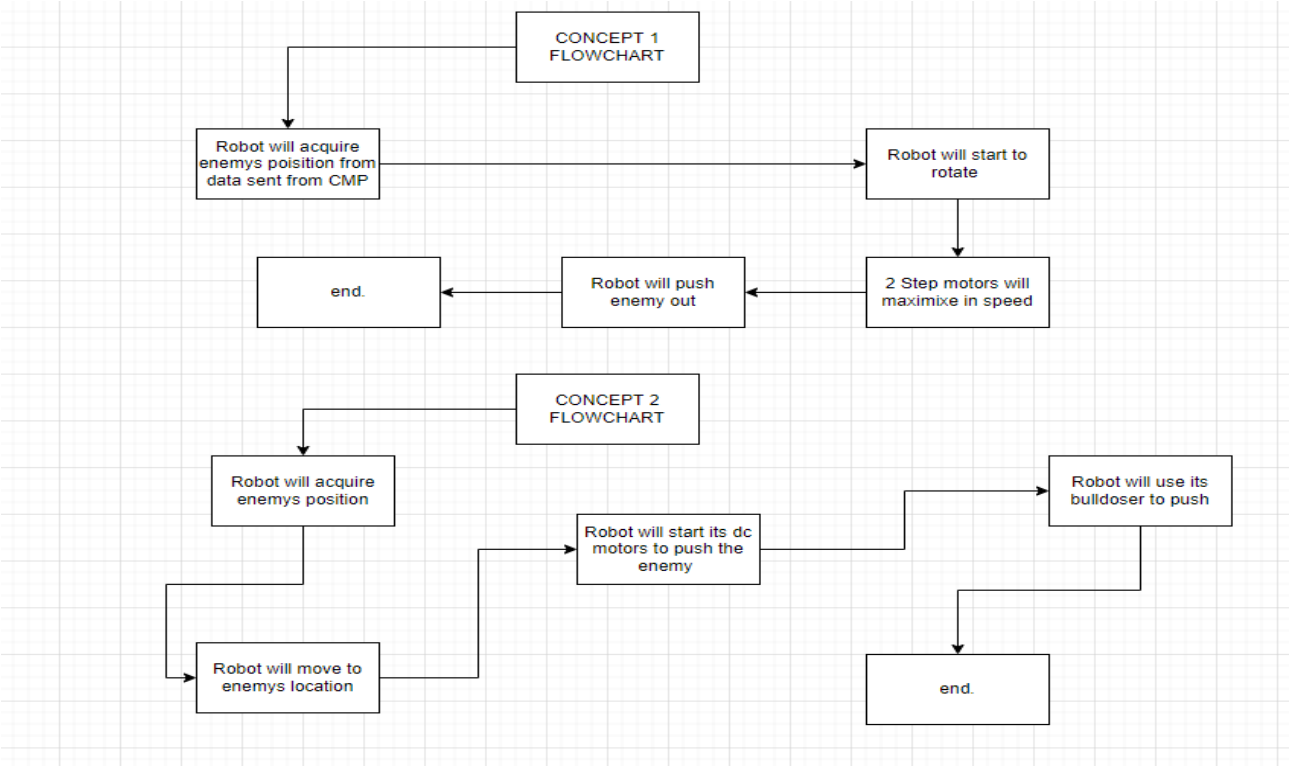


Figure 3. MCH Concept 1-2 Flowcharts.

Comparison of the Concepts:

As for mechatronics parts, we decided on concept one which is more reliable and faster solution.

Category/Concept	Concept 1	Concept 2
Accuracy	+ + +	+ +
Cost	+ + +	+ + + +
Complexity	+ +	+ + + +
Overall	8	6

1.3.3 MCH Requirement Analysis

Performance analysis

In order to stabilize performance during action, batteries, camera with image process and motors will be tested, then optimized for proper work.

Reliability analysis

Our final product will be durable enough to perform given tasks and will last as long as battery power remains.

1.3.4 CMP Requirement Analysis

As a computer engineer group, we will enable the robot to process images with the camera in the project. We will take an image from the camera and process it so that he can make a decision.

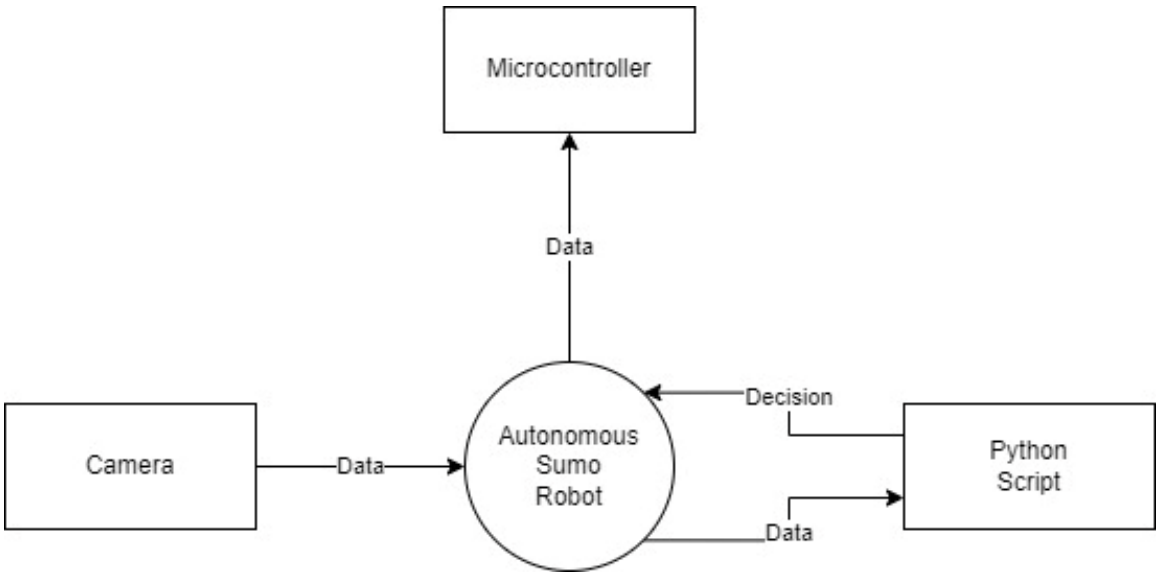


Figure 4. CMP Requirement Analysis.

1.4. Physical architecture

Basically, the system is comprised of 5 elements. First and most important unit is microcontroller, which is Raspberry Pi in this system. It is actuated by Li-po battery as power output. Remained units are directly or indirectly linked to Raspberry Pi.

There are 2 step motor units responsible from motion of the robot. Both motors are connected to the motor driver which is also connected to microcontoller.

According to the concept solution, there can be a camera on the robot or on the dohyo. The robot will move autonomously by processing images with this camera.

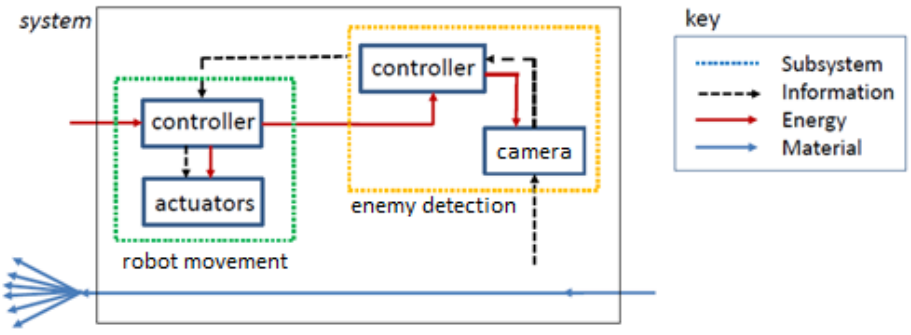


Figure 5. Interface diagram for the system

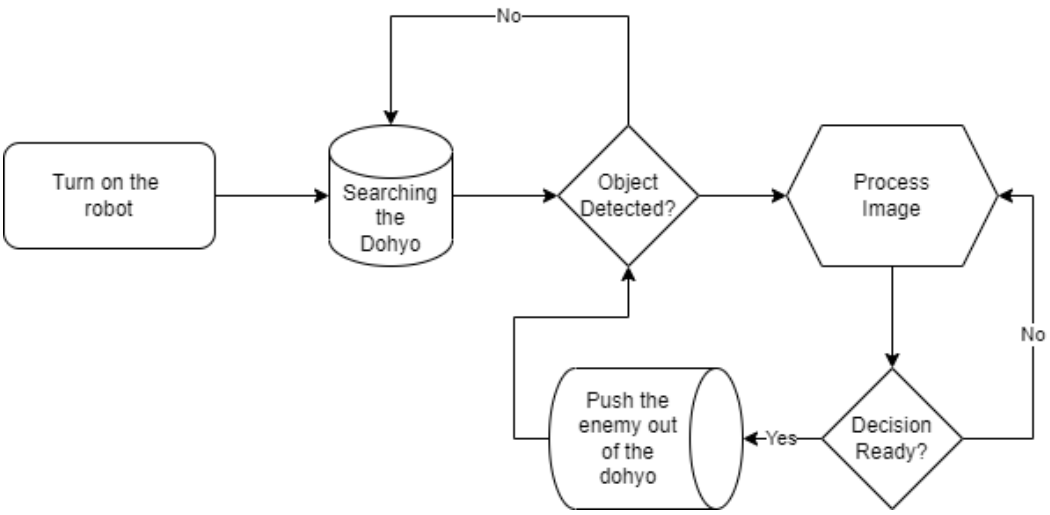


Figure 6. Process chart for the system

1.4.1 Mechanical Architecture

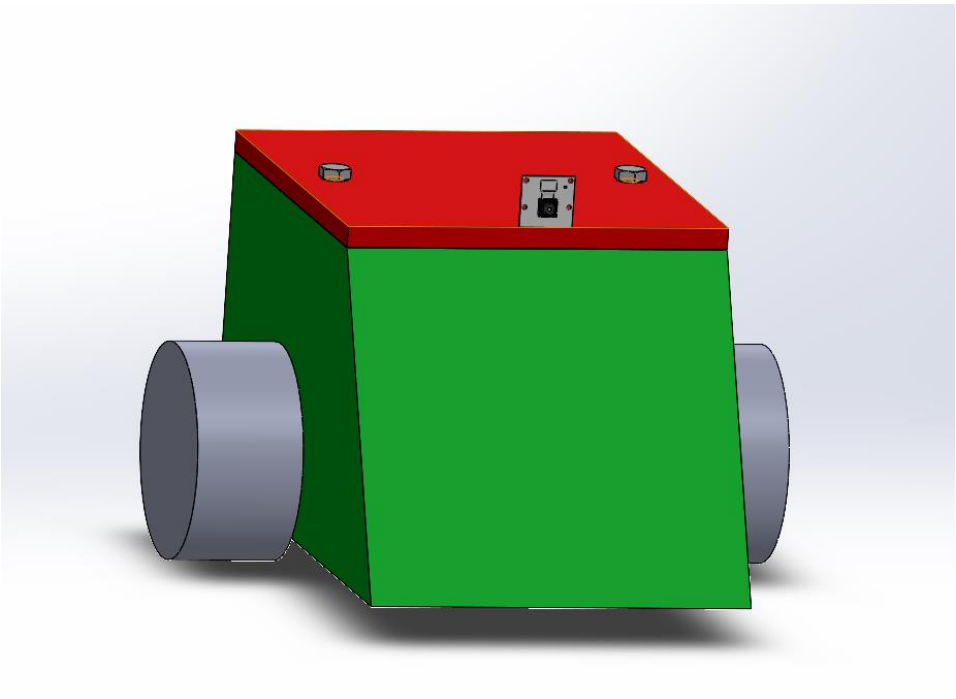


Figure 7. Mechanical Architecture

We have changed our design due to various factors. Main reason for changing our design is the pricing reasons. Although the body stayed same, there is only change in the front part.

2. WORK PLAN

In the first semester of this project, our goal is to complete theoretical part of the robot and create a draft for design.

In the second semester, robot will be produced.

2.1. Work Breakdown Structure (WBS)

As for MCH part, our process will be divided into two separate subsystems: Mechanical and electrical.

Mechanical part will cover design analyis and power harness sections.

Electrical part will cover design of electronic system and assembling motors and wiring connections.

As for CMP part, our process is to locate the enemies according to the data from the camera and send the coordinates of the enemies to the microcontrollers.

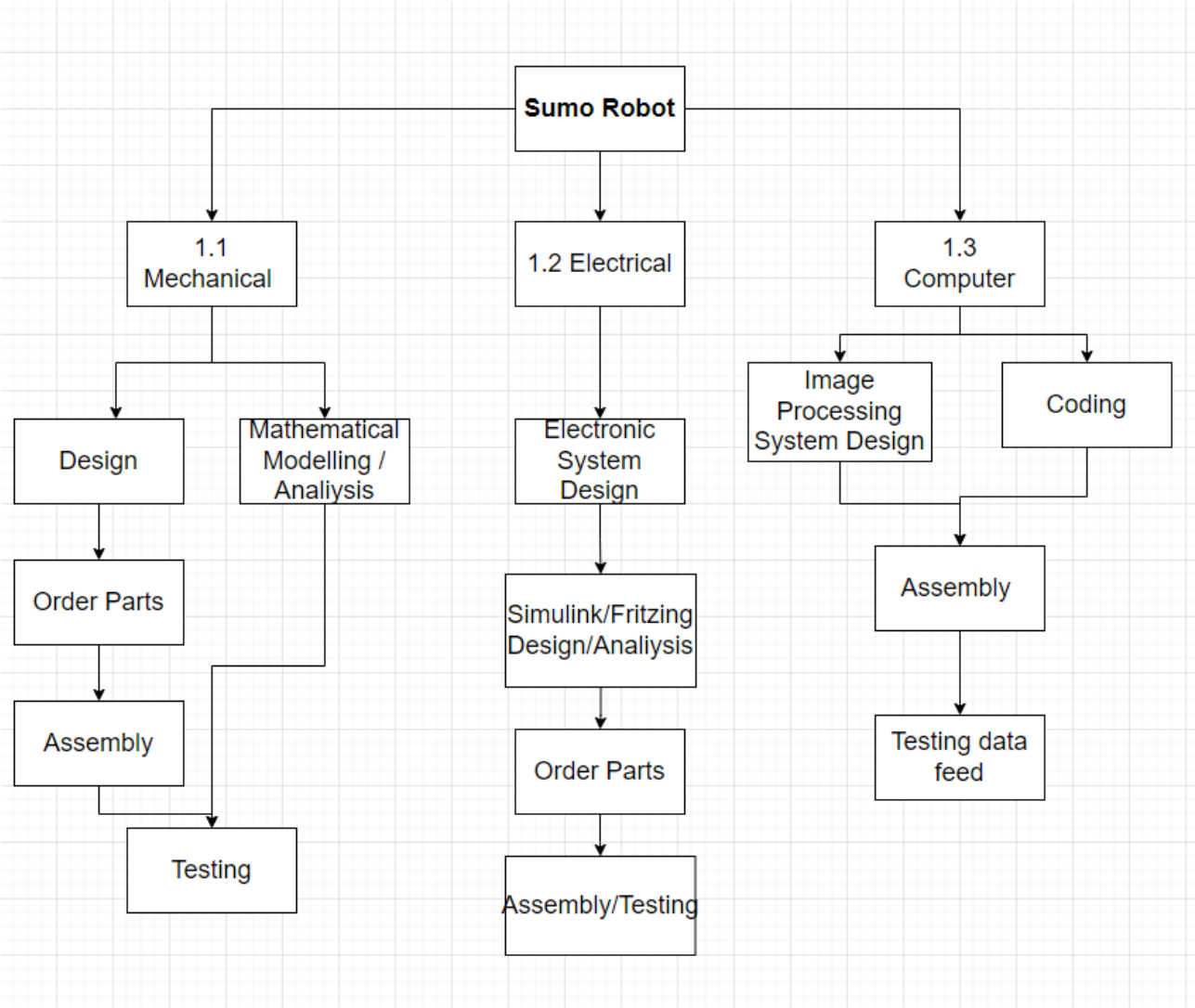


Figure 8. Work breakdown structure for the project.

2.2. Responsibility Matrix (RM)

Each team member with his support and responsibility of each objective is listed below.

Table 2. Responsibility Matrix for the team

Task	Canberk	Ardahan	Yunus	Yiğit	Arda	Berkay
Mechanic.	R			S	S	
Electrical	S			R	S	
Comm.	S	R	S	S	S	S
Control	S	S	R	S	S	S
Planning	S	S	S	S	S	R
Reporting	S	S	S	S	R	S
Integratio	R	S	S	S	S	S

R= Responsible S=Support

2.3. Project Network (PN)

After the production and with the arrival of nessesary parts our road map is shown below.

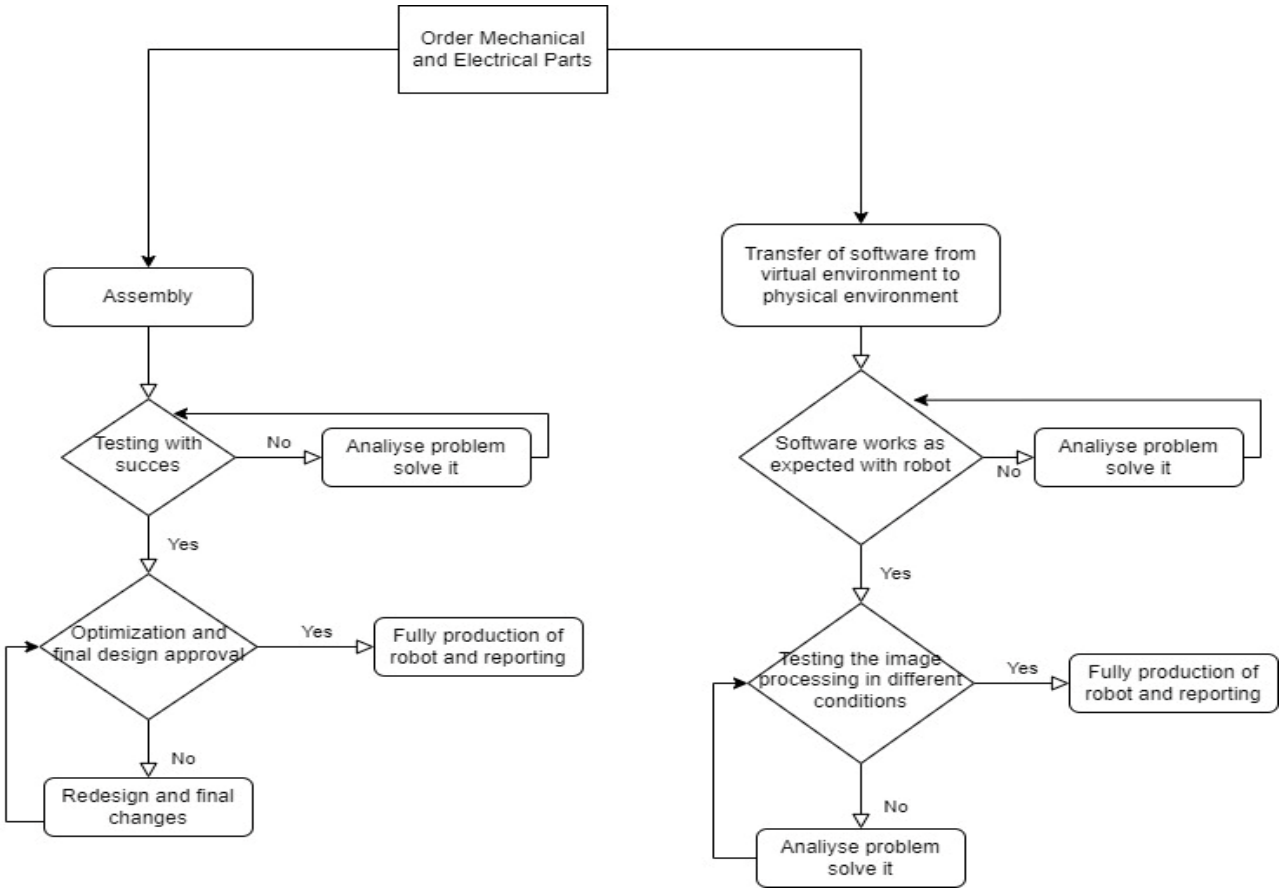


Figure 9. The project network

2.4. Gantt chart

As from the nature of the project it needed a detailed workplan which is our Gantt chart.

Each subsystems roadmap can be seen below at table 3.

#	TASK	EFFORT
1	Order both mechanical and electronical parts	1 Week
2	Assembly of parts	2 Weeks
3	Testing and integration of parts	1 Week
4	System integration of all subsystems	2 Weeks
5	Researching the software architecture to be used	1 Week
6	Creation of image processing infrastructure with camera	3 Weeks
7	Simulating the software in virtual environment	2 Weeks
8	Testing the software on the robot	2 Weeks
9	Reporting, analyzing feedbacks and redesign	Until 12. week
10	Draft report	7-8 Weeks
11	Final design and report	3-4 Weeks
12	Project exhibition	Last week

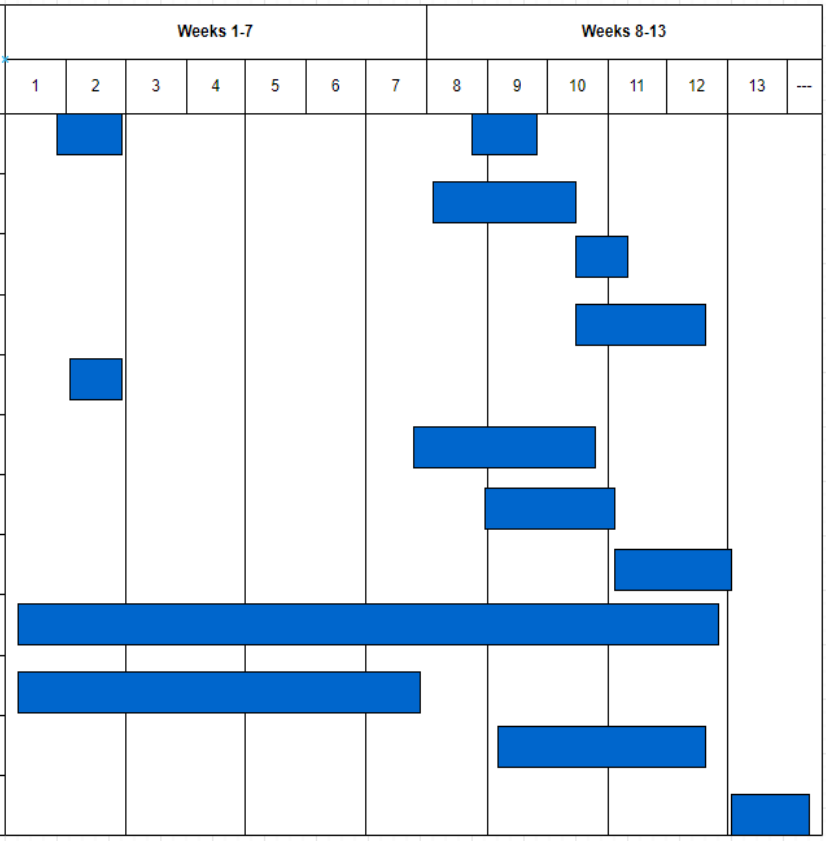


Table 3. Gantt chart for the materialisation phase of the project.

2.5. Cost

Listed below, there are several components with their prices in online market. Price may change depending on currency etc.

Table 4. Costs

ELECTRO-MECHANICAL			
Product name	**Qua.**	**Cost**	**Total cost**
Motor Driver	1	৳65,00	৳65,00
DC Motor	2	৳370,00	৳740,00
Wheels	1	৳150,00	৳150,00
Li-Po Battery+Charger	1	৳1200,00	৳1200,00
Body	1	৳250,00	৳250,00
Other Materials	1	৳200,00	৳200,00
			৳2605,00

COMMUNICATIONS AND CONTROL			
Product Name	**Qua.**	**Cost**	**Total Cost**
Raspberry Pi 4	1	৳3200,00	৳3200,00
Webcam	1	৳200,00	৳200,00
			৳3400,00
		System Total	**৳6005,00**

2.6. Risk assessment

Many mechanical and electronic components have their own risks of not working properly or fail. Most of the possible failures need to be determined. Solving plan of action in case of these scenarios is important process of testing and manufacturing this project.

Below there is a table of possible errors with their properties and plan of action.

Error event	Probability	Severity	Risk Level	Plan of action
Microprocessor Failure	Unlikely, Raspberry pi is reliable	Major Whole robot will shutdown	Medium	Have to check the code or cables
Battery Problem	Very Unlikely, batteries are on low percentage or inappropriate to complete the circuit	Major Robot will unable to move or move too slow.	Medium	Proper and full charged batteries are replaced
COVID 19 Lockdown	Likely, there will be limited access to school and work together physically	Moderate Integrating, building and testing the system will be difficult	High	Virtual system simulations may needed.Also proper internet and effective working tools will be required at home
Robot Upside Down	Very Unlikely, If robot's step motors rpm is too strong compared to its size and weight. Current estimates are well balanced	Moderate Robot will unable to move or turn	High	Balanced weight distribution and proper part placement will fix the problem
Expensive Manufacturing	Likely, Current calculations on expenses may change due to currency	Minor Manufacturing components material may differ or cost more	Medium	Searching for cheaper product that will make same work as a component.or material

Table 5. Risk assesment

3. SUB-SYSTEMS

Comparing different technologies that fall into the same category helps by giving the team options. An abrasion reading was performed so the team could get an idea of which parts and sub-parts could be used for the robots.

3.1. Processors for autonomous robots

The processor or microcontroller is basically the brain of a robot. For an autonomous robot, the processor receives signals from competitors' lines and detectors and sends them to the motor controller for acceleration. The reason why the microcontroller and the motor driver co exist in robots is that the microcontroller does not have sufficient output power. And the motor controller cannot decide the appropriate speed for the motors. In conclusion, the importance of a microcontroller in a robot is to calculate the appropriate speed for motors in certain situations and send and receive signals.

3.1.1. Requirements

Requirements for electronic subsystem is to designing and simulating complete circuit for the theoretical part.

After designing electronic system, Practical part of the requirements are ordering parts and then assembly them properly for the software test.

Finally, with the coordination of mechanical system, appropriately sized parts will be placed into the robot.

Technical requirements

Main components required in this system are shown on the table.

Microprocessor	**Motor Driver**	**Motor**	**Power Supply**
Raspberry Pi 4	L298N	DC Motor	Li-Po Battery

3.1.2. Technologies and methods

The processed image will be transferred to the Raspberry pi, then the Raspberry pi will transmit the incoming information to the motor driver and give a command, then the driver will power the motors with the help of the battery and the movement will begin.

This event will be constantly refreshed depending on the processed image and new commands will be given. When the detected enemy is detected, the sumo robot will move towards the enemy and push it down from the dohyo, and the image processing will continue until it finds a new target.

Battery Management

In this system, there will be single Li-Po battery (rechargeable) to give power for both Raspberry Pi 4 and to set in motion step motors.

3.1.3. Conceptualization

Designed and selected component diagram concept within the system are shown below:

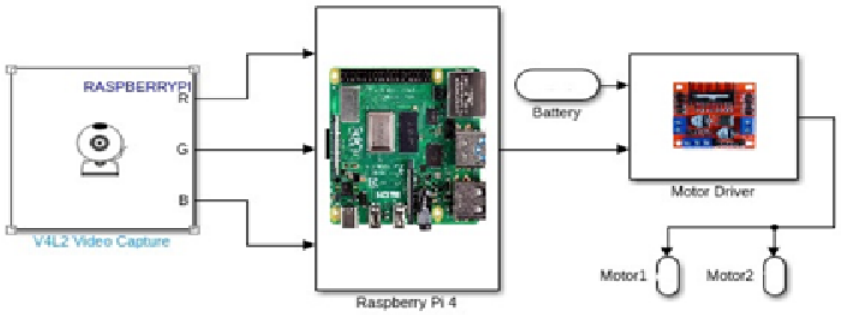


Figure 10. Conceptualization

3.1.4. Physical architecture

DC motors are connected indirectly to Raspberry Pi via motor driver.Both motors and Raspberry Pi is powered up by Li-Po battery.

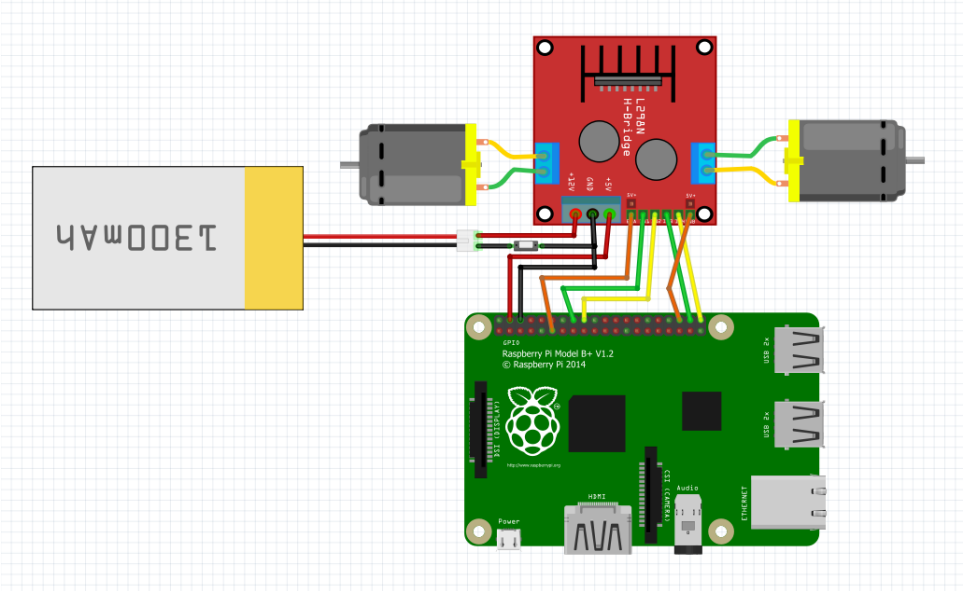


Figure 11. Materialization

3.1.5. Materialization

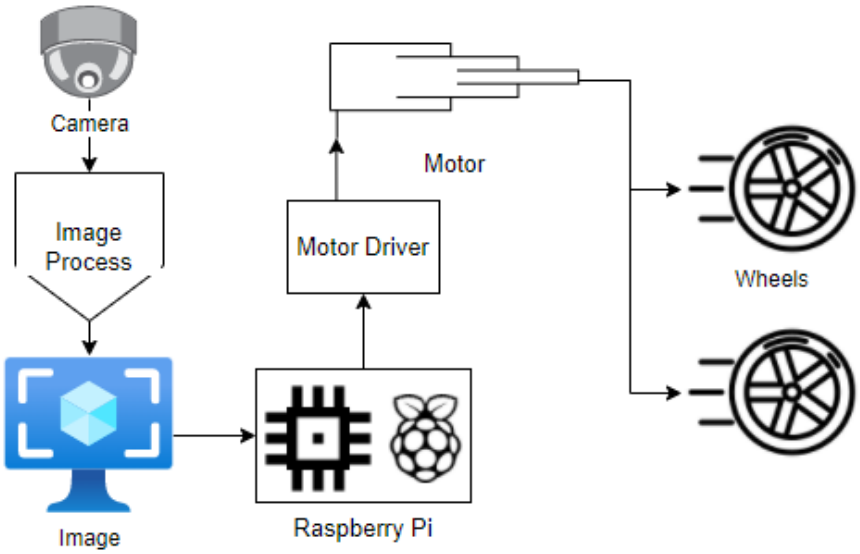


Figure 12. Materialization 2

3.2. Mechanical Subsystem

Mechanical system will be composed of two parts:

1) Mdf Robot Design
2) Selecting appropriate motor supports and wheels,
3) Mathematical Modelling
4) Simulation

3.2.1. Requirements

Requirements of mechanical subsystem is to determine the design of the robot and than select appropriate motor supports and their locations after determining positions for selected wheels and their shafts.

For the theoretical part we needed to do mathematical modelling and simulations about our robot. We achieved this by calculating loads and determining the governing equations. After that we done our simulations in solidworks due to accessibility to our design.

For the production part we agreed with a cnc shop that could build our chasis and then we fixed our motor mounts and any electrical components to their desired location.

3.2.2. Technologies and methods

There will be two wheels connected to DC motor which is controlled by Raspberry Pi 4.

Mdf outer cover dimensions will be given in both Solidworks and Autocad all dimensions are in MM in Solidworks and CM in Autocad

Bolts, nuts will be applied on the body to connect parts and to fix positions of electronical components.

Outer part will be produced from Mdf by using laser cutting technology and CNC techniques.

Whole body will be designed with Solidworks and Autocad.

Simulations will be done in two parts. Heat simulation to analyse effects of Li-Po and Raspberry's heat to body.

Static analysis to determine loads while moving and fighting.

3.2.3. Conceptualization

Robot's shape will act like natural bulldoser and it moves by two wheels powered by motors and robot itself can rotate by 360 degrees.

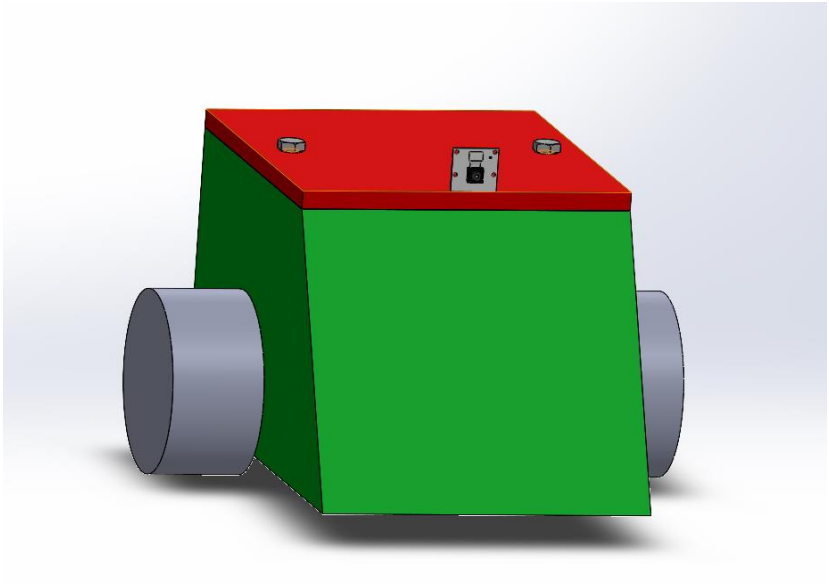


Figure 13. Front view of the robot

3.2.4. Materialization

Robot consists of two layers. First layer is outer coating with MDF.
Second layer is the interior which keeps motor supports and whole electrical parts will be stored in there.
View from backside Raspery Pi and the camera can be seen below.

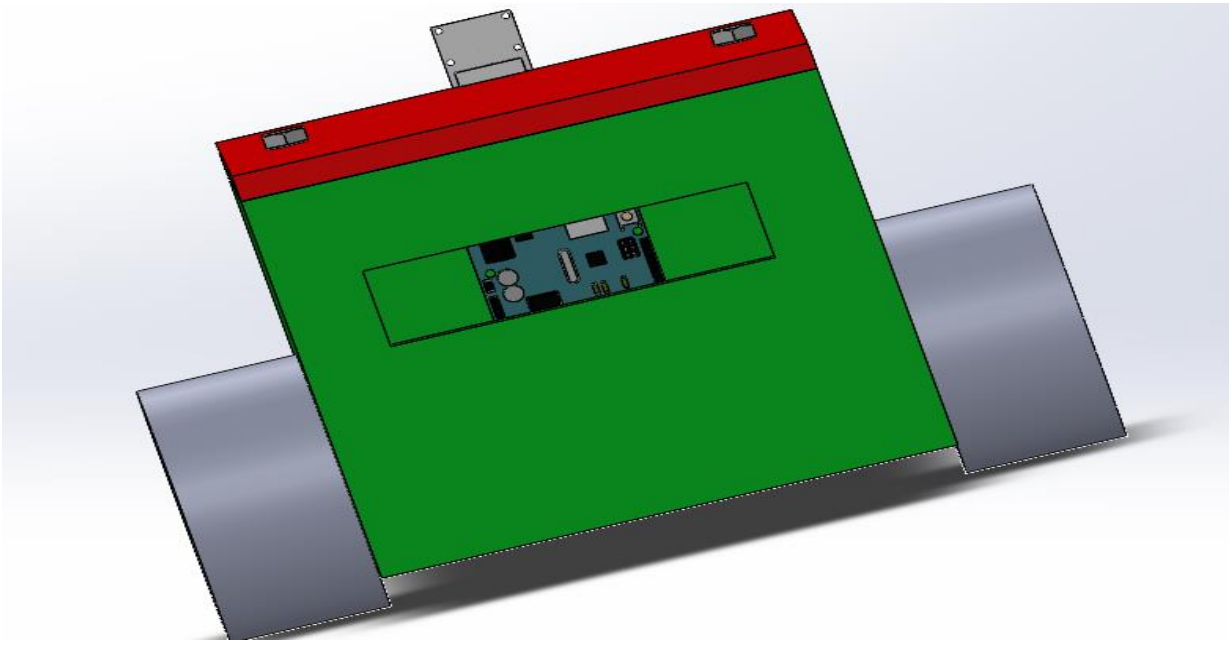


Figure 14. Backside view of the robot

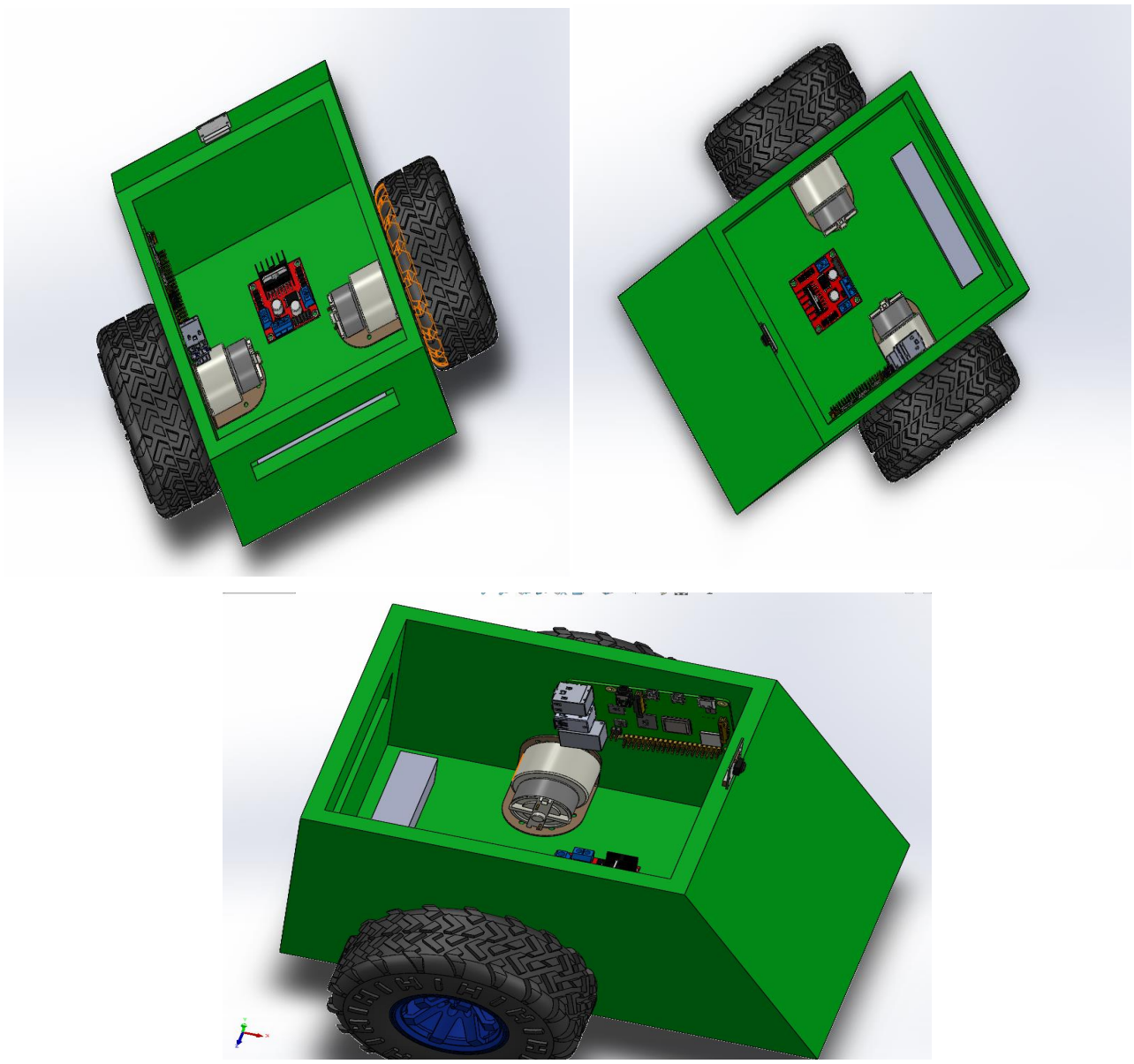


Figure 15. Interview of the robot

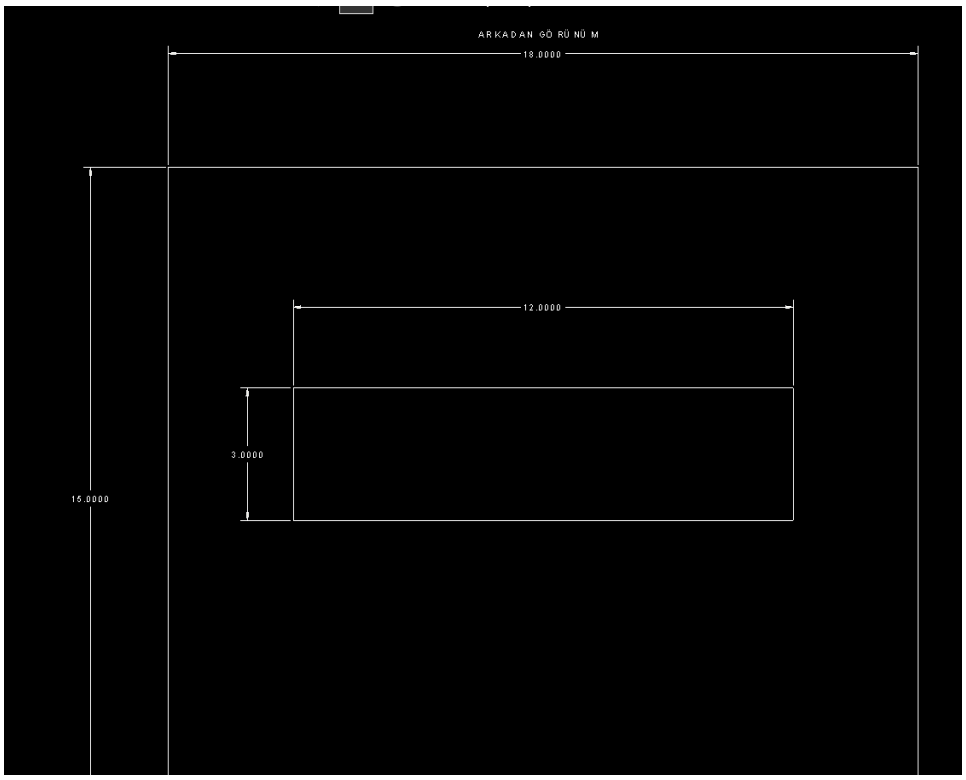


Figure 16. Autocad drawings from behind

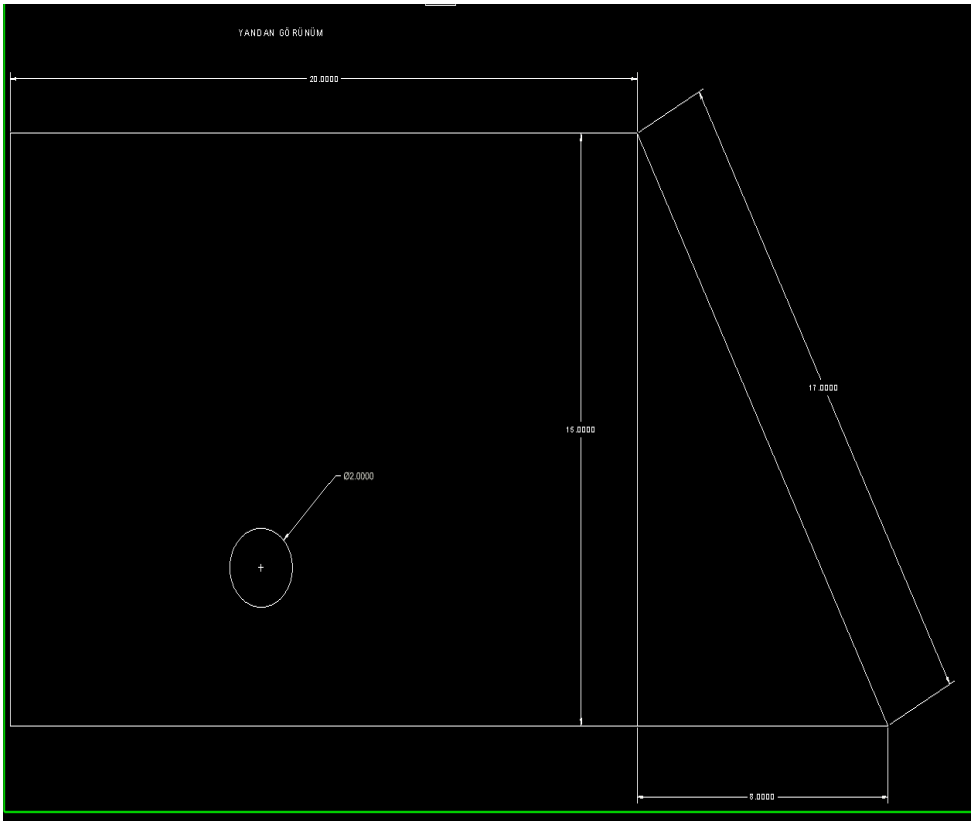


Figure 17. Autocad drawings from side

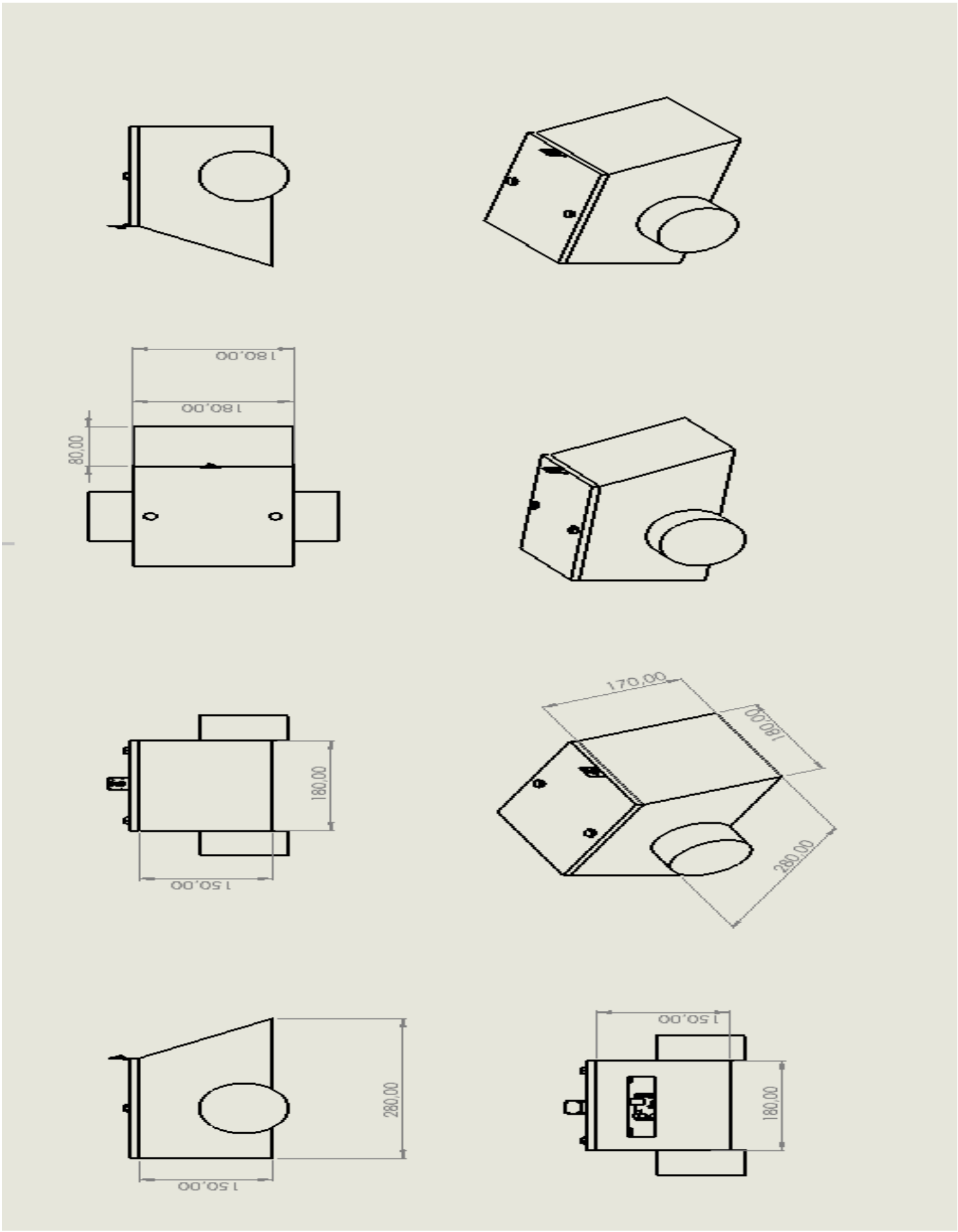


Figure 18. General view and measurements of the robot

3.2.5. Analysis

Simulation of Sumo robot

1- Statical Simulation

2- Heat Simulation

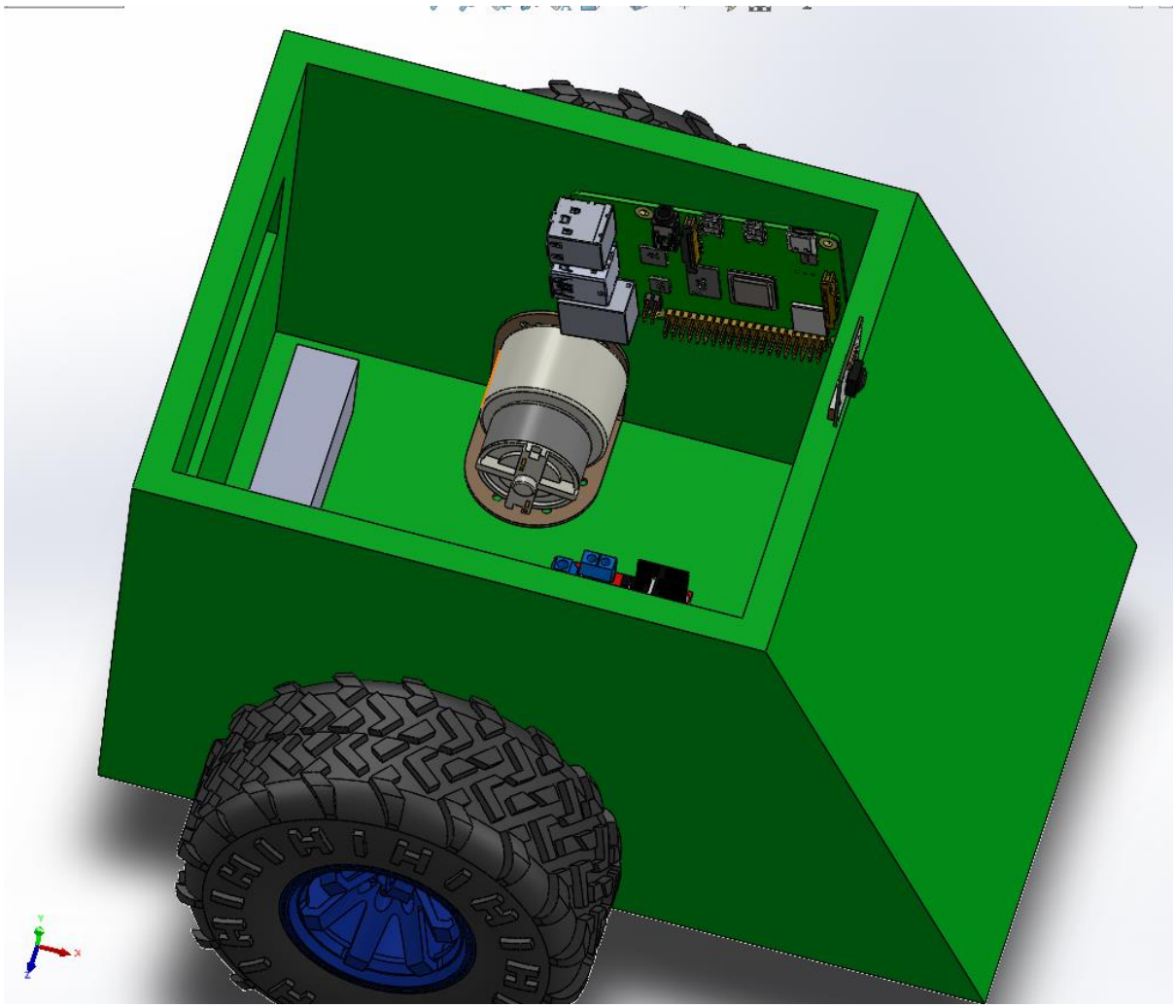


Figure 19. Top view of the robot

Material properties and items that conduce heat

Designmaterial of our robot is MDF

Property	Units	Thickness Class - mm			
		<=5	6 - 12	13 - 22	>23
Density	kg/m^3	800-850	775	725	650 -700
Bending Strength (MOR)	MPa	44	42	38	30 - 40
Bending Stiffness (MOE)	MPa	3800	3500	3300	3200
Internal Bond Strength	MPa	1.15	1.0	0.75	0.6
Surface Soundness	MPa	0.7	1.0	1.3	1.4
Screw Holding - Face	N	-	-	800	850
Screw Holding - Edge	N	-	-	1150	1000
Thickness Swell (24 Hr)	%	20-30	10-20	8-12	5-8
Formaldehyde E1 (Desiccator Method)*	mg/L	0.7 - 1.0	0.7 - 1.0	0.7 - 1.0	0.7 - 1.0

Table 6. Typical property values for standard MDF

CUSTOM MATERIAL CREATED AND ADDED IN SOLIDWORKS

Material properties

Materials in the default library can not be edited. You must first copy the material to a custom library to edit it.

Model Type: Linear Elastic Isotropic — Save model type in library

Units: SI - N/mm^2 (MPa)

Category: mdf

Name: mdf

Default failure criterion: Unknown

Description:

Source:

Sustainability: Undefined — Select...

Property	Value	Units
Elastic Modulus	4000	N/mm^2
Poisson's Ratio	0.25	N/A
Shear Modulus	2500	N/mm^2
Mass Density	775	kg/m^3
Tensile Strength	18	N/mm^2
Compressive Strength	10	N/mm^2
Yield Strength	18	N/mm^2
Thermal Expansion Coefficient	0.12	/K
Thermal Conductivity	0.3	W/(m·K)
Specific Heat		J/(kg·K)
Material Damping Ratio		N/A

Bulk Modulus		**2.6** GPa, or **0.37** 10^6 psi
Compressive (Crushing) Strength		**10** MPa, or **1.4** 10^3 psi
Density		**0.75** g/cm^3, or **46** lb/ft^3
Dielectric Strength (Breakdown Potential)		**0.5** MV/m, or **12** V/mil
Elastic (Young's) Modulus		**4** GPa, or **0.58** 10^6 psi
Elongation at Break	Typical	**0.5** %
Limiting Oxygen Index (LOI)		**24** %
Maximum Temperature	Typical Guideline	**130** °C, or **260** °F
Poisson's Ratio		**0.25**
Shear Modulus		**2.5** GPa, or **0.36** 10^6 psi
Specific Heat Capacity	Conventional	**1700** J/kg-K
	Volumetric	**1200** 10^3 J/m^3-K
Speed of Sound		**73** 10^3 m/s, or **230** 10^3 ft/s
Stiffness-to-Weight Ratio	Bulk	**3.5** MN-m/kg
	Shear	**3.3** MN-m/kg
	Tensile	**5.3** MN-m/kg
Strength-to-Weight Ratio	Compressive	**13** kN-m/kg
	Tensile, Ultimate	**24** kN-m/kg
Tensile Strength	Ultimate	**18** MPa, or **2.6** 10^3 psi
Thermal Conductivity	Ambient	**0.3** W/m-K
Thermal Diffusivity		🔒
Thermal Expansion	20 to 100°C	**12** µm/m-K

Table 7. Official values for MDF

We first analyzed in statical loads

Picture below are shown example fixtures of motor mounts.

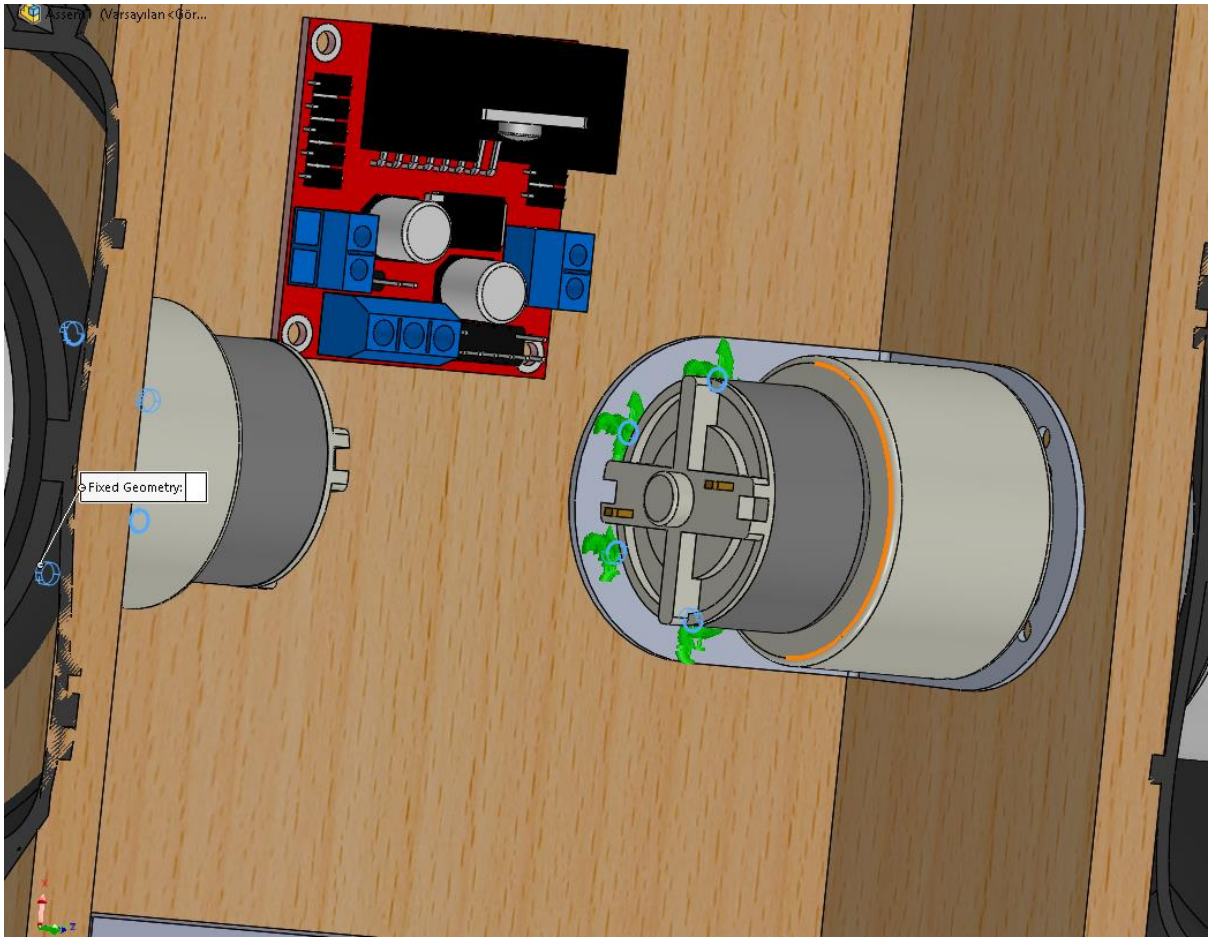


Figure 20. Fixtures of motor mounts

Results of simulation

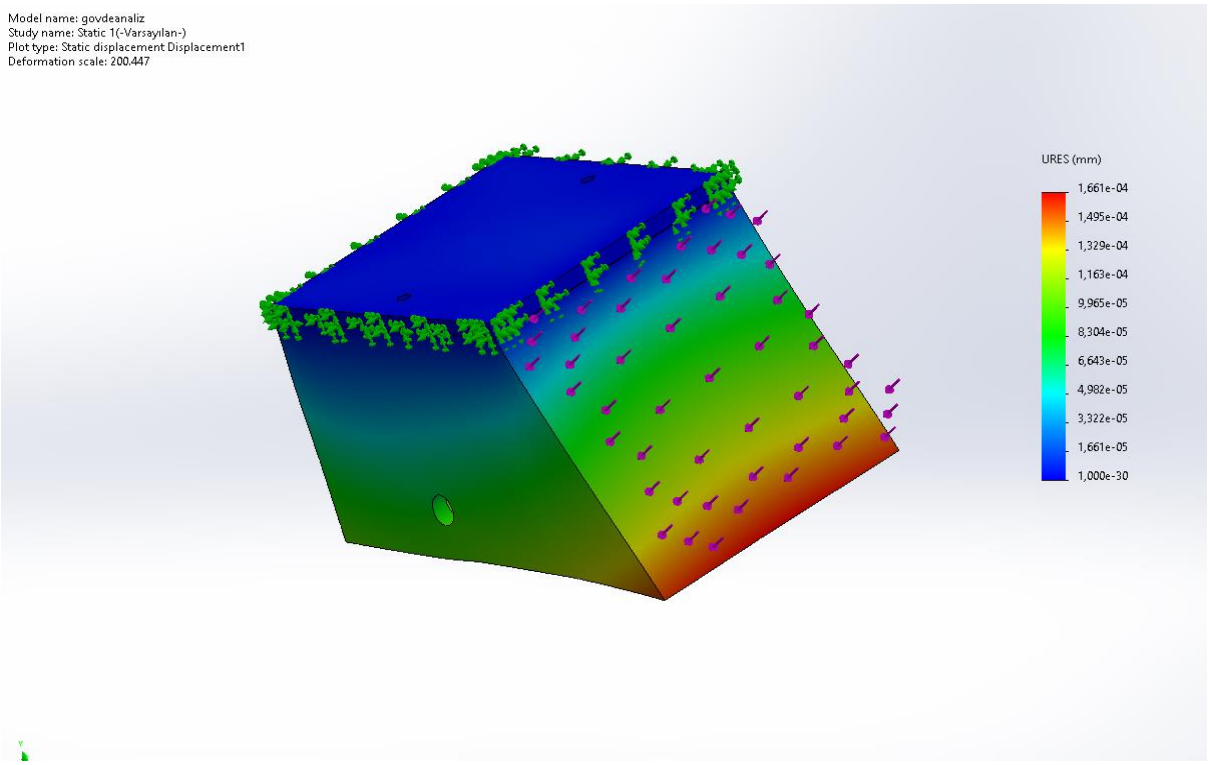


Figure 21. Static displacement analysis

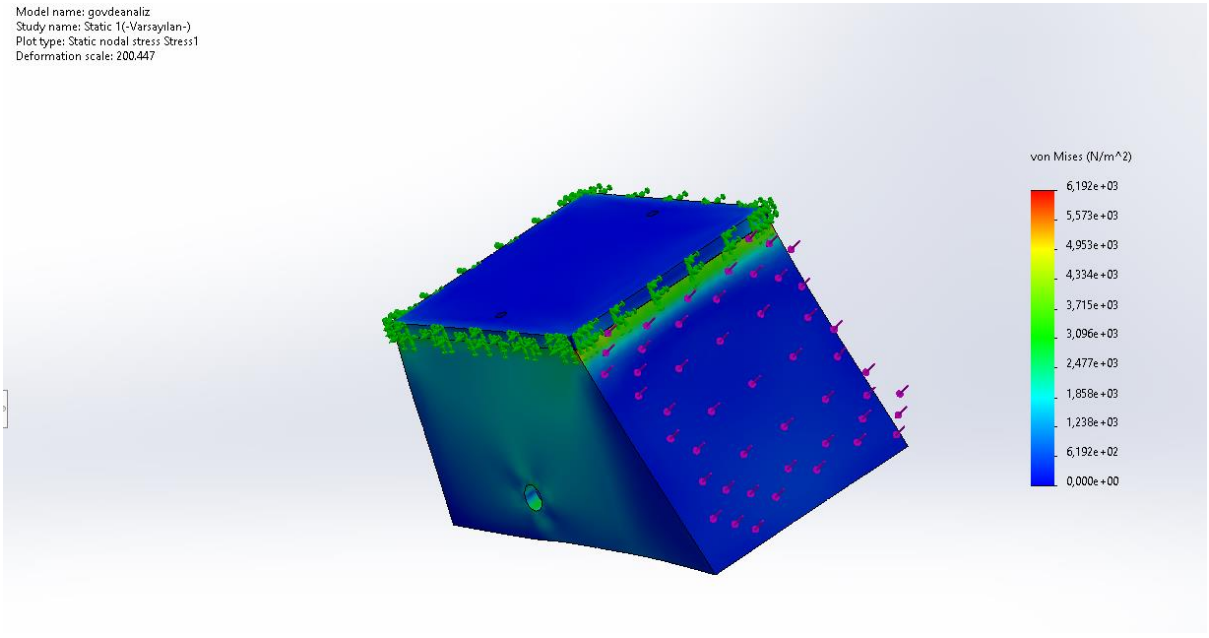


Figure 22. Von Mises stress analysis

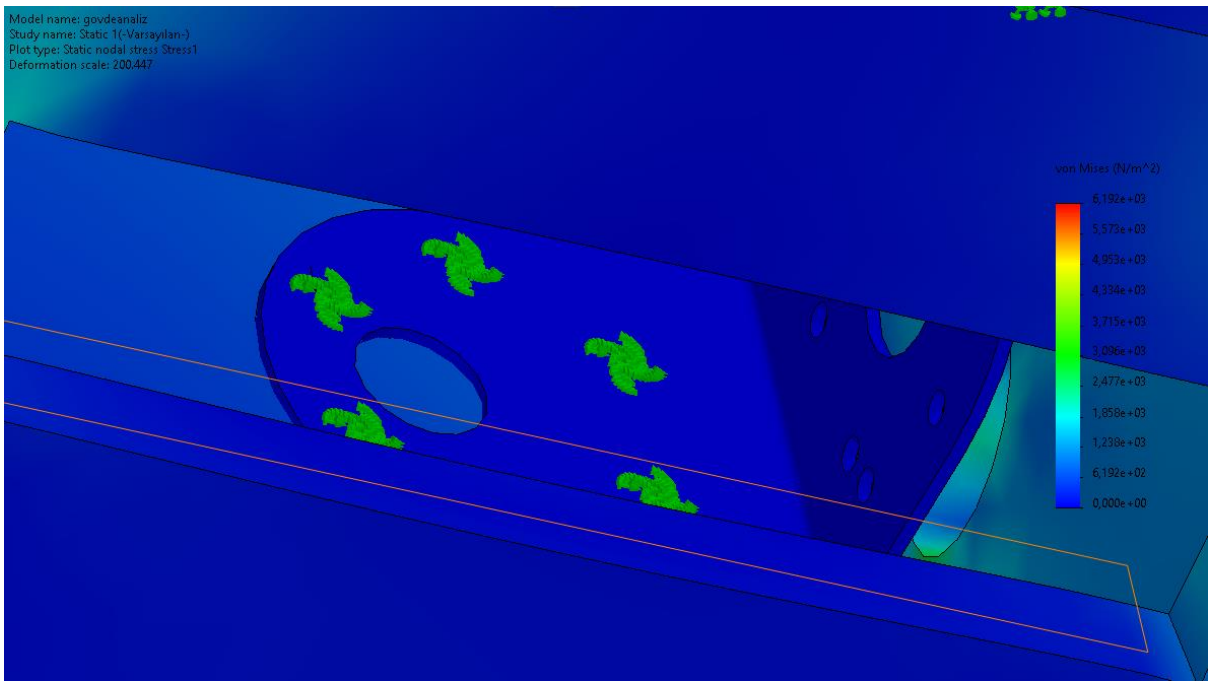


Figure 23. Motor mount fixtures analysis

We have applied force of 18 Newtons per item in milimeters. There is only slightly deformation due to lack of third Wheel in front of robot so we improvised and reduced stress to zero in our porduct.

Heat Analysis

Raspverry Pi can operate between 20 and maximum of 80 degrees celcius. In this simulation, it is obvious that there is no significant effect of heat in any kind.

Either Li-Po, Raspberry Pi nor motors can not produce significant heat to deform the robot.

But for simulations we estimated that Li-Po will work around 20 degrees and ambient temperature will be 25 so that in our analiysis we didn't see any effect of Li-Po battery.

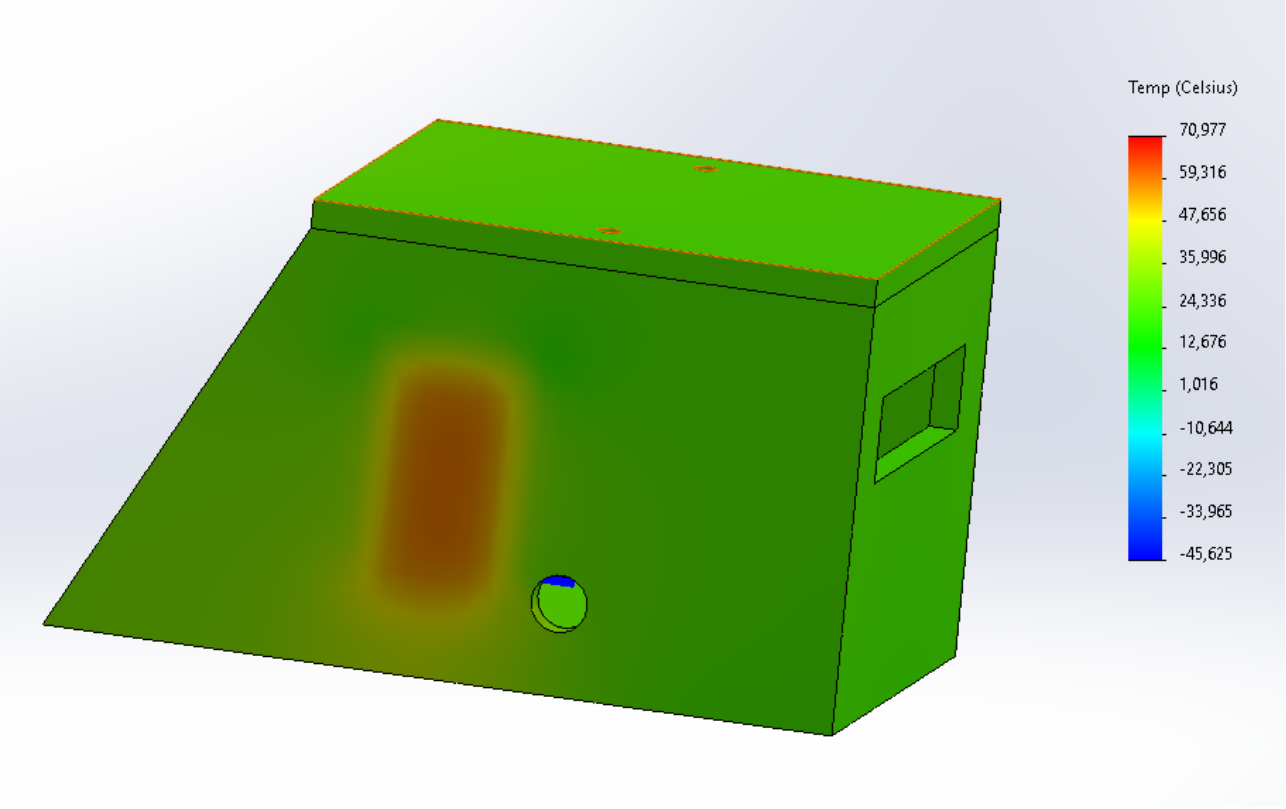


Figure 24. Heat analysis

There is no significant heat effect to our body. Only Raspberry will produce heat higher than ambient and we assumed that it will be working in maximum temperature limit so in reality it will not be that effective.

3.2.6. Mathematical Modeling

Mathematical modeling of Sumo robot

Definition and kinematics

Sumo robot has designed for two rotating wheels and our kinematics are based on this.

18=2b= the width of our robot

8mm=4r= wheels diameter

Qx= Angular positions of wheels

O-xy= coordinate system

Op= mobile center gravity

In our study, Sumo robot moves accordingly to the centerline so robot first centers enemy then moves to it.

$$\dot{x}_G \sin\ \psi - \bar{y}_G \cos\ \psi = 0$$

$$\begin{pmatrix} (\dot{x}_F) \\ (\dot{y}_F) \\ (\psi_F) \end{pmatrix} = T_{NH} \begin{pmatrix} 2Q \\ 2Q \end{pmatrix} \text{atau}\ \ q(t) = T_{NH}(q')Q(t)$$

TNH is non holomic transformation matrix. Q is coordinate and tetas or radial velocity.

$$q' = \left[x_F, y_p, \psi\right]^{\top} \text{atau}\ \ q' = \begin{bmatrix} x_F \\ y_F \\ \psi \end{bmatrix} za$$

$$\mathrm{TNH}\,(q')$$

$$= \begin{pmatrix} \frac{r}{2}\cos\,\psi + \frac{d\cdot r}{2b}\sin\,\psi & \frac{r}{2}\cos\,\psi - \frac{d\cdot r}{2b}\sin\,\psi \\ \frac{r}{2}\sin\,\psi = \frac{d_1 r}{2b}\cos\,\psi & \frac{r}{2}\sin\,\psi + \frac{d_{\cdot} r}{2b}\cos\,\psi \\ \frac{-r}{2b} & \frac{r}{2b} \end{pmatrix}$$

Our inverse kinematics are:

$$Q(t) = TNH^{-1}(q^{\top})(q^{t}(t))$$

Dynamics

Q= robots position

τ= actuators torque= 500 rpm

B(q)= Matrix vector dependent on torque

M(q)= 2x2 inertio matrix

V(q,q')= coriolis effect force and G(q) gravity

A(q)= transformation matrix

$$B(q)\tau = M(q)q'' + V(q,q')q' + G(q) - A^{T}(q)\lambda$$

Final Dynamics Matrix

$$B(q)\tau = M(q)Q'' + V(q,q')q' + \tau.$$

$$M(q) = \begin{bmatrix} \frac{r^2}{8b^2}(mb^2+1) + Iw & \frac{r^2}{8b^2}(-b^2-1) \\ \frac{r^2}{8b^2}(mb^2-1) & \frac{r^2}{8b^2}(mb^2-Iw) \end{bmatrix}$$

$$u(q,q') = \begin{bmatrix} 0 & \frac{r^2}{4b^2}(m_C d\psi') \\ -\frac{r^2}{4b} r_\tau d\psi' & 0 \end{bmatrix}$$

$$B(q) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Angular velocity of left wheel

$W_L = (I\omega + 2v_{lim})$)

/(2 Rwheel)

Angular velocity of right wheel

$W_R = (2v_{\text{lim}} - I\omega)/(2$ Rwheel)

Linear velocity $V = \omega \times R$

3.3. Computer Engineering Subsystem

The CMP sub-team consists of Bahçeşehir University senior students Yunus Elçi, Berkay Emre Yoran and Ardahan Dikmedaş. As we mentioned in the concept comparison chart, we preferred to connect the camera to the robot in this project. The aim of the CMP team is to develop the software that will be used with MCH teams hardware for autonomous sumo robot.

3.3.1 Requirements

The responsibility of the computer team on this project will be to process the data from the camera with image processing techniques, determine the location of the objects according to whether they are friend or foe, send their coordinates to microcontrols and enable the robot to move autonomously.

To achieve this, we did image processing with python using OpenCv. This software we have created can recognize objects with different corners and different colors as friends or enemies. We do this definition manually after learning the objects on the dohyo.HSL and HSV are alternative representations of the RGB color model designed by computer graphics researchers in the 1970s to more closely match the way human vision perceives color rendering properties. Because the HSV color range is far from the RGB color model we know, it is not always possible to capture the correct color range in software. . Although our perception of colors changes according to changing light sources, our perception of shapes does not change much according to any variable. Also, since we do the image processing via rasp, we don't always get very fast results depending on the moving camera. But according to the performance target we have set, the robot must detect the enemy within minutes and throw it out.

We restrict the movements of the robot in order to reach the performance target we have set. While making this restriction, instead of making the robot move very slowly, we make the robot's movements move piece by piece as if scanning a certain area in a second.

Table 8 Requirements for Actors.

Actors **Requirements**	**Transmitter**	**Receiver**
Get images from real World	The transmitter side of the program should be able to get images from real World and send it to the Rasp.	The receiver of our product is rasp model b. It is in charge of processing and interpreting incoming images.
Motor movement according to the interpreted pictures	Rasp sends the processed images to the motor driver.	The motor driver transmits the incoming electrical information to the motors.
The change of the target object's area according to the robot movement.	Since the image on the camera side gets bigger as the robot gets closer, the area of the object will change and it will be perceived as an object with a larger area on the Rasp side.	Rasp realizes that the area of the incoming image grows, accordingly adjusts its previous sensitivity setting as it gets closer to the object.

3.3.2. Technologies and methods

As computer engineering students, we will process the data coming from the camera in this project. Image processing and object detection techniques will be used here, as the data from the camera is an image. In this context, libraries such as OpenCv and Keras can be used in the Python programming language.

3.3.2.1 Image Processing

We process the images detected by the camera on the robot with the OpenCv library in the Python programming language. After these images are detected in the camera, they are sent to the raspberry pi device. The incoming image is blurred with the Gaussian Blur method. Blurred images are converted to gray and hsv format. Immediately after, threshold values are defined for the images to detect shapes in the images. The defined threshold values are applied to the images that we converted to gray in the previous step. Then, the lower limit and upper limits consisting of the Hsv value are determined to detect red, blue and green colors. Although these threshold values are suitable for Blue, Red and Green colors, they can give different results according to changing ambient light. Afterwards, we do the masking process to see only the blue, red or green color in the image.

Afterwards, the masked images are sent to the function we created for shape detection. Shape recognition is performed on a separate color for each picture.

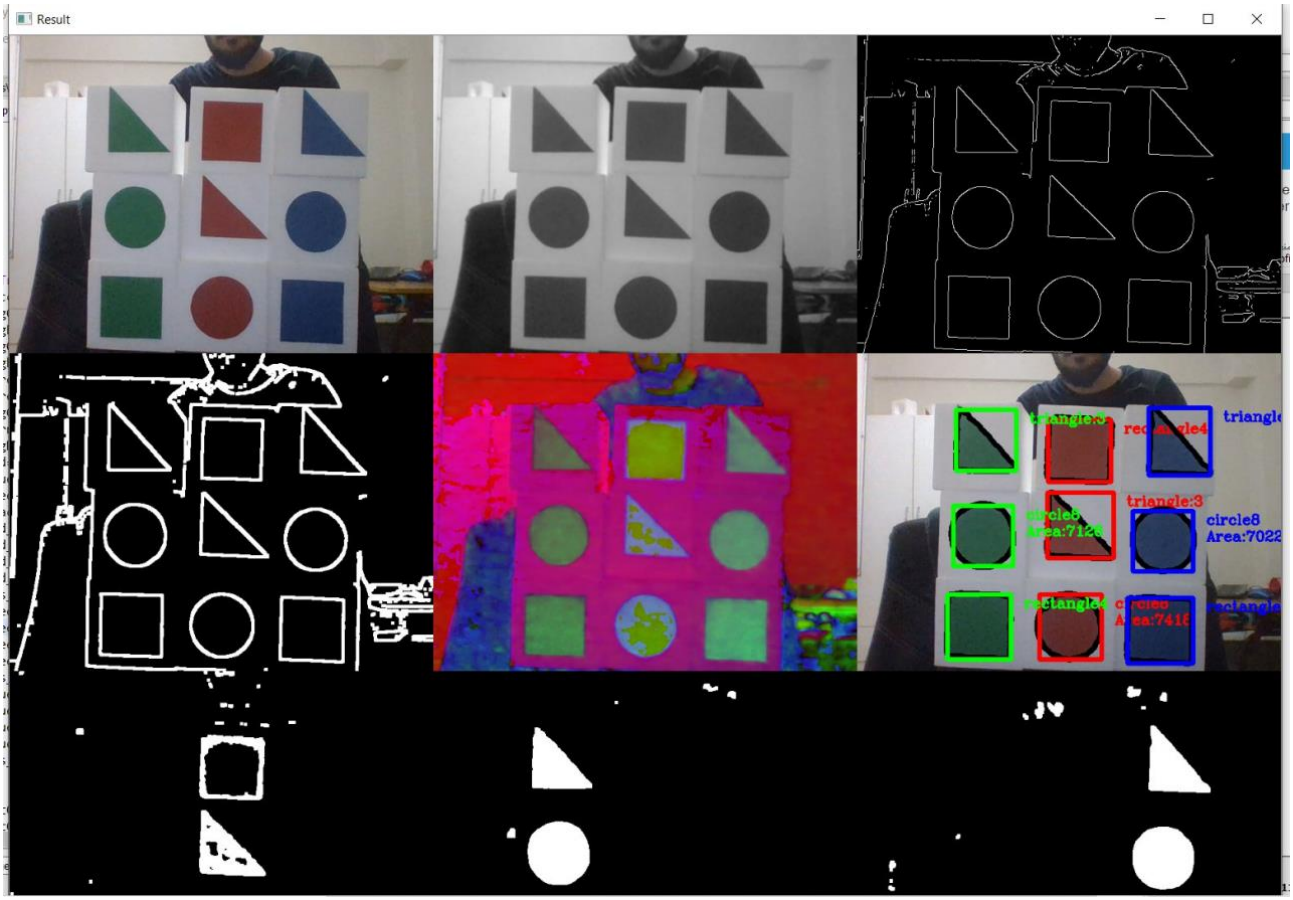


Figure 25. Image Processing Result

3.3.2.2 Autonomous Movement

According to the result in image processing, the autonomous movement of the robot is expected. In this section, we can define objects with different colors and shapes as friends or enemies. The color of the object and the number of edges can be decisive in this regard. In order to move, the robot first tries to center the enemy object in front of it. Here, the midpoint of the shape in the technically processed image is found and the camera is provided to look at that point. Then the object is considered in the middle and the robot moves towards the object. The object is pushed out of the dohyo. Here, the sensor we have keeps the robot in dohyo.

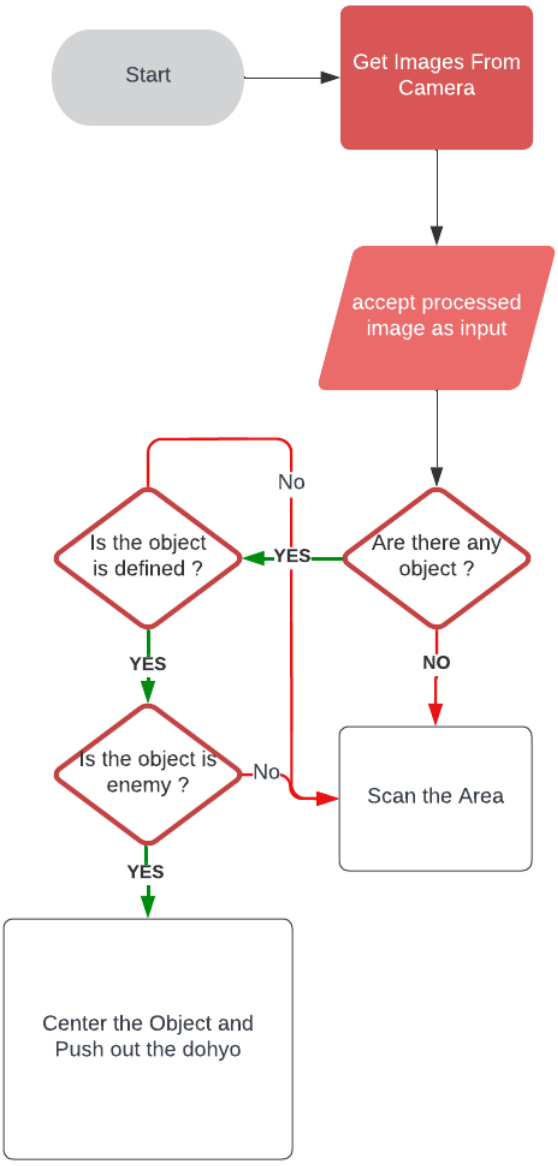


Figure 26, Autonomous Movement Algorithm

3.3.3. Implemantation

3.3.3.1 Build 1:

The development of the software started with image processing using OpenCv. We can say that the use of Hsv color definitions for the perception of colors has been the biggest difficulty for us in perceiving colors. Then, many debug processes were carried out to find the appropriate values of the threshold values to be applied to the images. A certain field threshold was defined so that every angular object in the images is not detected. After this definition, the desired objects were made identifiable.

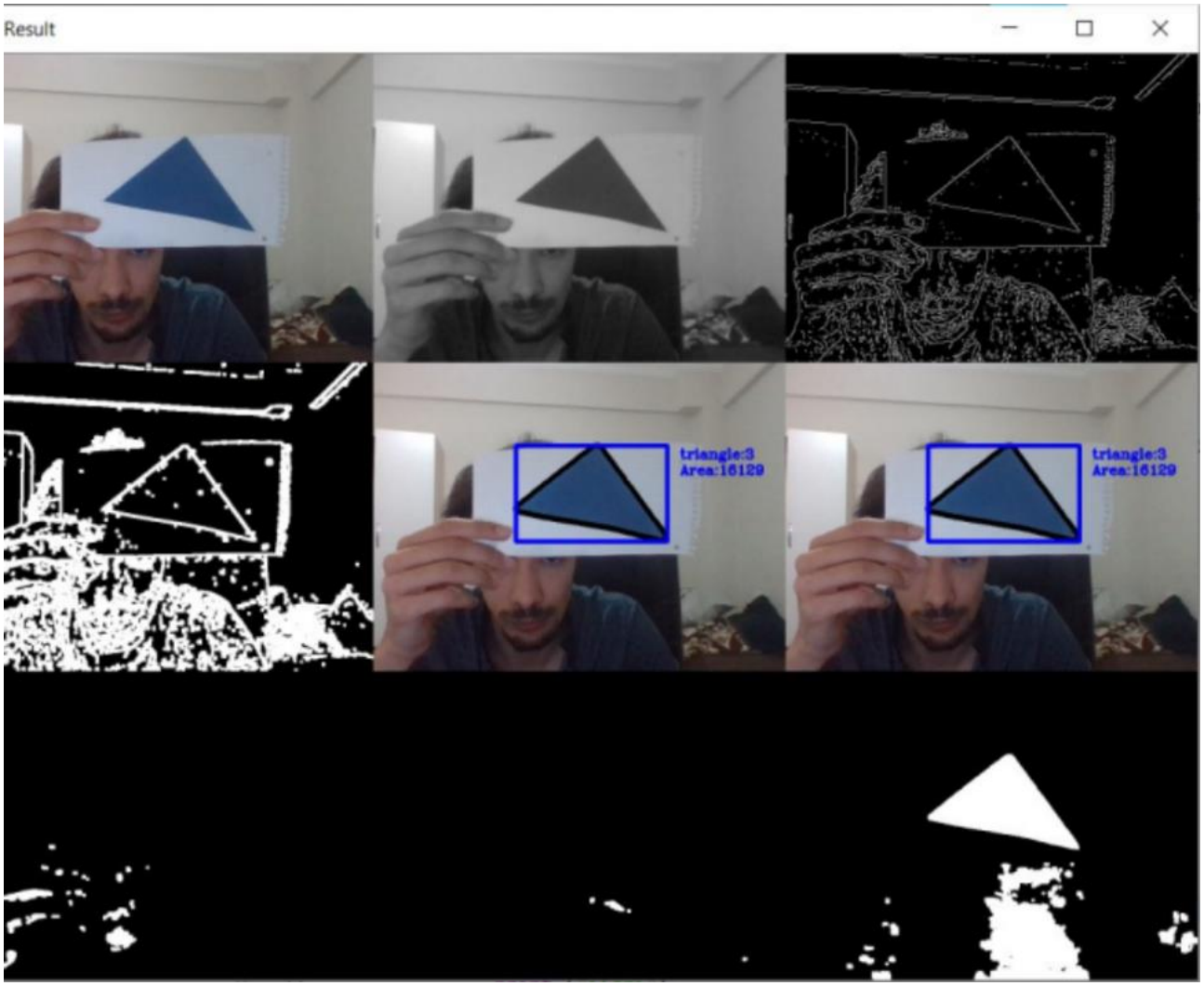


Figure 27. Build 1

3.3.3.2 Build 2:

After detecting the shapes we wanted, we started to create a simple motion algorithm. According to this algorithm, if the object was on the right, we would turn the robot to the right, and if the object was on the left, we would turn the robot to the left. Later, by making improvements in this algorithm, the center position of the object was changed to turn right if it is to the right of the center of the robot's field of view. According to this algorithm, the robot will equate the center of the object it sees with the center of its field of view and take action.

```
def move(state):
    global inumber
    if state == "ileri":
        print("ileri")

        p.ChangeDutyCycle(50)
        p2.ChangeDutyCycle(50)

        GPIO.output(in1, GPIO.LOW)
        GPIO.output(in2, GPIO.HIGH)
        GPIO.output(in3, GPIO.LOW)
        GPIO.output(in4, GPIO.HIGH)
        p.ChangeDutyCycle(40)
        p2.ChangeDutyCycle(40)
        state = "z"

    if state == "sol":
        print("sol")
        p.ChangeDutyCycle(45)
        p2.ChangeDutyCycle(45)

        GPIO.output(in1, GPIO.LOW)
        GPIO.output(in2, GPIO.HIGH)
        GPIO.output(in3, GPIO.LOW)
        GPIO.output(in4, GPIO.LOW)
        p.ChangeDutyCycle(37)
        p2.ChangeDutyCycle(37)
        state = "z"
```

Figure 28, Build 2 Example of move function

```
areaMin = 1700
if area > areaMin:

    peri = cv2.arcLength(cnt, True)
    approx = cv2.approxPolyDP(cnt, 0.02 * peri, True)

    x, y, w, h = cv2.boundingRect(approx)
    if (len(approx) == 3) & (collor == red):
        cv2.drawContours(imgContour, cnt, -1, (0, 0, 0), 7)
        cv2.rectangle(imgContour, (x, y), (x + w, y + h), collor, 5)
        center_of_object = (x + (w / 2))
        sens = area / 140
        # print(center_of_object)
        if center_of_object > center_width + sens:  # soldasya
            move("sag")
        elif center_of_object < center_width - sens:  # sağdaysa
            move("sol")
        else:
            move("ileri")
        cv2.putText(imgContour, "triangle:" + str(len(approx)), (x + w + 20, y + 20), cv2.FONT_HERSHEY_COMPLEX,
                    .7, collor, 2)
        cv2.putText(imgContour, "Area:" + str(int(area)), (x + w + 20, y + 45), cv2.FONT_HERSHEY_COMPLEX, .7,
                    collor, 2)
```

Figure 29, Build 2 Example of decision function

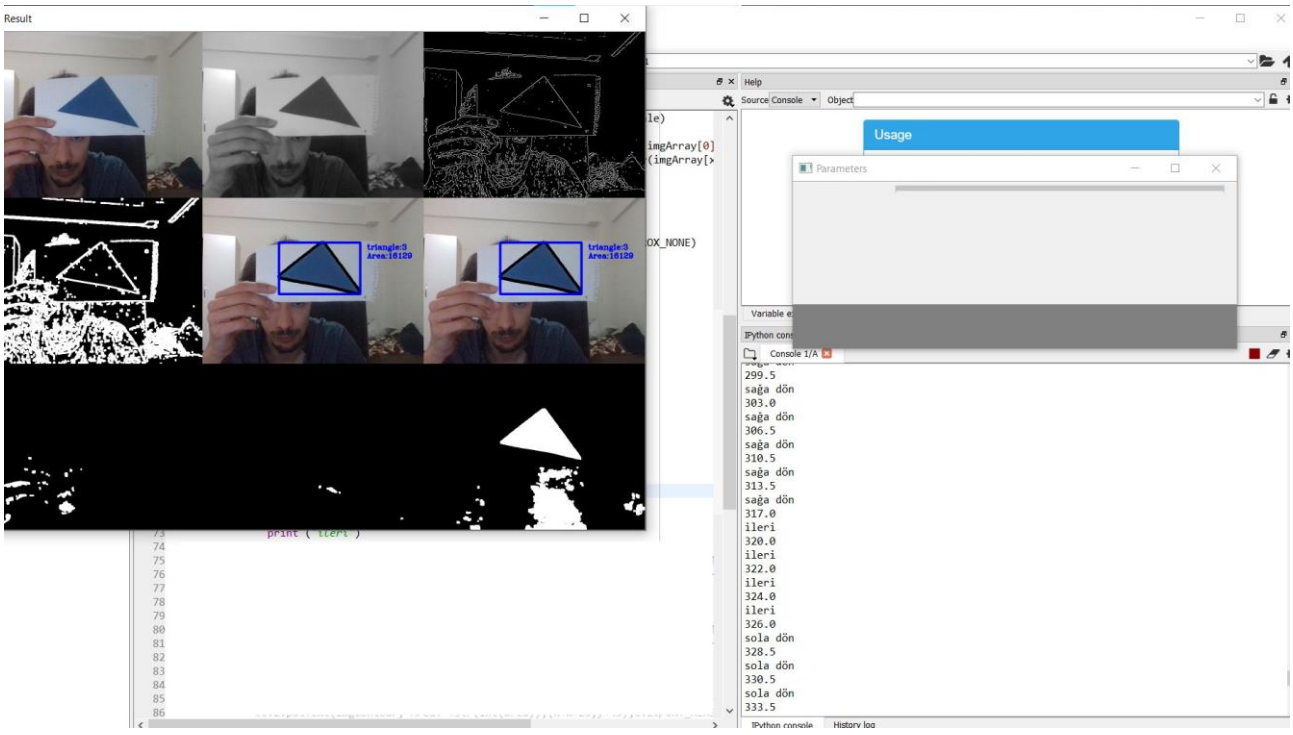


Figure 30, Build 2 Working State

3.3.3.3 Build 3 (Current Build):

The finalization of the program was improved by improving the behavior of the robot when approaching the enemy object. In the builds before this version, the program showed an undesirable behavior when approaching the object because the object's viewing angle changed. We define this situation to the program that the area of the object increases as it gets closer to the center of the object, and we change the sensitivity of the robot with this area. Thanks to this, the robot knows that when it gets to the bottom of the object, it has to keep pushing the object.

3.3.4. Conceptualization

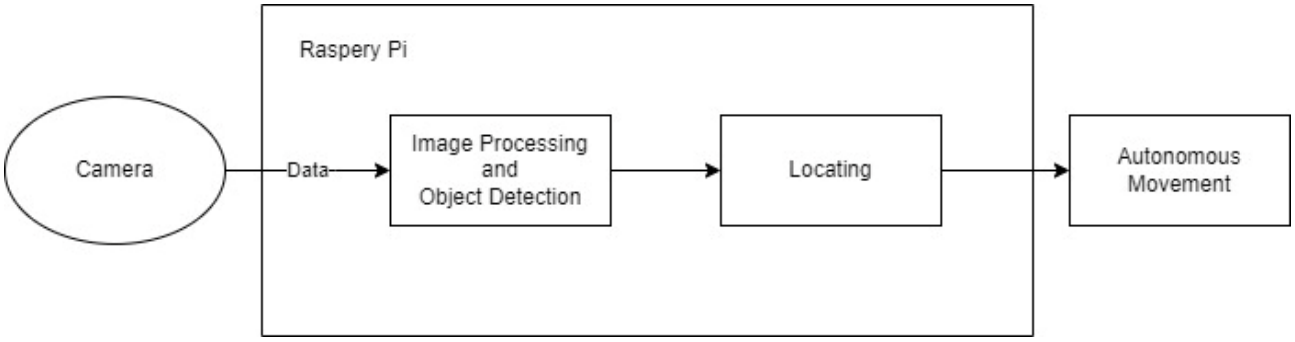


Figure 31. Conceptualization

3.3.5. Evaluation

3.3.5.1 Evaluation of Image Processing:

As previously discussed by the team and mentioned in the report, the image processing part was evaluated. According to this evaluation, the software should be able to separate objects according to their shapes and colors. Objects separated according to their color and shape should be defined as friends or enemies. The determined shapes should be reinterpreted according to the movement of the camera, and the sensitivity of the software should be able to detect the object as the area of the objects changes. As a result of our testing and debug processes, the software can successfully evaluate the desired situations.

3.3.5.2 Evaluation of Autonomous Movement:

This part included Raspberry pi device and Arduino Uno controller at the evaluation stage. Objects interpreted via Raspberry pi were transmitted to Arduino Uno device via serial port. As a result of this serial communication, Arduino uno device was expected to manage the motor driver and do the related movement. As a result, during the evaluation process, serial communication was slow due to the Arduino uno running much slower than expected, and the software had to limit the image processing speed it wanted. As a result of this, we disabled the Arduino uno device and enabled the motor driver to work directly with the Raspberry pi.

```
def move():
    global state
    if state == "ileri":
        print("ileri")

        p.ChangeDutyCycle(50)
        p2.ChangeDutyCycle(50)

        GPIO.output(in1, GPIO.LOW)
        GPIO.output(in2, GPIO.HIGH)
        GPIO.output(in3, GPIO.LOW)
        GPIO.output(in4, GPIO.HIGH)
        p.ChangeDutyCycle(37)
        p2.ChangeDutyCycle(37)
        state = "z"

    elif state == "sol":
        print("sol")
        p.ChangeDutyCycle(45)
        p2.ChangeDutyCycle(45)

        GPIO.output(in1, GPIO.LOW)
        GPIO.output(in2, GPIO.HIGH)
        GPIO.output(in3, GPIO.LOW)
        GPIO.output(in4, GPIO.LOW)
        p.ChangeDutyCycle(37)
        p2.ChangeDutyCycle(37)
        state = "z"
    elif state == "sag":
        print("sag")
        p.ChangeDutyCycle(20)
        p2.ChangeDutyCycle(20)
        GPIO.output(in1, GPIO.LOW)
        GPIO.output(in2, GPIO.LOW)
        GPIO.output(in3, GPIO.LOW)
        GPIO.output(in4, GPIO.HIGH)
        p.ChangeDutyCycle(20)
        p2.ChangeDutyCycle(20)
        state = "z"
    elif state == "scan":
        print("scan")
        p.ChangeDutyCycle(43)
        p2.ChangeDutyCycle(43)
        GPIO.output(in1, GPIO.HIGH)
        GPIO.output(in2, GPIO.LOW)
        GPIO.output(in3, GPIO.LOW)
        GPIO.output(in4, GPIO.HIGH)
        p.ChangeDutyCycle(37)
        p2.ChangeDutyCycle(37)
        state = "z"
    if state == "stop":
        print("stop")
        p.ChangeDutyCycle(20)
        p2.ChangeDutyCycle(20)
        state = "z"
```

Figure 32, Movement Implemantations On Raspberry Pi With Python

4. INTEGRATION AND EVALUATION

4.1. Integration

The parts of the robot will be assembled and delivered to the software team after we receive them, and the software team will make the robot ready for presentation after completing the image processing part.There will be several software tests during integration of final assembly.As a final test, a dohyo-like area with different shape and color of enemies will be created to make sure our robot will work on exact conditions for the presentation.

5. SUMMARY AND CONCLUSION

Taking everything into account, Autonomous Sumo Robot project requires knowledge of electronic, mechanical and computer science integrating as a whole system. Within the utilization of subsystems and external architecture, we are able to manufacture and test our project.By working as a team, we have knowledge of different disciplines of engineering.

As a final observation, this project will be made out of both mechanical, electronical and software based components. Using all these components properly will allow us to create our autonomous Sumo Robot which is capable of detecting enemy and ally objects within the dohyo by camera and take the enemies out of it with the help of ramp.

ACKNOWLEDGEMENTS

We wish to thank our advisers Doç. Dr. Yalçın ÇEKİÇ and Assist. Prof. Tarkan AYDIN for their encouragement and guidance during the research and development stages of this project.

-REFERENCES

1. Wikipedia, “Digital Image Processing”, [Online]. Available: https://en.wikipedia.org/wiki/Digital_image_processing. [Accessed: December 25, 2021].
2. Fritz, “Object Detection”, [Online]. Available: https://www.fritz.ai/object-detection/ [Accessed: December 25, 2021].
3. Wikipedia, “OpenCV”, [Online]. Available: https://tr.wikipedia.org/wiki/OpenCV. [Accessed: December 25, 2021].
4. Kaggle, “OpenCV Image Processing”, [Online]. Available: https://www.kaggle.com/shivamanhar/opencv-image-processing. [Accessed: December 25, 2021].
5. Pyimagesearch, “OpenCV Shape Detection”, [Online]. Available: https://www.pyimagesearch.com/2016/02/08/opencv-shape-detection/.[Accessed: December 25, 2021].
6. Researchgate, “The design and construction of an autonomous mobile mini-sumo robot” [Online]. Available: https://www.researchgate.net/publication/281267184_THE_DESIGN_AND_CONSTRUCTION_OF_AN_AUTONOMOUS_MOBILE_MINI-SUMO_ROBOT [Accessed: December 29, 2021].
7. Arduino,” How to Make Arduino Sumo Robot” Available: https://create.arduino.cc/projecthub/AhmedAzouz/how-to-make-arduino-sumo-robot-f44bd8 [Accessed: December 29, 2021].
8. Hackster, “Building a Sumo Robot “ Available: https://www.hackster.io/cytron-technologies/building-a-sumo-robot-45d703 [Accessed: December 29, 2021].
9. Modelling and Simulation of Vehicle Kinematics and Dynamics Avaliable: https://hh.diva-portal.org/smash/get/diva2:1115795/FULLTEXT02.pdf [Accessed: May 29 , 2017].
10. Applied Robotics with the Sumo Bot Avaliable: https://www.pololu.com/file/0J209/AppliedSumo-v1.0.pdf [Accessed: May 29 , 2017].
11. ME dept. Sumo Bot Competition Avaliable: https://ceias.nau.edu/capstone/projects/ME/2017/SumoBotA/Final%20Report.pdf[Accessed: 2016-17]
12. Movement modeling of mobile robot in MegaSumo category Avaliable: https://aip.scitation.org/doi/pdf/10.1063/1.5066472 [Accessed: 29 October 2018]

APPENDIX

APPENDIX A

Simulation Files And Cad Drawings:
https://drive.google.com/drive/folders/1Kim0b3sHXATxUpptYLjLtrM9tVdWjXWK?usp=sharing

Fritzing Files For Electrical Components:
https://drive.google.com/file/d/1WATLu4MrgfRTbLLq2ewma000s3rY203p/view?usp=sharing

APPENDIX B

Program Codes: https://github.com/yunuselci/pythonProject