Autonomous Guided Carrier Vehicle

Prepared By

Ahmed Hesham Ahmed Eid
Islam Moheb Abdel Hamid
Mohamed Ashraf Gamaeldin Daoud
Mohamed Fawzy Ibrahim Shehab

Supervised By

Dr. Mohamed Abdelaziz

July 5, 2018

Declaration Of Authorship

We, AGCV team members, declare that this thesis titled,'Autonomous Guided Carrier Vehicle (ACV)' and the work presented in it are our own.
we confirm that:

1. This work was done wholly or mainly while in candidature for a Bachelor degree at this University.
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
3. Where we have consulted the published work of others, this is always clearly attributed.
4. Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely our own work.
5. We have acknowledged all main sources of help.
6. Where the thesis is based on work done by ourselves jointly with others, we have made clear exactly what was done by others and what we have contributed ourselves.
7. Either none or part of this work has been published before submission.

Abstract

The purpose of this thesis is to improve and complete the autonomous vehicle implemented by our previous colleagues [1] to be used in one of Leoni's factories in Egypt. The focus on this thesis will be upon the design and implementation of the autonomous guided vehicle only, and not the H-Bot mechanism.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Project Aim

Innovative products require an interdisciplinary combination of mechanical engineering, electrical engineering and information technology. Product innovations make a decisive contribution to the way in which these products maintain their position in our global economy. As Mechatronics engineers, we are required to make possible new fundamental solutions which considerably improve the cost/benefit ratio of currently known products. The goal of this project is to acquire the best practical application to reach an industrial level challenge and experience. This was achieved by attempting to form a partnership with Leoni, a global supplier of wires, cables and wiring systems, to improve, design and implement an integrated system in their factory in Egypt. The system is an autonomous carrier vehicle that maneuvers inside the factory between certain stations to transport cables and wires. Pursuing this project opportunity, we considered to apply the knowledge attained from the previous years of study and our practice in the past projects on every aspect of each level of system design and integration throughout this project.

1.2 Problem Definition

The starting point is formed by an actual development order. The defined product was specified more precisely and described in the form of the measure against which the later product is to be assessed. As this project is directed to improve the design and re-implement an already existing product, it is crucial to understand what the product was initially intended for, and the defects that needs adjustments in the current product.

Firstly, it was declared by Leoni that the main problem they seek to solve is to reduce the time and individual effort consumed by transporting different packages filled with wires and connectors between stations inside the factory. It was required to build a carrier vehicle to transport up to 40 kg packages at a time, maneuvers autonomously in a low speed while avoiding obstructions or collisions within half a meter, and keeps a log of the orders made by each station to keep track of the components ordered via an easy interface.

Secondly, the previously implemented vehicle had the following drawbacks:

- The transported packages were unsecured to the vehicle.
- The vehicle was tested once, and did not work again due to design defects.
- The vehicle did not maneuver autonomously even in the conducted test.
- The recharge station of the vehicle was not finalized/concluded.
- A steering system was not fully implemented. Instead, a 2-wheel rear drive was designed.

1.3 Proposed Solutions

As we discussed and cleared the initial requirements and the problem statements in the previous section, we began the approach towards dividing the problems and figuring out their proper corresponding solutions. As an initial step, we visited the factory in order to obtain the previous vehicle. Later, we thoroughly analyzed and examined the previous vehicle to further study and understand the picture. In order to reduce the cost of this project, we decided to reuse some modules, e.g. ultrasonic and line-follower sensors, wheels, bearings and DC motors, in the new vehicle. However, it was clear after inspection that the mechanical and electrical aspects needed to change.

1.4 Solution Approach

As Mechatronics engineers, we base the principle for the planning and implementation of effective problem-solving behavior on a problem-solving micro-cycle [2], as shown in fig 1.1. The search for solutions to the previously defined problems of an already existing product takes place against the background of situation analysis and objective. This process takes the form in practice of a permanent alternation between synthesis steps and analysis steps. This can take place for example by calculation, simulation and trial experimentation. With the decision, it must be established whether the previous procedure for problem-solving has led to a satisfactory result. If this is not the case, a return must be made to the situation analysis and formulation of a goal. Otherwise, a decision is made on an alternative for a solution, which is to be made the basis of further planning.

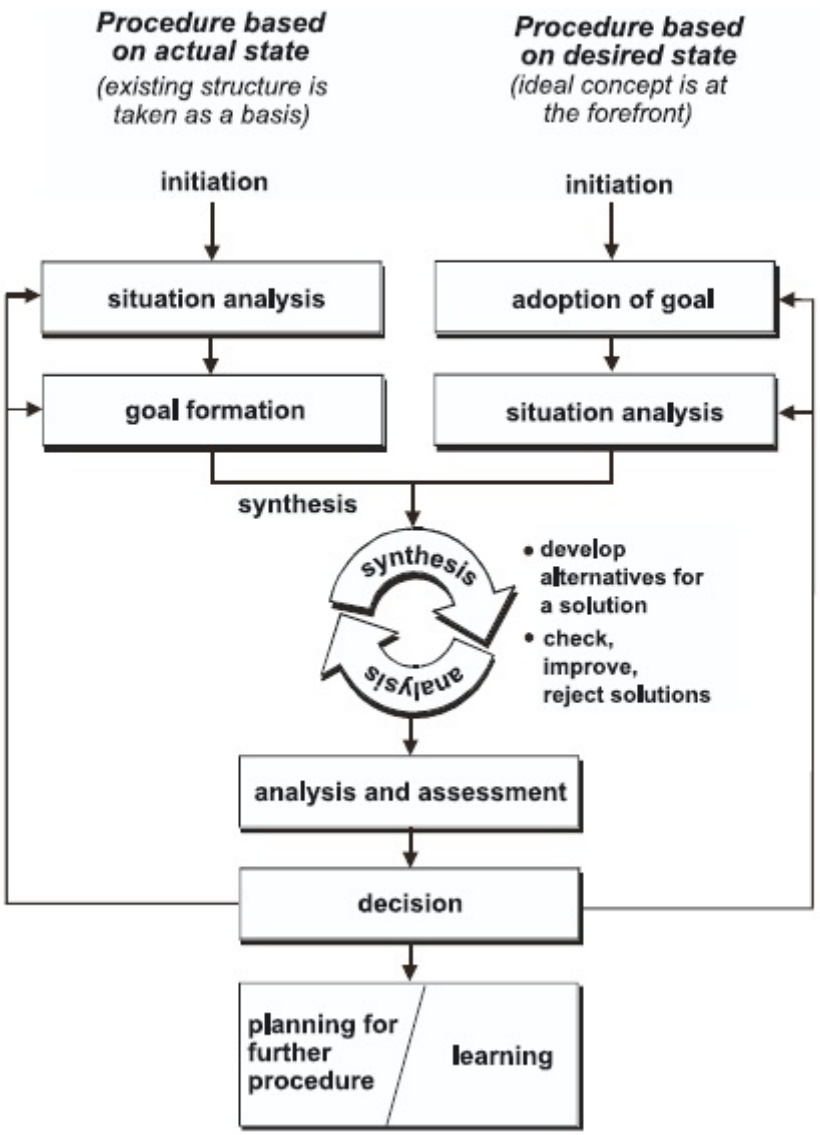


Figure 1.1: Problem-cycle as a micro-cycle

1.5 Formal Requirements

To summarize, the requirements set by Leoni's engineers can be concluded as follows:

- Vehicle's payload : 40 kg maximum.
- Maximum speed of the vehicle: 0.5 m/s
- Maximum track-width:1 m
- Working hours: 1 hour
- Maximum range of obstacle detection: 0.5 m
- Minimum height of obstacles that could be detected: 5 cm
- Minimum height of the vehicle from the bottom to the ground: 0.5 cm
- Minimum height of the vehicle from the top to the ground: 30 cm
- Minimum range of the network: 60 m
- Accuracy of vehicle position on the line: 5 cm
- Number of components supported by the ordering app: 80

The next chapters will discuss thoroughly the phases of designing and implementing every single module in our system to meet the gathered requirements.

1.6 Adjusted Requirements

In April 2018, we organized a meeting with Leoni Operation Management Department to view our progress. The meeting concluded that Leoni's engineers decided to re-define the type of products that need to be transported. To explain this adjustment, first, the following image in figure 1.2 shows the overall factory outline layout.

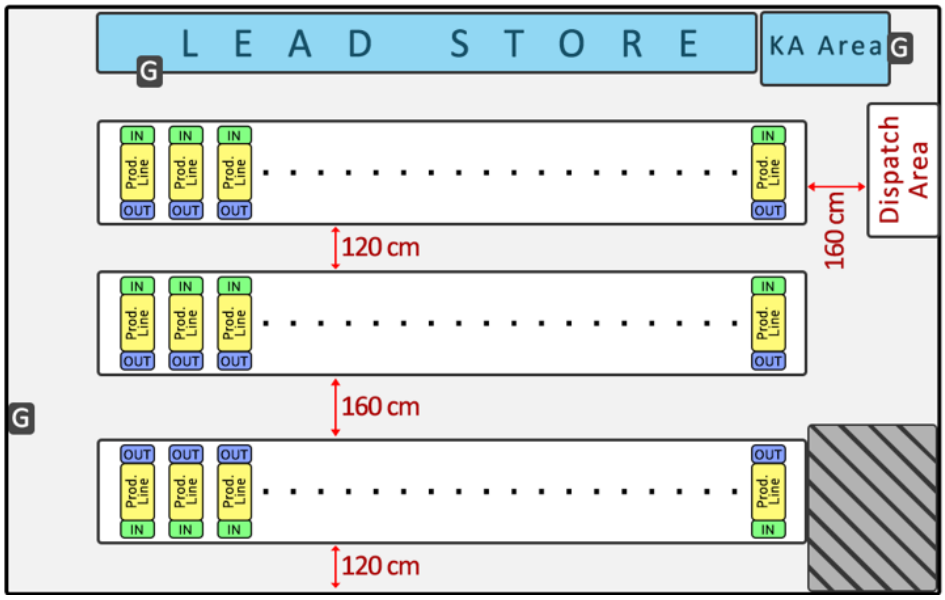


Figure 1.2: Leoni Factory Layout

Leoni's engineer needed one of two implemented solutions:

1. Setting the vehicle path from the inventory station (Lead Store) to the production lines input.

2. Setting the vehicle path from the production lines output to the dispatch area.

The first option included the Kanban bins transportation, present in lead store, that we made our initial design based on the initial formal requirements. The second option is the new decision that we started altering the design based upon the new request. The request has not only changed the path that we were building our design accordingly, the type of products that will be transported will no longer be about Kanban bins. The products that will be at the production line outputs are shown in figures 1.3a and 1.3b:

(a) Plastic Bags

(b) Carton Boxes

Figure 1.3: Plastic Bags and Carton Box

As Leoni's engineers wanted to increase the plastic bags productivity rate more than the carton boxes, we operated based on this choice.

There are 9 production line output stations that produce the plastic bags. Their locations are shown in the following layout demonstration in figure 1.4 :

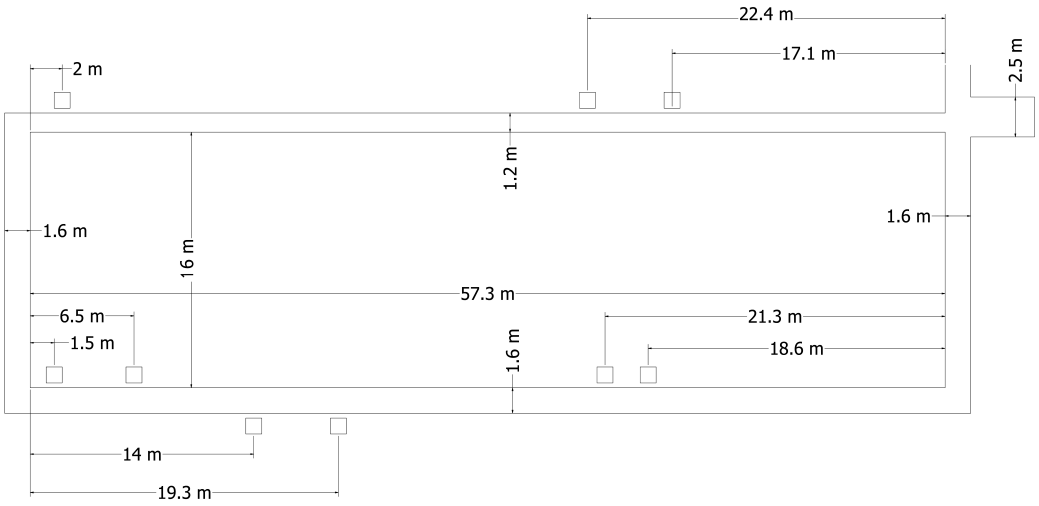


Figure 1.4: Plastic Bags Output Stations

1.7 Funding Agreement

Due to communication issues amongst Leoni's management, the deal has not been officially made, and hence, the funding was not received for the project. However, as we were dedicated to completing the project for learning new technical solutions to an industry-level solution. Mainly taking into consideration to reduce the cost by eliminating unnecessary components that will not affect the robot functionality.

Our progress in the project will be fully shown in the following chapters.

Chapter 2

Robot Mechanical Design Aspect Domain

2.1 Introduction

2.1.1 Problem Review

The overall target function of the robot is to carry up to 40 kg packages, and transport them to different locations inside a factory plant. Thus, an efficient powertrain system, a steering system, and a rigid frame with relatively low weight are required.

2.1.2 New Conceptual Design

As we begin our designs to fulfill the requirements, we began to prepare an initial conceptual design of our vehicle.

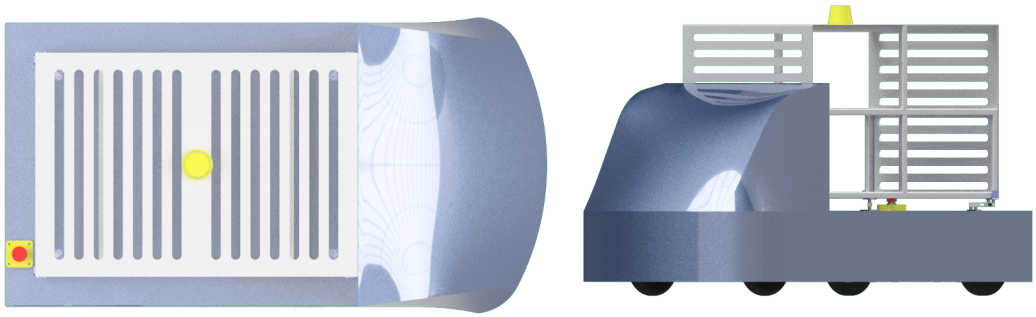


Figure 2.1: new chassis Design Cad (1)

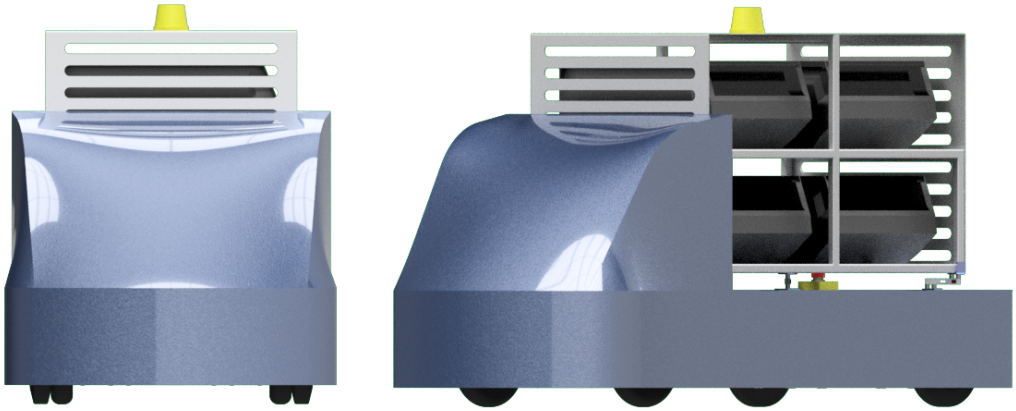


Figure 2.2: new chassis Design Cad (2)

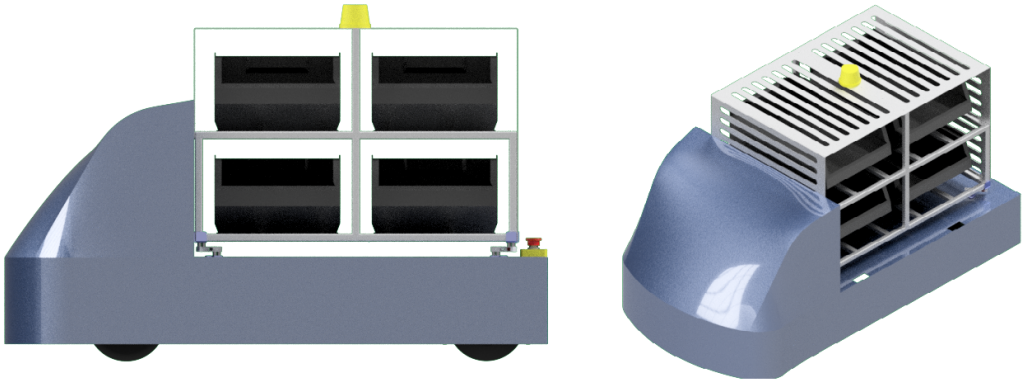


Figure 2.3: new chassis Design Cad (3)

The following sections illustrate the previous used mechanical subsystems, and both of their potentials and defects. Followed by the current mechanical development and enhancement.

2.2 Powertrain System Design Approach

Powertrain module is designed to transmit the power required to move the robot from DC motors, whilst it provides a mechanical connection to encoders so that they will be able to measure the actual wheel speed.

2.2.1 Previous Powertrain Configuration

It is clear that this system is able to transmit motor power to the wheels with a track width of 72 cm, and connects a encoder to measure the speed. However, shaft encoder mechanical shaft is fixed using only one bearing unit, which may result significant vibration on motor encoder side, especially that the used bearing is self-aligning ball bearing and is not fully constrained.

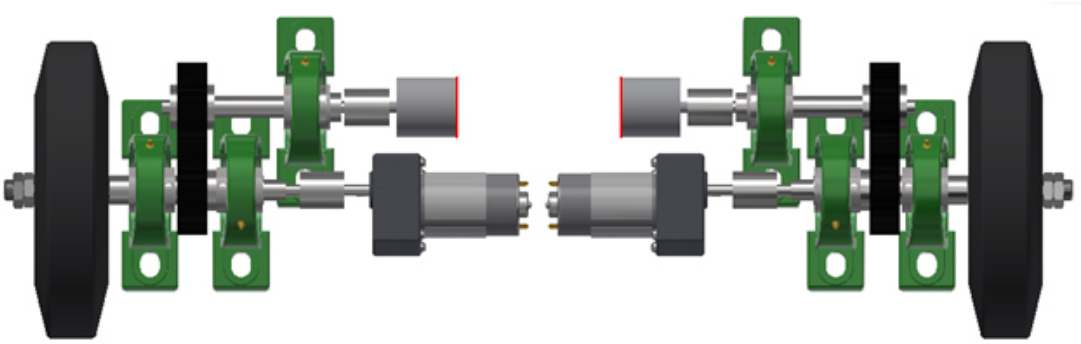


Figure 2.4: Previous Power Train Configuration

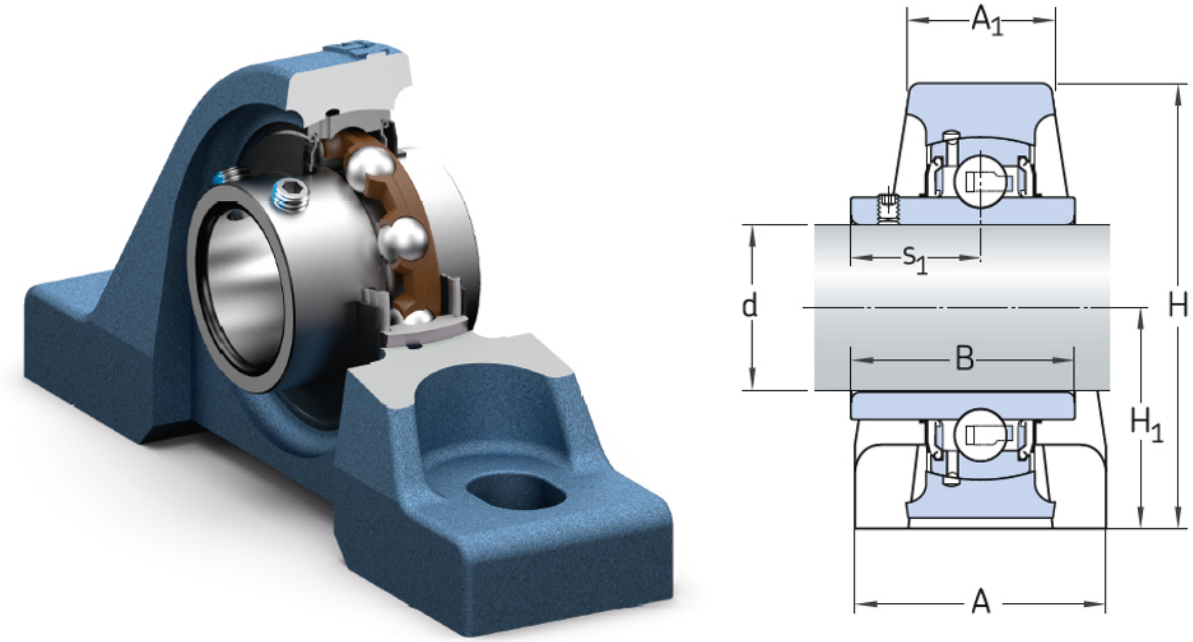


Figure 2.5: Self-Aligning Bearing Block

2.2.2 Current Powertrain System

As shown in fig 2.6 the used bearing pillow blocks are very efficient in supporting robot weight and powertrain shafts. Therefore, two additional blocks are added in the new design to support all shafts in mechanically proper way. The new configuration also reduces the track width from 72 cm to 68 cm.

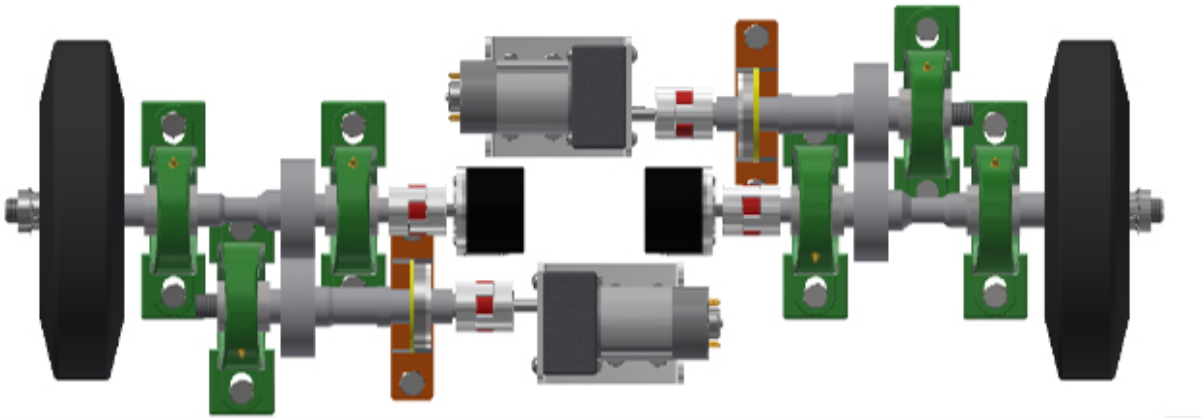


Figure 2.6: Improved Power Train Configuration

2.2.3 Powertrain Stress Analysis

Knowing the loads and required torques, Stresses on each parts were calculated to ensure mechanical satisfaction as shown in figure 2.16 .

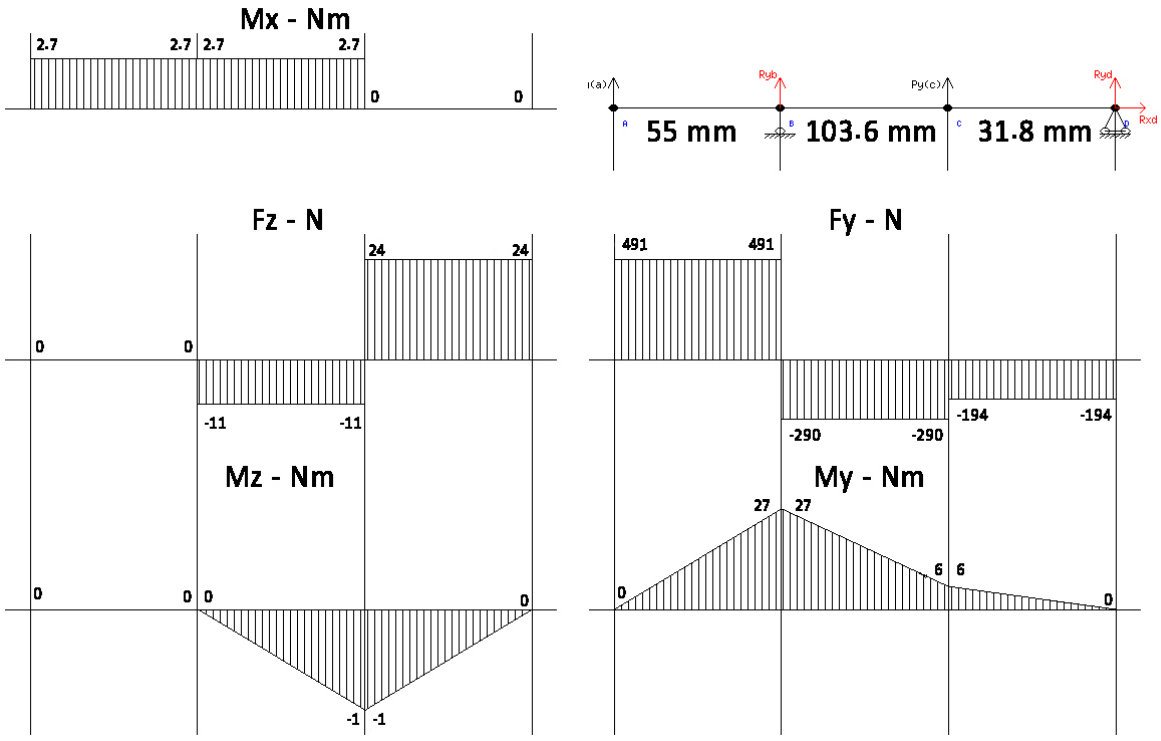


Figure 2.7: Power Train Stress Analysis

Thus, by computing stress at the most critical section at point B: T = 2.7 N.m, M = 27 N.m

$$\tau_{Design} = \frac{\sigma_{yield}}{2n} = \frac{250}{3} = 83.33\ MPa \tag{2.1}$$

$$\tau_{Design} = \frac{16 * Pi}{d^3}\sqrt{T^2 + M^2}$$

$$d^3 = \frac{16 * Pi}{83.33}\sqrt{2.7^2 + 27^2} \therefore d = 2.54mm$$

Gears stress analysis:

Since the following parameters are known:

$$C = 56mm, z = 28m = 2, F_t = 96.34N, F_r = 35N, b = 20mm, K_s = 2$$

$$v = \frac{D * Pi * N}{60000} \tag{2.2}$$

$$K_y = 0.48 - \frac{2.85}{z} \tag{2.3}$$

$$K_v = \frac{3}{3 + v} \tag{2.4}$$

$$F_t = \sigma_{design}\frac{K_y * K_v}{K_s} * b * m \tag{2.5}$$

$$\therefore \sigma_{design} = F_t * \frac{K_s}{K_y * K_v} * \frac{1}{b * m}$$

$$\therefore \sigma_{design} = 13.5328MPa$$

Regarding transmitting power between coaxial shafts flexible coupling were used. The used flexible coupling can transmit maximum torque of 60 kg.cm, while the motor torque is only 27 kg.cm. Thus, it can operate safely.

Figure 2.8: Used flexible coupling

2.3 Steering System Design Approach

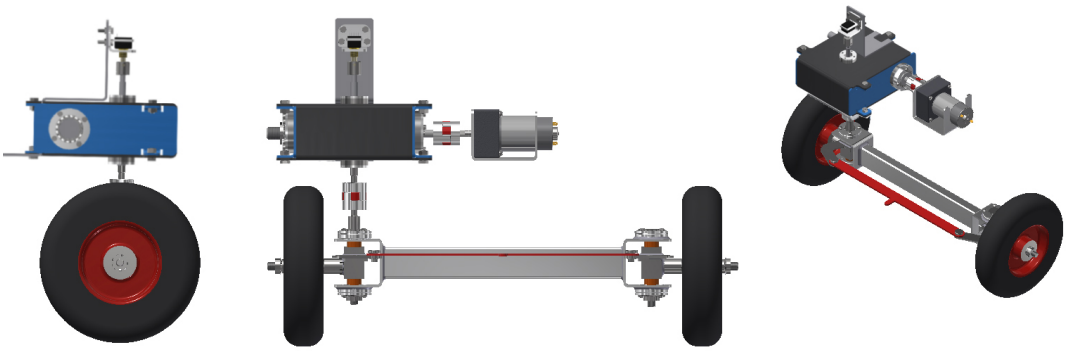


Figure 2.9: Steering system from different views

Steering system is significantly important in mobile robots applications, and different mechanisms can be used. The used steering mechanism is known by trapezoidal steering as it consists of four links. When the robot is moving very slowly, there is a kinematic condition between the inner and outer wheels that allows them to turn slip-free [4]. The condition is called the Ackermann condition. Matching of the kinematics of our trapezoidal steering and Ackermann condition was the main target in designing phase.

2.3.1 Previous Steering System Configuration

Previous year plan was to use differential steering based on varying motors speed, and hence, the robot can maneuver in different directions [1]. The design introduced adding castor wheels as front axle wheels, so the robot can move easily in any direction without any extra torque. However, using these castor wheels resulted severe robot skidding in cornering [1]. That is because castor wheels have two degrees of freedom [1], which makes their behavior unexpected at corners.

2.3.2 Steering System kinematics

Ackermann condition is expressed by [4]:

$$\cot\left(_{\delta o}\right) \; + \; cot\left(_{\delta i}\right) = \frac{w}{l} \tag{2.6}$$

where, δi is the steer angle of the inner wheel, and δo is the steer angle of the outer wheel. The inner and outer wheels are defined based on the turning center. [4]
If C is the mass center of the robot, it will turn on a circle with radius R,

$$\sqrt{a_2^2 + l^2 \cot^2 \delta} \tag{2.7}$$

where δ is the cot-average of the inner and outer steer angles.

$$\cot \delta = \frac{\delta o + \delta i}{2} \tag{2.8}$$

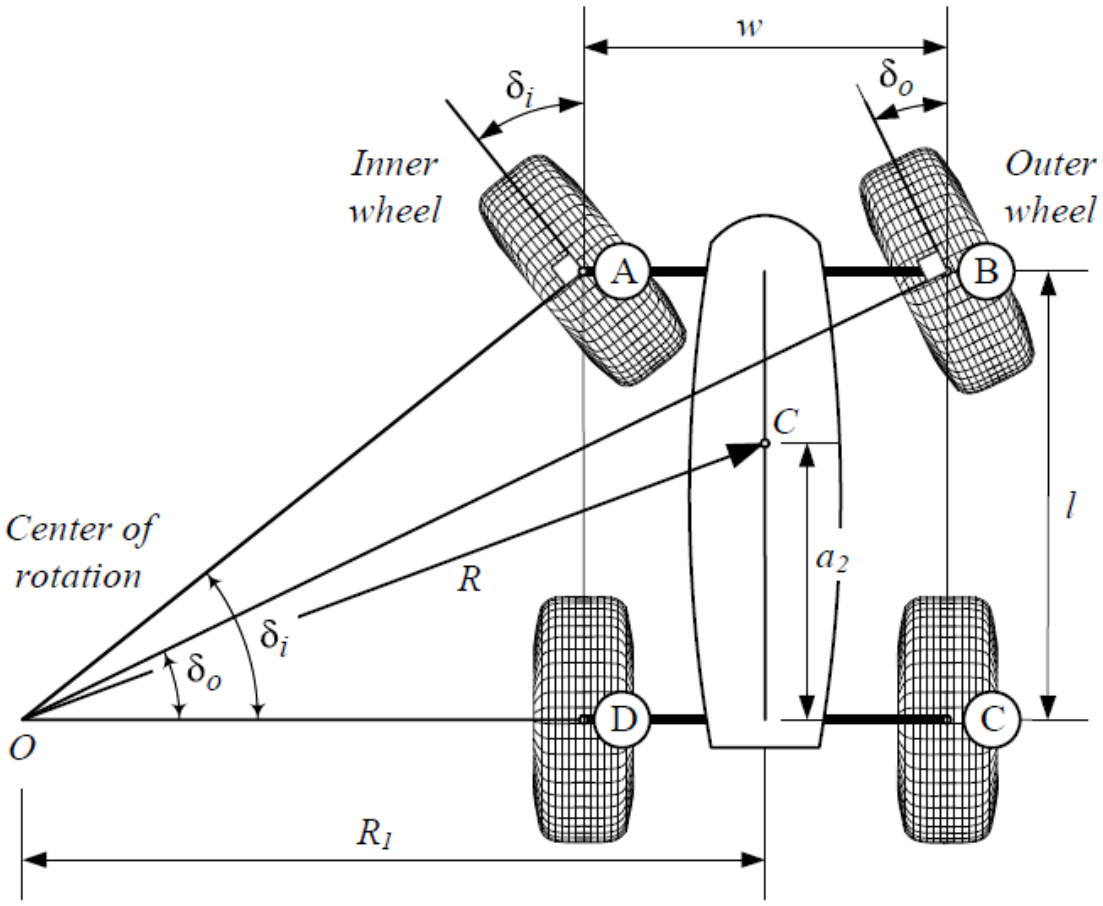


Figure 2.10: A front-wheel-steering vehicle and steer angles of the inner and outer wheels.

However, trapezoidal steering follows the same normal four-bar mechanism analysis. After many iterations of design parameters, we used the following parameters:

Table 2.1: Steering Parameters

Parameter	**Value**
Track width (w)	50 cm
Wheel base (l)	90 cm
Beta angle	69.075 Degree
Drag link (d)	7 cm

To prevent the wheel from touching the rigid frame while rotating, an offset of 65 mm was added between the axis of rotation and the wheel. Therefore, the four bar mechanism parameters will differ than those used in calculating Ackermann condition.
After solving position analysis equation of the above sketch and Ackermann equation using MATLAB, the following results were found as shown in fig 2.13:

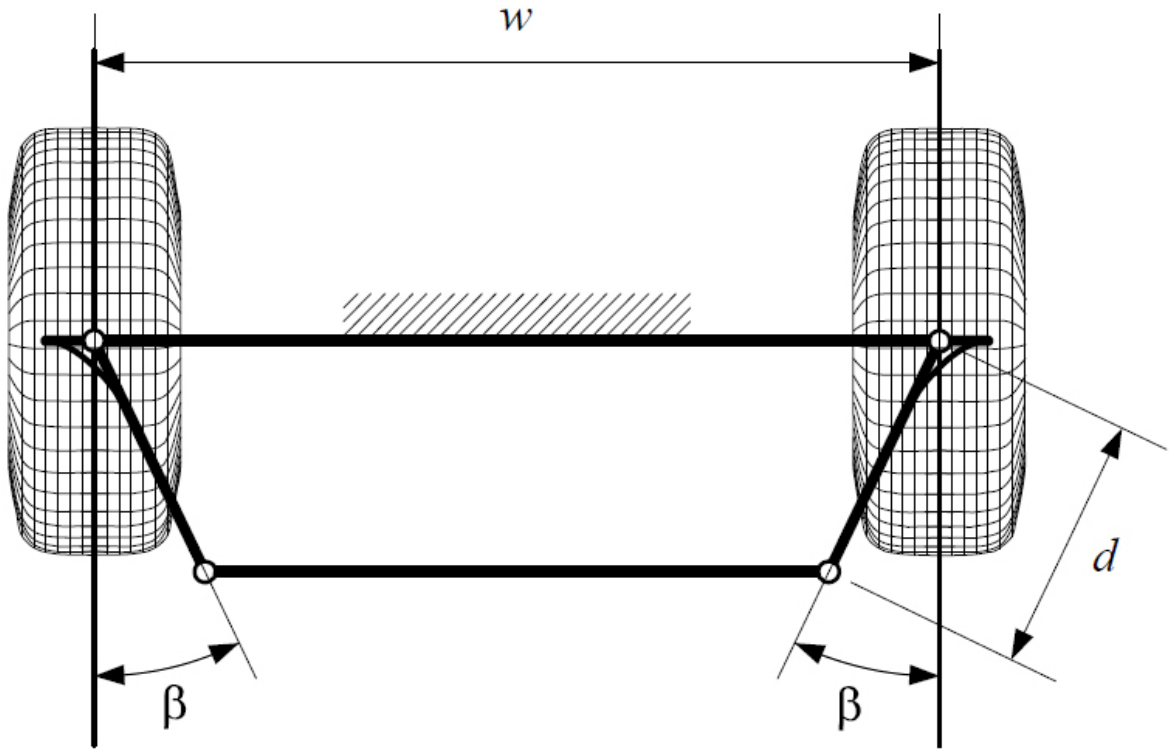


Figure 2.11: A trapezoidal steering mechanism.

Table 2.2: Trapezoidal Steering Parameters

Parameter	**Value**
Ground Link (w)	37 cm
Wheel base (l)	90 cm
Beta angle	69.075 Degree
Drag link (d)	7 cm
Tie rod (t)	32 cm

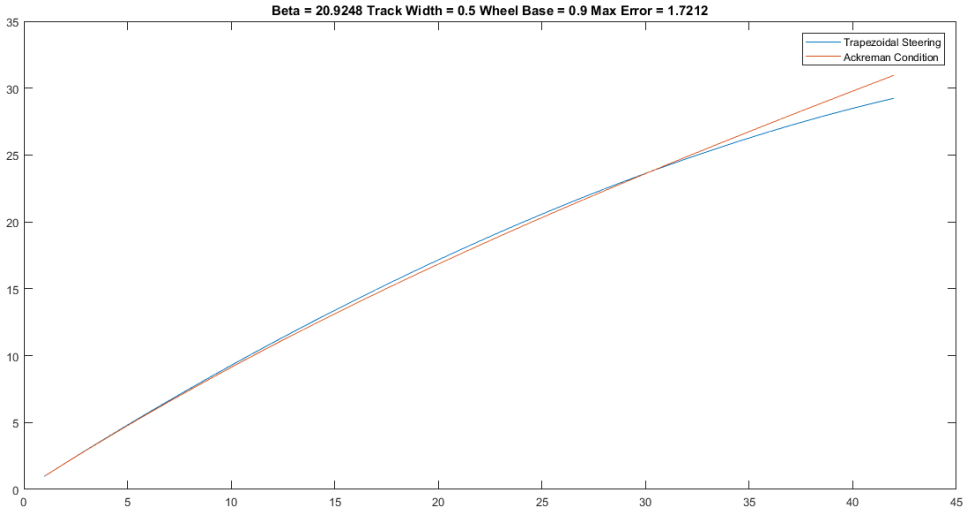


Figure 2.13: MATLAB results, Trapezoidal Steering Vs Ackermann Condition

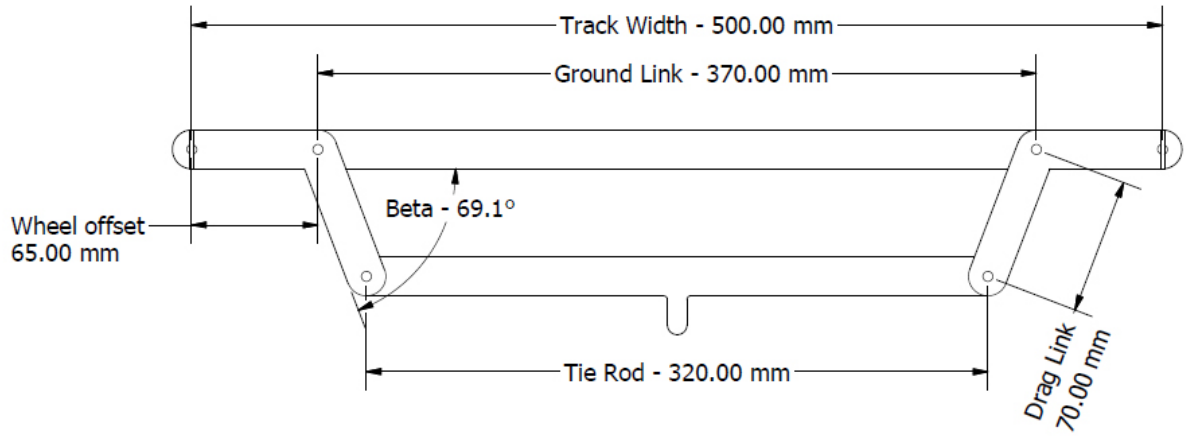


Figure 2.12: A trapezoidal steering mechanism.

Thus, our steering system is capable of achieving Ackermann condition with relatively small error. As the best fitting occurs at inner angle of 30 degree, the resulting steering radius of will be 1.86 m. MATLAB used program is available in google drive reference folder [10].

2.3.3 Caster Angle

Caster Angle is the angular displacement of the steering axis from the vertical axis of a steered wheel measured in the longitudinal direction. Adding positive caster angle makes the robot easier to drive and improves its directional stability. It was implemented in a C section as shown in figure2.14 , which connects the ground link with the wheel hub.

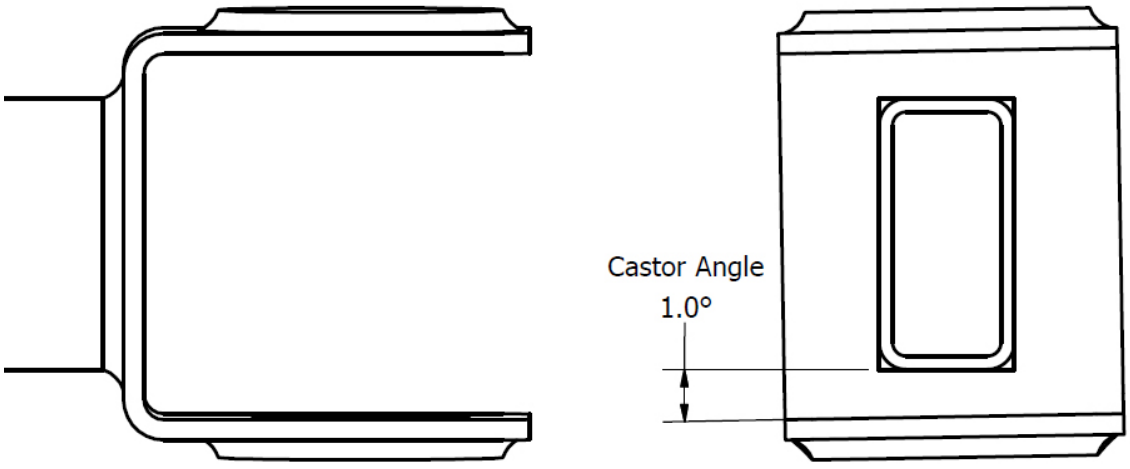


Figure 2.14: The drawing shows how the caster angle is manufactured in the steering system

2.3.4 Steering System Gearbox

To steer the robot smoothly, the used actuator should have extremely small output angular velocity. Since smallest available geared motor rpm is 70 rpm, which does not fit this application, additional reduction ratio is necessary. Worm/Worm gear unit, as shown in figure 2.15, can provide high reduction ratio within a small size unit. Consequently, the gearbox designed for steering system has the following parameters:

- The gear box reduction ratio is $62/3 = 20.67$
- The module = 1.8 mm

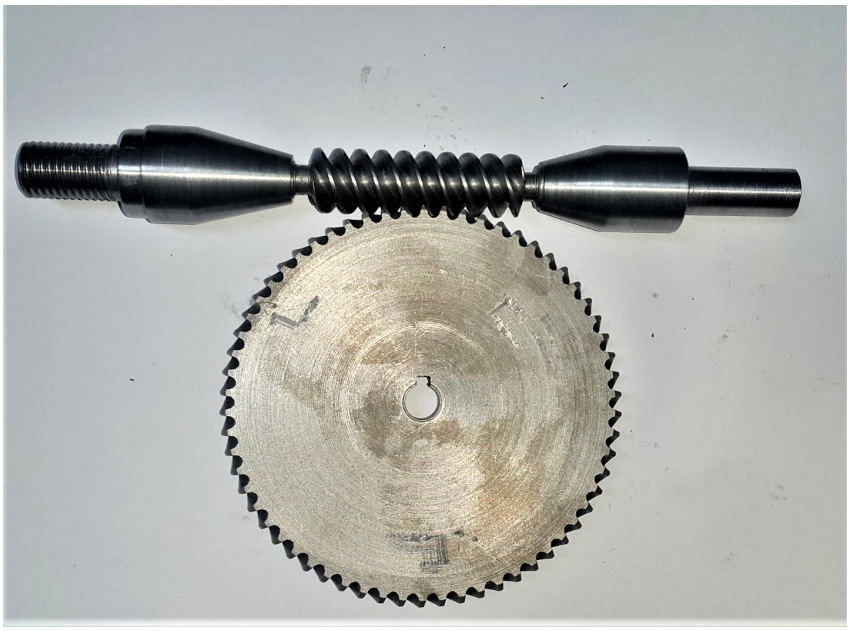


Figure 2.15: Worm and worm gear after manufacturing

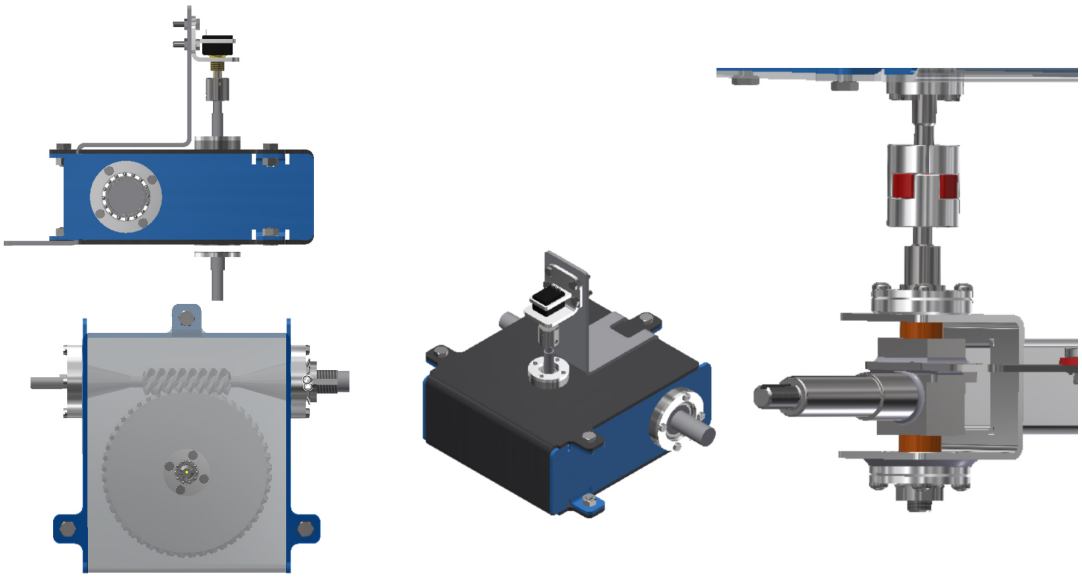
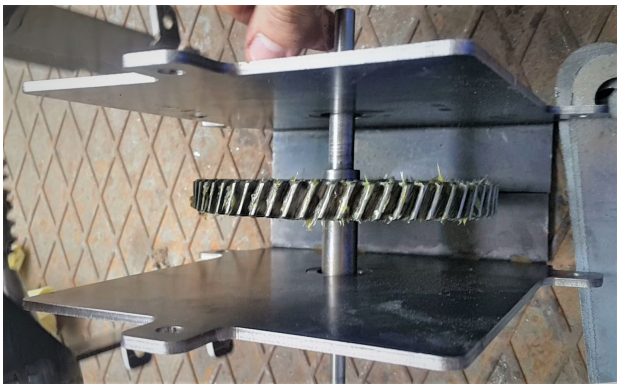


Figure 2.16: Designed gearbox from different views and its output shaft.

We reached the aimed gearbox after manufacturing as designed and that is clear as shown in the following figures 2.17a and 2.17b



(a) steering gearbox after manufacturing (1).

(b) steering gearbox after manufacturing (2).

Figure 2.17: steering gearbox.

2.3.5 Steering System Stress Analysis

Using Autodesk Inventor, stress analysis has been calculated on load carrying parts in the steering system as shown in figures 2.18 and 2.19 .

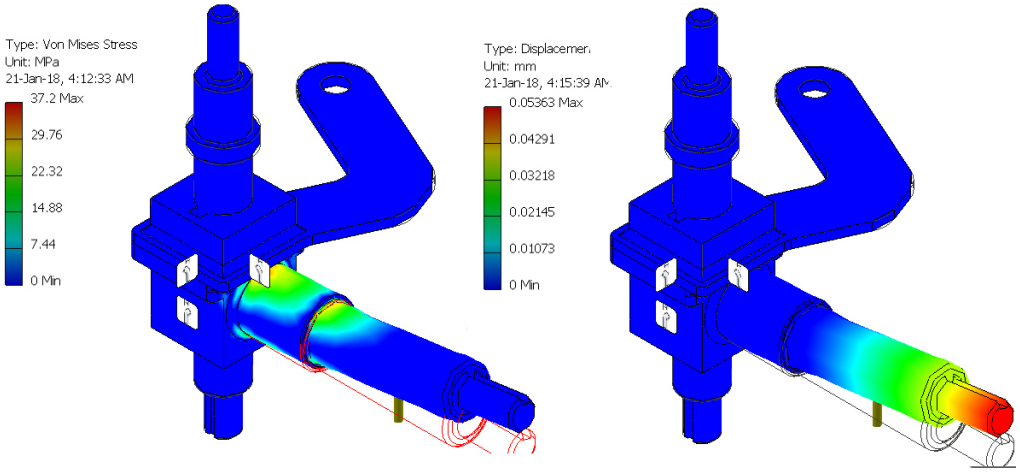


Figure 2.18: Resulted stress and deflection on wheel shaft.

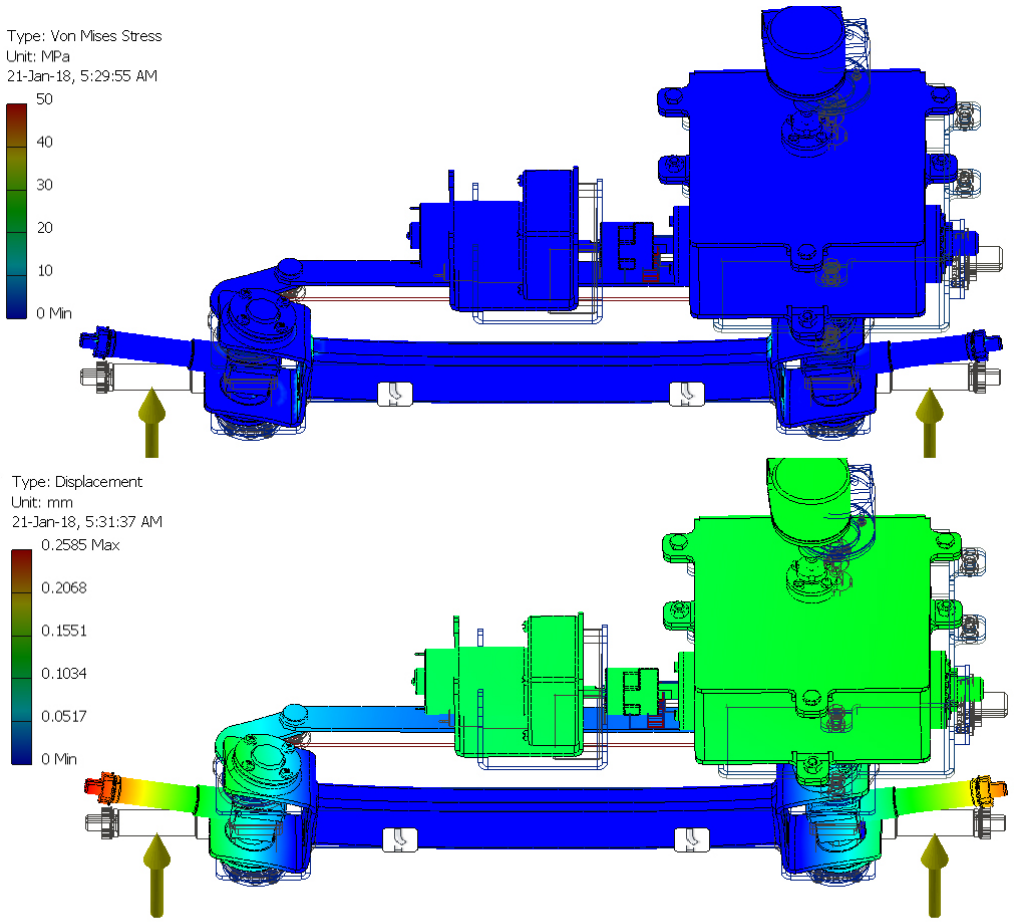


Figure 2.19: Resulted stress and deflection on steering system.

2.4 Chassis Design Approach

2.4.1 Previous Chassis Configuration

Regarding the design of the last year, they approached to make it more space convenient, easy to manufacture and cost-effective, as shown in the figure 2.20.

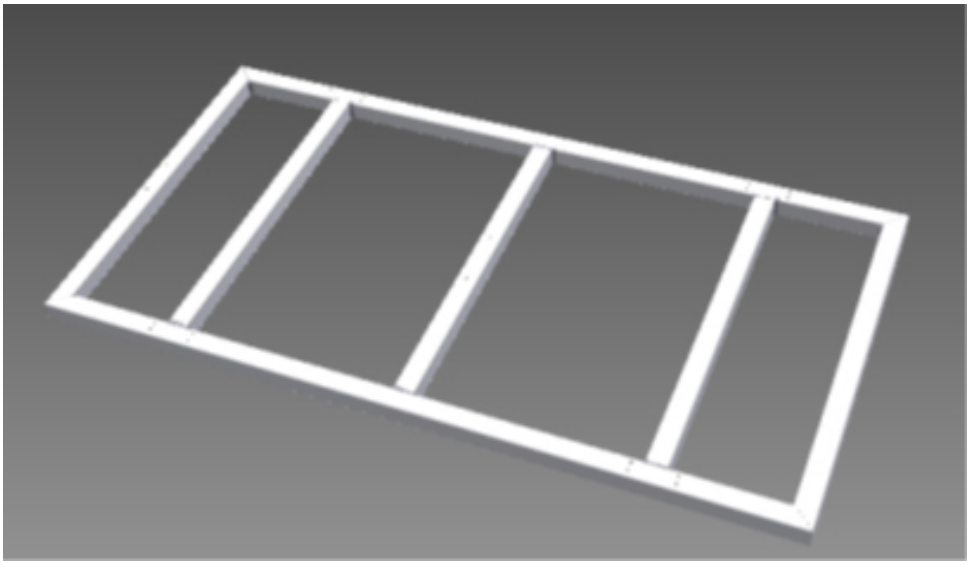


Figure 2.20: Previous Vehicle Frame

However, the stress analysis made shown that the frame faces a great deflection up to 2.16 mm on loading condition.

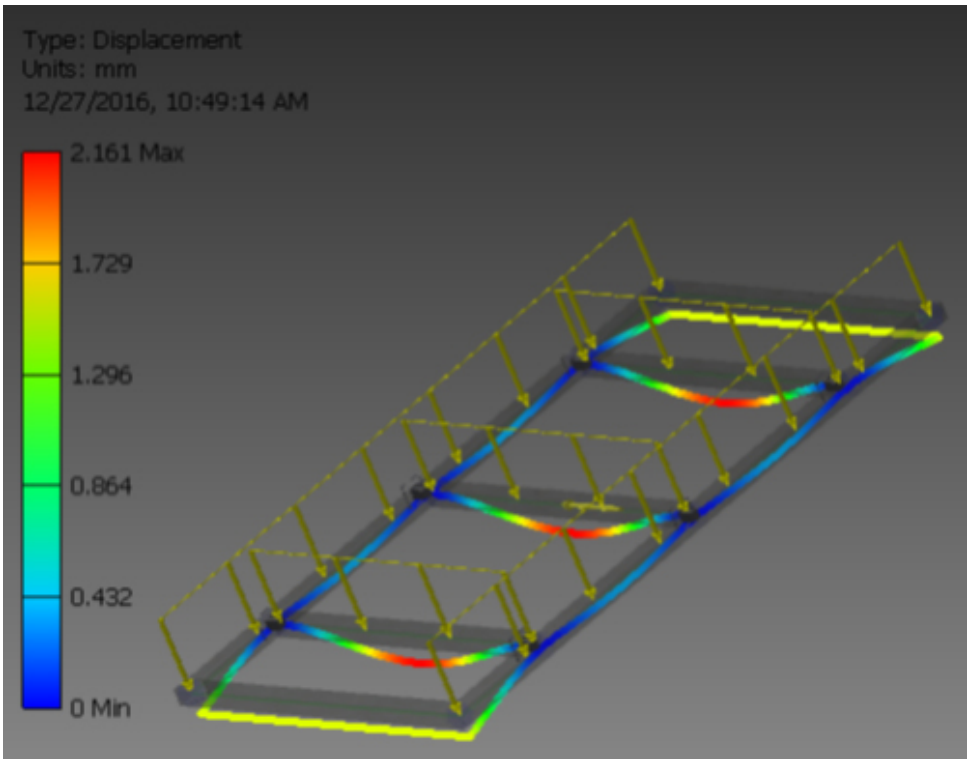


Figure 2.21: Previous Vehicle Frame

Furthermore, they did not consider a way of fixation of the packages, and just relied on putting them on a fixed plate, and that is clear in the figure below.

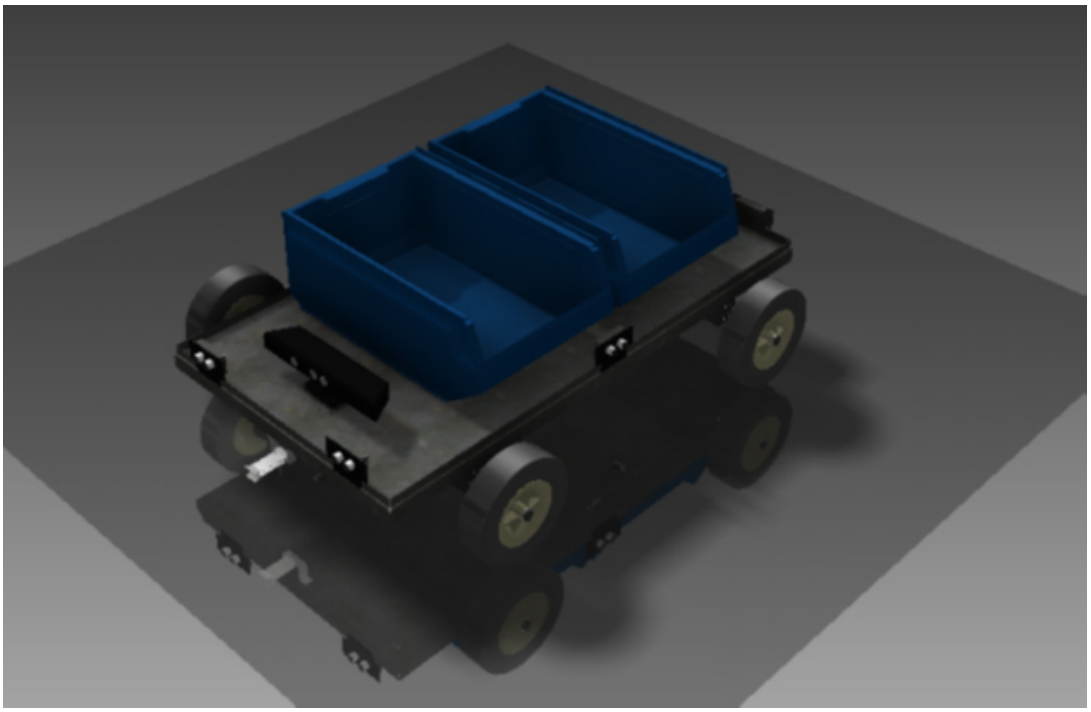


Figure 2.22: Previous Vehicle Kanban Bins Fixation

2.4.2 Current Chassis System

In order to design a chassis which could fit in a real industrial case, we figured out the international companies that manufacturing transportation robot. Eventually we find the robot which we took our inspiration form it, that is shown in the bellowed figures.

Figure 2.23: New Chassis Design Idea

Moving on, we decided to design a space frame chassis rather than the ladder frame they used last year as it not only makes the chassis to be more rigid, but we also could benefit from it in terms of light weight.
Before starting to make our design, we planned the spaces that we are going to use in our vehicle, such as the ECU, Powertrain, etc.

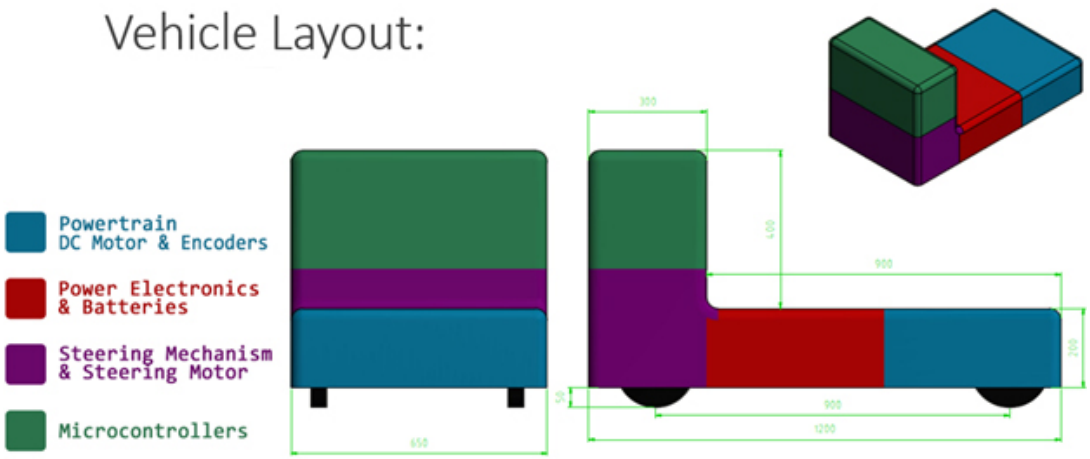


Figure 2.24: Vehicle Layout

After making many design concepts and obtaining inspiration from existing shapes from different international companies, we designed the chassis to be more dynamic and we also removed the extra bars that have not a real effect in terms of rigidity, as shown in the figure 2.25 .
Moreover, in order to solve the fixation of packages problem that they had last year we designed a frame which we could simply put packages in it.

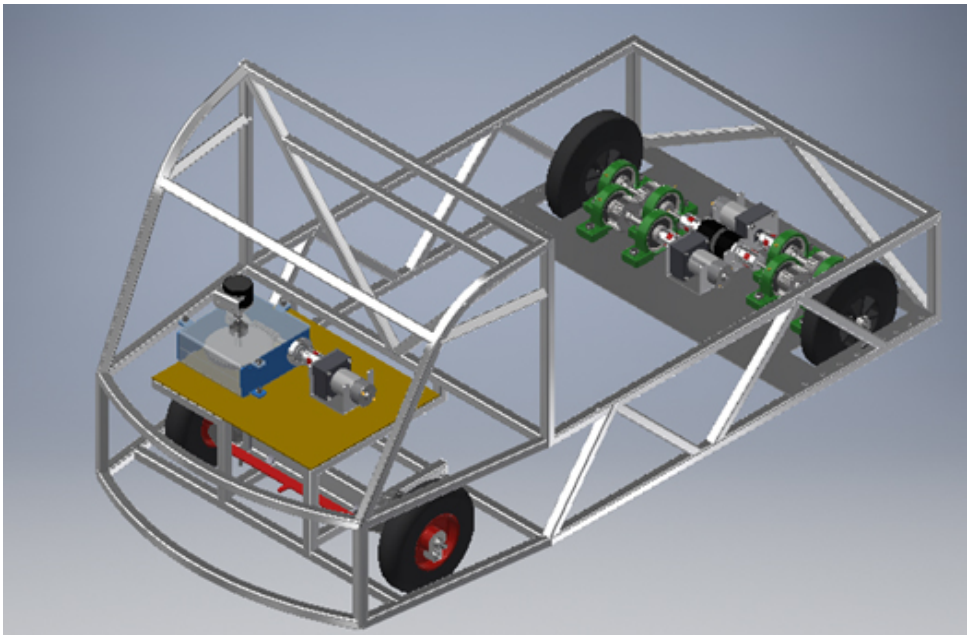


Figure 2.25: Integrated Steering and Power Train on the Frame

The following figure 2.26 shows the aluminum frame after fabrication and how it matches the designed one.

Figure 2.26: new frame after manufacturing

The following figures 2.27 and 2.28 show the aluminum chassis integration after manufacturing

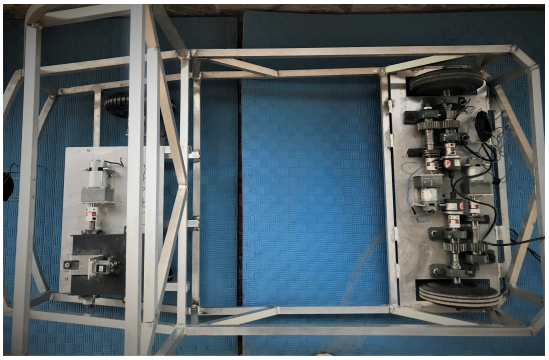


Figure 2.27: chassis integration after manufacturing (1)

Figure 2.28: chassis integration after manufacturing (2)

2.4.3 Chassis System Stress Analysis

There are five basic loads imposes on the chassis structure while it is running, however the most dangerous cases are the pure bending case and the combined bending and torsion case, as they have an extremely effect on the car. As a consequence, we studied firstly the pure bending case and it was clear that the maximum deflection was very suitable.

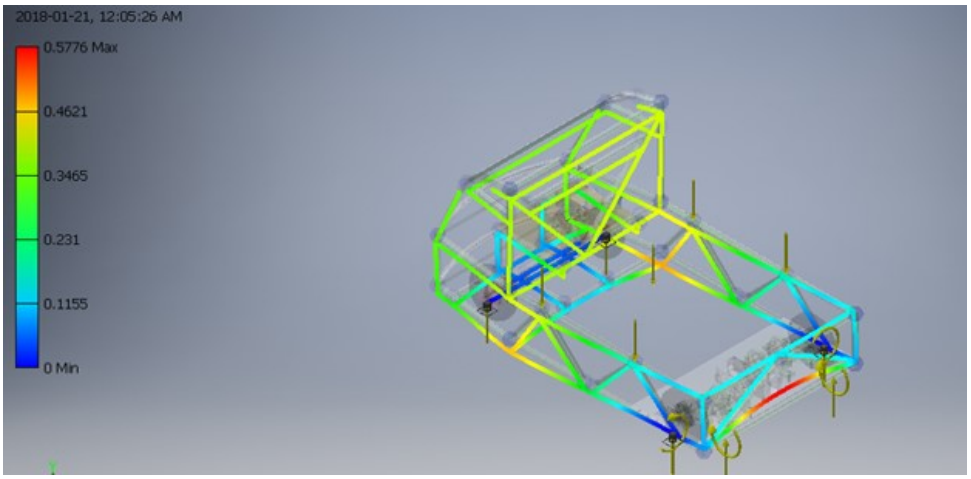


Figure 2.29: Frame Stress Analysis Simulation (1)

After that we made the analysis of combined bending and torsion and the results were the same as the pure bending case.

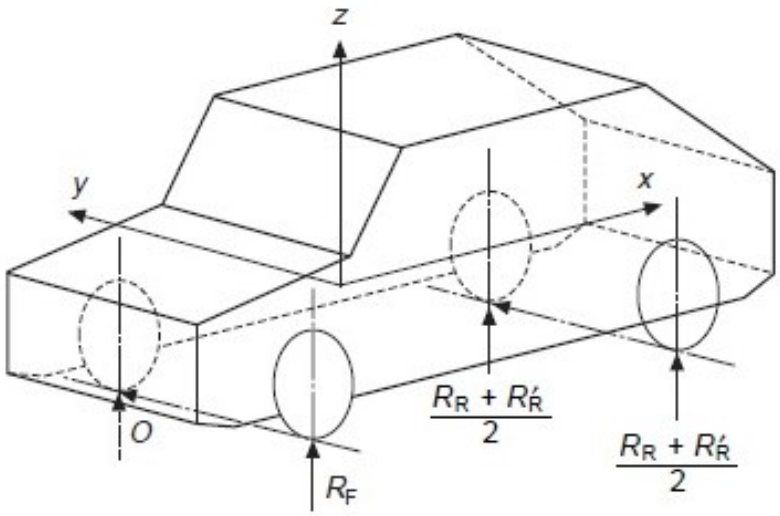


Figure 2.30: Vehicle combined bending and torsion

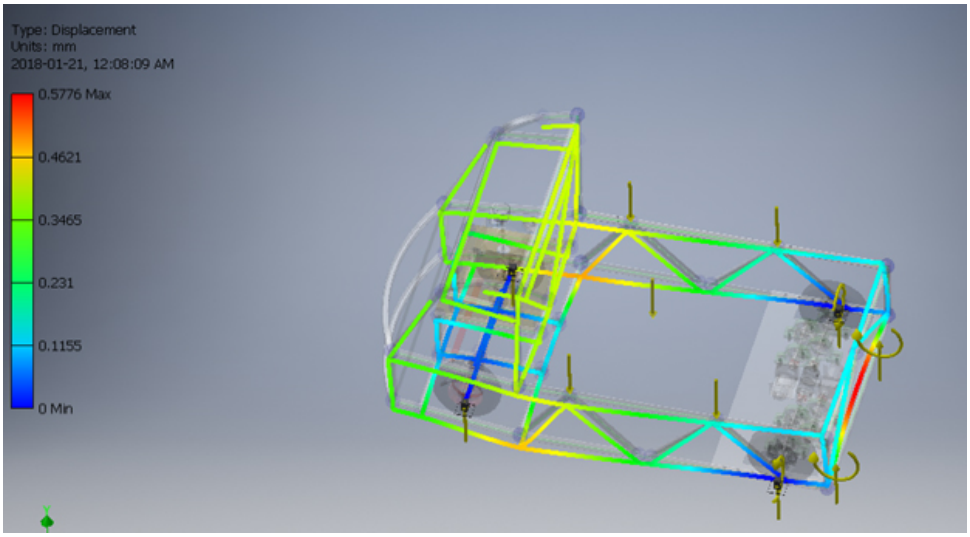


Figure 2.31: Frame Stress Analysis Simulation (1)

Also, we made an analysis to the chassis structure while imposing to lateral load at the worst possible condition occurs when the wheel reactions on the inside of the turn drop to zero, that is when the vehicle is about to roll over. In this condition the structure is subject to bending in the x–y plane. We could calculate them by using the equations that is in the coming figures.

$$Therefore the lateral force at the center of the gravity = \frac{MV^2}{R}\frac{Mgt}{2h} \tag{2.9}$$

$$The side force at the front tyres = Y_F = \frac{Mgtb}{2h(a+b)} \tag{2.10}$$

$$At the rear tyres = Y_R = \frac{Mgtb}{2h(a+b)} \tag{2.11}$$

In our design a= 470.308 mm b= 417.492 mm h= 232.595 mm t= 564 mm M= 60 Kg Therefor the lateral force at the centre of gravity = 715 N Yf=336 N Yr=378 N The result of the analysis due to lateral loading case and combined bending and lateral loading case are shown below.

Finally, after we made a stress analysis on the chassis structure, we analyzed also the Kanban frame while loading it with the 4 Kanban boxes and the results shown below.

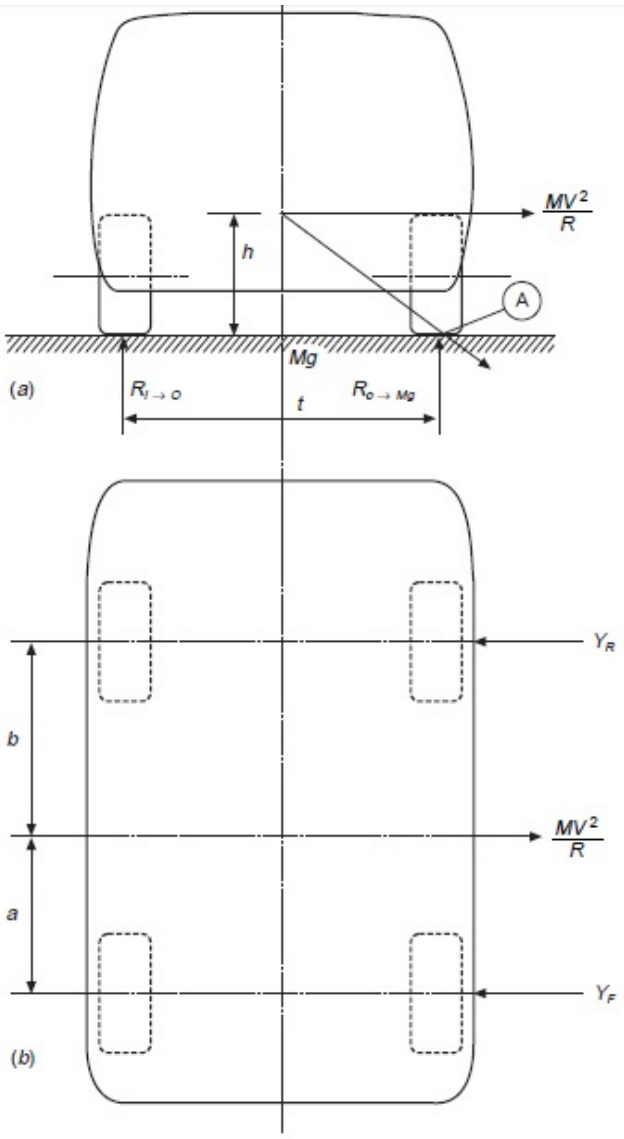


Figure 2.32: Maximum lateral loading

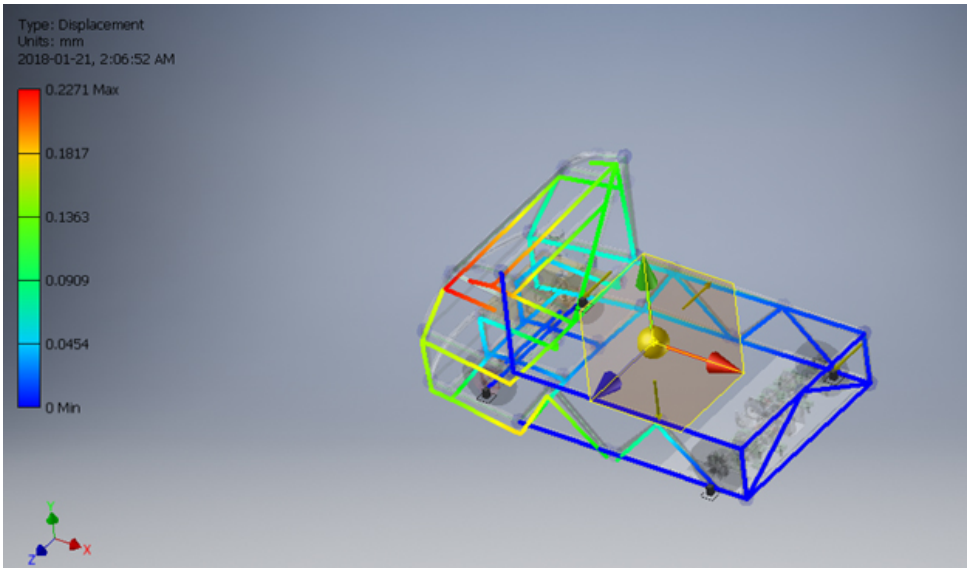


Figure 2.33: Combined Bending And Lateral Loading on the Frame (1)

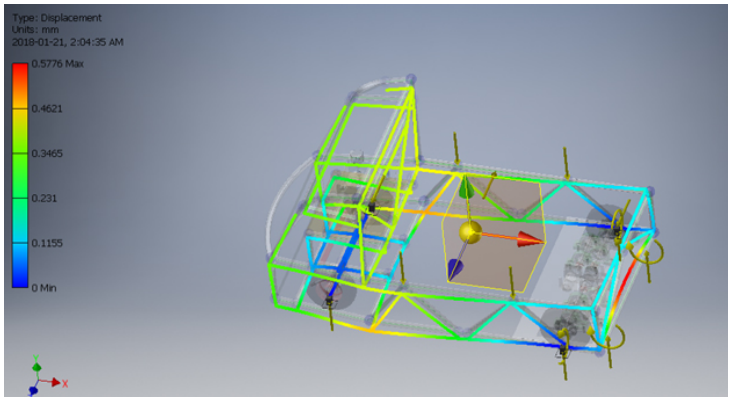


Figure 2.34: Combined Bending And Lateral Loading on the Frame (2)

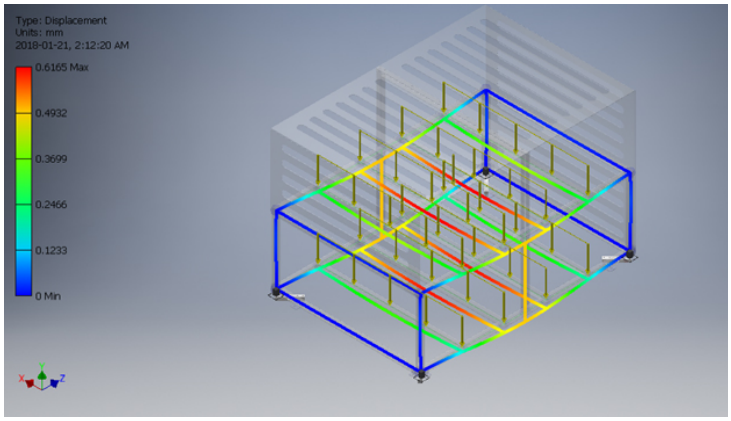


Figure 2.35: Kanban Carrying Frame Stress Analysis

2.4.4 Chassis System Manufacturing

Manufacturing such a space frame chassis requires some form of jig types that act as a practical necessity for fabrication in volume, or when dealing with tight tolerances, or both. With a jig, the parts in a fabrication are held rigidly in the correct location. If everything is cut to the right size, assembly simply becomes a matter of putting everything in the jig and welding it together.

We started out building the space frame without jigs. We noticed an angle error in the manufacturing of the lower layers of the chassis. The initial observational result can be shown in figure 2.36.

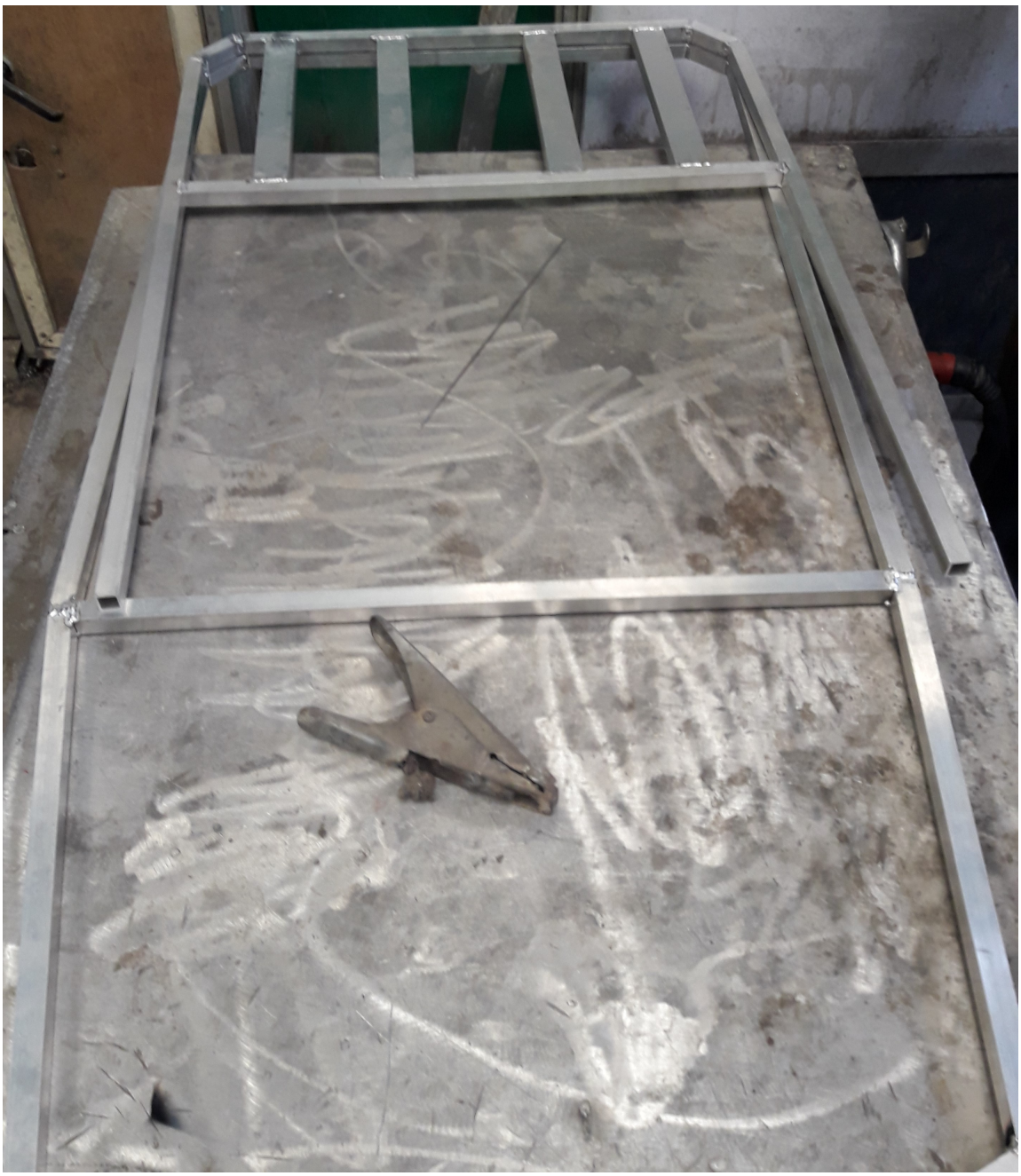


Figure 2.36: Space Frame Manufacturing Error

Jigs are not particularly difficult to construct. We designed a housing-form of a jig to place the aluminum links inside it. The jigs are made out of MDF sheets, constructed using a laser cutting machine. This was the accuracy would match the design that we made.

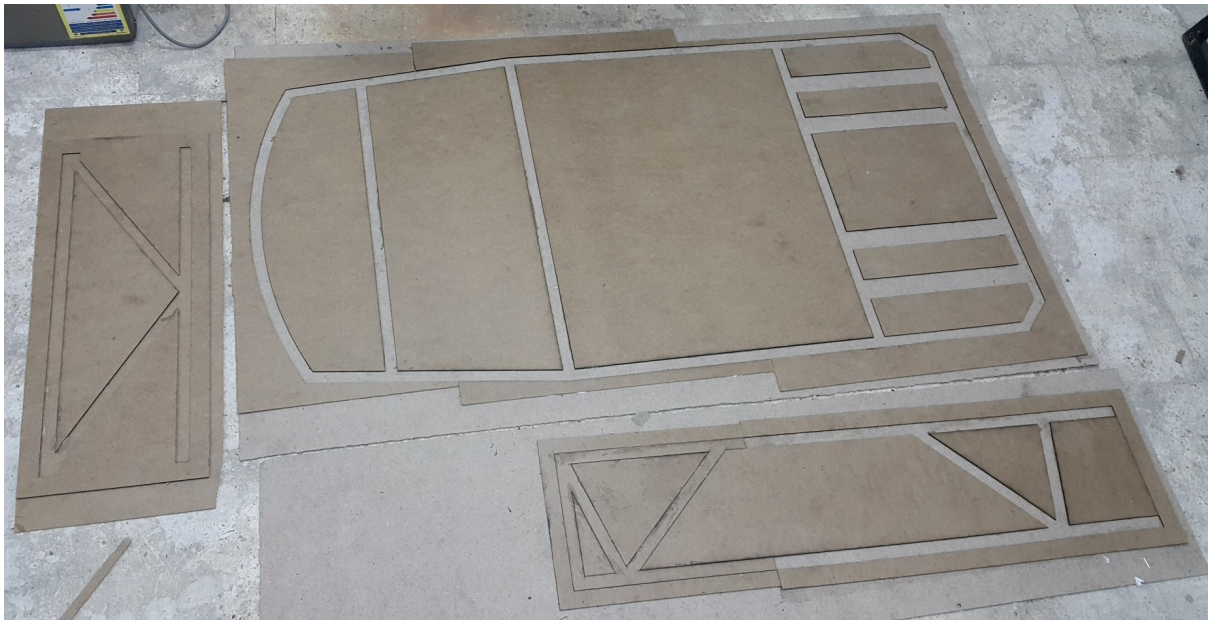


Figure 2.37: Space Frame Constructed Jigs

Using this method, the manufacturing steps can be summarized in the following figures.

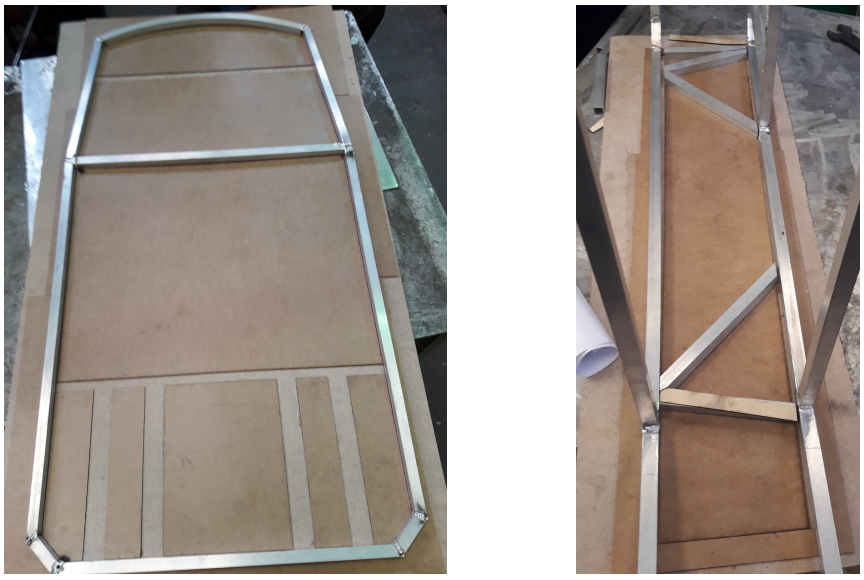
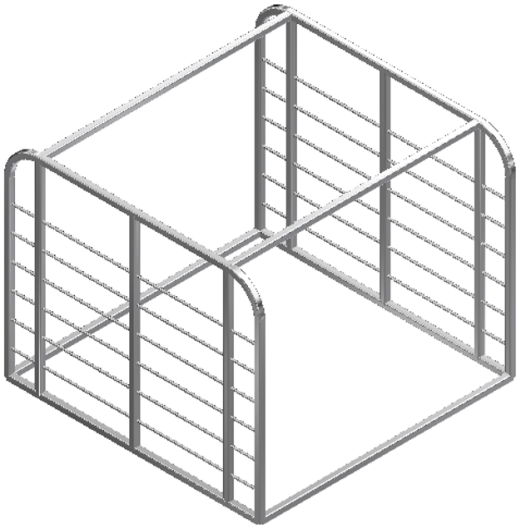


Figure 2.38: Aluminum Links Inside the Jig (1)

2.4.5 Trolley Frame

As the requirement has changed in April 2018, we designed a trolley frame, instead of our original Kanban frame design, similar to the regular trolley used by manual operating workers in the factory as shown in the following figures 2.39a and 2.39b.

(a) Manual Trolley Inside The Factory



(b) Designed Trolley Frame

Figure 2.39: Trolley Frame

The design included two important considerations

- The frame is twice the size of the regular trolley used in the factory.
- The frame is designed to adapt to the adjustable shelves that are used in the existing trolleys to carry the products.

The design specifications of the trolley frame can be summarized as shown in the following table:

width	80 cm
length	80 cm
Height	65 cm
Material	Aluminum
Weight	5.9 kg

Table 2.3: Trolley frame specifications

With a vehicle frame width of 78 cm, this means there is 1 cm offset from each side between the two frames. This will not cause any problems as the minimum width of the track in the factory is 120 cm and the maximum track width is 160 cm. The distributed load along the shelves support beams has caused the following results as shown in figure 2.40. The integrated frame in the vehicle is shown in figure 2.41 and 2.42

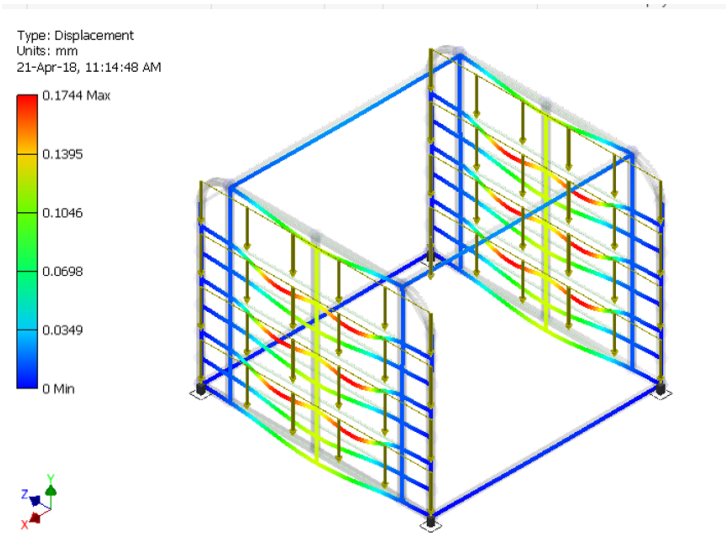


Figure 2.40: Maximum deflection in trolley frame

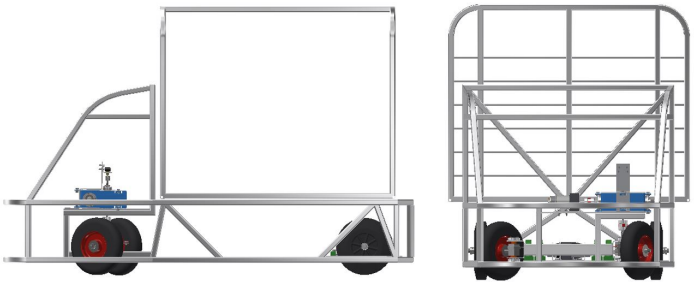


Figure 2.41: Integrated Trolley Frame (1)

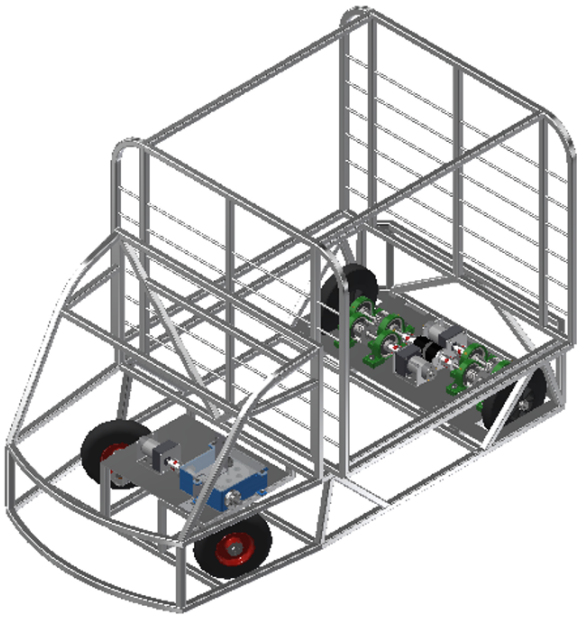


Figure 2.42: Integrated Trolley Frame (2)

2.5 Mechanical Sub-Systems Integration

2.5.1 Powertrain Integration

In order to fix the powertrain in our vehicle we fixed an aluminum sheet metal in the chassis bar first then we fixed the powertrain on it, as in the figures below.

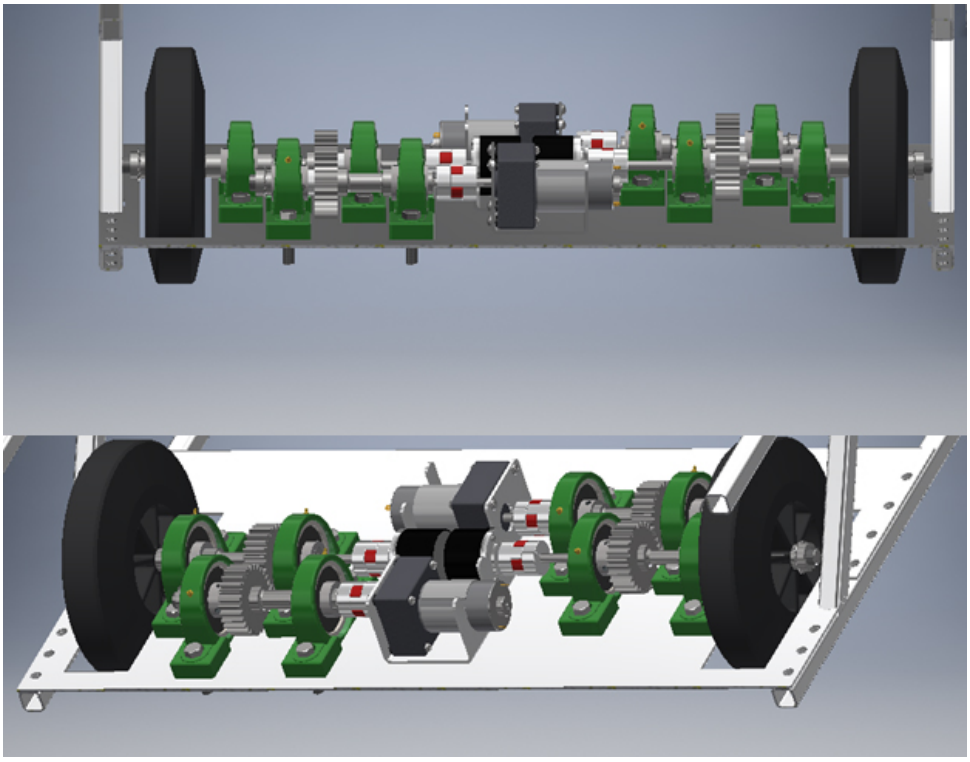


Figure 2.43: Power Train Integration

Figures 2.47 and 2.45 shows the powertrain after manufacturing

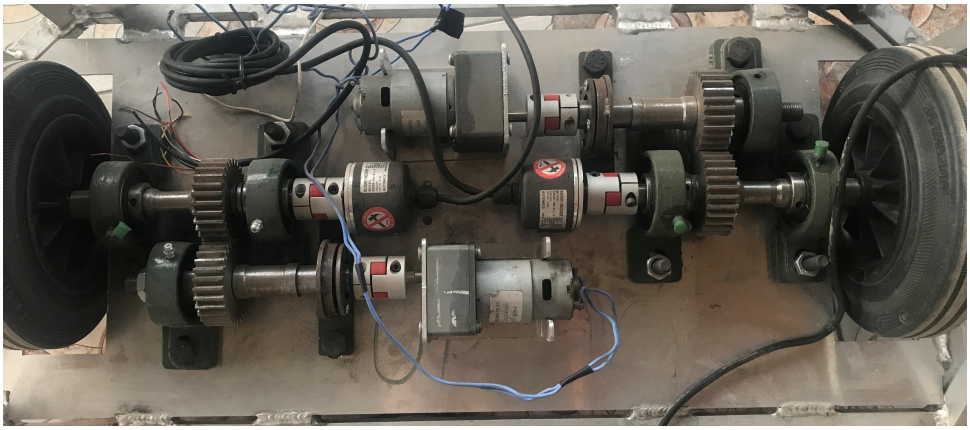


Figure 2.44: Power Train Integration after manufacturing

2.5.2 Steering System Integration

The next problem is to fix the gearbox, steering motor, and the encoder, in which we used a fixed aluminum sheet to fix all the mentioned components in it as it is shown below.

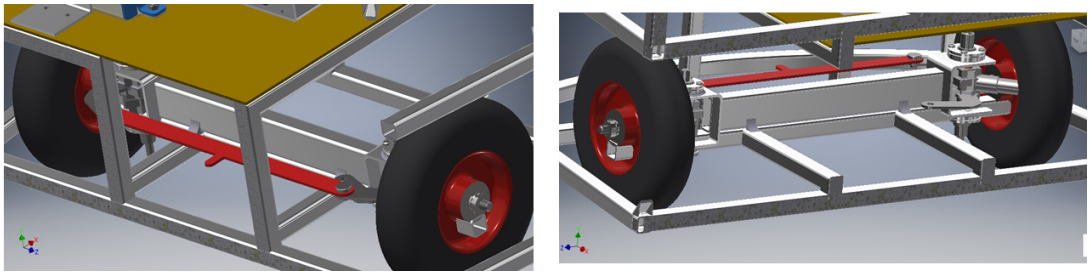


Figure 2.45: Steering System Fixation

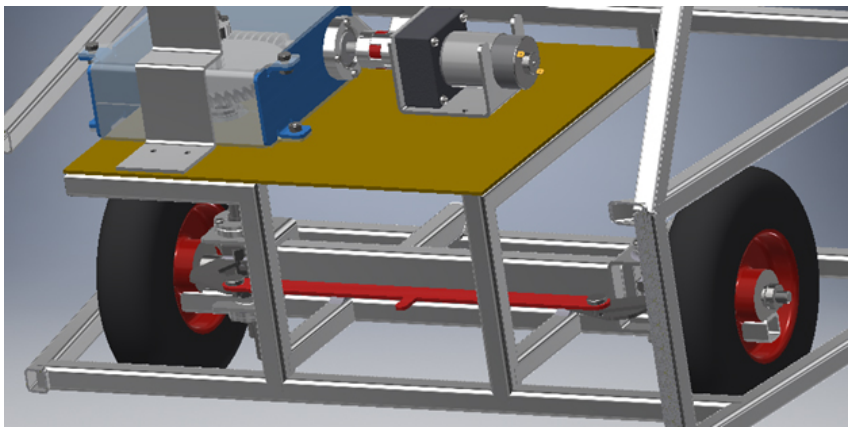


Figure 2.46: Steering Gearbox Fixation (1)

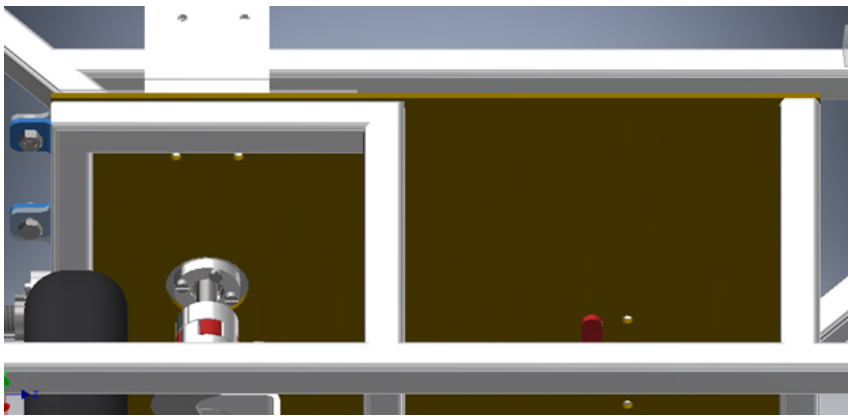


Figure 2.47: Steering Gearbox Fixation (2)

The following figures 2.48 and 2.49 show the steering gearbox fixation after manufacturing

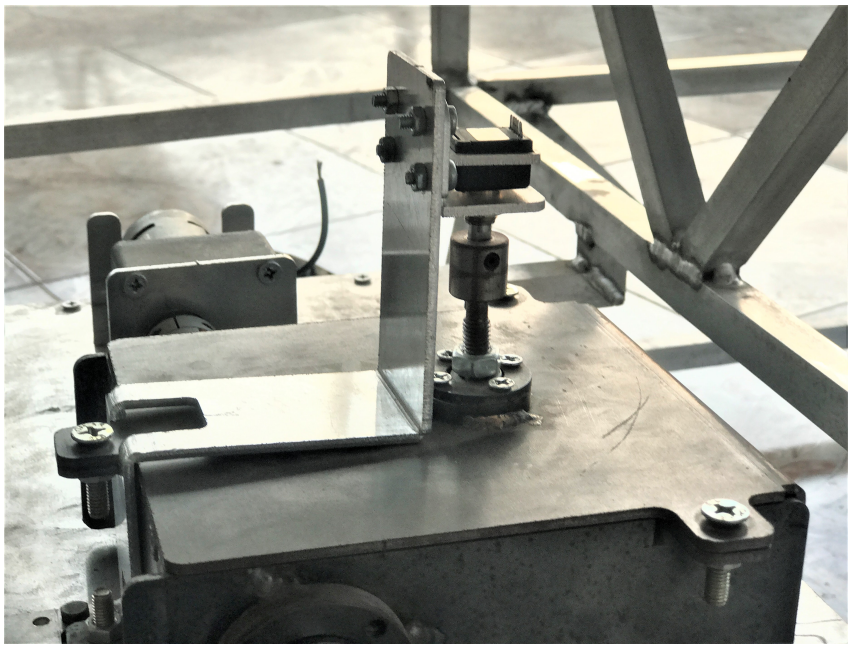


Figure 2.48: Steering Gearbox Fixation after manufacturing (1)

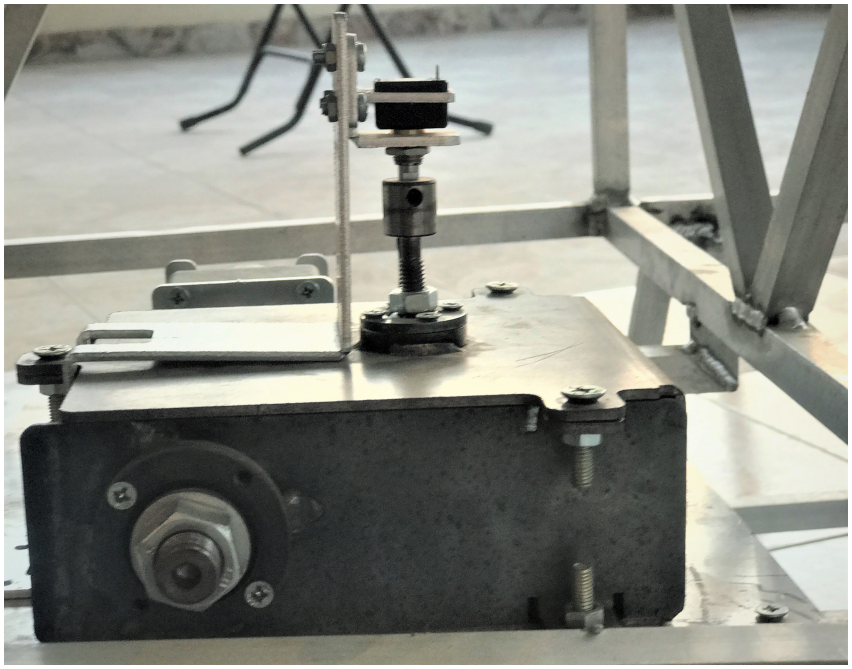


Figure 2.49: Steering Gearbox Fixation after manufacturing (2)

2.5.3 Battery system fixation

The fixation was based on our initial division plan for the vehicle as shown in figure 2.24. Hence, the batteries, fuse box and the motor drivers' boards have been placed together on a 3mm aluminum plate as shown in the following figures 2.50 and 2.51:

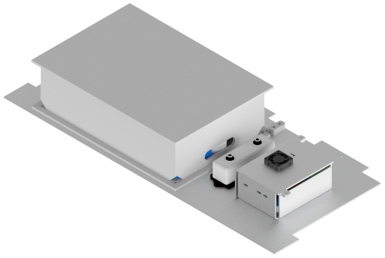


Figure 2.50: Batteries and Motor Drivers Plate

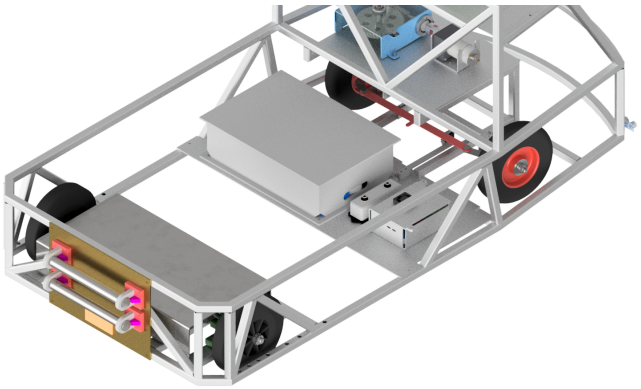


Figure 2.51: Batteries and Motor Drivers Plate Assembly in Vehicle

2.5.4 Electronic control unit fixation

According to fig 2.24 , the electronic control unit is placed at the top of the frontal area of the vehicle. Placed in their own cover box on a 3mm aluminum plate, the fixation assembly can be shown in the following figure 2.52

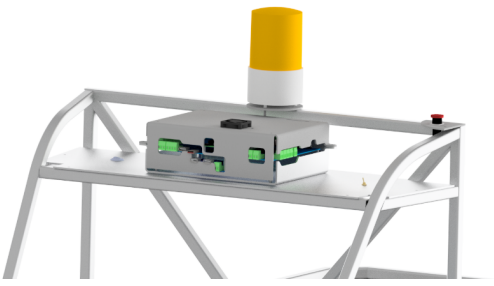


Figure 2.52: ECU Fixation

2.6 Sensors and Indicators Fixation

It is important to show mechanical considerations taken while fixing different sensors in the robot to provide safety and high performance to the robot.

2.6.1 Line Follower Sensor

line follower sensor is designed to be fixed in front of the front wheels in the center of the frame with an aluminum Link as shown in fig 2.53 , 2.54 and 2.55

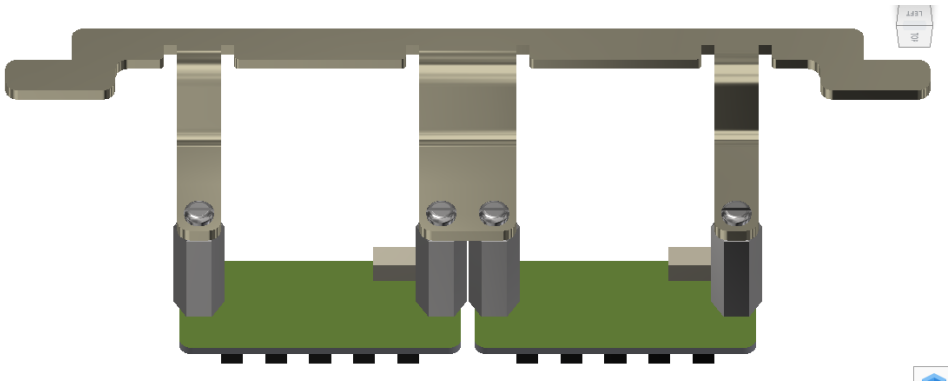


Figure 2.53: line follower sensor fixation design (1)

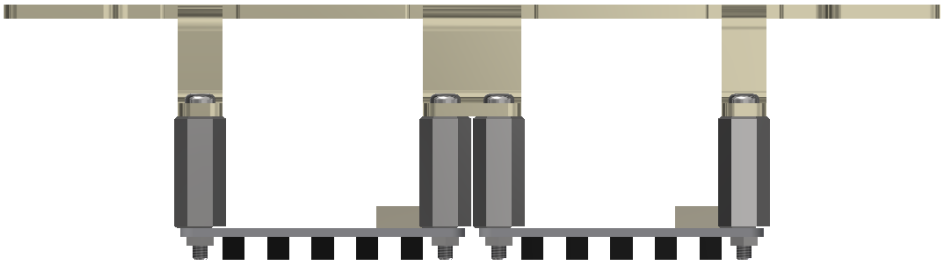


Figure 2.54: line follower sensor fixation design (2)

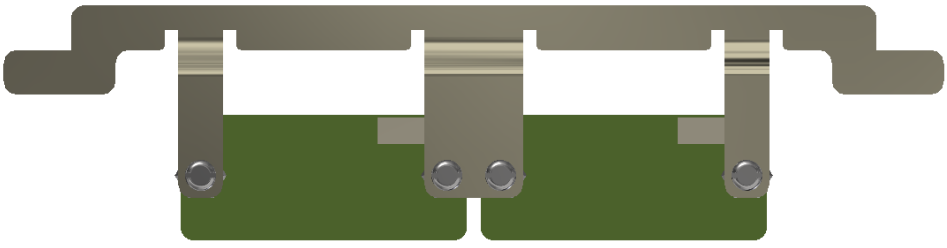


Figure 2.55: line follower sensor fixation design (3)

The following figures 2.56 and 2.57 show the line follower sensor fixation after manufacturing

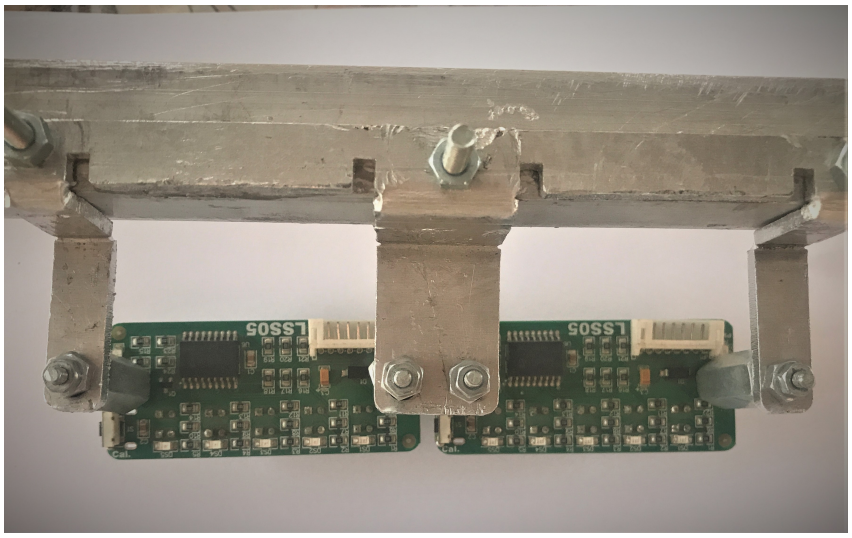


Figure 2.56: line follower sensor fixation after manufacturing (1)

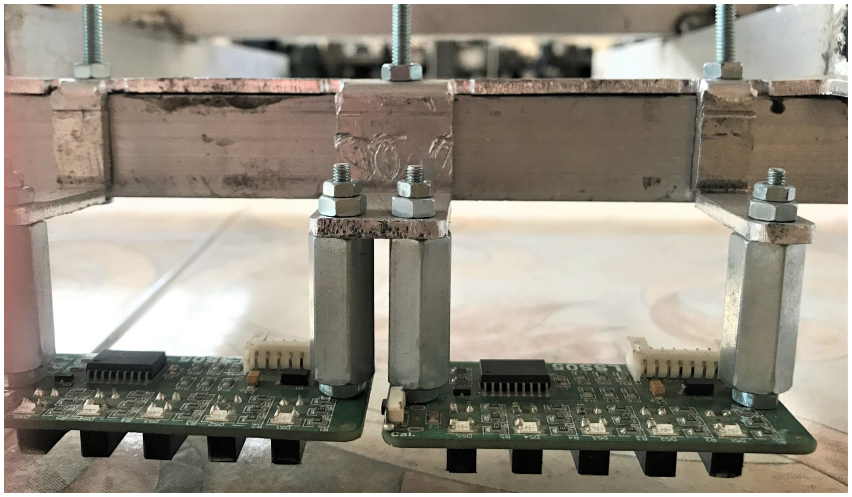


Figure 2.57: line follower sensor fixation after manufacturing (2)

2.6.2 Absolute Encoders

An absolute encoder provides excellent speed and distance feedback and, since there are few sensors involved, the systems are both simple and inexpensive.
As shown in fig 2.58, it is fixed on a L-shape aluminum sheet above the steering gearbox and connected to worm with a coupling.

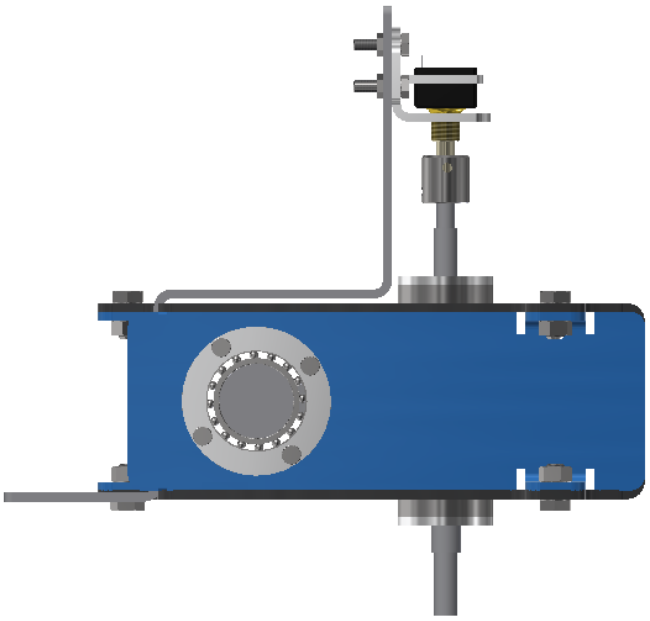
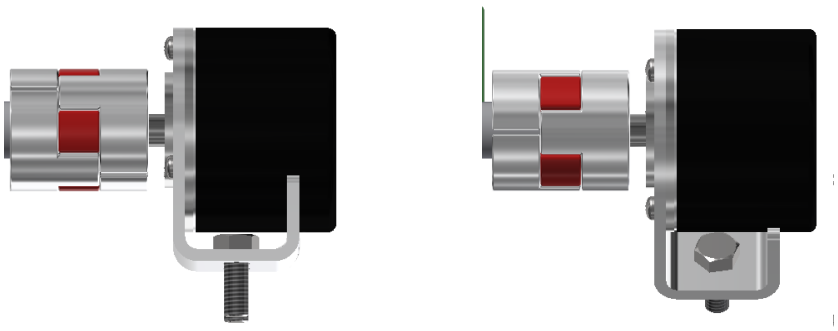


Figure 2.58: Absolute encoder fixation

2.6.3 Incremental Encoder

It is designed to connect the two incremental encoders to the motor shaft with a flexible coupling as shown in fig 2.59b and fig 2.59a



(a) Side View (b) 3D View

Figure 2.59: Incremental encoder fixation

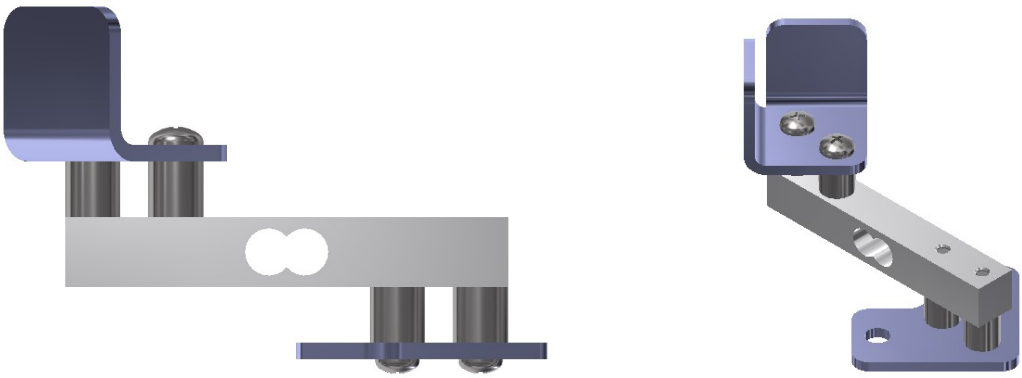
The following figure 2.60 shows the incremental encoder after implementation

Figure 2.60: Incremental encoder after implementation

2.6.4 Load Cells

Four bar load cells are used to measure the weight, one at each corner. The maximum expected measured load is 40 kg plus the frame weight. In order to measure it safely, each one of these four cells is able to measure up to 20 kg, which makes in total 80 kg.

Bar load cell needs to be hooked up between two plates in a "Z" shape, with fitting screws and spacers so that the strain can be correctly measured. This provides a moment of force, or torque, on the strain gauge rather than just compression force, which is easier to measure and much more accurate.
the designed concept achieves that consideration as shown in fig 2.61



(a) Side View (b) 3D View

Figure 2.61: Load Cell

2.6.5 Ultrasonic Sensors

The initial design was to place 8 ultrasonic sensors in our vehicle. However, due to funding reasons, the vehicle is now equipped with only 3 ultrasonic sensors that are placed as shown in shown in figures 2.62a and 2.62b

(a) Initial (b) Current

Figure 2.62: Ultrasonic Sensors Fixation Plan

2.6.6 Steering Limit Switches

In addition to the absolute encoder feedback that reads the steering position actuated by the steering motor, two limit switches were placed to detect the two extreme positions of our steering system. These limit switches detect a signal depending on the coupler rod position. Hence, there are three detection states of the two limit switches as shown in figures 2.63 2.64 and 2.65 :

- Neutral State
- Right State
- Left State

The neutral state describes any position of the coupler arm that neither of the limit switches is reading a signal. The other two states are referenced according to the extreme position that we seek to steer to.

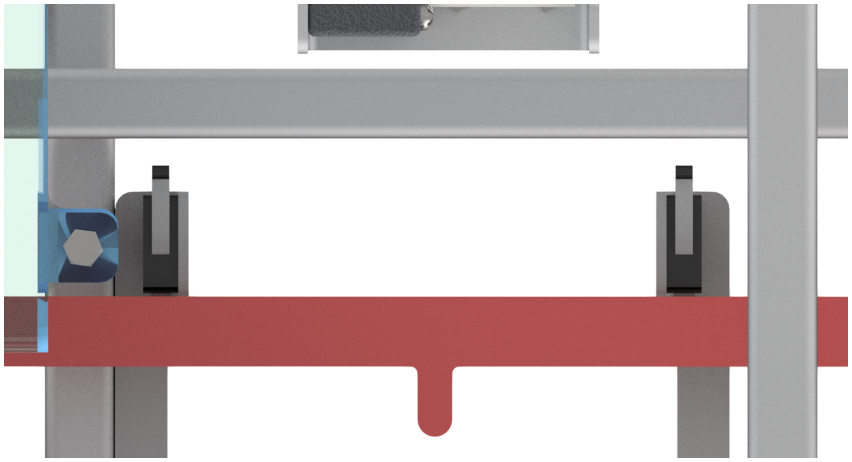


Figure 2.63: Neutral State

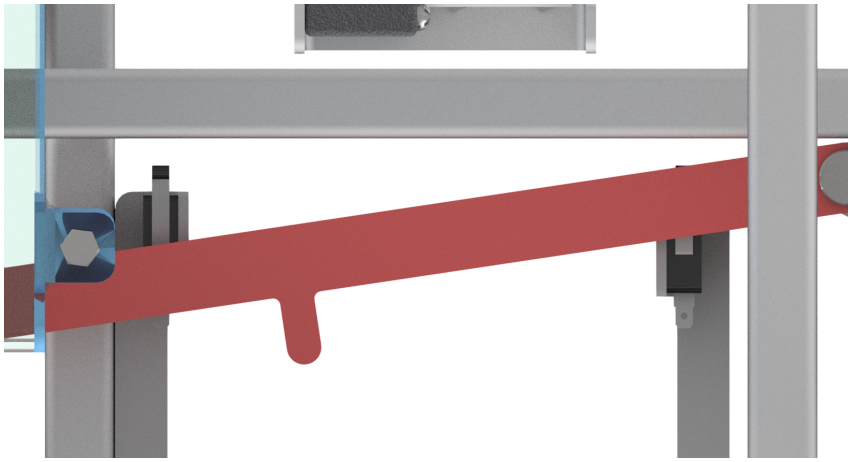


Figure 2.64: Right State

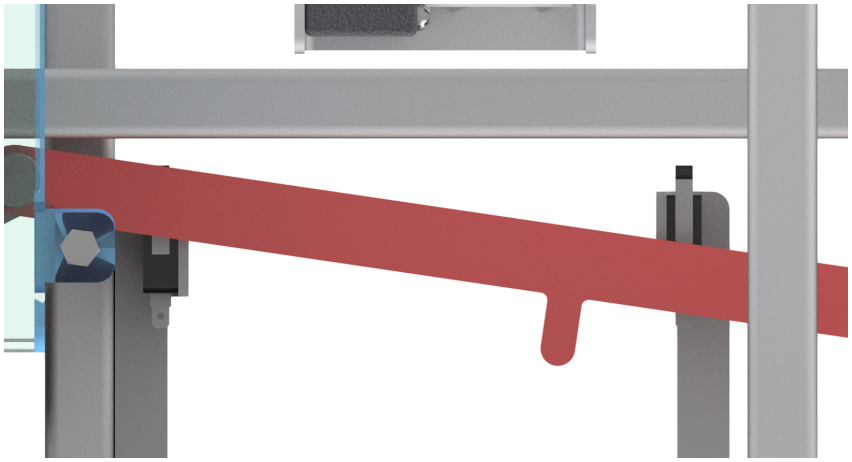


Figure 2.65: Left State

2.6.7 Docking Limit Switches

The limit switch is fixed between two long wooden plates as shown in figure 2.66 in order to ensure that the limit switch will contact the Fixed station when the connecting rods at the vehicle station engage with the fixed connectors in the Fixed station. Also, those wooden plates are fixed in a wooden base which is used in order to fix it in the vehicle station plate.

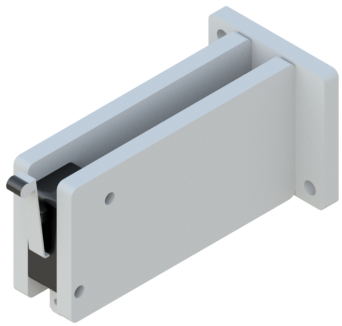


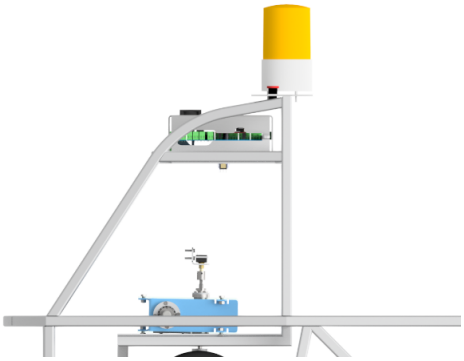
Figure 2.66: Docking Limit Switch Fixation

2.6.8 Operation Lamp

As an indication that our vehicle is operating, we added a 12V lamp at the top of the vehicle to be seen by workers as shown in figure 2.67a and 2.67b . The lamp is triggered to work when the vehicle

- has left the docking station.
- is moving along its path.
- is waiting for the workers to move in the products from the production line to the trolley attached to the vehicle.

(a) Operation Lamp



(b) operation lamp assembly

Figure 2.67: lamp Fixation

2.6.9 Siren Indicator

A siren as shown in figure 2.68 is added to the vehicle to alert the workers in one of two scenarios:

- The vehicle has detected an obstacle that prevents it from moving.
- The vehicle is in docking state for recharging, however the recharging wire in the docking station is not connected to the power socket in the factory.

Both scenarios require immediate intervention from the surrounding workers in either the path lanes or the docking station area.

Figure 2.68: Siren Indicator

Chapter 3

Electrical Aspect Domain

3.1 Electrical System Design Level

3.1.1 Power Train Motors

As discussed before, we will be reusing the old powertrain motors as they have the ratings to withstand a higher load than that of ours. This is due to the lighter weight of the frame and body of the vehicle.

Performance characteristics of a road vehicle [3] refer to its capability to accelerate, decelerate, and negotiate grades in a straight-line motion. The tractive effort developed by the tires and the resisting forces acting on the vehicle determine the performance potential of the vehicle.

As we reused the power train actuators from last year, we did not need to recalculate all the tractive effort derivation formulas. We continued with the existing motors and their ratings.

3.1.2 Steering Motor

The steering motor is used to actuate only the inner steer angle. The outer steer angle can be calculated using approximate method for the designed four bar mechanism. This motor actuates a gear box of a worm and worm gear to reduce the speed of steering and increase the torque. The motor is a similar DC geared motor to the powertrain motors with a shaft encoder feedback to construct a closed-loop control in the steering module.

Initially, the loads acting on the wheels [3] are presented as shown in figure:

The max load that the steering motor would need to overcome is the resulting of the normal force. The maximum of this load will occur when the vehicle is at a static position. The normal force equation [3] is a function of the tractive effort and the rolling resistance. The tractive effort of the vehicle is:

$$F_(max) = \frac{u * W * (l_1 - (f_r * h))}{L - uh} \tag{3.1}$$

where u is the adhesion coefficient, W is the vehicle weight, l1 is the front track distance, fr is the rolling resistance coefficient, h is the height of the cg of the vehicle, and L is the track length. The aerodynamic resistance of the vehicle is:

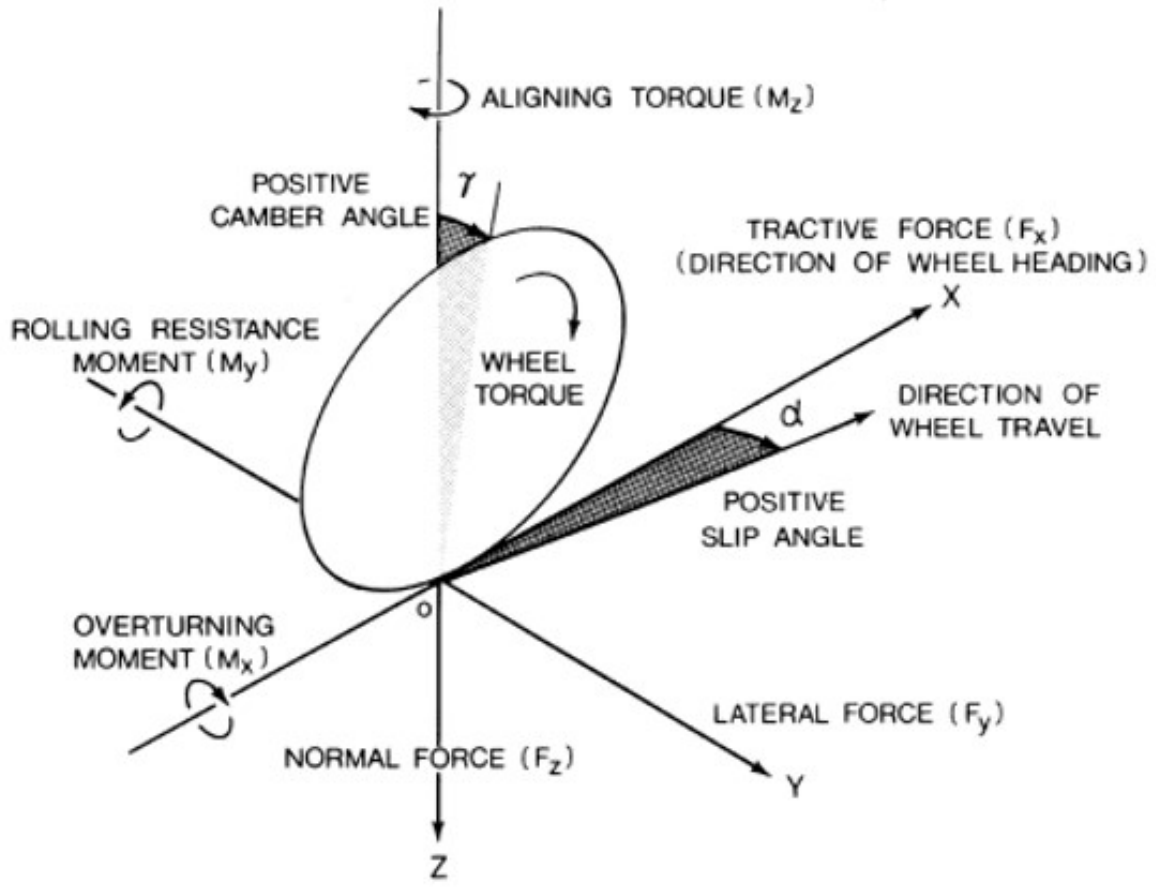


Figure 3.1: Tire Forces

$$R_a = \frac{1}{2}\rho C_d A_f v^2 \tag{3.2}$$

where Cd is the drag coefficient, Af is the frontal area, v is the vehicle speed and p is the air density. These equations were obtained to calculate the rolling resistance of the vehicle.

$$R_r = F_(max) - R_a - \frac{aW}{g} \tag{3.3}$$

where a is the vehicle acceleration and g is the gravitational acceleration. The vehicle's normal force acting on the front axle is:

$$W_f = \frac{l_2 W}{L} - (\frac{h}{L} * (F_(max) - R_r)) \tag{3.4}$$

where l2 is the rear track distance. As we substitute in the above equation with the vehicle's parameters, we obtained the value of the normal force. To obtain the total torque generated from each wheel's normal force due to the distance generated by the castor angle (dc) with an added safety factor (n) of 1.75 is:

$$T = W_f * d_c * n \tag{3.5}$$

The MATLAB code and results of these equations are presented in the appendix.

3.1.3 Ultrasonic Sensors

There are many types of sensors used in our vehicle. To fulfill the requirements set intially by Leoni's engineers, we added three ultrasonic sensors that is used with a proper range to detect workers and/or obstacles surrounding our vehicle.

3.1.4 Absolute Encoder

Figure 3.2: EMS22A30 Absolute Encoder

To control the robot steering and robot velocity, encoders are used to provide the required feedback signals. In steering system, position control is required since it is responsible for measuring the actual steering angle. While velocity control is required to control in transporting bins between stations.

Because of the steering system high sensitivity, any small error may drift the robot out of its desired path. Consequently, the used encoder, EMS22A30, in this subsystem has a high number of pulses per revolution of 1024 pulse per revolution (PPR), which results an error of 0.35 degree. Furthermore, the output steering angular speed is too low as it only about 3.88 rpm after a 20.6 gear reduction ratio, so the encoder will generate feedback signal with low frequency and the used microcontroller will be able to handle it.
The following equations demonstrate the analysis:

$$f = \frac{rpm}{60} * PPR \tag{3.6}$$

where f is the output signal frequency, rpm is the motor speed. By substituting with the datasheet values in this equation, we get:

$$f_(steering) = \frac{3.88}{60} * 1024 = 66Hz$$

for the steering system. Furthermore, the error would equal

$$\theta_(error) = \frac{360}{PPR} = 0.35^o \tag{3.7}$$

3.1.5 Incremental Encoder

Figure 3.3: Incremental Encoder

On the other hand, encoders used in powertrain system can generate only 100 PPR. That is because there is no gear reduction between the wheel shaft and the encoder shaft. As a result, wheels rpm may reach 70 rpm, which may overload the microcontroller in case of high-unneeded PPR. The expected error in the measured position is 6.3 mm, which is negligible.
This can be seen by substituting in the two previous equations.

$$f_(powertrain) = \frac{72}{60} * 100 = 120Hz$$

$$\theta_(error) = \frac{360}{100} = 3.6^o or 0.063rad$$

$$ExpectedError = (\frac{0.2}{2}) * 0.063 = 6.3mm$$

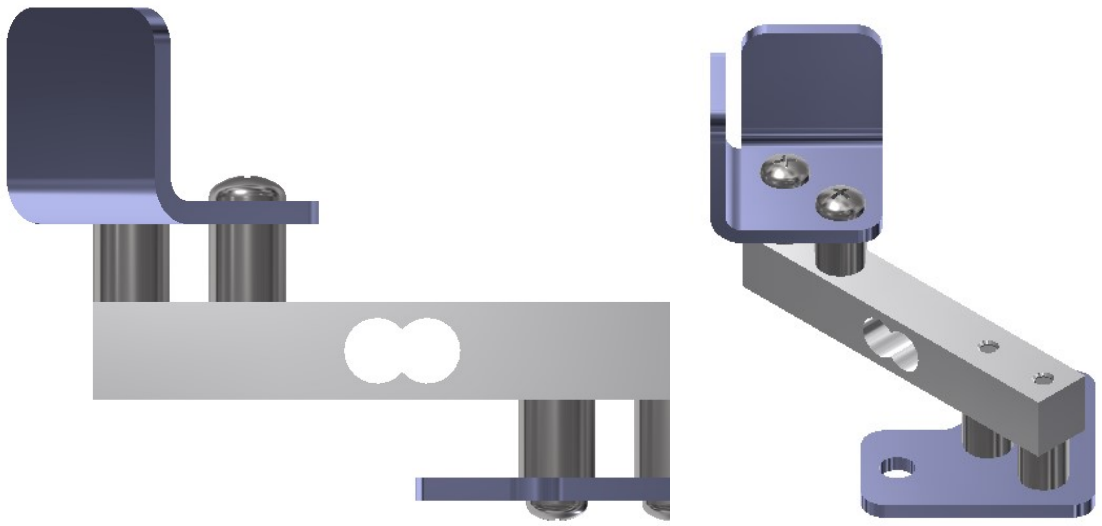
To summarize, here is a comparison with the computed results.

Encoders	PowerTrain	Steering
PPR	100 Pulse	1024 Pulse
Max. o/p freq	120 Hz	66 Hz
Error	0.0063 m	0.35 deg

Table 3.1: Vehicle Encoders Results

3.1.6 Load Cells

Since the carried load may vary and will not be equal to one exact value, tuning velocity control cannot depend on one single set of PID parameters. Changing the weight carried on the robot affects the system dynamics. Therefore, load cells are used to measure the actual carried load in each round, and based on it; a suitable set of PID parameters will be assigned to the microcontroller.



(a) Side View (b) 3D View

Figure 3.4: Load Cell

Four bar load cells are used to measure the weight, one at each corner.

3.1.7 Line Follower

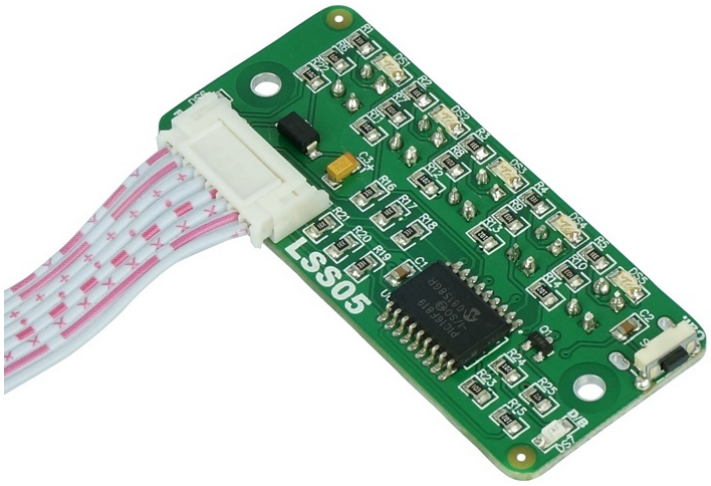


Figure 3.5: LSS05 Line Follower Sensor

As shown in figure 3.5 this line sensor is typically used in embedded systems and robots in line following tasks. It consists of 5 IR transmitter and IR receiver pairs. LSS05 can be used for either dark or bright line following. Any color with distinction or brightness difference is suitable for LSS05. Due to the width of the vehicle, we needed two of these modules for the thick, dark line that we will use on the factory ground. This module will be suitable for the Raspberry Pi 3 that we will use.

3.1.8 Battery

Last year's choice of batteries was based on a decision matrix that tended to be Lead Acid eventually. However, one of our main targets is reducing the weight of the vehicles. According to the battery calculations that we conducted (See 3.2.2), if we choose between Li-Ion and Lead Acid batteries, this would be the comparison results:

	Lead Acid	Li-ion
Nominal Volt	12 V	12 V
Pack Capacity	12 Ah	10 Ah
Packs needed	5	6
Weight	18.5 kg	2.8 kg

Table 3.2: Batteries Choice Comparison

Taking weight into consideration as discussed before, We chose the Li-ion packs. An international model number has been selected for this battery; that is OSN-Li1814-1440.

Figure 3.6: 14.1V 40Ah Samsung battery pack

width	80 cm
length	80 cm
Height	65 cm
Material	Aluminum
Weight	5.9 kg

However, as the funding plan was not executed, we decided to use some of our own batteries to perform the testing and validation phases of or project as shown in figure 3.7.

Figure 3.7: Used batteries in Our Vehicle

As we have set the battery type according to our targets from the beginning to operate in an indoor factory floor, we chose the motors that drive the wheels. These two motors shall be driven by a DC motor controller. We also chose to have the sensors for our object detection. All of these components are connected to the microcontroller. The summary for our selection is shown in the following table.

Component	Amount	Current
DC Motors	3	4
RasberryPi	1	2
TivaC	1	1
Arduino Mega	1	0.6
Line Follower	4	0.03
Ultrasonics	6	0.012
Encoders	2	0.04
Additional Loads	-	2.5
Total	-	18.372

Table 3.3: Electric Components Current Rations

As the components are gathered and summarized. We will calculate the battery capacity in Ampere-hour, but first to show the equation in terms of the battery's life factors:

$$BatteryCapacity = \frac{K_a * K_t * K_cf * K_c * TotalCurrents * Operatingtime}{K_dod} \tag{3.8}$$

where Ka is the aging factor, Kt is the temperature correction factor, Kcf is the correction factor, Kc is the capacity rating factor, K(dod) is the depth of discharge. However, the datasheet of the battery was not clear about some of these factors. We researched on common and similar batteries with referral numbers, but some factors are assumed within reason and a safety margin. By substituting in the above equation, assuming the vehicle will operate for an hour:

$$BatteryCapacity = \frac{1.25 * 0.8 * 1.017 * 1.5 * 18.372A * 1hr}{0.5} = 56Ahr$$

Finally, calculating the rated capacity in watt-hour equation,

$$BatteryCapacity = BatteryCapacity(Ah) * NominalVoltage \tag{3.9}$$

would be:

$$BatteryCapacity = 56 * 14.1 = 789.6Wh$$

3.1.9 Motor Drivers

A motor driver takes a low-current control signal and then turn it into a higher-current signal that can drive a motor. We used two different drivers. The first one contains two channels that are used in the rear drive motors (SparkFun Monster Motor Shield) as shown in fig 3.8 to help in distribution of the vehicle speed along our power train motors. The second driver contains one channel that is used with steering motor.

SparkFun Monster Moto Shield:

This is a high power dc motor driver based on VNH2SP30 chip from ST. It is designed for high power dc motor control applications with peak current up to 30A and continuous current of 14A. The module is easy to use with arduino or any other microcontroller. The motor driver also have thermal shutdown capability for protection of overheating, it also have current sensing and overvoltage protection.

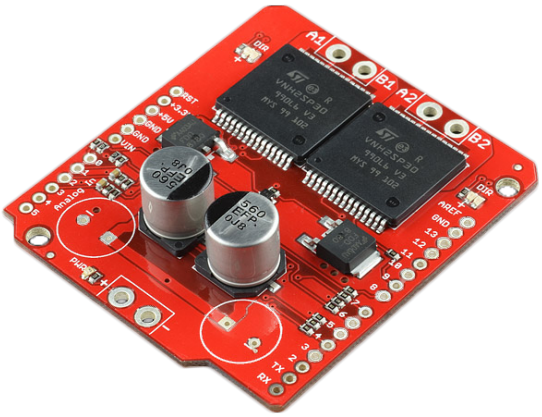


Figure 3.8: dual channel Monster motor Driver.

DC Motor Driver VNH2SP30:
It is very similar to sparkfun Monester moto driver, the only difference is that this module drive only one motor at a time (single channel) as shown in figure 3.9

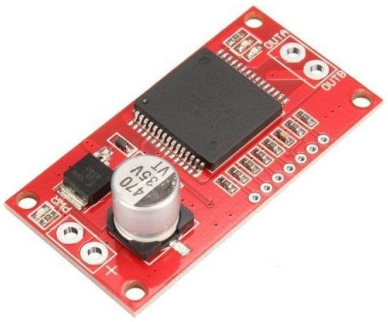


Figure 3.9: single channel VNH2SP30 motor Driver.

3.2 Wiring

3.2.1 Wiring Choice

The wires chosen in our project was either the regular jumper wires that transport a 3.3V or 5V signals with a max current of 20mA or a 1.5 mm copper wire that can withstand up to 9A as the copper wire multiplying factor is 6.

$$CurrentRating = multiplying factor * wire radius Current Rating = 6 * 1.5 = 9A$$

The usage of the jumper wires is for signaling and the 1.5mm wire is for power lines that include the 12V lines in the vehicle

3.2.2 Pluggable Terminals

For the ease of connecting and disconnecting the PCBs and drivers in the testing (or maintenance) phase from the vehicle without having any complications in the wiring, we decided to make all the PCBs and drivers with pluggable terminals shown in figure 3.10 pluggable. They can withstand high voltages and ensure tightening on any wire using the M3 screws inside them.

Figure 3.10: Pluggable Terminals.

3.2.3 Wiring Connectors

Connecting any wire to an electrical receptacle varies according to the socket type. For this reason, we used four different types of connectors. The types and their usages are shown in the following figure 3.11 :

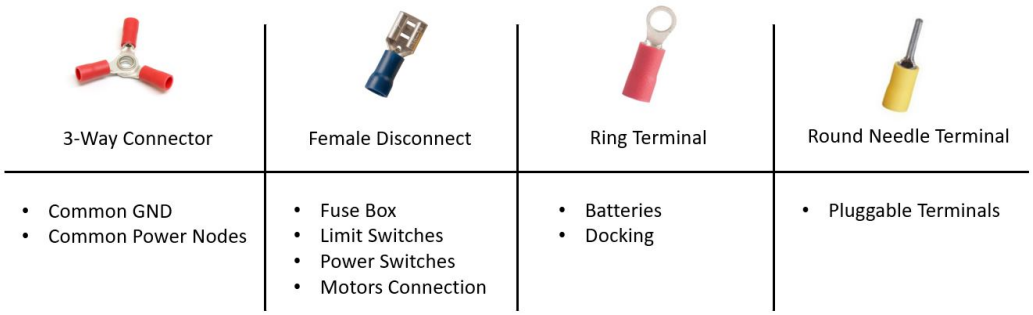


Figure 3.11: Types and Usages of Wiring Connectors.

3.2.4 Wiring Isolation

To ensure avoiding any type of an electrical short-circuit problem, we used 6mm and 8mm heat shrinks for every wire ending. Concerning wiring paths, we used a 15mm-wide tight braided expandable sleeving wire sheath.

3.3 Safety

3.3.1 Kill Switch

The first consideration in our vehicle is an accessible kill switch . It is placed on the upper right corner of the vehicle as shown in figure 3.12a.

(a) Kill Switch Position (b) Kill Switch Assembly

Figure 3.12: Load Cell

3.3.2 Fuse Box

As we have actuators with peak current of 14 A, we added a fuse box as shown in figure 3.13 with 10A fuses for the actuators, and one 5A fuse for the ECU.

Figure 3.13: fusebox.

3.3.3 Safety Signs

As any source of electricity is isolated in the vehicle, an important aspect that is vital to consider is the docking station that transmit AC voltage to the vehicle. This part has two clear text sign messages as shown in figure 3.14 that alert the workers to avoid the danger of getting electrocuted.

Figure 3.14: Fixed Docking Station Danger Signs.

3.4 PCBs and Closure

3.4.1 Main ECU PCB

This PCB servers initially as a shield for the two Cortex-based microcontrollers in our system (i.e. the TIvaC and the Raspberry Pi 3). Any main electrical component used in the vehicle other than the motor drivers passes through this PCB for signal conditioning via the pluggable terminals. The PCB contains 7 logic level converter ICs as the TivaC produces a logic of 3.3V and most sensors provide and require a 5V signal communication. A small fan is operated when the PCB receives an input power that cools the two heat sinks in the raspberry pi above the CPU and the GPU.

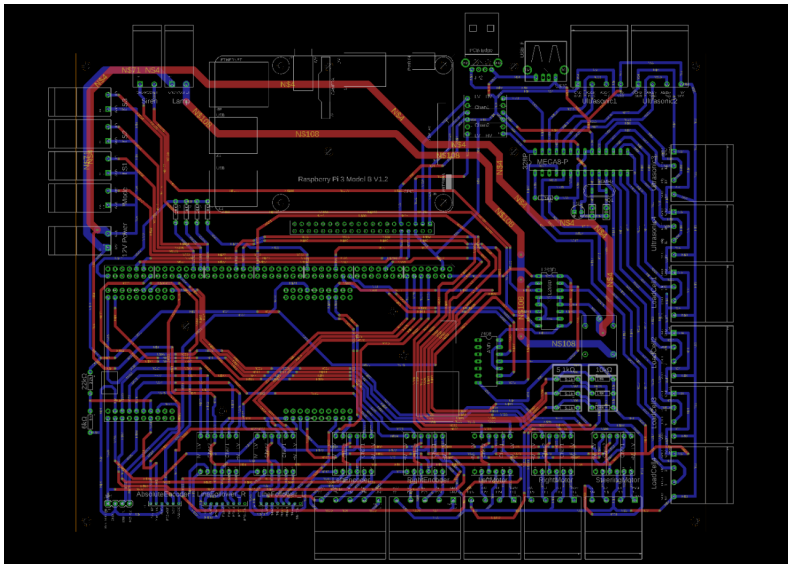


Figure 3.15: Rasberry pi and Tiva C - main PCB layout.

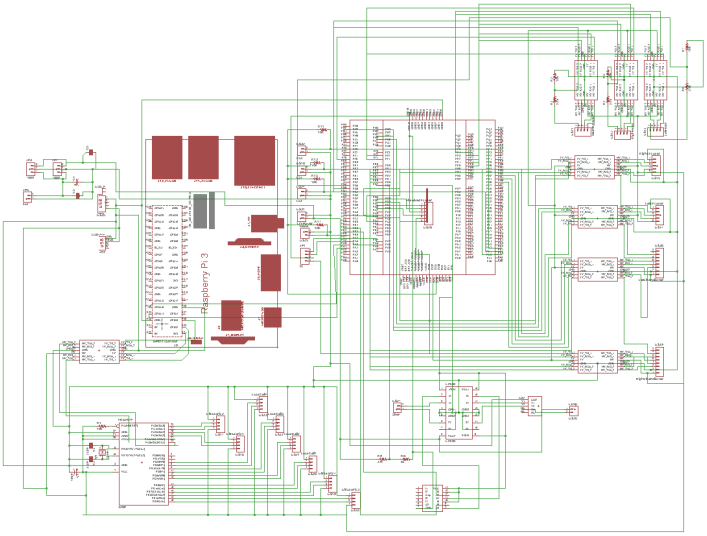


Figure 3.16: main pcb schematic.

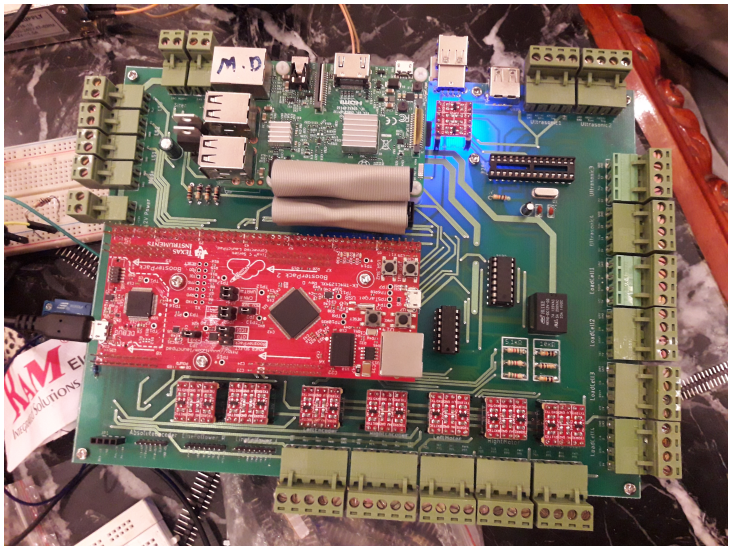


Figure 3.17: main pcb after fabrication

3.4.2 Motor Drivers PCB

This PCB acts as a shield for the two motor drivers ICs. It receives the signals from the ECU PCB to activate the drivers in the direction that is decided by the microcontrollers. A small fan is also placed for this PCB to cool the drivers to attempt to lower the temperature while in operation.

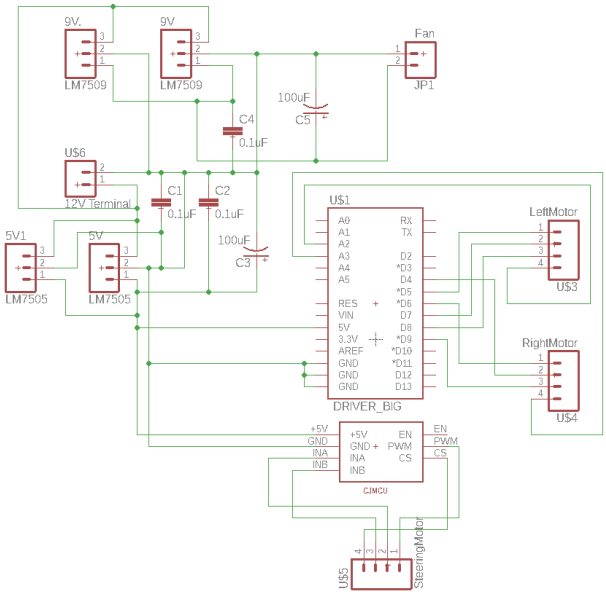


Figure 3.18: Motor driver Pcb schematic

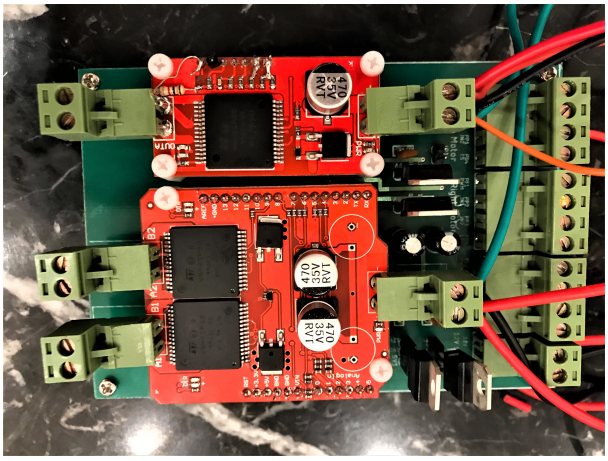


Figure 3.19: Motor driver Pcb after fabrication

3.4.3 DC-to-DC Converter PCB

This PCB has only two DC-to-DC Converters to convert the input 12V to 5V. The first one is used to supply for the Raspberry Pi 3 only and the second is used to supply for the TivaC, the Atmega chip and the rest of the sensors attached to the main ECU PCB terminals.

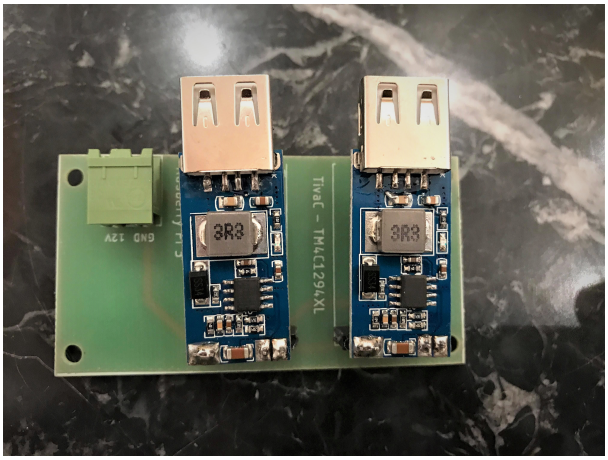


Figure 3.20: DC-Dc PCB after fabrication

Chapter 4

Control Aspect Domain And Software

4.1 Introduction

To achieve the robot target function properly, the robot should be smart enough to sense the required internal robot parameters as long as other external environment-based parameters, and to perform different signal conditioning algorithms to use this sensory information in decision-making. Consequently, the control algorithms are implemented mainly on two different levels, control and decision-making level, and low-level signal conditioning.

4.2 Control System Duties

The overall target function of the robot is to collect and carry certain packages from previously set locations, and to transfer these packages to the dispatch area inside the factory plant. As a result, the robot path should cover all possible loading points in addition to the docking station inside the dispatch area. These loading points are the outputs of certain production lines that require a trolley to transfer their output products. The proposed robot path with loading locations is shown in figure 4.1.

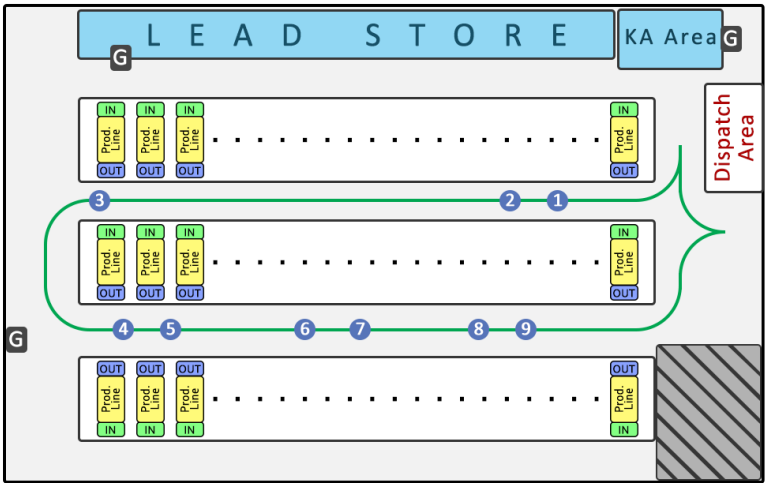


Figure 4.1: Proposed Robot Path with loading location

To maintain this path properly, the robot should follow the line constructed on the ground with fixed relatively safe speed, and it should know its location on the map so it can stop at loading locations and return to the dispatch area and docking station.

The robot should stay at the docking station while it is idle. At certain times during the work shift, the robot will start its cycle by moving toward the first loading location. At each loading point, the robot will wait for two minutes so the operator can load the products on it. After that, it will move to the next station until reaching the last one. Then, it will move back to the dispatch area to unload the products and to charge its battery again at the docking station.
To handle different loading situations, the robot is equipped with four load cells (only one is attached on the robot prototype) to determine actual loading capacity, so in case of overload situation, the robot will move directly to the dispatch area without stopping at next loading locations as there is no available capacity to receive additional loads.

As the working environment contains interaction with workers and operators and the pathway is not designed for the robot only, the robot is equipped with eight ultrasonic sensors (only four are attached to the robot prototype), a siren and a rotating warning lamp. The robot should stop immediately if there is any obstacle on its way. The siren and the lamp are used to alert the workers so, for example, the siren will sound when reaching the loading location or dispatch area, or if there is any error. Different sounding patterns are used depending on the situation.

4.3 Software Architecture

ARM ® is one of the most industry's leading supplier of microprocessor technology in the world. Over the years, ARM has developed many processors. Among their offerings is the ARM Cortex family. The ARM Cortex family subdivided into three profiles as mentioned below:

Cortex A Profile: It is based on ARMv7-A architecture, the processors are categorized for the applications requiring the capability to run complex operating systems like Android, Microsoft Windows.
Cortex R Profile: It is based on ARMv7-R architecture, the processors are categorized for real-time applications where high reliability, availability, fault tolerance, and real-time responses are required. These processors typically run a Real-Time Operating system (RTOS), along with appropriate user code.
Cortex M Profile: This profile uses ARMv6-M architecture design for its lower end processors. The processors are optimized for cost- and power-sensitive applications like smart metering, automotive and industrial control systems, and human interface devices. The profile consists of Cortex M0 and Cortex-M0 plus for ultra-low power and Cortex-M3 and Cortex-M4 for digital signal processing applications [7].

Based on the previous comparison, we used two different developing boards based on different arm processors. The first one is the raspberry pi 3, which is able to run Unix-based operating system and ROS (robot operating system) [6]s. The second one is TM4C129, and it is used for signal processing.

The following figure 4.2 illustrates the architecture of used microcontrollers and developing boards and their corresponding duties.

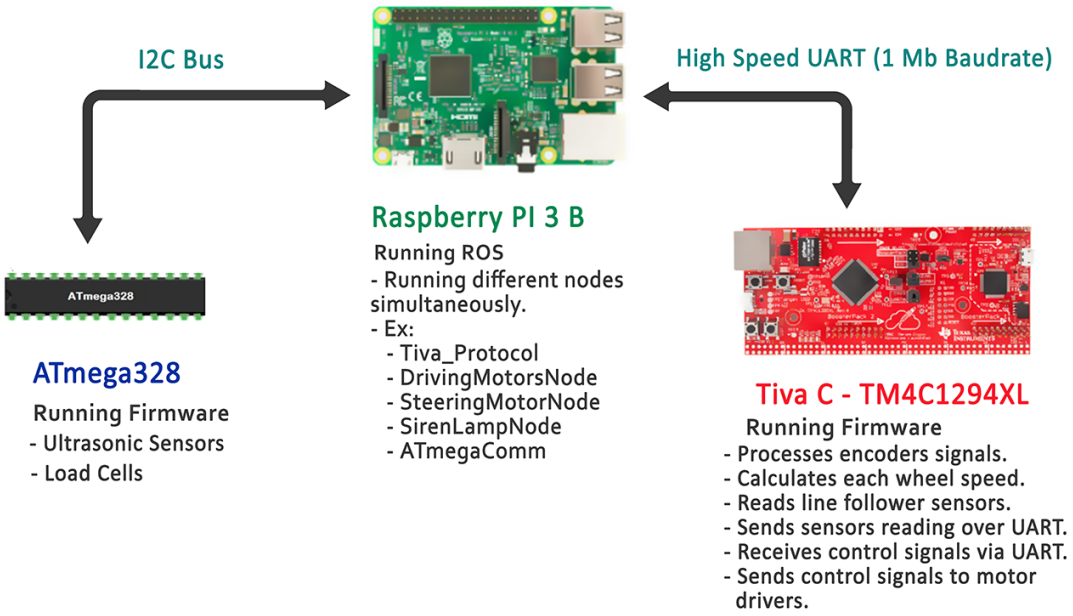


Figure 4.2: The Architecture Of Used Microcontrollers and Developing Boards

4.3.1 Control and decision-making level

One of the most trending operating system for robots is ROS (robot operating system). It is an open-source, meta-operating system for robots. It provides various services from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. ROS currently only runs on Unix-based platforms, that is why we used raspberry pi board for control and decision-making level as it is able to run different Unix-based operating systems. The used platform in our robot is Ubiquity Robotics Raspberry Pi image. It is based on Ubuntu 16.04 and it is pre-installed with ROS. More information about this image is available on the following webpage [5]

The raspberry pi 3 B board specifications can be summarized as mentioned below:

- SoC: Broadcom BCM2837
- CPU: 4 ARM Cortex A53 1.2GHz
- GPU: Broadcom VideoCore IV
- RAM: 1GB LPDDR2 (900 MHz)
- Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless

- Bluetooth: Bluetooth 4.1 Classic Bluetooth Low Energy
- Storage: microSD
- GPIO: 40-pin header, populated
- Ports: HDMI 3.5mm analogue audio video jack 4 x USB 2.0, Ethernet, Camera Serial Interface (CSI) Display Serial Interface (DSI)

We developed different nodes and created the required messages formats to make these nodes communicate easily without coding and decoding processes. ROS makes it easy to add new features and to make further development as the user just need to write his/her new nodes independently. Note that we are using line follower modules to guide the robot. However, ROS can work with Lidar and cameras to navigate fully autonomously. As a result, adding these sensors and programming their corresponding nodes properly can take our robot to the next level and make it able to navigate in any indoor environment safely.
Especially that it is equipped with different sensors that measure its internal parameters such as battery volt, motors current, speed, travelled distance and actual load capacity. Adding cameras to the robot was in our plan, but due to time constraints, only guiding the robot through line follower modules was developed as shown in figure 4.3 .

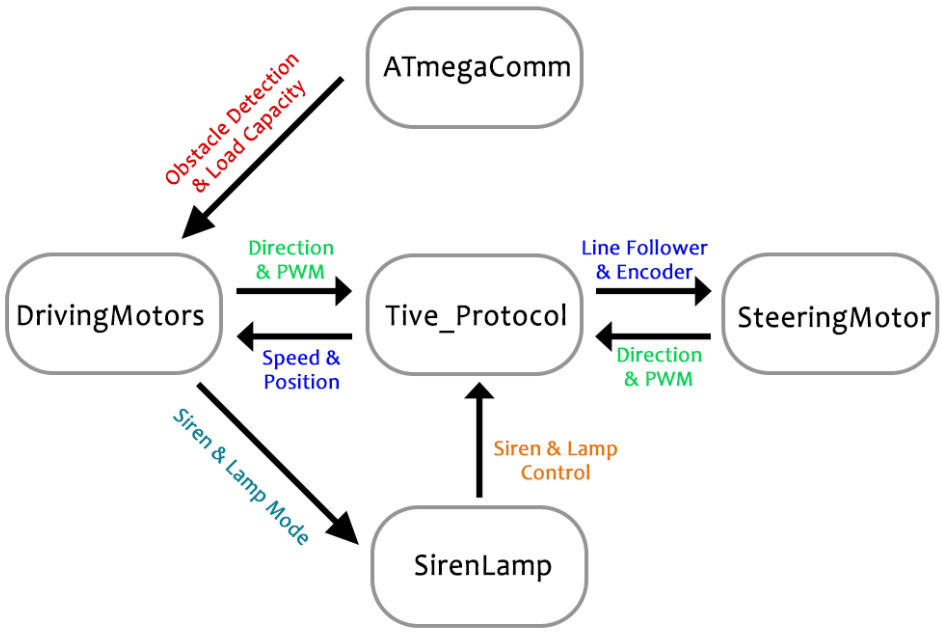


Figure 4.3: Communication Between Different Nodes

All nodes codes and messages are available in our google drive [10].

Type	Peripheral	Function
Sensor (Inputs)	Right and left wheel incremental encoders	Measuring robot speed and position
	Steering system absolute encoder	Measuring steering angle
	Line follower sensor modules	Detecting the line and the error
	Load cells	Determining the actual capacity
	Ultrasonic Sensors	Obstacle Detection
Actuators (Outputs)	Rear driving motors drivers	Controlling driving motor
	Steering motor driver	Controlling steering motor
	siren	Operating the siren
	Rotating warning lamp	Operating the lamp

4.3.2 Low-level signal conditioning

To follow the constructed line on the ground, maintaining a fixed speed, knowing the robot position, turning the steering system, obstacle detection, and load-capacity measurement, the peripheral devices used are listed in the table above.

As there are too many peripherals, adequate microcontroller with relatively large number of ports is required to handle these input/output signals. Based on the priority of each peripheral and the required processing power, they are distributed over two microcontrollers.

TM4C129 Board

Texas instruments has introduced a very powerful developing board with many different important modules and communication interfaces. The main board specifications [8] are:

☐ Processor: ARM® Cortex™-M4F Processor Core, up to 120 MHz.

☐ On-chip Memory: 1 MB Flash; 256 KB SRAM; 6KB EEPROM.

☐ Communication Interfaces: 8 UARTs, 10 I2Cs, 4 Quad SPI, 2 CAN.

☐ System Integration:

- Eight 32-bit general-purpose timers.

☐ - Motion Control: Advanced timers with eight PWM outputs.

☐ Analog: 24 Channels of 2x 12-bit ADC up to two MSPS.

This board is used as the main microcontroller to handle most critical peripherals that require either high frequency communication or complex signal conditioning processes. The following table summarizes all peripherals connected to the board and their corresponding pins in addition to their function.

port	pin	Board Module	Device	Function
A	0	UART0 (Rx)	PC	On-board diagnostics (OBD)
A	1	UART0 (Tx)	PC	On-board diagnostics (OBD)
A	4	UART3 (Rx)	Raspberry	Communication with raspberry pi
A	5	UART3 (Tx)	Raspberry	Communication with raspberry pi
A	6	UART2 (Rx)	Bluetooth	Communication with BT module
A	7	UART2 (Tx)	Bluetooth	Communication with BT module
B	0	GPIO	Mode Switch	Choosing between manual & automatic modes
B	2	GPIO	Lamp	Lamp Control
B	3	GPIO	Siren	Siren Control
C	4	Interrupt	Inc. Encoder 1	Counting channel Z pulses
C	5	Interrupt	Inc. Encoder 1	Counting channel A pulses
C	6	GPIO	Inc. Encoder 1	Counting channel B pulses
D	0	SPI2 - MISO (Rx)	Abs. Encoder	Master input slave output
D	1	SPI2 - MOSI (Tx)	Abs. Encoder	Master output slave input
D	2	SPI2 - SS	Abs. Encoder	Select Slave
D	3	SPI2 - CLK	Abs. Encoder	SPI2 Clock
D	4	ADC 7	MD1 Current sensors	Reading motor driver 1 current
D	5	ADC 6	MD2 Current sensors	Reading motor driver 2 current
D	6	ADC 5	MD3 Current sensors	Reading motor driver 3 current
D	7	ADC 4	Battery Volt	Reading battery voltage
E	0	GPIO	Line Follower 1 - P1	First pair reading
E	1	GPIO	Line Follower 1 - P2	Second pair reading
E	2	GPIO	Line Follower 1 - P3	Third pair reading

Port	Pin	Board Module	Device	Function
E	2	GPIO	Line Follower 1 - P3	Third pair reading
E	3	GPIO	Line Follower 1 - P4	Fourth pair reading
E	4	GPIO	Line Follower 1 - P5	Fifth pair reading
E	5	GPIO	Line Follower 1 - Cal.	Calibration Signal
F	0	PWM01	MD1 - Enable	Motor Driver 1 PWM Input
F	1	PWM02	MD2 - Enable	Motor Driver 2 PWM Input
F	2	PWM03	MD3 - Enable	Motor Driver 3 PWM Input
H	0	Interrupt	Inc. Encoder 2	Counting channel Z pulses
H	1	Interrupt	Inc. Encoder 2	Counting channel A pulses
H	2	GPIO	Inc. Encoder 2	Counting channel B pulses
K	0	GPIO	LS1	Steering Limit Switch 1
K	1	GPIO	LS2	Steering Limit Switch 2
K	2	GPIO	Back Limit Switch	Back Limit Switch
N	0	GPIO	Line Follower 2 - P1	First pair reading
N	1	GPIO	Line Follower 2 - P2	Second pair reading
N	2	GPIO	Line Follower 2 - P3	Third pair reading
N	3	GPIO	Line Follower 2 - P4	Fourth pair reading
N	4	GPIO	Line Follower 2 - P5	Fifth pair reading
N	5	GPIO	Line Follower 2 - Cal.	Calibration Signal
P	0	GPIO	MD 1 Dir1	Direction Input 1
P	1	GPIO	MD 1 Dir2	Direction Input 2
P	2	GPIO	MD 2 Dir1	Direction Input 1
P	3	GPIO	MD 2 Dir2	Direction Input 2
P	4	GPIO	MD 3 Dir1	Direction Input 1
P	5	GPIO	MD 3 Dir2	Direction Input 2

All codes are available in the google drive [10]. The developed program starts by initializing all ports and modules independently. Then, it iterates in a while loop. Almost all processes are performed using interrupt service routines.

Main function:
Main function starts by initializing all ports and timers, then it iterates in a while loop to poll two flags.

First flag indicates that 100 milliseconds has been elapsed since sending the last message, so the program creates a new message and sends it to the raspberry pi. The second flag indicates receiving a new message from raspberry pi. The following flowchart shows how main function works:

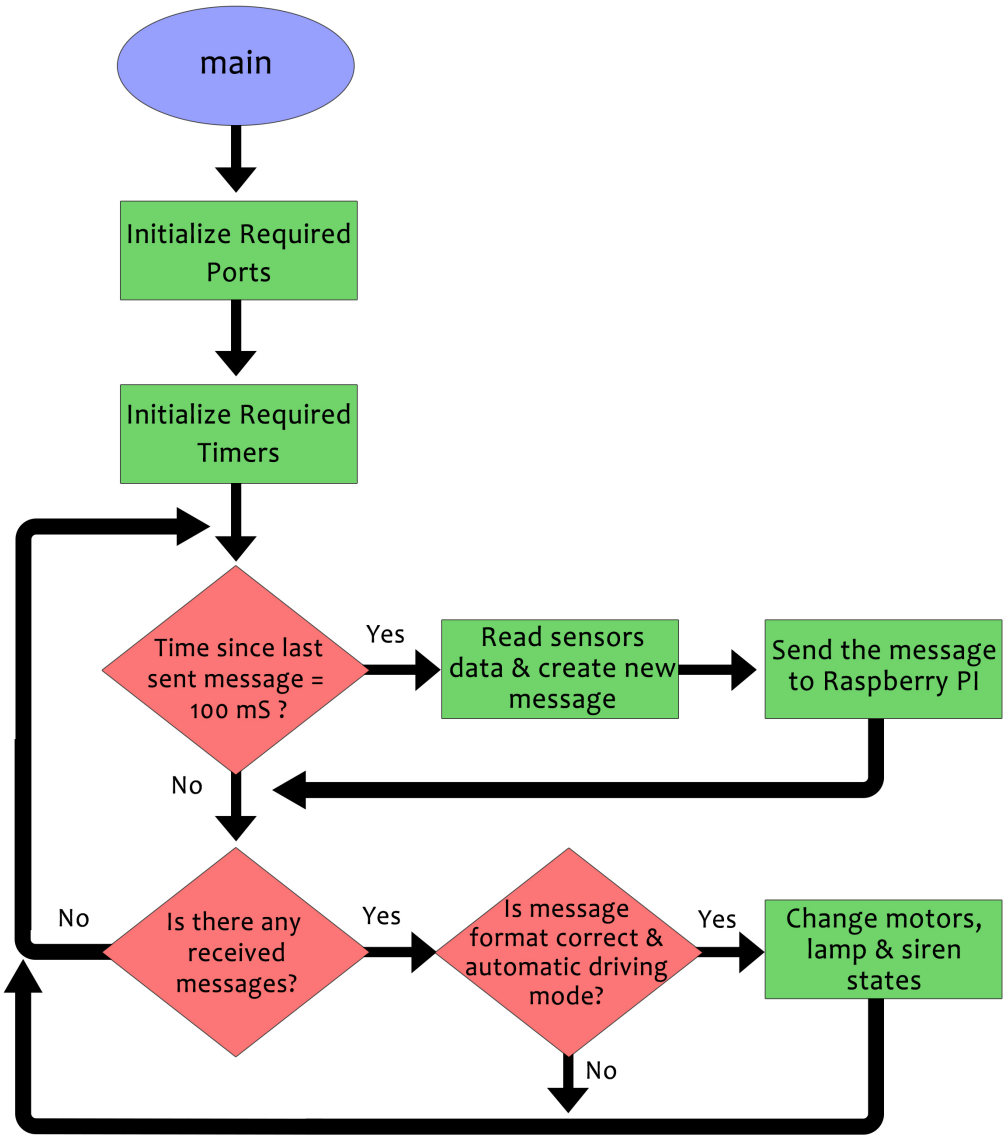


Figure 4.4: Main Function Flowchart

General Purpose Timers (Timer0 & Timer1): Timer0 acts as the main clock for the program. It elapses every 100 microseconds and increments a variable indicating the time in hundreds of microseconds.
Timer1 elapses every 100 milliseconds and its ISR updates robot speed and distance according to encoders' readings.
The following flowcharts show Timer0 & Timer1 ISRs:

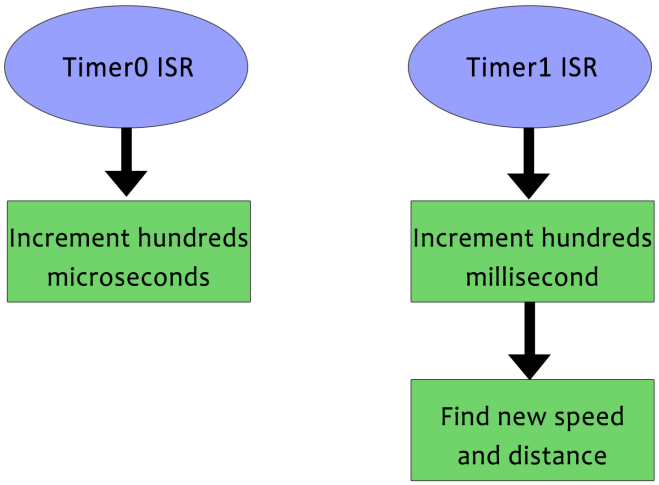


Figure 4.5: Interrupt Timers Flowchart

UART Communication (Port A):
Three different UART modules are programmed to communicate with different peripherals with the maximum possible baud rate for each one of these peripherals.

UART0 is used to interface with PC at a baud rate of 1.5 Mb/s to debug the program and to monitor sensors readings when necessary. Note that in default case, there is no communication on this channel.

UART2 is used to receive Bluetooth module commands sent from a smart phone at a baud rate of 38400 bit/s. Note that the robot will respond to the sent commands only if the mode switch is on manual drive mode. It is mainly used to test robot actuators and to drive it in emergency cases or when the line is not available.

UART3 is the main communication module as it responsible of receiving and sending messages from and to raspberry pi. The developed program sends all necessary sensors readings to raspberry pi after performing the required signal processing. In return, raspberry pi responds with the proper control actions to be made based on the sensors readings. The selected baud rate for this module is 1 Mb/s. Higher baud rates have showed some data losses. Note that this mode will be active only if the driving mode switch is on automatic mode.

The following flowcharts show UART2 & UART3 ISRs:

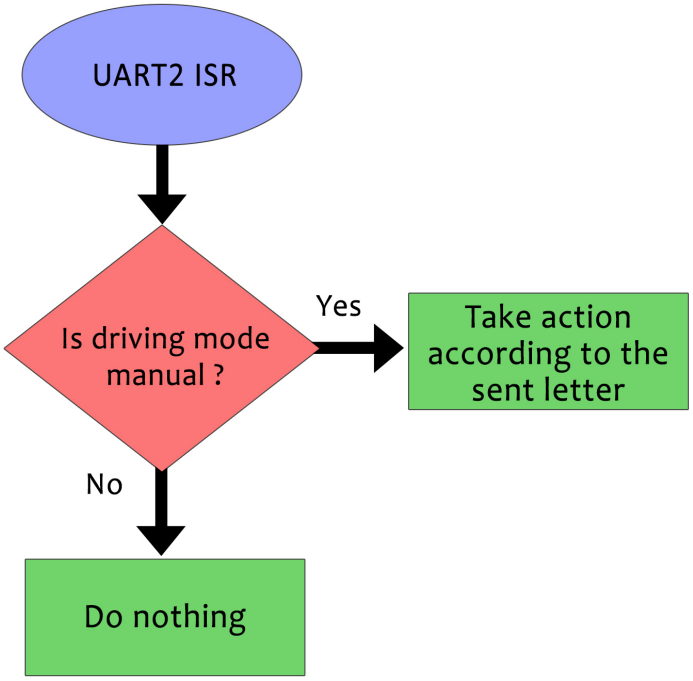


Figure 4.6: UART2 ISR Flowchart

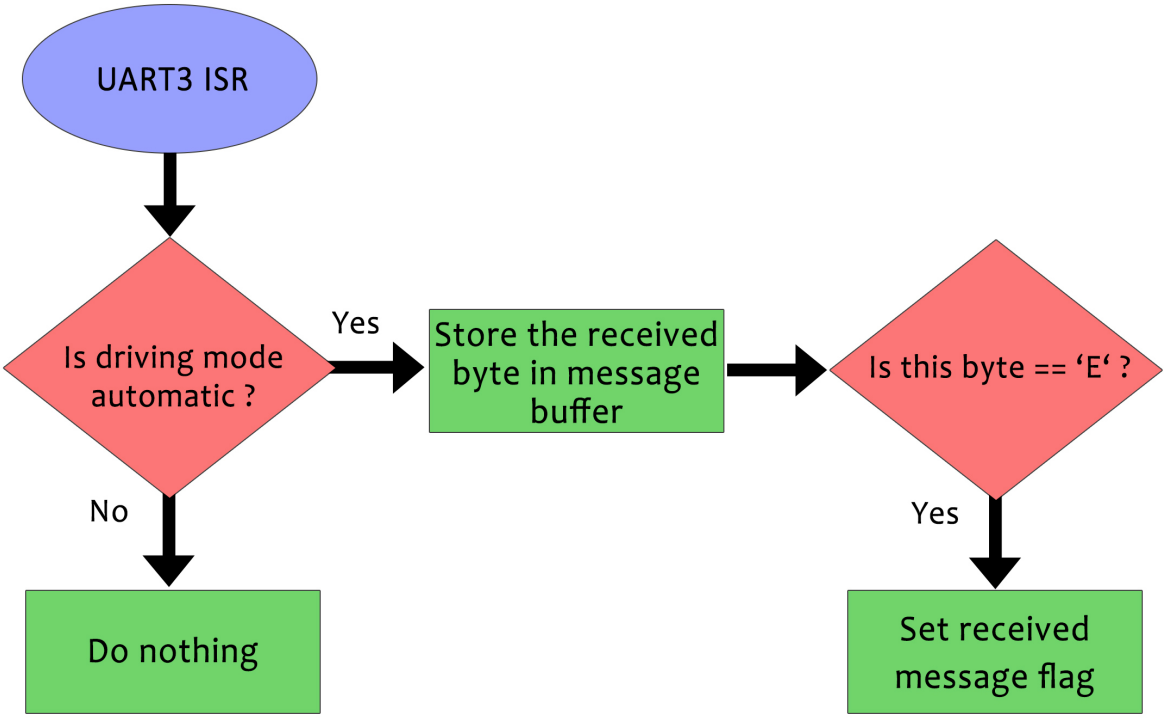


Figure 4.7: UART3 ISR Flowchart

Port B:
Port B is responsible of determining the state of the driving mode as long as setting the state of the siren and the lamp. The state of both of the siren and the lamp are received over UART.

Incremental Encoders interface (Port C and Port H):
An incremental encoder provides a specified amount of pulses in one rotation of the encoder. The used encoders have 3 different channels A, B and Z. A and B channels output are two lines of pulses that are offset in order to determine rotation direction. This phasing between the two signals is called quadrature. An index or 'Z' channel is provided as one pulse per revolution signal for homing and pulse count verification on the A and/or B channels.
the following figure 4.8 shows the incremental encoder principle .

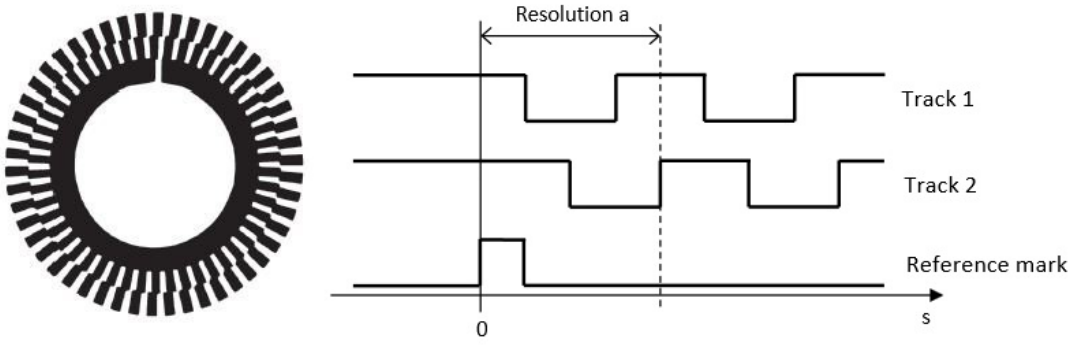


Figure 4.8: Incremental Encoder Principle

In our project, we are using two Autonics E50S encoders. Their main specifications are:

- Resolution: 100 PPR.
- Output phases: A, $\overline{A}$, B, $\overline{B}$, Z, $\overline{Z}$.
- Protection: IP65
- Weight: 275 gram.

Each one of these encoders is connected mechanically to a rear wheel to measure its speed and its traveled distance. We are using only channels A and B and they are connected to ports C and H. These ports are configured to detect the generated pulses by enabling ports interrupt and thus measure each wheel speed and distance.
The following flowcharts show Port C & Port H ISRs:

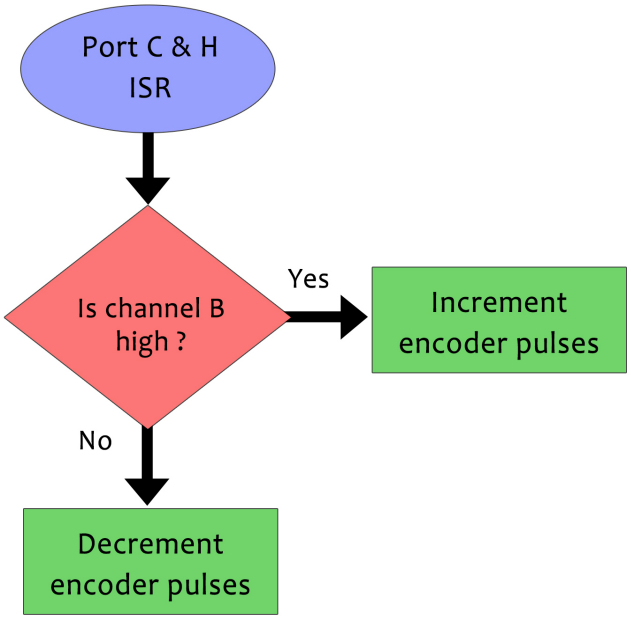


Figure 4.9: Port C & Port H ISRs Flowchart

Port D:
Port D is responsible of reading absolute encoder attached to the steering system as long as reading the three motors current sensors. In addition to measuring the battery voltage. Reading the absolute encoder is done using SPI protocol. The following image 4.10 shows how the sensor transmits its absolute output. SPI2 module is attached to this port so we configured it to read current steering angle.

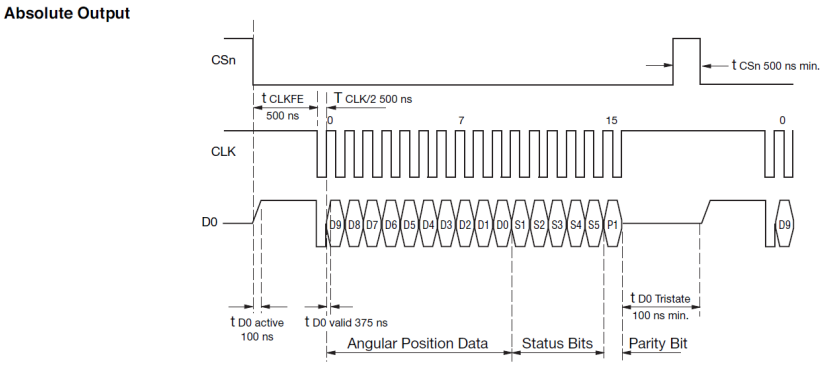


Figure 4.10: Absolute Encoder Output

The used absolute encoder is EMS22A - Non-Contacting Absolute Encoder from Bourns. Its specifications are:

- Resolution: 10 bit (1024 PRP).
- Maximum RPM: 10000 rpm. .
- Accuracy: 0.7 degree.
- Protection: IP 65.

The following image 4.11 shows the disc of 8 bit absolute encoder:

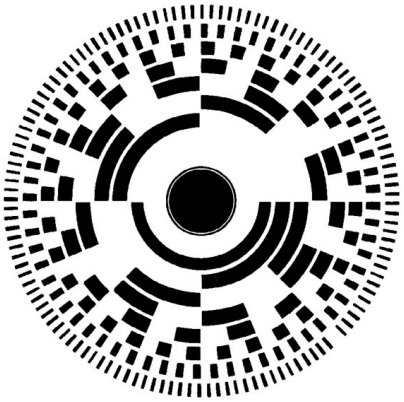


Figure 4.11: The Disc Of 8 Bit Absolute Encoder

Pins 4, 5, 6, and 7 are used as analog pins to read current sensor of each motor and robot battery voltage as an estimation of its state. Despite that the port is configured properly to interface with theses readings, the control algorithm does not use them in decision-making and they are available for future development.

Line follower Modules (Port E and Port N):
The robot is equipped with two line follower modules; each one contains five pairs of IR transmitter and receiver. Port E and Port N are configured as GPIO to read these two modules.

Pulse Width Modulation Generation (Port F):
The robot contains three different DC motors; two are attached to the rear wheels while one is attached to the steering system. These motors should operate at different speeds according to PWM signal generated from the microcontroller. Port F is configured to generate PWM signal with frequency of 15 kHz, which is adequate to the used motor drivers.

Port K:
Port K is configured as GPIO pins to read limit switches signal. All limit switches are pulled up.

Port P: Port P is configured as GPIO and it is responsible of setting the motors directions according to the sent commands.

Atmega328

This microcontroller is the one used in Arduino Uno developing board. Its capabilities are less than TM4C129 but they are sufficient to read load cells and ultrasonic sensors and to calculate the actual load on the robot. It uses I2C protocol to communicate with Raspberry Pi board. The following table shows the connection with load cells and ultrasonic sensors.

Pin	Function
A4(19)	I2C SCL
A5(18)	I2C SDA
A3(17)	Ultrasonic 1 - Trig
A2(16)	Ultrasonic 1 - Echo
A1(15)	Ultrasonic 2 - Trig
A0(14)	Ultrasonic 2 - Echo
13	Ultrasonic 3 - Trig
12	Ultrasonic 3 - Echo
11	Ultrasonic 4 - Trig
10	Ultrasonic 4 - Echo
9	Load Cell 1 - DOUT
8	Load Cell 1 - CLK
7	Load Cell 2 - DOUT
6	Load Cell 2 - CLK
5	Load Cell 3 - DOUT
4	Load Cell 3 - CLK
3	Load Cell 4 - DOUT
2	Load Cell 4 - CLK

4.4 System Design Concept

For a line-following robot to perform correctly without any errors during operation, a set of parameters must be observed and maintained continuously. Such as, the robots position on the line, as our vehicle is rear wheel drive so we will assume a constant velocity for the vehicle and we will apply position control (PID) Based on feedback of line follower sensor that return the position of the vehicle relative to the line. Until now we did not know the track of the factory and the environment that the vehicle will work in , so we worked on virtual environment in v-rep program and also make a virtual model of the vehicle to test our control algorithm in that environment .We not only use v-rep as simulator but also used it as control program with its lua's language .

4.4.1 Motors Control

The actuation of the motor is based on a feedback control that combined both PID control and the sensor readings. This closed loop feedback system allows for accurate and optimum performance of the vehicle in terms of maneuvering.

4.4.2 AGCV Communication Interface

A mobile app interface is to be built in order to allow for the communication between the workers and the vehicle to send or receive the desired Kanban bins.

4.5 Line Follower Control Methodology

All basic control systems rely on feedback.. feedback of the system output, to know what state the system is in the instant and adapt the system by changing its parameters so that the system reaches the desired output.. We call that a SETPOINT.. (S.P) To attain the S.P we change certain elements in the system, these elements (variables) are called Manipulated Variables (M.V)

So on coming to line followers our goal is to keep the bottom the track at all times, ideally the center of the bot on the center of the line at all times.. For the time being let this be our S.P... To achieve that we need to change the way out bot moves, i.e we control the motors. Assuming the bot has a sensor array of 5 sensors (5-channels kits), the algorithm would be as follows:

1. Initialize the set point (SP)
2. Read sensor data (PV)
3. Calculate the error e(t)=SP-PV
4. Calculate the output of the PID controller u(t).
5. Apply controller output to the actuators (steering motor)
6. Go back to step 2

Our aim is to keep the bot on the center of the line , for feedback we have sensors; i.e. we need to keep the center sensor on the center of the line. That's ok, but how can we convert 5 or 2n+1 sensor inputs of a single variable r(t). That's where simple mathematics and number system comes into use.

4.6 Line Follower V-REP simulation

We started our v-rep simulation by building our virtual vehicle using one of the built-in four wheel car in V-rep shown in figure 4.12.

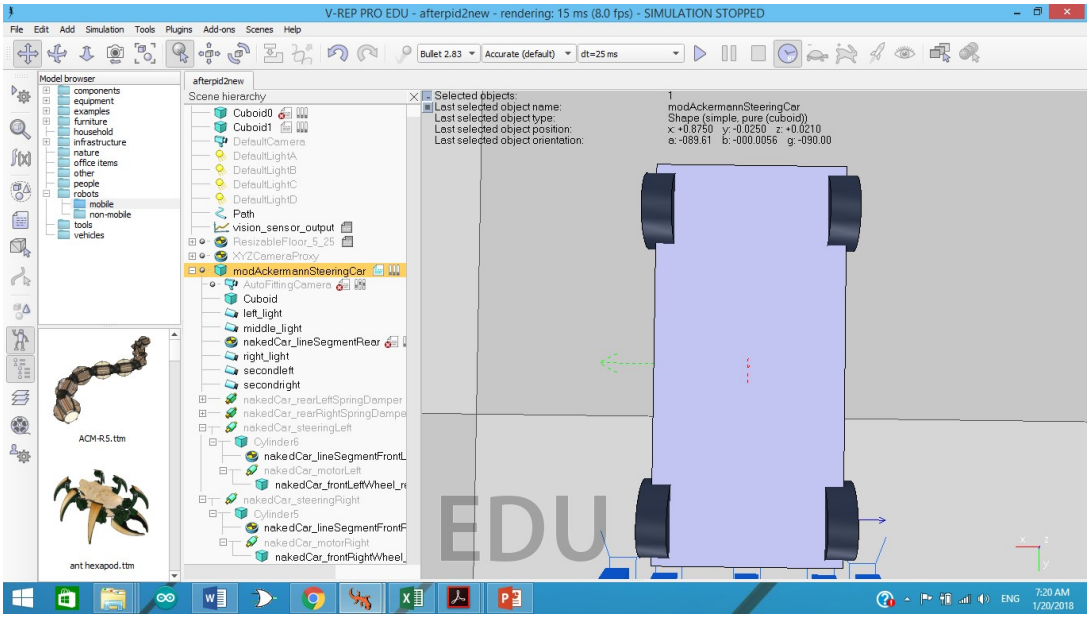


Figure 4.12: Wheel Vehicle Model

We apply Ackermann condition on its front wheel by using Ackermann relations illustrated in steering chapter. After that we started to implement line follower sensor (5-channels) which named as vision sensor in v-rep library as shown in figure 4.13. Then,

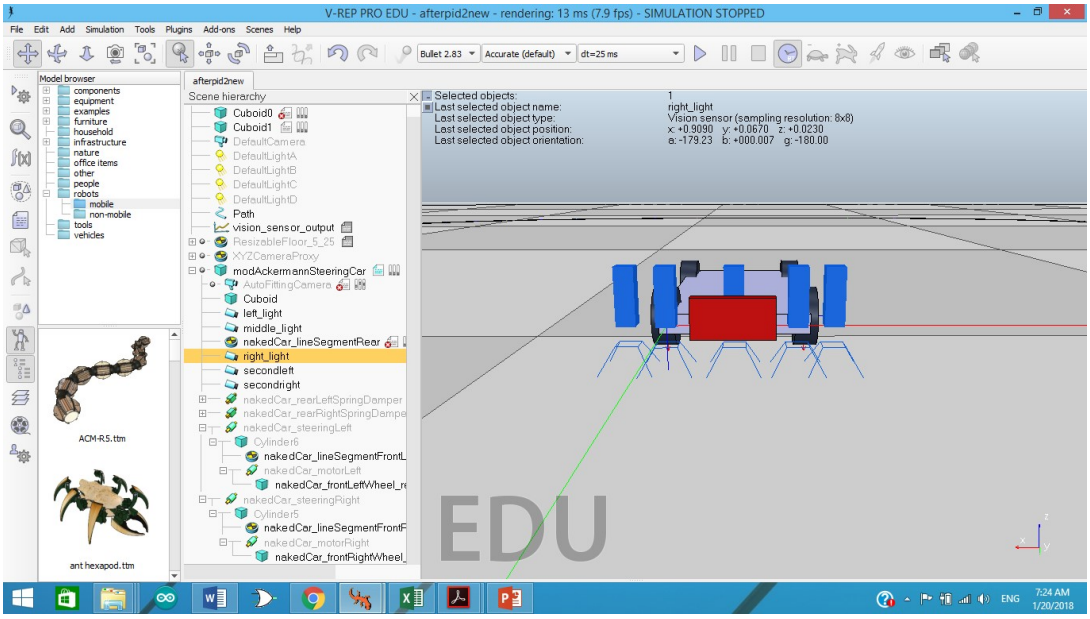


Figure 4.13: Ackermann scene with 5 channel vision sensor

we started to calibrate vision senor and make it in perspective mode from properties as shown in figure 4.14. Now each channel return one if it sees the black line .otherwise, it return zero. We check that by creating a random closed path in the scene colored in

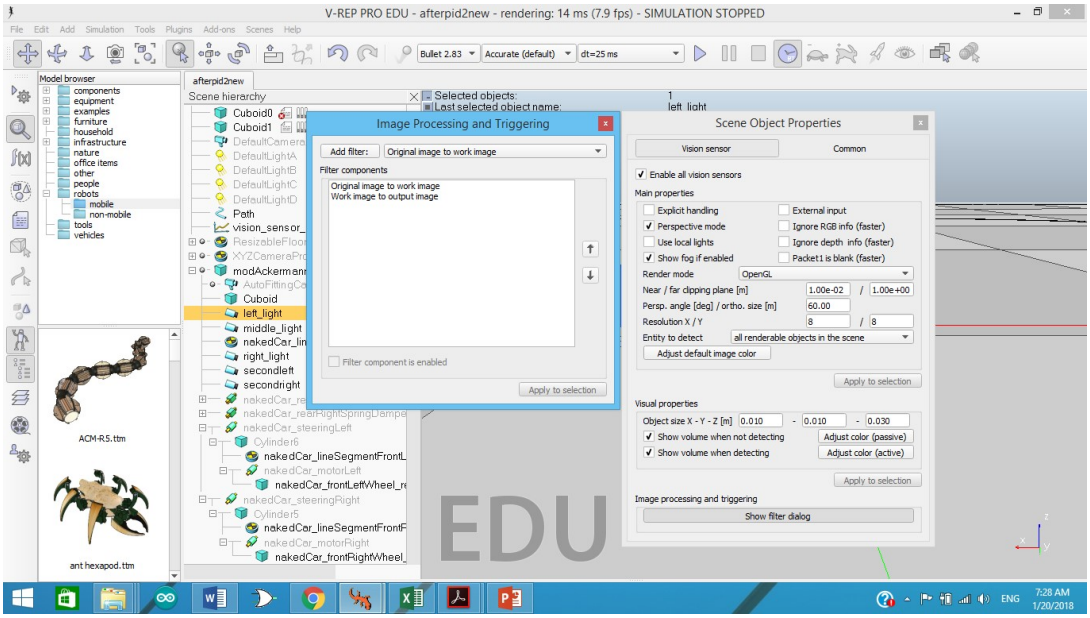


Figure 4.14: Calibration of vision sensor

black with width 2 cm as shown in figure 4.15.

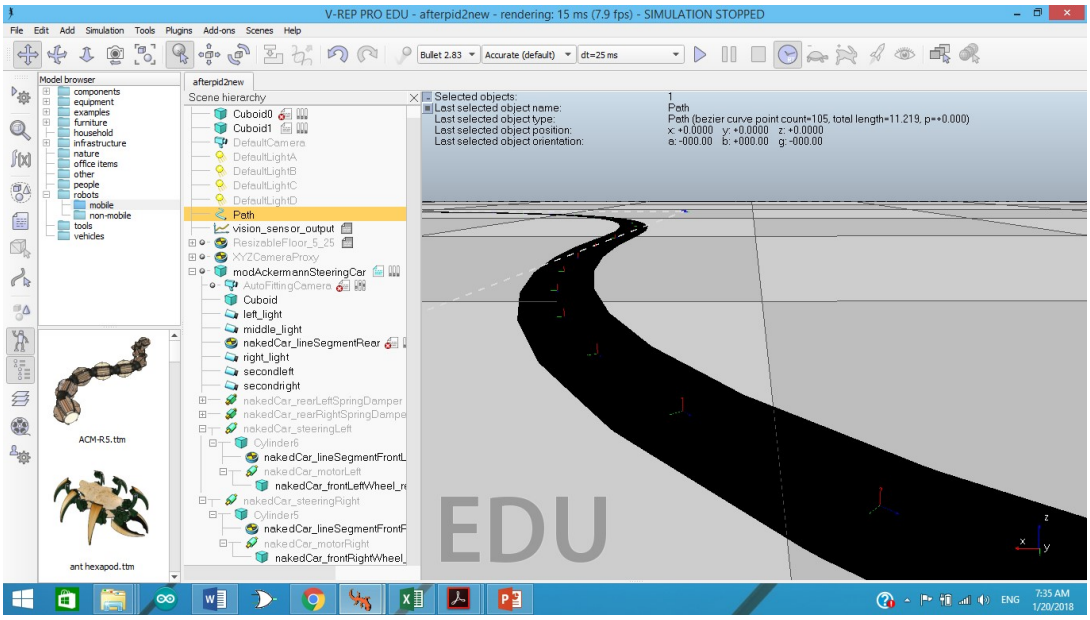


Figure 4.15: Closed black path in v-rep

We also make it visible to vision sensor and test the sensor with in using a graph representing the sensor reading with time shown in figure 4.16.

Then we start to fill the dynamics of the vehicle with reasonable values to test our algorithm, when the design of our vehicle finish, we will replace these values with the original one.Also, we enable the motors of each joint to get the pid parameters of each joint indi-

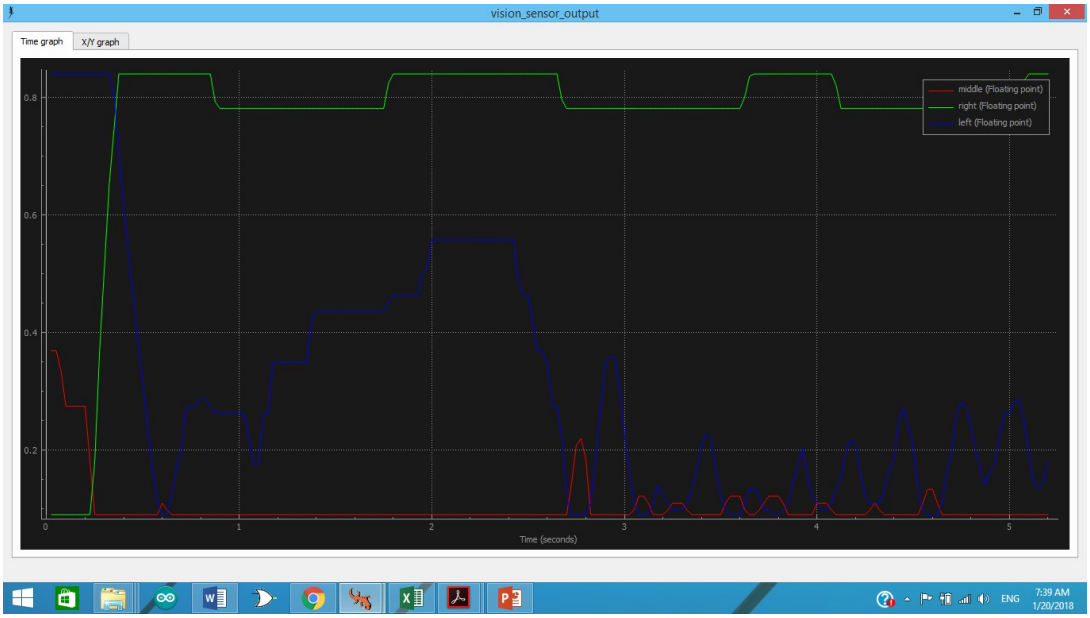


Figure 4.16: Relationship between sensor reading in y-axis and time in x axis

vidually to be ready for taking command as shown in the figures 4.17 and 4.18.

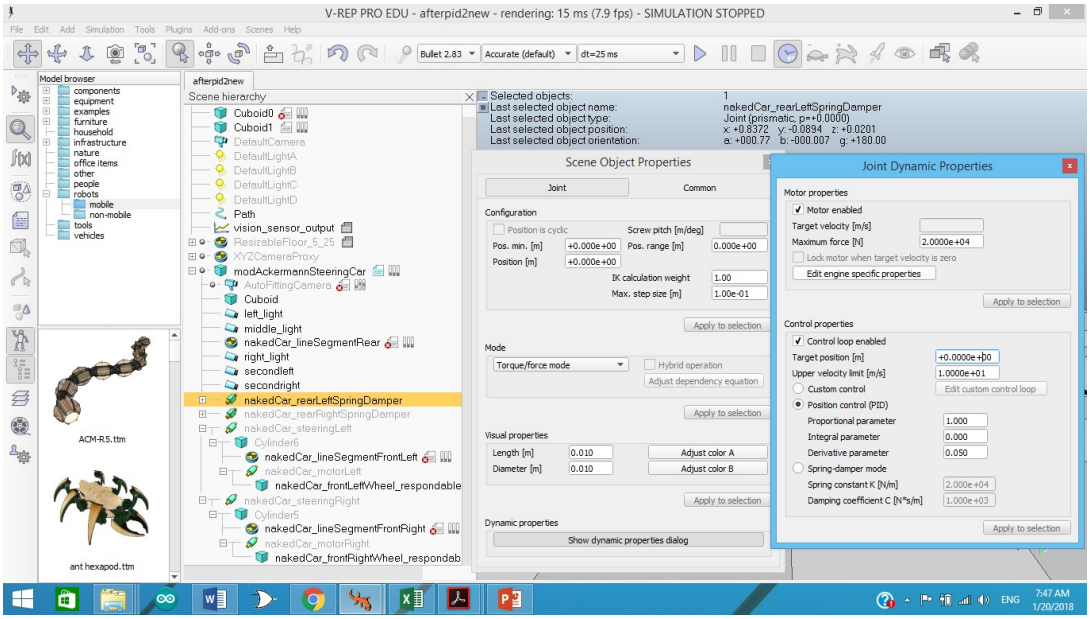


Figure 4.17: Dynamics of rear motors (1)

Concerning the front wheels and front joint dynamics :

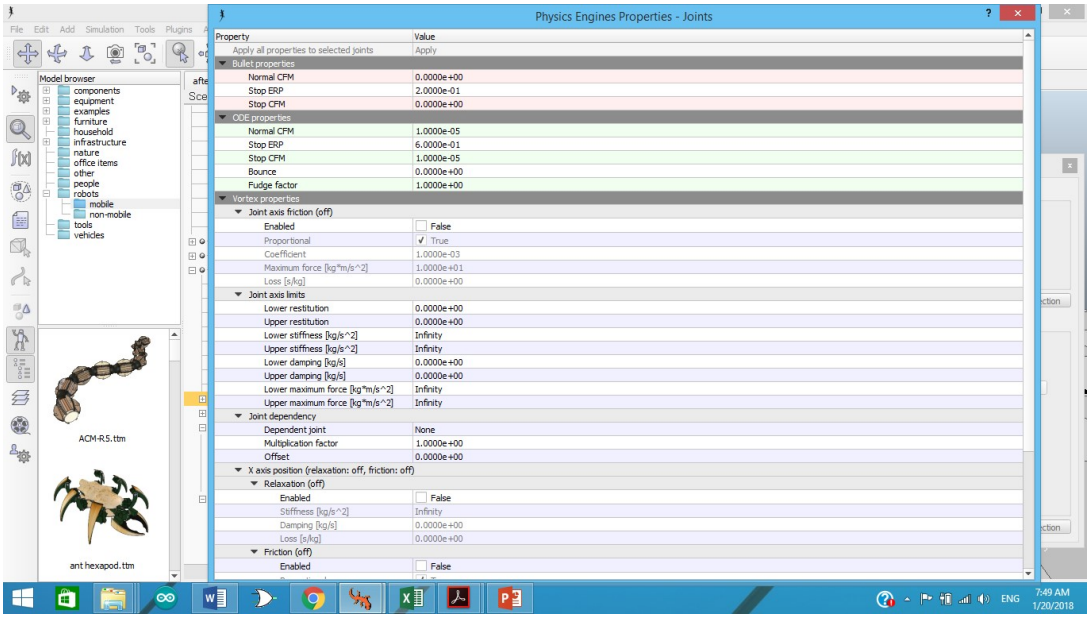


Figure 4.18: Dynamics of rear motors (2)

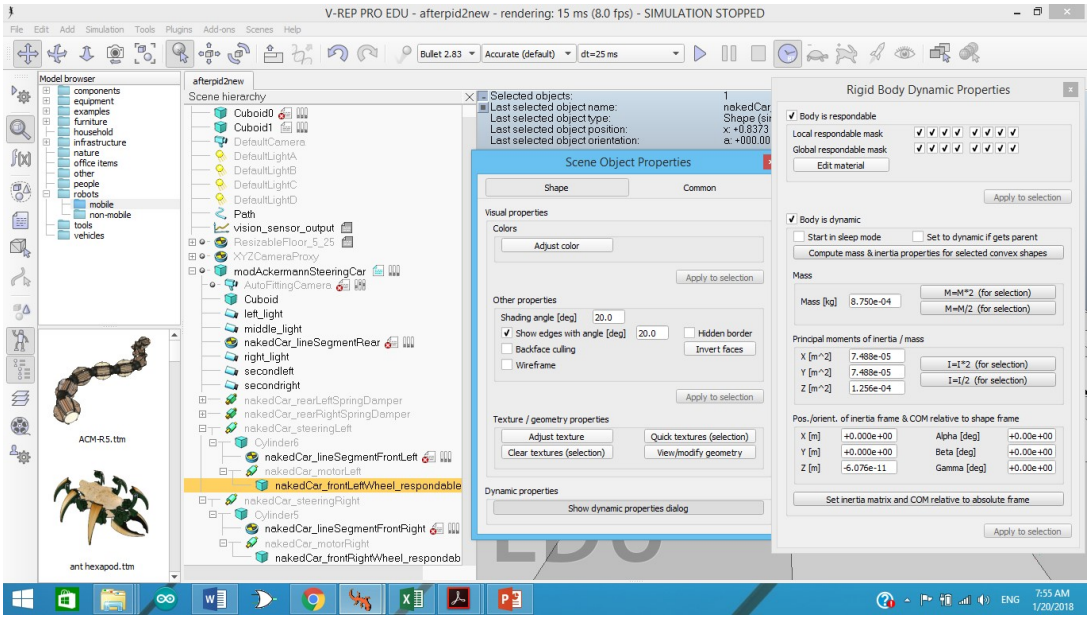


Figure 4.19: Wheel Dynamics

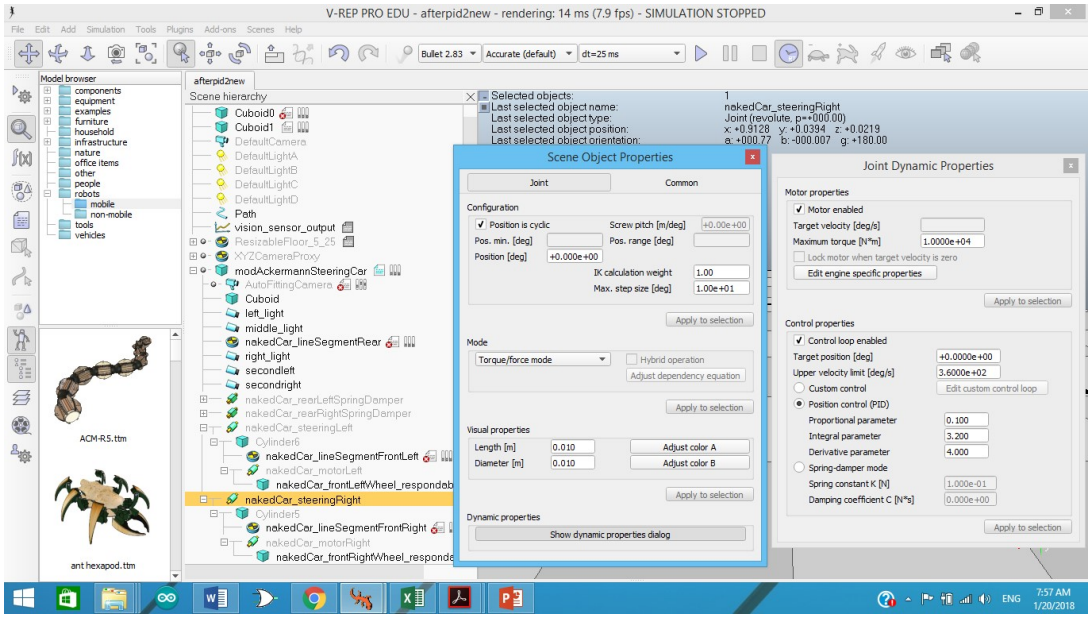


Figure 4.20: Steering Joint Dynamics (1)

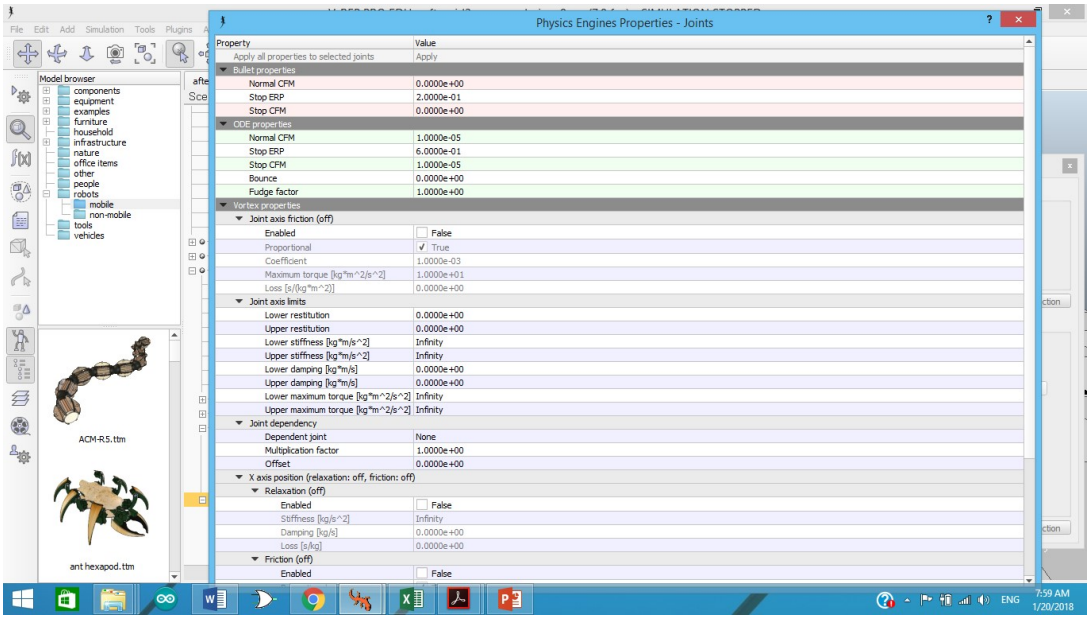


Figure 4.21: Steering Joint Dynamics (2)

Now the system is ready to be controlled and all joints could be actuated. First we will test our line follower control algorithm with on-off control to know whether the position and the number of sensors relative to the track/path is well or not.

4.6.1 V-REP simulation without PID control

We start to write our control algorithm using v-rep main script in lua's language. The following is the V-rep script

- Define the motors
- Define the Vision Sensors
- Initialization
- Control the joints

The code will be included in the appendix.
As an observation, we get the position of the sensors relative to the car, there is huge overshoot. We conclude that we must implement a closed loop control as PID.
Some on-off control have resulted as follows:

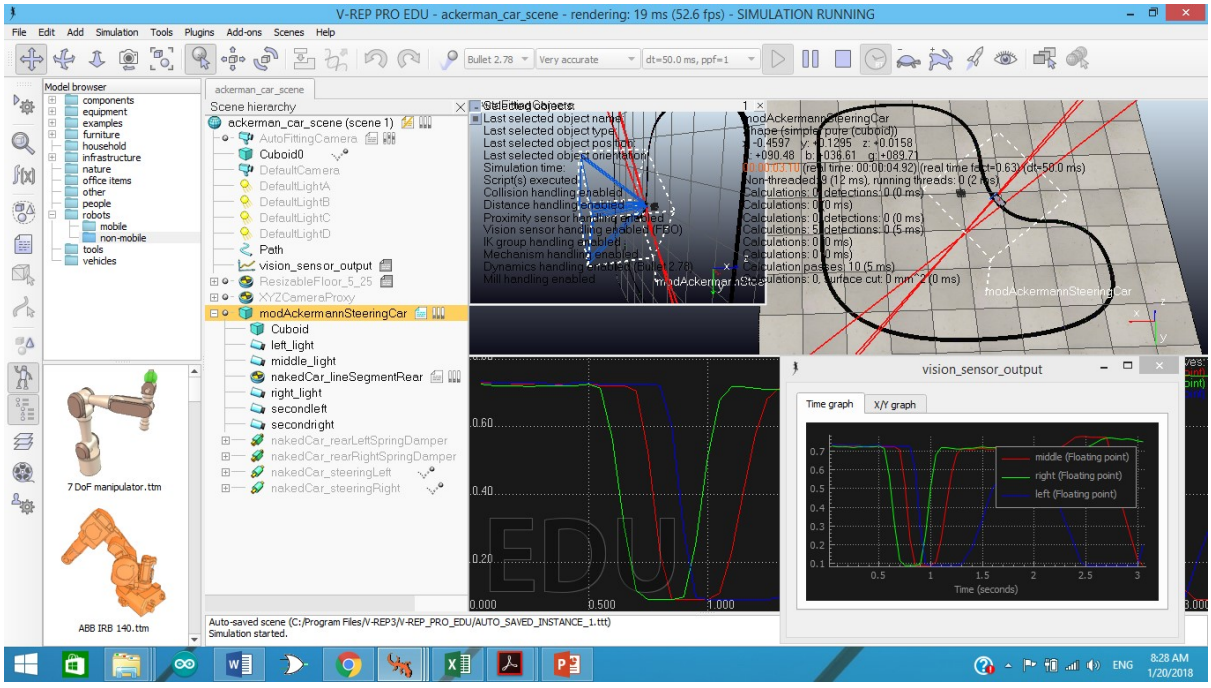


Figure 4.22: Simulation 1 without PID

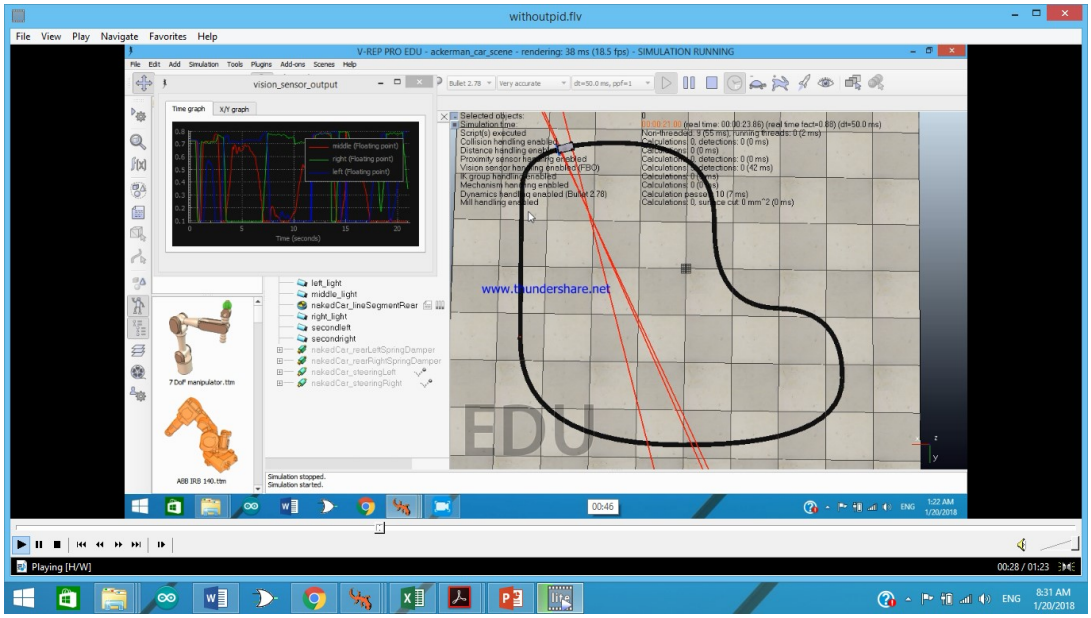


Figure 4.23: Simulation 2 without PID

4.6.2 V-REP simulation with PID control

In the case of the sensors used, an integrated circuit at the module generates as output a simple digital signal (HIGH: Dark; LOW: Light). A potentiometer installed at the module (see photo) will adjust the correct level of light to be considered "dark" or "light". It works on a way that when the reflected light color is black/dark, a HIGH ("1") digital level is generated at its output and a LOW ("0") for another lighter color. I used here an integrated module with 4 sensors and extra module with a sole sensor (different shape, but same logic). The combination is an array of 5 sensors that I found is good for a nice and smooth control, as explained below.

The array of 5 sensors is mounted on a way that if only one sensor is centered with relation to the black line, only that specific sensor will produce a HIGH. By the other side, the space between sensors should be calculated to allow that 2 sensors can cover the full width of the black line simultaneously, producing also a HIGH on both sensors (see the pictures above). The possible sensor array output when following a line are:

- 0 0 0 0 1
- 0 0 0 1 1
- 0 0 0 1 0
- 0 0 1 1 0
- 0 0 1 0 0
- 0 1 1 0 0
- 0 1 0 0 0
- 1 1 0 0 0

- 1 0 0 0 0

Having 5 sensors, permits a generation of an "error variable" that will help to control the robot's position over the line.

Let's consider that the optimum condition is when the robot is centered, having the line just below the "middle sensor" (Sensor 2). The output of the array will be: 0 0 1 0 0 and in this situation, the "error" will be "zero". If the robot starts to driven to the left (the line "seems move" right") the error must increase with a positive signal. If the robot start to move to the right (the line "seems move" left"), in the same way, the error must increase, but now with a negative signal.
The error variable, related with the sensor status will be

- 0 0 0 0 1, Error = 4
- 0 0 0 1 1, Error = 3
- 0 0 1 0 0, Error = 0
- 0 0 0 1 0, Error = 2
- 0 0 1 1 0, Error = 1
- 0 1 1 0 0, Error = -1
- 0 1 0 0 0, Error = -2
- 1 1 0 0 0, Error = -3
- 1 0 0 0 0, Error = -4

Controlling direction, Proportional Control

At this point, our robot is assembled and operational. You should perform some basic tests with the motors, read the output of sensors and testing them over a line. What is missing is the real "brain", the first steps of an "artificial intelligence". We will get this, implementing a control logic that will guarantee that the Robot will be kept following the line.

Suppose that the robot is running over a line and the sensor array output is: "0 0 1 0 0 ". The correspondent error is "0". In this situation, both motors should be running forward with constant speed.
At this point it is clear that as much the Robot driven to one side, bigger will be the error and faster it must return to center. The velocity with the Robot will react to the error will be proportional to it. This is called "Proportional Control", that is the "P" component of a more complex control network, the PID (Proportional, Derivative, Integral).

The real final code will integrated some additional logic and also some variables that must be initialized, etc. During the above explanation I left them out for simplicity, but everything should be clear looking at final code at the appendix.

PID Parameters are extracted from v-rep for each joint .The front steering motor PID

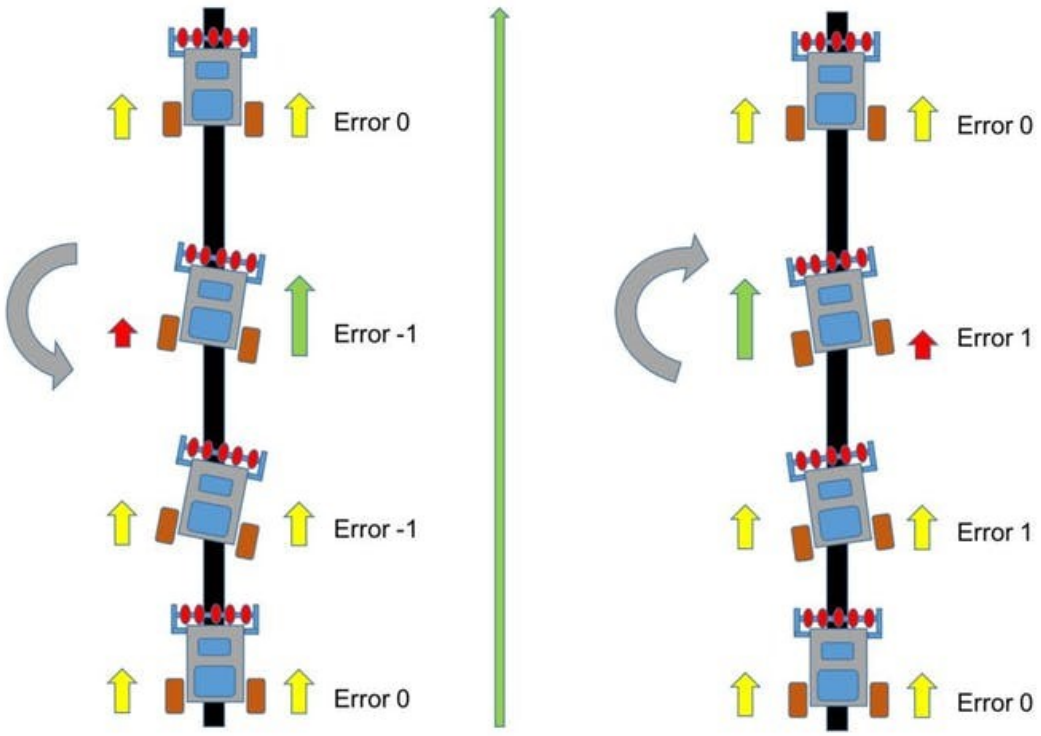


Figure 4.24: Simulation with P

parameters are used in the code and are extracted as shown in figure below using auto tuning in V-rep
The PID simulation results were obtained are shown in figures 4.27 and 4.28

In observation, the overshoot has minimized and the system is in steady state. Now it is time to check the inner and outer steering angles in the simulation with the excel sheet extracted after designing the steering mechanism. As shown in figure below at instant the inner delta was 14 degree while outer was 12.592.
When we compared it with the excel sheet it was the same as shown in below

Inner delta =14
Outer delta= 12.5916

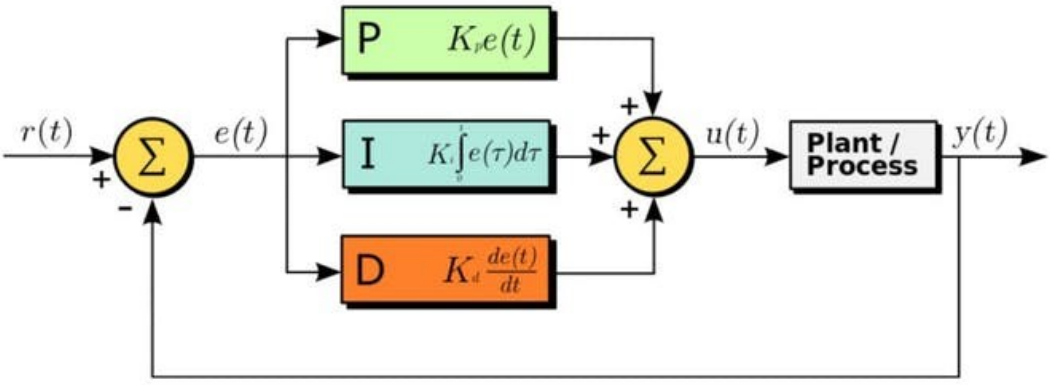


Figure 4.25: PID Controller

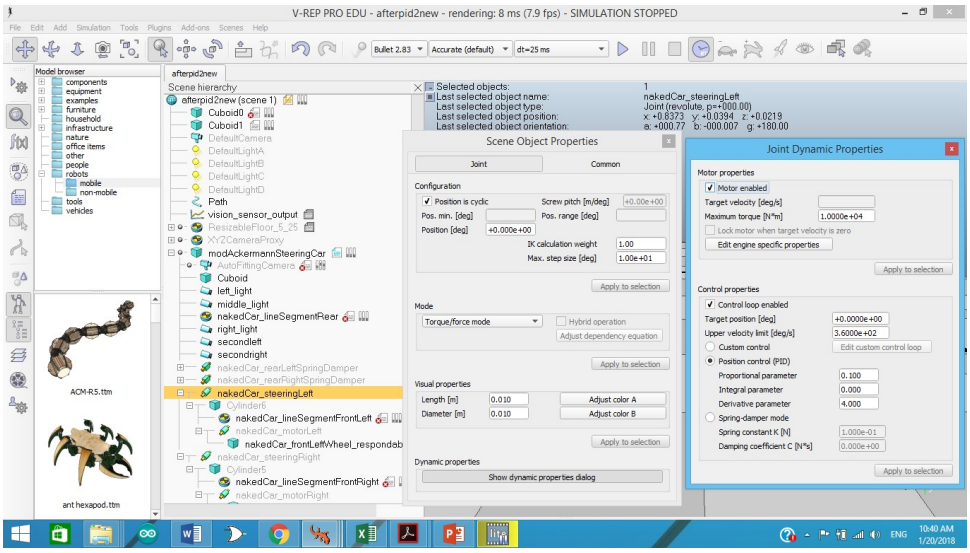


Figure 4.26: PID Values of Steering Motor

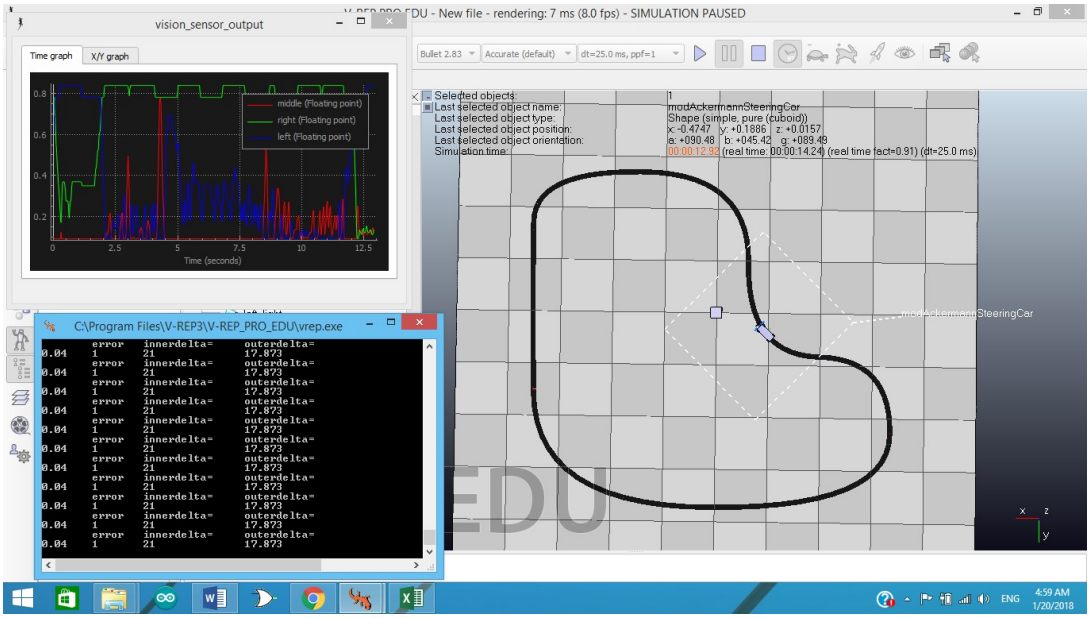


Figure 4.27: Simulation 3 with PID

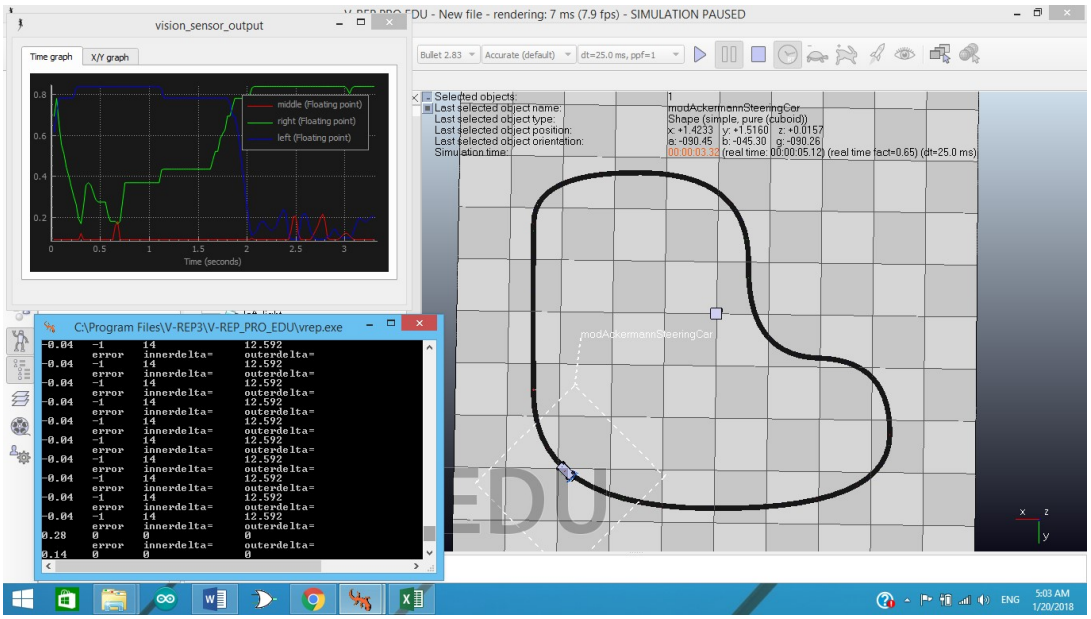


Figure 4.28: Simulation 4 with PID

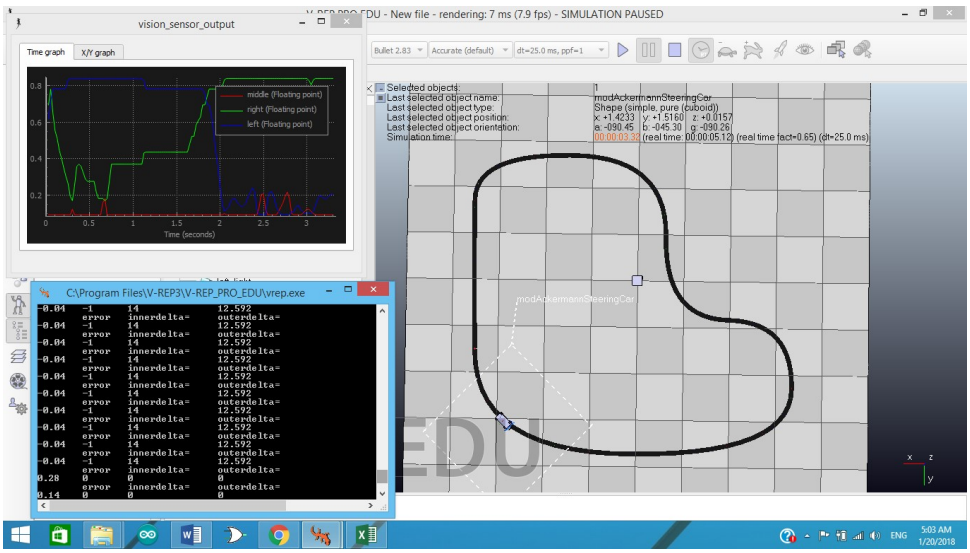


Figure 4.29: Inner and Outer delta angles while simulation

Chapter 5

Docking System

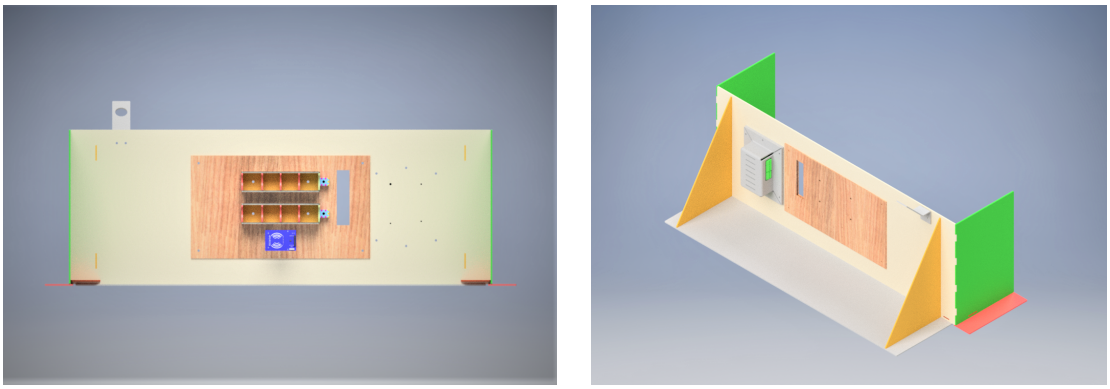
As the vehicle is required to perform in full autonomy without any interference, so it is supposed to have docking for recharging its batteries. The docking is divided into two stations, one is fixed besides an AC power source and the other is fixed at the end of the vehicle.

5.1 Ground Fixed Station

The idea of the design of the Ground Fixed Station taken from a paper written by a students in the department of electronics and automation at University of Brescia [9]. The station has two aluminum fixed connectors one for neutral and one for line, and their wires are connected to a control circuit which allows current to pass using AC relay when the vehicle's station is connected to the Ground Fixed station.

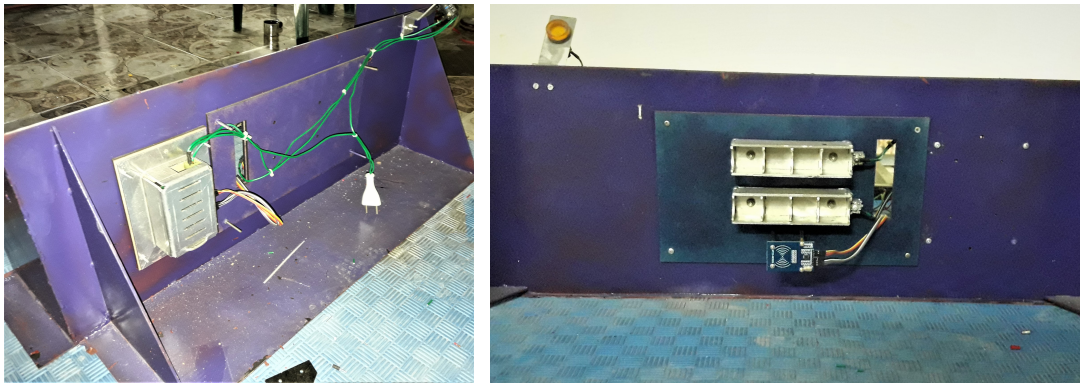
5.1.1 Mechanical Aspect

The station frame dimensions designed to suit the vehicle while docking . The station made of 3 mm steel sheet and webs welded in it in order to support the station while the vehicle moving backwards to dock in the station. Furthermore, the dimensions of the working site at Leoni factory are taken into consideration in the station design after meeting the engineers there and gather site drawing from them.



(a) Fixed Station Frontal view (b) Fixed Station Backward View

Figure 5.1: Fixed Station CAD



(a) After Fabrication (1) (b) After Fabrication (2)

Figure 5.2: docking Station After Fabrication

5.1.2 Electrical and Control Aspect

The two fixed connectors are connected to atmega circuit and Solid State Relay (SSR-40DA) shown in figure 5.3 in order to control the pass of the current from the AC source to the connectors. Those connectors are fixed on a wooden plate as an isolation in order to avoid short circuit, and wired used ring terminals with bolt and nut.

Figure 5.3: Solid State Relay

An RFID reader module shown in figure 5.4 is fixed in the station and connected to the atmega circuit while its card is fixed in the vehicle side in order to pass the current from the AC source to the during docking. The card which is fixed in the vehicle , was tested to get its ID and configured to the RFID reader, so the current will not pass until the vehicle docks in the station.

As the RFID reader reading range is from 0 to 20 mm, 60 mm spacers are used in order to decrease the distance between the RFID and the card to 15 mm while docking.

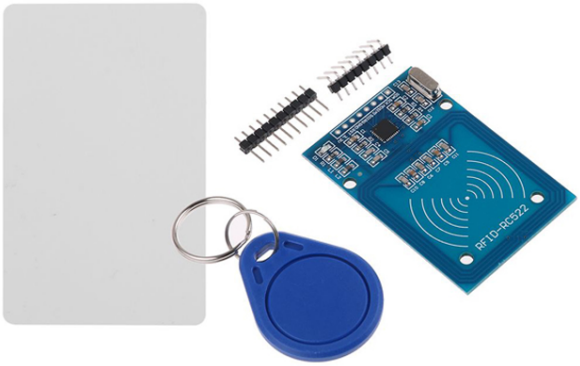


Figure 5.4: RFID module with the Card

Regarding safety aspects an AC indicator is connected in parallel with the two fixed connectors in order to light during docking to notify the workers that the place is hazardous , in addition to the control circuit which is not allowing the current to pass until the vehicle docks in the station.

5.2 Vehicle's Station

In the Vehicle's Station there is two fixed rods connected by wires to the chargers of the batteries, in addition to the RFID card which allows the fixed station identifies the Vehicle. Also, there is two fixed limit switches in order to stop the vehicle when it connects with the fixed station.

The docking station system takes the advantage of the algorithm of the vehicle which is

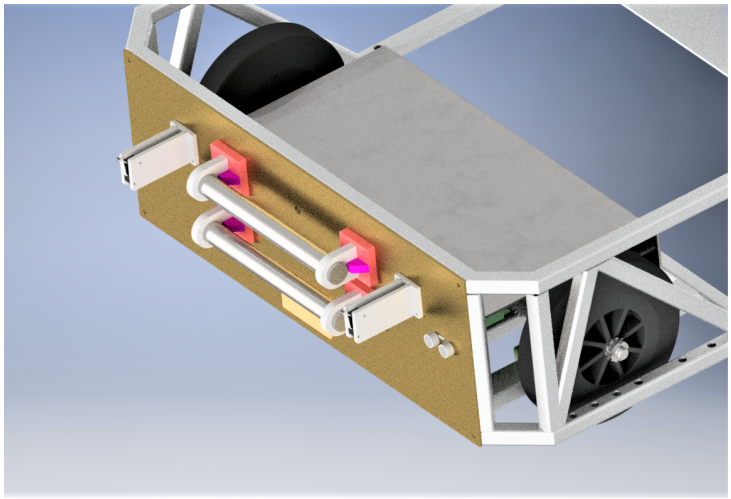


Figure 5.5: Vehicle Docking Integration CAD

based on line follower to reach the desired destination, that will help the vehicle to connect the rods fixed at it with the connectors fixed at the Ground Fixed Station precisely.

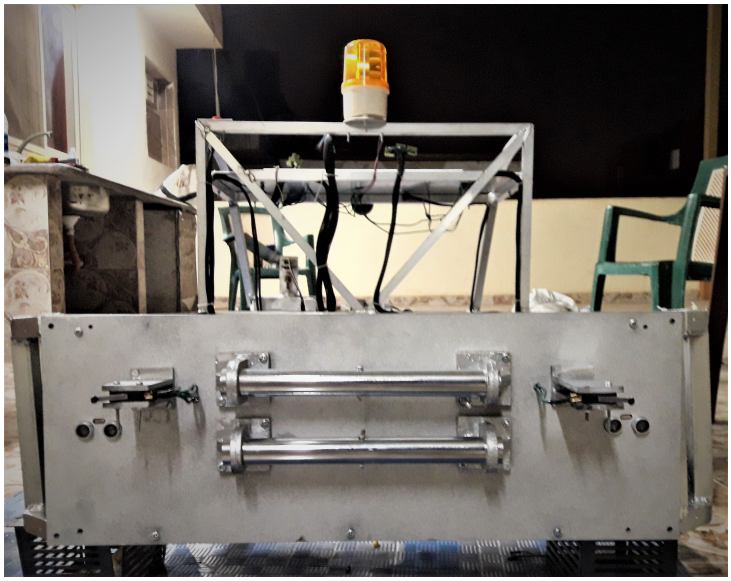
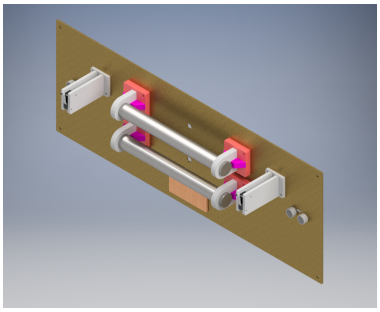


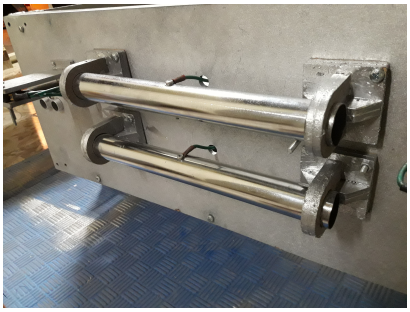
Figure 5.6: Vehicle Docking After Fabrication

5.2.1 Mechanical Aspect

The connecting rods used are from stainless steel, so in order to isolate it from the aluminum frame, wooden plate and hinges are used as a fixation to avoid any sort of electrical shock. Also the fixation of the rods are designed to keep the distance between the center of two rods equals to the distance between the center of the Fixed station connectors.



(a) Vehicle docking



(b) Docking After Fabrication

Figure 5.7: Vehicle Docking Integration

5.2.2 Electrical and control Aspect

In order to transport AC voltage from the stainless steel rods for chargers , 1.5 mm wire is connected to each rod using ring terminal with bolt and nut after drilling a through hole in each rod. Those wires are connected to female sockets which will be connected to the power distributer that feeds the batteries chargers.

As mentioned above there are two limit switches fixed at the vehicle station. The two limit switches are connected in parallel and a signal will be sent to the ECU when both limit switches contact with the Fixed Station in order to stop the vehicle. Using two

limit switches are important to ensure that the vehicle's end parallel to the fixed station and the connecting rods are completely engaged with Fixed Station connectors to ensure the efficiency of the connection , this is clearly shown in figure 5.8 . Also, two ultrasonic senors are used to avoid any obstacle in the way of the vehicle while moving backwards.

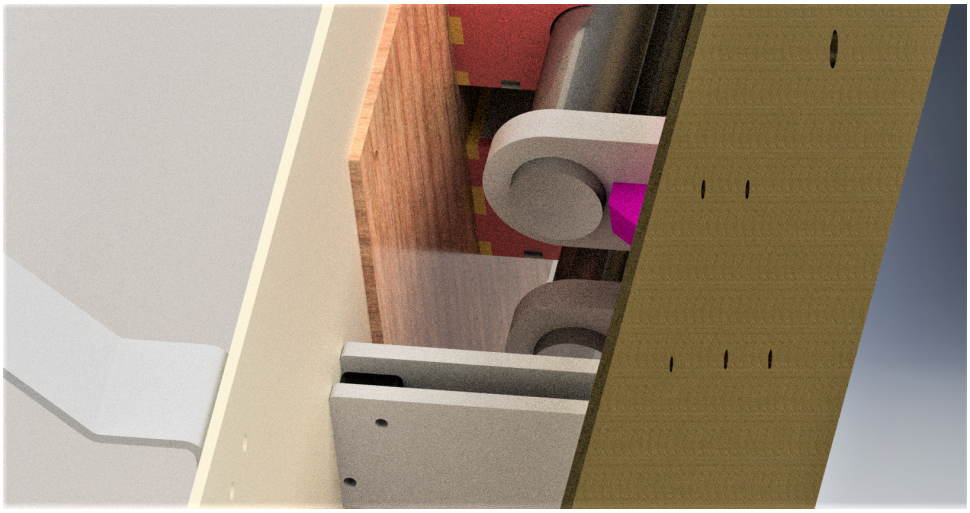


Figure 5.8: Docking Station While Connecting

Chapter 6

Conclusion

6.1 Requirements Review

Initially, this is a reminder review of the requirements set by Leoni's engineers:

- Vehicle's payload : 40 kg maximum.
- Maximum speed of the vehicle: 0.5 m/s
- Maximum track-width:1 m
- Working hours: 1 hour
- Maximum range of obstacle detection: 0.5 m
- Minimum height of obstacles that could be detected: 5 cm
- Minimum height of the vehicle from the bottom to the ground: 0.5 cm
- Minimum height of the vehicle from the top to the ground: 30 cm
- Minimum range of the network: 60 m
- Accuracy of vehicle position on the line: 5 cm
- Number of components supported by the ordering app: 80

6.2 Improved Designs

According to our initial progress, we have fulfilled some of the previously stated requirements and improved the design of the vehicle. To summarize, we have fulfilled and achieved the following:

- Vehicle's payload capacity of 40 kg
- Storage of packages to be transported
- Operating the vehicle with manual and autonomous modes
- Vehicle's maximum requirements (e.g. speed, height, width)

- Enhanced mechanical steering system and body
- Reduced vehicle weight and optimized the previous vehicle's parameters
- Obstacle detection within specified range
- Vehicle line following maneuver simulation using PID control

6.3 Unfulfilled Targets

Despite our progress so far, there has been some targets and objectives that were not fulfilled in the project duration. Some of these targets include:

- Detecting surrounding landmarks using perception.
- Implementing a neural network-based algorithms.
- Constructing a vehicle fiber-glass body.
- Conducting trials inside the factory.

There has been some setbacks that prevented us from fulfilling our final target. These setbacks include:

- Poor manufacturing technology in the Egyptian market.
- Unavailability of some components in the Egyptian market.
- Lack of funding due to not making the agreement from Leoni's side despite our commitment.
- Tight time schedule and narrow duration.

6.4 Project Reference Work Files

All material about this project can be found as a soft copy at our Google Drive [10].

We would like to further acknowledge and elaborate that this is our final thesis presented.

Chapter 7

Appendix

7.1 Steering Angles and equivalent turning radius

Outer Delta	Inner Delta	Turning Radius	Outer Delta	Inner Delta	Turning Radius
0.992341421	1	51.76192748	18.22494159	21.5	2.549103166
1.482829979	1.5	34.57161936	18.57240789	22	2.494003491
1.969581702	2	25.97570339	18.91627471	22.5	2.441352603
2.452631863	2.5	20.81755883	19.25652276	23	2.390992725
2.932013312	3	17.37831204	19.59313201	23.5	2.342779532
3.407756582	3.5	14.92130277	19.92608167	24	2.296580754
3.879889987	4	13.07820085	20.25535025	24.5	2.252274945
4.348439714	4.5	11.64437836	20.58091556	25	2.209750401
4.813429905	5	10.49705848	20.90275471	25.5	2.168904207
5.274882737	5.5	9.558110515	21.22084415	26	2.129641391
5.7328185	6	8.77544728	21.53515968	26.5	2.091874176
6.187255663	6.5	8.113008415	21.84567645	27	2.055521317
6.638210945	7	7.545036505	22.152369	27.5	2.020507501
7.085699369	7.5	7.052642776	22.45521124	28	1.986762828
7.529734331	8	6.621660642	22.75417652	28.5	1.954222327
7.970327644	8.5	6.241256835	23.04923759	29	1.922825538
8.407489599	9	5.903005484	23.34036666	29.5	1.892516128
8.841229008	9.5	5.600254589	23.62753537	30	1.86324155
9.271553251	10	5.327682563	23.91071487	30.5	1.83495273
9.698468321	10.5	5.080981461	24.18987578	31	1.80760379
10.12197887	11	4.856626616	24.46498823	31.5	1.781151794
10.54208822	11.5	4.651706359	24.7360219	32	1.755556519
10.95879846	12	4.463794314	25.002946	32.5	1.730780248
11.37211042	12.5	4.290852341	25.26572929	33	1.706787579
11.78202372	13	4.131155878	25.52434016	33.5	1.683545255
12.18853682	13.5	3.983235871	25.77874656	34	1.661022006
12.59164705	14	3.845833149	26.02891608	34.5	1.639188404
12.99135061	14.5	3.717862241	26.27481598	35	1.618016735
13.38764261	15	3.598382424	26.51641315	35.5	1.597480879
13.78051713	15.5	3.48657438	26.7536742	36	1.577556197
14.16996719	16	3.381721228	26.98656543	36.5	1.558219433
14.55598481	16.5	3.283193003	27.21505289	37	1.539448621
14.93856102	17	3.190433876	27.43910237	37.5	1.521222999
15.31768588	17.5	3.102951563	27.65867948	38	1.50352293
15.69334852	18	3.020308503	27.87374959	38.5	1.486329833
16.06553712	18.5	2.942114461	28.08427794	39	1.469626111
16.43423899	19	2.868020305	28.29022961	39.5	1.453395096
16.79944053	19.5	2.797712735	28.49156958	40	1.437620985
17.16112728	20	2.730909804	28.68826274	40.5	1.422288793
17.51928394	20.5	2.667357099	28.88027391	41	1.407384301

Bibliography

[1] Mostafa Osman, Mohamed Bahaa. *Autonomous Carrier Vehicle (ACV) & Robotic Mechanism*. July, 2017.

[2] VDI 2206, *Design methodology for mechatronic systems*. June 2004.

[3] J.Y. Wong *Theory of Ground Vehicles*, third edition.

[4] Reza N. Jazar *Vehicle Dynamics Theory and Applications*, 2008.

[5] Ubiquity Robotics
`https://downloads.ubiquityrobotics.com/`

[6] ROS Wiki Documentation
`wiki.ros.org`

[7] Muhammad Ali Mazidi. *TI ARM Peripherals Programming and Interfacing Using C Language for ARM Cortex*.

[8] *Tiva^{TM} C Series TM4C1294 Connected LaunchPad Evaluation Kit EK-TM4C1294XL User's Guide*.

[9] Docking and Charging System ,Ricardo Cassinis.

[10] AGCV Google Drive
`https://drive.google.com/drive/folders/0Bz3BFPuHdMFydzlWYzV2NFBPWUU`