Middle East Technical University

Department of
Electrical and Electronics Engineering

EE494 ENGINEERING DESIGN II

Car Chasing Robot
Final Report

Supervisor: Assoc. Prof. Emre Özkan
METU EE / C-112

Project Start: 4/10/2018
Project End: 26/5/2019
Project Budget: $450

Company Name : Duayenler Ltd. Şti.

Members	Title	ID	Phone
Sarper Sertel	Electronics Engineer	2094449	0542 515 6039
Enes Taştan	Hardware Design Engineer	2068989	0543 683 4336
Erdem Tuna	Embedded Systems Engineer	2617419	0535 256 3320
Halil Temurtaş	Control Engineer	2094522	0531 632 2194
İlker Sağlık	Software Engineer	2094423	0541 722 9573

May 10, 2019

Contents

1 Introduction

The self driving cars have been the buzz concept of the community lately. Many companies have established departments and teams to develop their own self driving cars. Increasing research on the concept brings the increasing demand to new technologies and approaches. DUAYENLER Ltd. Şti. (DUAYENLER) aims to catch this trend and be one of the pioneers in this field. To realize this goal, DUAYENLER develops a vehicle for Car Chasing Project. The outcome of the project is a vehicle that can follow a predefined path in an autonomous way. The vehicle not only drive on the path itself, but it will communicate with other vehicles as well.

This project, as most of projects, is a multi disciplinary tasks. It involves image processing, data processing, statistical estimation, networking, mechanical design and mechanical assembly. These disciplines are not adequate by themselves but also integration with each other. So, system design, testing and automation are key concepts to merge everything into a product. Fortunately, DUAYENLER is composed five team members who build up proficiency in those fields.

The fundamental blocks of the project are autonomous steering and its control. To build up the fundamental blocks and the project, several systems and subsystems are developed in a reasonable way. Firstly, the objectives of the project was determined. Secondly, the system was divided into six main system and the requirements for these subsystem were determined considering the project objectives. Thirdly, these subsystems then divided into two main subsystems. And the requirements for theses subsystems were constructed considering their parent system requirements. Lastly, the subsystems were designed considering these requirements.

This report includes detailed technical solutions for the subsystem blocks, their requirements and the respective justifications for the requirements. In addition to these detailed solutions, the report includes detailed test steps and the discussion on these tests for the subsystems of the project. Lastly, this report includes detailed power and cost analysis for the project and the Gantt chart for the project management purposes.

2 Design Description

THIS PART IS REWRITTEN FOR FINAL REPORT.

The ultimate objective of the project is to design and manufacture a self driving vehicle. The vehicle meets certain criteria that are defined in Standard Committee Report. The project is composed of six main systems together with twelve subsystems. The overall top-down organization of the project is shown in *Figure 1*.

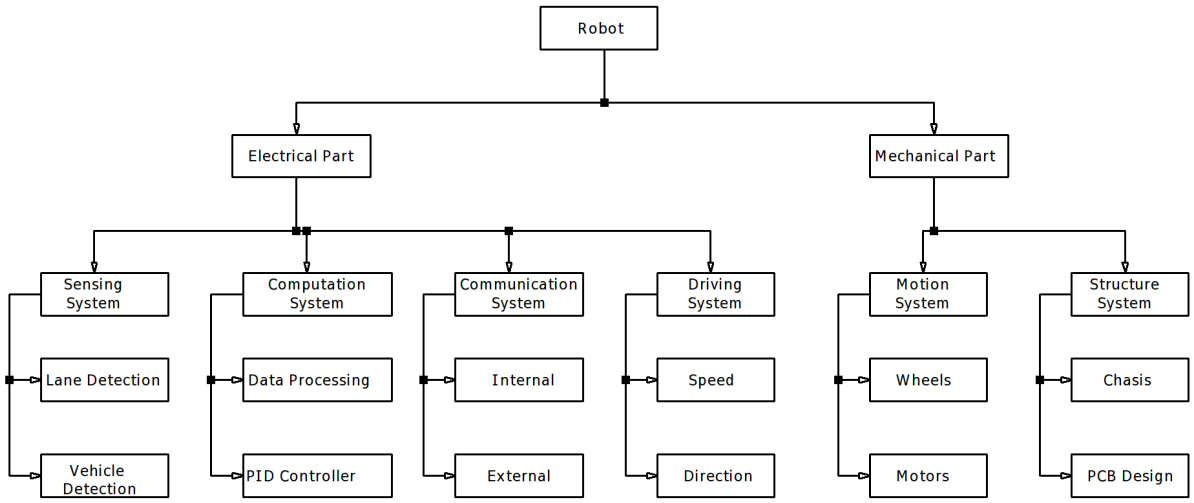


Figure 1: Organization of the Project

V-Model provides a tool for companies to structure and track their product development processes. DUAYENLER constructed V-Model for the project as shown in *Figure 2*.

This section includes finalized system and subsystem level design descriptions. The ultimate algorithms, calculations, theoretical approaches and diagrams are presented in the related subsections.

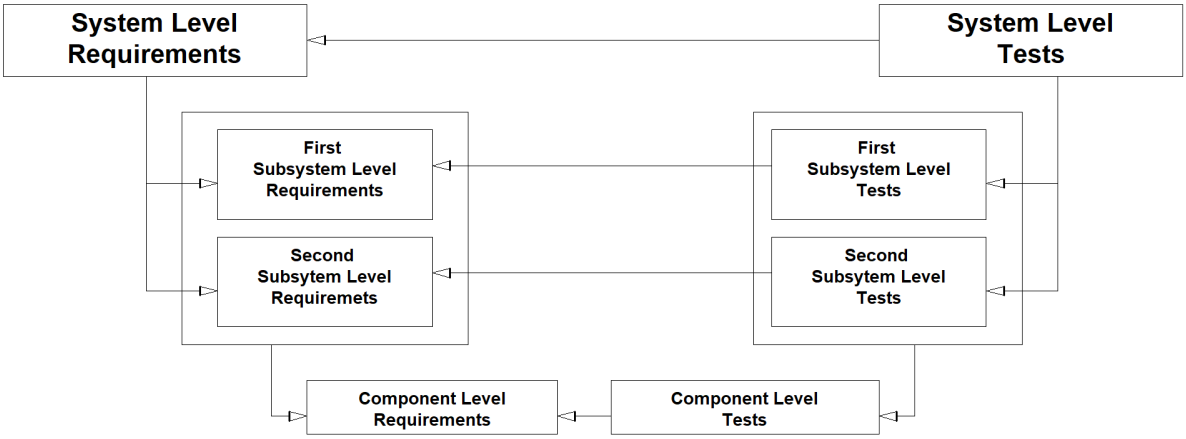


Figure 2: V-Model of the Project

2.1 Sensing System

This system is responsible for interpreting data from the environment. And the requirements for this system are as follows;

1. The system should detect the sides of the road.
2. The system should not be effected from external disturbances.
3. The system should detect the opponent vehicle.

The system has two subsystems namely,

- **Lane Detection Subsystem** is responsible for detecting sides of the path as its name suggests.
- **Vehicle Detection Subsystem** is responsible for detecting opponent vehicle if it is close to the vehicle more than 5 cm.

2.1.1 Lane Detection Subsystem

THIS PART IS REWRITTEN FOR FINAL REPORT.

A. Requirements for the Solution

1) The subsystem should be able to detect only the shades of green color.
2) The subsystem should be able to detect edges in the camera frame in any light condition.
3) The subsystem should be able to extract lane lines out of captured frame.

B. Solution for the Subsystem

The task of the subsystem is to detect the lane lines. The tool utilized to realize the task is OpenCV libraries together with developed pipelined algorithm. The block diagram of the subsystem is given in *Figure 3*.

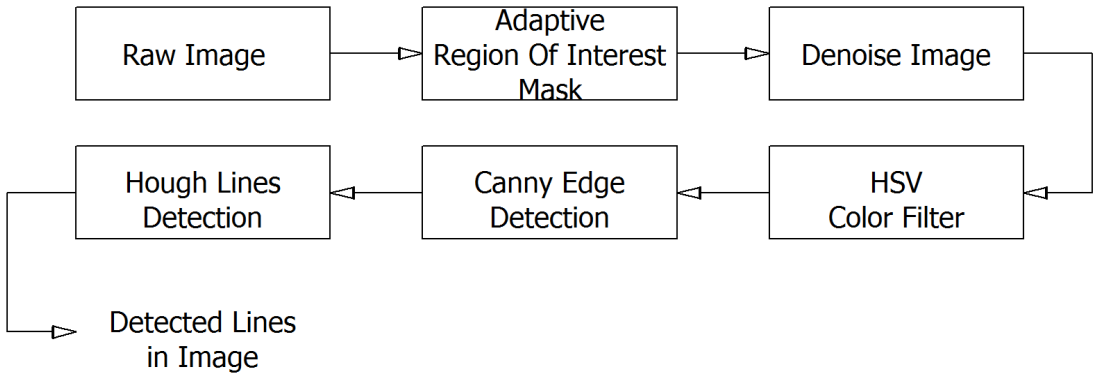


Figure 3: Block Diagram of the Lane Detection Subsystem

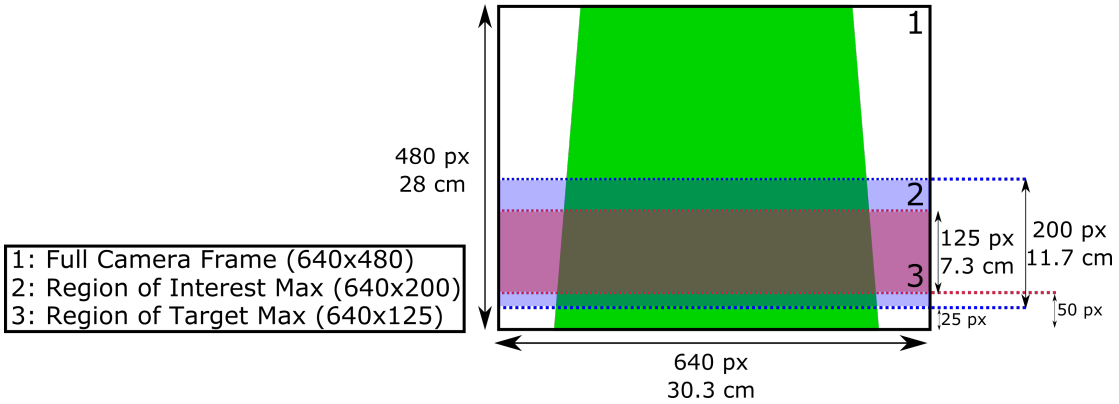


Figure 4: Explanation of Frame Regions

The input to this subsystem is provided by Raspberry Pi camera mounted on top of the vehicle. The camera frame resolution is 640x480 px. One thing to for the camera frame is that horizontal mapping and the vertical mapping of the pixels are not the same. That is, same amount of pixels in both directions do not correspond to same length in real life. The proposed solution first masks out a region of interest (ROI) of $200x640px$. The visual explanation of the frame sections is provided in *Figure 4*. The masking eliminates the process of excessive data and increases the process speed of the pipeline. Formerly, ROI size was fixed in $200x640px$. However, when the opponent vehicle starts to appear in ROI, this was causing wrong line detection in the subsystem. To eliminate such behaviour, adaptive ROI is implemented to avoid such improper situations. Adaptive ROI determines ROI size dynamically with the front distance sensor measurement provided by Vehicle Detection Subsystem and scales ROI size (only in vertical direction) with a linear function. The equations that define the height of ROI are as follows:

$$ROI\_Width = 640px \tag{1}$$

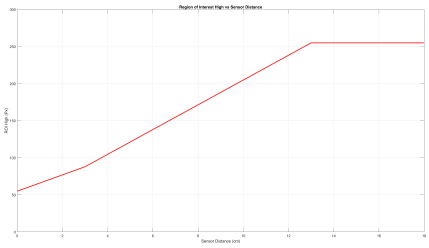
$$ROI\_Height = (ROI\_High - ROI\_Low)px \tag{2}$$

$$ROI\_Low = 25px \tag{3}$$

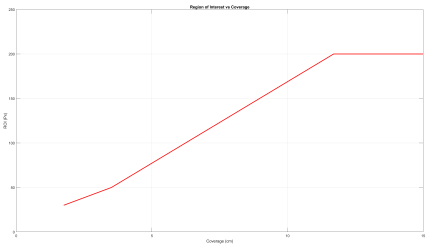
$$ROI\_High = \begin{cases} 225, & front\_distance \geq 13cm \\ 14 * (front\_distance) + 43, & 3 \leq front\_distance < 13cm \\ 10 * (front\_distance) + 55, & front\_distance \leq 3cm \end{cases} \tag{4}$$

The graphical representation of the equations are given in *Figure 5*

As the next step of the processing pipeline, the target color green is filtered by applying Gaussian denoise (with zero mean) and HSV filters. The lower bound for HSV filter is [H=60, S=120, V=106] and the upper bound is [H=82, S=255, V=245]. This process sets the pixels that are in the green threshold to white and the rest to black. Next, the edges are detected by Canny edge detector. As edges are found, the pixels that may constitute a line are found by Hough line detector. The resulting output is an array of coordinates in the form of $[x_1, y_1, x_2, y_2]$ where



(a) Graph of ROI_High vs front_distance



(b) Right, ROI_Height Variation

Figure 5: Graphs Defining ROI_Height

(x_1, y_1) is the starting point of the line and (x_2, y_2) is the end point of the line. The found coordinate array is passed to Data Processing Subsystem.

C. Discussions on the Solution

The main structure of the proposed solution has not changed since Conceptual Design Review Report. An addition to process pipeline is introducing adaptive ROI. This is done to be able to follow opponent vehicle on the back.

For this subsystem to be stable, HSV filter must produce a clean filtered result. The filter is responsive as long as it is tuned according to light condition.

2.1.2 Vehicle Detection Subsystem

1. Requirements for the Solution

   (a) The subsystem should detect the opponent to be caught with in a 5 cm

   (b) The subsystem should detect the chasing opponent if it reaches from back with in a 5 cm

   (c) The subsystem should trigger the handshake protocol

2. Solution for the Subsystem

   The subsystem is the first step of safely competing with an opponent in a racing path. This subsystem uses two time of flight distance sensor which is enhanced IR sensor. One at the back of the vehicle responsible for detecting the chasing opponent and one at the front of the vehicle responsible for detecting the chased opponent. The subsystem produces positive output if the chasing vehicle or chased vehicle is within a range of 5 cm from the vehicle. Since the sensor reading is performed using Raspberry Pi, the required trigger for handshake protocol can be easily accessed by the external communication subsystem.

3. Discussions on the Solution

   The proposed method is not entirely different than the one presented in Conceptual Design Report. The only difference is the devices reading the sensors. In Conceptual Design Report, sensors were connected to Arduino. However, the team decided that it is problematic. For example, in the case of sensor readings from Arduino, when opponent is close, Arduino reads the sensor, sends it to trigger handshake, then Pi sends/receives TCP messages, then sends Arduino to stop the motors. On the other hand, reading sensors from Pi simplifies and accelerates the work by eliminating the step to send sensor reading. Also, since Raspberry Pi is a general-purpose computer, it can handle with complex tasks easily, while Arduino cannot. Therefore, sensors are moved from Arduino to Raspberry Pi.

2.2 Computation System

THIS PART IS REWRITTEN FOR FINAL REPORT.

This system is responsible for computational works of the vehicle. The system mainly give meaning to data generated by the sensing system. The requirements for this system are as follows:

- The system should be able to produce middle line to follow
- The system should be able to control the robot

The system has two subsystems namely,

1. **Data Processing Subsystem** is responsible for processing the output data of lane detection unit and produce data for PID control unit.
2. **PID Controller Subsystem** is responsible for controlling the motors of the vehicle.

2.2.1 Data Processing Subsystem

THIS PART IS REWRITTEN FOR FINAL REPORT.

A. Requirements for the Solution

   1) The subsystem should be able to analyze data produced by Sensing System.
   2) The subsystem should be able to produce the angle information that is sent to the Controller Subsystem.
   3) The subsystem should be compatible with Raspberry Pi.
   4) The subsystem should be able to process one frame at most in 100 milliseconds together with Lane Detection Subsystem.
   5) The subsystem should be able to ignore disturbances on the path.

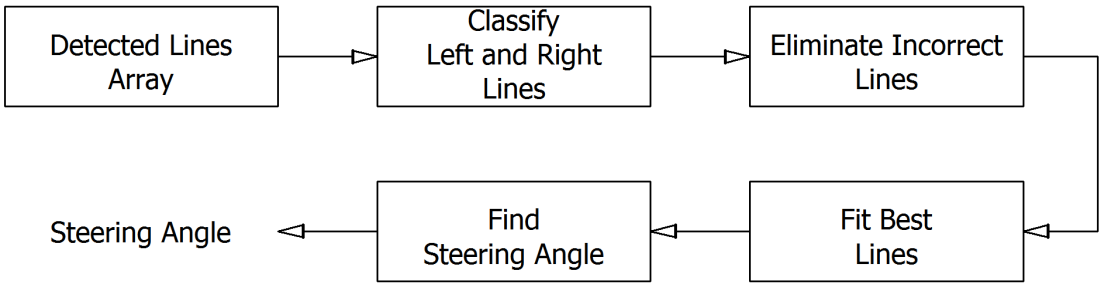


Figure 6: Block Diagram of the Data Processing Subsystem

B. Solution for the Subsystem

The task of this subsystem is to extract control parameters so that the vehicle can follow the path without falling on the ground. The input of the subsystem is the line coordinate array produced by Lane Detection Subsystem. The input is processed by a detailed algorithm and the outputs are lane angle and distance of vehicle to the right lane. The output parameters are explained in 2.2.2. The outputs are generated for the region of target (ROT) that is shown in *Figure 4*. The Lane Detection Subsystem extracted ROI out of full camera frame. The Data Processing Subsystem processes the data in the ROI but produces output for the ROT. The reason for such a distinction between frame regions is that, ROI is good to determine possible obstacles on the path but producing control outputs that are almost 12 cm away from the vehicle would decrease the performance of the PID Controller Subsystem. For this reason ROI is not used, instead ROT is used. As ROI is adaptive, ROT must also be adaptive to stay in ROI region. The equations that define the ROT are as below. The graphical representation of the equations are given in *Figure 7*.

$$ROT\_Width = 640px \tag{5}$$

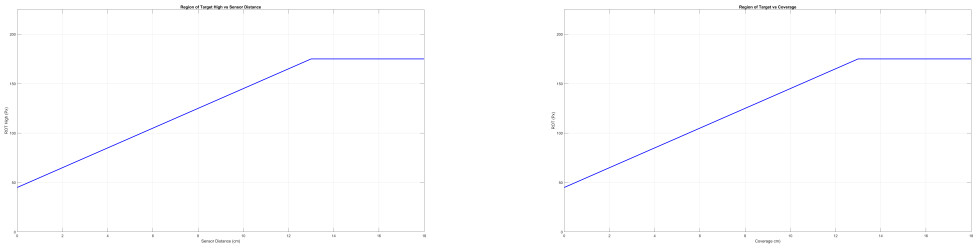
$$ROT\_Height = (ROT\_High - ROT\_Low)px \tag{6}$$

$$ROT\_Low = 50px \tag{7}$$

$$ROT\_High = \begin{cases} 175, & front\_distance \geq 13cm \\ 10 * (front\_distance) + 45, & 3 \leq front\_distance < 13cm \\ 21 * (front\_distance) + 12, & front\_distance \leq 3cm \end{cases} \tag{8}$$

There are four main steps to determine lane angle and distance of vehicle to the right lane. The first step is to classify the lines as left and right. The second step is to eliminate the possible incorrect lines, if any. The third step is to fit the best lines through the left and right lines and reduce the total number of lines to two that are left and right lane lines. The last step is to find control outputs. The block diagram of the subsystem is given in *Figure 8*.

Classifying a line as left or right requires the knowledge of the center of the path. If a pixel is part of the path, it is indicated by pixel value 255, that is result of HSV filtering. So, the path is constructed by white pixels. Analogously, if white pixels on a column is counted, that counts yield an information regarding the start and



(a) Graph of ROT_High vs front_distance (b) Right, ROT_Height Variation

Figure 7: Graphs Defining ROT_Height

end points of the path. The explanation will be made based on an example frame (with ROI masked) as in *Figure* **??**. Obviously, the white pixel count through every column in the arrow directions between red bars is 0. However, from red bars to yellow bars, white pixel count in columns starts to increase. The maximum pixel count in a column is known from ROI equations. If the column indices that are close to maximum pixel count in a columns can be determined, then the right and left bounds of the path are found. Then, the center of the image is simply $(right\_bound + left\_bound)/2$. *Algorithm 1.* As the center of the image is found, lines can be separated as left or right according to their coordinates.

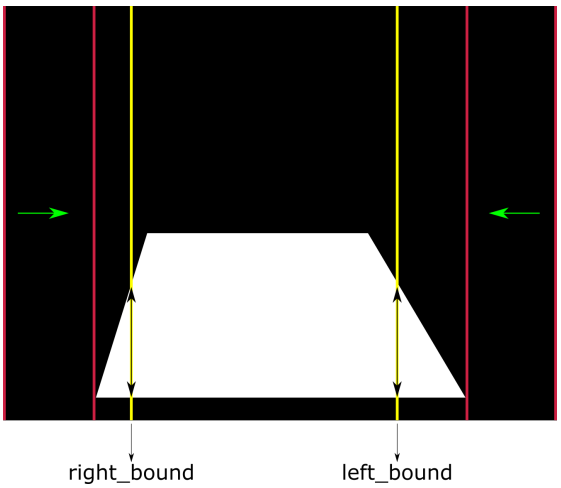
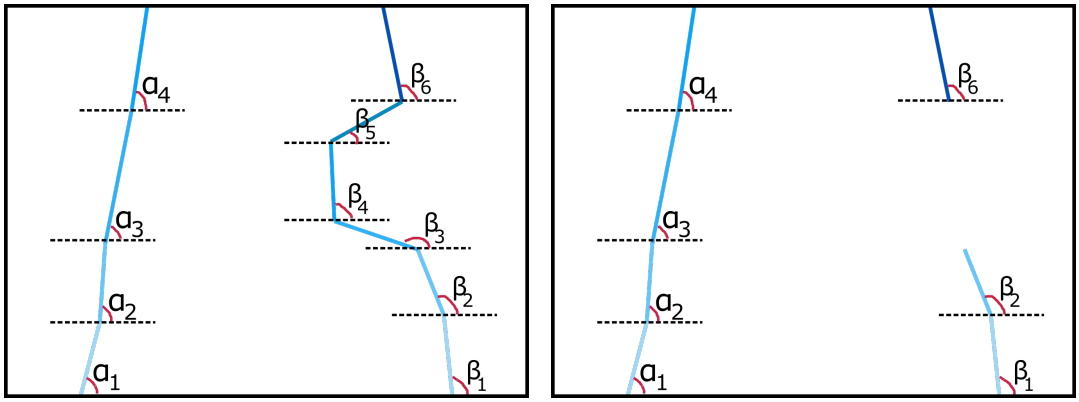


Figure 8: Block Diagram of the Data Processing Subsystem

The next step is to determine whether there are disturbances on the lane lines or not. This is the most complex part of the Data Processing subsystem. Actually the correctness of the steering angle depends on how successful this step is realized. The idea behind this step is evaluating the slopes consecutive lines and assessing whether change in the slope is ordinary or abnormal. The *Figure 9* exemplifies a possible scenario. In this figure, the blue lines represent the detected lines in ROI whereas α and β represent the slopes of the detected lines. Clearly, there are no disturbance on left lines since α values are similar to each other. However if β values are observed, possibly there is an obstacle on right line covered by β_3,

Algorithm 1: Finding Image Center

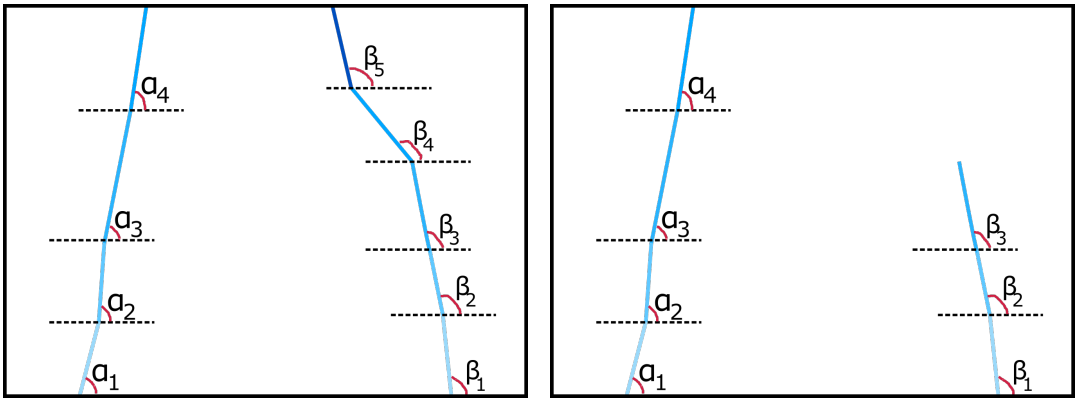
```
whitePixels[640]
confidenceCount = 190//thresholdPixelCount
for Every Row i do
    for Every Column j do
        if frame[i][j] == 255 then
            whitePixels[j]++
for Elements of whitePixels from left to right do
    Find the first index that has white pixel count greater than confidenceCount;
    That index is left_bound;
for Elements of whitePixels from right to left do
    Find the first index that has white pixel count greater than
     confidenceCount; That index is right_bound;
image_center = (left_bound + right_bound)/2
```



(a) Before Eliminating Incorrect Lines (b) After Eliminating Incorrect Lines

Figure 9: A Sample Scenario on Eliminating Incorrect Lines

β_4 and β_5. This can be concluded by observing slope differences $(\beta_2 - \beta_3)$ and $(\beta_5 - \beta_6)$. To ignore this obstacle, it is enough to remove lines with slopes β_3, β_4 and β_5 as in *Figure 9b*. Even though the count of lines is decreased, elimination of incorrect lines are realized and the best line fit will be more correct. Another scenario is shown in *Figure 10*. Again the shown lines are the ones in ROI. In this scenario, left line has no problems. Right lines, however, a bit problematic. The problem is revealed when $(\beta_3 - \beta_4)$ is observed. To determine whether $(\beta_1, \beta_2, \beta_3)$ or (β_4, β_5) is the correct set of lines, left lines are observed and the set which is more symmetric to left lines are selected as right lines. The resulting correction is shown in *Figure 10b*. This is the basic idea behind eliminating incorrect lines in Data Processing subsystem. This idea is generalized by considering other possible obstacle types and shapes. The generalized idea is complicated and would take too long to present here. The summarized idea is presented in *Algorithm 2*.



(a) Before Eliminating Incorrect Lines (b) After Eliminating Incorrect Lines

Figure 10: Another Scenario on Eliminating Incorrect Lines

Algorithm 2: Line Elimination Algorithm

```
array[n]lines
array[n]line_slope_angles
array[n − 1]slope_angle_differences
slopeAngle_threshold
slopeAngle_difference_threshold
for slope_angle_differences do
    if kth item > slopeAngle_difference_threshold then
        // Check kth and (k + 1)th items in line_slope_angles array
        if kth item in line_slope_angles array > slopeAngle_threshold then
            Mark index k in lines array problematic
        else if (k + 1)th item in line_slope_angles array > slopeAngle_threshold
         then
            Mark index (k + 1) in lines array problematic
Delete the lines between problematic indexes
```

The third step is to fit best lines through the remaining lines. This is realized by using built-in Least-Squares method. As a result of this step, the number of lines is dropped to two as left line and right line.

The last step is to find the steering angle. The target point is determined as the average of the middle points left and right lines. So the target point is always in the form of $(x_{avg}, 305)$. The y-coordinate is found by simple math (referencing from *Figure* **??**) $480px - 50px - 125px$. The current point of the vehicle is always $(320, 480)$. So the line connecting two points to each other constitutes the track path and the *arctan* of the slope gives the steering angle. Steering angle is in the $[-90, 90]$ range where negative values indicate to turn left and positive values indicate to turn right. This output is sent to PID Controller subsystem.

C. Discussions on the Solution

The proposed algorithm is mostly the same as in Conceptual Design Review Report. An improvement is made on the way algorithm determines the center of the image. With this new approach, image center is always determined correctly. Line classification and obstacle elimination show satisfactory results regarding robustness.

2.2.2 PID Controller Subsystem

A. Requirements for the Solution

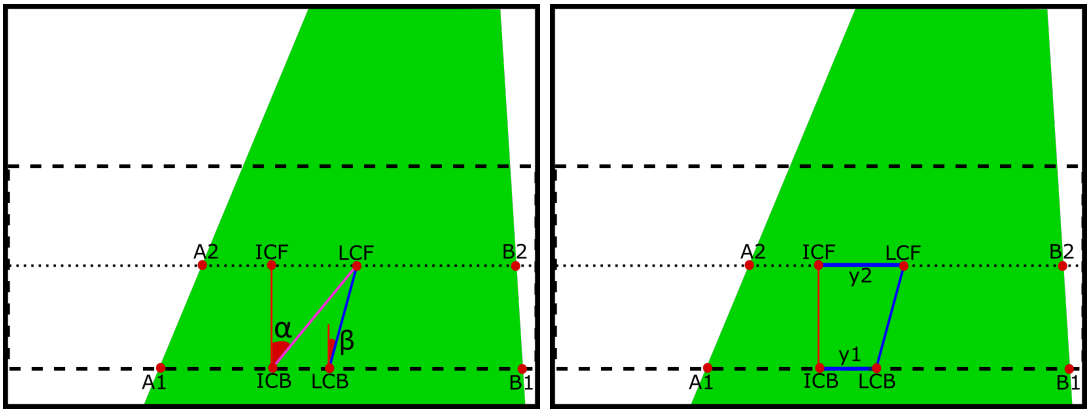
1) The subsystem should be able to control the motors
2) The subsystem should be able to react the external disturbances

A. Solution for the Subsystem

PID Controller Subsection ,as its name suggests, is the main controller element of the vehicle that is responsible for controlling the lateral movement of the vehicle. As the achieved purpose is to stay in the middle of the lane, this subsystem creates a PWM differences between motors in order to rotate the vehicle via differential drive.

For that purpose, the Data Processing Subsystem produces the necessary feedback elements for this subsystem. For the control purpose, in ideal circumstances data processing unit determines eight main point on its vision to create processed variables as in *Figure 11.* These can be explained namely as;

- **A1 & A2:** Beginning and end points of left line at ROT (Region of Target).
- **B1 & B2:** Beginning and end points of right line at ROT.
- **Image Center Back (ICB):** Beginning point of our heading line in ROT.
- **Image Center Front (ICF):** End point of our heading line in ROT.
- **Lane Center Back (LCB):** The middle point of the lane at the starting of the ROT. Can be found by averaging $A1$ & $B1$.
- **Lane Center Front (LCF):** The middle point of the lane at the end of the ROT. Can be found by averaging $A2$ & $B2$.



(a) Controllable Angle Variables (b) Controllable Distance Variables

Figure 11: Controlled Variables of the System

By utilizing these points and their coordinates, the data processing can produce four main variables that can bee used for PID controller and speed subsystems. These are;

- α**:** The angle between the current direction of the vehicle and the direction the vehicle should follow in order to arrive at point **LCF**. It is a main controlled variable for lateral position control with angle variable.
- β**:** The angle of the line that connects the points **LCB** and **LCF**. It represents the angle of the lane, and it can be used for longitudinal movement control in speed subsystem.
- **y1:** The instantaneous distance error of the vehicle from the center line. It can be calculated by subtracting the x-coordinate of **LCB** from the x-coordinate of **ICB**. Due to delays in the system, it is not fed to controller. However, it is a quite useful variable for observing the system.
- **y2:** The expected distance error of the vehicle from the center line at the end of ROT. It can be calculated by subtracting the x-coordinate of **LCF** from the x-coordinate of **ICF**. This results in a distance in a scale of pixels, to convert this to a distance in centimeter, the error can be multiplies by a constant. It is a main controlled variable for lateral position control with distance variable.

Modelling the Plant

Modelling a plant is a good practice in controller design applications, however, in our case the model for the vehicle is unstable, thus applying a bump test as in *Figure 12* results with a exponentially increasing processed data 'y2'. Thus, in this project, our aim is to apply bump test to closed loop system as in *Figure 13* with a known P-controller. An approximate plant model from there can be found as follows;

$$T(s) = \frac{G_c(s)G_p(s)}{1 + G_c(s)G_p(s)}$$

If the overall step response can be modelled resulting with $T(s)$

$$\boxed{G_p(s) = \frac{T(s)}{G_c(s) - T(s)G_c(s)}}$$

Using this plant model, parameters for PID controller can be designed using *Matlab Simulink*.

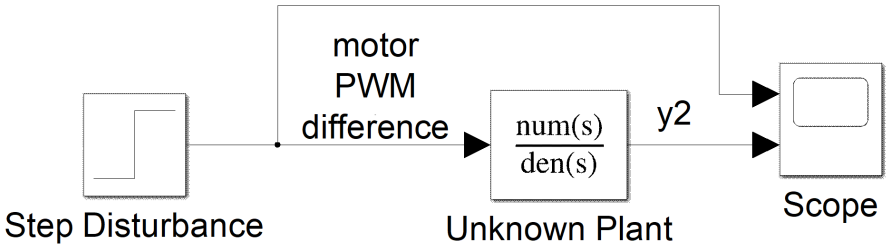


Figure 12: Bump Test for the Unknown Plant

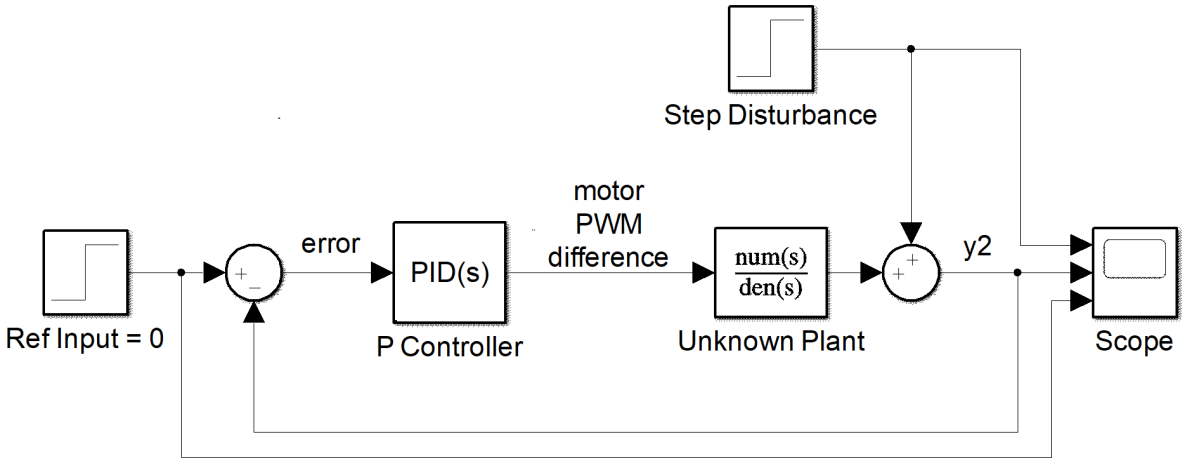


Figure 13: Bump Test for the Closed Loop System

Design & Implementation of the Controller

General PID controller can be expressed in *Laplace* domain as

$$G_c(s) = K_c(1 + \frac{1}{\tau_I s} + \tau_d s)$$

This is a transfer function that accepts the error signal as its input as in *Figure 13*. Since the reference input is always zero for our case, in other words, it is desired that the variables α and $y2$ is equal to zero for all time instants. Therefore, for our case, the error is equal to the negative version of controlled variable.

For implementation, the angle variable α and the distance variable $y2$ can be fed to the Arduino board by the help of *Internal Communication Subsystem*. The implementation is a Arduino code written to produce a PWM-offset value from the error data. The PID parameters found using the simulink can be inserted in this code easily.

A. Discussions on the Solution

Although the basic idea behind control algorithm is same as the algorithm it has been significantly improved after the conceptual design report to compensate our needs and handle the resources the Data Processing Subsystem have. The **PID Controller Subsystem Test** explained in **Section 3** were not fully executed due to integration problem between subsystems. However, the test that were conducted for the bump tests were promising.

2.3 Communication System

This systems is responsible for all communication responsibility of the system. It is one of the most crucial systems of this project since it is responsible for safe communication between subsystems with each other. It is also responsible for communication with other vehicles. And the requirements for this system are as follows;

- The subsystem should ensure safe internal communication
- The subsystem should ensure safe external communication

The system has two subsystems namely, ,

1. **Internal Communication Subsystem** is responsible for communication inside the vehicle mainly the communication between Raspberry Pi and Arduino.
2. **External Communication Subsystem** is responsible for the communication of the vehicle with the outside world mainly with the opponents.

2.3.1 Internal Communication Subsystem

A. Requirements for the Solution

1) The microcontrollers should be able to communicate with each other via serial communication
2) The internal communication speed should be compatible with the processing speed of the lane detection subsystem

B. Solution for the Subsystem

This subsystem covers the communication of the components inside vehicle. Currently, Raspberry Pi and Arduino are two components that requires communication. To prevent the large amount of cable connection, a serial communication protocol is implemented.

The serial communication is implemented via USB port. Since RPi is practically a computer, it can recognize Arduino as a device using a serial port such as /ttyUSB0 in case of a Linux based OS. When recognized, RPi can send any piece of strings to the Arduino via USB cable. The process of communication is as follows:

(a) Arduino should be connected to the Pi.
(b) Using Arduino IDE or any other method such as listing serial ports and checking for Arduino and so on, the serial port name should be detected
(c) Baud rates of two sides should be the same. 9600 is generally enough but if needed, it can be incremented to satisfy fast communication rate.
(d) On Arduino side, `Serial.begin(9600)` command should be executed and serial port should be read repeatedly to capture the incoming data
(e) On Pi side, using C++ messages can be send to serial port

Since the lane detection algorithm is implemented on C++, serial communication on the Raspberry side is also implemented on C++. Using <wiringPi.h> and <wiringSerial.h> libraries any string can be sent to the serial port specified by the string /dev/ttyACM0" with a specified baud rate.

Script 1: C++ class for serial communication

```
1  void ArduinoComm::sendToController(std::string payload) {
2  /************************/
3  int serialDeviceId = 0;
4  serialDeviceId = serialOpen("/dev/ttyACM0", 9600);
5  std::cout << "sender " << serialDeviceId << std::endl;
6  if (serialDeviceId == -1) {
7  std::cout << "Unable to open serial device" << std::endl;
8  return;
9  }
10 if (wiringPiSetup() == -1) {
11 return;
12 }
13 serialPuts(serialDeviceId, payload.c_str());
14 return;
15 }
```

On the Arduino side, the commands coming from serial port should be listened. The preferred way of achieving that task is to use `SerialCommand.h` library for the Arduino which allows executing a function depending on the incoming string.

Using .addCommand("str",func) function of the library any function *func* can be associated with any string coming from serial port. Moreover, the functions can have argument. For example, let the string "PWMSET" be execute a function setpwm() which requires the PWM value as argument. If incoming string is of the form "PWMSET 150", using .next() function of the library, the value 150 can be read and converted into integer and interpreted as the PWM value to be set by the function.

C. Discussions on the Solution

Using that library, consecutive commands with less that 1ms time separation are sent and the reliability of the library is tested. The results are positive. Since the lane detection algorithm is generating angle and position data approximately in every 7ms, the performance of the library is more than enough for this purpose.

2.3.2 External Communication Subsystem

A. Requirements for the Solution

1) The subsystem must be able to communicate with the opponent via P2P Wi-Fi protocol
2) The subsystem must start race with a handshake mechanism
3) Similarly, the subsystem must be able to trigger another handshake mechanism at the end of the race
4) For the second handshaking, the subsystem must be able to get the sensor data from vehicle detection subsystem to send messages

B. Solution for the Subsystem

The main solution for this subsystem is setting the device as a P2P host. Sockets are used to send or receive messages. Since the host can act as a server as well as a client, both socket functions are used to establish communication. The subsystem requires opponent ID from the user and sensor readings from the vehicle detection subsystem. It can initiate and terminate other subsystems.

Firstly, the socket data structures are created in both server and client sides. In the beginning of the race, the peer acting as a server listens to the port 5000, which is specified in Standard Committee. Then, the connect() function is called in the client side, so that the request is sent. At the same time, server side acknowledges the request by means of accept() function. At this moment, client side also sends acknowledgement by default, hence race starts. The peers are connected during the race.

The process for the finishing handshake also utilizes socket functions, but the process is a little bit different. For the win case, vehicle detection subsystem (front sensor) initiates the handshake. A catch message (ID00) is sent to the opponent.

If acknowledgment is taken from the opponent, stop message (ID10) is sent. On the other hand, for the defeat case, a catch message is received by the opponent. Then, vehicle detection subsystem is called. According to the returned value from the back sensor, the acknowledge (ID01) or reject (ID11) are sent. Also, LEDs corresponding to messages are lid for the finishing handshake.

C. Discussions on the Solution

The proposed method is a minor difference from the one presented in the Conceptual Design Report. In the previous report, creating hotspot on the server side is proposed. In that case, server side needs to connect to a router and client connects to the server. However, since it is stated that there should not be a router in the connection, the solution is rejected. Then, despite implementation difficulties, the team decided that a routerless communication standard such as ad hoc network or Wi-Fi direct is more suitable for the external communication subsystem.

2.4 Driving System

This system is responsible for the motion of the vehicle. Two parameters that are the direction and the speed of the vehicle is controlled by this unit accordingly to the information coming from the *Computation System*. And the requirement for this system are as follows;

- The subsystem should control motion subsystem according to output of the computation system

The system has two subsystems namely,

1. **Direction Subsystem** is responsible for the orientation of the vehicle and keeps the road and the vehicle aligned.
2. **Speed Subsystem** is responsible for the overall speed of the vehicle by adjusting it considering other effects on the vehicle.

2.4.1 Direction Subsystem

A. Requirements for the Solution

1) The subsystem should drive the motors according to computation system outputs
2) The system should ensure that the vehicle follows the lane

B. Solution for the Subsystem

As will be explained in more detail in *Structure System*, the vehicle has two DC motors and one caster-ball as a movement part. This subsystem uses differential drive in order to drive the vehicle. This subsystem will get two important parameter from other subsystems, namely;

- PWM Offset Value that determines the speed of the vehicle at longitudinal movement. This data is acquired from the **Speed Subsystem.**
- PWM Difference Value that determines the speed difference between the two motors. This difference helps the vehicle in lateral movement. This data is acquired from the **PID Controller Subsystem.**

H-bridge motor drivers are used to drive DC motors. L298N motor driver with voltage regulator is used for this purpose in this project.

C. Discussions on the Solution

The solution for this subsystem is a very similar solution as discussed in *Conceptual Design Report*. Since we performed the test for this subsystem even before the conceptual design stage and they were satisfying technical requirements, the solution was not altered.

2.4.2 Speed Subsystem

A. Requirements for the Solution

1) The subsystem should decrease the vehicle speed at the narrow lane
2) The subsystem should increase the vehicle speed at the wide lane
3) The subsystem should decrease the vehicle speed at the extreme disturbance

B. Solution for the Subsystem

This subsystem is responsible for determining the base speed of the both DC motors. In order to do so, this subsystem produces a *PWM Offsett* value that is sent to the *Direction Subsystem*. To produce this PWM value, the lane angle value β that was introduced in *Section 2.2.2*.

The algorithm relies on the inverse ratio principle, that is, as the lane angle increases the base speed for the motors is decreased. This can be formalized as follows;

$$PWM\ Offset\ =\ PWM\ Base\ - K_1\beta$$

Where the *PWM Base* value is the maximum PWM value for the motors as the lane angle β is equal to zero. And the coefficient K_1 is the constant that determines how fast the base speed is increased as the angle β increases.

C. Discussions on the Solution

The main solution proposed in the conceptual design for the longitudinal speed control were not fully realizable for the project, thus the algorithm is improved by the usage of the road angle β. This improvement allowed us more robust results in our tests.

2.5 Motion System

Duty of this system is maintaining mechanical rigidity of the driving system. And the requirement for this system are as follows;

- The system should ensure that the vehicle can drive itself with enough power.

The system has two subsystems namely,

1. **Wheels Subsystem** is responsible for transferring power from motor shaft to road.
2. **Motors Subsystem** is responsible for converting electrical power to mechanical power

2.5.1 Wheels Subsystem

A. Requirements for the Solution

1) The subsystem should ensure that the wheels can grip lane without slipping in all conditions

B. Solution for the Subsystem

As the previous suggestion in CDR, 2+1 combination (2 wheel with power and 1 caster ball) is preferred due to easier implementation and control. Although this placement weaker in balance and obstacle handling, importance of easier implementation and control are considered more beneficial.

While choosing wheels, high friction property is considered. Because of this reason, super soft and slick tire are chosen with lighten aluminum rim. Besides, larger width is preferred to increase hanging on the lane.

C. Discussions on the Solution

After wheel subsystem tests, we observe the choice gives what we expect. Although tires make dirty the path, their handling capability is fascinating. Therefore, this system satisfies requirements.

2.5.2 Motors Subsystem

A. Requirements for the Solution

1) The subsystem should ensure that the motors can supply enough torque to accelerate the vehicle
2) The subsystem should ensure that the motors can execute driving system outputs without deviation

B. Solution for the Subsystem

As the previous suggestion in CDR, DC motor selection did not change. The reason of this brushed gearhead DC motors are designed to this usage. Even though 3kg-cm is proposed, because the size and weight of the motors in this specs are not appropriate under 600 RPM condition, and eliminate the over engineering, this calculation turns into weight = torque at the shaft of the motor. RPM condition is set in CDR with equation (1). According to this equation 95.5 RPM is the minimum condition, but to be a strong competitor, 5 times of this value is idealized to goal speed. To handle with this value 100 RPM margin is set, to health of the motors during competition.

C. Discussions on the Solution

After motors subsystem tests, motors can move symmetrical without PWM offset, and vehicle can move fast enough with the motors. Also, they perform well in differential drive operation. Therefore, this system satisfies requirements.

2.6 Structure System

This system is responsible for mechanical structure of the vehicle. Placement and orientations of both electrical and mechanical components are considered in this system. And the requirements for this system are as follows;

- The system should ensure that structure is robust for external effects
- The system should ensure that structure is balanced
- The system should ensure that vehicle has a good appearance

The system has two subsystems namely,

1. **Chassis Subsystem** is responsible for the connections of mechanical components in the vehicle.
2. **Printed Circuit Board Subsystem** is responsible for the placement of electrical components.

2.6.1 Chassis Subsystem

A. Requirements for the Solution

1) The subsystem should ensure that the chassis is rigid
2) The subsystem should ensure that the chassis have enough space for components
3) The subsystem should ensure that the chassis can provide low center of mass
4) Camera holder should be integrated to the front of the vehicle

5) Camera holder should be as rigid as possible to reduce the vibration on the camera

6) Camera holder should be light weight so that does not effect the center of mass considerably

7) Camera holder should be adjustable in terms both elevation and camera angle

B. Solution for the Subsystem

Main purposes of this subsystem are protection of the critical elements of the robot and holding components together. The most important part of this section is weight distribution.

Current chassis structure relies on two newly-designed plexiglass layers. Raspberry Pi and Arduino is placed on the upper layer while motor driver and the battery are on lower one. To keep the center of mass of the vehicle close to the ground, battery is placed as low as possible. The connection of the motor driver and Arduino consists of eight cables two of which are the power lines. The cables are placed in a way that they cause no entanglement with any other parts. The connection between RPi and Arduino is currently accomplished by USB cable.

Since there is not much component on the vehicle, the space on the layers are enough to locate the components. However, placing the camera of RPi has been a great problem. The view angle of the camera turned out to be considerable small than expected. Other several cellphone cameras were tried but they are could not satisfy the requirement that both side of the lane should be visible either. The only solution was to elevate the camera. That is why a camera holder structure is designed and added to the system.

To satisfy the requirements the holder is built using 4mm plexiglass. The choice satisfies the rigidity and light weight possible. A thinner one would result in less rigidity and increased vibration on the system. The designed structure has the elevation range from 35 cm to 45 cm and a camera angle ranging from 0^o to 45^o. Having manufactured, the camera holder is integrated to the vehicle (*see Figure 15*). After integration, the view of the camera can completely cover the both edges of the path.

In addition, new motor holder part is designed to have better connection of the wheels to the vehicle. 3D isometric view of the piece can be seen in *Figure* .

Current version of chassis can be seen in *Figure 15*.

C. Discussions on the Solution

The only change done on this subsystem is to design a new plexiglass layers instead of using pre-designed ones. By that way, much more flexibility is obtained in terms

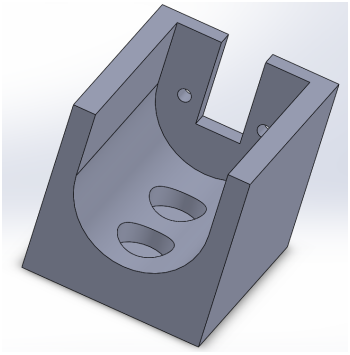


Figure 14: Motor holder 3D view

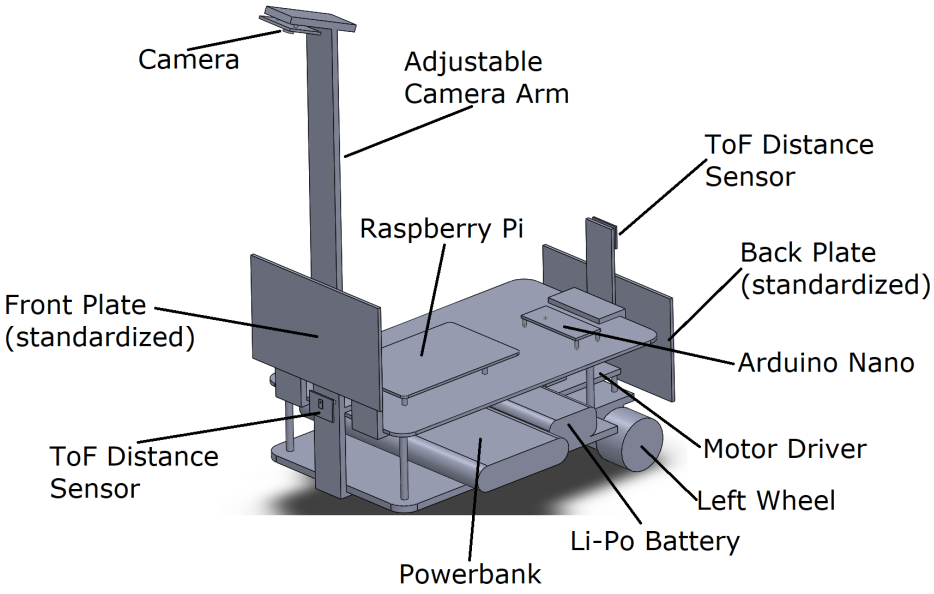


Figure 15: Isometric view of the 3D Drawing of the Vehicle

of placement of the components. On the other hand designed motor holder pieces provides rigid connection between chassis and the wheels.

2.6.2 Printed Circuit Board Subsystem

A. Requirements for the Solution

1) The subsystem should ensure that all the electronic components are placed on PCB
2) The subsystem should ensure that all the connections are firmly secured and robust to vibrations.

B. Solution for the Subsystem

The main role of this part is decreasing connection mess and increase vibration strength of the robot against disturbances. Also, this section increases rigidity of the whole system. The requirements of this subsystem are listed below:

This subsystem aims to make all the circuit connections rigid and compact. Currently, there is wire connections between Arduino-Motor driver and Arduino-RPi. However, addition of vehicle detection sensors and other lane detection alternatives will increase the amount of components, hence, wires. In addition, to use the space occupied by the Arduino UNO board, Arduino Mini can be used. This also allows to build the circuit board as shield for Arduino Mini. After that any other sensors and connections can be made through PCB. In other words, PCB acts as a breakout board for each item integrated to the system in a more rigid and compact way.

2.7 Compatibility of the Subsystems

The block diagram showing the interaction of the subsytem block with each other can be seen from the *Figure 16*. As can be seen from the figure also that, there are two main path within the project. One of them starts by processing the camera vision by the *Lane Detection Subsystem* and ends with the transfer of torque from *Motors Subsystem* to *Wheels Subsystem* which results with a movement of the vehicle. In this path the data before the *PID Controller* and *Speed* subsystems are processed within the Raspberry Pi without any problem, at this point the output data are transferred to Arduino by *Internal Communication Subsystem* without any problem. The rest of this path is processed by the Arduino without any problem as well.

The second path starts with the detection of the opponent by the *Vehicle Detection Subsystem*. The path continues within the Raspberry Pi until the *vehicle stop signal* is transferred to *Motors Subsystem* by *Internal Communication Subsystem* without any problem.

3 Detailed Tests for the Subsystems

1. Lane Detection Subsystem Tests

   (a) Light Condition Test

      i. Mirror the Raspberry Pi screen into Laptop via VNC
      ii. Execute the lane detection algorithm in Raspberry Pi
      iii. Change the location of the camera and Pi to conduct test
      iv. Observe the results in different locations
      v. If the visible lane sides can be detected without any additional object, the result of the test can be considered as success.

   (b) Visual Disturbance Test

      i. Mirror the Raspberry Pi screen into Laptop via VNC

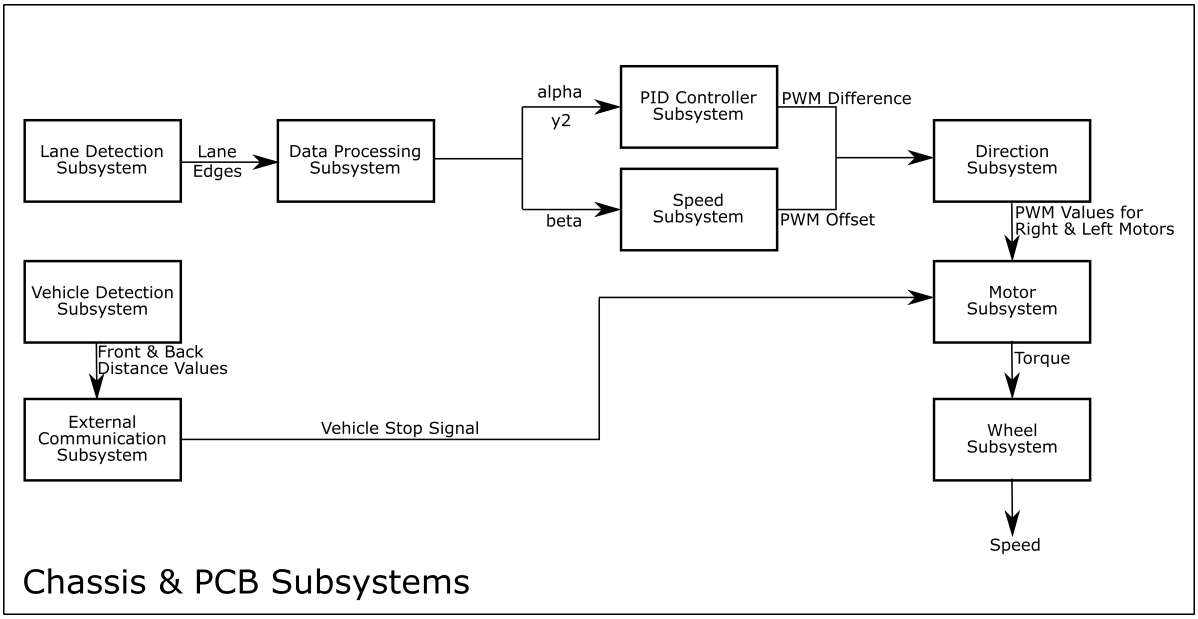


Figure 16: Block Diagram of the Project and the Interaction of the Subsystems

ii. Execute the lane detection algorithm in Raspberry Pi
iii. Put different objects into lane
iv. Observe the results with different disturbances
v. If the objects outside of lane is not detected and the objects inside the road only detected only at its border with road, the result of the test can be considered as success.

2. Vehicle Detection Subsystem Tests

(a) Front Vehicle Detection Test in Closed Environment:

i. Make the connection of the desired sensor and Arduino properly
ii. Hold the sensor at an angle of 90 degree with respect to ground
iii. Place the test object 5 cm in front of the desired
iv. Observe the output of the subsystem
v. Repeat the step 3 & 4 with different distances
vi. If the output of the subsystem generates logical positive for distances smaller than 5 cm and logical zero for distances greater than five, the test result can be considered as success

(b) Rear Vehicle Detection Test in Closed Environment:

i. Repeat the test steps of the *Front Vehicle Detection Test in Closed Environment* with the desired sensor for the desired rear sensor.

(c) Angled Approach Test:

i. Make the connection of the desired sensor and Arduino properly

ii. Hold the sensor at an angle of 90 degree with respect to ground
iii. Place the test object 5 cm in front of the sensor with 30 degree angle with respect to the sensor
iv. Observe the output of the subsystem
v. Repeat the step 3 & 4 with different distance and angle values
vi. If the output of the subsystem generates logical positive for distances smaller than 5 cm for all angle values with respect to sensor and logical zero for distances greater than 5 cm, the test result can be considered as success

(d) Vehicle Detection in Different Sunlight Conditions Test:

i. Repeat the test steps of the *Front Vehicle Detection Test in Closed Environment* in CCC (Cultural and Convention) ground under direct sunlight
ii. Repeat step 1 in CCC (Cultural and Convention) under artificial light, in other words, under no direct sunlight conditions
iii. Repeat steps 1 & 2 for different locations of E Building including Graduation Laboratory
iv. If the output of the subsystem generates logical positive for distances smaller than 5 cm under all light conditions and logical zero for distances greater than 5 cm, the test result can be considered as success

3. Data Processing Subsystem Tests

(a) Data Assessment Test

i. Link the output of Lane Detection subsystem to Data Processing subsystem.
ii. Asses if the output coincide with physical reality of the path

(b) Output Stability Test

i. Place vehicle on a fixed point on the path
ii. Observe outputs for a time interval to see if the subsystem provides stable output for the PID Controller Subsystem

4. PID Controller Subsystem Tests

(a) PID Parameters Test for Given Input:

i. Connect the Vehicle Motors to Motor Controller
ii. Connect the Motor Driver to Arduino
iii. Give the angle value that the subsystem should compensate
iv. Give the power to the motors
v. Observe the behaviour of the vehicle
vi. If the vehicle rotates with an angle given in step 3 without any feedback given, the result of the test can be considered as success.

(b) Bump Test for Distance Control:

i. Set-up a lane as in *Figure 17*.

ii. Make the necessary connection between motors Arduino and data processing unit

iii. Drive the vehicle with PID parameters to be tested.

iv. Collect the distance error between the center of the lane and current position of the vehicle.

v. Plot the time vs distance graph at Matlab using the collected distance errors.

vi. Calculate necessary performance parameters from the plot.

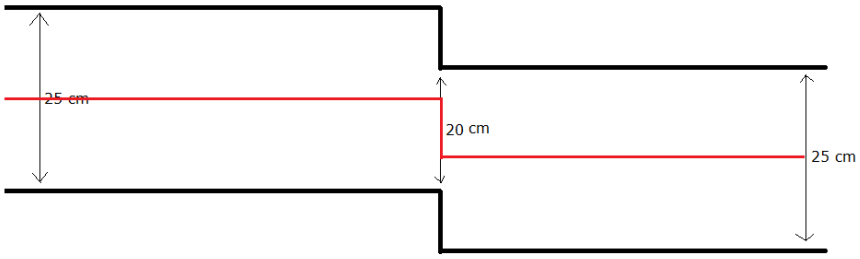


Figure 17: Bump Test for Distance Control

(c) Bump Test for Angle Control:

i. Set-up a lane as in *Figure 18*.

ii. Follow similar steps with *Bump Test for Distance Control*, this time, however, collect the error angle information and plot accordingly.

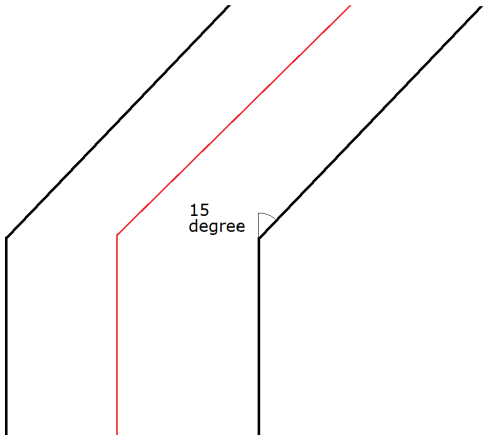


Figure 18: Bump Test for Angle Control

(d) Path Tracking Test:

i. Make the necessary connection between motors Arduino and data processing unit
ii. Place the vehicle to the desired empty path
iii. Observe the behaviour of the vehicle
iv. If the vehicle can follow the path smoothly, the result of the test can be considered as success.

(e) Tracking a Path with Obstacles Test:

i. Make the necessary connection between motors Arduino and data processing unit
ii. Place the vehicle to the desired path with obstacles
iii. Observe the behaviour of the vehicle
iv. If the vehicle can follow the path and compensate the steady state errors due to obstacles without showing oscillatory behaviour and in a reasonable time (in less than 2 seconds), the result of the test can be considered as success.

(f) Path Tracking Test with Physical Disturbances:

i. Make the necessary connection between motors Arduino and data processing unit
ii. Place the vehicle to the desired empty path
iii. Observe the behaviour of the vehicle
iv. If the vehicle can follow the path and compensate the steady state errors due to physical disturbance without showing oscillatory behaviour and in a reasonable time (in less than 2 seconds), the result of the test can be considered as success.

5. Internal Communication Subsystem Tests

(a) Data Retrieval Test

i. Generate data on Raspberry Pi in a rate that reflects the time consumed of Data Processing subsystem. This will yield a realistic data rate.
ii. Send random text data to Arduino.
iii. Do the initial integration between Arduino and Raspberry Pi.
iv. Send data from Raspberry Pi to Arduino.
v. Increase data speed to the specified data rate.
vi. Check the accuracy of the retrieved data.

6. External Communication Subsystem Tests

As mentioned in section 2, the main solution for this subsystem is making device a P2P host and communicating with sockets. However, since the Raspberry Pi is not modified as a P2P host yet, the tests are done using hotspot, as previously mentioned in conceptual design report. Note that, unlike P2P, routers are required

in this test. However, since sockets are functioning same way in P2P, the test is performed to confirm the approach.

(a) Raspberry Pi as Client Test:

i. Create a hotspot from the computer
ii. Connect the Raspberry Pi to the hotspot
iii. Modify the client code to be tested according to IP address of the computer
iv. Run the server code from computer
v. Run the client code from the Raspberry Pi
vi. Try the possible combinations from the terminals of both sides
vii. The test result can be considered as success if both sides respond according to the *Handshake Protocol*.

(b) Raspberry Pi as Server Test:

i. Create a hotspot from Raspberry Pi.
ii. Connect the computer to the hotspot
iii. Modify the client code to be tested according to IP address of the Raspberry Pi.
iv. Run the server code from Raspberry Pi.
v. Run the client code from the computer.
vi. Try the possible combinations from the terminals of both sides
vii. The test result can be considered as success if both sides respond according to the *Handshake Protocol*.

7. Direction Subsystem Tests

(a) Straight Drive Test:

i. Make the necessary connections between motors, motor controller and the Arduino
ii. Set the PWM values of the motors equal
iii. Observe the behaviour of the motors
iv. Increase the PWM value of the slower motor until a point the vehicle can go in a straight line.
v. Record this PWM difference to use in PID controller subsystem

(b) Circular Drive Test:

i. Make the necessary connections between motors, motor controller and the Arduino
ii. Desired curvature is decided
iii. According to motion of the vehicle PWMs of the motors are set
iv. PID parameters are set according to this test

8. Speed Subsystem Tests

   (a) Determination of Base Speed:

      i. Set β value that is supplied to *Speed Subsystem* equal to zero.
      ii. Give PWM Base as 255 RPM and observe the vehicle on the path
      iii. Decrease the PWM Value by 10 PWM and observe the vehicle.
      iv. Repeat the step 3 until the desired base speed value is observed.
      v. Record this value.

   (b) Determination of Constant K_1:

      i. Use the PWM Base value determined at the *Test 8a*
      ii. Set β value that is supplied to *Speed Subsystem* equal to ten degree.
      iii. Set K_1 value equal to one and observe the vehicle on the path.
      iv. Increase the coefficient K_1 and observe the behaviour of the vehicle on the path.
      v. Repeat step 4 until the desired coefficient K_1 is determined.
      vi. Record this value.

9. Wheels Subsystem Tests

   (a) Handling Test:

      i. Place the vehicle on the path
      ii. Apply a horizontal force
      iii. Observe the behaibour
      iv. If the vehicle is slipping, the test can be considered to be failure. If not, the the test result can be considered as success. In other word, friction between road and wheel should greater than road and ground.

10. Motors Subsystem Tests

   (a) Torque Test:

      i. Fix the motor at horizontal position with respect to ground
      ii. Attach an object of one kilogram
      iii. Contact the seller for more information

11. Chassis Subsystem Tests

   (a) Inertia test:

      i. Prepare a straight path
      ii. Power up the vehicle
      iii. Execute the edge detection and control algorithm
      iv. Give different type of disturbances

v. Observe the deviation from straight line
vi. Repeat the process with different component configurations

12. Printed Circuit Board Subsystem Tests

(a) Short test: Aims to check all the wanted connections are present. The test procedure is as follows:

i. Open multimeter for short circuit test
ii. Find the ends of each routing
iii. Check the continuity using multimeter probes
iv. Check if there is any unwanted short circuit
v. If exist, eliminate

4 Test Results

4.1 Test Results, Encountered Problems and Possible Solutions for Subsystems

Results of Lane Detection Subsystem Tests

The lane detection tests were conducted for the detection algorithm of the camera. A sample test result is shown in *Figure 19*. The tests reveal that the subsystem satisfies its requirements by detecting edges. Note that, not all lines are detected, only the lines that are in the ROI are detected as discussed in *Section 2.1.1*.

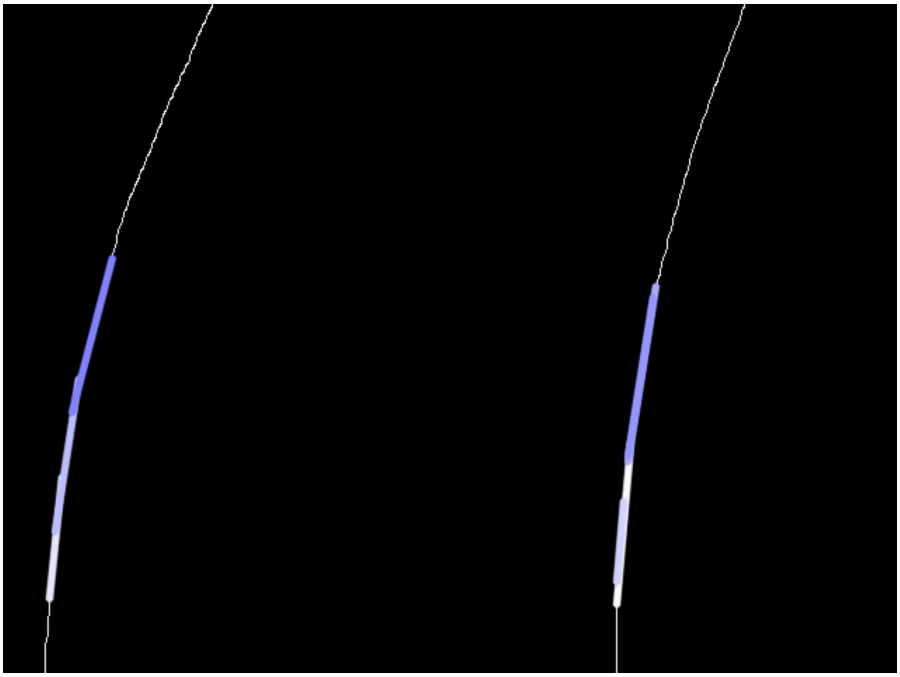


Figure 19: Lane Detection Test Result

Results of Vehicle Detection Subsystem Tests

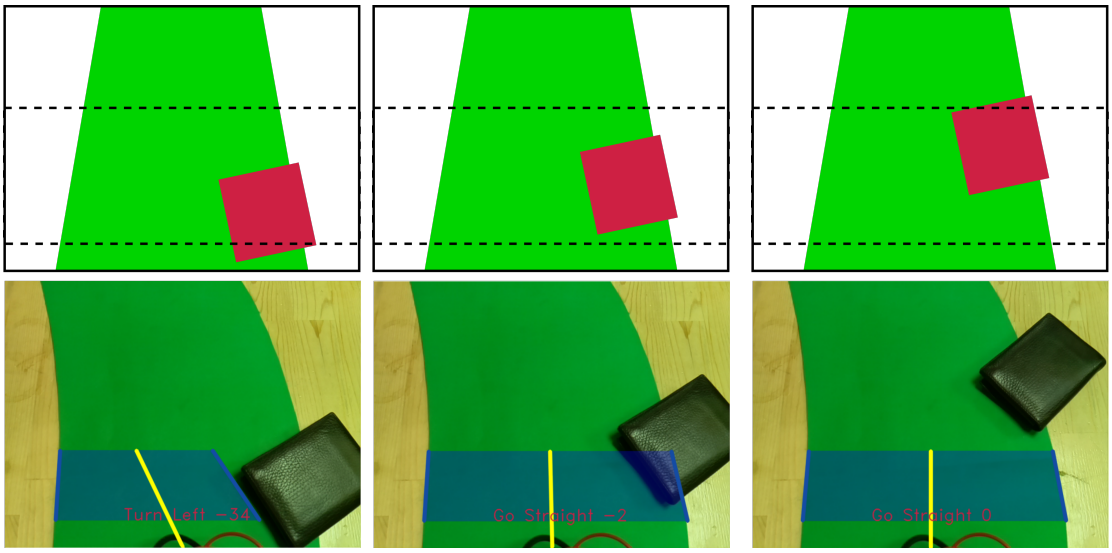
All the test procedures mentioned at the *Section 2* is applied to the VL6180X Time-of-Flight distance sensor. The sensor is showed very accurate result especially in *Angled Approach Test*. According to test, the sensor gives correct results up to 30 degrees. Considering the fact that the path itself is elliptical and there would always be an angle between vehicle even though it may be very small for some cases, it is confirmed that time of flight sensors are quite good choice for this subsystem.

Unlike ultrasonic and infrared sensors, time-of-flight sensors shows very accurate result inside the closed environments like laboratory under artificial lights. The results under direct sunlight especially in CCC is also good as expected. Thus, test results confirms the fact that the main solution should be an enhanced version of infra-red sensors namely the ones utilizing the "time-of-flight" concept.

Results of Data Processing Subsystem Tests

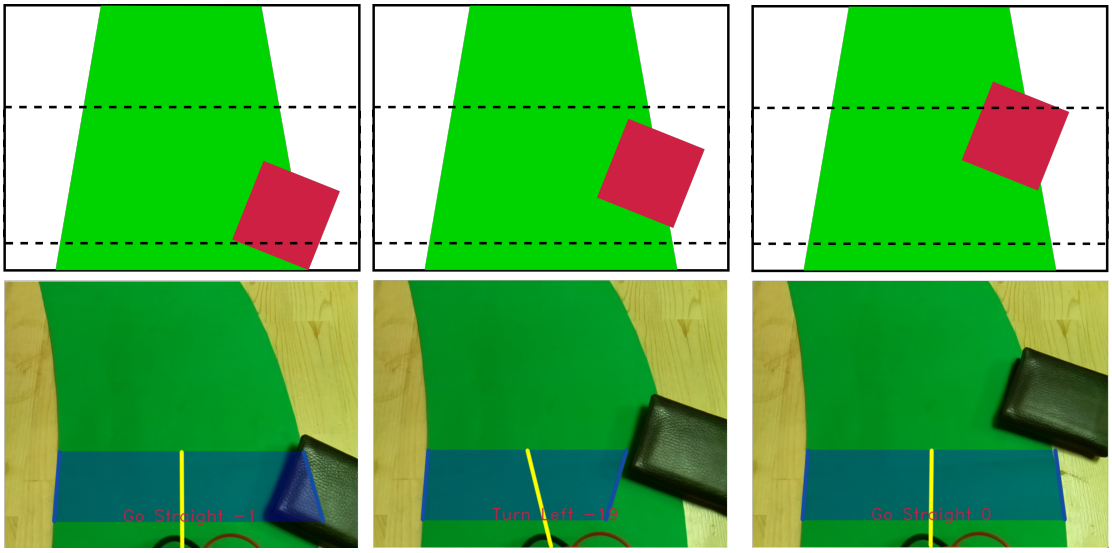
The tests in this section assesses the ability and performance of this subsystem by regarding its requirements. The number of tests realized is quite bit. The first set of tests cover the robustness of the subsystem by placing an obstacle of the subsystem. This test and its results are presented in *Figure 20,21 and 22*. It can be seen that the algorithm ignores the obstacles in 7 cases out of 9 tests. In two cases, the algorithm fails to ignore the obstacles and determines the steering angle as if obstacle constitutes the lane line. Besides the results, on the presence of obstacles, in some particular obstacle placements, the output of the subsystem is observed to be unstable.

The second set of tests cover the robustness of the subsystem as well, but under changing lighting conditions and on different surface materials. The results of this test is presented in *Figure 23 and 24*. The presented results are promising, the steering angles are true. A problem is that these results are a bit unstable when the luminosity difference between the shadows and flighty parts increase. The shadows are sometimes detected as lines and cause untrue lane line evaluations.



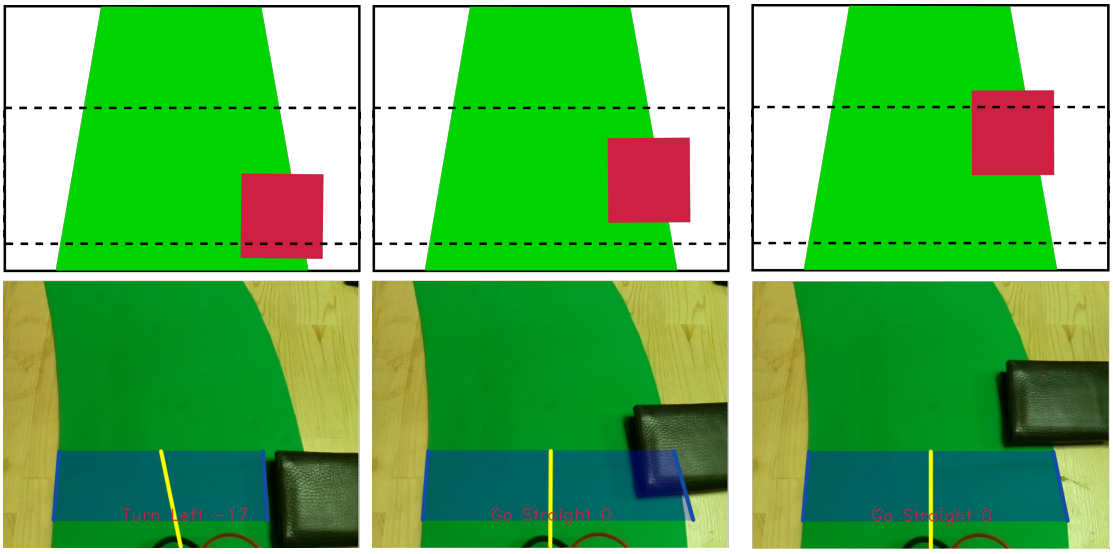
(a) Obstacle is at the Beginning of the Path (b) Obstacle is at the Middle of the Path (c) Obstacle is at the End of the Path.

Figure 20: A Test Scenario: Downward Inclined Obstacle on the Path.
Upper Half: Proposed Tests, Lower Half: Results



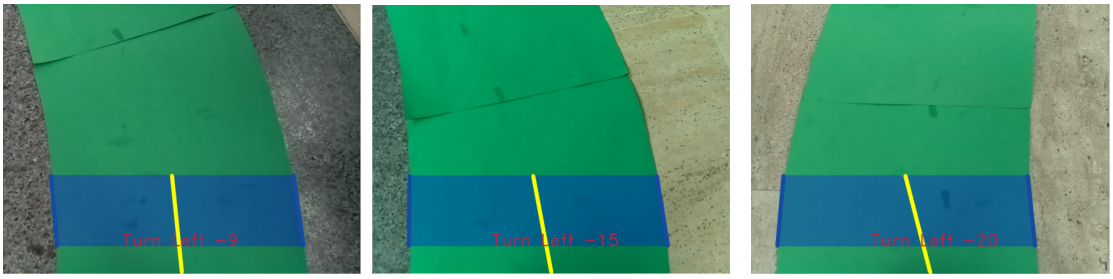
(a) Obstacle is at the Beginning of the Path (b) Obstacle is at the Middle of the Path (c) Obstacle is at the End of the Path.

Figure 21: A Test Scenario: Upward Inclined Obstacle on the Path.
Upper Half: Proposed Tests, Lower Half: Results



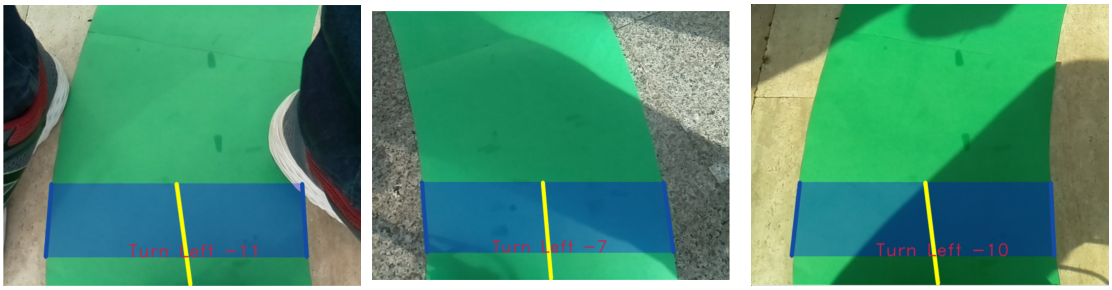
(a) Obstacle is at the Beginning of the Path (b) Obstacle is at the Middle of the Path (c) Obstacle is at the End of the Path.

Figure 22: A Test Scenario: Parallel Placed Obstacle on the Path.
Upper Half: Proposed Tests, Lower Half: Results



(a) Path is Placed on Black Marble (b) Path is Placed on Black and White Marble (c) Path is Placed on White Marble

Figure 23: A Test Scenario Results: KKM Indoor Path Detection



(a) Daylight and Shadow Test-1 (b) Daylight and Shadow Test-2 (c) Daylight and Shadow Test-3

Figure 24: A Test Scenario Results: KKM Outdoor Path Detection

Results of PID Controller Subsystem Tests

Due to some integration error related to internal communication tests, the controller parameters for the lateral movement were not determined sufficiently, thus the *PID Controller Subsystem Tests* introduced in *Section 3* were not completely performed on the vehicle.

Although the results of *Test 4a* were successful for given error values directly to Arduino, the *Tests 4b & 4c* showed unsatisfying results due to some communication errors between *Data Processing Subsystem* and *PID Controller Subsystem*.

Results of Internal Communication Subsystem Tests

The results are positive after all the necessary adjustments are done. The test is also repeated with the real lane detection algorithm and the real time sent data is read from LCD display connected to Arduino. The system was working properly at the rate at which lane detection algorithm produces data.

Results of External Communication Subsystem Tests

Test results can be seen in following figures.*Figure 25* shows DUAYENLER as server while *Figure 26* shows DUAYENLER as client.

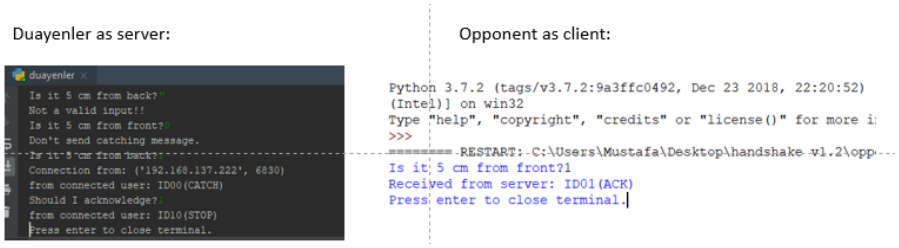


Figure 25: External Communication Subsystem Test Result

Figure 26: External Communication Subsystem Test Result

The first and simplest test has been done on one computer (or raspberry pi) using the same device as client and server, at the same time. To achieve that, the computer's (or raspberry pi) IP address should be defined in the host section defined in the client mode function. Secondly, the codes were tested on two computers. Thirdly, one raspberry pi and one computer were used for the test. All tests were successful if the server side is connected to the internet and client side is connected to the server via hotspot. The outputs of the tests were given in the *Figure 25* and *Figure 26*.

Results of Direction Subsystem Tests

The results of *Tests 7a & 7b* proposed in *Section 3* revealed to the team that, the differential drive method is capable of returning as desired and can keep up with the lane to be followed.

Results of Speed Subsystem Tests

The tests proposed in *Section 3* will be conducted after the tests for *Internal Communication Subsystem* is completed and the tests for *PID Controller Subsystem* is completed with constant base speed.

Results of Motion System Tests

After tests, handling capability of the system is approved. However, although motors are quite well with high PWM, their low PWM performance cause some problems. Therefore, motor subsystem could need a revision.

Results of Structure System Tests

As mentioned in Section 3, the disturbances applied on the vehicle and the deviation is between the levels that can be handled by other algorithms although some improvements are still needed. Furthermore, newly-designed chassis added rigidity to the structure which in turn provides better test results

5 Other Considerations

Besides, a Gannt Chart is prepared to have an detailed overview of future works and available in *Appendix A*.

5.1 Cost Analysis

Estimated cost anaysis for the project can be investigated at *Table 1*. The reproducible vehicle is expected to cost under 200 dollar as desired by the project requirements.

Table 1: Cost Analysis for the Project

Component	Number	Total Price (in Dollar)
Raspberry Pi 3B	1	48
Camera	1	23
Chassis Components	1	15
Arduino Nano	1	4.8
DC Motor	2	22
Wheel	2	8
Motor Driver	1	2.5
Powerbank	1	12
Li-po Battery	1	24
ToF Distance Sensor	2	18
LED headlight/LED	-	0.2
Total Project	-	176.7

As seen from the *Table 1*, budget is optimized with some changes such as Arduino Uno is replaced with its nano version, and upper layer of the chassis is designed thinner plexi glass. However, critical components, such as motors, ToF sensors and wheels, are selected for their performance. Powerbank selection is based on its size. The chosen one is the smallest powerbank which can give enough output to supply Raspberry Pi under full load. Camera and Raspberry pi have no other option in this project. Li-Po battery selection is based on duration and output voltage. 12V output is required during motor drive, and long term is required for demonstrations. Therefore, 1750 mAh 11.1V 3S battery is selected.

5.2 Power Analysis

Table 2: Estimated Cost Analysis for the Project

Component	Current (Avg),A	Power (Avg),W	Current (Max),A	Power (Max),W
Raspberry Pi 3B	0.85	4.25	2.5	12.5
Arduino Nano	80m	0.4	0.2	1
DC Motors & Motor Driver	0.4	4.8	1.1	12.12
Distance Sensor	19m	62.7m	40m	132m
Total	1.52m	9.153	3.84	25.75

The *Table 2* shows the consumption under regular case and extreme case. If extreme scenario is considered, full power consumption of Raspberry Pi is supplied from powerbank while motors are supplied by Li-Po battery as can be seen from the *Figure 27*. Also, sensors and Arduino are supplied from Pi because they do not have high demand. Powerbank has two output could supply 2.5A for 5V output, so Pi could supply in the worst case.

Li-Po battery has these specs: 1750 mAh 11.1V 3S 25C, so it can supply 43 ampere constants during discharge although the motors demand 1.1 ampere at stall condition.

All in all, sources are completely enough for consumption even in the worst case conditions.

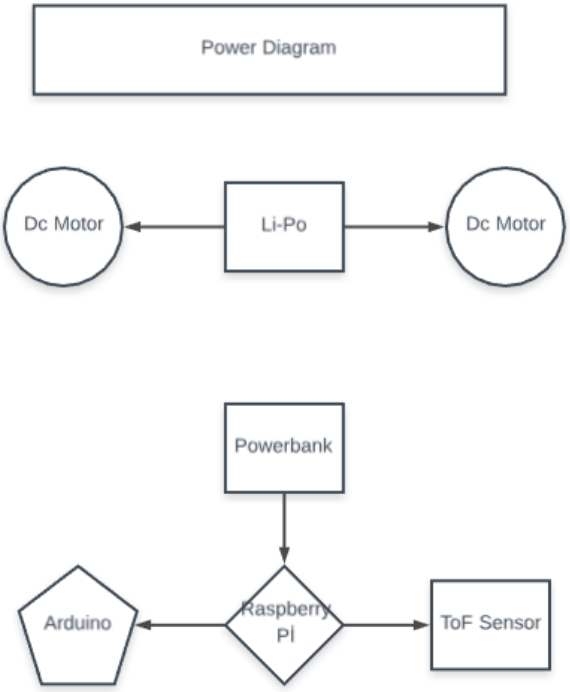


Figure 27: Electrical Architecture of the Project

6 Deliverables

THIS PART IS NEW FOR FINAL REPORT.

7 Budget

7.1 Actual Expenditures

7.2 Total Cost

8 Discussions

THIS PART IS NEW FOR FINAL REPORT.

This sections presents a discussion on several things such as safety issues, application areas and environmental effects of the final product.

8.1 Safety Issues

8.2 Application Areas

8.3 Environmental Effects

THIS PART IS NEW FOR FINAL REPORT.

The environmental effects can be discussed regarding the material used in body of the car. Chassis and camera arm are all made of plexiglass (PMMA) material. PMMA is a versatile, durable, recyclable and sustainable material. As it has such properties, PMMA based products products have replaced products which were previously made from wood, iron and other natural ingredients. With this way, pressure on natural supplies is decreased. Hence, the acrylic products can be considered as an environment friendly option.

Regarding the battery, a Li-Po battery is used on the vehicle. Production of Li-Po batteries require lots of chemical and energy. And, what is worse is that there are currently no good programs to recycle lithium-polymer batteries. So, it can be concluded that Li-Po batteries are not environment friendly.

9 Conclusion

THIS PART IS NEW FOR FINAL REPORT.

This report consists of a review of the conceptual design report solutions and picked solutions. The solutions are explained in detail, and comparison of the solution among alternatives are discussed.

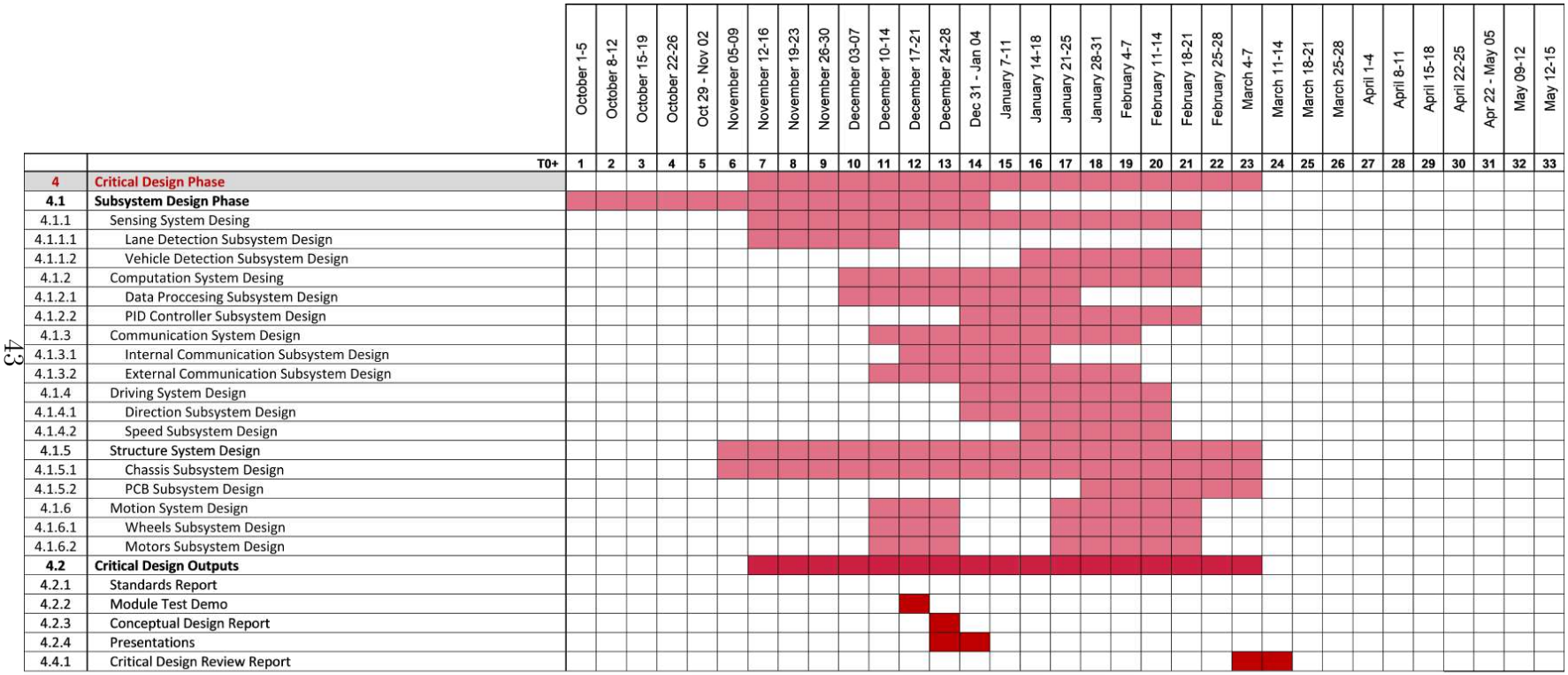
THIS PART IS NEW

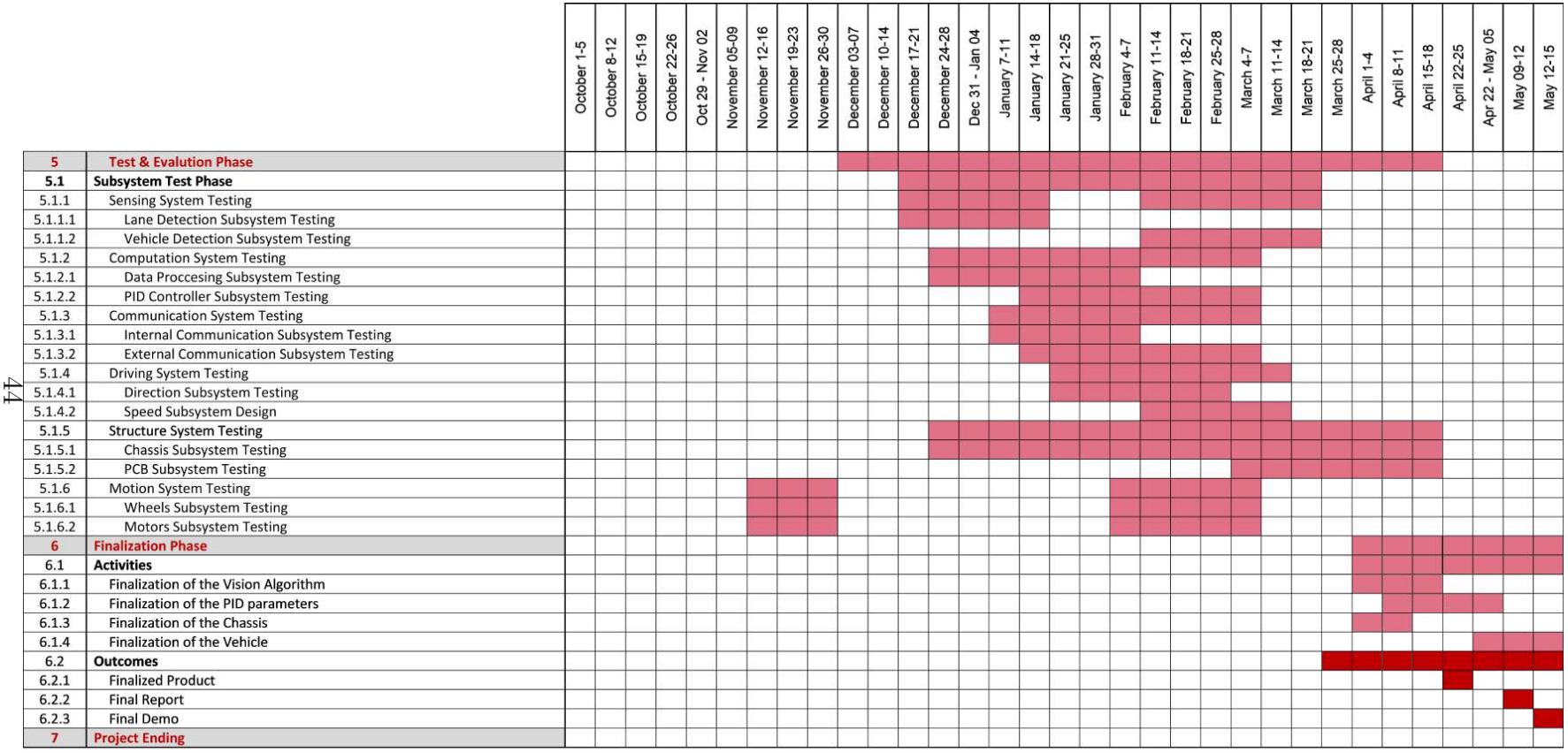
In especial, overview of the systems is shown in V-Model with the system block diagram. Then, subsystems are given in a detailed manner with clarifications of the final solution, comparison of the CDR solutions, test steps and test results. The system has been assembled into a prototype chassis. Thanks to that, tests are done on this chassis. Besides, final chassis is designed considering the feedback gathered by this chassis. Mentioned limitation in CDR are enhanced, as shown in the test result, even though there are still minor issues. In addition to this, other non-problematic subsystems' solutions are frozen. On the other hand, economic constraint which is 200$ is still higher than cost of the total project. The company believes that power of a company comes from its economic plans.

DUAYENLER will continue their hard work to enhance their design performance in every way, for every case and scenario. The company members believe that the final works will bring an innovative approach to the design of autonomous cars with its solutions and solution approach.

A Gantt Chart

		October 1-5	October 8-12	October 15-19	October 22-26	Oct 29 - Nov 02	November 05-09	November 12-16	November 19-23	November 26-30	December 03-07	December 10-14	December 17-21	December 24-28	Dec 31 - Jan 04	January 7-11	January 14-18	January 21-25	January 28-31	February 4-7	February 11-14	February 18-21	February 25-28	March 4-7	March 11-14	March 18-21	March 25-28	April 1-4	April 8-11	April 15-18	April 22-25	Apr 22 - May 05	May 09-12	May 12-15
	T0+	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
	1. Semester																																	
	Holiday																																	
	2. Semester																																	
1	**Concept Development Phase (3 Weeks)**																																	
2	**Assessment of System Requirement Phase (3 Week)**																																	
3	**Conceptual Design and Preliminary Design Phase (4 Weeks)**																																	
4	**Critical Design Phase (17 Weeks)**																																	
5	**Test & Evaluation Phase (20 Weeks)**																																	
6	**Implementation & Finalization Phase (8 Weeks)**																																	
7	**Project Ending (T0+33 Weeks)**																																	
1	**Concept Development Phase**																																	
1.1	**Activities**																																	
1.1.1	Literature Research and Determination and Similar Platform Specifications																																	
1.1.2	Feasibility Works																																	
2	**Assessment of System Requirement Phase**																																	
2.1	**Activities**																																	
2.1.1	Determination of Team Logo and Vision & Mission																																	
2.1.2	Problem Define State for All Projects																																	
2.1.3	Solve Defined Problem State for All Projects																																	
2.1.4	General Component Research																																	
2.2	**Outcomes**																																	
2.2.1	Business Statement Report																																	
3	**Conceptual Design and Preliminary Design Phase**																																	
3.1	**Activities**																																	
3.1.1	Preliminary Electrical System Design																																	
3.1.1.1	Preliminary Sensing Unit Design																																	
3.1.1.2	Preliminary Computational Unit Design																																	
3.1.1.3	Preliminary Driving Unit Design																																	
3.1.2	Preliminary Mechanical System Design																																	
3.1.2.1	Preliminary Motion Unit Design																																	
3.1.2.2	Preliminary Structure Design																																	
3.2	**Outcomes**																																	
3.2.1	Preliminary Report																																	


October 1-5
October 8-12
October 15-19
October 22-26
Oct 29 - Nov 02
November 05-09
November 12-16
November 19-23
November 26-30
December 03-07
December 10-14
December 17-21
December 24-28
Dec 31 - Jan 04
January 7-11
January 14-18
January 21-25
January 28-31
February 4-7
February 11-14
February 18-21
February 25-28
March 4-7
March 11-14
March 18-21
March 25-28
April 1-4
April 8-11
April 15-18
April 22-25
Apr 22 - May 05
May 09-12
May 12-15
T0+ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
4 Critical Design Phase
4.1 Subsystem Design Phase
4.1.1 Sensing System Desing
4.1.1.1 Lane Detection Subsystem Design
4.1.1.2 Vehicle Detection Subsystem Design
4.1.2 Computation System Desing
4.1.2.1 Data Proccesing Subsystem Design
4.1.2.2 PID Controller Subsystem Design
4.1.3 Communication System Design
4.1.3.1 Internal Communication Subsystem Design
4.1.3.2 External Communication Subsystem Design
4.1.4 Driving System Design
4.1.4.1 Direction Subsystem Design
4.1.4.2 Speed Subsystem Design
4.1.5 Structure System Design
4.1.5.1 Chassis Subsystem Design
4.1.5.2 PCB Subsystem Design
4.1.6 Motion System Design
4.1.6.1 Wheels Subsystem Design
4.1.6.2 Motors Subsystem Design
4.2 Critical Design Outputs
4.2.1 Standards Report
4.2.2 Module Test Demo
4.2.3 Conceptual Design Report
4.2.4 Presentations
4.4.1 Critical Design Review Report


October 1-5
October 8-12
October 15-19
October 22-26
Oct 29 - Nov 02
November 05-09
November 12-16
November 19-23
November 26-30
December 03-07
December 10-14
December 17-21
December 24-28
Dec 31 - Jan 04
January 7-11
January 14-18
January 21-25
January 28-31
February 4-7
February 11-14
February 18-21
February 25-28
March 4-7
March 11-14
March 18-21
March 25-28
April 1-4
April 8-11
April 15-18
April 22-25
Apr 22 - May 05
May 09-12
May 12-15
5 Test & Evalution Phase
5.1 Subsystem Test Phase
5.1.1 Sensing System Testing
5.1.1.1 Lane Detection Subsystem Testing
5.1.1.2 Vehicle Detection Subsystem Testing
5.1.2 Computation System Testing
5.1.2.1 Data Proccesing Subsystem Testing
5.1.2.2 PID Controller Subsystem Testing
5.1.3 Communication System Testing
5.1.3.1 Internal Communication Subsystem Testing
5.1.3.2 External Communication Subsystem Testing
5.1.4 Driving System Testing
5.1.4.1 Direction Subsystem Testing
5.1.4.2 Speed Subsystem Design
5.1.5 Structure System Testing
5.1.5.1 Chassis Subsystem Testing
5.1.5.2 PCB Subsystem Testing
5.1.6 Motion System Testing
5.1.6.1 Wheels Subsystem Testing
5.1.6.2 Motors Subsystem Testing
6 Finalization Phase
6.1 Activities
6.1.1 Finalization of the Vision Algorithm
6.1.2 Finalization of the PID parameters
6.1.3 Finalization of the Chassis
6.1.4 Finalization of the Vehicle
6.2 Outcomes
6.2.1 Finalized Product
6.2.2 Final Report
6.2.3 Final Demo
7 Project Ending