Middle East Technical University

Department of
Electrical and Electronics Engineering

EE493 ENGINEERING DESIGN I

Car Chasing Robot
Conceptual Design 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 Executive Summary

The developments in microlectronic industry and computer architecture brought the wind at their back to many industries, as automotive industry being one of them. With such advances in technology, automobile industry is trying to come up with different new technologies. Most of them being related the customer comfort, one technology differentiates from the rest, that is autonomous driving. Autonomous driving wil not just provide comfort the people inside the car but it also will change the way traffic works nowadays. The vehicles will create a new network of traffic that will have no human judge on the flow. Autonomous driving will enable many new concepts, however, there are a lot of sub-technologies that must be used in coordination to come up with a clear cut autonomous driving. Some of them are vehicle vision, inter-vehicle communication, vehicle-human interaction, vehicle safety and so on. To address the needs in development of aforementioned technologies, DUAYENLER Ltd. Şti. (DUAYENLER) is founded. DUAYENLER aims to be one of the pioneers of the industry with its novel approaches.

The company consists of talented engineers from different fields, namely computer, electronics and control. DUAYENLER employs an inter-disciplinary work on research and development phase. Indeed, this allows DUAYENLER to develop complex but simple looking technologies. DUAYENLER is able to accomplish and complete the tasks with all its energy and willpower.

DUAYENLER has several focus points to develop an autonomous driving technology. These are, vehicle vision, inter-vehicle communication, vehicle control and physical design of the vehicle. Actually, those are the main building blocks of the main technology. Vehicle vision summarizes the concept of understanding path properties, that is to be able to extract and differentiate path and obstacles in every weather condition. Obviously, inter-vehicle communication provides cooperation environment with opponent vehicles and enables a traffic hierarchy. As the vehicle has path vision, the decision of steering must be made that brings the need for vehicle control. Being able to follow a path and to head towards bends are essential for a vehicle. Besides, the physical structure of the vehicle must be able to suit with the rest of the technologies. The height, the weight, the width of the vehicle, the chassis material choice and additional implementation-specific features are handled in physical design. This structure and combination of the systems provide suitable environment for DUAYENLER to develop autonomous driving technology.

This report gives the reader a solid view and understanding of the project. Detailed design considerations with technical details and test results do not leave any vague point on the implementation. The duration of the project is 33 weeks, from the beginning of October 2018 to the second week of May 2019. The total cost is calculated to be approximately 616.5 $ and 1700 man-hour, whereas cost to produce the commercial vehicle is 199.5 $. Along with the vehicle, the customer is provided with user manual, elliptical path, rechargeable batter and the charger. The vehicle has two (2) years of warranty.

2 Introduction

Autonomous car technologies are the hot topic of today. Companies which want to lead the technology have already started on this topic with their R&D teams. To bring new approach in this area, DUAYENLER Ltd. Şti. (DUAYENLER) aims to be one of the leaders in this area. Based on this goal, DUAYENLER designed and produced a prototype for Car Chasing Project. The purpose of this project is an autonomous car can follow a predefined path, and it can communicate with the other car on the path.

DUAYENLER with its five team members divided this project into pieces, such as image processing, mechanical design, differential drive etc., and assign tasks to members according to their professions. After modules are designed, integration and testing were a collective work of all members.

The main blocks of the project are autonomous steering and its control. These blocks are built from systems that are built from subsystems. Therefore, at first stage, objectives of the project were specified. Then, the project was divided into six systems with their requirements. As a next stage, systems were divided into subsystems. Finally, from bottom to top design was completed.

This report contains detailed descriptions of solutions for subsystem blocks, their requirements, tests and test results of the prototype. Moreover, deliverables, detailed cost analysis of prototype and overall project are covered by this report. Finally, report contains safety issues, environmental effects of the prototype, and possible extensive application of the prototype.

3 Design Description

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*.

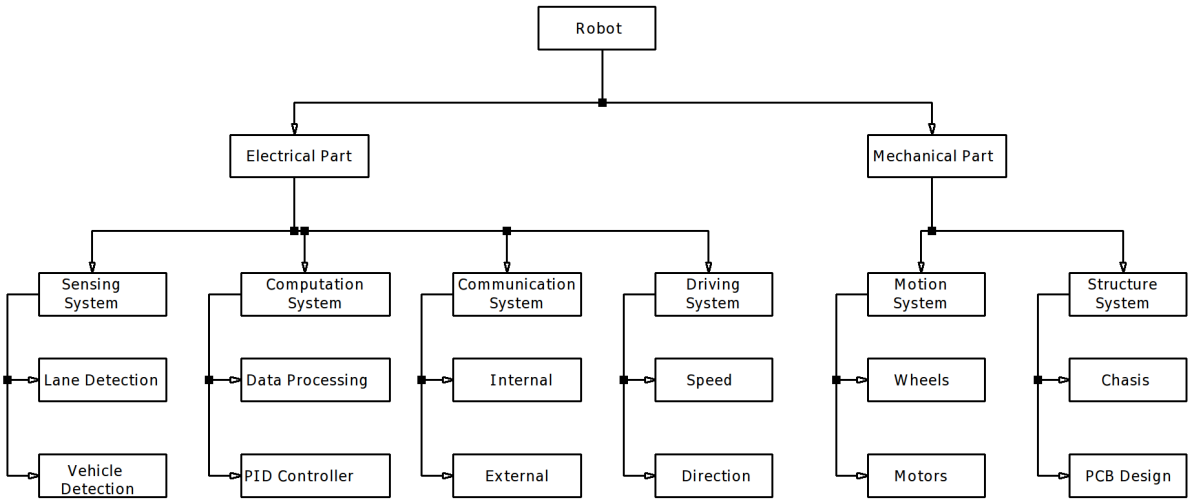


Figure 1: Organization of the Project

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.

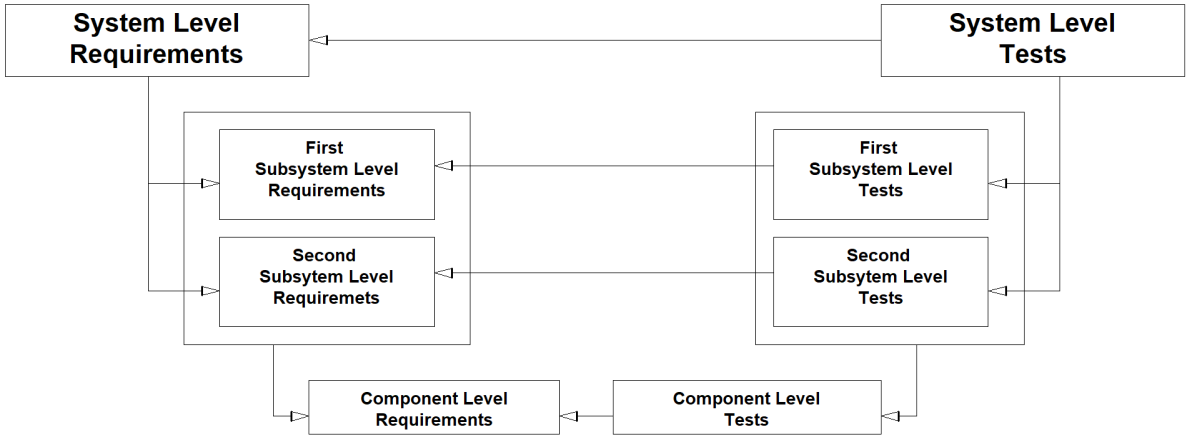


Figure 2: V-Model

3.1 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,

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

3.1.1 Lane Detection Subsystem

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*.

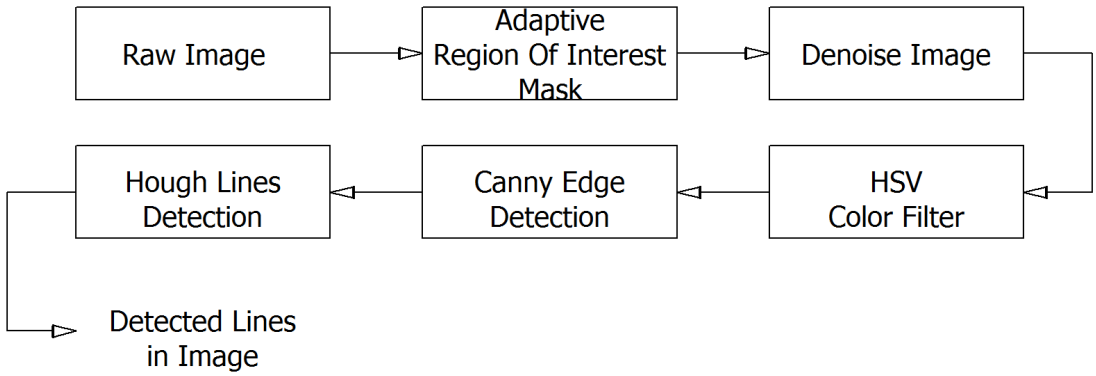


Figure 3: Block Diagram of the Lane Detection Subsystem

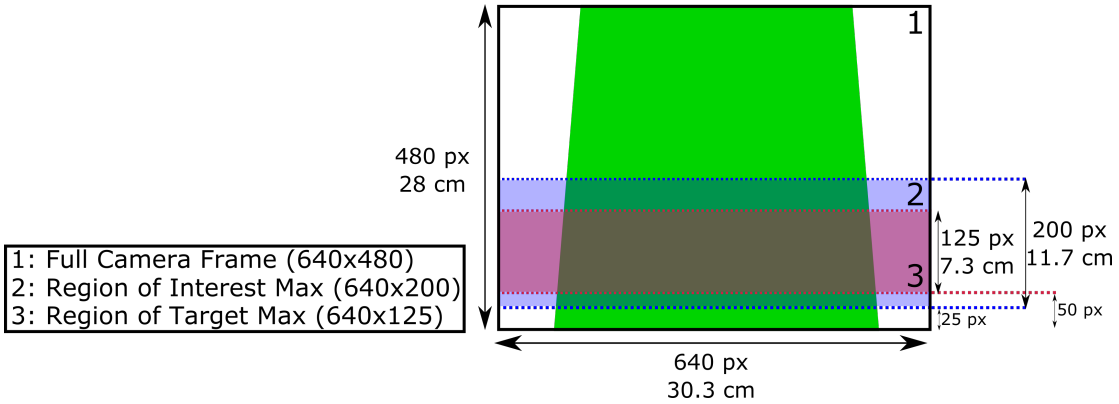


Figure 4: Explanation of Frame Regions

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 $640x200px$. 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 $640x200px$. 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:

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

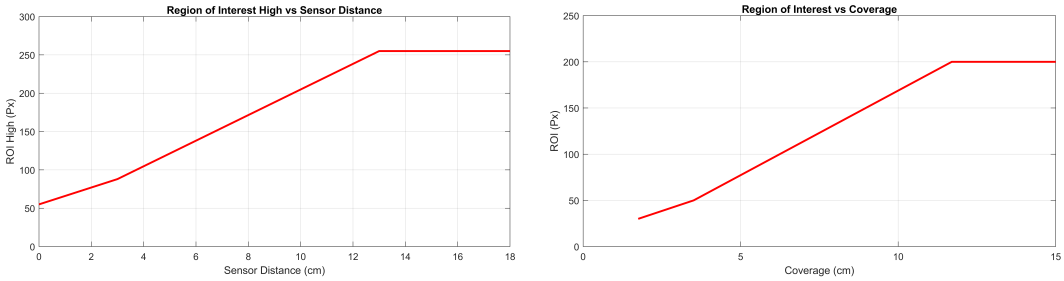
$$ROI\_Height = (ROI\_High - ROI\_Low)px \tag{2}$$

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

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

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) 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.

3.1.2 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 sensors, called vl6180x. Time of flight sensors are enhanced IR sensors. One at the back of the vehicle is responsible for detecting the chasing opponent and one at the front of the vehicle is responsible for detecting the chased opponent. The sensors use I2C protocol. The serial data coming from sensors is between 0 and 255, which corresponds to the reading in terms of millimeters. That is, vl6180x can make measurements up to 25.5 cm. Since sensor reading is performed using Raspberry Pi, the required trigger for handshake protocol can be easily accessed by the external communication subsystem.

   Note that, having 2 sensors means that there are 2 I2C slaves. In I2C, master device sends data to slaves according to their addresses. The main problem emerging from

this is that if two sensors have the same address, reading sensors is not possible. Therefore, the addresses of the sensors must be changed. However, there is also another problem. Changing address is also done by sending serial data to sensors. If both sensors are connected at the same time to Raspberry Pi, changing addresses is not possible because there are still 2 devices with same addresses. To solve this issue, the chip enable(CE) inputs of the sensors is used. Firstly, only one device is activated by applying high to its CE input pin. After changing its address, other sensor's CE pin can be safely made high. By doing that, two I2C devices with different addresses are obtained. From this point on, sensors can be read successfully.

3. Discussions on the Solution

   The proposed method is not changed after Critical Design Review Report. Changing I2C slave addresses is done to successfully use both sensors. Also, the sensor readings are integrated to handshake code.

3.2 Computation System

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.

3.2.1 Data Processing Subsystem

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.

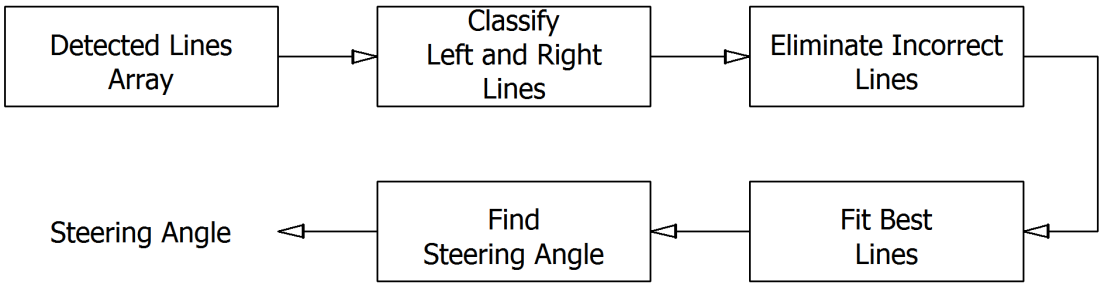


Figure 6: Block Diagram of the Data Processing Subsystem

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 3.2.2. 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*.

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

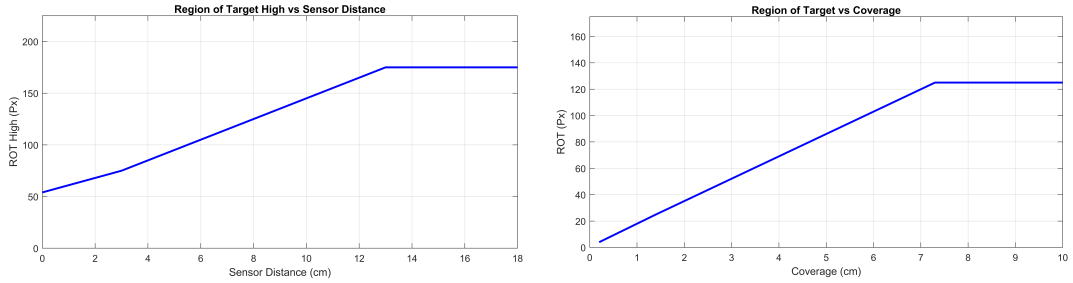
$$ROT\_Height = (ROT\_High - ROT\_Low)px \tag{6}$$

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

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

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 6*.

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



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

Figure 7: Graphs Defining ROT_Height

pixels on a column is counted, that counts yield an information regarding the start and end points of the path. The explanation will be made based on an example frame (with ROI masked) as in *Figure 8*. 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.

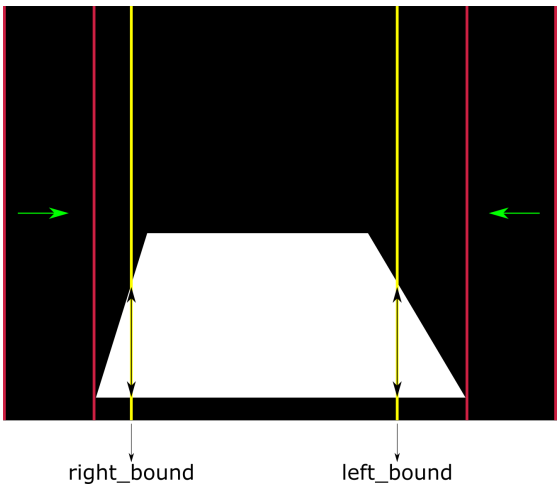
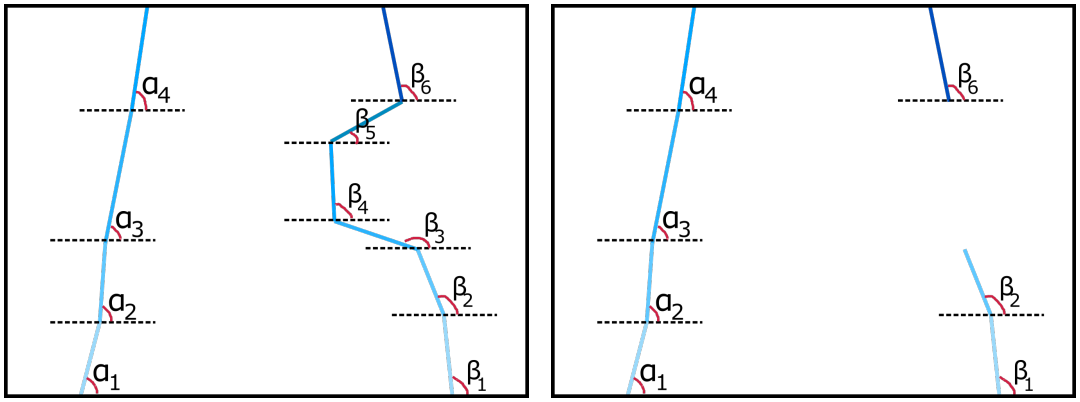


Figure 8: Finding Center of the Path

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

Algorithm 1: Finding Image Center

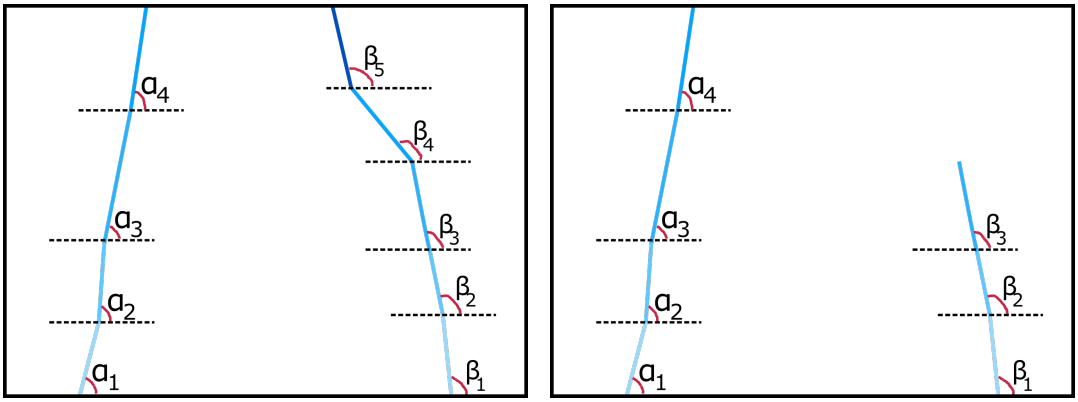
```
whitePixels[640]
confidenceCount = 190//thresholdPixelCount
for Every Row i do
    for Every Column j do
        if frame[i][j] == 255 then
            whitePixels[j]++
for Elements of whitePixels from left to right do
    Find the first index that has white pixel count greater than confidenceCount;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

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, 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



(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
```

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*.

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 generate control outputs $y1$ and $/beta$ (to be detailed in *PID Controller Subsystem*. The parameter $y1$ is found by pixel 320 subtracted by x-coordinate of the right line. The parameter β is the difference between the slope angle of the right line and the left line. The calculated output parameters are fed into PID Controller as Subsystem as inputs.

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.

3.2.2 PID Controller Subsystem

1. Requirements for the Solution

   (a) The subsystem should be able to control the motors

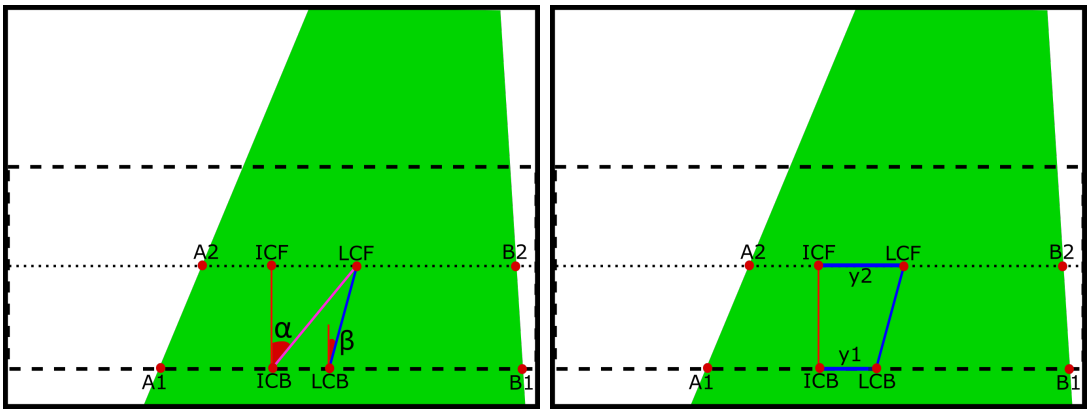
   (b) The subsystem should be able to react the external disturbances

2. Solution for the Subsystem

   PID Controller Subsection is the main subsystem of the vehicle whose reponsibility is keeping the vehicle at the middle of the path it is following. To achieve this task, this subsystem includes a PID controller for 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.

In our application, we have decided to use distance variable y_1 and angle variable for control purposes. The main pupose of the PID Subsystem is to compansate the lateral distance error of the vehicle, i.e., staying exactly on the center lane. In our control design, we have descided to use y_1 as control signal and, zero as reference signal. Basic block diagram can be seen at *Figure 30*. The output of this PID controller then send to the DC motors as PWM difference. The actual base PWM is produced by the Speed Subsystem.

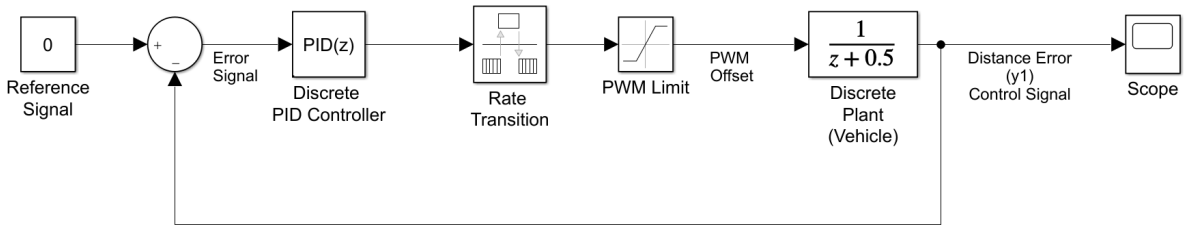


Figure 12: Block Diagram of the Project and the Interaction of the Subsystems

Pulse Transfer Function of a Digital PID Controller

However, since the controller operates on the microcontroller, therby, on discrete-time domain. We firstly find its pulse transfer function and transferred it to

discrete-time domain. General PID controller can be expressed in *Laplace* domain as

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

Discritization of I Controller

Casual discrete time approximation of the I-Controller can be written as;

$$y[k] = y[k-1] + \frac{u[k] + u[k-1]}{2} * T_s$$

Where $y[k]$ is output and $u[k]$ is the output. Taking its Z-transform, the pulse transfer function becomes,

$$Y(z) = Y(z)z^{-1} = \frac{T_s}{2}\left(U(z) + U(z) * z^{-1}\right)$$

$$G_I(z) = \frac{Y(z)}{U(z)} = \frac{T_s(1 + Z^{-1})}{2(1 + Z^{-1})}$$

Discritization of D Controller

Casual discrete time approximation of the D-Controller can be written as;

$$y[k] = \frac{e[k] - e[k-1]}{T_s}$$

Where $y[k]$ is output and $e[k]$ is the error input. Taking its Z-transform, the pulse transfer function becomes,

$$Y(z) = Y(z)z^{-1} = \frac{T_s}{2}\left(E(z) + E(z)z^{-1}\right)$$

$$G_D(z) = \frac{Y(z)}{E(z)} = \frac{(1 - Z^{-1})}{T_s}$$

Therefore pulse transfer function of the PID controller becomes,

$$G_{PID}(z) = K_p + \frac{T_s(1 + Z^{-1})}{2(1 + Z^{-1})} + \frac{(1 - Z^{-1})}{T_s}$$

$$\boxed{G_{PID}(z) = \frac{K_p(1 - z^1) + K_i\frac{T_s(1+z^{-1})}{2} + K_d(1 - 2z^{-1} + z^{-2})}{1 - z^{-1}}}$$

Implementation of the PID Controller

$$\frac{Y(z)}{E(z)} = \frac{K_p(1-z^1) + K_i\frac{T_s(1+z^{-1})}{2} + \frac{K_d}{T_s}(1-2z^{-1}+z^{-2})}{1-z^{-1}}$$

$$Y(z) - z^{-1}Y(z) = K_pE(z) - K_pz^{-1}E(z) + \frac{K_iT_s}{2}E(z) + \frac{K_iT_s}{2}E(Z)z^{-1}..$$
$$+\frac{K_d}{T_s}E(z) - 2z^{-1}E(z) + \frac{K_d}{T_s}z^{-2}E(z)$$

$$y[k]-y[k-1] = K_pe[k]-K_pe[k-1]+\frac{K_iT_s}{2}+K_iT_se[k-1]/2+\frac{K_d}{T_s}e[k]-2e[k-1]\frac{K_d}{T_s}+\frac{K_d}{T_s}e[k-2]$$

$$y[k]-y[k-1] = (K_p+\frac{K_iT_s}{2}+K_d)e[k]+(-K_p+\frac{K_iT_s}{2}-2\frac{K_d}{T_s})e[k-1]+(\frac{K_d}{T_s})e[k-2]$$

If we add all past terms together to find $y[k]$,

$$y[k] - y[k-1] = (K_p + \frac{K_iT_s}{2} + \frac{K_d}{T_s})e[k] + (-K_p + \frac{K_iT_s}{2} - 2\frac{K_d}{T_s})e[k-1]..$$
$$+(K_d)e[k-2]$$
$$y[k-1] - y[k-2] = (K_p + \frac{K_iT_s}{2} + \frac{K_d}{T_s})e[k-1] + (-K_p + \frac{K_iT_s}{2} - 2\frac{K_d}{T_s})e[k-2]..$$
$$+(K_d)e[k-3]$$
$$y[k-2] - y[k-1] = (K_p + \frac{K_iT_s}{2} + \frac{K_d}{T_s})e[k-2] + (-K_p + \frac{K_iT_s}{2} - 2\frac{K_d}{T_s})e[k-3]..$$
$$+(K_d)e[k-4]$$
$$...$$
$$...$$
$$y[k-n] - y[k-n-1] = (K_p + \frac{K_iT_s}{2} + \frac{K_d}{T_s})e[k-n] + (-K_p + \frac{K_iT_s}{2} - 2\frac{K_d}{T_s})e[k-n-1]..$$
$$+(K_d)e[k-n-2]$$
$$y[k] - y[k-n-1] = (K_p + \frac{K_iT_s}{2} + \frac{K_d}{T_s})e[k] + (K_iT_s - \frac{K_d}{T_s})e[k-1] + (K_iT_s)e[k-2]..$$
$$+(K_iT_s)e[k-2].... + (K_iT_s)e[k-n-1]$$

Therefore;

$$\boxed{y[k] = y[k-n-1] + K_p(e[k]) + K_d(\frac{e[k]-e[k-1]}{T_s}) + K_i(T_s\sum_{i=0}^{n+1}e[k-i])}$$

This equation was utilized on Arduino microcontroller considering the past three errors for the integral term.

Sampling Time Matching and Controller Output Limit

In our implementation, the sampling time of the plant, i.e., the processing time in which the Raspberry Pi processes each picture frame is approximately 54 miliseconds, however, the sampling time of each Arduino loop is very small in comparison to our sampling time. To handle this proplem, we implemented delay function of the Arduino to match the sampling times.

A saturation limit is also present to keep the overall PWM signal send to the DC motors within a 0-255 PWM range to prevent oscillation. Another saturation limit also controls the output of the controller subsystem to avoid any undesired errors that might happen at the video processing.

Overall PID algorithm can be investigated at *Algoritm 3*.

Algorithm 3: PID Controller Algorithm

```
maxSum // Integral Wind-up term
Ts // Sampling time
// Update of past error array
for int i = (pastSize - 1) → 0 do
    pastError[i] = pastError[i - 1]
pastError[0] = error
// Calculation of Derivative Term
delta = (pastError[0] - pastError[1]) / Ts
// Calculation of Integral term
sum = sum + (pastError[0] + pastError[1] + pastError[2]) / 3 * Ts
sum = min(max(sum , -1 * maxSum), maxSum) // Anti-Wind-up
// Calculation of PID output
motorCmd = int(Kp * error + Kd * delta + Ki * sum)
motorCmd = min(motorCmd,Max_Delta_PWM)
delay(Ts-duration) // duration:duration of each Arduino loop
```

Classification of Path Rotation

Due to unidealities of the motors, the controller tuned for the CCW movement failed to pass the test proposed under *Section 4.4*, therefore, the PID parameters and the base speed parameters were seperated in order to have same performance at both sides.

It should be also noted that, the *Data Processing Subsystem* produces β in $0-180$ degree range if the path is located with a positive angle with respect to vehicle or produces β in $180-360$ degree range otherwise. In that notation sense, if the β is

in $0-180$ degree range the vehicle is being operated on a path at counterclockwise direction and at clockwise diirection if the β is in $180-360$ degree range.

However, since the main frame of the video input, i.e., orientation of the vehicle is moving constantly, the counterclockwise movement can be understood by the vehicle if an undesired input changes the direction of the vehicle dramatically. To avoid these undesired effect on the controller and base speed calculation, an array that holds past five angle information was created. At every frame, the angle classified as 1 or 0 according to its degree range. The purpose of this array is to elliminate the undesired rotation determination array by checking last five angle data. The basic algoithm can be further investigated at *Algorithm 4*.

According to the value of *yonFin* variable, PID parameters tuned for both direction is fed to the controller algorithm.

Algorithm 4: Path Rotation Classification Algorithm

```
pastYon[yonSize] // Holds past rotations, has elements 0 or 1
// Update pastYon array
for int i = (yonSize - 1) → 0 do
    pastyon[i] = pastyon[i - 1]
pastyon[0] = yon
yonSum = 0
for int k = 0 → (yonSize-1) do
    yonSum = yonSum + pastyon[k]
yonAvg = yonSum/yonSize
if yonAvg < 0.5 then
    yonFin = 0
else
    yonFin = 1
```

3. Discussions on the Solution

   The proposed algorithm lacks the modelling of the system, thus, tuning of the PID parameters were handled by try and error. The state-space model of the plant will further be investigated under the term project of the another undergraduate course of our department, EE498:CONTROL SYSTEM DESIGN AND SIMULATION. However, it was not included due to its incompleteness.

   Other than the modelling part and rotation sensitivity, the proposed methods are same as the ones proposed in the conceptual design review report. And they function as expected.

3.3 Communication System

This systems is responsible for all communications of the system. It is one of the most crucial systems of this project since it provides a link between different parties of the

vehicle as well as communication with other vehicles. The requirements for this system are as follows;

1. The subsystem should ensure safe internal communication
2. The subsystem should ensure safe external communication

The system has two subsystems namely, ,

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

3.3.1 Internal Communication Subsystem

1. Requirements for the Solution
   (a) The microcontrollers should be able to communicate with each other via serial communication
   (b) The internal communication speed should be compatible with the processing speed of the lane detection subsystem
2. Solution for the Subsystem

   This subsystem covers the communication of the components in the vehicle. The only communicating parties are Raspberry-Pi and Arduino-Nano. That is why a serial communication protocol between them is enough to accomplish internal communication. The most reliable serial communication protocol that is USB is used for the vehicle. Since RPi is a computer, it can act as master and recognize Arduino as a device connected to its serial port.

   The technical details of communication is as follows:

   Raspberry side

   Serial communication can be implemented in any language but the choice is C++ since other algorithms are also implemented using it. Using <wiringPi.h> and <wiringSerial.h> libraries any piece of string can be sent to Arduino. However, the port of Arduino should be determined using Arduino IDE or other methods. In Linux based OS it is most probably /dev/ttyUSB0". Besides, the baud rate should be the same for each parties. Then using *Algorithm 5*, the payload string can be send via serial port.

Algorithm 5: RPi serial communication

include wiringPi, wiringSerial
set serial_device_ID
serial_device_ID= serialOpen("< serial_port >", < baud_rate >)
if *setup successfull* **then**
    send payload string
    SerialPuts(< serial_device_ID >, < payload >)
else
    connection unsuccessful

Arduino side

On the Arduino side instead of simply using `serialRead(.)` command, the `SerialCommand.h` library is used for more stable operation. This library allows us to associate any custom function to the specific incoming string. An example for reading a PWM value is shown in *Algorithm 6*

Algorithm 6: Arduino serial communication

include SerialCommand.h
declare object *scmd* of the library
declare global variable *value*
set baud rate
begin serial port
$scmd.addCommand(" < payload\_string > ", assc\_func)$

void assc_func
char* arg
arg = scmd.next()
if *arg is not NULL* **then**
    value=arg

In our case, baud rate is set to 9600 which can provide enough speed. Using the described structure any required value can be send to Arduino such as the mid-point of the road, target angle, distance sensor measurements and so on.

3. Discussions on the Solution

   Using `SerialCommand.h` library, consecutive commands with less than 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. The library works properly in the finalized version of the design

3.3.2 External Communication Subsystem

1. Requirements for the Solution

   (a) The subsystem should be able to communicate with the opponent via P2P Wi-Fi protocol

   (b) The subsystem should start race with the 3-way handshake mechanism required for establishment of TCP connection

   (c) Similarly, the subsystem should be able to trigger similar handshake mechanism during the race, which is referred as the main handshake mechanism throughout this report

   (d) For the second handshaking, the subsystem should be able to get the sensor data from vehicle detection subsystem to send messages

   (e) LEDs with corresponding colors should be lid to identify and display the messages that are sent

2. Solution for the Subsystem

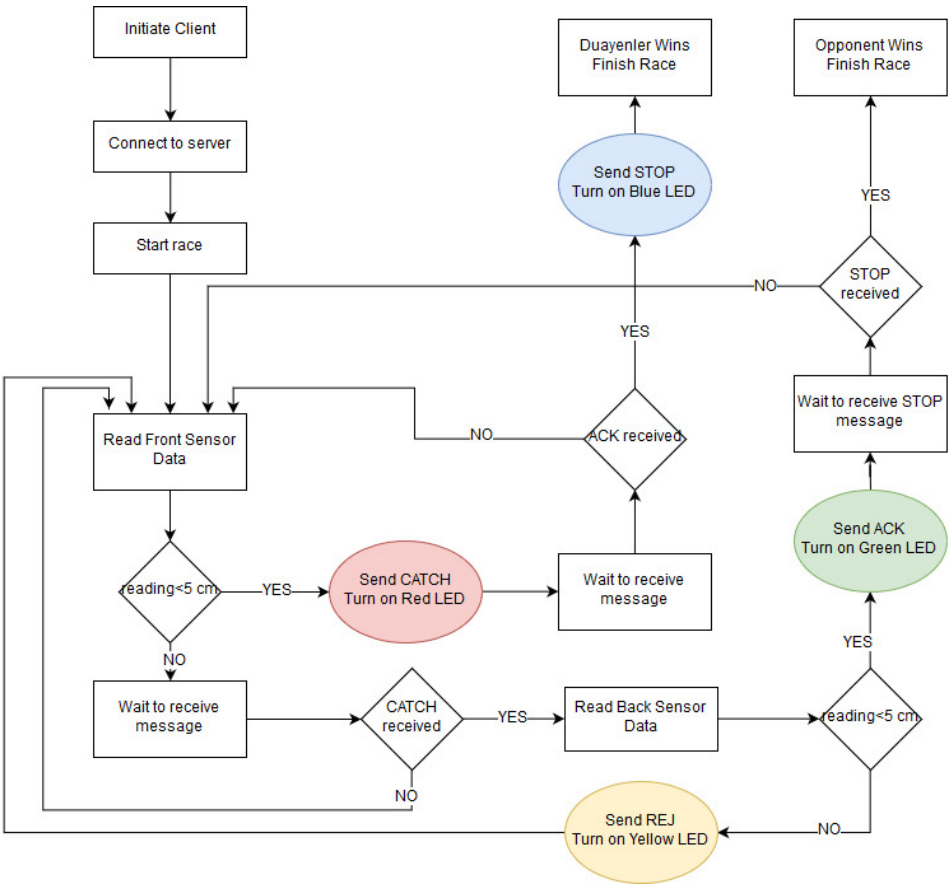


Figure 13: ASM Chart of the Algorithm of the External Communication Subsystem for Client Side

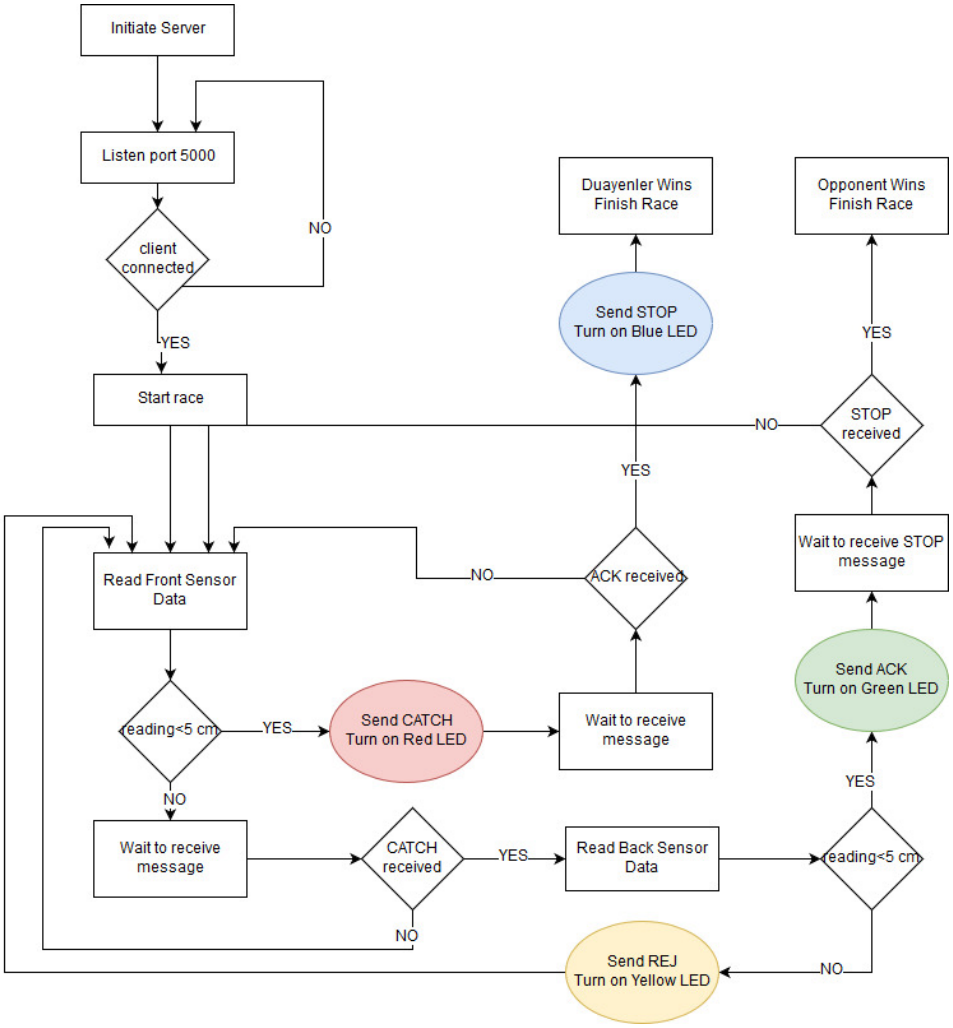


Figure 14: ASM Chart of the Algorithm of the External Communication Subsystem for Server Side

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 and host type(client/server) from the user and sensor readings from the vehicle detection subsystem. It can initiate and terminate other subsystems when necessary.

The first step of the game is setting up a Wireless Ad-Hoc Network. The fact that which side will provide the network is decided according to the the agreement with the opponent. After that decision, by changing the Pi's configuration files accordingly, one side sets up the network and the other connects to it after rebooting.

To implement the communication, the socket data structures are created in both server and client sides. At 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. This mechanism is common in every protocol that utilizes TCP, and called 3-way handshake.

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.

The ASM chart describing the algorithm of the external communication subsystem can be seen in *Figure 14* and *Figure 13*, for client and server sides, respectively. The fact that which team will be server or client is decided with the agreement between the teams before the race. Notice that, the only difference between client and server codes is the TCP connection parts before starting the race.

In the simplest form, there are two repeated actions inside the main loop. Firstly, front sensor is read. Secondly, device waits to receive message before timeout occurs. Setting proper timeout value is a little bit tricky. It is done according to the tests conducted with the opponents. The details of it can be found in the "External Communication Subsystem Tests and Results" section.

3. Discussions on the Solution

   The proposed method is not changed after the Critical Design Review Report. In the previous report, creating an ad-hoc wifi is proposed, but was not implemented. Implementation has been done by making appropriate changes in Pi's configuration files. The code was previously written in Python for simplicity. To unite it with image pocessing code, it is rewritten in C++. Besides, improvements such as adding proper timeout value and LED integration is made.

3.4 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;

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

The system has two subsystems namely,

1. **Direction Subsystem** which is responsible for the orientation of the vehicle and keeps the road and the vehicle aligned.

2. **Speed Subsystem** which is responsible for the overall speed of the vehicle by adjusting it considering other effects on the vehicle.

3.4.1 Direction Subsystem

1. Requirements for the Solution

   (a) The subsystem should drive the motors according to computation system outputs

   (b) The system should ensure that the vehicle follows the lane

2. Solution for the Subsystem

   As explained more detail in *Structure System*, the vehicle has two DC motors to provide differential derive and a caster-ball for balance and easy rotation of the vehicle. This subsystem acquires two important parameters 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 align itself in lateral direction. This data is acquired from the **PID Controller Subsystem.**

   Using Arduino and H-bridge motor driver, acquired PWM data is supplied to the motors for their speed control. L298N is selected for motor driver since it can drive both motor together and easily controlled by Arduino.

3. Discussions on the Solution

   The solution for this subsystem is fixed at the *Conceptual Design Report* stage. Since then, some improvements are made to match the voltage values at the same PWM value. Other than that, the solution fully satisfies the requirements.

3.4.2 Speed Subsystem

1. Requirements for the Solution

   (a) The subsystem should decrease the vehicle speed at the narrow lane

   (b) The subsystem should increase the vehicle speed at the wide lane

   (c) The subsystem should decrease the vehicle speed at the extreme disturbance

2. 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 3.2.2*.

   Main algorithm of this subsystem 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;

$$\text{PWM Offset} \ = \ K_{max} \ - K_1\beta$$

   where K_{max} is the maximum PWM value the base PWM should reach as it is on the straight path, the value for K_{max} can be determined by try and error test.

   However, it is also desired that the minimum PWM values shoul satisfy satisfying speed perfomance for the vehicle, therby a minimum PWM restrictions were put on this PWM calculation. It can be formulated as;

$$\text{PWM Offset} \ = \ max\{K_{max} \ - K_1\beta, K_{min}\}$$

   .

   where K_{min} is the minimum allowable PWM value for the DC motors, and it can be found by try and error. Moreover, since the path has varying properties, it was desired to have K_{min} is also dependent on β value. The implemented algorithm can be summuried at *Algoritm 7* and the hypotatical β vs base PWM graph can be investigated at *Figure 15*.

Algorithm 7: Base Speed Algorithm

```
if β > β1 then
    K_min = K_min1
else
    K_min = K_min2
Base_pwm = max(K_max − K1 ∗ β, K_min)
```

3. Discussions on the Solution

   The main solution idea is the same as the one proposed at *Conceptual Design Review Report*. The main difference is the dependency of the base PWM value on road rotation as it was explained in *Section 3.2.2*. The test results proved that the algorithm provides satisfactory performance to control longitudinal speed.

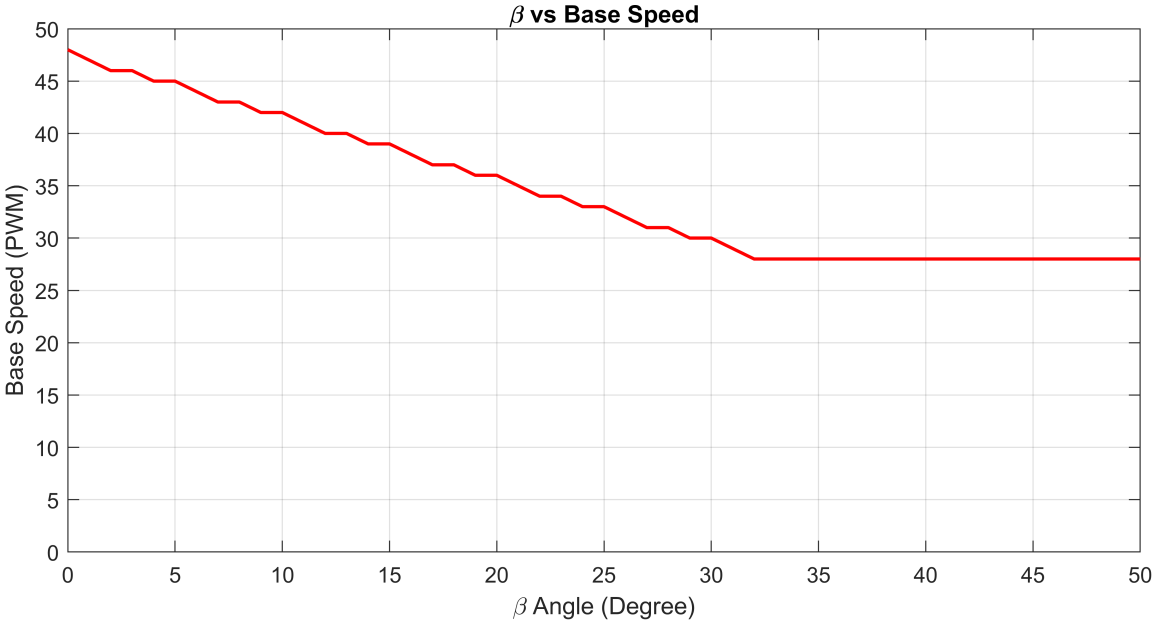


Figure 15: Therotical Base Speed PWM vs Theoratical β Angle

3.5 Motion System

This system is responsible for maintaining mechanical rigidity of the driving system. The requirement for this system is as follows;

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

The system has two subsystems namely,

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

3.5.1 Wheels Subsystem

1. Requirements for the Solution
   (a) The subsystem should ensure that the wheels can grip lane without slipping in all conditions
   (b) Vehicle should be able to turn steep bends of the specified lane
2. Solution for the Subsystem

   Wheel structure is one of the critical subsystem since it affects the structure of the PID controllers for direction and speed. The preferred structure has two separately

driven wheels for differential drive at the back and a caster ball in front for enabling the turning of the vehicle.

The location of the wheels are moved 5 cm to the front from the back end. This design change comes with two beneficial results. First, the distance between camera and the wheels are shortened which in turn reduces the delay between lane detection and driving. More specifically, the distance between the the target point determined by the lane detection system and the actual present point is decreased. Second, a torque component after the wheels become present which in turn allows caster ball to move over obstacles more easily.

In determining the wheels, high friction property is considered. Hence, super soft and slick tire are chosen with lighten aluminum rim. Besides, wheels with larger width is preferred to increase the possibility of sticking on the lane.

3. Discussions on the Solution

   After wheel subsystem tests, we observe the choice gives what we expect. Although tires make the path dirty, they fasten to the road quite successfully. Therefore, this system satisfies requirements.

3.5.2 Motors Subsystem

1. Requirements for the Solution

   (a) The subsystem should ensure that the motors can supply enough torque to accelerate the vehicle

   (b) The subsystem should ensure that the motors can execute driving system outputs without deviation

2. Solution for the Subsystem

3. Discussions on the Solution

3.6 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. The requirements for this system are as follows;

1. The system should ensure that structure is robust for external effects

2. The system should ensure that structure is balanced

3. The system should ensure that vehicle has a good appearance

The system has two subsystems namely,

1. **Chassis Subsystem** which is responsible for the mechanical connections of components in the vehicle.

2. **Printed Circuit Board Subsystem** which is responsible for preventing cable crowd caused by connection of electrical components.

3.6.1 Chassis Subsystem

1. Requirements for the Solution

   (a) The subsystem should ensure that the chassis is rigid

   (b) The subsystem should ensure that the chassis have enough space for components

   (c) The subsystem should ensure that the chassis can provide low center of mass

   (d) Camera holder should be integrated to the front of the vehicle

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

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

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

2. 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.

   Chassis structure relies on two custom-designed plexiglass layers. Raspberry Pi, Arduino Nano and Voltage regulator are placed on the upper layer while motor driver, Li-Po battery and powerbank are on lower one. To keep the center of mass of the vehicle close to the ground, Li-Po battery and powerbank are 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 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 16*). After integration, the view of the camera can completely cover the both edges of the path. Lastly LEDs for handshake protocols are also placed on the camera holder at the updated design.

CAD model of the chassis can be seen in *Figure 16*.

Figure 16: Isometric view of the 3D Drawing of the vehicle

3. Discussions on the Solution A few minor updates have been made since *Critical Design Review Report* such as changing the location of wheels and voltage regulator. However, no changes is needed in the overall design of the chassis. The vehicle now has good look with changed plexiglass colors of both layers and camera holder. In terms of center of mass, the vehicle tested itself by falling out of the elevated path without tipping over many times.

3.6.2 Printed Circuit Board Subsystem

1. Requirements for the Solution

   (a) The subsystem should ensure that all the electronic components are placed on PCB

   (b) The subsystem should ensure that all the connections are firmly secured and robust to vibrations.

2. 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.

   In the finalized design, there are two places that required PCB. First one is RPi GPIO pinouts. Two distance sensor and handshake LEDs are controlled by RPi. In order to have a compact connection between sensors and LED module, a PCB is designed. Its purpose is just to convert GPIO pinouts to compact headers for sensors and LED module. It is designed as a socket that can be placed on the RPi's GPIO pins. The second PCB is designed for handshake LEDs such that only one cable with 5 pins is enough to connect it to RPi.

   Other than that a cable with 6 pinned header is designed in order to connect Arduino to the motor driver. A switch is placed between Li-Po battery and motor driver so that the race scenario can be executed properly.

3.7 Compatibility of the Subsystems

The block diagram showing the interaction of the subsytem block with each other can be seen from the *Figure 17*. 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.

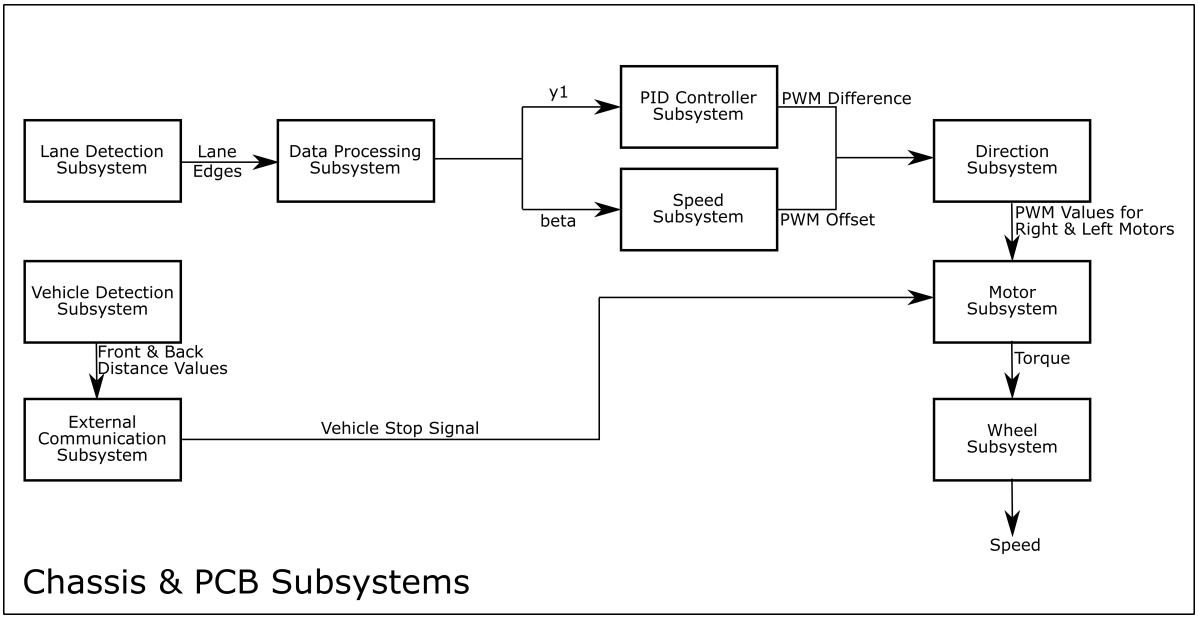


Figure 17: Block Diagram of the Project and the Interaction of the Subsystems

4 Detailed Tests for the Subsystems

This section presents followed test procedures and the outcome of the tests.

4.1 Lane Detection Subsystem Tests and Results

1. Light Condition Test

   (a) Mirror the Raspberry Pi screen into Laptop via VNC

(b) Execute the lane detection algorithm in Raspberry Pi

(c) Change the location of the camera and Pi to conduct test

(d) Observe the results in different locations

(e) If the visible lane sides can be detected without any additional object, the result of the test can be considered as success.

2. Visual Disturbance Test

(a) Mirror the Raspberry Pi screen into Laptop via VNC

(b) Execute the lane detection algorithm in Raspberry Pi

(c) Put different objects into lane

(d) Observe the results with different disturbances

(e) 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.

Results of Lane Detection Subsystem Tests

The lane detection tests are conducted to asses reliability of the detection pipeline. A sample test result is shown in *Figure 18*. The tests reveal that the subsystem satisfies its requirements by detecting lane lines. Note that, not all lines are detected, only the lines that are in the ROI are detected. One thing to note for the robustness of this subsystem is subsystem works as expected long as HSV filter is tuned for the medium.

4.2 Vehicle Detection Subsystem Tests and Results

1. Vehicle Detection Test in Closed Environment with One Sensor:

(a) Make the connection of the desired sensor and Pi properly

(b) Hold the sensor at an angle of 90 degree with respect to ground

(c) Place the test object 5 cm in front of the desired

(d) Observe the output of the subsystem

(e) Repeat the step 3 & 4 with different distances

(f) If the output of the subsystem is very closed to the actual measurement (i.e., with an error rate approximately 1 percent) the test result can be considered as success.

2. Vehicle Detection Test in Closed Environment with Two Sensors:

(a) Repeat the test steps of the *Front Vehicle Detection Test in Closed Environment with One Sensor* when both sensors are connected to the rear of the vehicle.

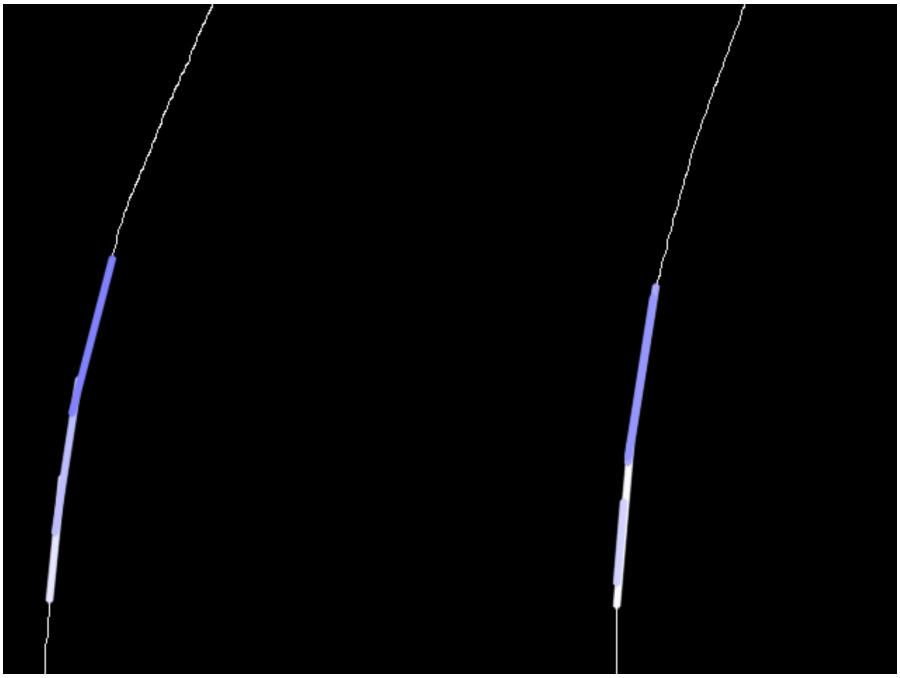


Figure 18: Lane Detection Test Result

3. Angled Approach Test:
   (a) Make the connection of the desired sensor and Pi properly
   (b) Hold the sensor at an angle of 90 degree with respect to ground
   (c) Place the test object 5 cm in front of the sensor with 30 degree angle with respect to the sensor
   (d) Observe the output of the subsystem
   (e) Repeat the step 3 & 4 with different distance and angle values
   (f) If the output of the subsystem is very closed to the actual measurement (i.e., with an error rate approximately 1 percent) the test result can be considered as success.

4. Vehicle Detection in Different Sunlight Conditions Test:
   (a) Repeat the test steps of the *Front Vehicle Detection Test in Closed Environment with Two Sensors* in CCC (Cultural and Convention) ground under direct sunlight
   (b) Repeat step 1 in CCC (Cultural and Convention) under artificial light, in other words, under no direct sunlight conditions
   (c) Repeat steps 1 & 2 for different locations of E Building including Graduation Laboratory
   (d) If the output of the subsystem is very closed to the actual measurement (i.e., with an error rate approximately 1 percent) the test result can be considered as success.

Results of Vehicle Detection Subsystem Tests

All the test procedures mentioned at the **Section 4.2** is applied to the VL6180X Time-of-Flight distance sensor. At first, test results were satisfying for the back sensor, but not for the front one. Tests were repeated with switching front and rear sensors. Again, the sensor in the front of the vehicle measured erroneously. Then it is realized that the position of the front sensor is problematic. In the first design of the chasis, the front sensor was placed behind a hole under the the stick supporting camera. It is realized that this configuration was blocking the sensor. Then, in the second chasis design, front sensor is brought in front of the stick. After repeating tests results, it is confirmed that the problem with the front sensor is eliminated.

Time-of-flight sensors show 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. In addition to the success in different light conditions test, the sensor shows 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.

4.3 Data Processing Subsystem Tests and Results

1. Data Assessment Test
    (a) Link the output of Lane Detection subsystem to Data Processing subsystem.
    (b) Asses if the output coincide with physical reality of the path
2. Output Stability Test
    (a) Place the vehicle on a fixed point on the path
    (b) Observe the variation of the control outputs

Results of Data Processing Subsystem Tests

The tests asses the ability and performance of the subsystem. Numerous tests are made. The first set of tests cover the robustness of the subsystem by placing an obstacle of the subsystem. The test scenario and its results are presented in *Figure 19,20 and 21*. 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. Since the implementation is frozen after Conceptual Design Report, there has not been any significant change in the test performances since then.

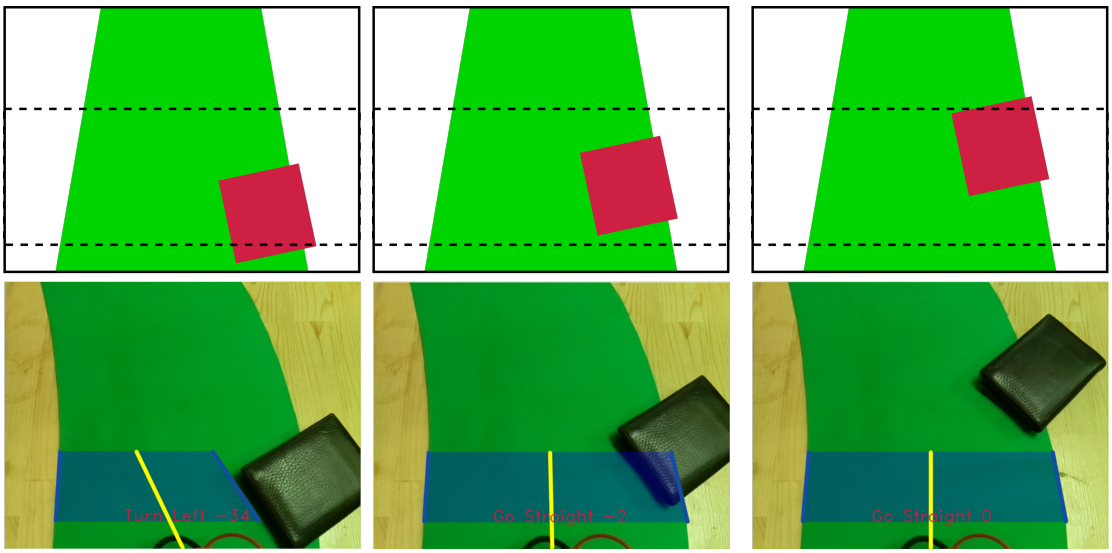
The second set of tests cover the robustness of the subsystem as well, but under changing lighting conditions and on different surface materials. The results of this test

is presented in *Figure 22 and 23*. 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.

The third and the last test is made to see if the outputs are consistent when the vehicle is fixed on a point on the path. This test reveals the robustness of Data Processing Subsystem as it gives a measure on the stability of control outputs. PID Controller can work as expected, only if the provided inputs are predictable and repeatable. The test case and the results are given in *Figure 24, 25 and 26*. Statistical measures graphical data is also visible in *Table 1*. The stability test is satisfactory, as the variance of the three measurements are quite low.

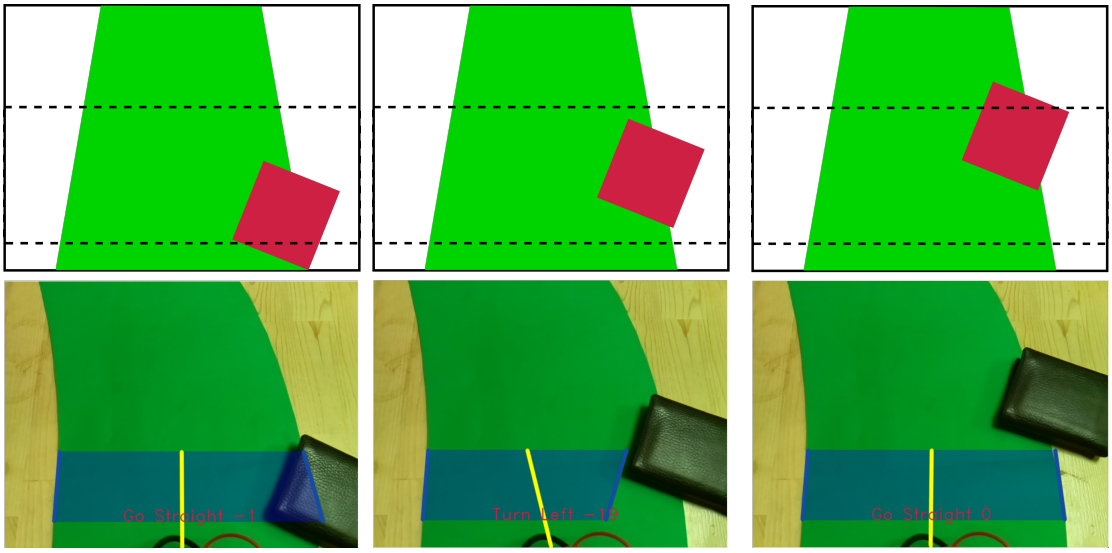
Table 1: Statistical Stability Results

Test	mean(β)	variance(β)	mean(y1)	variance(y1)
Straight Area	-3.3	0.7	3.7	0.2
Bend Area	-12.3	0.2	-15.6	0.1
Curved Area	-21.8	1.8	-44.5	0.5



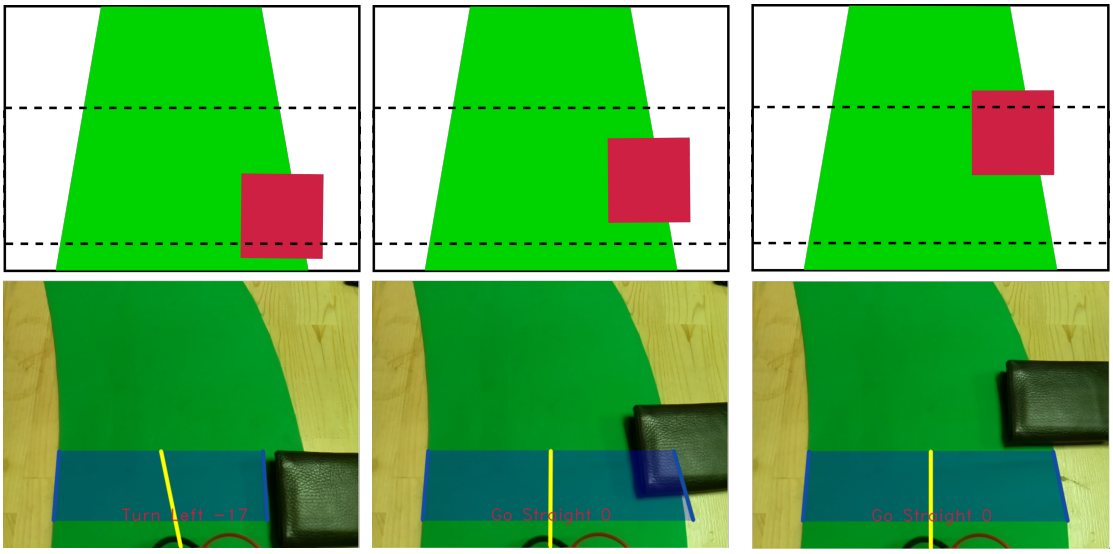
(a) Obstacle is at the Beginning of the Path (b) Obstacle is at the Middle of the Path (c) Obstacle is at the End of the Path.

Figure 19: A Test Scenario: Downward Inclined Obstacle on the Path.
Upper Half: Proposed Tests, Lower Half: Results



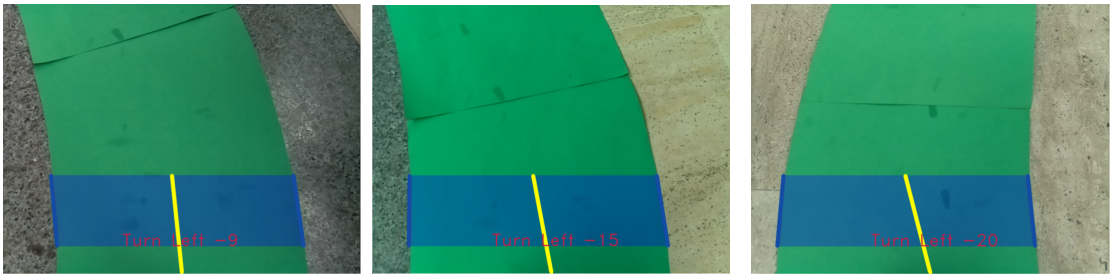
(a) Obstacle is at the Beginning of the Path (b) Obstacle is at the Middle of the Path (c) Obstacle is at the End of the Path.

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



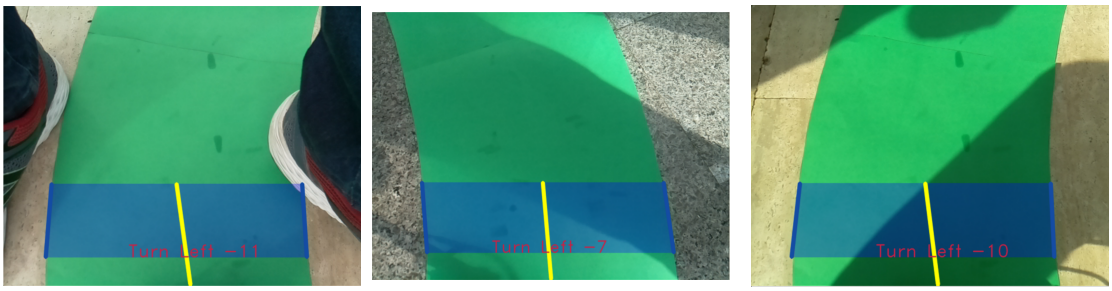
(a) Obstacle is at the Beginning of the Path (b) Obstacle is at the Middle of the Path (c) Obstacle is at the End of the Path.

Figure 21: 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 22: 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 23: A Test Scenario Results: KKM Outdoor Path Detection

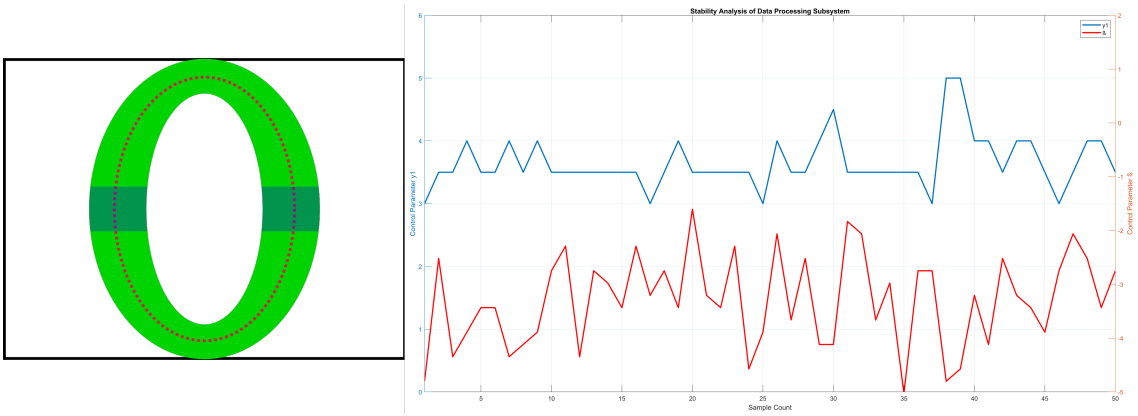


Figure 24: Left: The Vehicle is placed on Straight Area.
Right: The Control Outputs

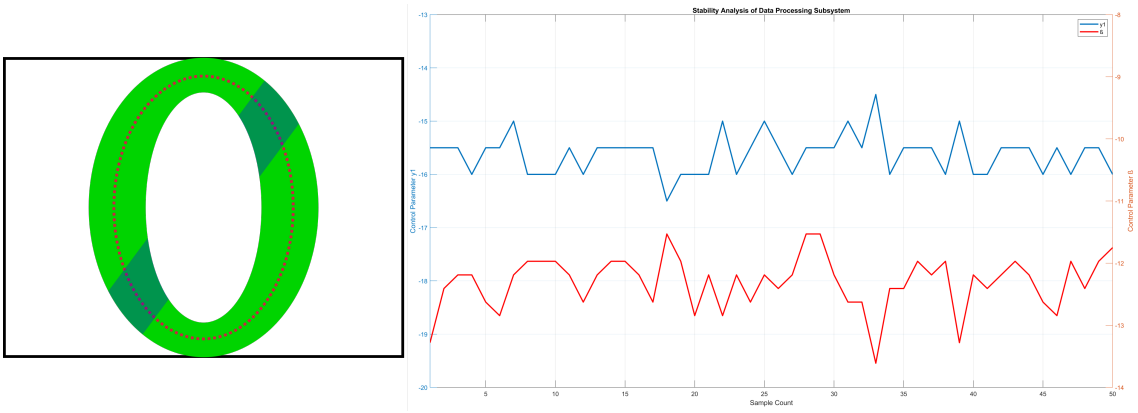


Figure 25: Left: The Vehicle is placed on Bended Area.
Right: The Control Outputs

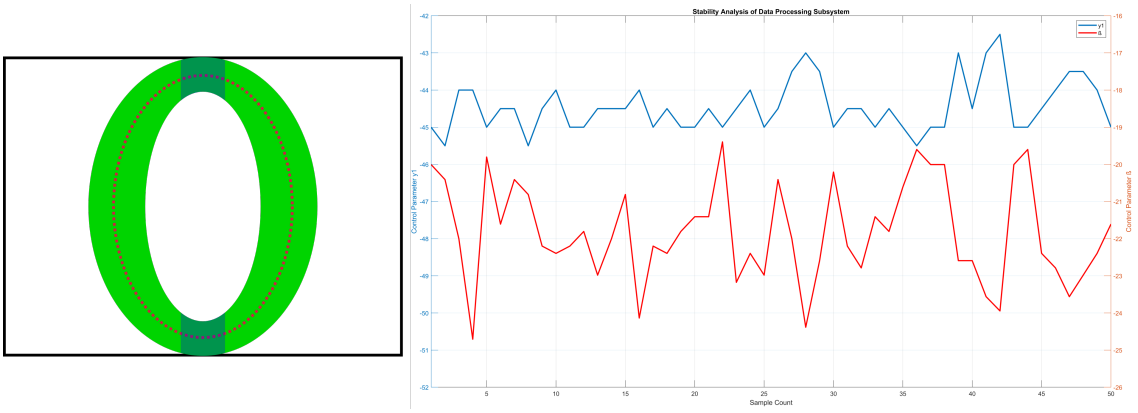


Figure 26: Left: The Vehicle is placed on Curved Area.
Right: The Control Outputs

4.4 PID Controller Subsystem Tests and Results

1. PID Parameters Tuning Test :

   (a) Charge each Li-Po battery cell up to 3.95 V

   (b) Check the regulator output voltage, if it is not at 10.44, play with internal potentiometer to make it so

   (c) Start the data processing from Raspberry Pi

   (d) Equate K_d and K_i values to zero

   (e) Give the power to the motors

   (f) Sweep the K_p value to follow the part wityh unstable, oscillatory behaivor

   (g) Sweep the K_d value to damp these oscillation, stop before the system become unstable

   (h) Sweep the K_i value to reduce steady-state error, stop before the system become unstable

(i) Observe the behaviour of the vehicle

(j) If the vehicle is able to complete the path at ten consecutive lap, the controller can be classified as working.

2. Bump Test for Distance Control:

(a) Set-up a lane as in *Figure 27*.

(b) Make the necessary connection between motors Arduino and data processing unit

(c) Drive the vehicle with PID parameters to be tested.

(d) Collect the distance error between the center of the lane and current position of the vehicle.

(e) Plot the time vs distance graph at Matlab using the collected distance errors.

(f) Calculate necessary performance parameters from the plot.

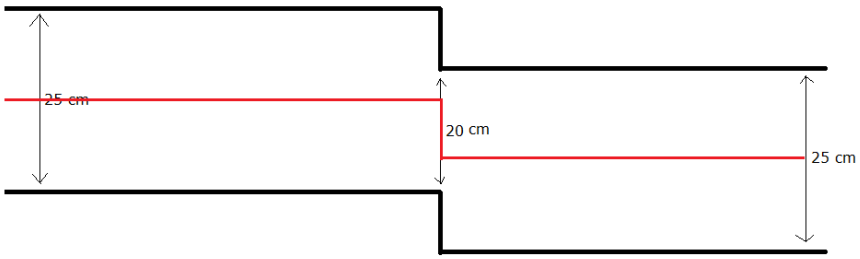


Figure 27: Bump Test for Distance Control

3. Tracking a Path with Obstacles Test:

(a) Make the necessary connection between motors Arduino and data processing unit

(b) Place the vehicle to the desired path with obstacles

(c) Observe the behaviour of the vehicle

(d) If the vehicle can follow the path and compensate the steady state errors due to obstacles without showing oscillatory behaviour and in less than 2 seconds, the result of the test can be considered as success.

4. Path Tracking Test with Physical Disturbances:

(a) Make the necessary connection between motors Arduino and data processing unit

(b) Place the vehicle to the desired empty path

(c) Observe the behaviour of the vehicle

(d) If the vehicle can follow the path and compensate the steady state errors due to physical disturbance without showing oscillatory behaviour and in less than 2 seconds, the result of the test can be considered as success.

5. Sampling Time Consistency Test:

(a) Make the necessary connection between motors Arduino and data processing unit

(b) Place the vehicle to the desired empty path

(c) Record the data proccessing time of each frame to a text file.

(d) Plot the time vs processing time plot.

(e) If the processing time, i.e., sampling time of the plant is oscillating inside 10% band of thesampling time value used by PID controller subsystem, the test can be classified as success.

Results of PID Controller Subsystem Tests

The resuts for *PID Parameters Tuning Test* can be seen at *Figures 28 & 29*. The results satisfied the claimed succes requirement by completing at least 20 consecutive lap in one run for both direction. The controller also showed satisfactory performance to other test.

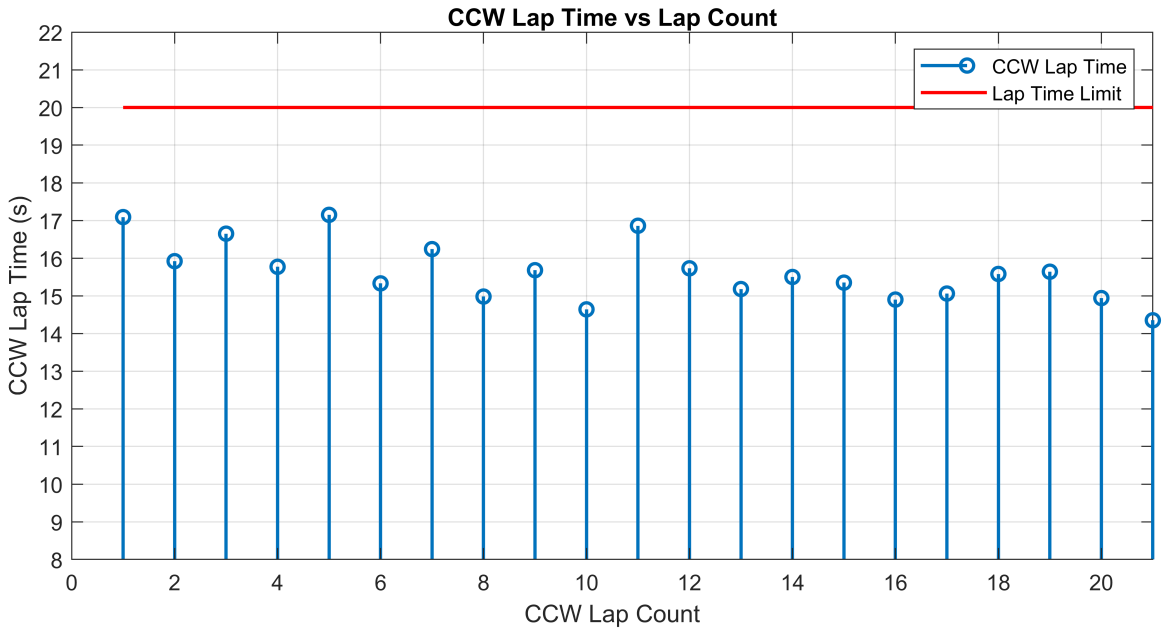


Figure 28: Results of PID Tune Test for CCW Movement on the Path

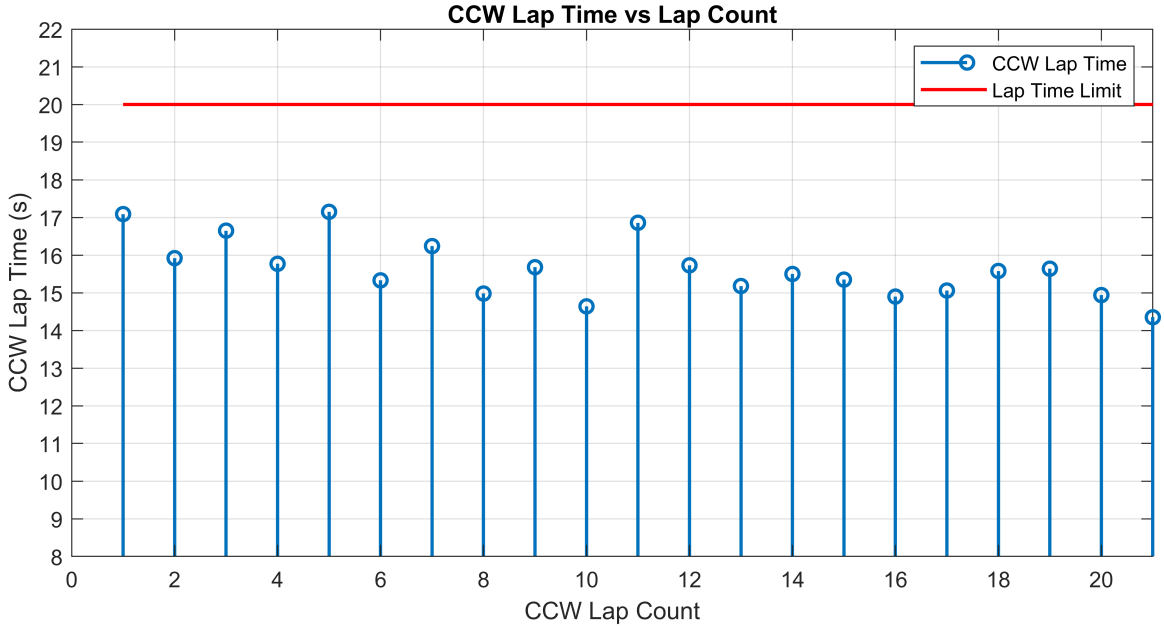


Figure 29: Results of PID Tune Test for CW Movement on the Path

The resuts for *Sampling Time Consistency Test* proved the stability of the *Data Processing Unit* and the relaibility of the PID Controller Subsystem, since the sampling time for the discrete time systems is very crucial and should be consistent at all time.

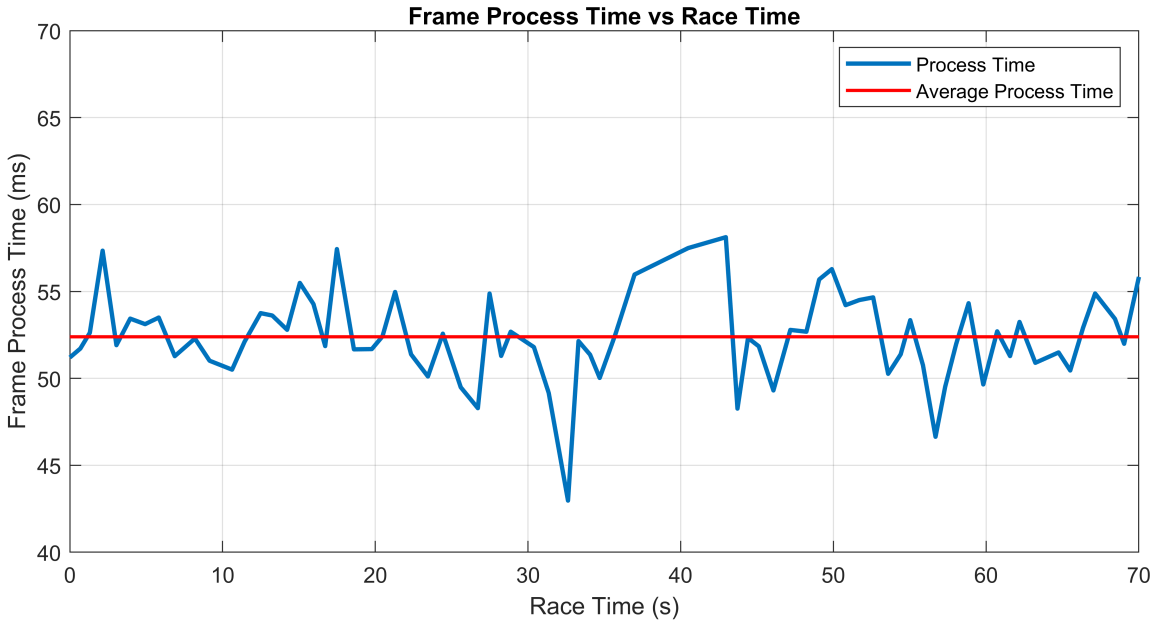


Figure 30: Sampling Time Consistency Test Result

4.5 Internal Communication Subsystem Tests and Results

1. Data Retrieval Test

   (a) Generate data on Raspberry Pi in a rate that reflects the time consumed of Data Processing subsystem. This will yield a realistic data rate.

   (b) Do the initial integration between Arduino and Raspberry Pi.

   (c) Send random text data to Arduino.

   (d) Send data from Raspberry Pi to Arduino.

   (e) Increase data speed to the specified data rate.

   (f) Check the accuracy of the retrieved data.

Results of Internal Communication Subsystem Tests

To check the accuracy of the sent data an LCD display is connected to Arduino and observed the values on the display. The observed values are correct implying that the system works properly.

4.6 External Communication Subsystem Tests and Results

1. Catching the Opponent Test:

   (a) Create an Ad-Hoc Wi-Fi from Pi so that opponent can connect (or connect to opponent's Ad-Hoc Wi-Fi)

   (b) Enter opponent's ID number and host type (server or client) to the terminal

   (c) Approach the opponent less than 5 cm

   (d) The test result can be considered as success if blue LED is lid after RED LED without significant delay(i.e. no crashing) and the device immediately stops.

2. Caught by the Opponent Test:

   (a) Create an Ad-Hoc Wi-Fi from Pi so that opponent can connect (or connect to opponent's Ad-Hoc Wi-Fi)

   (b) Enter opponent's ID number and host type (server or client) to the terminal

   (c) Let the opponent to approach less than 5 cm

   (d) The test result can be considered as success if green LED is lid without significant delay(i.e. no crashing) and the device immediately stops.

3. Rejected by the Opponent Test:

   (a) Create an Ad-Hoc Wi-Fi from Pi so that opponent can connect (or connect to opponent's Ad-Hoc Wi-Fi)

(b) Enter opponent's ID number and host type (server or client) to the terminal

(c) Place an object in front of the front sensor in a distance less than 5 cm

(d) The test result can be considered as success if RED LED is lid without significant delay and the device does not stop.

4. Rejecting the Opponent Test:

(a) Create an Ad-Hoc Wi-Fi from Pi so that opponent can connect (or connect to opponent's Ad-Hoc Wi-Fi)

(b) Enter opponent's ID number and host type (server or client) to the terminal

(c) Let the opponent to place object in front of the front sensor in a distance less than 5 cm

(d) The test result can be considered as success if yellow LED is lid without significant delay and the device does not stop.

Results of External Communication Subsystem Tests

As mentioned in the design description section of the external communication subsystem, there exists a waiting period to receive messages from opponent. This duration, which is also called timeout, directly affects the frequency of the front sensor readings, because it is spent between consecutive readings. In the test results, setting it to high values(higher than 1s) gives good results in *Caught by the Opponent Test*, since it increases the probability to receive messages from the opponent. However, it results in crashing in *Catching the Opponent Test*, because of the latency of the sensor reading. On the other hand, setting very low timeout(lower than 100 ms) speeds up the process and yields successful results in *Catching the Opponent Test*. However, it decreases the probability of getting messages from the opponent, so *Caught by the Opponent Test* fails. After test results, it is decided that 0.3 seconds is an appropriate value for timeout. By doing so, the probability of crashing is made lower and receiving messages is made higher.

4.7 Direction Subsystem Tests and Results

1. Straight Drive Test:

(a) Make the necessary connections between motors, motor controller and the Arduino

(b) Set the PWM values of the motors equal

(c) Observe the behaviour of the motors

(d) Increase the PWM value of the slower motor until a point the vehicle can go in a straight line.

(e) Record this PWM difference to use in PID controller subsystem

2. Circular Drive Test:

   (a) Make the necessary connections between motors, motor controller and the Arduino

   (b) Desired curvature is decided

   (c) According to motion of the vehicle PWMs of the motors are set

   (d) PID parameters are set according to this test

Results of Direction Subsystem Tests

The results of *Tests 1 & 2* 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. Thereby, the design finalized with differential drive vehicle.

4.8 Speed Subsystem Tests and Results

1. Determination of Minimum Base Speed Coefficients Test:

   (a) Charge each Li-Po battery cell up to 3.95 V

   (b) Check the regulator output voltage, if it is not at 10.44, play with internal potentiometer to make it so

   (c) Start the data processing from Raspberry Pi

   (d) Use the PID parameter values determined at *Test 1*

   (e) Give the power to the motors

   (f) Sweep the K_{min1} value to find large enough minimum PWM value so that the vehicle do not leave the lane at corners

   (g) Sweep the K_{min2} value to find large enough minimum PWM value that the vehicle do not speed up too much on straight sections of the path so that it does not enter the corners too fast.

   (h) Observe the behaviour of the vehicle

   (i) If the vehicle is able to complete the path at ten consecutive lap, the coefficients can be classified as working. Record these values

2. Determination of Maximum Base Speed Coefficients Test:

   (a) Charge each Li-Po battery cell up to 3.95 V

   (b) Check the regulator output voltage, if it is not at 10.44, play with internal potentiometer to make it so

   (c) Start the data processing from Raspberry Pi

   (d) Use the PID parameter values determined at *Test 1*

(e) Give the power to the motors

(f) Sweep the K_{max} value to find small enough minimum PWM value so that the vehicle do not speed up too much on straight sections of the path so that it does not enter the corners too fast.

(g) Observe the behaviour of the vehicle

(h) If the vehicle is able to complete the path at ten consecutive lap, the coefficient can be classified as working.

3. Determination of Slow Down Coefficients Test:

(a) Charge each Li-Po battery cell up to 3.95 V

(b) Check the regulator output voltage, if it is not at 10.44, play with internal potentiometer to make it so

(c) Start the data processing from Raspberry Pi

(d) Use the PID parameter values determined at *Test 1*

(e) Give the power to the motors

(f) Sweep the K_1 value to find large enough value so that the vehicle slow down properly but does not saturate to the minimum PWM limit

(g) Observe the behaviour of the vehicle

(h) If the vehicle is able to complete the path at ten consecutive lap, the coefficient can be classified as working.

Results of Speed Subsystem Tests

The result of each parameter tuning test can be found at *Figure 31*

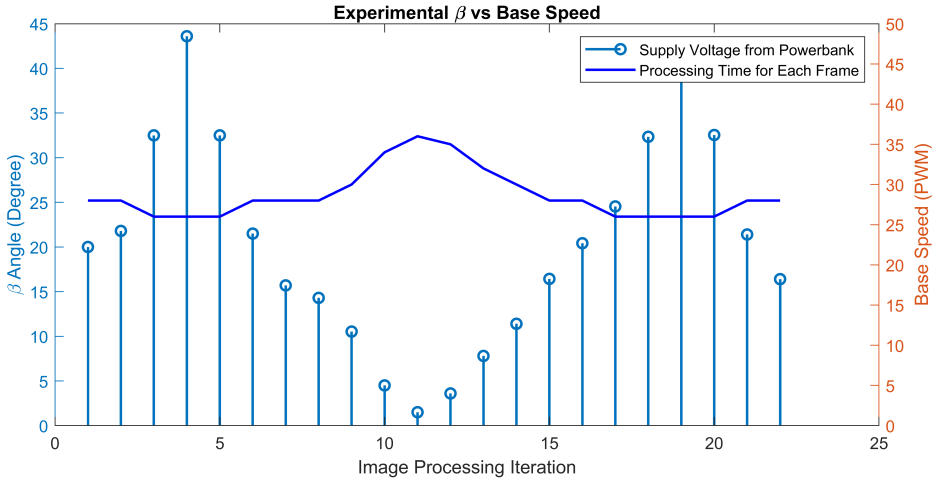


Figure 31: Experimental Base Speed PWM vs Experimental β Angle

4.9 Wheels Subsystem Tests and Results

1. Handling Test:

   (a) Place the vehicle on the path

   (b) Apply a horizontal force

   (c) Observe the behavior

   (d) 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.

Results for Wheels Subsystem Tests

Since the weight distribution and total weight of the vehicle is properly adjusted, the slipping of the wheels is expected to be quite unlikely which is what is observed at the test.

4.10 Motors Subsystem Tests and Results

1. Torque Test:

   (a) Fix the motor at horizontal position with respect to ground

   (b) Attach an object of one kilogram

   (c) Contact the seller for more information

Results of Motion System Tests

4.11 Chassis Subsystem Tests and Results

1. Inertia test:

   (a) Prepare a straight path

   (b) Power up the vehicle

   (c) Execute the edge detection and control algorithm

   (d) Give different type of disturbances

   (e) Observe the deviation from straight line

   (f) Repeat the process with different component configurations

Results of Chassis Subsystem Tests

As mentioned previously, the disturbances applied on the vehicle and the deviation is between the levels that can be handled by other algorithms. Furthermore, custom-designed chassis added rigidity to the structure which in turn provides better test results.

4.12 Printed Circuit Board Subsystem Tests and Results

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

   (a) Open multimeter for short circuit test

   (b) Find the ends of each routing

   (c) Check the continuity using multimeter probes

   (d) Check if there is any unwanted short circuit

   (e) If exist, eliminate

Results of PCB Subsystem Tests

When checked, there is no unwanted short circuit and all other connections were successful for both PCBs and cables. Thus, the subsystem satisfies the requirements.

5 Budget

In this section spendings of the project will be investigated under two part. Fistly, real spendings will be analyzed and secondly, the cost of reproducable product will be investigated.

5.1 Actual Expenditures

Besides the component spendings spesified at *Table 2*, 100 $ worth mechanical components, such as rechargeble drill and approximately 100 $ worth electrical components from porevious term projects were utilized throughout the project. Thereby, a total of approximate 616.5 $ budget and 1700 man-hour were spent on this project.

Table 2: Total Spendings for the Project

Component	Approximate Spendings (in Dollar)
Raspberry Pi	50
Raspberry Pi Camera	47
Chassis & Mechanical Components	45
Arduino	20
DC Motor & Wheel	100
Motor Driver and Regulator	9.5
Powerbank	25
Li-po Battery	23
Sensor & Cable	60
Other Spendings	37
Total Project	416.5

5.2 Cost Analysis of Reproducible Product

Cost anaysis for the **Kobra 6.5** can be investigated under *Table 3*. The reproducible vehicle costs under 200 dollar as stated by the project requirements. However, it should be noted that the 200 dollar budget does not cover additional expenses such as li-po battery charger and micro-USB chager.

Table 3: Cost Analysis for the Project

Component	Number	Total Price (in Dollar)
Raspberry Pi 3B	1	47
Waveshare Raspberry Pi Camera (D)	1	22
Chassis Components	1	13
Arduino Nano	1	4.8
Polulu 150:1 6 V 200 RPM DC Motor	2	42
Bond Silicon Wheel	2	8
Motor Driver and Voltage Regulator	1	4
Xiaomi 10000 mAh Mi Powerbank (Ver 3)	1	12
ProFuse 11.1 V 1750 mAh Li-po Battery	1	23
Polulu VL6180X ToF Distance Sensor	2	17
LED headlight/LED	-	0.2
Styrofoam, Glue & Cardboard	-	6.5
Total Project	-	199.5

As can be reffered from the *Table 3*, some component modification were made for the final design after conceptual design review report. For example, DC motors were replaced with the ones having lower rpm values, or powwerbank as replaced with the one having quick charge capabilities to elliminate unidealities in performance.

6 Power Analysis

Table 4: 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	0.35	4.2	0.95	11.4
Distance Sensor	19m	62.7m	40m	132m
Total	1.349	8.9127	3.69	25.032

The *Table 4* shows the consumption under regular case and extreme case. If extreme scenario is considered, full power consumption of Raspberry Pi is supplied from power-bank while motors are supplied by Li-Po battery as can be seen from the *Figure 32*.

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.

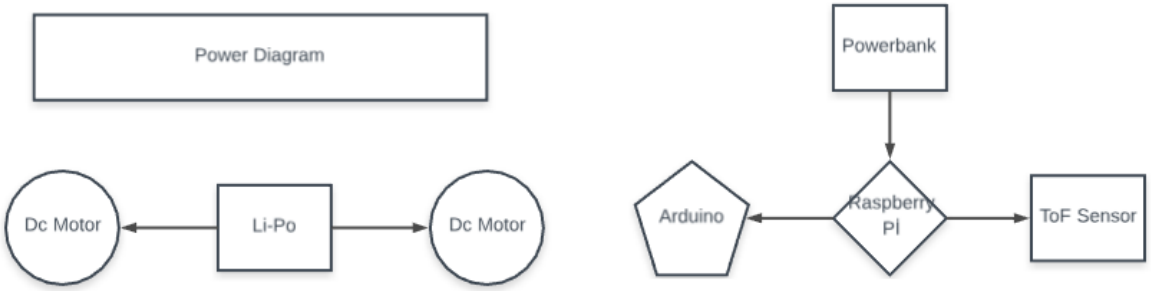


Figure 32: Electrical Architecture of the Project

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. Although, previous chosen powerbank, in therory, was capable of producing enough power for Raspberry Pi, in reality, its fluctiations in voltage output caused undesired low voltage warnings from Raspberry Pi which leads to closing of two cores of its cpu. As a result, this fluctiations in voltage caused undesired fluctiations in processing time as can be seen *Figure 33*

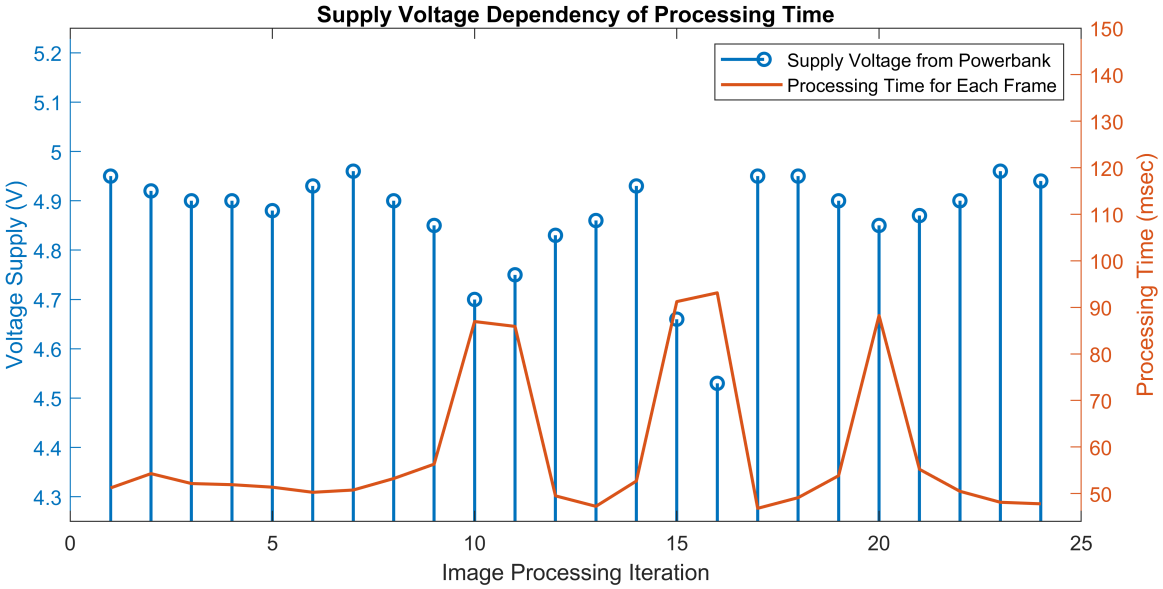


Figure 33: The Relation Between Supply Voltage from Powerbank vs Process Time

7 Deliverables

Deliverables of the product are listed as follows;

- KOBRA 6.5
- Race Track
- Flash stick with sofware
- User Manual
- Micro USB Cable

8 Discussions

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

Safety is the most important consideration of every project, so during design, it is considered after every solution step. Product is tested under tougher than general customer use such as no cooling of raspberry pi or constant usage of 8 hours etc.

The most dangerous sections are Li-Po battery, Arduino and Motor driver, in other words, where the current flows. Although Li-Po battery is dangerous for a toy, in design it placed between two layers of chassis to protect it from damage or direct sun light. Besides of this, these three components share the same ground path, so charging cycle is tested, additionally, to avoid any ground loop.

Raspberry pi is the main computation component of Kobra 6.5, so it has some overheating issues, especially hot days. Therefore, It is cooled by FAN and heat sink system. Additionally, it's power totally isolated from dangerous high current line (as mentioned previous paragraph) to eliminate any over voltage/current condition.

Thanks to these considerations, if there is no over charging condition of Li-Po battery, this product can be used totally safely.

8.2 Application Areas

At first sight, this project seems like a toy for children. As an RC car, two children can play with each other for fun. However, when project analyses in detail such as considering sub-systems, it is a small model of unmanned vehicle which has anti collusion and obstacle avoidance features. Furthermore, with modification of the sub-systems, this project can be evaluating different areas such as lane tracking algorithm can be enhanced to object tracking and this vehicle can follow specified object, or handshake algorithm can be evaluating to IoT applications which requires machine communications.

Considering these applications or more, this project could be primitive model of many specific applications.

8.3 Environmental Effects

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 report consists of the summary of the results of entire design effort. For each subsystem, the solutions are explained in detail and the results of the tests are discussed in detail.

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 CDRR solutions, test steps and test results. The system has been assembled into a final chassis. Mentioned limitations in CDRR are enhanced, as shown in the test result. 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. Finally, safety issues, application areas and environmental effects are discussed.

This work is done in a way that it is capable of handling every way, every case and scenario, by the great efforts of the members of DUAYENLER. The company members believe that this project will bring an innovative approach to the design of autonomous cars with its solutions and solution approaches.

A USER MANUAL

ABOUT KOBRA 6.5: KRAL KOBRA

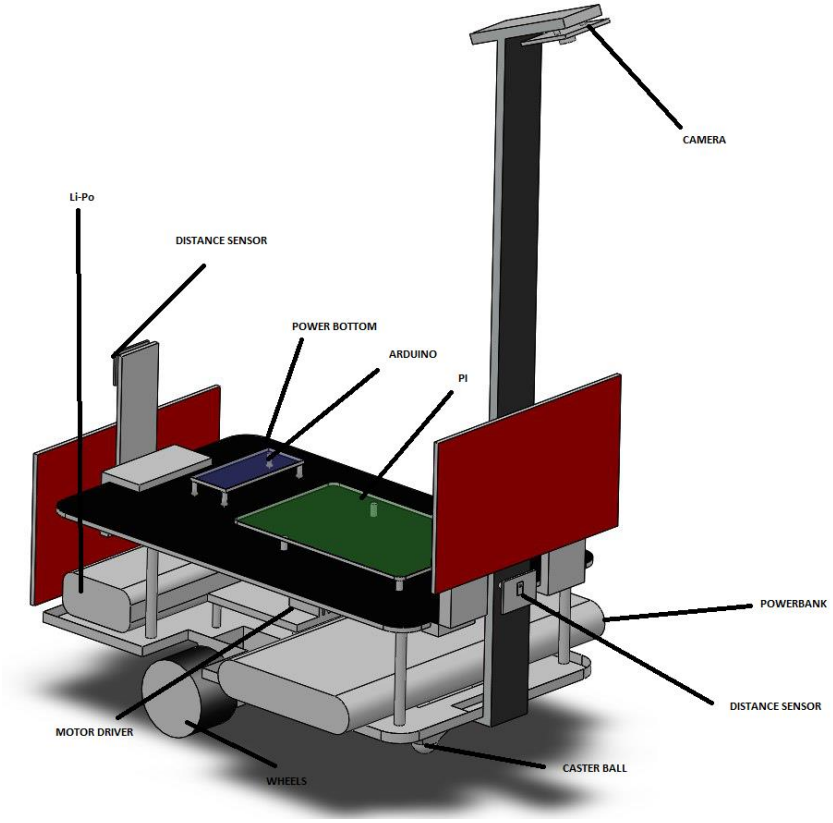


Figure 1: Kobra 6.5 information

WARNING: Do not allow children to dismantle the KOBRA 6.5!

WARNING: Do not overcharge the KOBRA 6.5!

BOX INCLUDES

1x Kobra 6.5

1x Race Lane

1x USB Cable

1x Flash stick

NOTE: A external HDMI capable Monitor, 3 Cell Li-Po battery charger and Quick Charge 3.0 capable plug-in charger should be bought separately to use KOBRA 6.5 safely.

USAGE PROCEDURE:

Before race:

1- Check battery levels (1xLi-Po and 1xLi-Ion)
2- Connect the Micro USB cable of Powerbank to Raspberry Pi with monitor
3- Run delivered software to connect the opponent
4- Set the lane up
5- Place two KOBRA 6.5 on the lane
6- Check connection according to LEDs
7- Push the power button to start the race

CONNECTION

By using provided software, you can connect two KOBRA 6.5. Contact DUAYENLER ltd. Şti. if you face any undesired problem with the software.

CHARGING OF KOBRA 6.5

KOBRA 6.5 has two power supply into its chassis, so two of them should be charged correctly.

Li-Po battery should charge up to 12.6 V, overcharging is DANGEROUS.

NOTE: Li-Po charger which can avoid from overcharge is strongly suggest using.

Li-Ion battery could charge up to 100%.

Warranty Issues

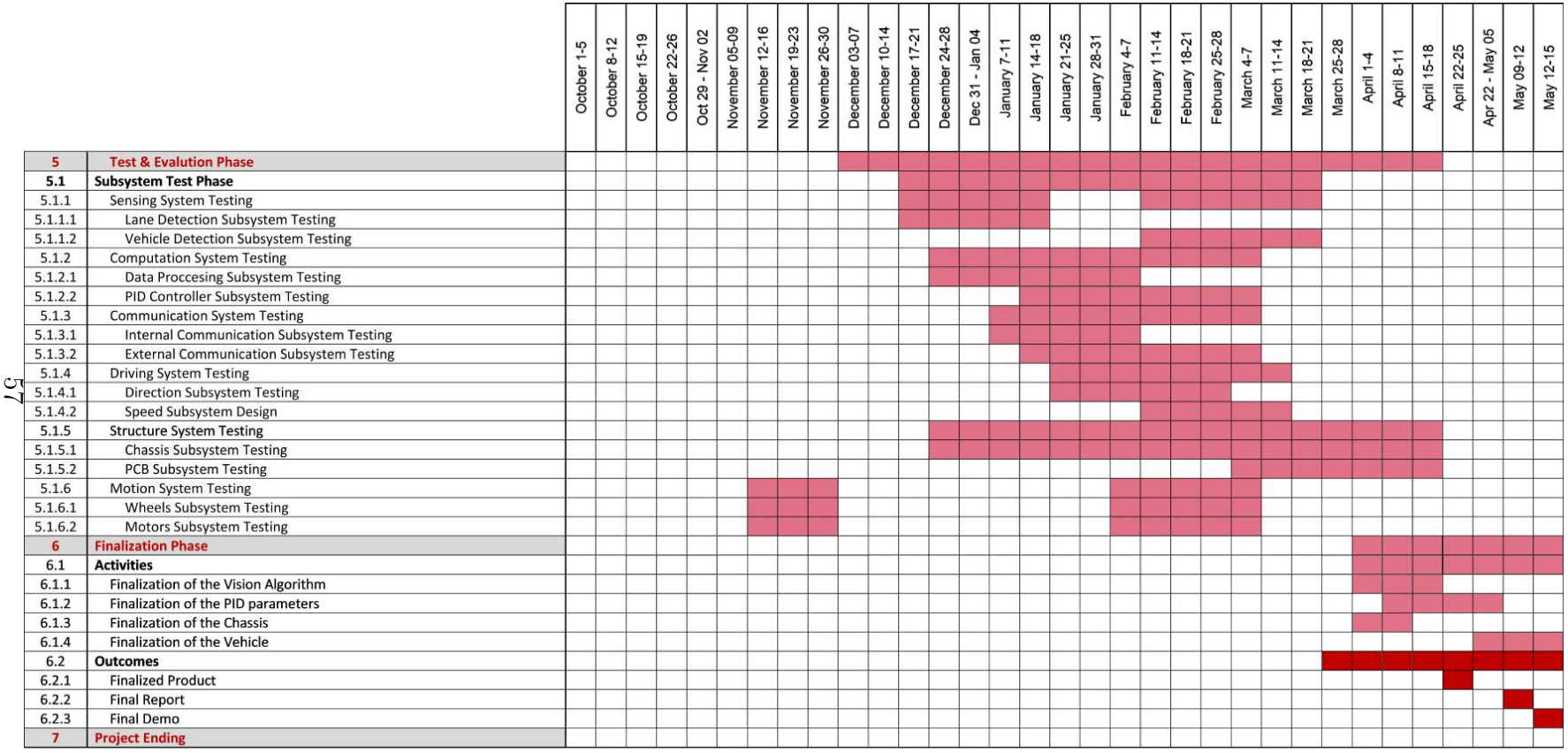
Following conditions will cause your KOBRA 6.5 to lose its warranty.

- Do not use KOBRA 6.5 in any environment with water.
- Do not throw KOBRA 6.5 height higher than 10 cm.
- Do not apply any kind of force to KOBRA 6.5 to check its strength.
- Do not use KOBRA 6.5 at temperature higher than 40 Celsius.
- Do not use damage batteries

B 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