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 [WRITE HERE ACTUAL COST VALUE], whereas cost to produce the commercial vehicle is [WRITE HERE ONLY PRODUCTION VALUE]. 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.

DONT FORGET THESE VALUES

2 Introduction

3 Design Description

THIS PART IS REWRITTEN FOR FINAL REPORT.

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

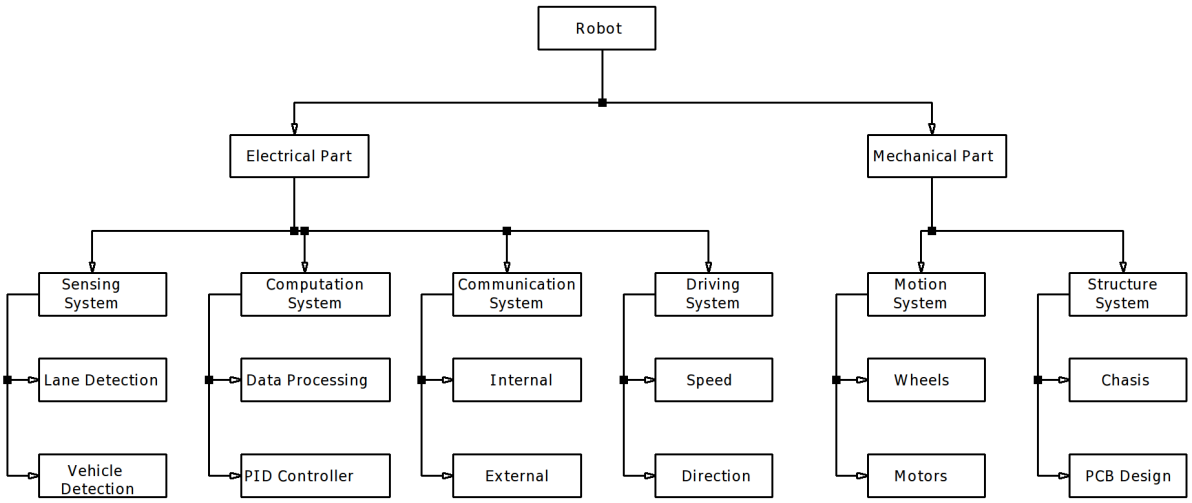


Figure 1: Organization of the Project

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

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

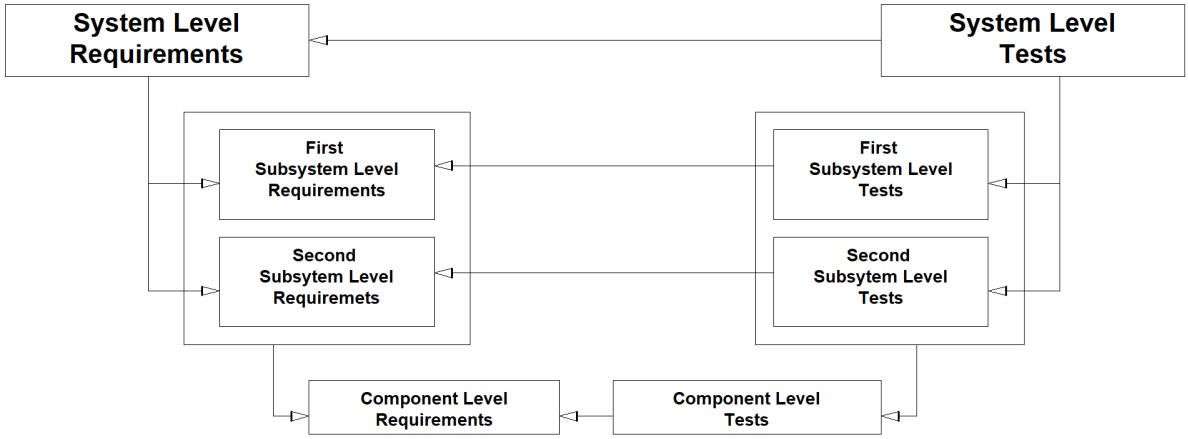


Figure 2: V-Model

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

THIS PART IS REWRITTEN FOR FINAL REPORT.

A. Requirements for the Solution

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

B. Solution for the Subsystem

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

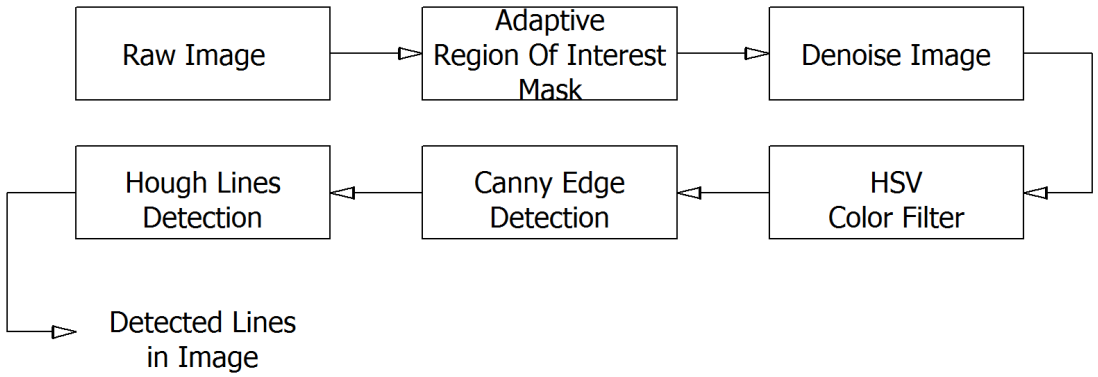


Figure 3: Block Diagram of the Lane Detection Subsystem

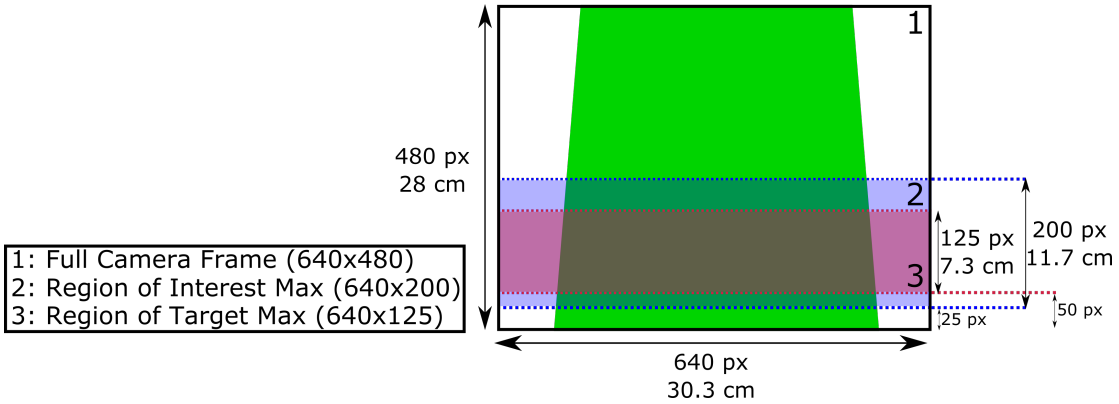


Figure 4: Explanation of Frame Regions

The input to this subsystem is provided by Raspberry Pi camera mounted on top of the vehicle. The camera frame resolution is 640x480 px. One thing to for the camera frame is that horizontal mapping and the vertical mapping of the pixels are not the same. That is, same amount of pixels in both directions do not correspond to same length in real life. The proposed solution first masks out a region of interest (ROI) of $640x200px$. 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:

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

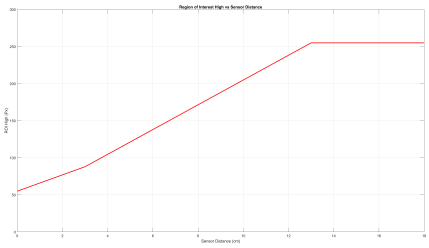
$$ROI\_Height = (ROI\_High - ROI\_Low)px \tag{2}$$

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

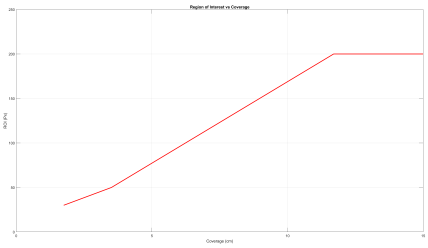
$$ROI\_High = \begin{cases} 225, & front\_distance \geq 13cm \\ 14 * (front\_distance) + 43, & 3 \leq front\_distance < 13cm \\ 10 * (front\_distance) + 55, & front\_distance \leq 3cm \end{cases} \tag{4}$$

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

3. Discussions on the Solution

3.2 Computation System

THIS PART IS REWRITTEN FOR FINAL REPORT.

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

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

The system has two subsystems namely,

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

3.2.1 Data Processing Subsystem

THIS PART IS REWRITTEN FOR FINAL REPORT.

A. Requirements for the Solution

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

B. Solution for the Subsystem

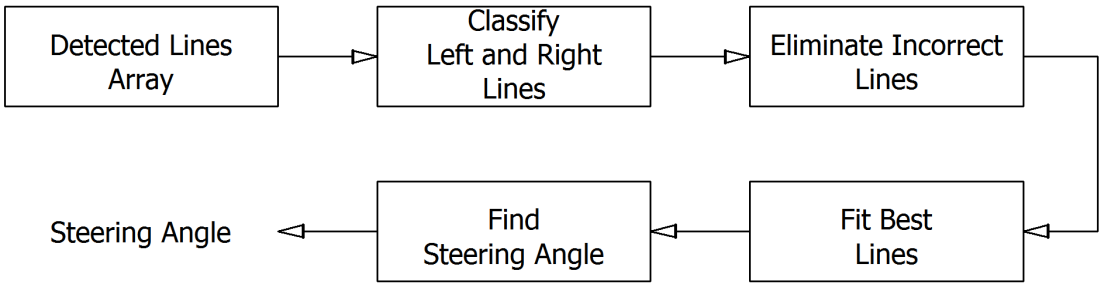
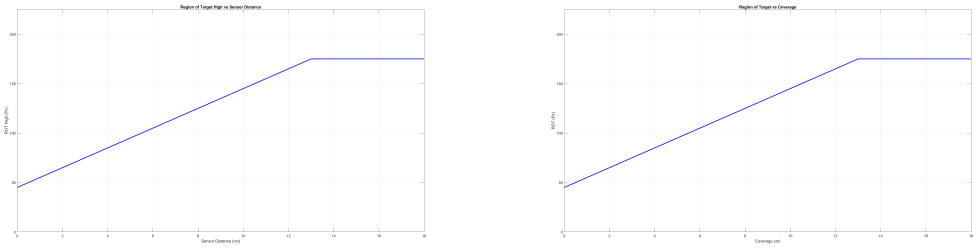


Figure 6: Block Diagram of the Data Processing Subsystem



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

Figure 7: Graphs Defining ROT_Height

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

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.

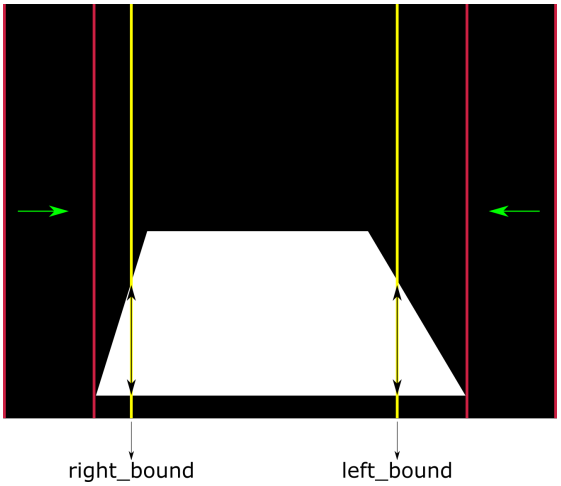
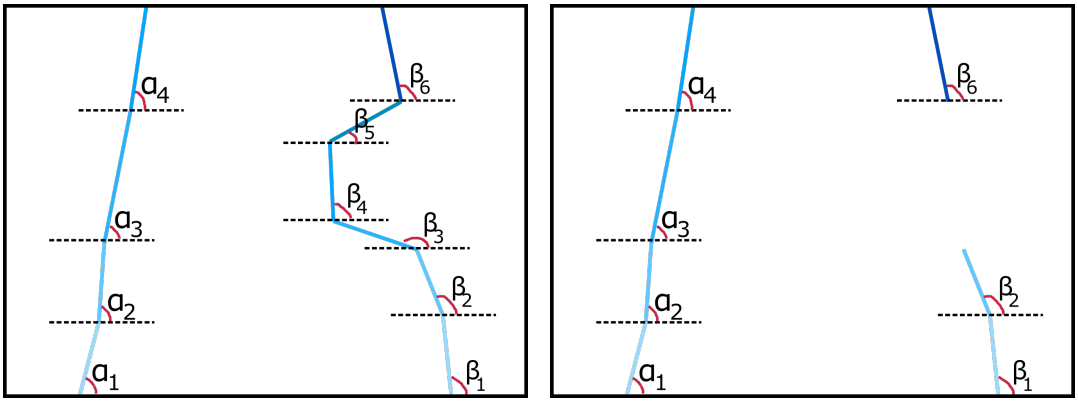


Figure 8: Finding Center of the Path

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
```

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



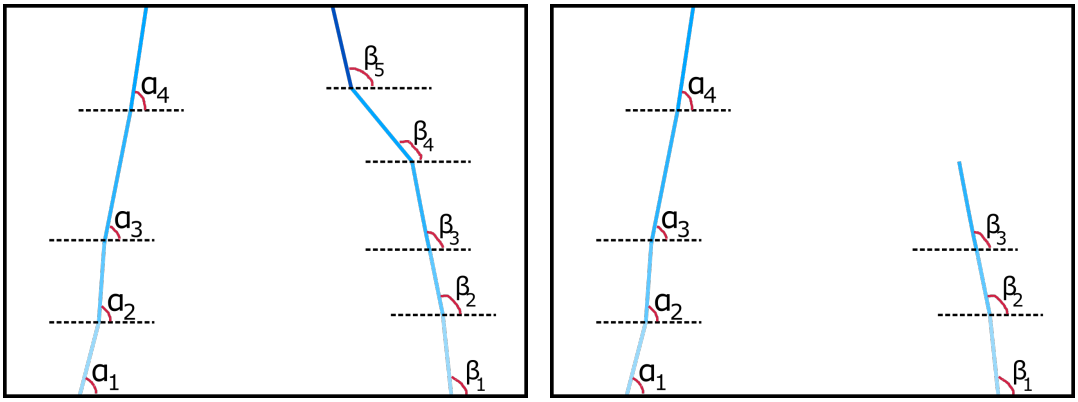
(a) Before Eliminating Incorrect Lines (b) After Eliminating Incorrect Lines

Figure 9: A Sample Scenario on Eliminating Incorrect Lines

the correctness of the steering angle depends on how successful this step is realized. The idea behind this step is evaluating the slopes consecutive lines and assessing whether change in the slope is ordinary or abnormal. The *Figure 9* exemplifies a possible scenario. In this figure, the blue lines represent the detected lines in ROI whereas α and β represent the slopes of the detected lines. Clearly, there are no disturbance on left lines since α values are similar to each other. However if β values are observed, possibly there is an obstacle on right line covered by β_3, β_4 and β_5. This can be concluded by observing slope differences $(\beta_2 - \beta_3)$ and $(\beta_5 - \beta_6)$. To ignore this obstacle, it is enough to remove lines with slopes β_3, β_4 and β_5 as in *Figure 9b*. Even though the count of lines is decreased, elimination of incorrect lines are realized and the best line fit will be more correct. Another scenario is shown in *Figure 10*. Again the shown lines are the ones in ROI. In this scenario, left line has no problems. Right lines, however, a bit problematic. The problem is revealed when $(\beta_3 - \beta_4)$ is observed. To determine whether $(\beta_1, \beta_2, \beta_3)$ or (β_4, β_5) is the correct set of lines, left lines are observed and the set which is more symmetric to left lines are selected as right lines. The resulting correction is shown in *Figure 10b*. This is the basic idea behind eliminating incorrect lines in Data Processing subsystem. This idea is generalized by considering other possible obstacle types and shapes. The generalized idea is complicated and would take too long to present here. The summarized idea is presented in *Algorithm 2*.

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

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



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

Figure 10: Another Scenario on Eliminating Incorrect Lines

Algorithm 2: Line Elimination Algorithm

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

the $[-90, 90]$ range where negative values indicate to turn left and positive values indicate to turn right. This output is sent to PID Controller subsystem.

C. Discussions on the Solution

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

3.2.2 PID Controller Subsystem

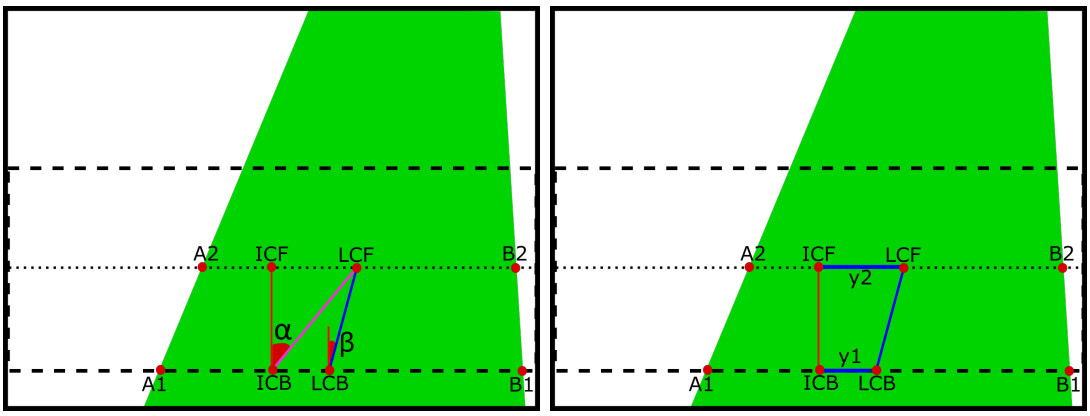
1. Requirements for the Solution

(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

- **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.

Design & Implementation of the Controller

3. Discussions on the Solution

3.3 Communication System

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

Script 1: C++ class for serial communication

```
void ArduinoComm::sendToController(std::string payload) {
/************************/
int serialDeviceId = 0;
serialDeviceId = serialOpen("/dev/ttyACM0", 9600);
std::cout << "sender " << serialDeviceId << std::endl;
if (serialDeviceId == -1) {
std::cout << "Unable to open serial device" << std::endl;
return;
}
if (wiringPiSetup() == -1) {
return;
}
serialPuts(serialDeviceId, payload.c_str());
return;
}
```

3. Discussions on the Solution

3.3.2 External Communication Subsystem

1. Requirements for the Solution

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

   (b) The subsystem must start race with a handshake mechanism

   (c) Similarly, the subsystem must be able to trigger another handshake mechanism at the end of the race

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

2. Solution for the Subsystem

3. Discussions on the Solution

3.4 Driving System

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

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
3. Discussions on the Solution

3.5 Motion System

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

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
2. Solution for the Subsystem
3. Discussions on the Solution

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

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

   Current version of chassis can be seen in *Figure 13*.
3. Discussions on the Solution

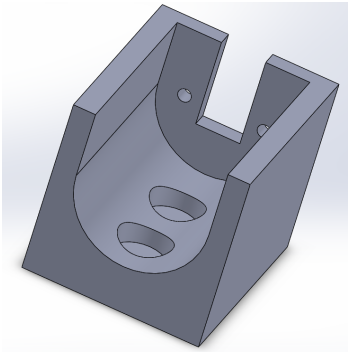


Figure 12: Motor holder 3D view

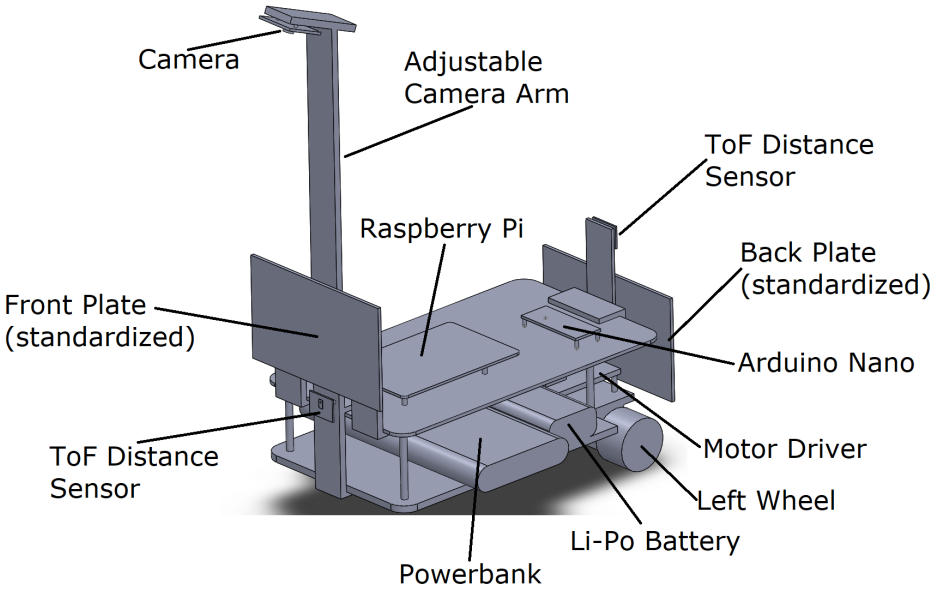


Figure 13: Isometric view of the 3D Drawing of the Vehicle

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

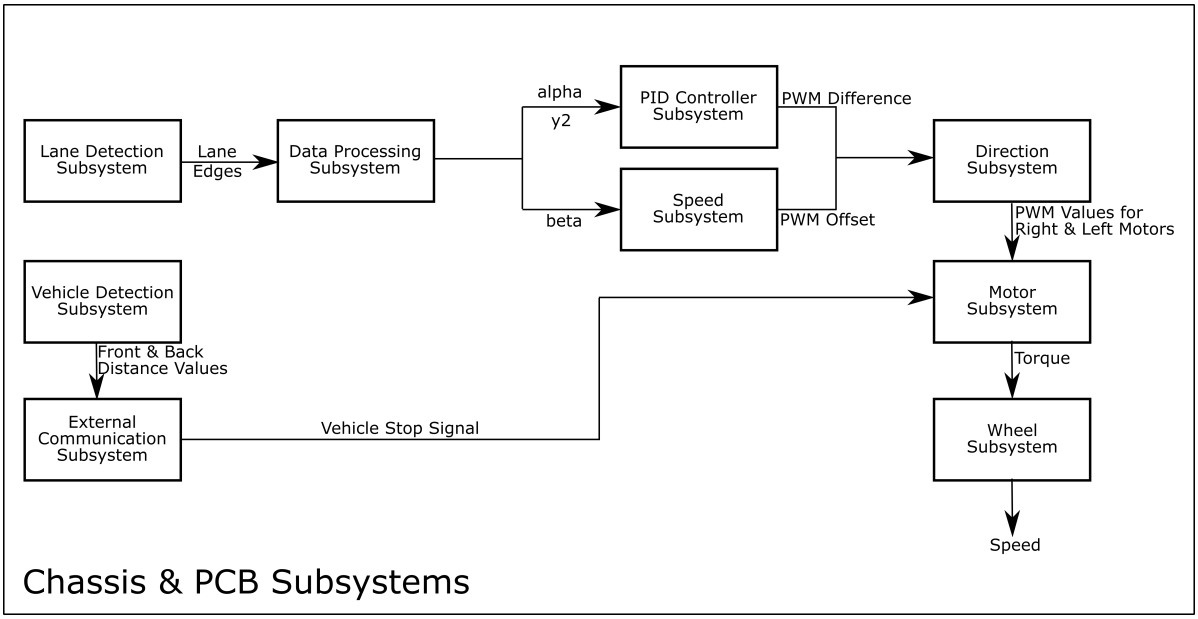


Figure 14: Block Diagram of the Project and the Interaction of the Subsystems

3.7 Compatibility 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

THIS PART IS REWRITTEN FOR FINAL REPORT.

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.

3. 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 15*. 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.

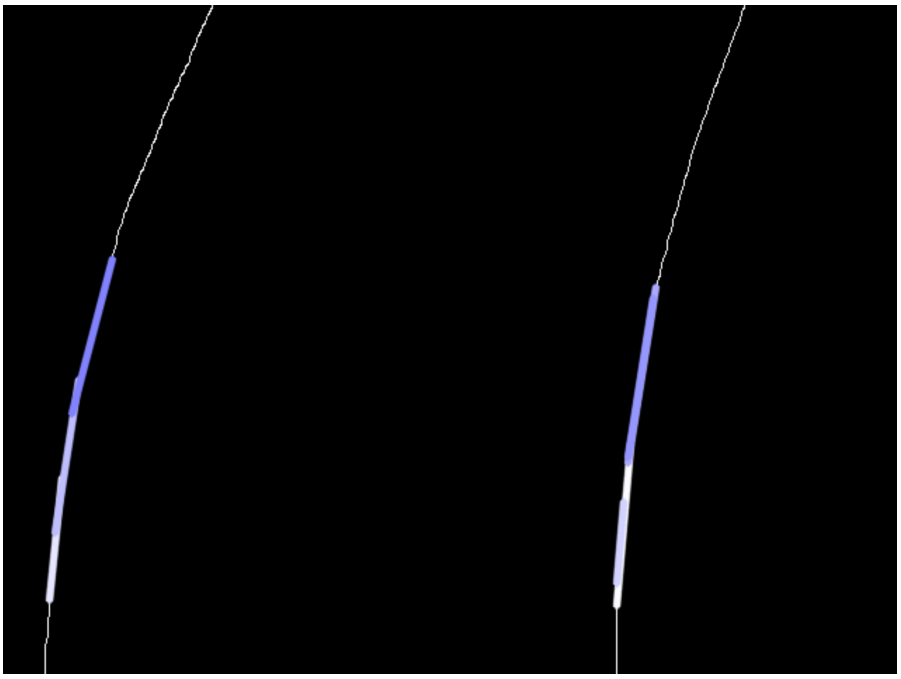


Figure 15: Lane Detection Test Result

4.2 Vehicle Detection Subsystem Tests and Results

1. Front Vehicle Detection Test in Closed Environment:

   (a) Make the connection of the desired sensor and Arduino 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 generates logical positive for distances smaller than 5 cm and logical zero for distances greater than five, the test result can be considered as success

2. Rear Vehicle Detection Test in Closed Environment:

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

3. Angled Approach Test:

   (a) Make the connection of the desired sensor and Arduino 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 generates logical positive for distances smaller than 5 cm for all angle values with respect to sensor and logical zero for distances greater than 5 cm, the test result can be considered as success

4. Vehicle Detection in Different Sunlight Conditions Test:

   (a) Repeat the test steps of the *Front Vehicle Detection Test in Closed Environment* 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 generates logical positive for distances smaller than 5 cm under all light conditions and logical zero for distances greater than 5 cm, the test result can be considered as success

5. Results of Vehicle Detection Subsystem Tests

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

3. Results of Data Processing Subsystem Tests

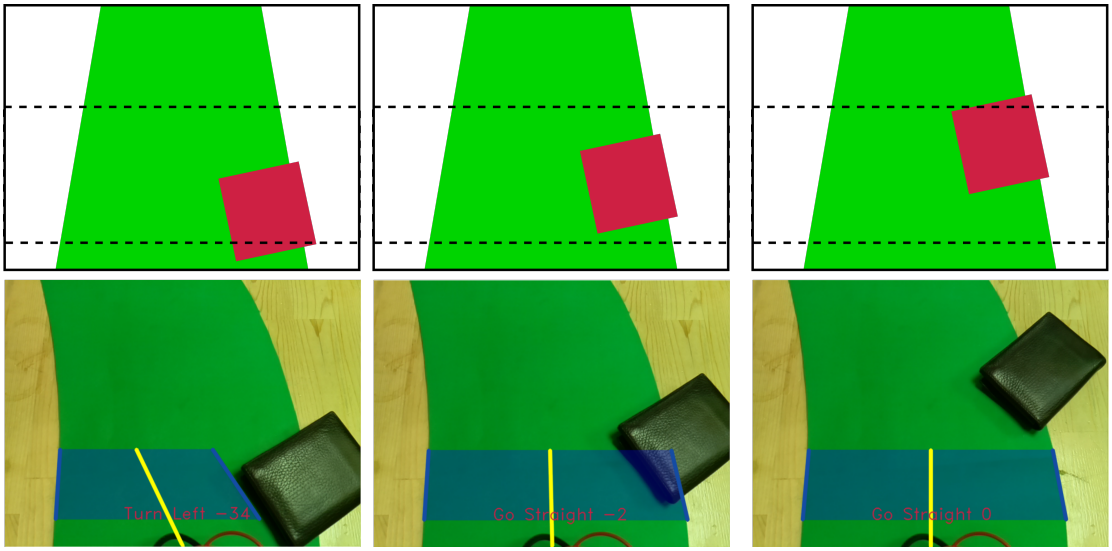
The tests asses the ability and performance of the subsystem. Numerous tests are made. The first set of tests cover the robustness of the subsystem by placing an obstacle of the subsystem. The test scenario and its results are presented in *Figure 16,17 and 18*. It can be seen that the algorithm ignores 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 19 and 20*. 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 21, 22 and 23*. 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.

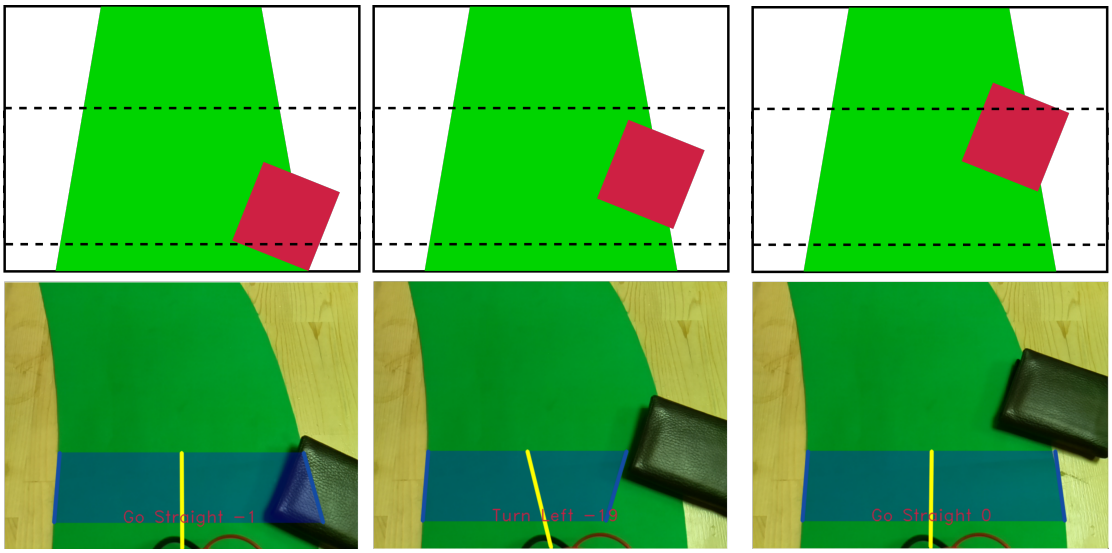
Table 1: Statistical Stability Results

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



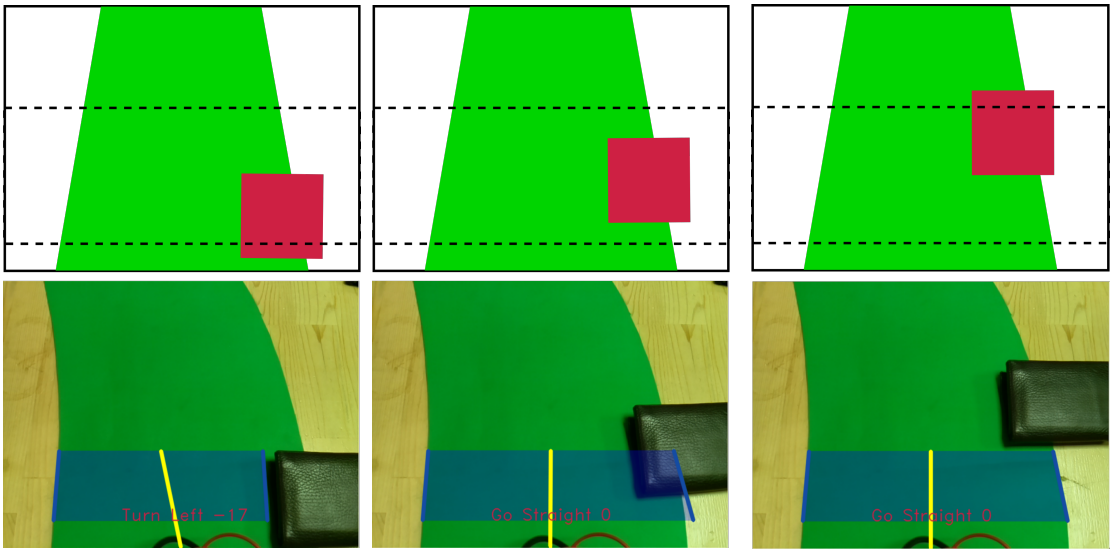
(a) Obstacle is at the Beginning of the Path (b) Obstacle is at the Middle of the Path (c) Obstacle is at the End of the Path.

Figure 16: A Test Scenario: Downward Inclined Obstacle on the Path.
Upper Half: Proposed Tests, Lower Half: Results



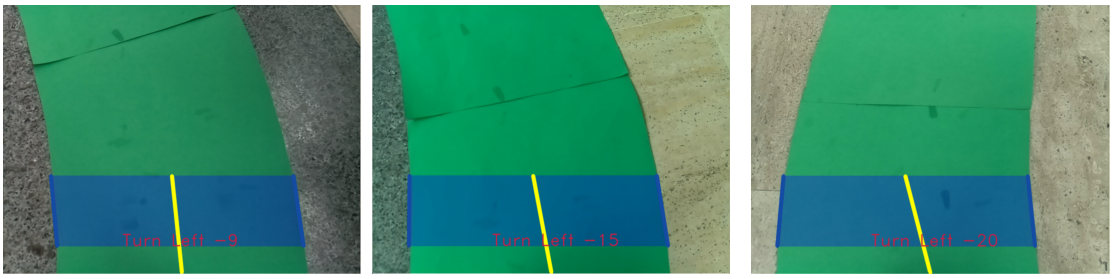
(a) Obstacle is at the Beginning of the Path (b) Obstacle is at the Middle of the Path (c) Obstacle is at the End of the Path.

Figure 17: A Test Scenario: Upward Inclined Obstacle on the Path.
Upper Half: Proposed Tests, Lower Half: Results



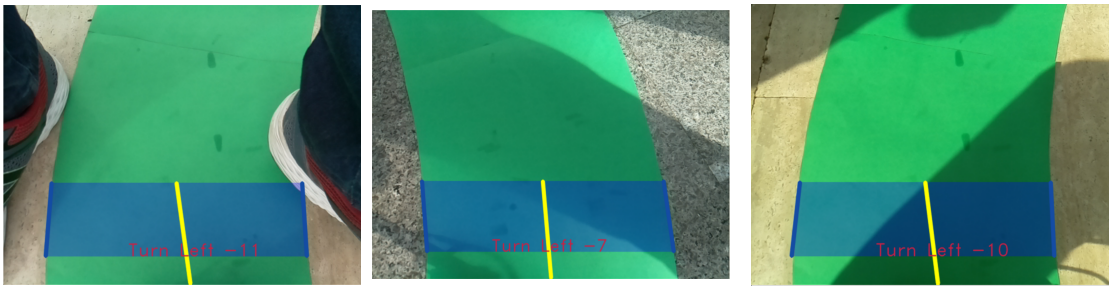
(a) Obstacle is at the Beginning of the Path (b) Obstacle is at the Middle of the Path (c) Obstacle is at the End of the Path.

Figure 18: 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 19: 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 20: A Test Scenario Results: KKM Outdoor Path Detection

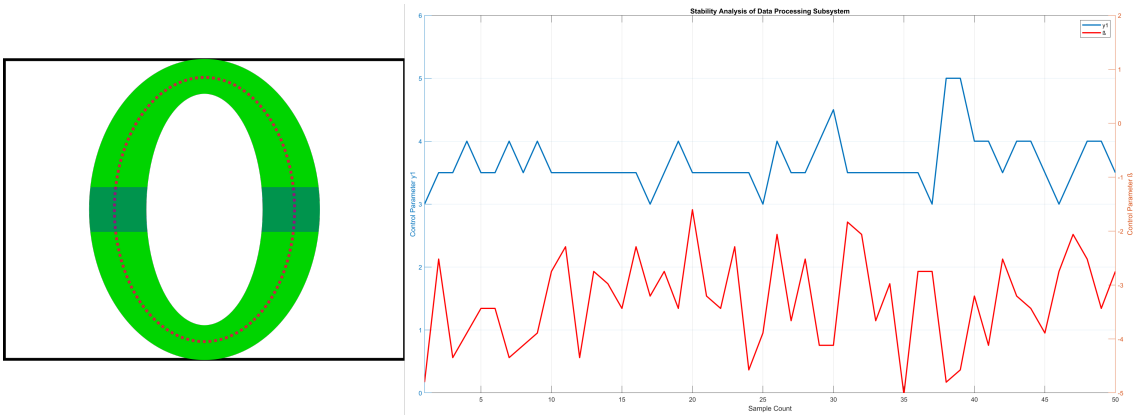


Figure 21: Left: The Vehicle is placed on Straight Area.
Right: The Control Outputs

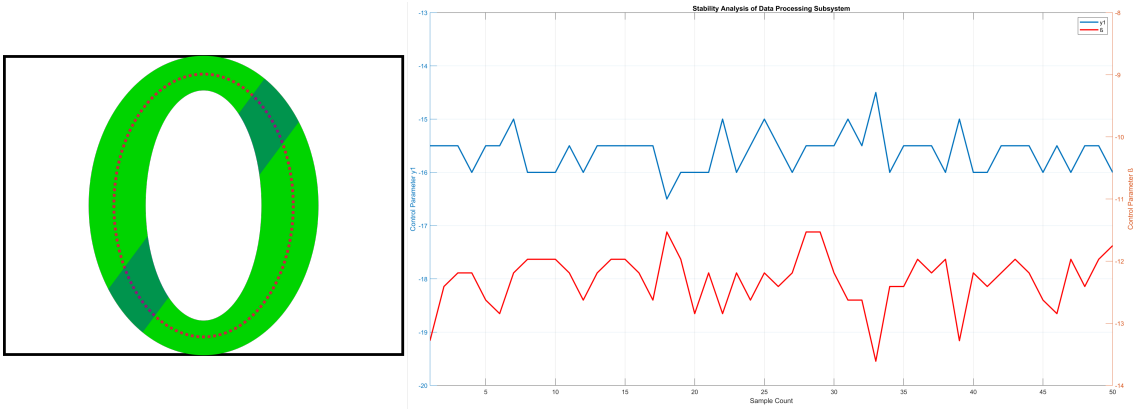


Figure 22: Left: The Vehicle is placed on Bended Area.
Right: The Control Outputs

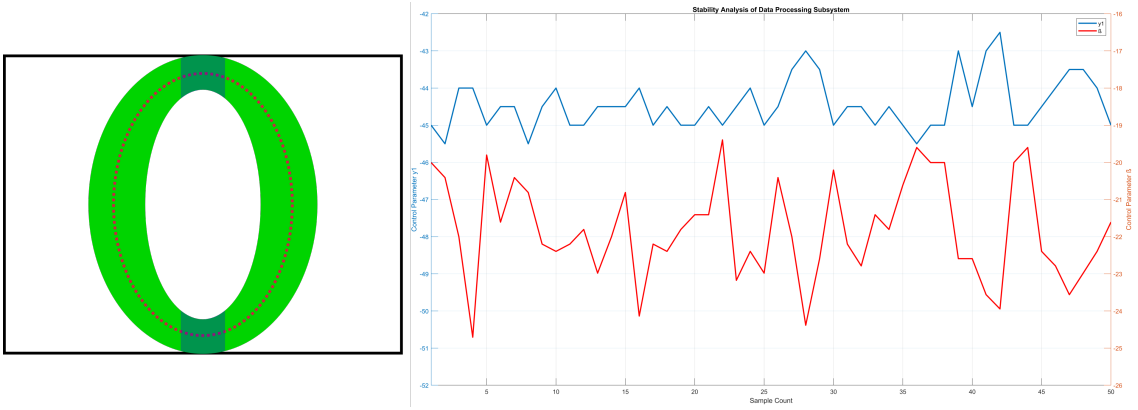


Figure 23: Left: The Vehicle is placed on Curved Area.
Right: The Control Outputs

4.4 PID Controller Subsystem Tests and Results

1. PID Parameters Test for Given Input:

   (a) Connect the Vehicle Motors to Motor Controller

   (b) Connect the Motor Driver to Arduino

   (c) Give the angle value that the subsystem should compensate

   (d) Give the power to the motors

   (e) Observe the behaviour of the vehicle

   (f) If the vehicle rotates with an angle given in step 3 without any feedback given, the result of the test can be considered as success.

2. Bump Test for Distance Control:

   (a) Set-up a lane as in *Figure 24*.

   (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.

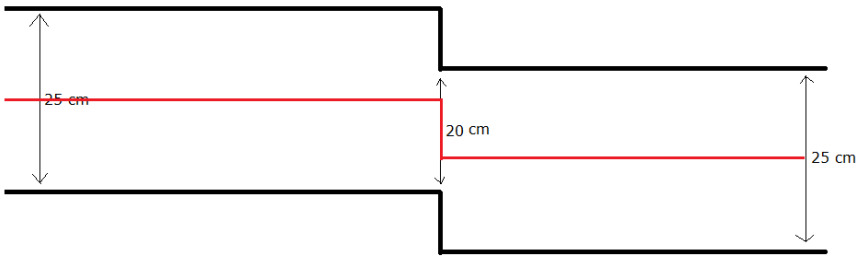


Figure 24: Bump Test for Distance Control

3. Bump Test for Angle Control:

   (a) Set-up a lane as in *Figure 25*.

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

4. Path Tracking Test:

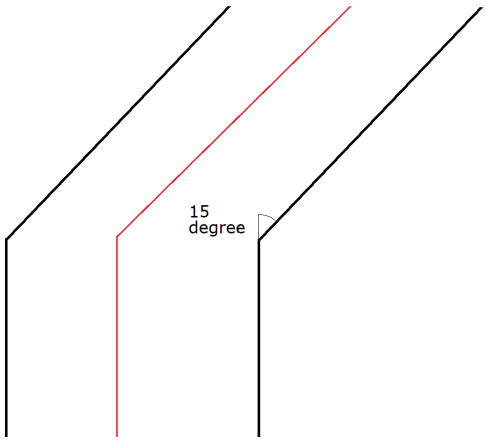


Figure 25: Bump Test for Angle Control

(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 smoothly, the result of the test can be considered as success.

5. 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 a reasonable time (in less than 2 seconds), the result of the test can be considered as success.

6. 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 a reasonable time (in less than 2 seconds), the result of the test can be considered as success.

7. Results of PID Controller Subsystem Tests

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) Send random text data to Arduino.

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

   (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.

2. Results of Internal Communication Subsystem Tests

4.6 External Communication Subsystem Tests and Results

1. Raspberry Pi as Client Test:

   (a) Create a hotspot from the computer

   (b) Connect the Raspberry Pi to the hotspot

   (c) Modify the client code to be tested according to IP address of the computer

   (d) Run the server code from computer

   (e) Run the client code from the Raspberry Pi

   (f) Try the possible combinations from the terminals of both sides

   (g) The test result can be considered as success if both sides respond according to the *Handshake Protocol*.

2. Raspberry Pi as Server Test:

   (a) Create a hotspot from Raspberry Pi.

   (b) Connect the computer to the hotspot

   (c) Modify the client code to be tested according to IP address of the Raspberry Pi.

   (d) Run the server code from Raspberry Pi.

   (e) Run the client code from the computer.

   (f) Try the possible combinations from the terminals of both sides

   (g) The test result can be considered as success if both sides respond according to the *Handshake Protocol*.

3. Results of External Communication Subsystem Tests

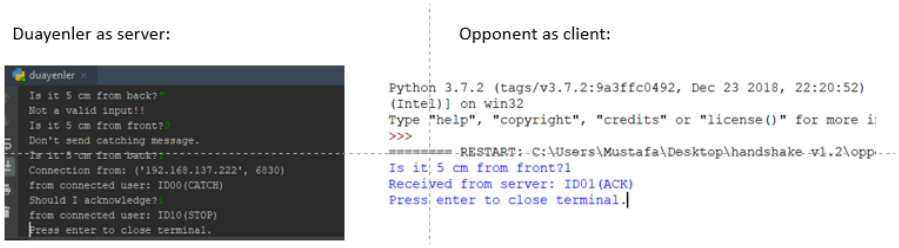


Figure 26: External Communication Subsystem Test Result

Figure 27: External Communication Subsystem Test Result

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

3. 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.

4.8 Speed Subsystem Tests and Results

1. Determination of Base Speed:

   (a) Set β value that is supplied to *Speed Subsystem* equal to zero.
   (b) Give PWM Base as 255 RPM and observe the vehicle on the path
   (c) Decrease the PWM Value by 10 PWM and observe the vehicle.
   (d) Repeat the step 3 until the desired base speed value is observed.
   (e) Record this value.

2. Determination of Constant K_1:

   (a) Use the PWM Base value determined at the *Test 1*
   (b) Set β value that is supplied to *Speed Subsystem* equal to ten degree.
   (c) Set K_1 value equal to one and observe the vehicle on the path.
   (d) Increase the coefficient K_1 and observe the behaviour of the vehicle on the path.
   (e) Repeat step 4 until the desired coefficient K_1 is determined.
   (f) Record this value.

3. Results of Speed Subsystem Tests

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 behaibour
   (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.

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

2. 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

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
2. Results of Structure System Tests

5 Other Considerations

5.1 Cost Analysis

Table 2: Cost Analysis for the Project

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

real spend-ings

5.2 Power Analysis

Table 3: Estimated Cost Analysis for the Project

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

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

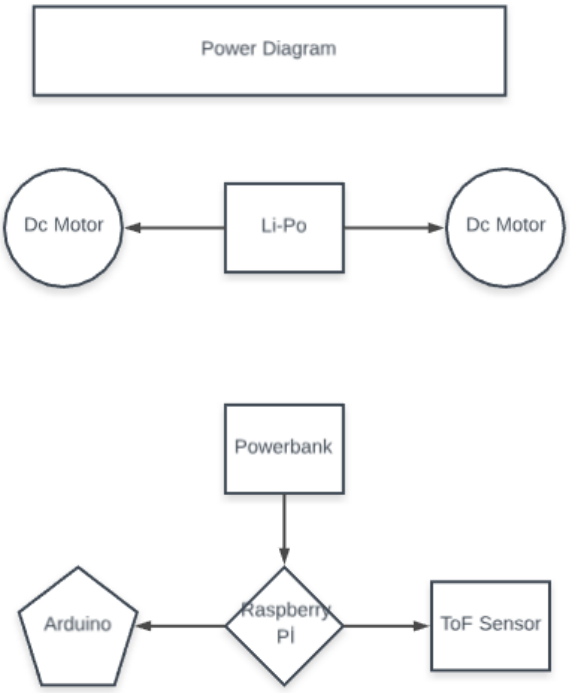


Figure 28: Electrical Architecture of the Project

6 Deliverables

7 Budget

THIS PART IS NEW FOR FI-NAL RE-PORT.

THIS PART IS

7.1 Actual Expenditures

7.2 Total Cost

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

8.2 Application Areas

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 PART IS NEW FOR FINAL REPORT.

THIS PART IS NEW FOR FINAL REPORT.

THIS PART IS NEW FOR FINAL REPORT.

THIS PART IS NEW FOR FINAL REPORT.

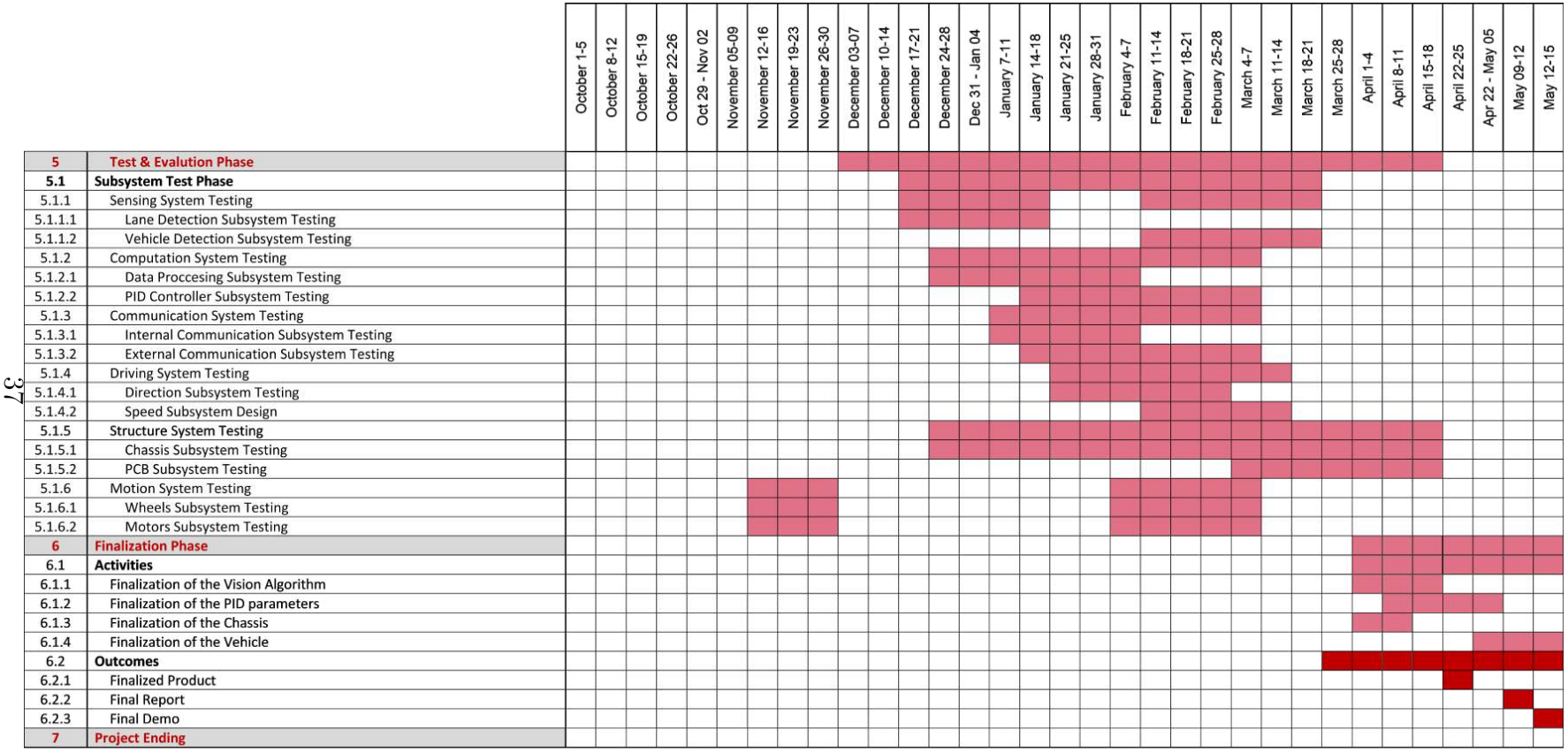
THIS PART IS NEW FOR FINAL REPORT.

THIS PART IS NEW

A Gantt Chart

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

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


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