Middle East Technical University

Department of
Electrical and Electronics Engineering

EE494 ENGINEERING DESIGN II

Car Chasing Robot
Critical Design Review 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

March 08, 2019

Contents

1 Introduction

2 Overall System

The main objective of this project is to design and produce a self driving mini-car that can follow a path with a varying properties with in a 200 dollar budget. The project can be investigated under six main systems and their twelve subsystems. *Figure 1* shows the organization structure of the project.

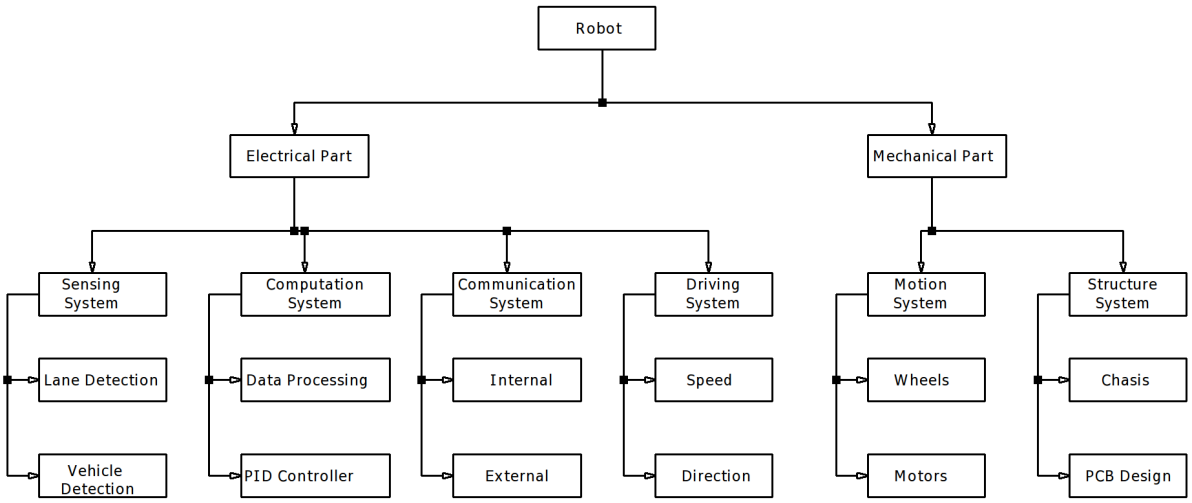


Figure 1: Organization of the Project

V-Model is a very popular tool for system engineers to plan their projects. To ease the project tracking process, the V-Model was constructed by the DUAYENLER. The overall look of the V-Model can be seen at *Figure 2*. This section includes the explanation, requirements, test procedures and test results for the subsystems. In this section, the requirement for the systems and subsystems will be given. Moreover, the solution approaches for the subsystems will be explained in detail. Lastly, the solutions will be assessed by their performance according to their requirements. The test procedures and the discussion on the test results will be discussed in the preceding sections.

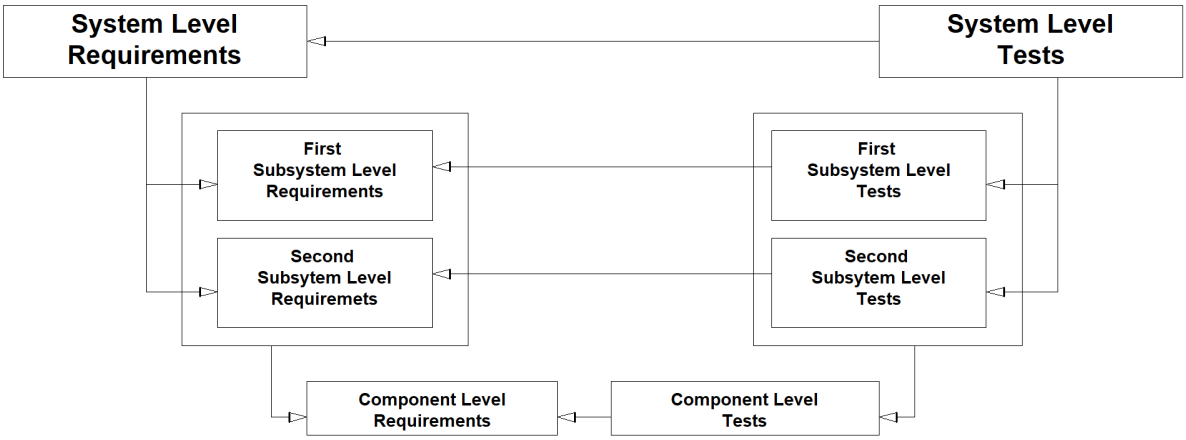


Figure 2: V-Model

Are they done?

justification for requirements

Compliance with Reqs

2.1 Sensing System

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

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

The system has two subsystems namely,

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

2.1.1 Lane Detection Subsystem

1. Requirements for the Solution
   (a) The subsystem should be able to detect only the shades of green color
   (b) The subsystem should be able to detect edges in the camera frame in any light condition

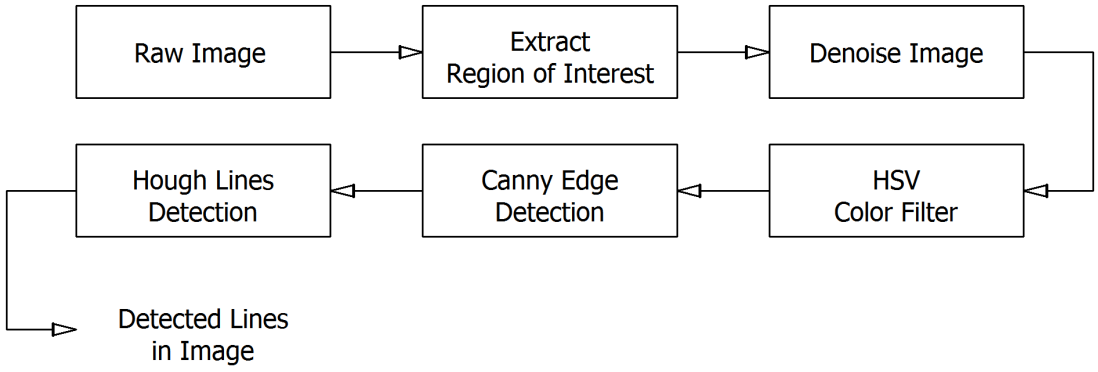


Figure 3: Block Diagram of the Lane Detection Subsystem

2. 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 input to this subsystem is provided by Raspberry Pi camera mounted on top of

the vehicle. The proposed solution first masks out a region of interest (ROI) of $480x250px$. The masking eliminates the process of excessive data and increases the process speed of the pipeline. The explanations and visual of the ROI is provided in *Section 2.2.1, Figure 5*. Then, 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 colors 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 (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. The block diagram of the subsystem is given in *Figure 3*.

3. Discussions on the Solution

   The main structure of the proposed solution has not changed since Conceptual Design Report. An addition to process pipeline is extracting a ROI. This is done to both improve the performance and remove possible distractions that are present in the image. A point to note in this algorithm is Hough line detector. It is a probabilistic function, meaning that even if the captured frame is the same as previous frame, the found line coordinates may differentiate a bit. However, this is not a big issue. One weak point was the lane detection under extreme lighting conditions. The filter is adjusted to be able to detect in bright lighting conditions but dark lighting is problematic. To solve this issue, the team decided to place LED strips to front bumper of the vehicle.

2.1.2 Vehicle Detection Subsystem

1. Requirements for the Solution

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

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

   (c) The subsystem should trigger the handshake protocol

2. Solution for the Subsystem

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

3. Discussions on the Solution

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

2.2 Computation System

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

1. The system should be able to produce middle line to follow
2. The system should be able to control the robot

The system has two subsystems namely,

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

2.2.1 Data Processing Subsystem

1. Requirements for the Solution

   (a) The subsystem should be able to analyze data produced by sensing system

   (b) The subsystem should be able to produce the angle information required by the controller subsystem

   (c) The subsystem should be able to work on Raspberry Pi

   (d) The subsystem should be able to process one frame at most in 100 milliseconds

   (e) The subsystem should be able to tell differences between disturbances and lane

   (f) The subsystem should be able to interpret the middle of the lane if both sides are present at the frame

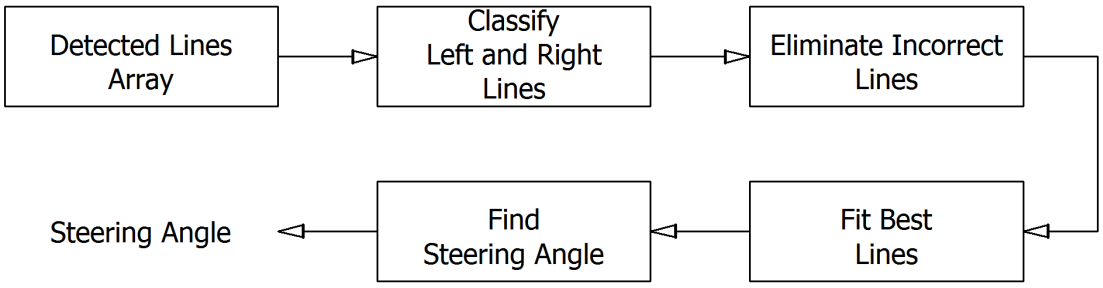


Figure 4: Block Diagram of the Data Processing Subsystem

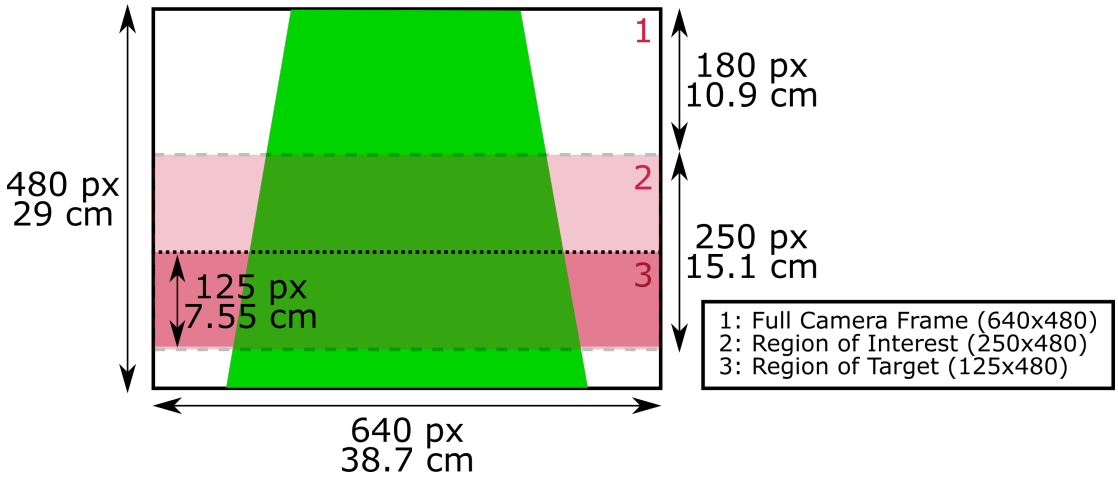


Figure 5: Block Diagram of the Lane Detection Subsystem

2. Solution for the Subsystem

The task of this subsystem is to determine the steering angle of the vehicle so that the vehicle can steer to the target point. The input of the subsystem is the coordinate array produced by Lane Detection subsystem. The input is processed by a detailed algorithm and the output is steering angle. The processing is done by taking into consideration the reference frame. The reference frame is shown in *Figure 5*. This figure summarizes the three main frames. The Lane Detection subsystem extracted ROI out of full camera frame. The Data Processing subsystem processes the data of ROI but produces output for the Region of Target (ROT). The reason is, ROI is good to determine possible obstacles on the path but steering to a target point that is almost 15 cm away from the vehicle is not realistic. For this reason ROI is not used for determining steering angle, instead ROT is used.

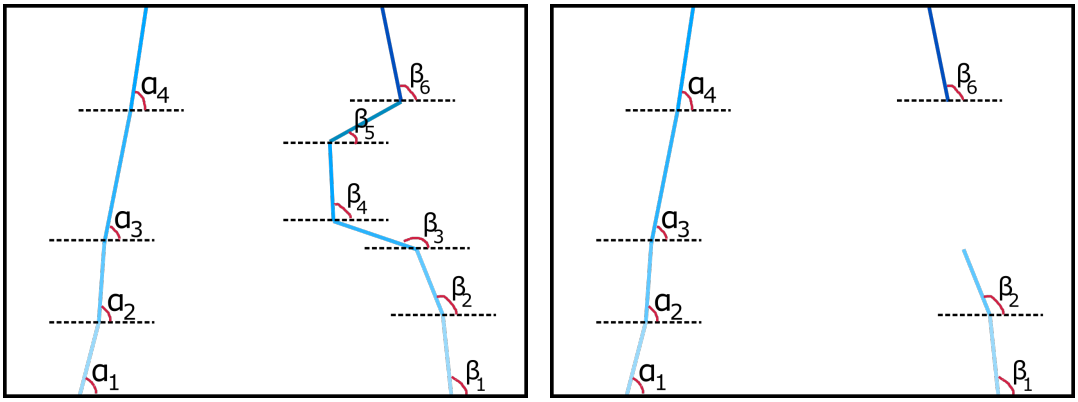
There are four main steps to determine the steering angle. The first step is to classify the lines as left and right. The second step is to eliminate the possible incorrect lines, if any. The third step is to fit the best lines through the left and right lines and reduce the total number of lines to two that are left and right lines. The last step is to find steering angle by connecting 320th horizontal pixel with the average end points of the lane lines. The block diagram of the subsystem is given in *Figure 4*.

Classifying a line is done in two steps. First, lines are divided into two whether they start from right or left of horizontal 320th pixel. Then, the center of both sets are found. Eventually, according to the average of both centers, lines are classified as left or right. This process is summarized in *Algorithm 1.*

Algorithm 1: Classifying Lines as Left and Right

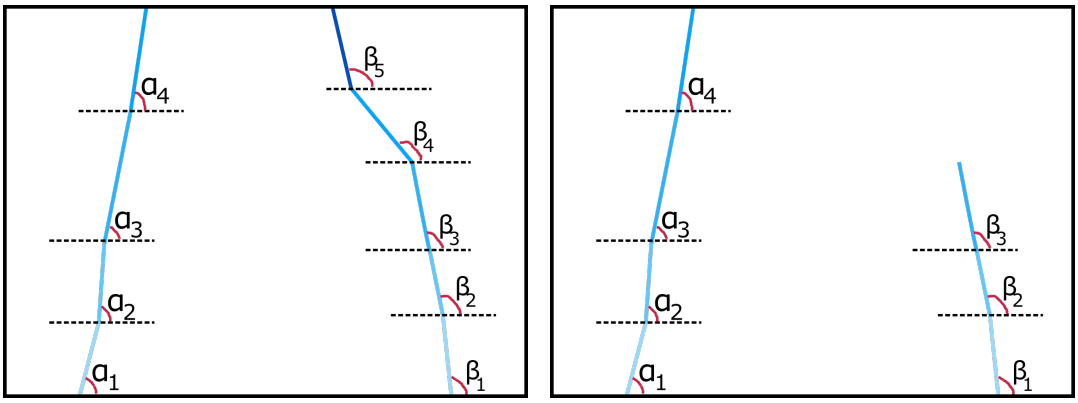
```
for All Lane Lines do
    if Starting Point > 320 then
        Temporarily classify as right line
    else
        Temporarily classify as left line
r_center = center of temporary right lines
l_center = center of temporary left lines
lane_center = (r_center + l_center)/2
for All Lane Lines do
    if Starting Point > lane_center then
        Classify as right line
    else
        Classify as left line
```

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 6* 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 6b.* 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 7.* 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 7b.* 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.



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

Figure 6: A Sample Scenario on Eliminating Incorrect Lines



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

Figure 7: Another Scenario on Eliminating Incorrect Lines

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. Then next step is to classify the line points as left or right. The lines are firstly eliminated according to their slopes. If slope of a line is not in invalid slope region of ± 0.005, then it is a valid line. This process is done to get rid of unnecessary low sloped lines. After elimination, line points set must be determined as left or right. At this point, this classification is done according to double checking. The center of the image, that is 320th vertical pixel. If initial and final horizontal points of the image is in the same half, the line belongs to that half. This method works nicely in most regions of the path. However, it is a bit error prone in case of a sharp turning angle or losing one of the lanes . This algorithm will be improved to be more robust.

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

YANLIŞ BİLGİ VAR BURADA KİM YAZDI BUNLARI

from *Figure 5*) $480px - 50px - 125px$. The current point of the vehicle is always $(320, 480)$. So the line connecting two points to each other constitutes the track path and the $arctan$ of the slope gives the steering angle. Steering angle is in the $[-90, 90]$ range where negative values indicate to turn left and positive values indicate to turn right. This output is sent to PID Controller subsystem.

3. Discussions on the Solution

   The proposed algorithm is different from the one presented in Conceptual Design Report. The reason is that the old algorithm didn't have any obstacle rejection feature and it had problems in determining steering angle. For this reason, new algorithm is developed from scratch. The proposed algorithm works nice as test results are discussed in *Section 4.1*. The obstacles are most of the time are detected and ignored. One weak point is that, the algorithm may not be able to ignore some obstacles in certain inclinations. That is, as algorithm is based on slope evaluation, certain obstacle placements may fool the algorithm.

2.2.2 PID Controller Subsystem

1. Requirements for the Solution

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

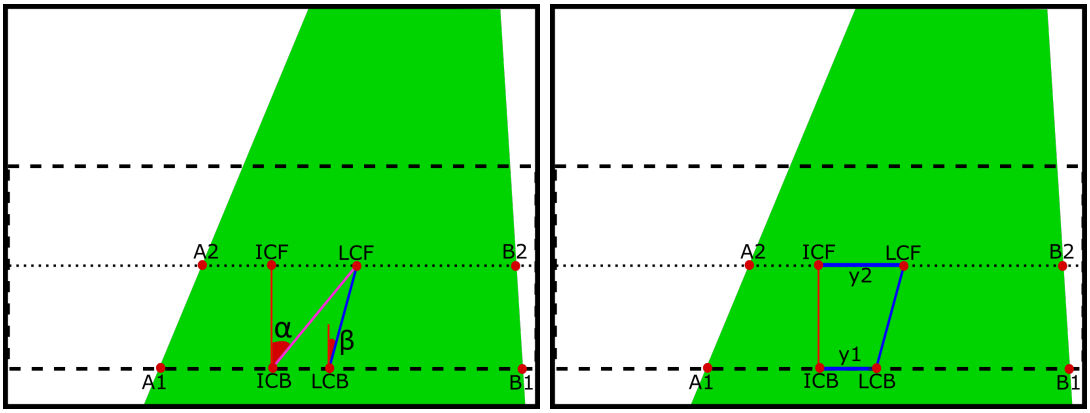
   (b) The subsystem should be able to react the external disturbances

2. Solution for the Subsystem

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

   For that purpose, the *Data Processing Subsystem* produces the necessary feedback elements for this subsystem. For the control purpose, in ideal circumstances data processing unit determines eight main point on its vision to create processed variables as in *Figure 8*. 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 8: Controlled Variables of the System

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

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

Modelling the Plant

Modelling a plant is a good practice in controller design applications, however, in our case the model for the vehicle is unstable, thus applying a bump test as in *Figure 9* results with a exponentially increasing processed data 'y2'. Thus, in this project, our aim is to apply bump test to closed loop system as in *Figure 10* with a known P-controller. An approximate plant model from there can be found as follows;

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

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

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

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

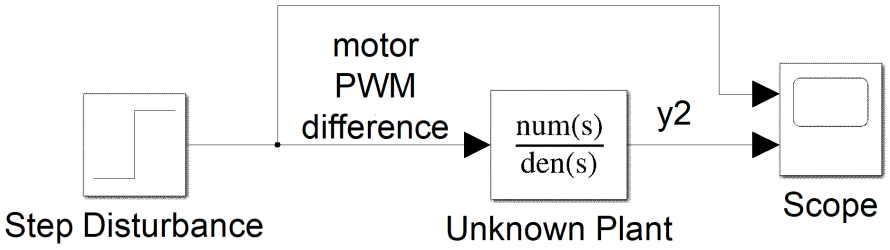


Figure 9: Bump Test for the Unknown Plant

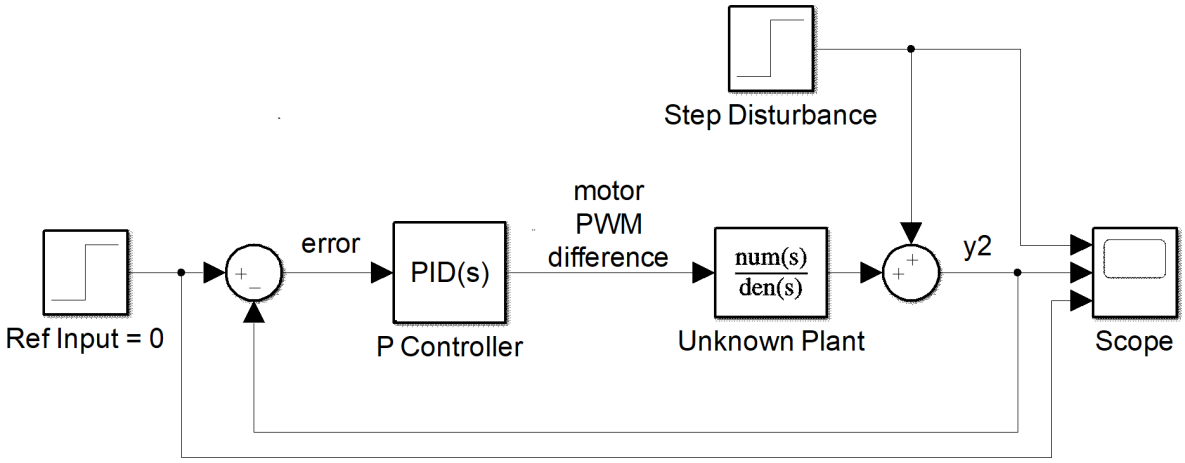


Figure 10: Bump Test for the Closed Loop System

Design & Implementation of the Controller

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

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

This is a transfer function that accepts the error signal as its input as in *Figure 10*. Since the reference input is always zero for our case, in other words, it is desired that the variables α and $y2$ is equal to zero for all time instants. Therefore, for our case, the error is equal to the negative version of controlled variable.

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

3. Discussions on the Solution

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

2.3 Communication System

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

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.

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

```
import serial
ar=serial.Serial("/dev/ttyUSB0",9600)
```

Script 1: Serial object decleration in Python

```
#include "./arduinoModule/rs232.h"
int cport_nr = 24; /* /dev/ttyUSB0 */
int bdrate = 9600; /* 9600 baud */
char mode[] = { '8', 'N', '1', 0 }; // 8 data bits, no parity,
    1 stop bit
char str_send[2][BUF_SIZE]; // send data buffer
unsigned char str_recv[BUF_SIZE]; // recv data buffer
RS232_OpenComport(cport_nr, bdrate, mode);
```

Script 2: Serial communication setup in C

There are several serial communication protocols that can be used to maintain the connection such as SPI, I2C. However, the first choice is to use USB serial port of the Arduino. Since RPi is practically a computer, it can recognize Arduino as a device using a serial port such as /ttyUSB0 in case of a Linux based OS. When recognized, RPi can send any piece of strings to the Arduino via USB cable. The process of communication is as follows:

(a) Arduino should be connected to the Pi.
(b) Using Arduino IDE or any other method such as listing serial ports and checking for Arduino and so on, the serial port name should be detected
(c) Baud rates of two sides should be the same. 9600 is generally enough but if needed, it can be incremented to satisfy fast communication
(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 any language C++ or Python, messages to serial port can be send

There are minor differences when implementing the code in Python, C++ and C. Python is the most practical one:

Python has a library called serial by which any type of data can be send through serial ports. Script 1 is used to declare a serial object. Then using `ar.write("some`

`string \r".encode())`, the string "some string" can be send to Arduino. Note that "\r" carriage return character carries a high importance because it shows that a string is terminated and any other incoming data belongs to the new piece of string.

As alternatives, the implementation on C is also examined. Sample codes to implement the same communication in C is in Script 2. Since it is a low level language, specification of buffer size and other parameters should be done in the code.

On Arduino side, there are also several option that we can read the incoming data. Using `Serial.read()` command is one of the simplest solutions. However, it contains some issues like conversion from string to integer and when to stop. Furthermore, the incoming data should be considered in character basis for the exact control.

Alternative Solutions for Internal Communication Subsystem

The communication between Raspberry Pi and Arduino can be realized by the use of other peripheral protocols or libraries.

There is a `SerialCommand.h` library for the Arduino which allows executing a function depending on the incoming string. Using .addCommand("str",func) of the library any function can be associated with any string coming from serial port. Moreover, the functions can have argument. For example, let the string "PWMSET" be execute a function setpwm() but the PWM value is required. If incoming string is of the form "PWMSET 150", using .next() function of the library, the value 150 can be read and converted into integer and interpreted as the PWM value to be set.

Another solution is to use I2C or SPI protocols to set communication between Arduino and Raspberry Pi. These options require simple wiring connections between the pins. A point to remark is being cautious to operating voltage levels of pins of Arduino and Raspberry Pi.

Besides aforementioned alternatives, Wi-Fi communication can also be considered between Arduino and Raspberry Pi. This solution requires deployment of a Wi-Fi module on Arduino. Some hardware exists in market such as ESP8266 and EMW3165. This solution would require a Wi-Fi network in the medium which makes the solution less implementable.

A last solution would be to remove Arduino from the system. This would require the handover of operations of Arduino to Raspberry Pi. The resulting system could be considered as a single board solution, that is, all relevant subsystems operate

on Raspberry Pi. This option would cause heating problems on Raspberry Pi. A cooling mechanism should also be applicable in this case.

Communication subsystem enables the robot to communicate with the opponent using the handshake protocol agreed on standard committee. According to the standard committee, Wi-Fi modules must be used to implement handshaking. Since Raspberry Pi was used in the project, there is no need to get a separate Wi-Fi module; the internal Wi-Fi module of the Raspberry Pi was used.

3. Discussions on the Solution

2.3.2 External Communication Subsystem

1. Requirements for the Solution

2. Solution for the Subsystem

   Socket programming is an effective tool to implement client-server communication algorithms. It can be implemented in Python or C++. Our algorithms are written in Python for now, yet it can easily be converted to C++ if the team members decide that it is necessary. The algorithms for client and server sides are slightly different. *Figure* **??** shows the functions that are used for client and server sides to create communication between client and server.

   Here is the summary of the key functions from socket library:

   - socket.socket(): Creates a new socket using the given address family, socket type and protocol number.
   - s.bind(address): Binds the socket to the address defined previously.
   - s.listen(backlog): Sets up the maximum number of connections that can be made to the socket, which must be at 1 for the project.
   - s.accept(): Waits until connection arrives, then accept the client connection. Returns the client socket connected to the server as (conn, address) pair, where conn is a new socket object and address is the address bound to this socket
   - s.connect(): Provides client to connect to the server
   - s.send(): Transmits message to the remote socket.
   - s.recv(): Receives message from the remote socket
   - socket.close(): closes the socket; i.e., ends the communication with the opponent at the end of the race.

   It is stated in the standard committee that each team must be assigned a static IP to communicate with the other robots. Duayenler has the static IP stated as "192.168.1.7" and the ID as "07". Since Raspberry Pi 3 comes with a built-in wireless adapter, configuring it as a Wi-Fi hotspot is possible. To assign given IP

to the robot, Raspberry Pi must be set as an access point from the terminal.

In the algorithm that was implemented for the handshake, in a continuous loop, the front and rear sensors' values are been checked. There are two functions which are for client and server modes, respectively. If the front sensor senses the opponent in 5 cm range, our main code visits the client mode function. If the rear sensor senses the opponent in 5 cm range, server mode function runs. If our robot is in the server mode, the rear sensor value is again checked. The acknowledge message ($< ID > 01$) or reject message ($< ID > 11$) is sent according to the sensor value.

3. Discussions on the Solution

2.4 Driving System

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

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.

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

   - PWM Offset Value that determines the speed of the vehicle at longitudinal movement. This data is acquired from the **Speed Subsystem.**

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

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

3. Discussions on the Solution

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

2.4.2 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

   The best place to implement the state machine is Arduino as motor driving is also controlled by it. Main requirement for this system to operate is a measure of error or success for the direction subsystem. It can be the same as the error input of controller i.e. turning angle or a function of it. The state machine will act depending on the value of the error input. When it is above a critical level, the vehicle will show a steep deceleration to compensate the error of the direction unit. In other cases, It is wise to implement the speed controller in the form of at least PD controller. In other words, the change in the overall speed will also be maintained by a controller whose error input is not necessarily tried to be made zero but rather below a pre-specified level. State machine diagram can be seen in *Figure* **??**

   This unit acts as a complementary module for direction unit. It will act as a state machine. In one state, the unit will try to increase the speed of the vehicle by making overall increase in both PWM values of DC motors. The feedback of this system will be the cost function mentioned in driving unit. If that cost exceeds a specified level, unit goes to another state in which the unit will decrease the overall speed to allow direction unit to operate more correctly. In short, this unit tries to compensate the error of the direction unit by changing the overall speed of the vehicle. The requirements of this subsystem are listed below:

3. Discussions on the Solution

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

line_angle-base_speed

2.5 Motion System

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

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

2.5.1 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

   As the previous suggestion in CDR, 2+1 combination (2 wheel with power and 1 caster ball) is preferred due to easier implementation and control. Although this placement weaker in balance and obstacle handling, importance of easier implementation and control are considered more beneficial. While choosing wheels, high friction property is considered. Because of this reason, super soft and slick tire are chosen with lighten aluminum rim. Besides, larger width is preferred to increase hanging on the lane.
3. Discussions on the Solution

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

2.5.2 Motors Subsystem

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

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

3. Discussions on the Solution

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

2.6 Structure System

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

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.

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

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

   To satisfy the requirements the holder is built using 4mm plexiglass. The choice satisfies the rigidity and light weight possible. A thinner one would result in less rigidity and increased vibration on the system. The designed structure, whose layout can be seen in *Figure* **??**, 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* **??**). After integration, the view of the camera can completely cover the both edges of the path (*see Figure* **??**)

   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. The chassis is supposed to be light and strong because

of the competition purposes. However, it should balance the robot to be able to handle turns. The requirements of this subsystem are listed below:

Current version of chassis which were used in critical module demo can be seen at *Figure* **??**.

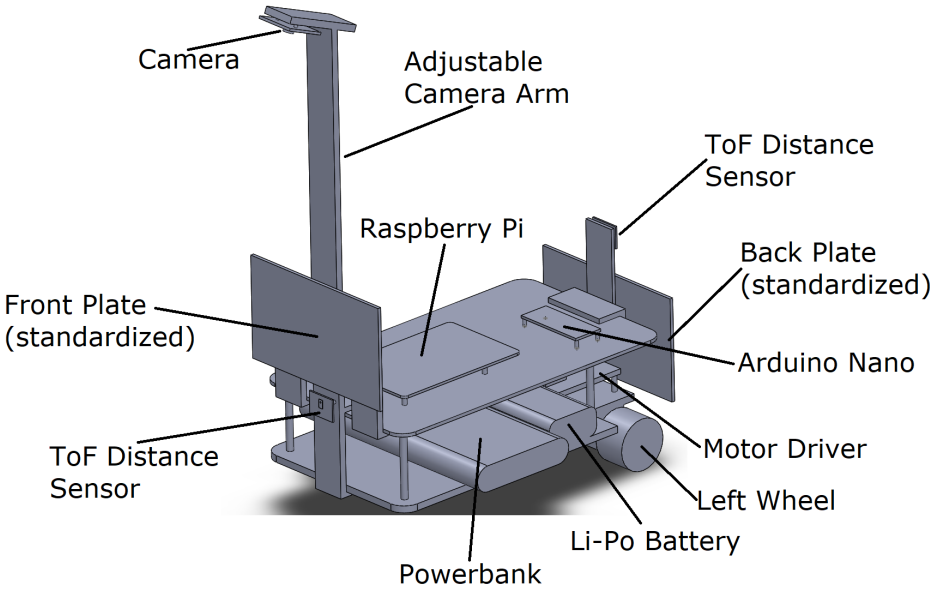


Figure 11: Isometric view of the 3D Drawing of the Vehicle

choose one

3. Discussions on the Solution

2.6.2 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. The requirements of this subsystem are listed below:

   This subsystem aims to make all the circuit connections rigid and compact. Currently, there is wire connections between Arduino-Motor driver and Arduino-RPi. However, addition of vehicle detection sensors and other lane detection alternatives will increase the amount of components, hence, wires. In addition, to use

the space occupied by the Arduino UNO board, Arduino Mini can be used. This also allows to build the circuit board as shield for Arduino Mini. After that any other sensors and connections can be made through PCB. In other words, PCB acts as a breakout board for each item integrated to the system in a more rigid and compact way.

3. Discussions on the Solution

2.7 Compatibility of the Subsystems

The block diagram showing the interaction of the subsytem block with each other can be seen from the *Figure 12*. 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.

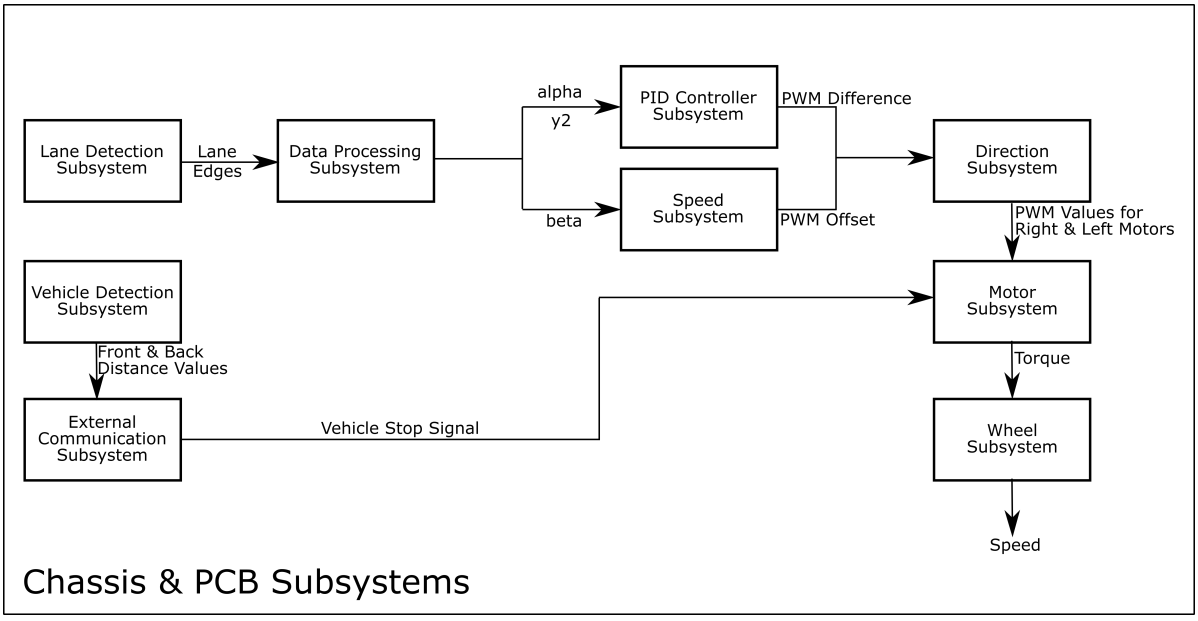


Figure 12: Block Diagram of the Project and the Interaction of the Subsystems

3 Detailed Tests for the Subsystems

1. Lane Detection Subsystem Tests

   (a) Light Condition Test

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

   (b) Visual Disturbance Test

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

2. Vehicle Detection Subsystem Tests

   (a) Front Vehicle Detection Test in Closed Environment:

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

   (b) Rear Vehicle Detection Test in Closed Environment:

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

   (c) Angled Approach Test:

      i. Make the connection of the desired sensor and Arduino properly
      ii. Hold the sensor at an angle of 90 degree with respect to ground
      iii. Place the test object 5 cm in front of the sensor with 30 degree angle with respect to the sensor
      iv. Observe the output of the subsystem

v. Repeat the step 3 & 4 with different distance and angle values

vi. If the output of the subsystem generates logical positive for distances smaller than 5 cm for all angle values with respect to sensor and logical zero for distances greater than 5 cm, the test result can be considered as success

(d) Vehicle Detection in Different Sunlight Conditions Test:

i. Repeat the test steps of the *Front Vehicle Detection Test in Closed Environment* in CCC (Cultural and Convention) ground under direct sunlight

ii. Repeat step 1 in CCC (Cultural and Convention) under artificial light, in other words, under no direct sunlight conditions

iii. Repeat steps 1 & 2 for different locations of E Building including Graduation Laboratory

iv. If the output of the subsystem generates logical positive for distances smaller than 5 cm under all light conditions and logical zero for distances greater than 5 cm, the test result can be considered as success

3. Data Processing Subsystem Tests

(a) Data Assessment Test

i. Link the output of Lane Detection subsystem to Data Processing subsystem.

ii. Asses if the output coincide with physical reality of the path

4. PID Controller Subsystem Tests

(a) PID Parameters Test for Given Input:

i. Connect the Vehicle Motors to Motor Controller

ii. Connect the Motor Driver to Arduino

iii. Give the angle value that the subsystem should compensate

iv. Give the power to the motors

v. Observe the behaviour of the vehicle

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

(b) Bump Test for Distance Control:

i. Set-up a lane as in *Figure 13*.

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

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

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

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

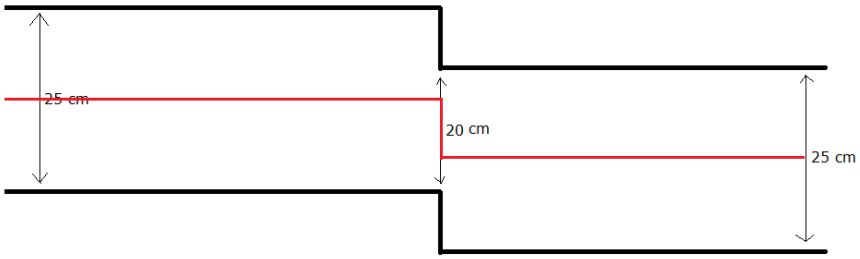


Figure 13: Bump Test for Distance Control

vi. Calculate necessary performance parameters from the plot.

(c) Bump Test for Angle Control:

i. Set-up a lane as in *Figure 14*.

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

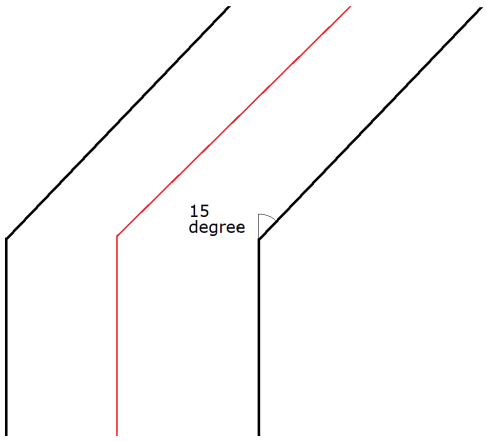


Figure 14: Bump Test for Angle Control

(d) Path Tracking Test:

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

ii. Place the vehicle to the desired empty path

iii. Observe the behaviour of the vehicle

iv. If the vehicle can follow the path smoothly, the result of the test can be considered as success.

(e) Tracking a Path with Obstacles Test:

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

ii. Place the vehicle to the desired path with obstacles

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

(f) Path Tracking Test with Physical Disturbances:

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

5. Internal Communication Subsystem Tests

(a) Data Retrieval Test

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

6. External Communication Subsystem Tests

(a) Raspberry Pi as Client Test:

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

(b) Raspberry Pi as Server Test:

i. Create a hotspot from Raspberry Pi.
ii. Connect the computer to the hotspot

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

7. Direction Subsystem Tests

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

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

8. Speed Subsystem Tests

   (a) Determination of the error input:
      i. Make all the necessary connection
      ii. Start up the vehicle
      iii. Execute lane detection and controller algorithms
      iv. Set the error input of the both controller algorithms the same
      v. Observe the behavior
      vi. Repeat the same process with a linear function of the input
      vii. Observe the success of the tracking algorithm

   (b) Determination of the critical error value:
      i. Make all the necessary connection
      ii. Start up the vehicle
      iii. Execute lane detection and controller algorithms
      iv. While the vehicle is moving, give disturbance of different types

v. Record the maximum value of the error encountered during the disturbances.
vi. Find the maximum value

9. Wheels Subsystem Tests

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

10. Motors Subsystem Tests

   (a) Torque Test:
      i. Fix the motor at horizontal position with respect to ground
      ii. Attach an object of one kilogram @@ -1070,9 +1091,10 @@
      iii. Contact the seller for more information

11. Chassis Subsystem Tests

   (a) Inertia test:
      i. Prepare a straight path
      ii. Power up the vehicle
      iii. Execute the edge detection and control algorithm
      iv. Give different type of disturbances
      v. Observe the deviation from straight line
      vi. Repeat the process with different component configurations

12. Printed Circuit Board Subsystem Tests

   (a) Short test: Aims to check all the wanted connections are present. The test procedure is as follows:
      i. Open multimeter for short circuit test
      ii. Find the ends of each routing
      iii. Check the continuity using multimeter probes
      iv. Check if there is any unwanted short circuit
      v. If exist, eliminate

4 Testing Stage

4.1 Test Results, Encountered Problems and Possible Solutions for Subsystems

Results of Lane Detection Subsystem Tests

The lane detection tests were conducted for the detection algorithm of the camera. A sample test result is shown in *Figure 15*. The tests reveal that the subsystem satisfies its requirements by detecting edges. Note that, not all lines are detected, only the lines that are in the ROI are detected as discussed in *Section 2.1.1*.

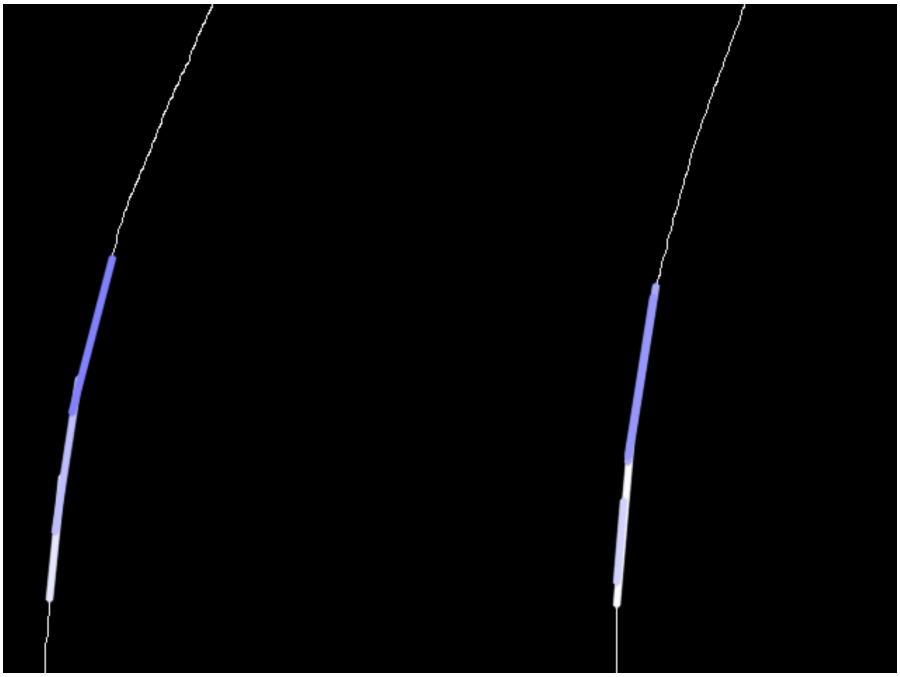


Figure 15: Lane Detection Test Result

Results of Vehicle Detection Subsystem Tests

All the test procedures mentioned at the **Section 2** were applied to the ultrasonic sensor (HC-SR04) and two infrared sensors (TCRT5000 & QRD1114). The ultrasonic sensor were showed very inaccurate result especially in *Angled Approach Test*. The test results can be examined at *Table'1*. 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 was decided that ultrasonic sensors are not a good choice for this subsystem. However, it is always possible that these sensors can be supportive sensors for the subsystem.

Table 1: The Results for the Angled Approach Test for HC-SR04

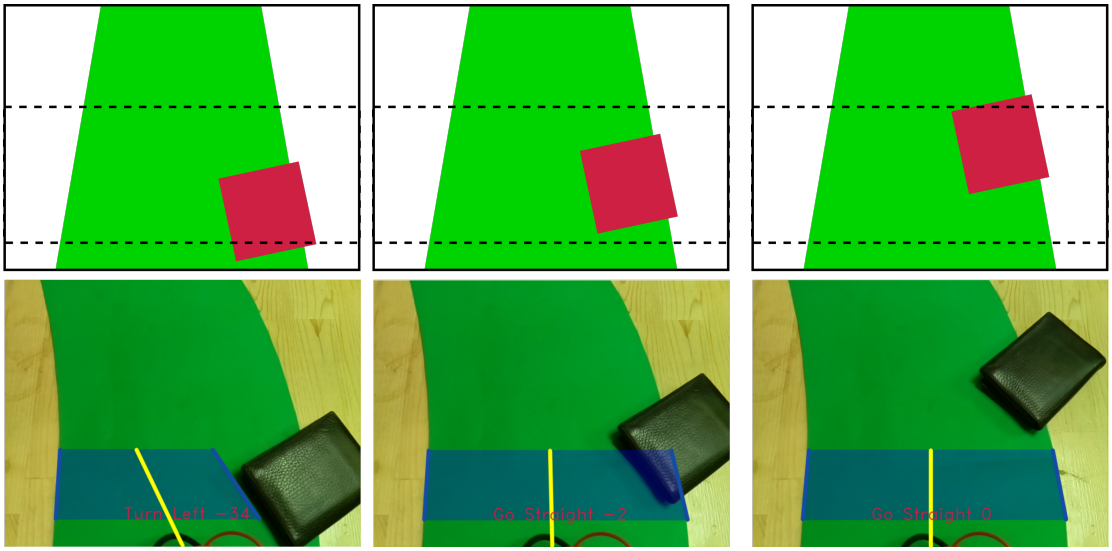
Actual Distance	The Angle	Measured Distance
3 cm	90	3.15 cm
5 cm	90	5 cm
20 cm	90	20.01 cm
40 cm	90	40.25 cm
5 cm	45	6.78 cm
20 cm	45	28.8 cm
30 cm	45	42.4 cm

Unlike ultrasonic sensors, infrared sensors showed very accurate result inside the closed environments like laboratory under artificial lights. However, the results under direct sunlight especially in CCC were not as good as expected. Thus, it was decided that the main solution should be an enhanced version of infra-red sensors namely the ones utilizing the "time-of-flight" concept. Moreover, laser sensors might be good alternative for these subsystems.

Results of Data Processing Subsystem Tests

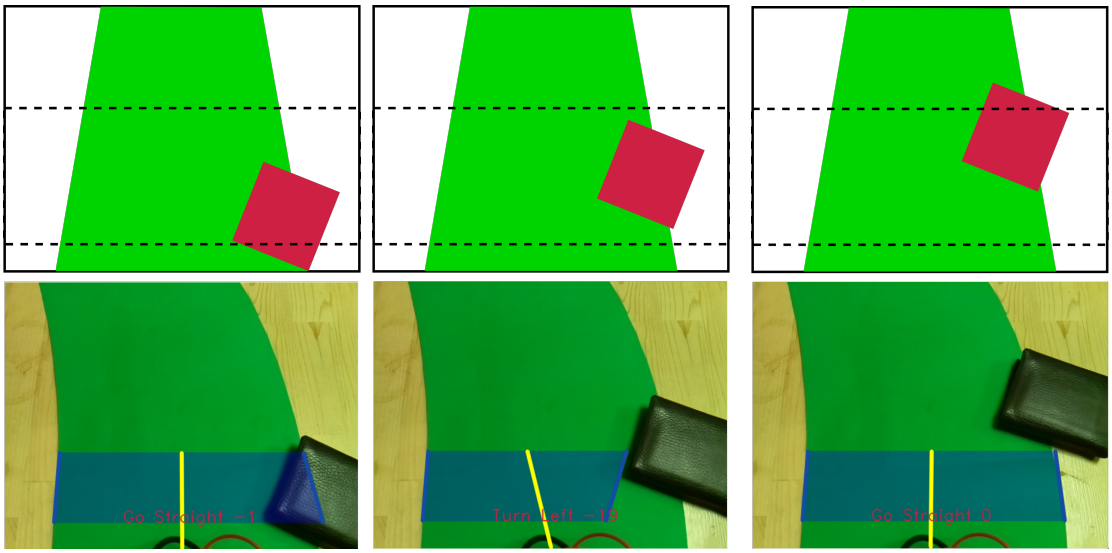
The tests in this section assesses the ability and performance of this subsystem by regarding its requirements. The number of tests realized is quite bit. The first set of tests cover the robustness of the subsystem by placing an obstacle of the subsystem. This test and its results are presented in *Figure 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.

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.



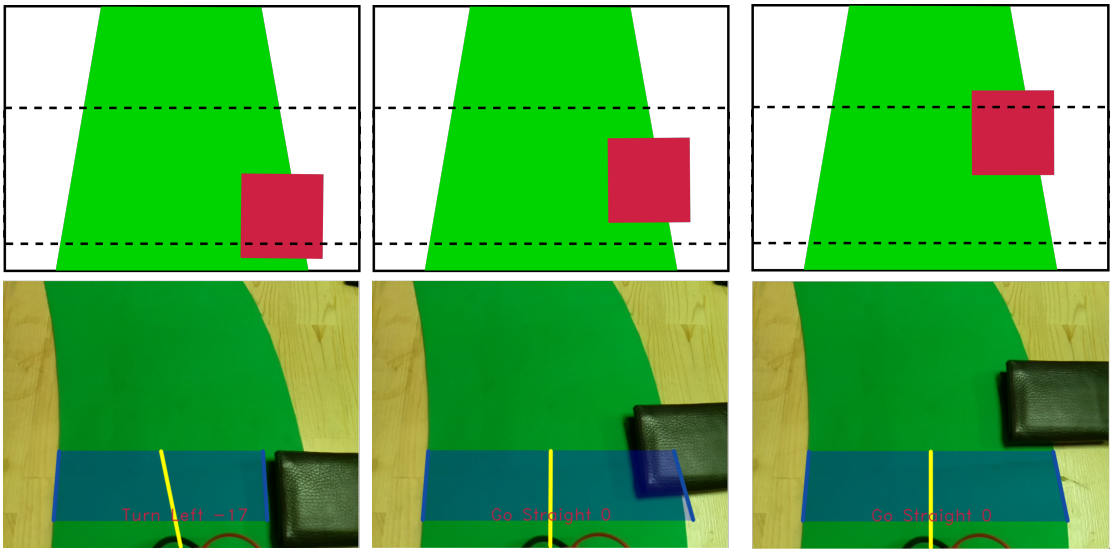
(a) Obstacle is at the Beginning of the Path (b) Obstacle is at the Middle of the Path (c) Obstacle is at the End of the Path.

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



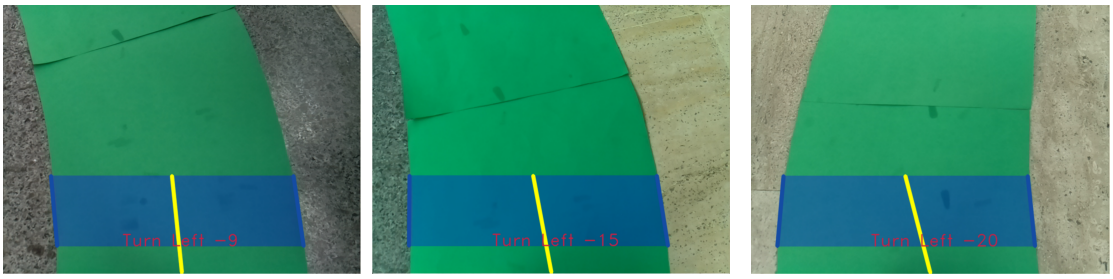
(a) Obstacle is at the Beginning of the Path (b) Obstacle is at the Middle of the Path (c) Obstacle is at the End of the Path.

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



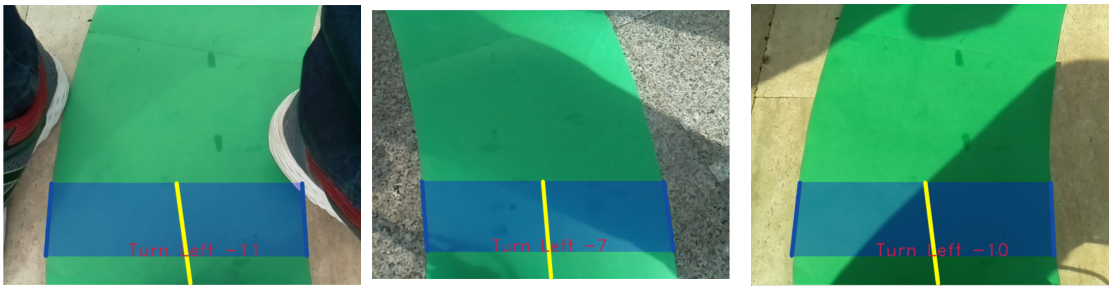
(a) Obstacle is at the Beginning of the Path (b) Obstacle is at the Middle of the Path (c) Obstacle is at the End of the Path.

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

Results of PID Controller Subsystem Tests

Due to other limitations, the initial version of the PID controller subsystem was tested only for *PID Parameters Test for Given Input*. The test results were promising for the time being. Other tests are planning to be conducted on the subsystem in the following semester.

Results of Internal Communication Subsystem Tests

The results of the tests revealed that all steps are successful but the last step. The data send rate is determined to be 25 strings per second. The string length varies between one and three characters. The data is fully received if the rate is slower than 25 strings per second. However, the data loss and improper decoding is observed on the Arduino side. This must corrected by means of coding or switching to an alternative solution.

Results of External Communication Subsystem Tests

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

Results of Direction Subsystem Tests

The test were conducted for the motor pairs used in the Critical Module Demo. The test can be repeated for new motor pairs if needed.

Results of Speed Subsystem Tests

No test result is currently available due to mentioned reasons.

Results of Motors Subsystem Tests

The tests are done. The results are negative because motors torque value did not match with the declaration of the supplier. 3kg-cm is the decelerated value, but motors can only produce 750 g-cm. Therefore, this system is failed in torque requirement, and re-considered in the following period.

RPM test has not been done since torque value is not supporting test setup.

4.2 Robustness of the Design

5 Other Considerations

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

5.1 Cost Analysis

Estimated cost anaysis for the project can be investigated at *Table 2*. The reproducible vehicle is expected to cost under 200 dollar as desired by the project requirements.

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

As seen from the *Table 2*, budget is optimized with some changes such as Arduino Uno is replaced with its nano version, and upper layer of the chassis is designed thinner plexi glass. However, critical components, such as motors, ToF sensors and wheels, are selected for their performance. Powerbank selection is based on its size. The chosen one is the smallest powerbank which can give enough output to supply Raspberry Pi under full load. Camera and Raspberry pi have no other option in this project. Li-Po battery selection is based on duration and output voltage. 12V output is required during motor drive, and long term is required for demonstrations. Therefore, 1750 mAh 11.1V 3S battery is selected.

5.2 Power Analysis

Table 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

The *Table 3* shows the consumption under regular case and extreme case. If extreme scenario is considered, full power consumption of Raspberry Pi is supplied from powerbank while motors are supplied by Li-Po battery as can be seen from the *Figure 21*. Also, sensors and Arduino are supplied from Pi because they do not have high demand. Powerbank has two output could supply 2.5A for 5V output, so Pi could supply in the worst case. Li-Po battery has these specs: 1750 mAh 11.1V 3S 25C, so it can supply 43 ampere constants during discharge although the motors demand 1.1 ampere at stall condition. All in all, sources are completely enough for consumption even in the worst case conditions.

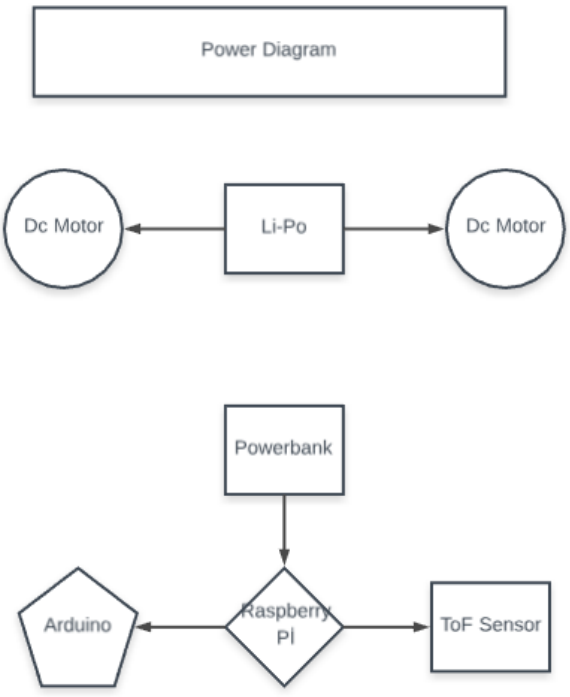


Figure 21: Electrical Architecture of the Project

6 Conclusion

7 Disclaimer

All information and content contained in this report are provided solely for proof-of-concept. DUAYENLER Ltd. Şti. guarantees that the report and information contained obeys the restrictions and rules ordered by the Standard Commitee.

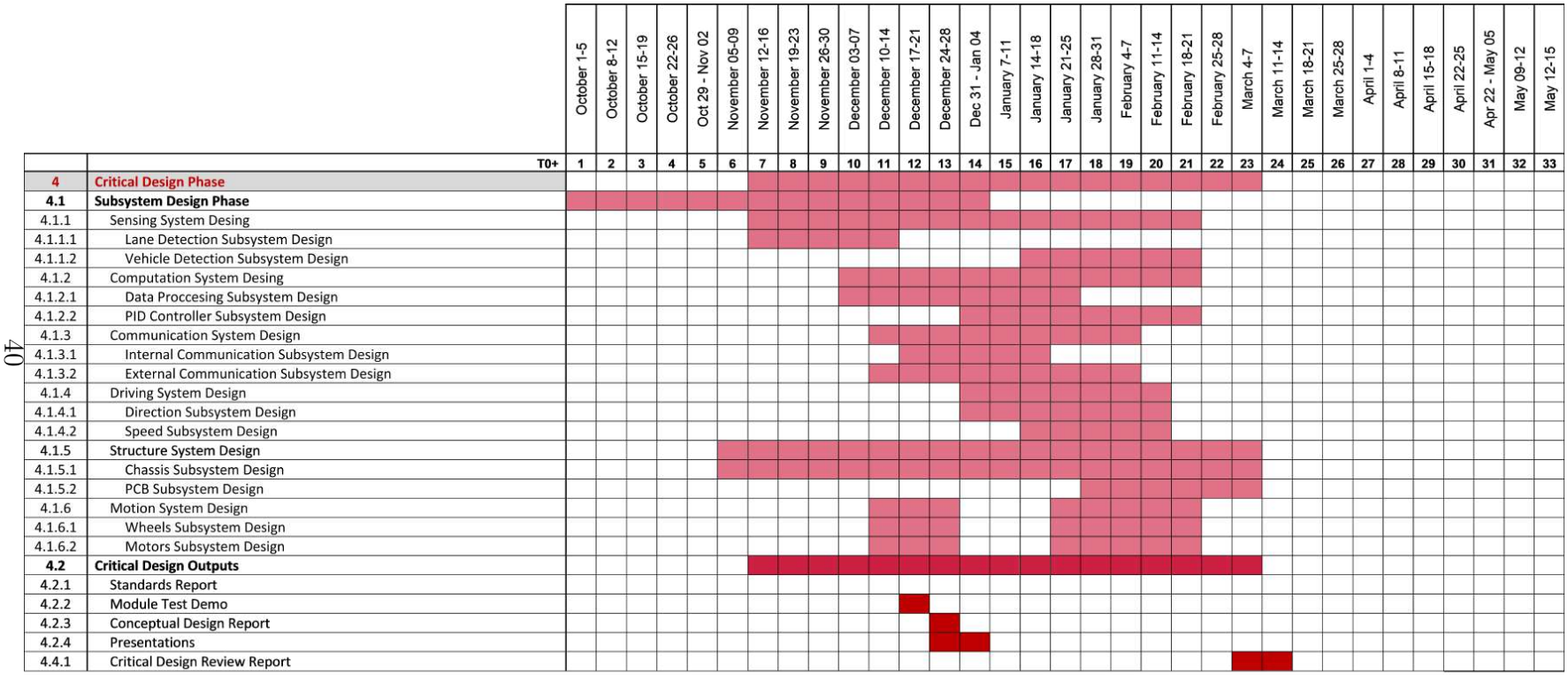
Halil TEMURTAŞ Erdem TUNA Enes TAŞTAN

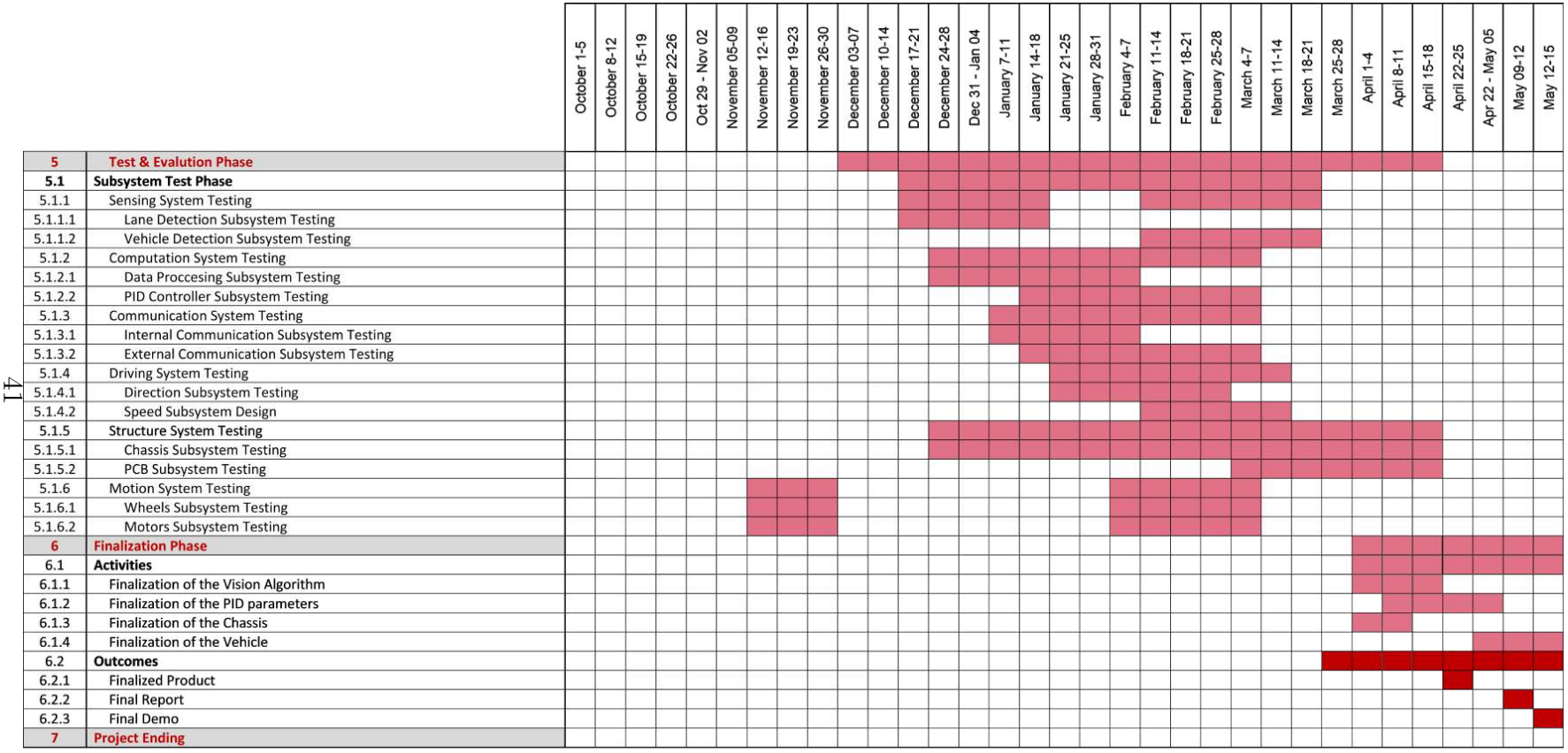
Sarper SERTEL İlker SAĞLIK

08 March 2018

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