MIDDLE EAST TECHNICAL UNIVERSITY

DEPARTMENT OF
ELECTRICAL AND ELECTRONICS ENGINEERING

EE493 ENGINEERING DESIGN I

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

December 26, 2018

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: System organization of the project

2.1 Sensing System

This system is responsible for interpreting data from the environment.

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.

It 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) Lane Detection Subsystem Requirements
      i. The subsystem should be able to detect only the shades of green color
      ii. The subsystem should be able to detect edges in the camera frame in any light condition
      iii. The subsystem should be able to tell differences between disturbances and lane
      iv. The subsystem should be able to interpret the middle of the lane if both sides are present at the frame

   (b) Vehicle Detection Subsystem Requirement
      i. The subsystem should detect the opponent to be caught with in a 5 cm
      ii. The subsystem should detect the chasing opponent if it reaches from back with in a 5 cm
      iii. The subsystem should trigger the handshake protocol

2. Solution for the Subsystem

The subsystem basically detects the lane. This subsystem uses OpenCV libraries for processing camera frames. The input is captured from a Raspberry Pi camera that is mounted to the vehicle. The captured frame is firstly, preprocessed by a denoising filter and HSV color filter. The edges in the frame is detected by Canny Edge Detector algorithm. The output of Canny is a binary image filled with ones and zeros. The resulting binary image is processed with Hough Line detector to find pixels that possibly form a line. After the lines are found, the processed frame is sent to Data Processing subsystem. The block diagram of this subsystem is given in *Figure* **??**.

The first processing on the captured frame is denoising by blurring. The blurring filter is a GaussianBlur of (3x3) matrix with zero variance both in x and y directions. Next, the filtered image is color filtered in HSV color space. The filter range of the HSV filter is adjusted only to include shades green. The lower bound for HSV filter is [H=60, S=120, V=106] and the higher bound is [H=82, S=255, V=235]. Later the color filtered image is processed by Canny Edge Detector. This process eliminates all pixels except those are constituting an edge. Edge pixels are usually formed when there is a transition from one object to other. To find pixels constituting a line, Hough Lines function is used.

Hough Lines function outputs two pixels points on the image that form a line. The output of the function is tens of such line points. The accuracy of the function can be adjusted by playing with the input parameters. The function basically draws all lines that might go through a point in the free space (in polar coordinates) and intersects those lines. If the origin points of the intersected lines are within a specified bound, then the line is defined and it can be expanded further. If found points on a line don't exceed a threshold, that line is discarded. The adjustment of such parameters are done with trial and error approach. An important note is that this function is probabilistic, meaning that even if the input frame is never changed, the output line points would be close to previous point but not the same. An example call for this function together with its parameters is given *Script 1*. The output, line detected frame, is sent to Data Processing subsystem.

```
HoughLinesP(input=img_threshed, output=line,
rho=1, theta=CV_PI / 180, threshold=15,
minLineLength=30, maxLineGap=40);
```
Script 1: Hough Lines Function with its Parameters

3. Discussions on the Solution

2.1.2 Vehicle Detection Subsystem

1. Requirements for the Solution

(a) The system should be able to produce middle line to follow

(b) The system should be able to control the robot

    i. Data Processing Subsystem Requirements

       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

    ii. PID Controller Subsystem

       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

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

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. It 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

2. Solution for the Subsystem

The data processing subsystem is the main computation unit for the vehicle. The main objective of this subsystem is to give PID controller subsystem a meaningful data from unorganized data coming from lane detection subsystem.

```
1  std::vector<cv::Vec4i> lines  // like a 4x4 matrix
2  double slope_thresh = 0.005;  // absolute threshold slope
3  for (lines_points)
4          startP = Point(x1, y1);
5          endP = Point(x2, y2);
6          line_slope = startP/endP;
7          if(abs(line_slope) >= slope_thresh) line_is_valid;
8          else line_is_not_valid;
9
10 for (lines_is_valid)
11         if(x1>320 && x2>320) it_is_right_line;
12         else if(x1<320 && x2<320) it_is_left_line;
13         else discard_the_line;
```

Script 2: The Algorithm to Classify the Lane Lines as Right or Left

The input of this system is an edge detected binary image. Next, points of the lines are classified as left or right borders of the lane. The elimination of the wrong points are done concurrently with the classification. Then, filtered points are fitted in two separate lines to create left and right borders of the lane. As the lane borders are found, the next and the last step is to determine the direction of the vehicle. The output of this subsystem is a turning angle and a direction. The whole process is summarized in *Figure* **??** as a block diagram. The requirements of this subsystem are listed below.

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.

After finding the all left and right line points, actual left and right lines are constructed by applying the least square method to the both point sets. The result of this method is 4 points (x1,y1,x2,y2) to refer the left lane line and 4 points to refer the right lane line.

The last process on the image is predicting the turn angle and direction. The prediction of direction is done by comparing the slopes of the left and the right lines. The turning is to be made to the side of the lane line having less slope. The angle is determined by computing the angle between the normal line of the

current direction and guide line. Guide line is constructed with the average of the middle points of the right and the left lines and current point of the vehicle. Vehicle's current point is assumed to be in the middle of the beginning of the image. Thus, the beginning point of the guideline is the current position of the vehicle and the endpoint is the target point to be reached. The output of this processing is shown in *Figure* **??**. The turning angle and the direction is sent to PID Controller subsystem.

The proposed algorithm for data processing has some flaws that are already discussed. To overcome those problems, DUAYENLER has some alternative solutions. An improvement might be on finding lane boundaries. This operation is currently realized with Hough Lines function. Another approach would be to scan rows an accurately edge detected image. If a white pixel is encountered, this would imply a lane boundary and the last time a white pixel is encountered would mean te other lane boundary. If the edge detected image contains noise, the solution might yield a wrong result. However, this can be overcomed by setting maximum number of white points that can be detected in a row.

Another improvement could be on positioning on the lane. The proposed algorithm assumes that the vehicle follows the lane always in the center. This assumption is not the case always. The new algorithm will differentiate between the center of the lane and center of the capture image. By averaging coordinate points of both lane boundaries the actual lane center can be found and this point must be in some proximity with respect to the center of the image.

3. Discussions on the Solution

2.2.2 PID Controller Subsystem

1. Requirements for the Solution

2. Solution for the Subsystem

In our case the desired steady state error to be compensated by the controller is angle information coming from the data processing unit. For this purpose discrete time version of a general PID controller will be utilized. General form for discrete time PID controller may be seen below:

$$G_{PID}(z) = K_{bias} + K_p + K_I \frac{1}{1 - z^{-1}} + K_D(1 - z^-1)$$

Using bilateral Tustin transform where

$$z = \frac{1 + \overline{s}}{1 - \overline{s}}$$

The discrete time controller can be designed as if it is in continues domain.

Theoretically very similar approach for designing a controller is to design in a S-domain. For this kind of controllers, general form of the continues time PID controller may be seen below:

$$G_{PID}(s) = K_{bias} + K_p + K_I \frac{1}{s} + K_D s$$

Another alternative method is to design the controller directly in discrete domain without using bilateral Tustin transform.

3. Discussions on the Solution

2.3 Communication System

.aaa

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

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.

```
1  import serial
2  ar=serial.Serial("/dev/ttyUSB0",9600)
```
Script 3: Serial object decleration in Python

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

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

Script 4: Serial communication setup in C

(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 3 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 4. 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 controller. It 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

2. Solution for the Subsystem

Depending on the configuration of the wheels, exact control of the vehicle might vary. However, there are certain methods to accomplish orientation. The vehicle will definitely have two wheels or palettes that will be driven by two separate DC motors. That configuration allows differential drive method to orient the vehicle. PWM values of the motors can be adjusted such that the speed difference between them results in a turn as much as desired angle. The exact difference values on the PWM values depends on the specs of the used motors and voltage sources.

Two different H-bridge motor drivers are proposed to be used to drive DC motors: L298N and L293D. Both can drive two motors separately with one IC. However, maximum current rating of the former one is larger being 2A while L293D can supply 0.6A per channel.

Direction unit is responsible for the orientation of the vehicle. It stores the last required orientation and the new one coming from the controller. After that, it tries to make the orientation as close as new one. Both data can be represented as vectors.The angle between those two vector is tried to be minimized by the controller. Before moving on to the operation, note that the angle can be used as a measure of the error that the direction unit have. The less the angle the more correctly operates the direction unit. The requirements of this subsystem are listed below:

3. Discussions on the Solution

2.4.2 Speed Subsystem

1. Requirements for the Solution

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

line_angle-
base_speed

2.5   Motion System

This system is responsible for maintaining mechanical feasibility of the driving system. It has two subsystems namely,

1. *Wheels Subsystem* which is responsible for the design of the wheel structure depending on the required speed-torque parameters.

2. *Motors Subsystem* which is responsible for the correct selection of the motor that is suitable to requirements.

2.5.1   Wheels Subsystem

1. Requirements for the Solution

2. Solution for the Subsystem

   The wheel placement combination that will be utilized is 2+1 combination. This combination can be assembled by placing 2 car wheels (with motors) to the back and the one boll caster to the front or vice versa. These configurations provide easy implementation and fairly reliable handling on the path. However, for certain obstacles may significantly disturb vehicles balance in this configuration.

3. Discussions on the Solution

2.5.2   Motors Subsystem

1. Requirements for the Solution

2. Solution for the Subsystem

   One of the widely used motor type is brushed gearhead DC motors. They are cheap and require enough torque for this competition. They have mainly two section one is the normal DC motor and gear head. Gear is a must to enhance

torque performance of the DC motor. In other words, gear balance rpm and torque of the motor.

Motors are the one of the important physical components of the project. They represent the whole system's actions. As stated in the project definition, the path should be completed within 20 seconds and total length of the path is approximately 3 meters. Using this specifications, a simple calculation for RPM requirement yields

$$v = \frac{3}{20} = \pi \times D \times \frac{RPM}{60}$$
$$RPM = \frac{2.865}{D} \tag{1}$$

where D is the diameter of the wheels in meter. Currently used wheels has 6.5 cm diameter, which implies that RPM=44. This means that under load average RPM needs to exceed this value. To estimate torque value, it is necessary to predict components weight. When weight of the current components are measured, result is around 1kg, so rated torque of the each motor expected to 3 times larger than this value to obtain well acceleration in loaded condition.

3. Discussions on the Solution

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. It 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

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° to 45°. 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* **??**.

3. Discussions on the Solution

2.6.2 Printed Circuit Board Subsystem

1. Requirements for the Solution

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

3   Detailed Requirements and Justifications

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* **??**. This section includes the explanation, requirements, test procedures and test results for the subsystems.

3.1   System & Subsystem Requirements

1. Sensing System Requirements

2. Computation System Requirements

3. Communication System Requirements

    (a) The subsystem should ensure safe internal communication

    (b) The subsystem should ensure safe external communication

        i. Internal Communication Subsystem Requirements

            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

        ii. External Communication Subsystem Requirements

            A. The subsystem should be able to communicate with the opponent via Wi-Fi protocol

            B. The subsystem should be able to execute handshake protocol

4. Driving System Requirements

    (a) The subsystem should control motion subsystem according to output of the computation system

        i. Direction Subsystem Requirements

            A. The subsystem should drive the motors according to computation system outputs

            B. The system should ensure that the vehicle follows the lane

        ii. Speed Subsystem Requirements

            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

5. Motion System Requirements

   (a) The system should ensure that the vehicle can drive itself with enough power

       i. Wheels Subsystem Requirements

          A. The subsystem should ensure that the wheels can grip lane without slipping in all conditions

       ii. Motor Subsystem Requirements

          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

6. Structure System Requirements

   (a) The system should ensure that structure is robust for external effects

   (b) The system should ensure that structure is balanced

   (c) The system should ensure that vehicle has a good appearance

       i. Chassis Subsystem Requirements

          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

       ii. PCB Subsystem Requirements

          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.

3.2 Justifications on Design

justification for re- quire- ments

Compliance with Reqs

4 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) 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.

(c) 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.

(d) 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

(c) Torque Test:

i. Fix the motor at horizontal position with respect to ground

ii. Attach an object of one kilogram

iii. Power up the motor

iv. Increase the weight to a point where the motor is not pulling anymore

v. Record the value and check with expected results

vi. If the result is not comparable with expected values and very low, motor can be considered as broken

vii. Contact the seller for more information

9. 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

10. 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

5 Testing Stage

5.1 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. The results were promising. The algorithm sweeps up the surrounding disturbances completely. The sample outputs together with Data Processing subsystem are shown in *Figure* **??**.

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 are done. The results are positive and reflects the expectations. The turn angle and direction are properly output. A flaw of this subsystem is that both right and the left lane lines must be determined. Otherwise, the prediction does not give stable

results. This must be improved together with the Lane Detection subsystem regarding the code algorithm. The sample outputs can be seen in *Figure* **??** *and Figure* **??**.

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.

5.2 Robustness of the Design

6 Plans

Each team member is assigned to a subsystem according to their interest and qualification. *Figure* **??** summarizes the assignments. Besides, a Gannt Chart is prepared to have an detailed overview of future works and available in *Appendix* **??**.

Future plans regarding the improvement of the project are listed here in system level.

6.1 Estimated Cost/Power of Project/Gantt

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: Estimated Cost Analysis for the Project

Component	Price (in Dollar)
Raspberry Pi	48
Camera	23
Chassis	20
Arduino	5
Motors	22
Wheels	8
Motor Driver	2.5
Powerbank	12
Li-po Battery	15
Distance Sensors	18
Additional Components	10
Additional Payments	15
Total Project	198.5

7 Conclusion

8   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

27 December 2018