T.C.
BAHÇEŞEHİR UNIVERSITY

FACULTY OF ENGINEERING AND NATURAL SCIENCES

AUTONOMOUS VEHICLE-PROJECT 1080

Capstone Project Final Report

Denizhan Aksakal-SEN

Mehmet Fahri Bilici-CMP

Senem Tuğçe Demiral-CMP

Zeynep Ergin-CMP

Alperen Ertürk-EEE

Dr. Andrew John Beddall

Dr. Görkem Kar

Dr. Özge Yücel Kasap

ISTANBUL, May 2020

STUDENT DECLARATION

By submitting this report, as partial fulfillment of the requirements of the Capstone course, the students promise on penalty of failure of the course that

- they have given credit to and declared (by citation), any work that is not their own (e.g., parts of the report that is copied/pasted from the Internet, design or construction performed by another person, etc.);
- they have not received unpermitted aid for the project design, construction, report or presentation;
- they have not falsely assigned credit for work to another student in the group, and not take credit for work done by another student in the group.

ABSTRACT

AUTONOMOUS VEHICLE

Denizhan Aksakal-SEN

Mehmet Fahri Bilici-CMP

Senem Tuğçe Demiral-CMP

Zeynep Ergin-CMP

Alperen Ertürk-EEE

Cankat Saraç-EEE

Faculty of Engineering and Natural Sciences

Dr. Andrew John Beddall

Dr. Görkem Kar

Dr. Özge Yücel Kasap

May 2020

Abstract — Autonomous vehicles aim to provide a more comfortable drive with fewer errors by reducing the human factor. With low-cost components which provide the vehicle to detect objects, lines and line angles as visual inputs, distance measurements, and decisions with respect to those inputs while using Raspberry Pi's processor unit to compute those decisions, it is aimed to achieve such a status where vehicle moves according to the limitations. This report provides the process of verification and the progress of making an autonomous vehicle.

Key Words: Autonomous, YOLO, Hough Transform, Euclidean Transform, Canny Detection, PWM Signal, Communication, Optimisation,

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CMP	*Computer Engineering*
EEE	*Electrical and Electronics Engineering*
SEN	*Software Engineering*
YOLO	*You Only Look Once*
R-CNN	*Region Convolutional Neural Network*
ROS	*Robot Operating System*
Lidar	*Light Detection and Ranging*
API	*Application Programming Interface*
GPU	*Graphical Process Unit*
CPU	*Central Process Unit*
FPS	*Frame per Second*

The general subject is making an autonomous vehicle.

CMP Department (Machine Learning and decision software)

Senem is responsible for lane and road strip detection and response algorithms.

Zeynep is responsible for traffic stop signs, and speed limit signs detection and response algorithms.

Fahri is responsible for decision algorithms with required sensors. This member is also co-responsible with an EEE student for information flow between the decision software and vehicle hardware.

Some of the above students will help with implementing computing and the required camera system.

EEE Department (Vehicle and sensors)

Cankat is responsible for the code of the vehicle movements (velocity adjustment according to the voltage of the driver, servo angle adjustment, etc.). This member is also responsible for the vehicle's basic movement commands.

Alperen is responsible for hardware optimization and physical build of the vehicle (adjusting gears, etc.), computing platform with required sensors (e.g., ultrasonic distance sensor). This member is also co-responsible with a CMP student for information flow between the decision algorithm and vehicle hardware.

SEN Department (Mobile application, Data recording, and display)

Denizhan will prepare a mobile application and a suitable database for recording all actions/decisions taken by the vehicle.

1.1. Identification of the Need

Product is an autonomous car that is also known as a driverless car, robot car, self-driving car, or autonomous vehicle.

1.2. Definition of the Problem

The vehicle accident rates have risen due to the increase of the vehicles which are used by humans. To decrease the accident rates, the vehicles should be replaced with autonomous feature cars. This feature's purpose is based on lowering the human factor.

Functional feature objectives wanted:

- The vehicle should be able to move with a wide range of speed so that the performance can be measured as a function of vehicle speed. For this, the vehicle speed needs to be measured by the vehicle.

- The speed of the vehicle should be able to be controlled by road signs.

- The vehicle should be capable of staying within the lane at all times.

- The lane should be no wider than three car width.

- The vehicle should be able to avoid objects by changing lane safely, otherwise by braking (the vehicle should not crash into any encountered objects).

Functional feature objectives achieved:

- The speed of the vehicle can be controlled by an algorithm-motor driver.

- The vehicle can move at three different paces with the required angle.

-The vehicle can detect the line

Problems occurred

- Lack of computing power(overloaded processor)
- Following the lane

1.3. Standards and constraints

- Environmental effects

Vehicle accident rates will be drop. Reduce carbon emission

- Social effects

Since the driver does not have to drive the vehicle, s/he will able to enjoy the travel with the others if there are any.

- Economic effects

Because the autonomous option requires a greater amount of money, those vehicles will be purchased with a major expense. Also, the drop rate of car accidents will affect the economy.

- Ethical issues

Some people might say it is better to be used by humans rather than the machine itself.

- Health and safety

It is safer for the machine to drive itself than a conflict-error, which is caused by humans, and resting for the driver will be at better ease.

1.4. Conceptual Solutions

Communication concepts;

Arduino slave-raspberry master, no communication (Only Works with raspberry's processor)

To overcome the overload in raspberry's processor, the Electrics and Electronics Engineering department came up with a solution, which is creating a wired communication network between raspberry and Arduino. The Project right now uses only raspberry's processor; if needed, the concept of communication will change from none to master and slave with USB

Line detection concepts;

Deep learning was the other conceptual method for line detection. Due to the power of the raspberry may not be enough; it was not the chosen concept for the vehicle. Image processing was the one that was suitable for the system.

Vehicle code decision types

Behavior cloning, the decision algorithm

1.5. Physical Architecture

To be able to work, the system needs inputs. The system has two inputs which are visual and sound waves. The visual inputs separate in two; object detection and line detection. A sound wave is used to measure the distance. Raspberry uses those inputs to make calculations in a subsystem by using modules and certain codes. After completing those calculations, Raspberry sends information signals to the drivers and motors by using a decision algorithm. While deciding the algorithms, Raspberry sends the work that is printed as logs to the database which can be observed via using a mobile app. The system interface diagram design can be observed in Figure 1.

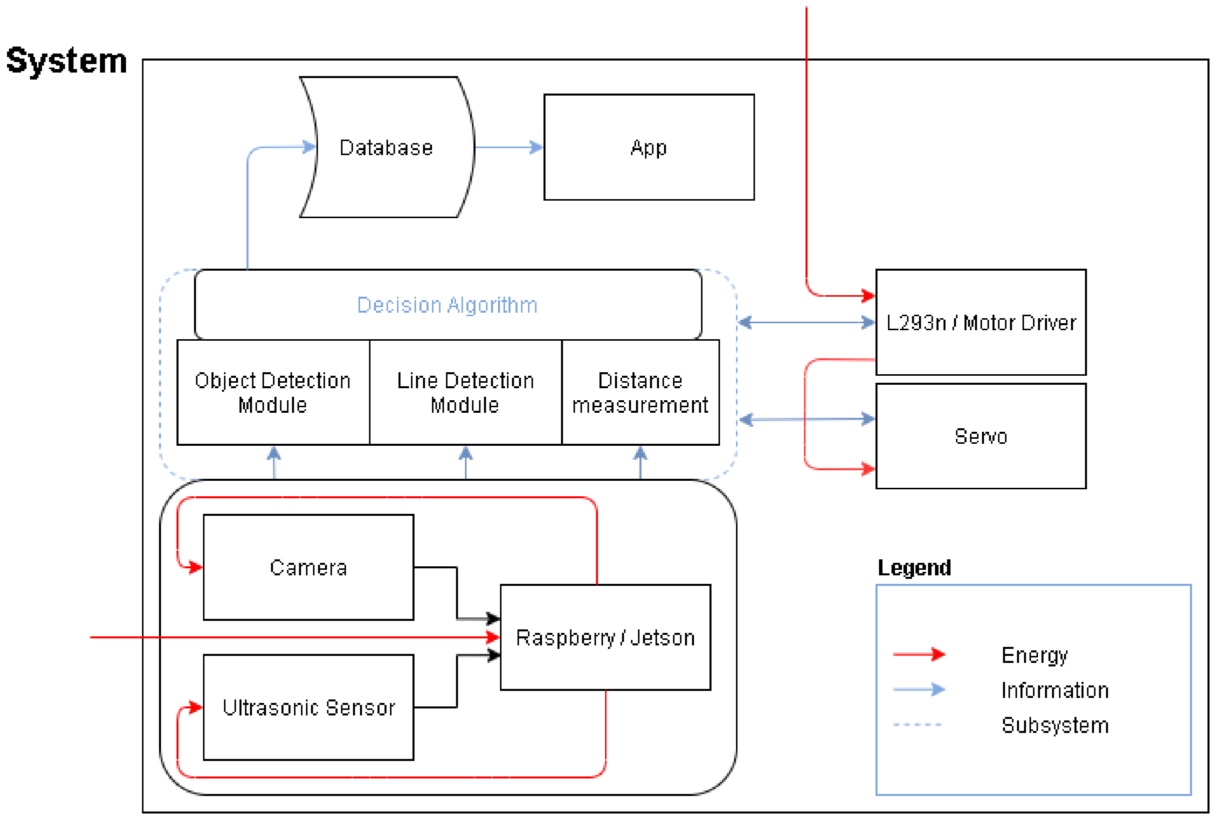


Figure 1. System interface diagram

2. WORK PLAN

2.1. Work Breakdown Structure (WBS)

1. Autonomous Car
    - 1.1 Car chassis, wheels, motors, battery
        - 1.1.1 Mechanical/Electrical
            - 1.1.1.1 Order/manufacture parts
            - 1.1.1.2 Circuit scheme draw
            - 1.1.1.3 Assemble
    - 1.2 Car remote communication and control
        - 1.2.1 Communication
            - 1.2.1.1 Order/manufacture control parts
            - 1.2.1.2 Assembly and coding
        - 1.2.2 Motor Control
            - 1.2.2.1 Order/manufacture control parts
            - 1.2.2.2 Assembly and coding
    - 1.3 Decision Algorithm
        - 1.3.1 Object Detection
        - 1.3.2 Line Detection
        - 1.3.3 Distance measurement
    - 1.4 Hardware-Code Optimization
    - 1.5 Car decisions and actions
        - 1.5.1 Database
        - 1.5.2 Mobile App
            - 1.5.2.1 Reading logs
            - 1.5.2.2 Display outputs
    - 1.6 Support
        - 1.6.1 Line
        - 1.6.2 Object
        - 1.6.3 Database
        - 1.6.4 Code organization
        - 1.6.5 Research

2.2 Responsibility Matrix (RM)

Task	Denizhan	Mehmet Fahri	Senem Tuğçe	Zeynep	Alperen
Mechanical & Electrical					R
Communication		R			R
Motor Control					R
Object Detection		S		R	
Line Detection		S	R		S
Distance Measurement		R			R
Database	R	S			
Mobile App	R				
Integration	S	R	S		R

Table 1. Responsibility matrix for the team

R = Responsible

S = Support

2.3. Project Network (PN)

The project network is shown in Figure 2 according to WBS.

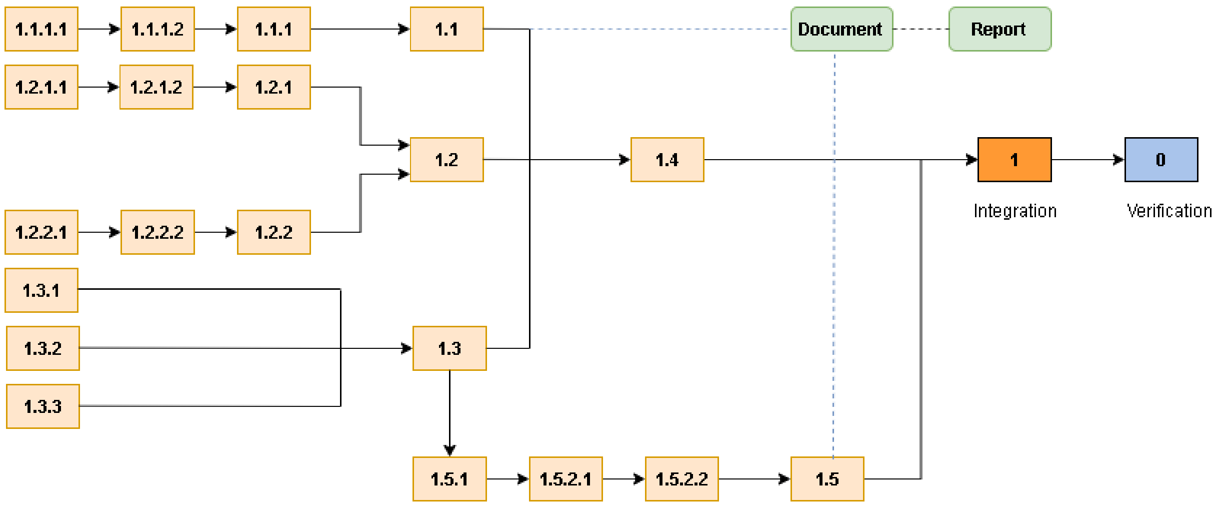
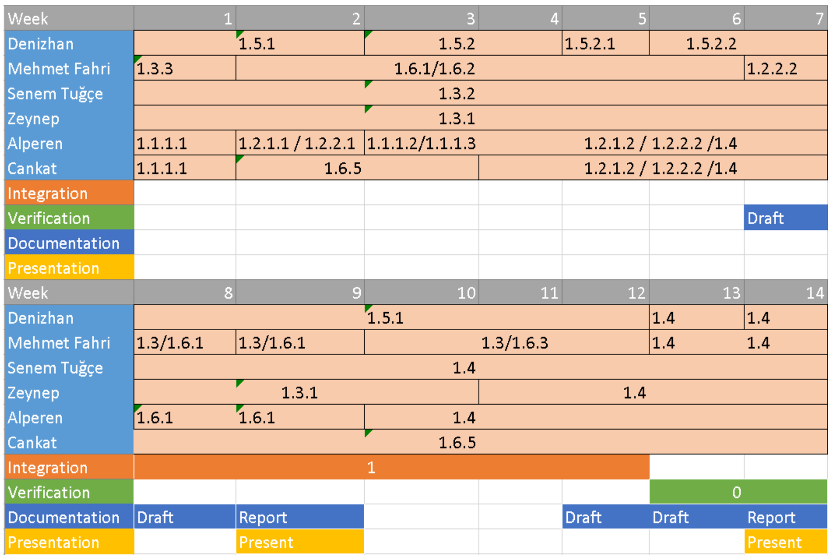


Figure 2. Project network

2.4. Gantt Chart

According to WBS and Project network, a timeline is created for more efficient work as it stated in Figure 3



Week	1	2	3	4	5	6	7
Denizhan		1.5.1	1.5.2		1.5.2.1	1.5.2.2	
Mehmet Fahri	1.3.3		1.6.1/1.6.2				1.2.2.2
Senem Tuğçe			1.3.2				
Zeynep			1.3.1				
Alperen	1.1.1.1	1.2.1.1 / 1.2.2.1	1.1.1.2/1.1.1.3		1.2.1.2 / 1.2.2.2 /1.4		
Cankat	1.1.1.1	1.6.5			1.2.1.2 / 1.2.2.2 /1.4		
Integration							
Verification							Draft
Documentation							
Presentation							

Week	8	9	10	11	12	13	14
Denizhan			1.5.1			1.4	1.4
Mehmet Fahri	1.3/1.6.1	1.3/1.6.1		1.3/1.6.3		1.4	1.4
Senem Tuğçe			1.4				
Zeynep		1.3.1			1.4		
Alperen	1.6.1	1.6.1	1.4				
Cankat			1.6.5				
Integration			1				
Verification						0	
Documentation	Draft	Report			Draft	Draft	Report
Presentation		Present					Present

Figure 3. Timeline for project

3. DESIGN PROCESS

3.1. Computer Engineering

3.1.1. Definition of the Problem

Thousands of accidents occur around the world every year due to carelessness. Single neglect can cause the loss of many people. Some of the drivers are not aware of traffic rules.

This rule also applies to an autonomous car. If there is no driver in your vehicle, then your vehicle must be capable of detecting signs, lines, and any object on the road. Object detection, line detection, and ultrasonic sensor are essential at this point. A car that cannot detect objects; lines cannot be classified as an autonomous car. The vehicle can able to detect the lines and go between these lines. Therefore, nowadays, it is not efficiently possible to use it in traffic.

3.1.2. Review of Technologies and Methods

In the coding part, Google Colab was used because it is easy to share and understand where the missing and error part.

3.1.2.1 Object detection

The autonomous car must be capable of recognizing the objects near the road. We use object detection for training the vehicle. This is a technology that does the detection and classification of the objects from the camera, image, or video. This is a computer vision-related technology, which is mentioned as an overview in Figure 4.

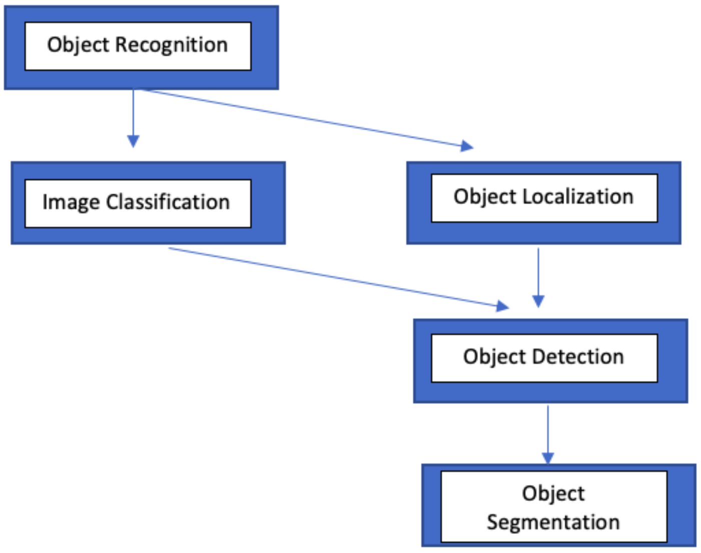


Figure 4. Overview of Object Recognition Computer Vision Tasks

Object detection uses image classification and object localization. Image classification determines the type or class of the object; it gets the image with the objects in it as an input and serves the image with the classified object as an output. Image localization detects the location of the object in the image and serves the image with the location frame around the object. The classification with respect to the image is stated in Figure 5.

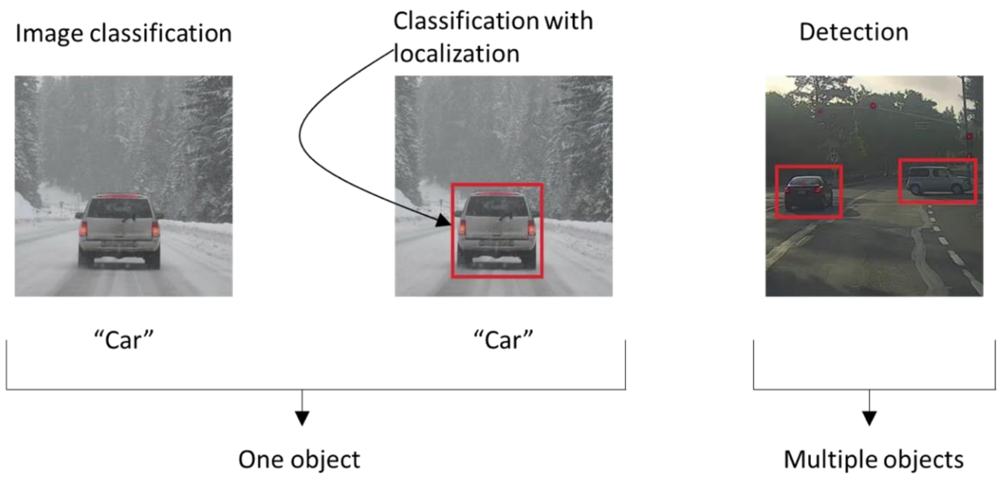


Figure 5. Image Classification

Image Classification (left); Classification with localization (center); Detection of multiple objects (right).

Image Reference: http://datahacker.rs/deep-learning-object-localization/

3.1.2.2 Distance Measurement

In the system, ultrasonic sensors were used to find the distance from any object in front of the car. In the ultrasonic sensor, there are two sides, one sound output, and sound input. Ultrasonic sensors are using a 40kHz sound wave. To determine the distance, it sends sound waves from the output side, and if it encounters anything, it bumps and returns to the source of the wave to the input side. Another part of the ultrasonic sensor receives this bump, and it calculates the time between the sensor and the object, which is in front of the car. Because we know the speed of the sound wave (343.2 m/s), it multiplies time and speed, and then we found distance. In Raspberry Pi, we will connect this sensor using raspberry's GPIO pins. For the programming part, we will use the GPIO library in python.

3.1.2.3 Line Detection

For the Line Detection module, the line detection pipeline was designed.

Line Detection Pipeline :

Read the Image

Gray Scale Transform

Canny Edge Detection

Hough lines Identification

Find the Road Lines

Read the Image

imread() method is used when the user wants to load an image in the program from the defined file. imread() function that identified in OpenCV library. In imread() method, the image name must be specified. In the vehicle system, frames are read in this function. The image which is shown in Figure 6 is used to test those functions.

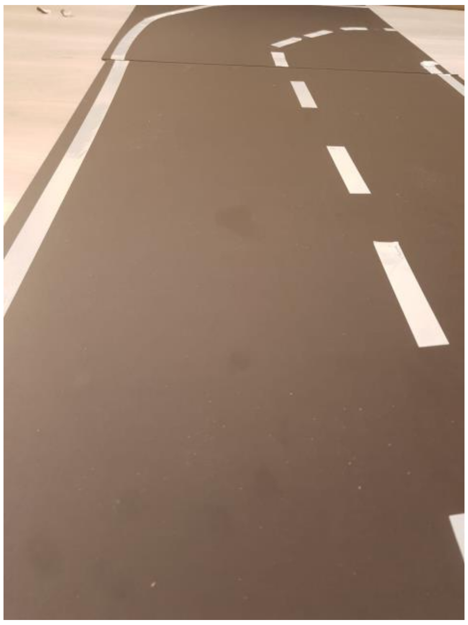


Figure 6. Image example

Gray Scale Transform

In Line Detection module, GrayScale Transform makes it difficult to detect the lines, so Gray Scale Transform was not added in the vehicle system.

Canny Edge Detection

In Canny Edge Detection, which is shown in Figure 7, edges are identified. Then the color of the edges must be changed from the rest of the places in the image. The Canny Edge Detection function has three parameters. The first parameter reads the image. The second and third parameters are low and upper ranges.

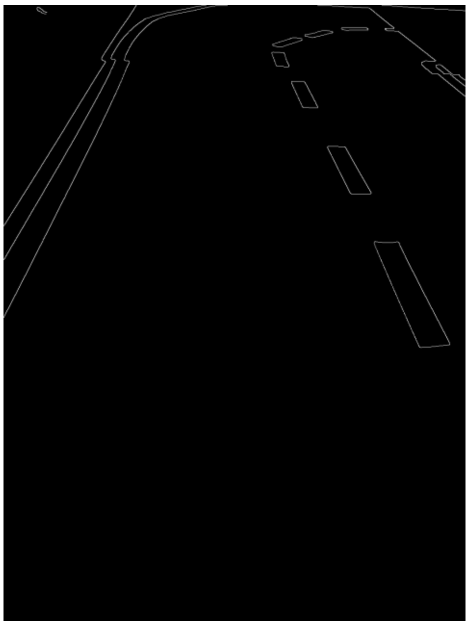


Figure 7. Canny edge

Find the Road Lines

After applied every process identified above, lines will be detected successfully, and the result is shown in Figure 8

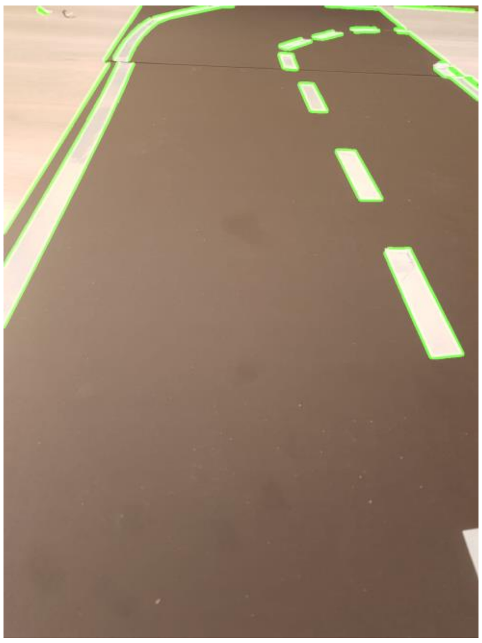


Figure 8. Detected lines

3.1.3. Standards and Constraints

3.1.3.1 Line Detection

For the line detection part, the camera angle view should be adjusted properly because a smooth image is essential for line detection. If the image is not smooth, then detection will not work correctly.

The camera may not work decently at certain heights.

We cannot expect the project to work as always in daylight.

We do not identify accident scenarios.

3.1.3.2 Object Detection

YOLO (You Only Look Once) is a technology that is used for object detection.

The basis of YOLO is that this algorithm handles the object detection process as a single regression problem.

In the YOLO algorithm, bounding boxes and the class possibilities for the boxes are estimated by a single neural network in one evaluation. This makes the algorithm work extremely fast.

Since the whole detection is a single network, it can be optimized end-to-end directly on detection action.

YOLO takes an image and split it into an SxS grid, each of the grid there are bounding boxes. For each box, the network builds a class possibility. The bounding boxes with the class possibility are selected to locate the object in the image.

The limitations for estimating about bounding box may occur by YOLO, because every grid cell forecasts only two boxes and they can have just one class. As a result of this limitation, the objects which are near are been limited for the detecting operation.

3.1.3.3 Decision Algorithm

Rasberry computing power limits the Decision Algorithm. Because of image processing uses Rasberry's most of the CPU power.

3.1.4. Conceptualization

3.1.4.1 Object detection

For object detection, there are several algorithms, such as R-CNN, Faster R-CNN, YOLO, and each algorithm has some drawbacks about them. Object detection becomes very slow with the R-CNN algorithm because R-CNN does a ConvNet forward pass for each object proposal.

YOLO algorithm can be the best choice for doing real-time object detection. The reason makes this algorithm fastest compared to other algorithms is that YOLO passes the image through a neural network at once and it can guess the coordinates and classes of the object in that image.

Fast R-CNN makes mistakes about background patches in an image for objects because of couldn’t seeing larger content. YOLO makes much fewer background errors than Fast R-CNN.

3.1.4.2 Line Detection

Deep Learning based line detection is the other conceptual. A neural network could be trained. Due to a lack of time and limited Rasberry computing power, Deep Learning was not used in the vehicle.

3.1.4.3 Decision Algorithm

After researches, in an autonomous car, Behavior Cloning is used a lot. In Behavior Cloning, car driven by an agent in the track. While the car is driving, the camera takes photos of the roads. These photos are saved in the folder. While photos are saving, current steering angle and current speed saved in excel file. Then the model can be trained from this information. After this process, road photos can give as input, and the model returns steering angle and speed.

Since lack of time and Behavior Cloning was not using a line detection model in the system.

3.1.4.4 Ultrasonic Sensor

LiDAR is the other option to measure the distance between any object ant vehicle. LiDAR is more accurate than an ultrasonic sensor. However, it is much more expensive, and the size is big for a vehicle, although it is heavier for the vehicle.

3.1.5. Physical Architecture

3.1.5.1 Object detection

YOLO algorithm architecture is about 24 convolutional layers and two fully connected layers.

First, the input image is taken and resized to 448×448 pixels by YOLO. Also, the image goes through the convolutional network, and after, it gives output in the form of 7×7×30 tensor. The information about coordinates of the box's rectangle and possibility dispensation over all classes which the system is trained for is given by the tensor.

3.1.5.2 Decision Algorithm

Decision Algorithm Pipeline:

The first image is taken. In this picture, if any lines detected, then ultrasonic sensor controls if there are any object or not. If there is not any object more than 30 cm, the vehicle will go. If the vehicle detects the object between 30 cm and 20 cm, speed will decrease, and the vehicle will print a warning message. Finally, if the distance between vehicle and object, the vehicle will stop and will print a warning message again.

Since only the Line Detection module and Ultrasonic Sensor were integrated into the vehicle system. So Decision Algorithm only works depends on Line Detection outputs.

3.1.6. Risk Assessment

Due to COVID-19, the school was closed, but the project has to continue. In this period, the integration part was started. So, the integration part continued by connecting online. VNC Viewer helps to connect Raspberry Pi.

3.1.6.1 Object detection

In terms of object detection, there are some risky situations for our car. The idea of the autonomous car which cannot detect and consider the signs around the road is not acceptable and safe. In the object detection module, in some situations, according to YOLO architecture or environmental issues, localization error is seen.

3.1.6.2 Line Detection

In the Line Detection module, there may occur some situations that make the system risky. The camera angle is important for the situations mentioned. The camera will not work properly and will not see the lines. So that lines will not be detected and vehicle unable to turn wheels.

3.1.6.3 Decision Algorithm

In the Decision Algorithm, if CPU loads, the vehicle unable to move.

3.1.7. Materialization

3.1.7.1 Object Detection

For Object Detection, it works separately from the vehicle due to not integrated yet, and our module predicts objects in 56 milliseconds.

3.1.7.2 Line Detection

The line Detection module detects every line in the image out of the track. After researches, it understood that if the threshold increases, then the problem will solve. Finally, after tested in the vehicle, it works properly.

Secondly, in Line Detection module, due to the camera angle module was not working properly. After changing the camera angle, the module worked normally.

In the Line Detection module, the steering angle is calculated. Then the result of this calculation, angle sends to Servo Motor, and front wheels turn depends on this angle. In this calculation results, there was some problem that faced. Angle changed rapidly because the camera takes every frame, and this is the cause of vehicles that are not working properly. It solved by rounding angels. For example, if the angle is between 80-89, it rounds the angel to the minimum value. This process was done in the Decision Algorithm part.

3.1.7.3 Ultrasonic Sensor

As mentioned before (3.1.5.3), if the ultrasonic sensor detects the object between 30 cm and 20cm, it will decrease the speed. If it is closer than 20 cm, it will stop. When determining these values, it tested on the vehicle and understood that it reduces the possibility of the crash.

3.1.8. Evaluation

3.1.8.2 Object Detection

The Object Detection module was not integrated into the vehicle so that the optimization part was not started yet.

3.1.8.2 Line Detection

The vehicle should detect the line and go between 2 lines. The line Detection module was integrated into the vehicle and tested. After integrated and tested, it understood that vehicle detects the lines, but when the vehicle comes the curved part of the road, it could not work properly. This problem is trying to fix it. For most of our tests, the vehicle adjusts itself to stay between the lane on the track.

3.1.8.3 Ultrasonic Sensor

The vehicle should detect the object in front of itself, then change the appropriate line. From now on, the vehicle detects the object but does not change the road. It stops.

3.2. Electrical and Electronics Engineering

3.2.1. Definition of the Problem

Climate change- fuel consumption, human reaction time, vehicle accident rates

3.2.2. Review of Technologies and Methods

Raspberry pi 4 is a processing unit – board for computing algorithms with required conditions.

L293n is an H bridge motor driver to run dc or step motors with or without a PWM signal. PWM signal lets us decide the velocity of the motor.

Servo Motor, a servo motor is an electrical component that allows us to rotate or push an object. It is just made by simple motor inside is typically servo mechanism. If Servo Motor consumes DC energy, they called DC Servo Motor. Otherwise, they called as AC Servo Motor. We can get enough torque servo motors, even if it is small and lightweight packages. This system is used in many electronics systems. Servo motors are controlled by Pulse with Modulation in another way to say PWM. The system of PWM is to create a minimum, and maximum pulse and repetition rate (frequency) also in the basic model of servo motors only get HIGH (1) or LOW (0) PWM responses.

The Electrics and Electronics department had two communication methods for the project. They were non-communication and master-slave wired communication model. This conceptual method for the raspberry to work as a master and make Arduino as a slave was for decreasing the processor load level, if the processor is already in great shape, the vehicle will not need any communication network.

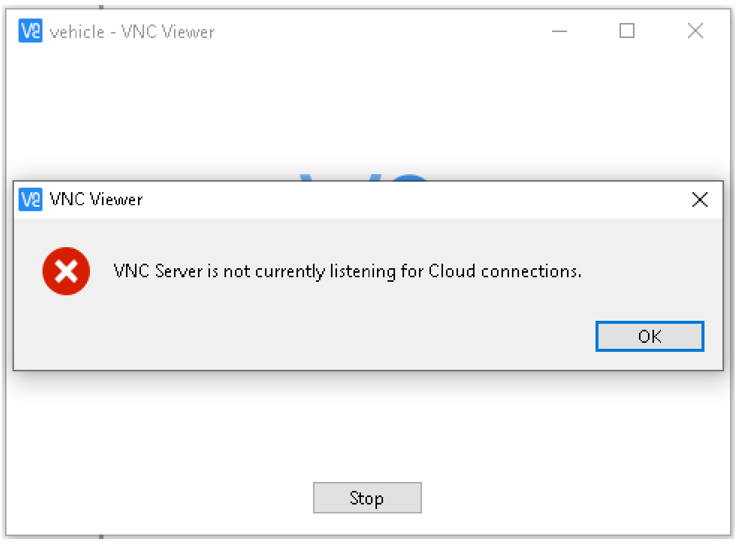
3.2.3. Standards and Constraints

• Project is based on one car model.

• We do not identify accident scenarios.

• If the speed of the car is high or the surrounding factors are more, due to insufficient Raspberry Pi power, all processes may slow down.

• As it shown in figure 9, lots of connection issues happen while testing, this is one of the major problem since it is harder to integrate and test the vehicle.


vehicle - VNC Viewer

VNC Viewer

VNC Server is not currently listening for Cloud connections.

Figure 9. Connection issues

3.2.4. Conceptualization

Every year, there are approximately 1.2 million people who lose their lives because of car accidents. The number of deaths is equal to 3,000 daily, concepts of a self-driving car is sensing the environment and moving safely. Main concepts of Electrical Electronics Engineering department are:

1) Sensor fusion
2) Communication
3) Control

The concepts that the EEE department is responsible for; firstly the ideal sensor fusion can see, detect and understand the system around the car, secondly if it is needed, a communication system via using raspberry as a master and Arduino as a slave (to reduce processor load) is composed by Electrical and Electronics Engineering department, thirdly with using the camera it is planned to keep the vehicle in the road and lastly control all the component system of the car.

3.2.5. Physical Architecture

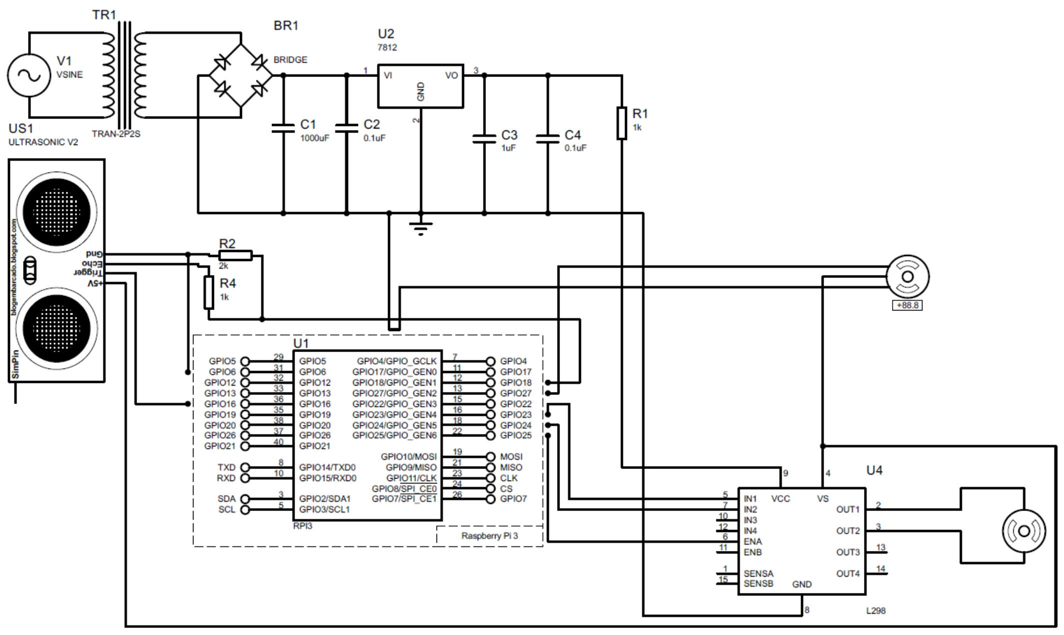


Figure 10. Proteus scheme of the project

Due to the lack of Proteus' model libraries (such as raspberry pi four b+ model, batteries), some of the components are drawn manually like batteries, as the scheme mentioned in Figure 10. To make a voltage supply, a voltage rectifier circuit is drawn using a diode bridge and transformers.[26] The Project is also drawn by the Electrics and Electronics Engineering department by using Fritzing, and the scheme that can be observed in Figure 11 became more understandable by all members since Fritzing models are easier to understand.

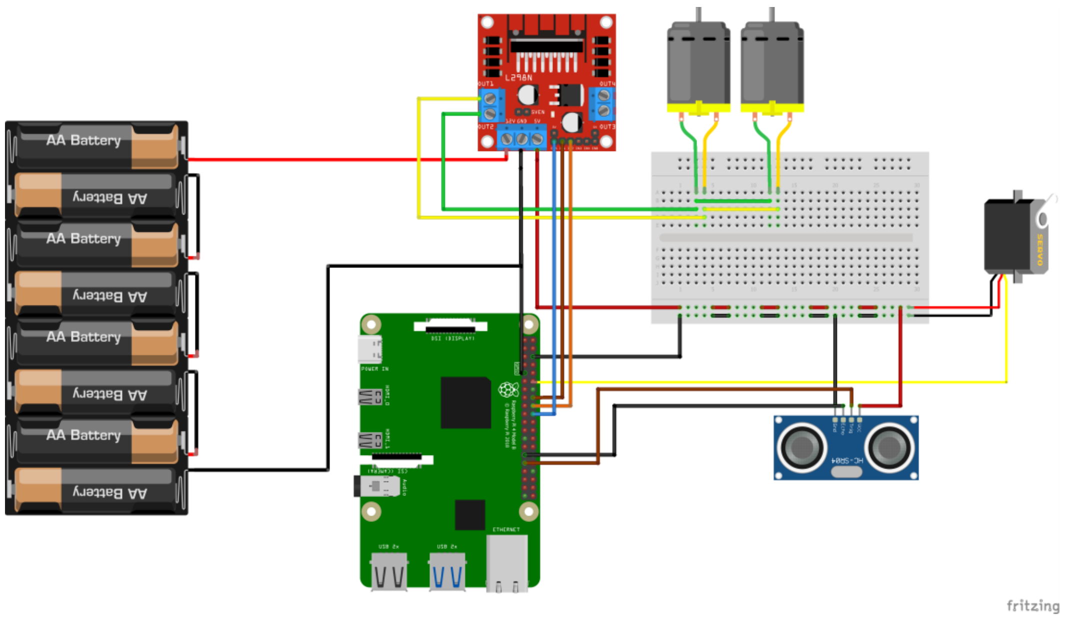


Figure 11. Fritzing scheme of the project

3.2.6. Risk Assessment

Due to a late start, the timeline delayed a bit. Vehicle build must be finished in the third week, but it finished in the fifth week. To solve this pacing problem, task responsibilities have changed.

Hardware quality for well working autonomous vehicles requires 41.855,43 TL according to the Open Zeka[29]. So basically, the budget that the autonomous vehicle project team is low. The hardware is especially expensive, and there must be a failure budget. F.e: Raspberry is around 400 TL, but the group has 1800 TL total, which is provided by the university. However, we realized that Raspberry might not be enough for visual detections. This project needs at least 3000 TL to complete the vehicle with failures.

3.2.7. Materialization

The building process of the vehicle started with disassembling an RC car and making it appropriate for the components to fit. The chassis of the vehicle is shown in Figure 12.

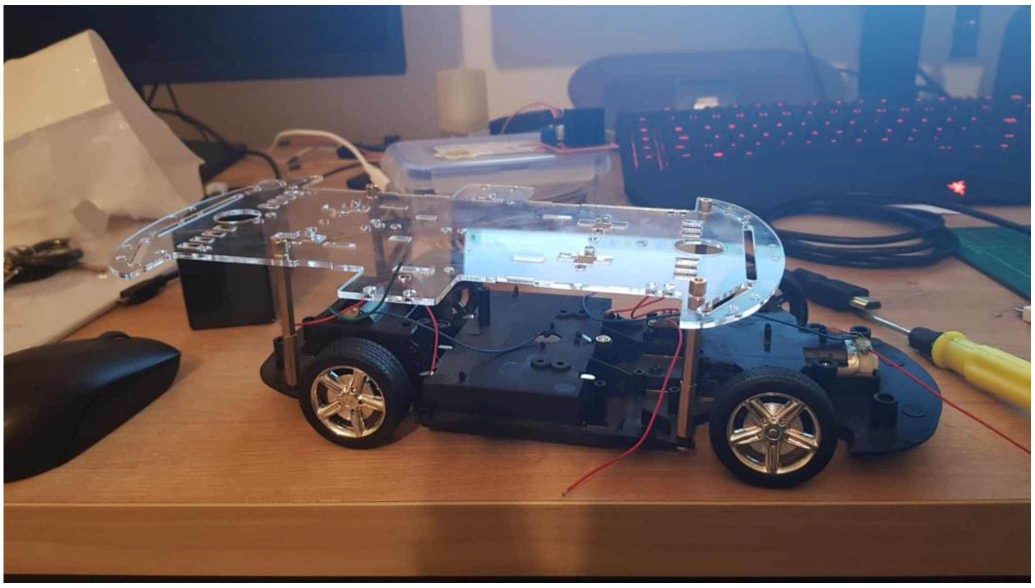


Figure 12. Chassis of the vehicle

Due to the vehicle does not have any ports for servo but a port for DC motor, the tests are tried with DC motor steering at first, as shown in Figure 13.

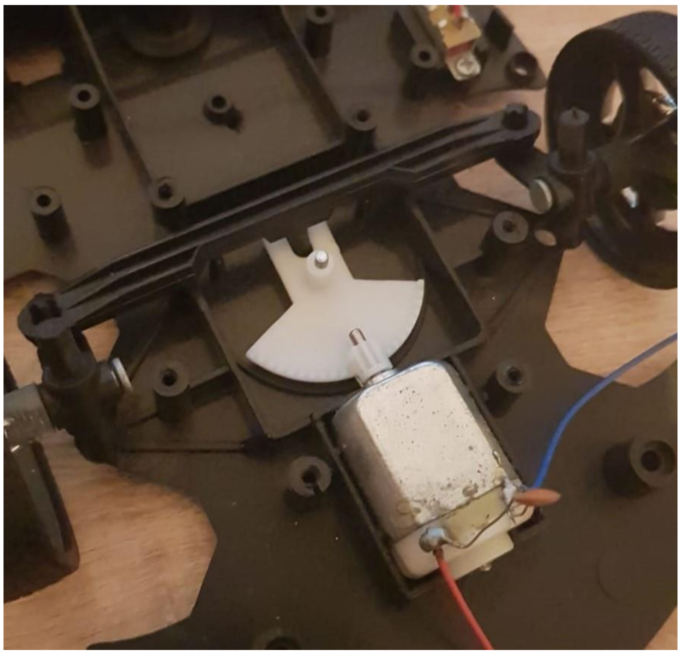


Figure 13. Steering with DC motor

After a few tests with DC motor, DC motor noted down as inappropriate for turns. Turns should be done with angles, so with breaking some parts of the vehicle, the team managed to create a port for servo motor. Also, ultrasonic is connected to Raspberry pi with using a breadboard, as stated in Figure 14.

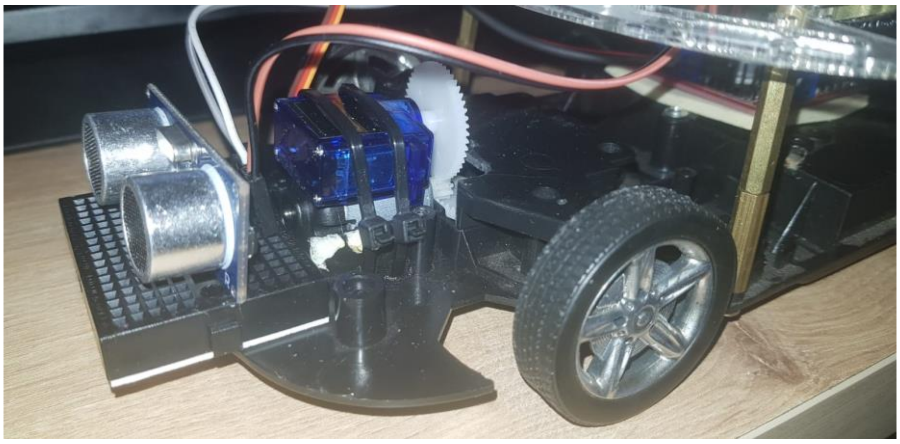


Figure 14. Ultrasonic sensor binding and steering with a servo to control with angles

The previous DC motor has not enough torque while turning at higher angles. So the vehicle's DC motor port is extended using side chisel. The extended port can be seen in Figure 15.

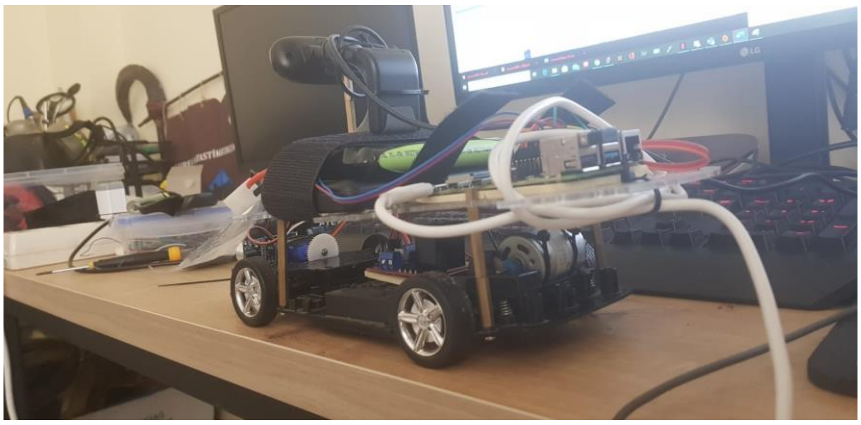


Figure 15. Changing DC motor to a higher torque motor

A webcam port is created for a better field of view, the motors connected with H-bridge motor driver. An adjustable webcam port and H-bridge motor driver can be observed in Figure 16.

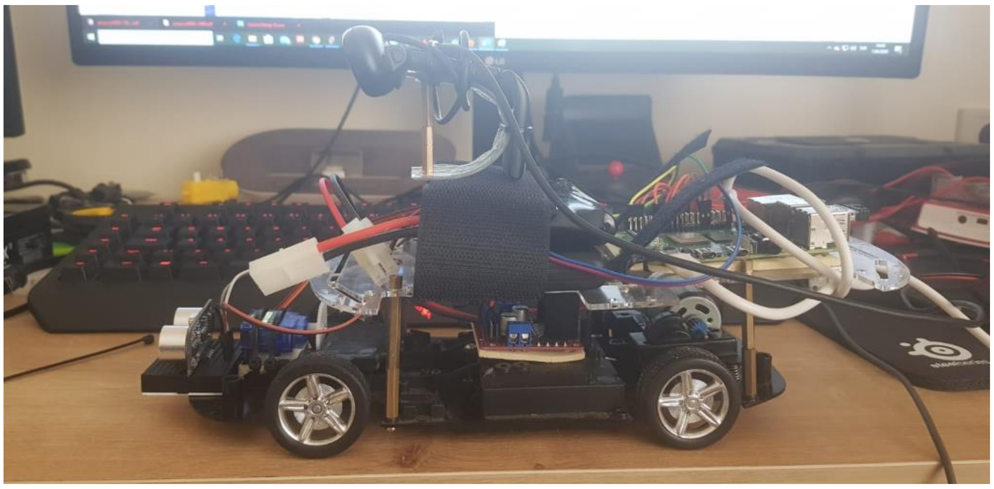


Figure 16. L298n Motor Driver connection with adjustable webcam port

The end of the assembly seems pretty stable. All components have fixed on the chassis to make them immobile while the vehicle is working. The finished assembly is shown in Figure 16.

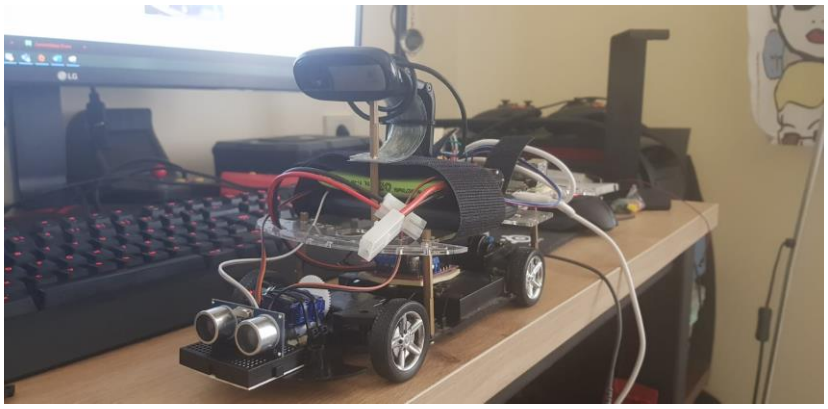


Figure 17. End of the assembly

During the process of testing, the vehicle got broken. So all of the hardware carried to the new chassis, as shown in Figure 18.

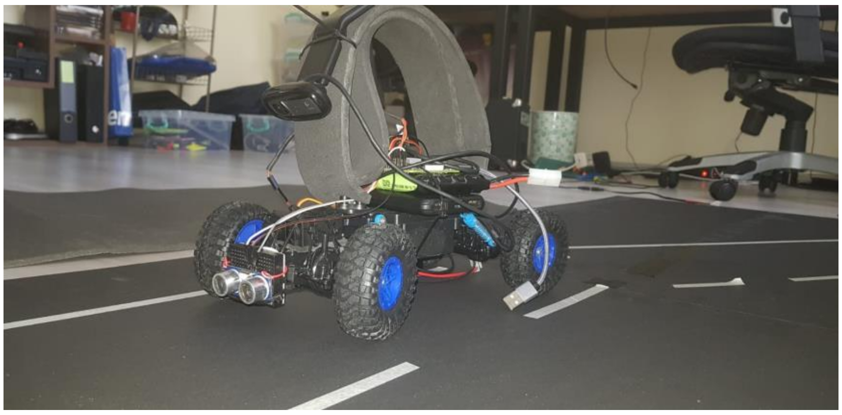


Figure 18. New chassis

Due to a weak extension of the servo motor, the new chassis steering is also got broken. A new and more smaller vehicle bought and all hardwares are carried to the newest one. Size comparison with respect to the other chassis' are shown in figure 19.

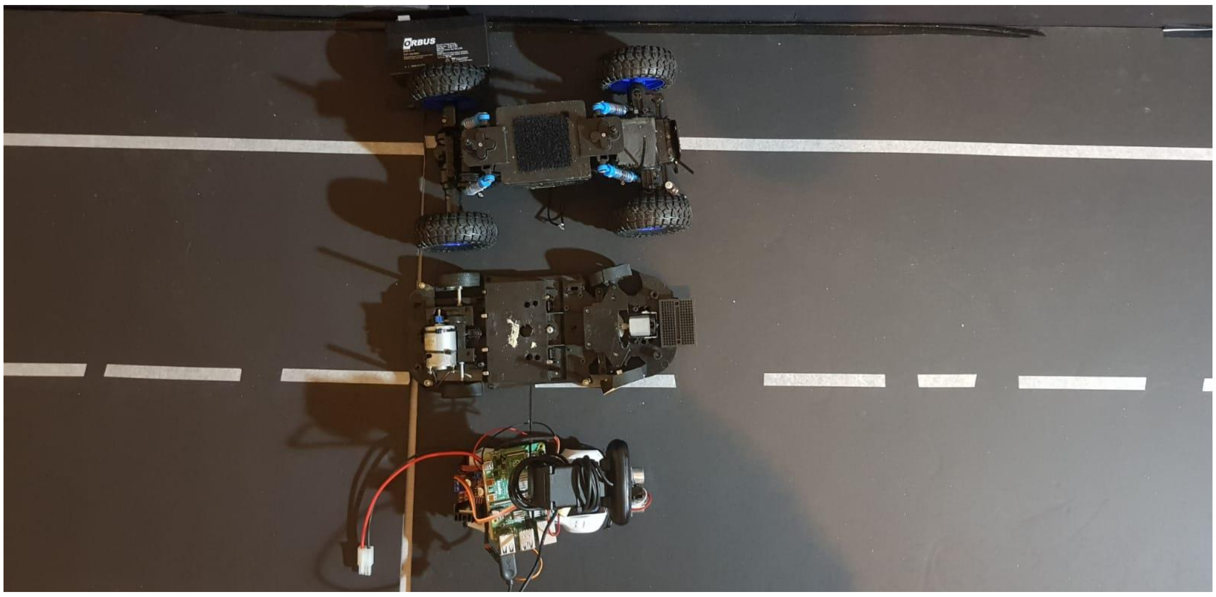


Figure 19. Size comparison

It was a bit challenging to carry hardwares to the newest chassis because it was really small. Look of the newest chassis is shown in figure 20.

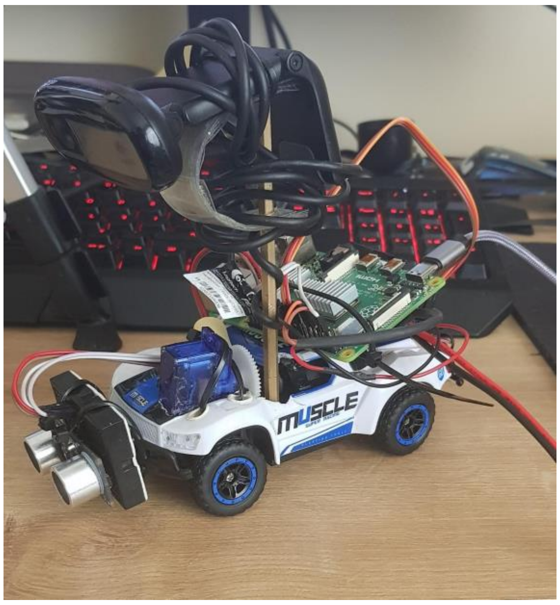


Figure 20. Updated chassis

3.2.8. Evaluation

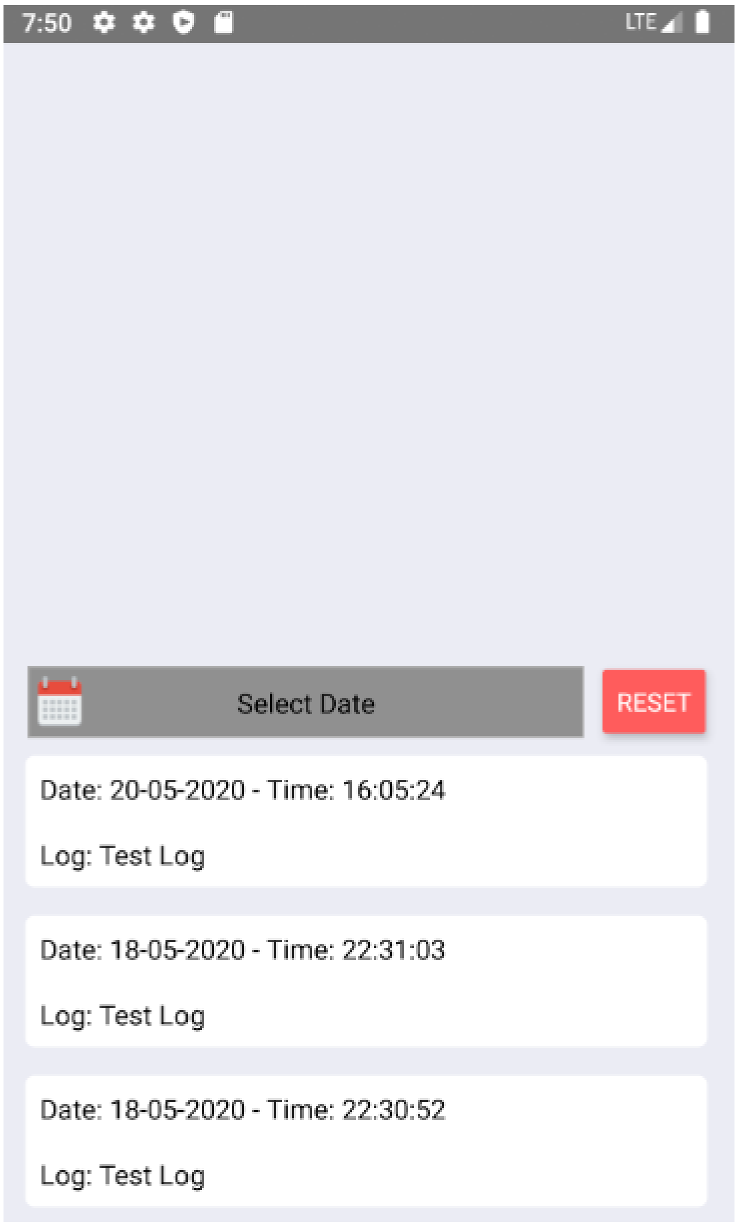
- The vehicle should be able to move with a wide range of speeds so that performance can be measured as a function of vehicle speed. For this, the vehicle speed needs to be measured by the vehicle. Speed measurement can be done manually using the duty cycle function, and the project does not need any external sensors to calculate the speed since the group members can control the pace of the vehicle.
- To make steering with ease, the vehicle steering turned into the servo. With this method, the decision algorithm can call the angle that is needed.
- Giving too much afford to call the other class's function although we set the Python code construction properly
- Broken hardware components delayed the process. f.e. Overloaded DC motor stopped working, due to too much weight 5V DC motor did not fulfill the required torque, so the motor changed to a more powerful one

3.3. Software Engineering

3.3.1 Interface Requirements

3.3.1.1 User Interfaces

In this application, the GUI has a simple interface, as shown in Figure 21. A blank screen will be used additional features later. FlatList used for multiple data view, DatePicker used for filtering logs. The screen can be slideable to view old logs, so interface does not need an external scroll.



Date: 20-05-2020 - Time: 16:05:24

Log: Test Log

Date: 18-05-2020 - Time: 22:31:03

Log: Test Log

Date: 18-05-2020 - Time: 22:30:52

Log: Test Log

Figure 21. User interface

3.3.1.2 Software Interfaces

The mobile application is written in React-Native and connected to Firebase. React-Native. React-Native is written using the Javascript programming language. The system includes these tools and libraries in Table 2.

Library	Version
react-native	0.61.5
Node.js	10.16.3
react-native-webview	9.4.0
react-native-datepicker	1.7.2
firebase	7.9.3
@firebase/database	0.5.22

Table 2. Libraries and versions

3.3.2 Functional Requirements

3.3.2.1 Behaviors of the Software Application

The following functions contain the most basic features of the application in Table 3.

Actor Name	**Name of Behavior**	**Description of Behavior**
Mobile Application	*firebaseConfig()*	*Configuration of DB*
Mobile Application	*readData()*	*Data reading from DB*
Mobile Application	addItemsRefListener()	*Data listener for Firebase*

Table 3. Basic features

3.3.2.2 Attributes of the Software Application

The following attributes contain information that maintains the most important values of the application in Table 4.

Actor Name	**Name of Attribute**	**Description of Attribute**
Mobile Application	*log_index*	*the index number of log*
Mobile Application	*log_data*	*the output from a car*
Mobile Application	*db_status*	*connection control for DB*

Table 4. Important information

3.3.2.3 Design and Implementation

Database Management System

Firebase Console used for DBMS. The most significant reason for choosing Firebase is that it provides a real-time database module. It can be accessed from the web page. It provides developing options, quality options, and analytics options for developers.

Database Schema

The database contains a table as :

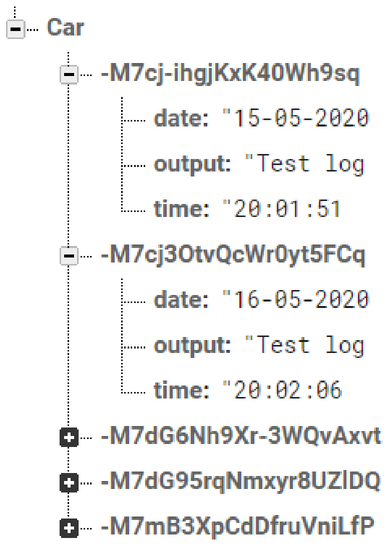
- index – primary key
- date - nvarchar(255)
- time - nvarchar(255)
- output - nvarchar(255)

Database Physical Model

"index" in the table above was used as the primary key. Index is randomly generated while writing output. “date” and “time” is the separated value of datetime. The reason for their separation is the use of calendar. The output value includes the action or decision taken by the car.

Completed Database Screenshots

The table and values in the picture below are for sample visualization only as seen in
Figure 22.



```
Car
  -M7cj-ihgjKxK40Wh9sq
    date: "15-05-2020
    output: "Test log
    time: "20:01:51
  -M7cj3OtvQcWr0yt5FCq
    date: "16-05-2020
    output: "Test log
    time: "20:02:06
  -M7dG6Nh9Xr-3WQvAxvt
  -M7dG95rqNmxyr8UZlDQ
  -M7mB3XpCdDfruVniLfP
```

Figure 22. Database screenshots

3.3.3 Nonfunctional Requirements

3.3.3.1 Performance Requirements

As a performance requirement, sufficient space must be left on the processor or ram in the Raspberry so that the image processing modules do not fail, and if any module fails or is overloaded, it will affect the writing of the logs into the database.

3.3.3.2 Safety Requirements

As a safety requirement, failure in the image processing modules or sending the log of the car's decision/action wrong or missing will cause data loss. This may result in missing or incorrect data to be sent to the application.

3.3.3.3 Security Requirements

As a security requirement, a lot of configurations are required for a database connection. Even if this configuration information is used by others, there should be also a rule to read and write on the database side. It should prevent the database from being accessed, written, and read by anyone.

3.3.3.4 Software Quality Attributes

The real-time module should provide correctness. Database configurations should provide interoperability between the mobile application and the car. Mobile application and the car should be tested simultaneously for the reliability of the given log. Finally, the car and mobile app must be re-tested at different times for the reusability qualification.

3.3.3.5 Business Rules

As a business rule the following scenario is used;

If the car is running, the decision-making algorithm and the line detection algorithm are running smoothly, and a log will occur.

Then logs are then written to the database and presented to the user through the mobile application.

Else no other log occurs, the user cannot follow the car's decisions and actions.

3.3.4 Use-case Modeling

3.3.4.1 Actor Glossary

The project has two main actors. One is the autonomous vehicle and the other is the user.

3.3.4.2 Use-case Glossary

To express functional requirements the use-cases shown in Table 5 are defined.

Use-case Name	**Description**	**Participating Actors**
Writing Logs	Car write logs to db	*Car*
Checking Logs	User check car's actions/decisions	*User and car*

Table 5. Use case

3.3.4.3 Use-case Scenarios

This use case scenario includes the transfer of logs of the car to the database shown in Table 6.

Use-case Name	Writing Logs
Use-case Description	Car's actions/decisions will be reported to db
Actors	Car
Pre-Condition	The vehicle should be on the road and the road should be easily detectable by the camera.
Post-Condition	The car must have made its decision according to the situation on the road and took action accordingly and send it to the database.
Normal Flow	Step 1: Engine should be started Step 2: If there is no object in the direction of the vehicle, it can go in the direction of the vehicle. Step 3: The vehicle will continue to detect the road and proceed as long as it does not encounter any situation. Step 4: As long as it continues to progress, it will continue to report to the database.
Alternate Flow	Alt-Step 2: If there is an object in front of it, the vehicle will stop, but it will report it to the database. Alt Step 3: If the vehicle cannot detect the road, it will continue to travel based on older data for a while.
Business Rules	If the car does not access the internet, it will stop writing to the database.

Table 6. Use case scenarios

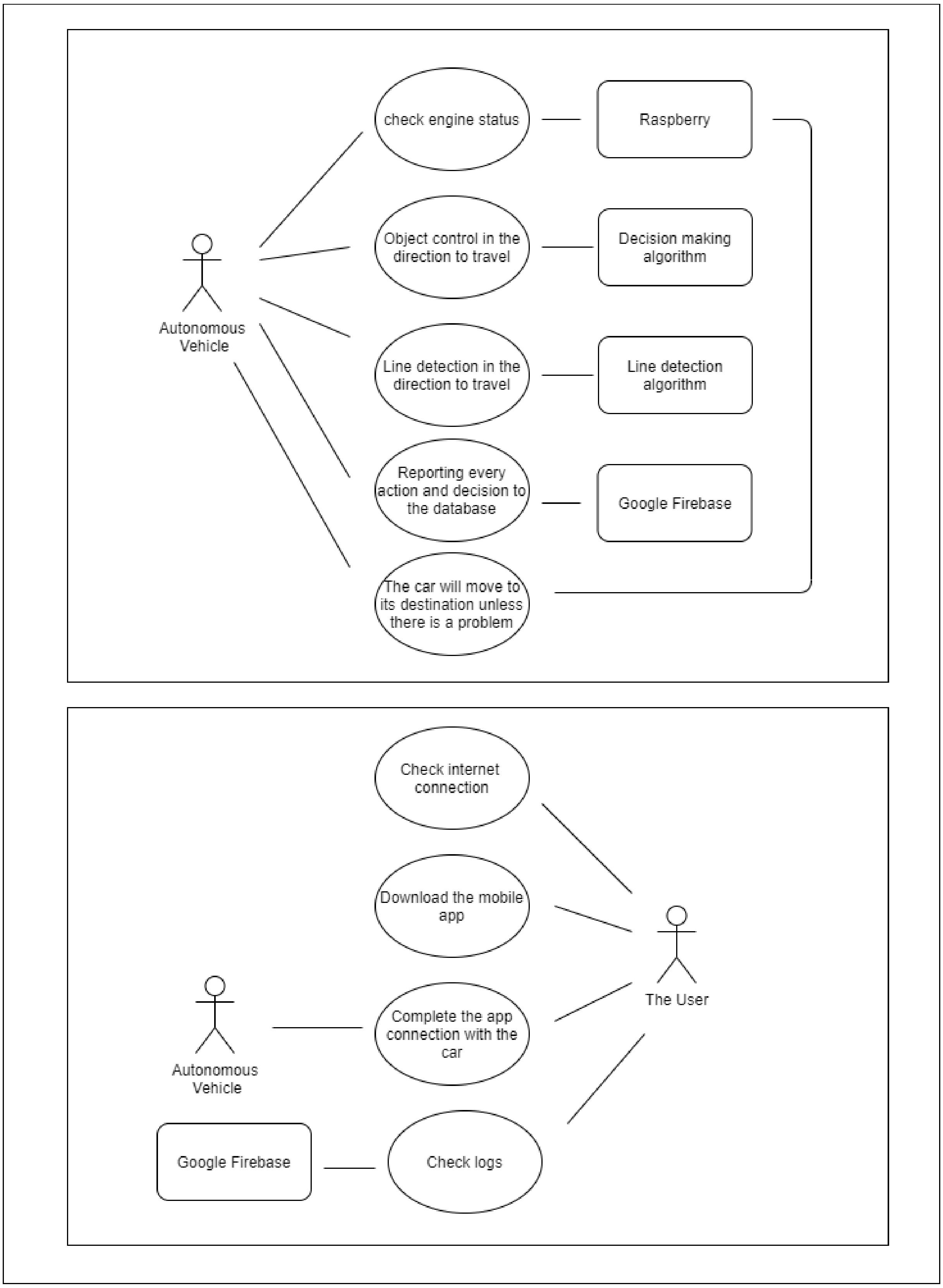
This use case scenario contains notation between the database and the application shown in Table 7.

Use-case Name	Checking Logs
Use-case Description	Car's actions/decisions will be displayed on the app
Actors	User and car
Pre-Condition	The user must have an autonomous vehicle and download the application to her/his phone and complete the connection between them.
Post-Condition	The user will be able to follow the actions and decisions of the vehicle step by step.
Normal Flow	Step 1: The user should download the mobile application to his phone. Step 2: The user must establish the link between the autonomous vehicle and the application. Step 3: If the connection is provided, the user can monitor the actions and decisions of the vehicle or review them later.
Alternate Flow	Alternative step 2: If the application connection with the car is not provided, the data cannot be accessed. Alternative step 3: If the user loses internet access even if they have made the application connection with the car, they will not be able to access the current data.
Business Rules	If the mobile application does not access the internet, it will stop displaying the logs.

Table 7. Notation between database and application

3.3.4.4 Use-case Diagram

The figure 23's top represents use case 1., the bottom represents use case 2.



Autonomous Vehicle
check engine status
Raspberry
Object control in the direction to travel
Decision making algorithm
Line detection in the direction to travel
Line detection algorithm
Reporting every action and decision to the database
Google Firebase
The car will move to its destination unless there is a problem
Check internet connection
Download the mobile app
Complete the app connection with the car
The User
Autonomous Vehicle
Google Firebase
Check logs

Figure 23. Use case 1 and 2

3.3.5 Data Modeling

The main task and functions are listed below

- The user can review the car's decisions/actions on the mobile app.
- The user will have information about the engine of the car, the direction, and the decision taken.
- The user does not have to inform the system about changes in the external environment.
- The user wants to know the car's decisions and actions through the app.
- The user wants to be informed about whether the engine is running and when there is any malfunction.

3.3.5.1 Activity Diagram

The figure 24's left represents the use case 1. and right part represents use case 2.

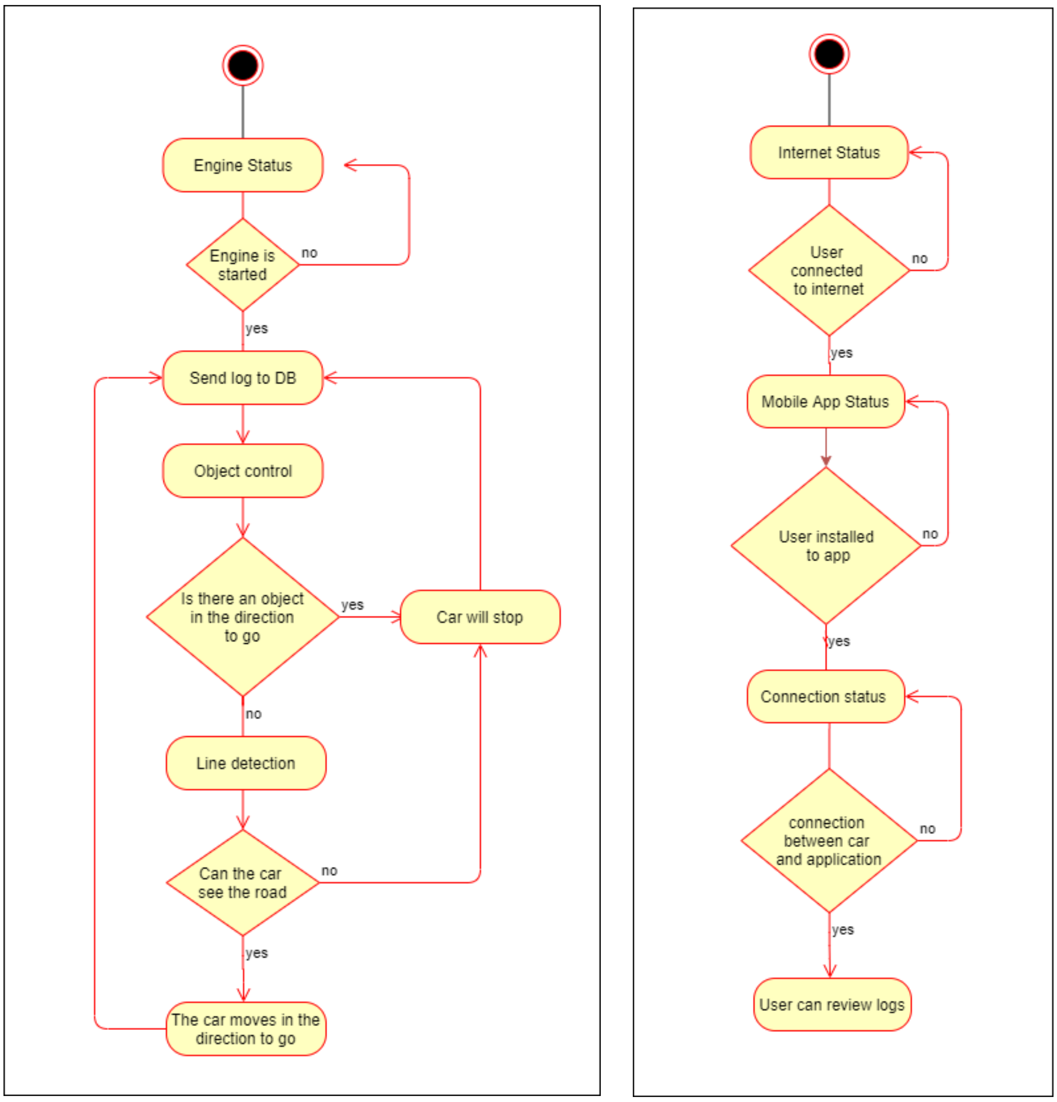


Figure 24. Use case 1 and 2

3.3.5.2 Sequence Diagrams

The sequence diagram shows the interaction of objects between two actors in Figure 25.

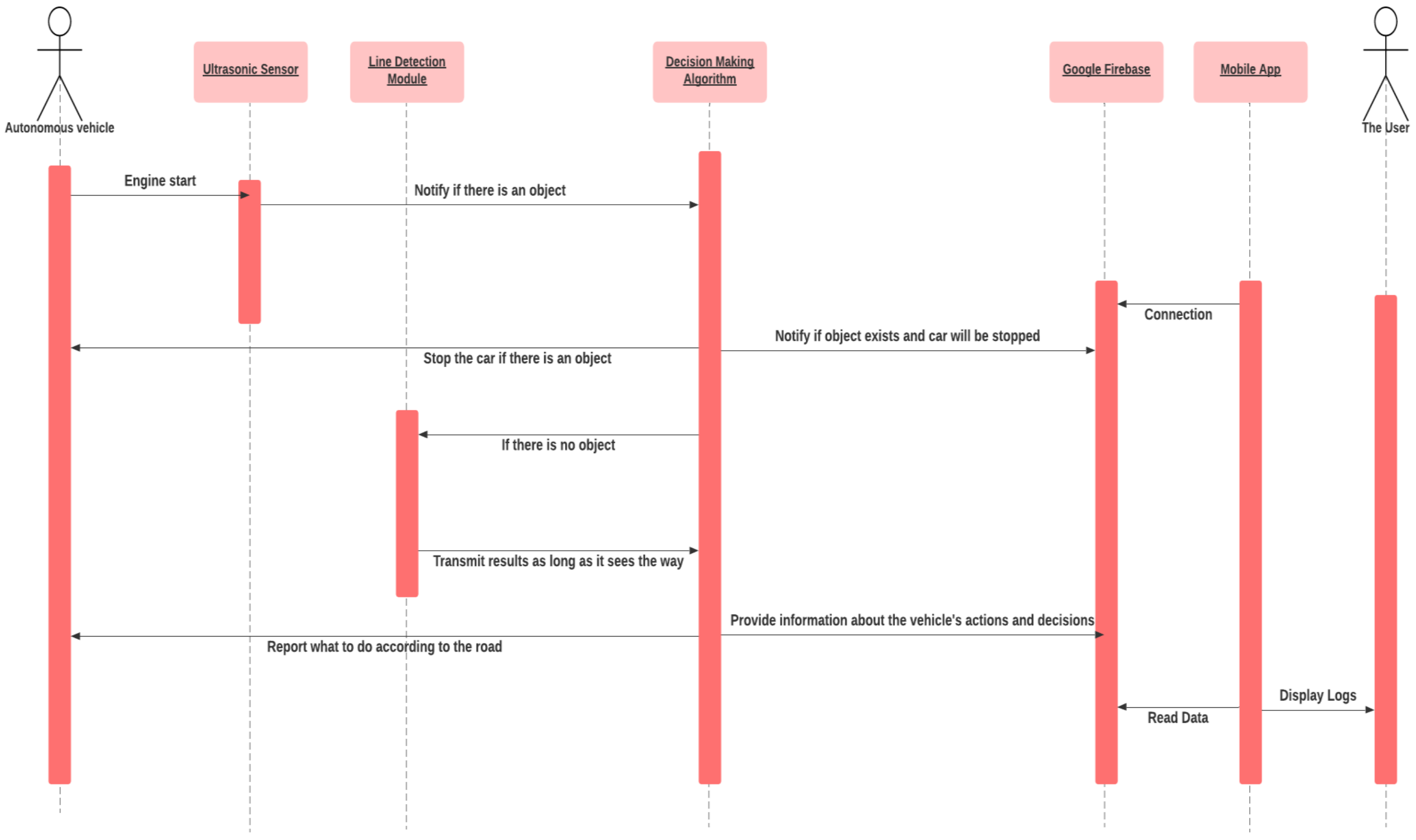

Autonomous vehicle
Ultrasonic Sensor
Line Detection Module
Decision Making Algorithm
Google Firebase
Mobile App
The User
Engine start
Notify if there is an object
Connection
Notify if object exists and car will be stopped
Stop the car if there is an object
If there is no object
Transmit results as long as it sees the way
Provide information about the vehicle's actions and decisions
Report what to do according to the road
Display Logs
Read Data

Figure 25. Sequence diagrams

3.3.5.3 Process Diagram

The main functions of our application are shown in Figure 26. As seen on the figures, our main functions are database configuration, database connection check, database reader, and its listener and include props and state definitions and assignments.

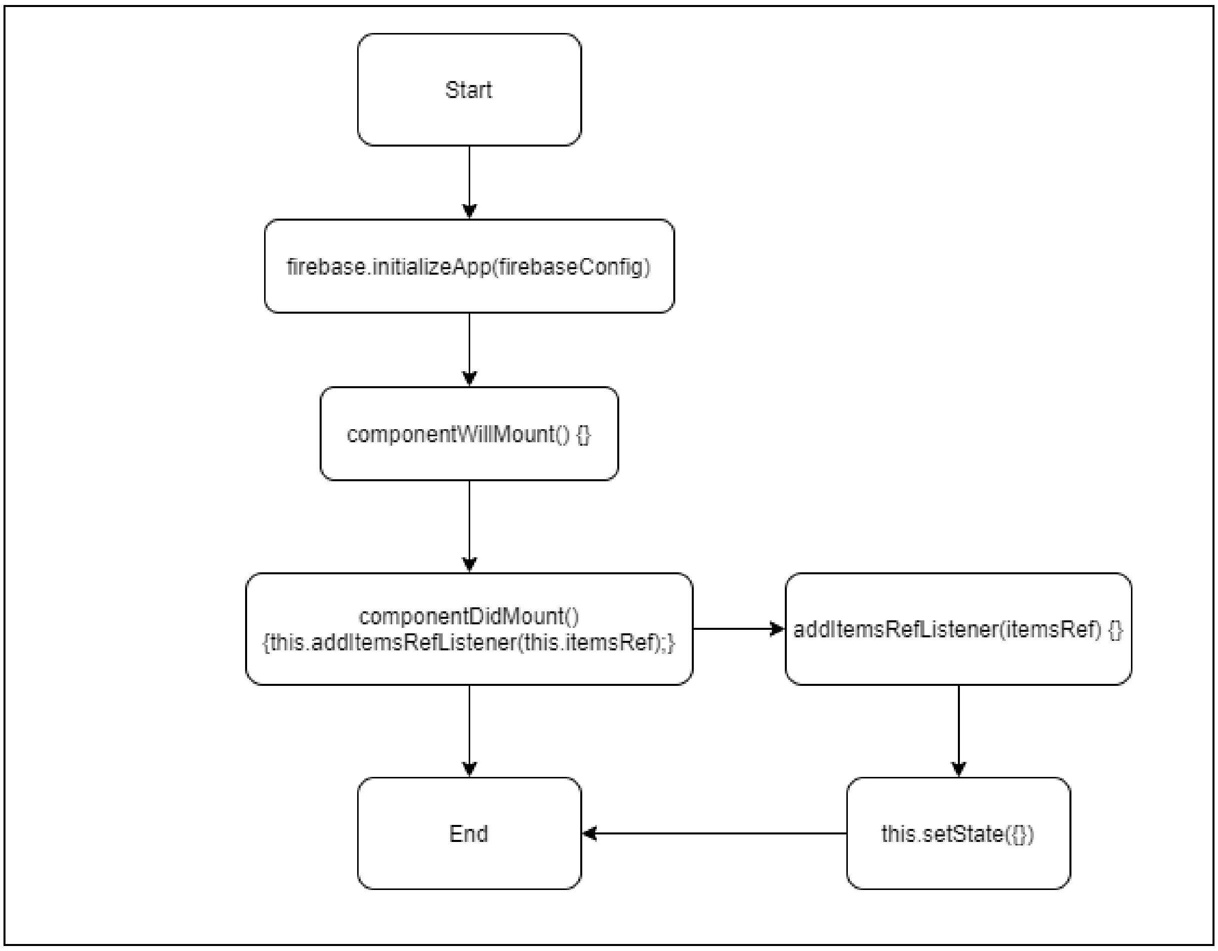

Start
firebase.initializeApp(firebaseConfig)
componentWillMount() {}
componentDidMount() {this.addItemsRefListener(this.itemsRef);}
addItemsRefListener(itemsRef) {}
this.setState({})
End

Figure 26. Process diagram

4. RESULTS

4.1 Optimization

Lane Width

Line widths are reduced on a great scale to make lines fit the camera angle. Since the required hardware for a well working autonomous car has more expenses, the codes are adapted concerning the low budget vehicle.

Sharp Turns

While entering sharp turns, vehicle FPS and lane width were not enough to detect the lines. To get by from this issue, lane widths are decreased as it mentioned at the Lane Width part and to optimize the hardware with lane model, only at the higher angle turns which is above 120, and below 60 degrees, the duty cycle is higher to get rid of the torque issue, and it runs periodically. F.e; 1 sec forward, 1 sec stop at higher angles.

Lane Swapping

When the ultrasonic sensor detects the object distance between 70 cm and 80 cm, it will reduce the speed, and then if the distance is less than 70 cm, the vehicle will stop and wait 5 seconds. After 5 seconds, if the object still in the same place, Lane Swap function will work. Lane Swap function works only for one lane for now. From now on, it swaps left lane. Lane Swap function will be developed depends on finding where the strips are. It works The Lane Swapping function turns the vehicle's wheels 45 degrees to the left and moves for about 3 seconds. Then waits 1.5 seconds and turns the wheels 60 degrees to the right, moving for 2 seconds. So it is moving into the left lane. Finally, the line Detection module is triggered and continues to follow the lane. After lane swapping, the Line Detection Module sometimes cannot detect the lines properly due to the camera angle. So line following is not working properly after lane changes.

Hooked Cables

Even the threads are managed, the cables still exceed the lane widths. Since cables exceed the lanes, vehicle threads may be hooked by an object while overtaking it. Moreover, it reduces the pace or even stops the vehicle in a significant amount.

Servo Voltage

Servo voltage is supplied by the same battery with DC motors to make Raspberry pi run with optimal values.

4.2 Verification

Hardware Test

All hardware components have been tested and verified.

Log Test

SEN part is completed, the vehicle movement logs can be seen via mobile app efficiently.

Control Test

DC and servo control functions are verified.

Line Detection

Tests have been accomplished by using a webcam and the road model. Lines are appropriately detected.

Ultrasonic Test

According to tests, the ultrasonic sensor needs to aim the object with 90 degrees. Otherwise, the sound that ultrasonic sends reflects from the object in front of it and can not measure the distance properly. The ultrasonic functions have verified.

List the cost of each component and any other costs. This can be a tabulated list, as shown in Table 8 below.

Component	**Cost (TL)**
Raspberry pi 4b+	423.74
Raspberry pi Adaptor	25
Chassis(prop chassis+rc car chassis)	184.90
L298n(x4)	40
MG945 servo	45.94
RS-380 DC motor	19.51
12 V rechargeable battery	121.06
Breadboard, cables and other materials	161.11
Black cardboard(x7)	104.30
TOTAL	**1125.56**

Table 8. Costs

5. CONCLUSION

Before COVID-19, hardware components are tested, after COVID-19 epidemic explosion, the group exposed to work from home. While doing that, majority of the requirements have accomplished but unfortunately, the internet connection problems made project harder to integrate. The integrated modules are; ultrasonic sensor, line detection, real time log application. Unfortunately, the object detection module was not integrated.

Accomplished tasks; Lane following, autonom break with respect to distance, lane changing, sending car decisions to an application. (Those are accomplished in the older chassis, due to broken steering component the hardwares are carried to the new chassis)

Accomplished skills; integrating different modules with a certain decision algorithm and make it working in a harmony.

The project hardwares can be upgraded. The components that might upgrade in the future are; board: Raspberry Pi to Jetson, sensor: ultrasonic to LIDAR, if the board is still Raspberry Pi, Google's TPU accelarator can be used to increase visual performance, webcam to ZED webcam for higher field of view.

ACKNOWLEDGMENT

We wish to thank our advisers Dr. Görkem Kar, Dr. Özge Yücel Kasap and Dr. Andrew John Beddall for their supports.

This work was partly funded by Bahçeşehir University.

REFERENCES

1. **https://www.pyimagesearch.com/2017/09/18/real-time-object-detection-with-deep-learning-and-opencv/**
2. **https://thedatafrog.com/en/human-detection-video/**
3. **http://bakeaselfdrivingcar.blogspot.com/2017/06/project-1-lane-detection-with-opencv.html**
4. **https://www.kaggle.com/soumya044/lane-line-detection**
5. **https://stackoverflow.com/questions/55208430/nonetype-error-when-using-cv2-houghlinesp-function-put-by-images**
6. **https://pysource.com/2018/03/07/lines-detection-with-hough-transform-opencv-3-4-with-python-3-tutorial-21/**
7. **https://www.geeksforgeeks.org/line-detection-python-opencv-houghline-method/**
8. **https://www.geeksforgeeks.org/opencv-real-time-road-lane-detection/**
9. **https://www.aisangam.com/blog/read-write-and-display-video-frames-with-opencv/**
10. **https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html**
11. **https://answers.opencv.org/question/179778/distance-between-2-points-in-opencv-pixel-to-cm/**
12. **https://medium.com/@denizkilinc/python-ile-veri-tan%C4%B1maya-ve-temel-i%CC%87statisti%C4%9Fe-dal%C4%B1%C5%9F-7e1028270ac**
13. **https://linuxhint.com/python_vectors_matrices_arrays/**
14. **https://www.instructables.com/id/Autonomous-Lane-Keeping-Car-Using-Raspberry-Pi-and/**
15. **https://towardsdatascience.com/deeppicar-part-4-lane-following-via-opencv-737dd9e47c96**
16. **https://python-control.readthedocs.io/en/0.8.3/steering.html**
17. **STMicroelectronics L298 Dual Full-Bridge Driver datasheet**
18. **https://www.electronicshub.org/raspberry-pi-l298n-interface-tutorial-control-dc-motor-l298n-raspberry-pi/**
19. **https://github.com/yohendry/arduino_L298N**
20. **https://firebase.google.com/docs/database/web/start**
21. **https://reactnative.dev/docs/getting-started**
22. **https://rnfirebase.io/**

23. https://reactnavigation.org/docs/drawer-based-navigation/

24. https://nodejs.org/api/documentation.html

25. https://github.com/nhorvath/Pyrebase4

26. https://electronics.stackexchange.com/questions/73863/cap-value-for-full-wave-rectifier-circuit

27. https://arxiv.org/pdf/1807.05511.pdf

28. https://www.electronicoscaldas.com/datasheet/MG995_Tower-Pro.pdf

29. https://embedded.openzeka.com/urun/mini-otonom-arac-kiti/

APPENDIX VEHICLE MOTOR CONTROL

```
import sys
import RPi.GPIO as GPIO
from time import sleep
in1 = 24
in2 = 23
ena = 25
in3 = 17
in4 = 27
enb = 22
#GPIO.setwarnings(False)
#fbdeki setup
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(in1,GPIO.OUT)
GPIO.setup(in2,GPIO.OUT)
GPIO.setup(ena,GPIO.OUT)
servoPIN = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(servoPIN, GPIO.OUT)

c = GPIO.PWM(servoPIN,50) # GPIO 17 for PWM with 50Hz
c.start(2.5) # Initialization

p=GPIO.PWM(ena,1000)
#rldeki setup
GPIO.setup(in3,GPIO.OUT)
GPIO.setup(in4,GPIO.OUT)
GPIO.setup(enb,GPIO.OUT)

def setpstart(x):
   p.start(x)
def ileri():
   GPIO.output(24,GPIO.HIGH)
   GPIO.output(23,GPIO.LOW)

def geri():
   #Arac geri gider
   GPIO.output(in1,GPIO.LOW)
   GPIO.output(in2,GPIO.HIGH)

def dur():
   #Arac durur
   GPIO.output(in1,GPIO.LOW)
   GPIO.output(in2,GPIO.LOW)

def dusuk_ilerle():
   #Arac dusuk hizda ilerler
   p.ChangeDutyCycle(15)

def orta_ilerle():
   #Arac orta hizda ilerler
   p.ChangeDutyCycle(20)
```

```
def yuksek_ilerle():
   #Arac yuksek hizda ilerler
   p.ChangeDutyCycle(25)


def servo(angle):
   angle = float(angle)
   c.ChangeDutyCycle(2+(angle/18))
   #time.sleep(0.5)
   #c.ChangeDutyCycle(0)
```

APPENDIX LINE DETECTION

```
import cv2
import numpy as np
import math

def CroppedImage(edges):

  height, width = edges.shape
  mask = np.zeros_like(edges)


  poly = np.array([[
       (0, height * 1 / 2),
       (width, height * 1 / 2),
       (width, height),
       (0, height),
    ]], np.int32)

  cv2.fillPoly(mask, poly, 255)
  croppedEdges = cv2.bitwise_and(edges, mask)


  return croppedEdges


def DrawLine(frame, lines):
    frame = np.copy(frame)
    blank = np.zeros((frame.shape[0], frame.shape[1], 3), dtype=np.uint8)
    counter = 0
    for line in lines:
        #print(counter,'--', line)
        counter+=1
        for x1, y1, x2, y2 in line:
            cv2.line(blank, (x1,y1), (x2,y2), (0, 200, 0), thickness=10)

    frame = cv2.addWeighted(frame, 1, blank, 1, 0.9)
    return frame


old_angle= 90

def LineDetectionImage(frame):
  image = frame

  height = image.shape[0]
  width = image.shape[1]

  #GrayScale = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
  MedianBlurDetection = cv2.medianBlur(image,5)
  CannyDetection = cv2.Canny(image, 100, 200)
  try:
      CroppedImageFunction = CroppedImage(CannyDetection)
      rho=1
      theta=np.pi/180
      lines = cv2.HoughLinesP(CroppedImageFunction,
                  rho,
                  theta,
                  threshold=30,
                  lines=np.array([]),
                  minLineLength=8,
                  maxLineGap=4)

      #print('0',lines[0][0])
      #print('1',lines[1][0])
      #print('2',lines[2][0])
      #print('3',lines[3][0])
      #total_lines =

      _, _, left_side, _ = lines[0][0]
      _, _, right_side, _ = lines[1][0]

      mid = int(width / 2)
      x_side = (left_side + right_side) / 2 - mid
      y_side = int(height / 2)

      Radian = math.atan(x_side / y_side)
      MiddleAngle = int(Radian * 180.0 / math.pi)
      SteeringAngle = MiddleAngle + 90




      WithImage = DrawLine(image, lines)
      global old_angle
      old_angle=SteeringAngle
      response = [WithImage,SteeringAngle]
      return response
  except Exception as e:
      print(e)
      return [frame,old_angle]
```

APPENDIX MOBILE APPLICATION

```
import React, {Component} from 'react';
import {
  TouchableOpacity,
  Button,
  View,
  Text,
  StyleSheet,
  FlatList,
} from 'react-native';
import * as firebase from 'firebase/app';
import 'firebase/database';
import {WebView} from 'react-native-webview';
import DatePicker from 'react-native-datepicker';

console.disableYellowBox = true;

const firebaseConfig = {
  apiKey: '-',
  authDomain: '-',
  databaseURL: '-',
  projectId: '-',
  storageBucket: '-',
  messagingSenderId: '-',
  appId: '-',
  measurementId: '-',
};
if (!firebase.apps.length) {
  firebase.initializeApp(firebaseConfig);
}
const data = [];

export default class HomeScreen extends Component {
  constructor(props) {
    super(props);
    this.itemsRef = firebase.database().ref('Car');
    this.state = {
      arrData: [],
      date: '',
      font: styles.post,
    };
  }

  addItemsRefListener(itemsRef) {
    itemsRef.on('value', snap => {
      if (snap.val()) {
        itemsRef.once('value').then(snapshot => {
          const items = [];
          let counter = 0;
          snapshot.forEach(child => {
            if (this.state.date !== '') {
              if (this.state.date === child.val().date) {
                counter++;
                items.push({
                  date: child.val().date,
              time: child.val().time,
              output: child.val().output,
            });
          }
          this.setState({font: styles.post});
        } else {
          items.push({
            date: child.val().date,
            time: child.val().time,
            output: child.val().output,
          });
          this.setState({font: styles.post});
        }
      });

      if (this.state.date !== '' && counter === 0) {
        items.push({
          date: 'None',
          time: 'None',
          output: 'No data on this day',
        });
        this.setState({font: styles.empty});
      }
      this.setState({arrData: items.reverse()});
    });
  }
 });
}

componentWillMount() {
 //this.getData();
}

componentDidMount() {
 this.addItemsRefListener(this.itemsRef);
}

renderPost = post => {
 return (
  <View style={styles.feedItem}>
   <View style={{flex: 1}}>
    <View
     style={{
      flexDirection: 'row',
      justifyContent: 'space-between',
      alignItems: 'center',
     }}>
     <View>
      <Text style={styles.index}>
       Date: {post.date} - Time: {post.time}
      </Text>
     </View>
    </View>
    <Text style={this.state.font}>Log: {post.output}</Text>
    <View style={{flexDirection: 'row'}} />
   </View>
  </View>
  );
};

render() {
  return (
    <View style={styles.container}>
      <View style={{flex: 5}}>
        {/*<WebView*/}
        {/*  source={{*/}
        {/*    uri: 'https://www.youtube.com/embed/21X5lGlDOfg',*/}
        {/*  }}*/}
        {/*/>*/}
      </View>
      <View
        style={{
          flexDirection: 'row',
          justifyContent: 'center',
          alignItems: 'center',
        }}>
        <DatePicker
          style={{
            width: 310,
            backgroundColor: '#909090',
            marginTop: 8,
            marginBottom: 2,
          }}
          date={this.state.date}
          mode="date"
          placeholder="Select Date"
          format="DD-MM-YYYY"
          minDate="01-01-2020"
          maxDate="31-12-2020"
          confirmBtnText="Confirm"
          cancelBtnText="Cancel"
          customStyles={{
            dateIcon: {
              position: 'absolute',
              left: 2,
              top: 4,
              marginLeft: 0,
            },
            dateInput: {
              marginLeft: 0,
            },
            placeholderText: {
              fontSize: 15,
              color: '#000',
            },
            dateText: {
              fontSize: 15,
              color: '#000',
            },
          }}
          onDateChange={date => {
            this.setState({date: date});
            this.addItemsRefListener(this.itemsRef);
          }}
          />
          <TouchableOpacity
            style={{
              marginLeft: 10,
              height: 30,
            }}>
            <Button
              title="Reset"
              color="#ff5c5c"
              onPress={date => {
                this.setState({date: ''});
                this.addItemsRefListener(this.itemsRef);
              }}
            />
          </TouchableOpacity>
        </View>
        <View style={{flex: 4}}>
          <FlatList
            style={styles.feed}
            data={this.state.arrData}
            renderItem={({item}) => this.renderPost(item)}
            keyExtractor={this.keyExtractor}
            showsVerticalScrollIndicator={false}
          />
        </View>
      </View>
    );
  }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#EBECF4',
  },
  feed: {
    marginHorizontal: 16,
  },
  feedItem: {
    backgroundColor: '#FFF',
    borderRadius: 5,
    padding: 8,
    flexDirection: 'row',
    marginVertical: 8,
  },
  index: {
    fontSize: 15,
    fontWeight: '500',
    color: '#000',
  },
  post: {
    marginTop: 16,
    fontSize: 15,
    color: '#000',
  },
  empty: {
    marginTop: 16,
    fontSize: 15,
    color: '#ff5c5c',
    fontWeight: 'bold',
  },
});
```

APPENDIX VEHICLE LOG WRITE

```
import pyrebase
from datetime import datetime

configDB = {
   "apiKey": '-',
   "authDomain": '-',
   "databaseURL": '-',
   "projectId": '-',
   "storageBucket": '-',
   "messagingSenderId": '-',
   "appId": '-',
   "measurementId": '-',
}

firebase = pyrebase.initialize_app(configDB)
db = firebase.database()

dt_date = datetime.now().strftime("%d-%m-%Y")
dt_time = datetime.now().strftime("%H:%M:%S")

data = {'date': dt_date, 'output': 'Test Log', 'time': dt_time, }
db.child("Car").child(db.generate_key()).set(data)

# last_record = db.child("Car").order_by_key().limit_to_last(1).get().val()
# print(last_record)
```

APPENDIX MAIN

```
import arabakontrol
import time
import linev2
from SensorClass import distanceSensor
import imutils
import cv2
from imutils.video import VideoStream
import math
import firebasedlog


vs = VideoStream(src=0,resolution=(50,50)).start()
sensor = distanceSensor(6,5)
last_angle = 90
arabakontrol.servo(last_angle)

def roundup(x):
   return int(math.ceil(x / 10.0)) * 10

def turning_with_angle(angle):
   arabakontrol.servo(angle)
   arabakontrol.setpstart(25)
   arabakontrol.dur()
   arabakontrol.ileri()

def straight_forward(angle):
   arabakontrol.servo(angle)
   arabakontrol.setpstart(18)
   arabakontrol.dur()
   arabakontrol.ileri()

def turn_with_time(angle):
   stop_counter = 0
   go_counter = 0
   arabakontrol.servo(angle)
   while stop_counter < 5:
      arabakontrol.dur()
      stop_counter +=1
      time.sleep(0.05)
   stop_counter = 0
   while go_counter < 5:
      arabakontrol.setpstart(33)
      arabakontrol.ileri()
      go_counter+=1
      time.sleep(0.05)
   go_counter=0

def sensor_control():
   return sensor.get_distance()

def move_back(angle):
   geri_counter = 0
   arabakontrol.setpstart(20)
   arabakontrol.dur()
```

```
    arabakontrol.servo(angle)
    while geri_counter < 15:
        arabakontrol.geri()
        geri_counter+=1
        time.sleep(0.05)
        print(geri_counter,'going back')

def slow_forward(angle):
    arabakontrol.servo(angle)
    arabakontrol.setpstart(13)
    arabakontrol.ileri()

def laneSwap(direction):
    arabakontrol.setpstart(25)
    swapcounter = 0
    secondCounter = 0
    stopCounter = 0
    if direction == 1:
        arabakontrol.servo(135)
        while swapcounter <= 28:
            arabakontrol.ileri()
            swapcounter += 1
            time.sleep(0.05)
            print(swapcounter,'going forward')
                        firebasedblog.saveLog('going forward')
        arabakontrol.servo(30)
        while stopCounter <=15:
            arabakontrol.dur()
            stopCounter +=1
            time.sleep(0.05)
            print(stopCounter, 'Stopping')
                        firebasedblog.saveLog('Stopping')
        while secondCounter <= 20:
            arabakontrol.ileri()
            secondCounter +=1
            time.sleep(0.05)
            print(secondCounter, 'I am okay, lets go!!')
                        firebasedblog.saveLog('I am okay, lets go!!')

    else:
        arabakontrol.servo(45)
        while swapcounter <= 25:
            arabakontrol.ileri()
            swapcounter += 1
            time.sleep(0.05)
            print(swapcounter,'going forward')
                        firebasedblog.saveLog('going forward')
        arabakontrol.servo(30)
        while stopCounter <=15:
            arabakontrol.dur()
            stopCounter +=1
            time.sleep(0.05)
            print(stopCounter, 'Stopping')
                        firebasedblog.saveLog('Stopping')
        while secondCounter <= 15:
            arabakontrol.ileri()
            secondCounter +=1
        time.sleep(0.05)
        print(secondCounter, 'I am okay, lets go!!')
                    firebasedblog.saveLog('I am okay, lets go!!')

def angle_optimizer(angle):
    global last_angle
    if last_angle < 160 and last_angle > 20:
        if abs(last_angle - angle) <= 20:
            return angle
        else:
            if last_angle > 90 :
                angle -= 10
                return angle
            elif last_angle <= 90:
                angle += 10
                return angle

    else:
        return angle

waitcounter = 0

while True:
    frame = vs.read()
    lanes,steer_angle = linev2.LineDetectionImage(frame)
    steer_angle = roundup(steer_angle)
    print('Stable Angle',steer_angle)
    cv2.imshow('Yol',lanes)
    print('Line Angle',steer_angle)
    #if steer_angle != 270 and steer_angle <160 and steer_angle>20:
        #print('Stable Angle',steer_angle)
        #steer_angle = angle_optimizer(steer_angle)
        #last_angle = steer_angle #stored the needed value

    if steer_angle <= 120 and steer_angle >= 60:
        distance = sensor_control()
        print('Distance :',distance)
        if distance >= 80:
            straight_forward(steer_angle)
        elif distance <= 80 and distance >= 70:
            print('An object detected. Slowing...')
                    firebasedblog.saveLog
            slow_forward(steer_angle)
        elif distance < 70:
            print('An object detected in close range. Stopped...')
                firebasedblog.saveLog('An object detected in close range. Stopped...')
            arabakontrol.dur()
            waitcounter += 1
            print(waitcounter)
            if waitcounter == 30:
                laneSwap(1)
                waitcounter = 0
    elif (steer_angle > 120 or steer_angle < 60) and steer_angle != 270 and steer_angle <161 and steer_angle>21:
        turn_with_time(steer_angle)
    else:
        #print('angle does not seem normal, check it, go back',steer_angle)
        #move_back(last_angle)
        turn_with_time(last_angle)
        pass

    key = cv2.waitKey(1) & 0xFF
    if key == ord("q"):
        break
```