S-Car

Project Engineering

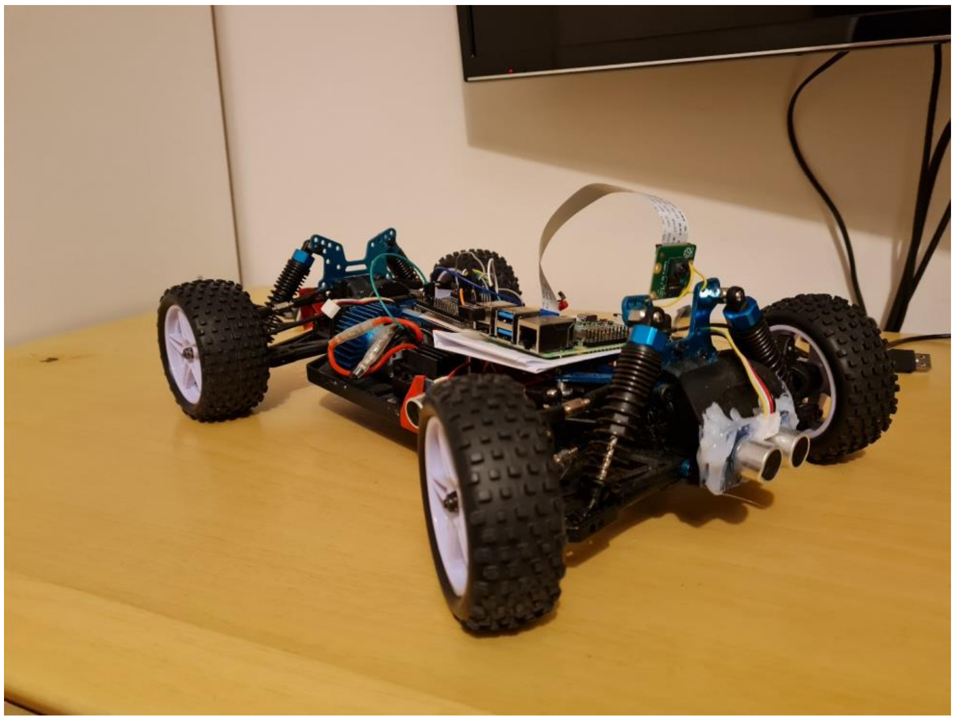
Mohamed Otaki – G00346067

Michelle Lynch – Supervisor

Bachelor of Engineering (Honours) Software and Electronic Engineering

Galway-Mayo Institute of Technology

2020/2021



Declaration

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering (Honours) in Software & Electronic Engineering at Galway-Mayo Institute of Technology.

This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

Mohammad Otaki

Acknowledgements

Firstly, I would like to thank my lectures at GMIT for their hard work and commitment throughout the year. Secondly, my supervisor Michelle Lynch, who guided me in the right direction when ever I went of path. Thirdly, my colleagues, who continuously supported me and kept me focused during this time. Finally, I would like to thank my family and friends, who encouraged me throughout my time in college.

Table of Contents

1 Summary 7
2 Poster 8
3 Introduction 9
4 S-Car 9
5 Goals 9
5.1 Computer vision 10
6 Project Architecture 11
7 Project Plan 12
7.1 Research 12
8 Testing 13
9 Positioning of components 13
10 Assembly 14
11 Software 16
11.1 Android Application 16
11.1.1 Use Case Diagram 17
11.1.2 SOLID Principles 18
11.1.3 SQLite Database 20
11.1.4 Bluetooth 21
11.1.5 Encryption 22
11.1.6 MapQuest 23
11.2 Autonomous Driving 24
11.3 Parking Assist 25
11.4 Data Store .... **Error! Bookmark not defined.**

11.5 SQL Database .... **Error! Bookmark not defined.**

11.6 Server 25

12 Challenges 26

13 Recommendations 26

14 Conclusion 27

15 References 28

16 Appendix 28

16.1.1 Pictures of the car .... **Error! Bookmark not defined.**

16.2 Software used 28

Table of Figure

Figure 1 - Project Poster............ 8

Figure 2 Architecture Diagram............ 11

Figure 3 14

Figure 4 - Motor Drive Code 15

Figure 5 App Login and Home screen 16

Figure 6 - Single Responsibility Example............ 18

Figure 7 - Open Closed Principle Example 19

Figure 8 - SQLite Table 20

Figure 9 - Storing Data in SQLite 20

Figure 10 - Bluetooth Connection Code 21

Figure 11 – MapQuest Activity Result 23

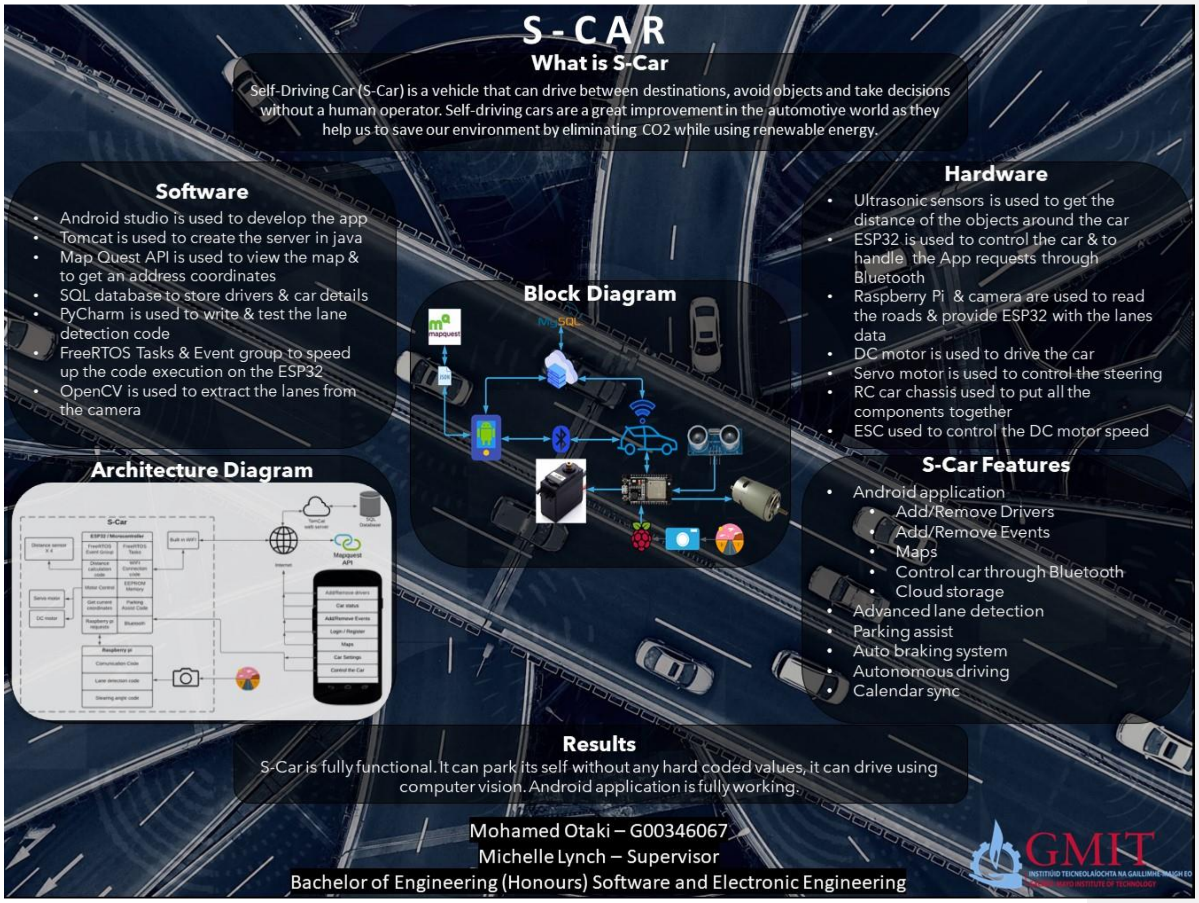
Figure 12 - Lane Detection Result............ 24

1 Summary

This report outlines the feasibility of the S-Car. This report discusses how I developed the S-Car. The main goal was to construct a self-driving car for my final year college project. The requirements of the car were that it should be contributing to the environment and eco-friendly. The main objective of the car was that it should be autonomous driven. I have achieved the objectives I set out in the report. The report also outlines the the challenges I faced, and how I overcame them.

The report discusses the different stages and steps I took to develop the S-Car. I got a basic RC Chassis and developed it using multiple sensors and used software's such as android studio and Arduino. The car is fully functional, and it can be controlled by a mobile phone app that I developed. The app gives the user the ability to control and monitor the S-Car. The technologies I used on the S-Car are Computer Vision, Free RTOS, Tomcat Server, SQL, Android studio, Python, Bluetooth, Wi-Fi, Android, Java, C++, OpenCV, PyCharm and MapQuest.

2 Poster



S-CAR

What is S-Car

Self-Driving Car (S-Car) is a vehicle that can drive between destinations, avoid objects and take decisions without a human operator. Self-driving cars are a great improvement in the automotive world as they help us to save our environment by eliminating CO2 while using renewable energy.

Software

- Android studio is used to develop the app
- Tomcat is used to create the server in java
- Map Quest API is used to view the map & to get an address coordinates
- SQL database to store drivers & car details
- PyCharm is used to write & test the lane detection code
- FreeRTOS Tasks & Event group to speed up the code execution on the ESP32
- OpenCV is used to extract the lanes from the camera

Block Diagram

Hardware

- Ultrasonic sensors is used to get the distance of the objects around the car
- ESP32 is used to control the car & to handle the App requests through Bluetooth
- Raspberry Pi & camera are used to read the roads & provide ESP32 with the lanes data
- DC motor is used to drive the car
- Servo motor is used to control the steering
- RC car chassis used to put all the components together
- ESC used to control the DC motor speed

Architecture Diagram

S-Car Features

- Android application
  - Add/Remove Drivers
  - Add/Remove Events
  - Maps
  - Control car through Bluetooth
  - Cloud storage
- Advanced lane detection
- Parking assist
- Auto braking system
- Autonomous driving
- Calendar sync

Results

S-Car is fully functional. It can park its self without any hard coded values, it can drive using computer vision. Android application is fully working.

Mohamed Otaki – G00346067
Michelle Lynch – Supervisor
Bachelor of Engineering (Honours) Software and Electronic Engineering

GMIT

Figure 1 - Project Poster

3 Introduction

As part of my final year project, I had to do a project of my choice. I went down the lane of automotive as this industry really excites me and I have a huge interest in this sector. I decided to build a self-driving car as they are considered the future of automotive. It is still in the early stages of development. Companies like Tesla, have significant control and are investing hugely into self-driving cars. In 2015, they released their first self-driving car. It was considered a new concept and it was very basic. This is where I get my inspiration from. The self-driving system, I am developing is based on Tesla but with some improvements that I think is viable for the success of them. I have named my car, The S-Car.

4 S-Car

Self-Driving Car (S-Car) is a vehicle that can drive between destinations, avoid objects, and take decisions without a human operator. Self-driving cars are a great improvement in the automotive industry. They play a vital role in keeping our environment clean as they only use renewable energy to run. They do not produce an CO2 emission. Self-Driving Cars also impact to the safety of the road users. It has significantly reduced accidents caused by drunk drivers and drivers that use their phones. The big advantage of the S-Car is that it has high processing power by using Free RTOS Tasks. This gives the driver the ability to control the car via an app while the car is operational. It provides high comfort and safety to the driver.

5 Goals

The main goal of carrying out this project was to make driving safer and easier. Self-driving cars are more stable and safer on the road as they can access hazards and can react quicker than a human. Also, they are equipped with many sensors that contribute to the safety of the driver and to the people outside.

Another goal of the S-Car is to make the environment cleaner. S-Car is ecofriendly as it uses renewable energy in results less carbon footprint. The S-Car has zero emissions and does not have any impact on the environment.

The S-Car includes autonomous driving by using advance lane detection system, this helps the car to steer itself on the road.

6 Background

6.1 Computer vision

Computer vision is a field of artificial intelligence that trains computers to interpret and understand the visual world. it uses a variety of digital images, videos, and sensors to identify objects such as lane on a road and pedestrians or any other hazards on the road. It also has the ability after interpreting to react. For example, in a self-driving car the computer vision will identify the lanes on the road and keep the car within them. It helps to steer the car in between the lanes on a road by using Open CV Library.

Comparing the computer vision to my S-car. The S-Car uses the Hough Line Transform algorithm from Open CV Library to get the lines in the current frame. Then it finds the left line and the right lines by calculating the slope. If a line has negative slope, it will add it to the left lines array and if it has positive slope, it will add it to the right lines array. After separating the left lines by the right lines then it will average out both lines to get the best result.

The concept of self-driving car first came in 1939s but back then there was not enough development in technology to produce it [1]. It was only considered a concept. In today's world, the technology has made a significant advancement and all the automotive car makers are in a race to build a self-driving car.

7 Project Architecture

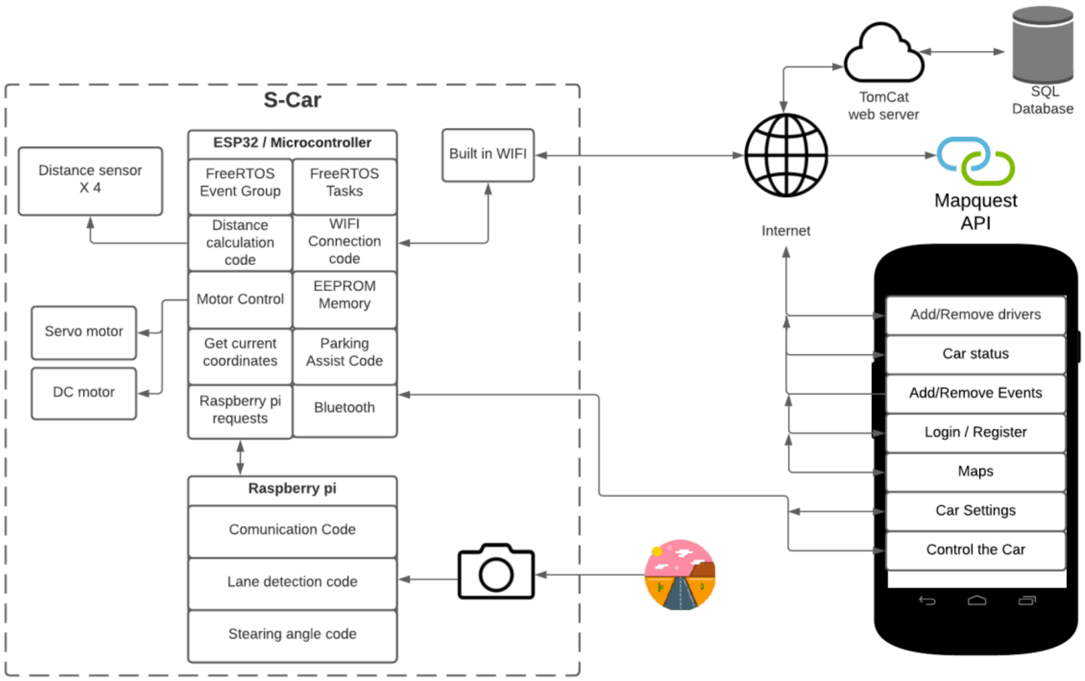


Figure 2 Architecture Diagram

8 Project Plan

As part of my final year in college, I had to do a project of my choice. I decided to build a Self-driving car (S-Car). I decided to work on this project as this will impact the betterment of the society.

In this project I followed Agile methodology as Agile is flexible of accepting any changes and has faster development life cycle.

8.1 Research

The research was the easy part of the project. Most of the information was widely available online. Firstly, I had chosen a chassis for the car. I had a number of options available to me. I choose the RC Car Chassis as it portraits a real car. Then, I had to select the components they would require for the operational of the car. There are a number of components. This is when I faced my first challenge.

Commented [MO-S1]: Add a link to this car or short explanation of the car or an image

While I was doing research, I came to realise that I would require two boards as one board was not efficient enough. Using one board will disrupt the operational efficient of the S-Car and it will cost more.

The S-Car required computer vision, which was new to me. This required me to learn new set of programming skills before I officially start working on my project, I had to make sure that I understand all the aspects of phyton programming language.

The next step of this project was to make sure that the components are inline with the feasibility and a viability of the S- Car. It was difficult to pick the right component which is compatible with the rest of car. For example, I had issues with the motor. The main issue with the DC motor I ordered was not able to carry the load of the car, so I decided to pick another motor. This forced me to go back to the research stage of the project and select a new motor that met my requirements. This affected the timeline of my project because of Covid it was hard to get fast delivery.

9 Testing

When I received all the components, I had to check that they all were operational, and they were compatible with the S car. The main components I had to test were the ultrasonic sensors, DC Motor, Server motor, the ESP32, Raspberry Pi and the Electronic Speed Control. After I was stratified, I moved on the next stage, but this phase did encounter challenges.

The Ultrasonic sensors were tested by writing enough code to get the distance value. I measured it in intervals of 30cm making sure the distance was correct. I double-checked the figure by measuring it with a measuring tape.

The concept of testing the DC Motor and the ESC was the same as the ultrasonic sensors, write enough code to spin the motor in each direction.

10 Positioning of components

After the testing stage was completed, I had to assemble the car making sure all the components worked with each other. Finding the right position on the car for the components was one of the hardest parts of the project. It required me to fix the components in the exact place. If I did not get it in the exact position the reading from the sensors would not be accurate. One advantage of locating the position the components on the car was that it made the wiring easier and tidier. It saved me a lot of time as it only required me to attach the wires from the components to the boards.

The ultrasonic sensor was the most difficult part to place on the car. The ultrasonic sensor had to be a certain distance apart so it would not interfere with each other. Also, the positioning of this was very important as this was a main feature it would have a major impact of the performance on the car.

11 Assembly

The first step was to fix the ESP32 position on the top of the chassis. This made it easier to have access to all ESP32 pins and hide the wires as it is showing in Figure 4. The power to the ESP32 is gotten by connecting 7.2V NI-MH battery to the Vin pin.

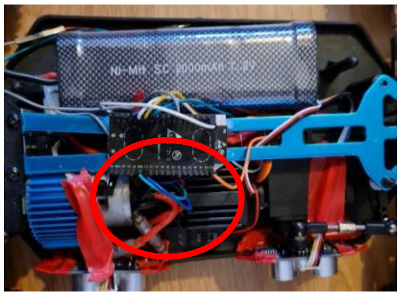
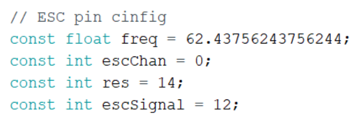


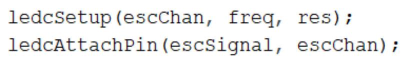
Figure 4 - ESP32 Positioning

The next step was to put in the DC motor and the ESC on the car chassis. The DC motor is responsible for the movement and ESC controls the speed of the car. I powered the ESC with 7.2V direct from the battery. These components were vital as they carried out the main functions of the car.

After each step I was testing the components and taking notes. To control the speed of the DC Motor, Pulse Width Modulation (PWM) had to be used. This is a method to create a square wave to switch the motor between on and off and as a result the speed can be controlled. To use the PWM on the ESP32 a channel must be configured with the frequency and the resolution. Then attach the channel to the required pin as showing in figure 3.



```
// ESC pin cinfig
const float freq = 62.43756243756244;
const int escChan = 0;
const int res = 14;
const int escSignal = 12;
```



```
ledcSetup(escChan, freq, res);
ledcAttachPin(escSignal, escChan);
```

Figure 3

The next step was to create functions to control the operation of the DC motor. I created serval functions for the DC motor as each function control the motor differently.

Servo Motor

The servo motor is responsible for the steering of the car. The coding concept of the servo motor is similar to the DC motor as I had to use the PWM. After the coding and assembly of these components, I tested them out. The results were both functioning as it should be.

Ultrasonic Sensors

Ultrasonic sensors are used to get the distance of the objects around the car. I used four sensors on the car one at each side. They were equally distanced from each other, so they do not interfere with one another. After positioning all the sensors on the chassis, I powered them with 3.3V from the ESP32

Raspberry Pi & Camera

This is one of the main boards which is responsible for the autonomous driving. This is one of the main components I was working with. this boards gets the image from the camera and process and output the data to the ESP32 by serial communication. The board is powered up by 7.2V battery to the 5V Pin. The camera is attached to the board and placed on the front of the car.

Figure 4 - Motor Drive Code

12 Software

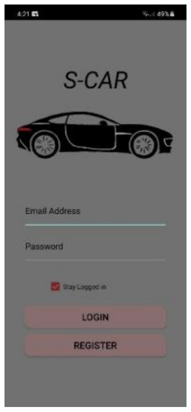
12.1 Android Application

S-Car application is an app. The S-Car application was built using android studio and Java as the main coding language. It gives the driver the ability to access the car using their mobile phone device via Bluetooth. The app allows the driver to control the car remotely and connect the car to the Wi-Fi if needed. It also allows it to add more drivers in case someone new is driving the car. They can customise it to their needs and even add restrictions like time limits.

The owners can also add, update, or remove events using the S-Car application. This feature has the ability to turn on the car and set the temperature before the driver enters the car. It saves the destinations of the event when it occurs.

The S-Car app uses treading to connect to the server. It also uses treading for connecting with the Bluetooth. When the app is not connected to the online server the user can only use the app in offline mode. This restricts some of the features available from the app. It will only allow to control the car.

The app has a simple and elegant design. One of the features, I like on the app is to check the status of the car. This displays the battery, temperature and if the car is locked or unlocked. It also recommends when the car should be serviced.



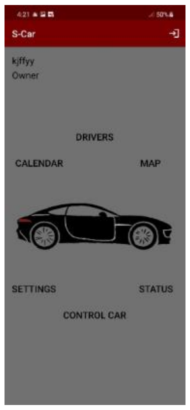
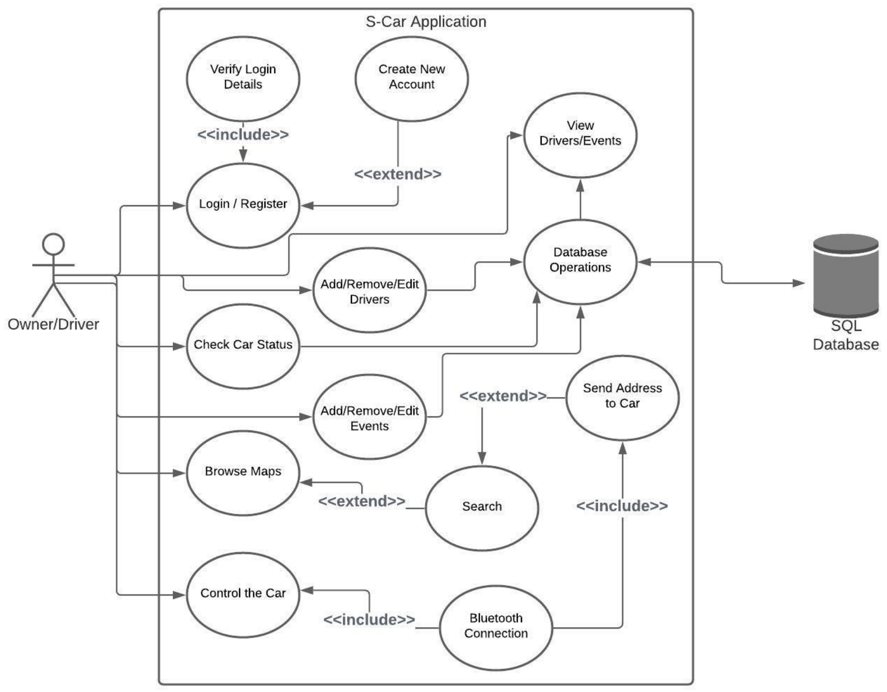


Figure 5 App Login and Home screen

12.1.1 Use Case Diagram

This diagram shows the functionality of the S-Car application. It outlines the features available to the driver.

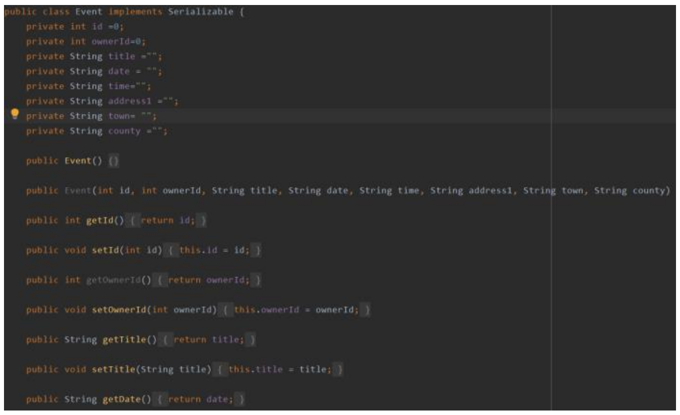


12.1.2 SOLID Principles

While coding the application I followed two of the SOLID principles, Single Responsibility Principle and Open Closed Principle.

Single Responsibility Principle

Single Responsibility Principle mean that every class or function in an application should have one responsibility. Figure 6 shows the Event class and as seen the class is only responsible for storing an event detail.

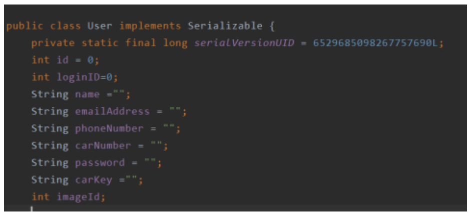


```java
public class Event implements Serializable {
    private int id =0;
    private int ownerId=0;
    private String title ="";
    private String date = "";
    private String time="";
    private String address1 ="";
    private String town= "";
    private String county ="";

    public Event() {}

    public Event(int id, int ownerId, String title, String date, String time, String address1, String town, String county)

    public int getId() { return id; }

    public void setId(int id) { this.id = id; }

    public int getOwnerId() { return ownerId; }

    public void setOwnerId(int ownerId) { this.ownerId = ownerId; }

    public String getTitle() { return title; }

    public void setTitle(String title) { this.title = title; }

    public String getDate() { return date; }
```

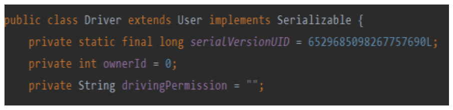
Figure 6 - Single Responsibility Example

Open Closed Principle

Open Closed Principle means that the classes should be open for an extension but closed for modification. This allow the classes to be extended without modifying their source code. In my application I used the Open Closed Principle as its showing in figure 7. The driver class extends the user as the additional drivers have time limit to their accounts. This helped me to treat the driver as a user and write less code.



```java
public class User implements Serializable {
    private static final long serialVersionUID = 6529685098267757690L;
    int id = 0;
    int loginID=0;
    String name ="";
    String emailAddress = "";
    String phoneNumber = "";
    String carNumber = "";
    String password = "";
    String carKey ="";
    int imageId;
```

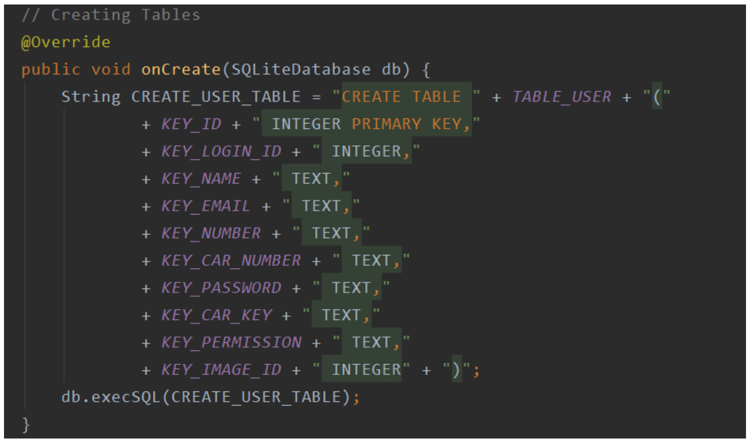


```java
public class Driver extends User implements Serializable {
    private static final long serialVersionUID = 6529685098267757690L;
    private int ownerId = 0;
    private String drivingPermission = "";
```

Figure 7 - Open Closed Principle Example

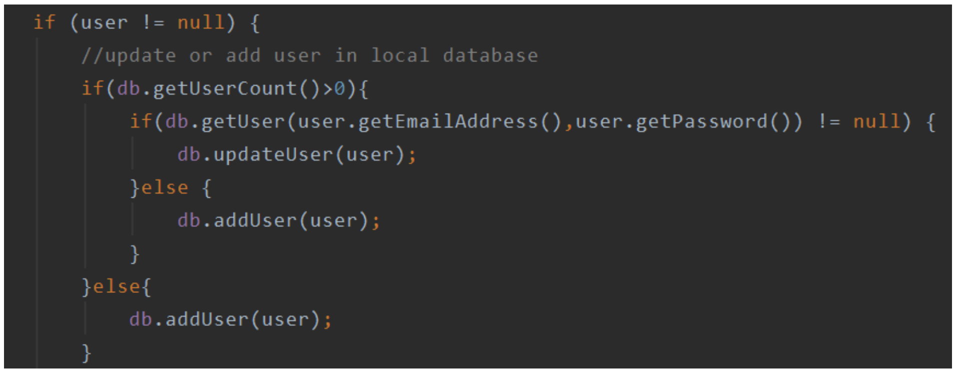
12.1.3 SQLite Database

SQLite software library that provides a relational database management system. In my application I created one table to store the users login details and the car key as showing in figure 8. This will allow the users to have access to the car without having internet connection. SQLite database will store the user details when the user tries to successfully login while having internet connection. Then the main database will return a user object to store it in SQLite. If the user already exists, the user details will be updated as showing in figure 9.



```java
// Creating Tables
@Override
public void onCreate(SQLiteDatabase db) {
    String CREATE_USER_TABLE = "CREATE TABLE " + TABLE_USER + "("
            + KEY_ID + " INTEGER PRIMARY KEY,"
            + KEY_LOGIN_ID + " INTEGER,"
            + KEY_NAME + " TEXT,"
            + KEY_EMAIL + " TEXT,"
            + KEY_NUMBER + " TEXT,"
            + KEY_CAR_NUMBER + " TEXT,"
            + KEY_PASSWORD + " TEXT,"
            + KEY_CAR_KEY + " TEXT,"
            + KEY_PERMISSION + " TEXT,"
            + KEY_IMAGE_ID + " INTEGER" + ")";
    db.execSQL(CREATE_USER_TABLE);
}
```

Figure 8 - SQLite Table

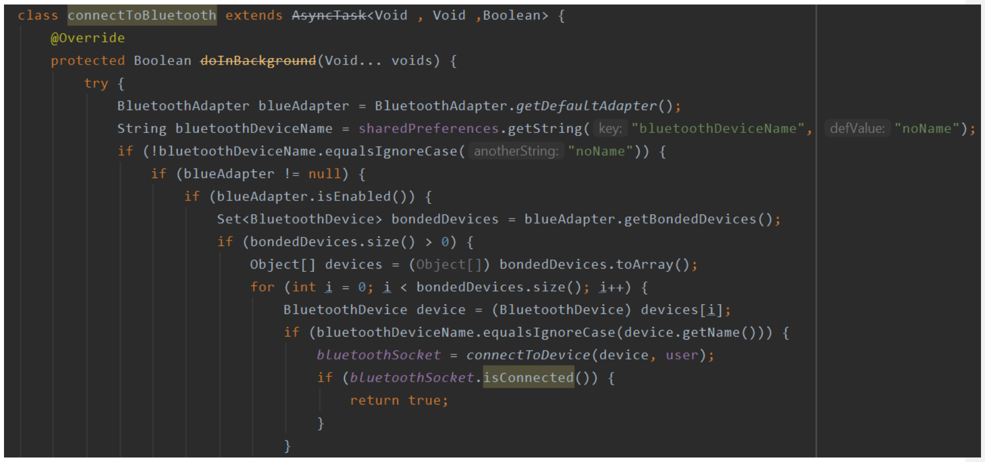


```java
if (user != null) {
    //update or add user in local database
    if(db.getUserCount()>0){
        if(db.getUser(user.getEmailAddress(),user.getPassword()) != null) {
            db.updateUser(user);
        }else {
            db.addUser(user);
        }
    }else{
        db.addUser(user);
    }
```

Figure 9 - Storing Data in SQLite

12.1.4 Bluetooth

Bluetooth is a wireless technology that uses the radio frequency to share data between two devices. I used Bluetooth in my project to allow the users to control their cars using their phones. The user can control the car, activate the parking assist, activate the autonomous driving, or connect the car to the Wi-Fi. The application will try to establish Bluetooth connection when the user login successfully. It will check the Shared Preferences for any saved devices, if a device name was found the application will establish the connection with that device and save the Bluetooth Socket, if not it will ask the user to connect to the car in settings. I used AsyncTask to run the Bluetooth connection code as the connection may take some time to be established. AsyncTask helps to run the required code in the background without interfering with the user interface, this helps my application to have quick user interface without any delay.



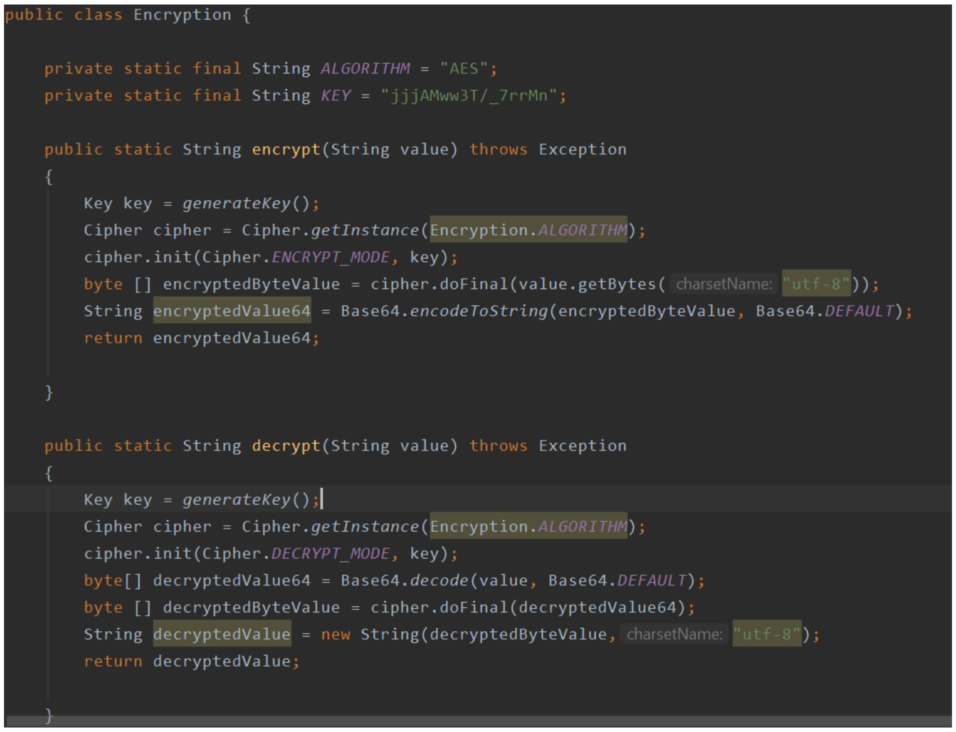
```
class connectToBluetooth extends ~~AsyncTask~~<Void , Void ,Boolean> {
    @Override
    protected Boolean ~~doInBackground~~(Void... voids) {
        try {
            BluetoothAdapter blueAdapter = BluetoothAdapter.getDefaultAdapter();
            String bluetoothDeviceName = sharedPreferences.getString( key: "bluetoothDeviceName",  defValue: "noName");
            if (!bluetoothDeviceName.equalsIgnoreCase( anotherString: "noName")) {
                if (blueAdapter != null) {
                    if (blueAdapter.isEnabled()) {
                        Set<BluetoothDevice> bondedDevices = blueAdapter.getBondedDevices();
                        if (bondedDevices.size() > 0) {
                            Object[] devices = (Object[]) bondedDevices.toArray();
                            for (int i = 0; i < bondedDevices.size(); i++) {
                                BluetoothDevice device = (BluetoothDevice) devices[i];
                                if (bluetoothDeviceName.equalsIgnoreCase(device.getName())) {
                                    bluetoothSocket = connectToDevice(device, user);
                                    if (bluetoothSocket.isConnected()) {
                                        return true;
                                    }
                                }
```

Figure 10 - Bluetooth Connection Code

12.1.5 Encryption

Encryption is a very important step as it helps to protect the user's data locally and on the internet. It is done by converting plain text data into ciphertext data by using Advanced Encryption Standard (AES) algorithm and a special key. I used encryption to encrypt all the user's data and to store it safely in the database. Encryption also adds to the safety of the car as it prevents the car being hacked.

Figure 10 shows the encryption class [2].



```
public class Encryption {

    private static final String ALGORITHM = "AES";
    private static final String KEY = "jjjAMww3T/_7rrMn";

    public static String encrypt(String value) throws Exception
    {
        Key key = generateKey();
        Cipher cipher = Cipher.getInstance(Encryption.ALGORITHM);
        cipher.init(Cipher.ENCRYPT_MODE, key);
        byte [] encryptedByteValue = cipher.doFinal(value.getBytes( charsetName: "utf-8"));
        String encryptedValue64 = Base64.encodeToString(encryptedByteValue, Base64.DEFAULT);
        return encryptedValue64;

    }

    public static String decrypt(String value) throws Exception
    {
        Key key = generateKey();
        Cipher cipher = Cipher.getInstance(Encryption.ALGORITHM);
        cipher.init(Cipher.DECRYPT_MODE, key);
        byte[] decryptedValue64 = Base64.decode(value, Base64.DEFAULT);
        byte [] decryptedByteValue = cipher.doFinal(decryptedValue64);
        String decryptedValue = new String(decryptedByteValue, charsetName: "utf-8");
        return decryptedValue;

    }
```

12.1.6 MapQuest

Map quest is an American company that provide many map services. I used MapQuest SDK in my application to allow the users to see the map [3]. Also, I added search bar so the user can search for any place and display the fastest route on the map. This is done by sending a Json object that includes the starting address, destination address and the API key to MapQuest, then MapQuest will response with a Json object that includes all the route details. The users have the ability to send the route details to the car by pressing the floating button. Figure 11 shows the result of the map activity.

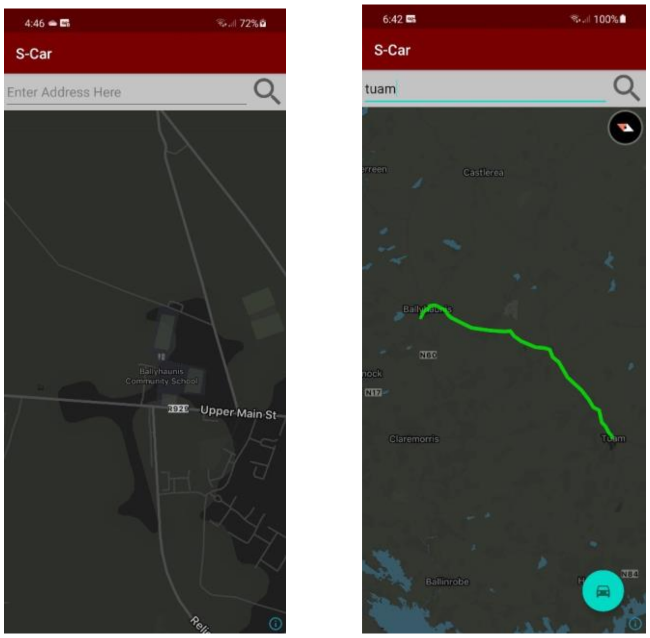


Figure 11 – MapQuest Activity Result

12.2 Autonomous Driving

Autonomous driving is a feature that I added to the car to assist the driver while driving on the motorways. It uses computer vision to detect the lanes on the road and to calculate the steering angle. This is done by converting the current frame to grey scale image as grey scale image has only one channel, dealing with one channel can speed up the process time. Then apply the gaussian filter to smooth the image and then get the edges by using the Canny function. Now the image can be cropped to ignore all the spaces that the lanes wont reach, then pass the image to Hough Line Transform algorithm to get the lines in the image. Then it finds the left line and the right lines by calculating the slope. If a line has negative slope, it will add it to the left lines array and if it has positive slope, it will add it to the right lines array. After separating the left lines by the right lines then it will average out both lines to get the best result.

My autonomous driving system will try its best to get the two lanes on the road, if one of the lanes were missing my system will predict the missing lane from its previous value and calculate the steering angle using the available lane. This was achieved by using OpenCV and by learning Python coding language. Figure 12 shows the result of the system when it is predicting a missing lane and the result when the two lanes are found.

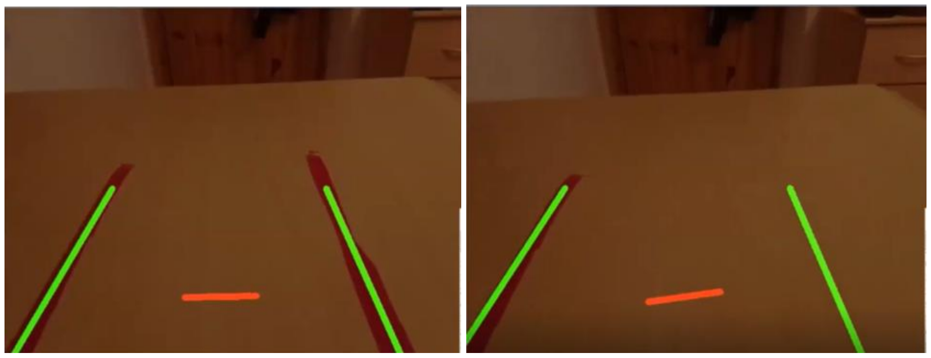


Figure 12 - Lane Detection Result

12.3 Parking Assist

Parking assist is a feature that will help the users to park their cars by using ultrasonic sensors to scan the road around the car.

12.4 Server

I used Apache Tomcat to build my server. The reasons I picked Tomcat were its lightweight, open source, high security and stable. My server will handle all the requests that coming from the app and the car. The content type that I am using to communicate with the app is octet-stream. This type is more efficient as it has less space and remove the need to encode and decode the data. The other content type that I am using is Json, I used Json to communicate with the car as its easer to extract it in C++.

13 Challenges

There were a few challenges I faced during my project, but I took them as a learning curve. I had to learn a new coding language for the computer vision. This was the main hurdle in my project because if I did not understand this it was unbale to create any code and the S-Car would not be functionally to its full capacity.

The other challenge I encountered was to write more advanced code to speed up the process time of the ESP32. The ESP32 was taking too long to get the ultrasonic sensors value, and this resulted in the car being crashed as it did not have enough time to stop. It also was slow in receives instructions from the app. I overcame this by adding tasks and dividing up the work to the two cores on the ESP32.

14 Recommendations

In my opinion the S-car should be equipped with more sensors as this will have a huge impact on how the car functions.

If the S-car had more sensors like the encoder sensor to calculate the distance travelled, then the parking assist would preform better. In my project the parking assist feature only using the ultrasonic sensors and this can affect the parking assist feature performance.

15 Conclusion

At the start of my final year project my object was to develop a self-driving car and contribute to the environment. I have achieved my goals, the S-Car is driving autonomously. The S-Car uses the camera along with Raspberry pi to complete this goal. Despite a few complications I faced through this project it was a very successful. The S-Car is eco-friendly as it runs on renewable energy.

The self-driving still has a long way to go. It is on the first stage of the S-Curve. It faces many challenges like the gaining the trust of the driver and the governments. Statistically it is safer on the road as it is autonomous driven. A self-driven car is beneficial to the disability's person, they can control the can via an app on their device. as anything it does have its disadvantages but the advantages out way the negatives and they are the future and people should start preparing for this.

16 References

[1] Bonnie Gringer, “History of self-driving cars” [Online”],2021. Available:

https://www.titlemax.com/resources/history-of-the-autonomous-car/#:~:text=In%20GM's%201939%20exhibit%2C%20Norman,made%20this%20concept%20a%20reality.

[2] R.Mahatha, “Encryption code”[Online”],2021. Available:

https://stackoverflow.com/questions/41223937/how-can-i-encrypte-my-password-android-studio

[3] MapQuest, “MapQuest SDK”[Online”],2021. Available: https://developer.mapquest.com/documentation/

17 Appendix

17.1 Software used

- Word
- Android Studio
- Arduino
- Power Point
- OpenCV
- PyCharm
- HeidiSQL
- IntelliJ IDEA
- Apache Tomcat
- GitHub
- screencast-o-matic