Arab American University

Faculty of Engineering

Computer Systems Engineering

SENIOR PROJECT (II)

SMART PARKING SYSTEM

2024

Students Statement

We, the undersigned students, certify and confirm that the work submitted in this project report is entirely our own and has not been copied from any other source. Any material that has been used from other sources has been properly cited and acknowledged in the report.

We are fully aware that any copying or improper citation of references/sources used in this report will be considered plagiarism, which is a clear violation of the Code of Ethics of the Arab American University.

May Lahlooh (201910890) *Eman Alyat (201911906)*

Rahmeh Daraghmeh (201912476)

Supervised by: Dr. Hazem Khanfar

Computer Systems Engineering Dept.
Submitted in partial fulfillment of the requirements of B.Sc. Degree in Computer Systems Engineering

February, 2024

Supervisor Certification

This to certify that the work presented in this senior year project manuscript was carried out under my supervision, which is entitled:

“**Smart Parking System**”

May Lahlooh (201910890) *Eman Alyat (201911906)*

Rahmeh Daraghmeh (201912476)

I hereby that the aforementioned students have successfully finished their senior year project and by submitting this report they have fulfilled in partial the requirements of B.Sc. Degree in Computer Systems Engineering.

I also, hereby that I have **read, reviewed and corrected the technical content** of this report and I believe that it is adequate in scope, quality and content and it is in alignment with the ABET requirements and the department guidelines.

Dr. Hazem Khanfar,

ACKNOWLEDGMENT

As we concluded our senior project, we expressed a lot of gratitude to all those who supported us. As a first step, we would like to thank Allah for allowing us to complete the project regarding smart parking systems. For all the suggestions and recommendations made by the supervisor of our project, Prof. Hazem Khanfar. Last but not least, I would like to thank our friends and loved ones for their unseen support and understanding throughout this journey.

ABSTRACT

In conjunction with escalating car numbers, several challenges become more noticeable. In the day-to-day lives of individuals, some are struggling to discover satisfying parking. Others are suffering to park their cars by themselves effectively. For this reason, a lot of problem-solving engineers were working to reduce the effects of these difficulties.

In this project, a supportive system implemented with the aim of enhancing the car parking process. The smart parking application will be a guide for any driver willing to find a suitable spot for his car. The application shows the number of empty spots in parking registered with the mobile application on a Google map of parking. The parking supported by an infrastructure to detect the number of empty spots. The application controls the movements of a prototype car in four directions to reach the desired place. The car model parks automatically by activating the auto-park option using the application. The components in the model make auto parking happen. Arduino, reed switch, ultrasonic sensors, and DC motors combine to create an auto-parking system. Following detection by the reed switch, the ultrasonic sensors on both sides take precise measurements to assess the optimal parking space. The car model will move forward in searching of an appropriate spot if no suitable spot is available.

TABLE OF CONTENTS

LIST OF FIGURES vii

LIST OF TABLES ix

LIST OF SYMBOLS AND ABBREVIATIONS x

CHAPTER 1: INTRODUCTION 1

1.1 PROBLEM STATEMENT AND PURPOSE 1

1.2 PROJECT AND DESIGN OBJECTIVES 1

1.3 INTENDED OUTCOMES AND DELIVERABLES 2

1 . 4 SUMMARY OF REPORT STRUCTURE 2

2 CHAPTER 2: BACKGROUND 3

2 . 1 OVERVIEW 3

2.2 RELATED WORK 3

2.2.1 SMART PARKING SYSTEM(SPS) ARCHITECTURE USING ULTRASONIC DETECTOR 3

2.2.2 AUTONOMOUS PARALLEL PARKING CAR MAKING USING ARDUINO 4

2.2.3 THE SMART PARKING MANAGEMENT SYSTEM 5

2.3 COMPARISON TABLE 6

3 CHAPTER 3: METHODS AND MATERIALS 7

3.1 SYSTEM DESIGN AND COMPONENTS 7

3.1.1 SYSTEM DESIGN 7

3.1.2 HARDWARE COMPONENTS 8

3.1.3 SOFTWARE COMPONENTS 14

3.1.4 HARDWARE DESIGN 15

3.1.5 SOFTWARE DESIGN 16

3.2 DESIGNSPECIFICATIONS AND CONSTRAINTS 2 0

3.2.1 SPECIFICATIONS 20

3.2.2 CONSTRAINTS 21

3.3 DESIGN ALTERNATIVE: 2 1

3.4 SYSTEM ANALYSIS AND OPTIMIZATION 2 2

3.4.1 FLOW CHART 22

3.4.2 USE CASE 23

3.4.3 USE CASE TABLE 24

3.4.4 SEQUENCE DIAGRAM 25

3.5 SIMULATION AND/OR EXPERIMENTAL TEST 28

3.5.1 TEST LOGIN VALIDATION AND VERIFICATION 28

3.5.2 TEST SIGNUP VALIDATION AND VERIFICATION 28

3.5.3 SYSTEM CONNECTIONS USING THE FRITZING SIMULATOR 29

4 CHAPTER 4: RESULTS AND DISCUSSIONS 33

4.1 RESULTS 33

4.1.1 HARDWARE RESULT 33

4.1.2 SOFTWARE RESULT 34

4.2 DISCUSSION 36

4.2.1 PARKING MODEL 38

5 CHAPTER 5: PROJECT MANAGEMENT 39

5.1 TASKS, SCHEDULE AND MILESTONES 39

5.2 RESOURCES AND COST MANAGEMENT 40

5.3 LESSONS LEARNED 40

6 CHAPTER 6: IMPACT OF THE ENGINEERING SOLUTION 41

6.1 ECONOMICAL, SOCIETAL AND GLOBAL 41

6.2 ENVIRONMENTAL AND ETHICAL 41

7 Chapter 7: CONCLUSION AND RECOMMENDATION 42

7.1 SUMMARY OF ACHIEVEMENTS OF THE PROJECT OBJECTIVES 42

7.2 NEW SKILLS AND EXPERIENCES LEARNT 42

7.3 RECOMMENDATION FOR FUTURE WORK 42

PREFERECES 43

APPENDICES 45

Appendix A: 45

Appendix B: 50

LIST OF FIGURES

Figure **Page**

FIGURE 2. 1: LCD DISPLAY FOR THE FIRST RELATED WORK 4
FIGURE 2. 2: PROTOTYPE CAR FOR THE SECOND RELATED WORK 4
FIGURE 2. 3: IMPLEMENTATION FOR THE THIRD RELATED WORK 5
FIGURE 3. 1: BLOCK DIAGRAM OF THE PROJECT 7
FIGURE 3. 2: ARDUINO UNO (ATMEGA328) 9
FIGURE 3. 3: ULTRASONIC SENSOR 9
FIGURE 3. 4: INFRARED SENSOR 10
FIGURE 3. 5: MOTOR DRIVE MODULE BOARD (L298N DUAL H-BRIDGE) 11
FIGURE 3. 6: DC MOTOR 11
FIGURE 3. 7: REED SWITCH 12
FIGURE 3. 8: ESP8266-01 12
FIGURE 3. 9: FEMALE TO MALE JUMPER WIRES 13
FIGURE 3. 10: FEMALE-TO-FEMALE JUMPER WIRES 13
FIGURE 3. 11: MALE-TO-MALE JUMPER WIRES 13
FIGURE 3. 12: PCB BOARD 14
FIGURE 3. 13: FIRESTORE DATABASE (PARKING COLLECTION) 14
FIGURE 3. 14: FIRESTORE DATABASE (USERS COLLECTION) 15
FIGURE 3. 15: REAL-TIME DATABASE 15
FIGURE 3. 16: HARDWARE DESIGN 16
FIGURE 3. 17: FIRST PAGES OF THE APP 16
FIGURE 3. 18: LOGIN AND SIGNUP PAGES 17
FIGURE 3. 19: HOME PAGE AND PARKING MAP 18
FIGURE 3. 20: AUTO PARKING OPTION PAGE 18
FIGURE 3. 21: GOOGLE MAP OF THE APP 19
FIGURE 3. 22: CONTROL CAR PAGE 19
FIGURE 3. 23: EDIT DRIVER PROFILE PAGES 20
FIGURE 3. 24: FLOW CHART FOR THE SYSTEM 22
FIGURE 3. 25: FLOW CHART FOR THE SMART PARKING 23
FIGURE 3. 26: USE CASE FOR THE MOBILE APPLICATION 23
FIGURE 3. 27: USE CASE FOR AUTO PARKING 24
FIGURE 3. 28: USE CASE FOR THE OBJECT RECOGNITION 24
FIGURE 3. 29: AUTO-PARKING SEQUENCE DIAGRAM 25
FIGURE 3. 30: OBJECT RECOGNITION SEQUENCE DIAGRAM 26
FIGURE 3. 31: LOGIN SEQUENCE DIAGRAM 26
FIGURE 3. 32: SIGNUP SEQUENCE DIAGRAM 27
FIGURE 3. 33: MOVE THE CAR IN FOUR DIRECTIONS SEQUENCE DIAGRAM 27
FIGURE 3. 34: LOGIN VERIFICATION 28
FIGURE 3. 35: SIGNUP VERIFICATION 29
FIGURE 3. 36: CAR'S COMPONENTS SIMULATION 30
FIGURE 3. 37: SMART PARKING INFRASTRUCTURE SIMULATION 30
FIGURE 4. 1: APPLICATION LOGO 34

Figure 4. 2: Real-Time Database for Parking Spots 34
Figure 4. 3: Google Map of the Application Reflects Real-Time Data 35
Figure 4. 4: Parking Map of the Application Reflects Real-Time Data 36
Figure 4. 5: Case1, Prototype Car Parks on First Left Spot 37
Figure 4. 6: Case2, Prototype Car Parks on the Second Right Spot 37
Figure 4. 7: Case3, Prototype Car Parks on the Last Right Spot 38
Figure 4. 8: Infrared Detection for the Prototype Car 38
Figure 5. 1: Gantt Chart 1 39
Figure 5. 2: Gantt Chart 2 39
Figure 5. 3: Gantt Chart 3 39

LIST OF TABLES

Table **Page**

TABLE 1: LIST OF SYMBOLS AND ABBREVIATIONS x
TABLE 2: COMPARISON BETWEEN RELATED WORK AND THE PROJECT 6
TABLE 3: COMPARISON BETWEEN ARDUINO TYPES 8
TABLE 4: USE CASES DESCRIPTION 24
TABLE 5: ULTRASONIC SENSOR PINS AND THEIR FUNCTION 31
TABLE 6: REED SWITCH PINS AND THEIR FUNCTION 31
TABLE 7: INFRARED SENSOR PINS AND THEIR FUNCTION 31
TABLE 8: ESP8266-01 PINS AND THEIR FUNCTION 31
TABLE 9: MOTOR DRIVER PINS AND THEIR FUNCTION 32
TABLE 10: RESOURCES MANAGEMENT 40

LIST OF SYMBOLS AND ABBREVIATIONS

This is where you should put all the symbols used and abbreviations **(must be sorted)**. It is stored in a table format same as the table of contents. Use inserts a row to add more entries….etc

Table 1: List of Symbols and Abbreviations

LIST OF SYMBOLS	
μm/Mm/Cm/M	Micrometer/Millimeter/Centimeter/Meter
KHZ/MHz/ GHZ/ THZ	Kilohertz/Megahertz/Gigahertz/Terahertz
V	Voltage
A	Ampere
KB	Kilobytes
LIST OF ABBREVIATIONS	
IOT	Internet of Things
GUI	Graphical User Interface
SPS	Smart Parking System
LED	Light Emitting Diode
LCD	Liquid Crystal Display
DC Motor	Direct Current Motor
PWD	Pulse Width Modulation
USB	Universal Serial Bus
ICSP	In-Circuit Serial Programming
IO	Input/output
USART	universal synchronous/asynchronous receiver/transmitter
VIN	vehicle identification number
IR	Infrared Sensor
TX/RX	Transmitter and Receiver pins
IDE	Integrated development environment (IDE) is a software application.

CHAPTER 1: INTRODUCTION

By the end of 2022, there is 290,000 vehicles in the West Bank [1]. This large number resulted in various issues, including challenges such as traffic congestion for drivers on the roads and difficulties in finding parking spaces. Smart parking systems have become an effective solution to the parking-related problems faced by drivers in West Bank, as they can be helpful in organizing parking and fixing challenges such as difficulty in finding parking spaces and parking individuals’ cars with minimize collisions.

1.1 PROBLEM STATEMENT AND PURPOSE

Nowadays, with the increasing population, their own cars are increasing. This leads to many economic and environmental problems that affect people's lives. Traffic congestion and collisions, air pollution, a lack of space, and a lot of wasted fuel and cost in the process of searching. In addition, a large number of individuals suffer from not having a suitable park since there are no appropriate spots for a large number of cars, which works as an obstacle. Since traditional parking cannot be convenient in populated areas where parking spaces are limited, Hence, a way of organizing and managing the parking is needed to make this process more efficient, convenient, and faster. The system will help to reduce these problems by developing an appropriate application to provide a more suitable and organized way of indicating the empty spots on each parking space in the city. The application will give the driver the option to make a self-park without intervention for the supported cars with advanced technologies.

1.2 PROJECT AND DESIGN OBJECTIVES

In our project, our main objectives are to implement a smart parking system that allows drivers to discover and select the nearest available parking spots from parking around the city without wasting time or fuel. The application shows the updated numbers of empty spots in the parking lot. The application will also contain a graphical user interface (GUI) that contains buttons to control the movements of the prototype car in different directions: left, right, forward, and backward. The driver can either align his car traditionally by himself or activate the icon of the self-park option. If the driver turns on the icon, then the attached sensors in the car must take accurate measurements to preserve safe distances between the vehicles.

1.3 INTENDED OUTCOMES AND DELIVERABLES

As the end of the project, we implemented a smart system that manages the process of searching for empty spots. The prototype car, supported by advanced technologies, will be able to park in a specific place through the application's other features, which permit the driver to guide his car. To enhance this process, the application will display the number of empty spots on its location in the Google Maps.

1 . 4 SUMMARY OF REPORT STRUCTURE

In Chapter 2, background is introduced. In Chapter 3, system design is analyzed. In Chapter 4, results and discussions are presented. In Chapter 5, resource management is specified. In Chapter 6, the impact of the engineering solution is discussed. In Chapter 7, conclusions and recommendations are made.

2 CHAPTER 2: BACKGROUND

2 . 1 OVERVIEW

The implementation of parking divided into fixed dimensions of spots supported by infrared sensors to detect available spots in real time. The data collected by the sensors will indicate the number of empty spaces in the parking lot to all users who download the mobile application to help them park more efficiently. The prototype car can move in different directions according to the user using the mobile application. The prototype car is supported by a reed switch and ultrasonic sensors to ensure fixed distances between spots and an automatic park without collisions.

2.2 RELATED WORK

The issues facing an increasing population inspire many people to minimize the troubles associated with them. As a result, many systems are developed using different strategies. In this section, we introduce some of these systems. Each one fixed the problem from a different point of view.

2.2.1 SMART PARKING SYSTEM (SPS) ARCHITECTURE USING ULTRASONIC DETECTOR

This system is developed to help drivers find parking spots in multilevel parking in a shorter time. It is a sensor-based system that depends primarily on ultrasonic sensors. Equipping the parking lot with one ultrasonic sensor for each individual parking spot to calculate the number of empty parking spots. A LED display board provided at the entrance gate of the parking lot with real-time information about the spots. The systems' spots are categorized into 4 types; display these categories by different colors of LEDs. The first type of spot is the vacant spot, illustrated by a green LED. The second displayed in red. Also, handicapped and booked spots are shown in blue and yellow, respectively. The display board has additional features to guide the drivers in the direction of vacant spots, as shown in figure 2.1.
To count the number of spots, the system designed by stabilizing one ultrasonic sensor above each individual spot to detect if there is a car or if it is available by calculating the time needed between the sent pulse and the reflection of it. Another sensor along the separated lines will detect if there is an improper parking. Improper parking occurs when a car parks on the line; this will occupy two spots where there is just one car in the parking place [2].

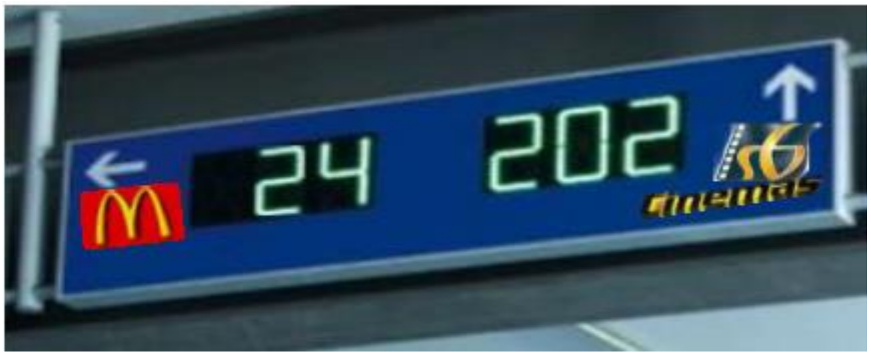


Figure 2. 1: LCD Display for the First Related Work

2.2.2 AUTONOMOUS PARALLEL PARKING CAR MAKING USING ARDUINO

The project designed using Arduino Mega to support self-park characteristics according to specific assumptions. It consists of four DC motors, each one installed on each wheel, and 4 ultrasonic sensors distributed at the front, rear, and 2 on the left side, as shown in figure 2.2. The prototype car parks in two ways, depending on the spot's dimensions. The first case is vertical parking, which happens when the length of the car is greater than the parking spots. In this case, the left sensors measure the length of the spot, and the car rotates 90 degrees to the left, then moves forward until it reaches 10cm from the parking wall. On the other hand, when the length of the spot is less than the length of the car, the system activates the second case, which is parallel parking. In parallel parking, the car moves until the sensor on the other edge sees the wall again. In this situation, the car comes back a little, then moves backward by rotating right 45 degrees. The sensors at the two edges measure continuously and stop when their values are equal. Then, the car moves forward until the front sensor’s value becomes less than 10 cm [3].

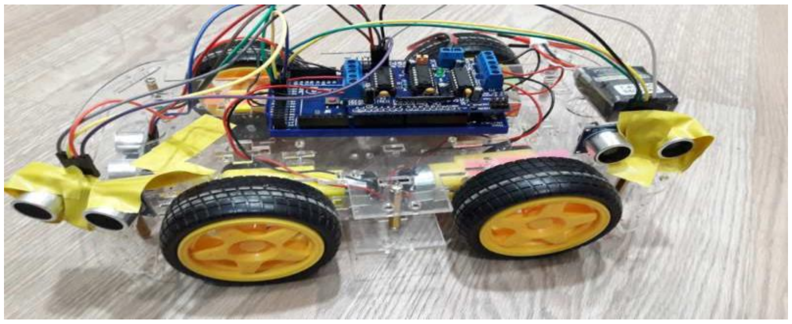


Figure 2. 2: Prototype Car for the Second Related Work

2.2.3 THE SMART PARKING MANAGEMENT SYSTEM

A smart parking management system provides a management strategy to fix the challenges of parking in large cities. It is an internet of thing-based systems (IOT) that cooperates between advanced technologies. The sensors installed on each spot, as shown in figure 2.3, detect if it is vacant or not. This information is sent to the cloud by Arduino UNO to process them and evaluate the output as a result of the mobile application. The outcome displays a map of all parking spaces in the area to allow the user to select the best one for him. In addition, the application provides more options, like booking and payment services. The driver can see real-time information on parking in his region and book a spot within 24 hours. The system is supported by cameras at the entry to the garage to calculate the expected cost by estimating the time of entry and exit from the parking [4].

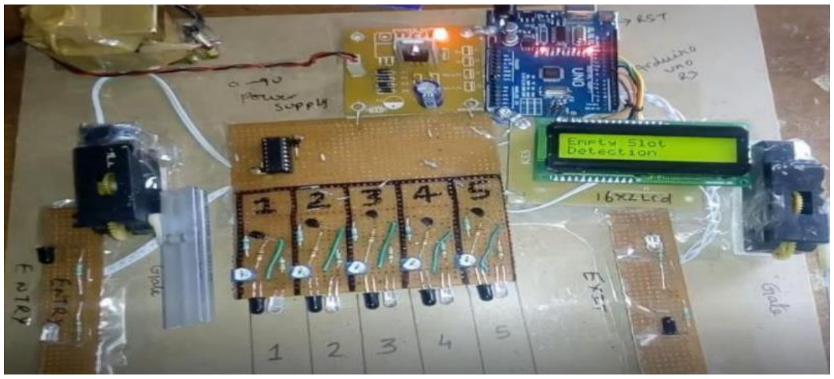


Figure 2. 3: Implementation for the Third Related Work

2.3 COMPARISON TABLE

Table 2 shows how our project is similarities and differences between our project and the pervious project mentioned in the related work section.

Table 2: Comparison between Related Work and the Project

	Similarities	**Differences**
Smart Parking System (SPS) Architecture Using Ultrasonic Detector	Both have an installed sensor that detect if the status of the spot is vacant or not.	The outcome of this project is shown on LCD display at the entrance of the parking, but in our project the outcome will be expressed in the mobile application.
Autonomous Parallel Parking Car Making Using Arduino	Both have the same hardware approach for auto parking.	There is no mobile application provides options for controlling the movement and guide drivers for their optimal parking.
THE SMART PARKING MANAGEMENT SYSTEM	Both are similar in software and hardware techniques. The two projects have sensors on the spots connected with a mobile application to manage a process of parking.	In this project, there is no auto parking option in their mobile application. While in our application, we will not provide a booking option.

Our project cooperates the advantages between the mentioned related works. It has features enable the car to park by itself also it has a technique to detect if the spots are vacant or not.

3 CHAPTER 3: METHODS AND MATERIALS

3.1 SYSTEM DESIGN AND COMPONENTS

3.1.1 SYSTEM DESIGN

Figure 3.1 shows the block diagram of the project. A smart parking system that integrates hardware and software to reflect the process of real auto parking cars. Providing an effective way for selecting optimized parking by developing a mobile application that shows available spots using infrared sensors to detect car existence from the spots. The information stored on the Arduino UNO microcontroller is passed to the firebase through an esp8266-01. For a safe car park, the application gives the option to park the car automatically based on the reed switch and ultrasonic sensors installed on the car. The Arduino UNO waits until the reed switch is detected. Then, upon the ultrasonic readings, there are controlling instructions passed to the motor driver, which controls the movements of DC motors.

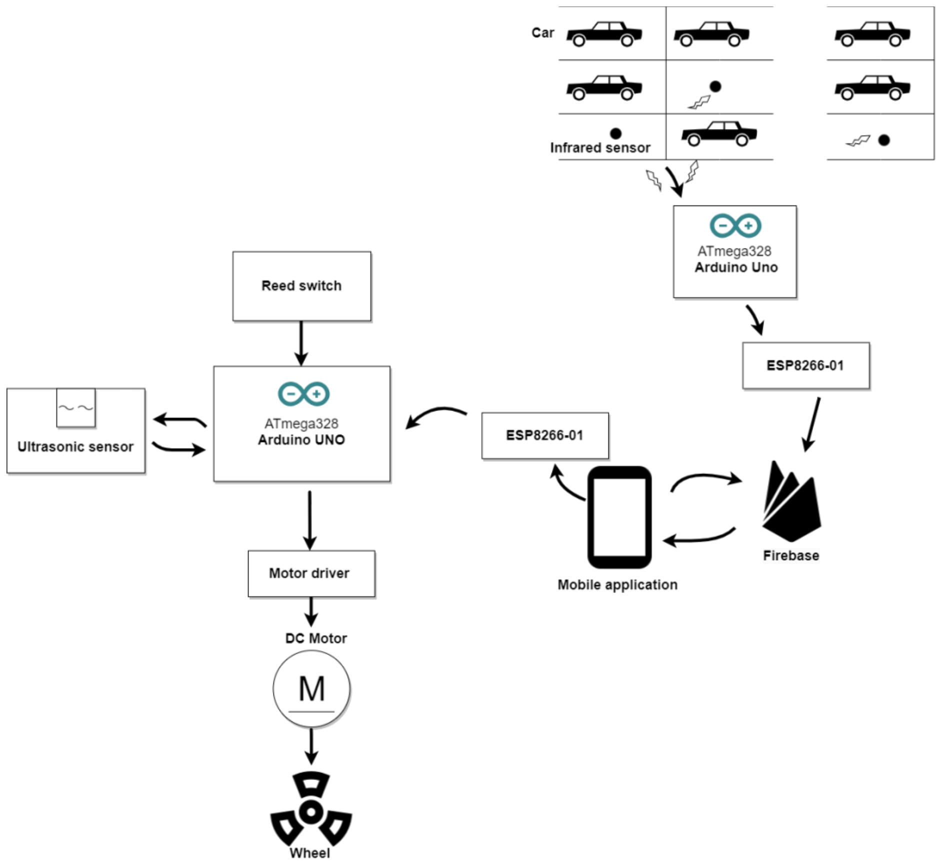


Figure 3. 1: Block Diagram of the Project

3.1.2 HARDWARE COMPONENTS

In this section, we will illustrates the main hardware component of our project.

Hardware components:

- Arduino UNO
- Ultrasonic sensor
- Infrared sensor
- Esp8266-01
- DC motor
- Motor driver
- Reed switch
- PCB board
- Wires
- Power supply

ARDUINO

Table 3: Comparison between Arduino Types

Arduino Name	Microcontroller	Crystal Oscillator	Operating Voltage(V)	Digital IO/PWM	Analog pins	Price (ILS)
Arduino Uno	ATmega328	16MHz	7-12	14/6	6	35
Arduino Mega	ATmega2560	16MHz	5	54/15	16	100
Arduino Nano	ATmega328	16MHz	5	14/6	8	30
Arduino Micro	ATmega32u4	16Mhz	7-20	20/7	12	28

Depending on the defined requirements, the Arduino UNO is the appropriate type according to the number of input/output pins and size for the car model and smart parking infrastructure [5].

- **ARDUINO UNO:**

A microcontroller chip used in controlling various hardware devices contains an ATmega328P microcontroller. It has 14 digital input/output pins and analog input pins. The chip consists of components facility programming the chip.

Components of Arduino UNO:

It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator (crystal frequency), a USB connection, a power jack, an ICSP header and a reset button.

In the project, we used this microcontroller to achieve the functionalities of making an auto-park car and a smart parking model [6].

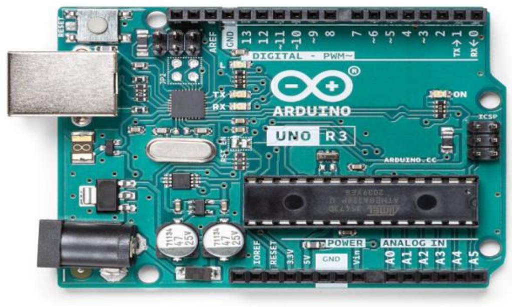


Figure 3. 2: Arduino UNO (ATmega328)

ULTRASONIC SENSOR:

It is a type of detecting sensor that requires a frequency above 20 kHz and measures the distance between objects or obstacles. It could measure from 2 cm to 400 cm, as shown in figure 3.3. It works by sending ultrasonic sound waves and receiving their reflection. By calculating the time needed to reflect and the sound speed, the distance cloud can be measured [7].

The installed ultrasonic in the prototype car was used to calculate the spot dimensions to verify if the distance was suitable to park or not.

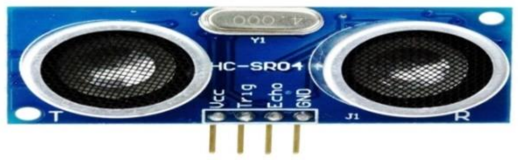


Figure 3. 3: Ultrasonic Sensor

INFRARD SENSOR:

It is one of the most widely used sensors for detecting objects. IR sensor uses a wavelength range from 2 to 14 µm requires between 300 GHZ to 430 THZ frequency.

IR consists of two major components, which are the IR emitter and the IR receiver. The IR emitter sends infrared light, and reflections of it are taken by the IR receiver. Whenever the object is closer, the reflecting signal becomes strong [8].

This sensor was used in the parking model to verify the object in each parking spot, and this data is stored in the real-time firebase as zero and one. Based on this data, the number of available spots was calculated according to the number of ones to display the result in the app.

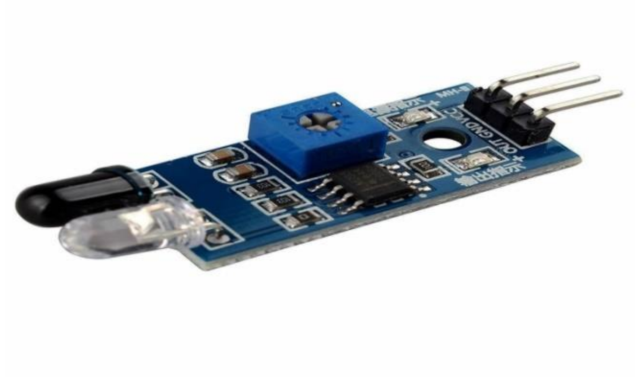


Figure 3. 4: Infrared Sensor

MOTOR DRIVER:

It is an electronic device, which used for controlling the speed and direction for DC motors. As shown in figure 3.5 It acts like an amplifier to convert the low power signal from the microcontroller to high power signal which can drive the motor. The input voltage needed 5V to drive motors with voltage 5-35V with maximum 2A current for each bridge [9].

The L298N motor driver is capable of overseeing the motion of two DC motors. In the prototype car, this device was employed to regulate four DC motors, enabling the vehicle to move in four directions. This was achieved by linking the right-side motors and the left-side motors together, ensuring synchronized movement on each side.

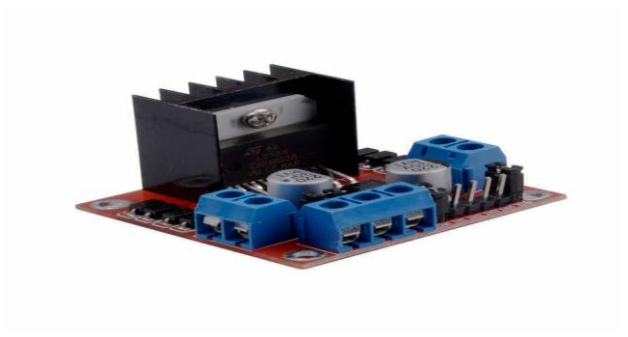


Figure 3. 5: Motor Drive Module Board (L298N Dual H-Bridge)

DC MOTOR:

It’s a device control, which converts the input electrical current into mechanical energy to make the car model movements. A four DC motors required to move the prototype car in four direction. Each motor move clock-wise or counter clock-wise based on the sequence of high and low value [10].

Figure 3. 6: DC Motor

REED SWITCH:

It is a type of sensor that depends on a magnetic field to sense any metal near it. It has two ferromagnetic elements to make a closed circuit. By default, the circuit between them opens, but once any magnetic material is near, the two ends attract each other and close the circuit [11]. In the project, the reed switch was used to detect the center of the road between each of the two spots by installing the sensor at the bottom of the prototype car and a magnet in the center between each of the two spots. Once the magnet is detected by the sensor, it sends the value to the Arduino to decide if the spot is suitable to park the prototype car or continue moving forward until it detects the next magnet.

Figure 3. 7: Reed Switch

ESP8266-01:

It is a type of ESP8266 module that has many functionalities. It has more limited resources than advanced ESP8266 chips. About 1MB for the flash memory and eight pins for different purposes like enable, power, ground, and serial communication pins. It can act like a microcontroller or Wi-Fi module [12].

In the project, we used two esp8266-01 chips. On the prototype car, esp-01 receives the commands from the application and sends it to the Arduino UNO to control the car according to these commands. On the parking model, esp-01 receives the data about infrared status from the Arduino UNO serial and stores it in the real-time firebase.

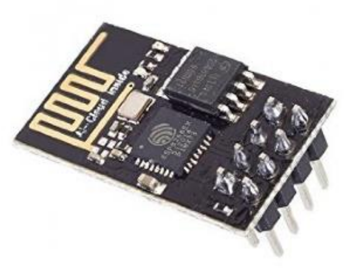


Figure 3. 8: ESP8266-01

WIRES

In the figures bellow, shows the type of the wires we used in our project.

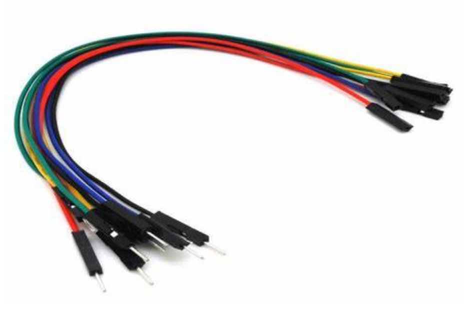


Figure 3. 9: Female to Male Jumper Wires

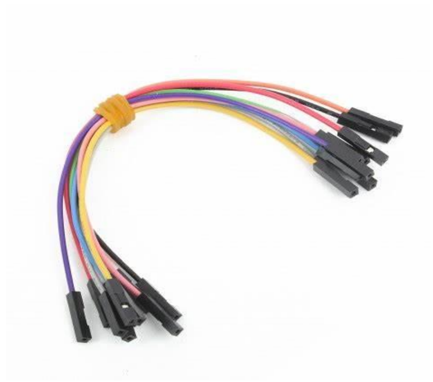


Figure 3. 10: Female-to-Female Jumper Wires

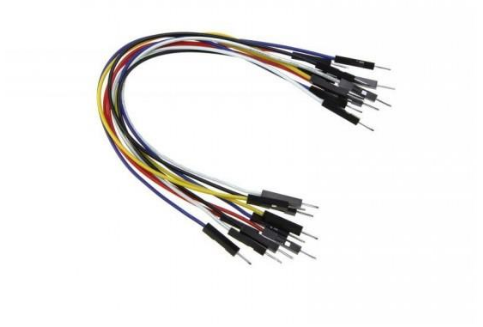


Figure 3. 11: Male-to-Male Jumper Wires

PCB board

A board used to mechanically support and electrically connect electronic components using conductive pathways.

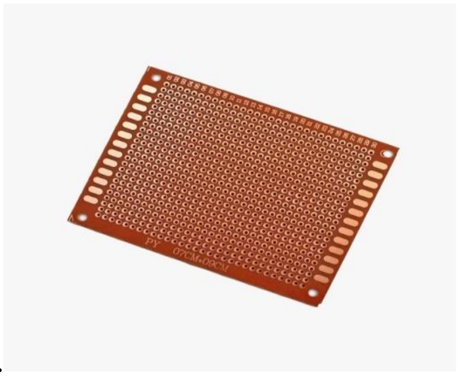


Figure 3. 12: PCB Board

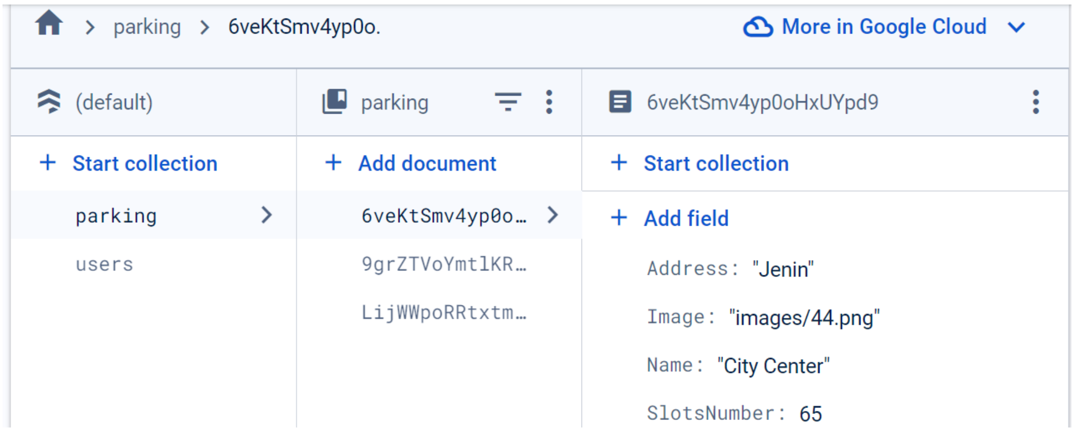
3.1.3 SOFTWARE COMPONENTS:

In this section, we will explain the software tools that we used in our project.

- **Firebase Database**

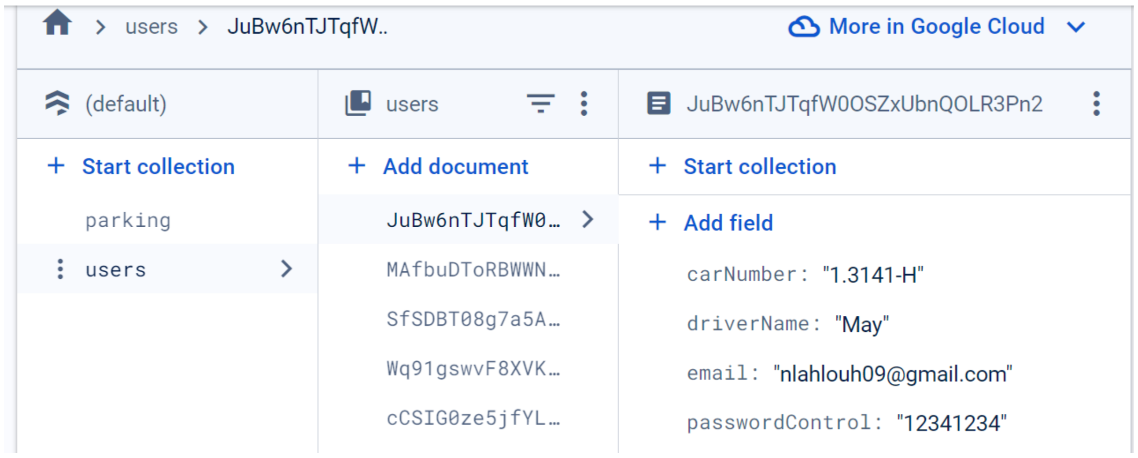
Generally, the system starts by sending the infrared sensor value for each spot to the Arduino UNO, and the Arduino will hand it over to the ESP8266-01, which will store it in the real-time database as a JSON object. The application uses these values to show the number of empty spots on the Google map. Moreover, the application offers an illustration page to show whether the spot is vacant or not.

Besides, the driver information entered during registration and the parking information will be stored on the firestore database as collections.



Address: "Jenin"
Image: "images/44.png"
Name: "City Center"
SlotsNumber: 65

Figure 3. 13: Firestore Database (Parking Collection)


users > JuBw6nTJTqfW..

More in Google Cloud

(default)	users	JuBw6nTJTqfW0OSZxUbnQOLR3Pn2
+ Start collection	+ Add document	+ Start collection
parking	JuBw6nTJTqfW0…	+ Add field
users	MAfbuDToRBWWN…	carNumber: "1.3141-H"
	SfSDBT08g7a5A…	driverName: "May"
	Wq91gswvF8XVK…	email: "nlahlouh09@gmail.com"
	cCSIG0ze5jfYL…	passwordControl: "12341234"

Figure 3. 14: Firestore Database (Users Collection)

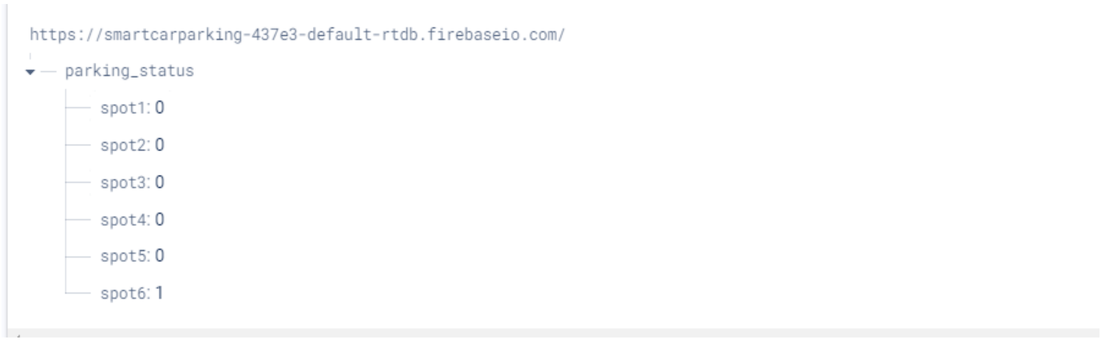

```
https://smartcarparking-437e3-default-rtdb.firebaseio.com/
parking_status
    spot1: 0
    spot2: 0
    spot3: 0
    spot4: 0
    spot5: 0
    spot6: 1
```

Figure 3. 15: Real-time Database

- **Mobile Application**

We built it using the Dart language and Flutter framework using the Android Studio then we connected it with the firebase database.

3.1.4 HARDWARE DESIGN

The hardware design for both the car prototype and parking infrastructure is in place. This setup emulates smart parking system. Within the project, the model encompasses six spots as shown in figure 3.16, with each spot equipped with an infrared sensor to update the spot's status in a real-time database. Furthermore, we have developed a prototype car with automated parking capabilities to emulate the functionality of a smart vehicle.

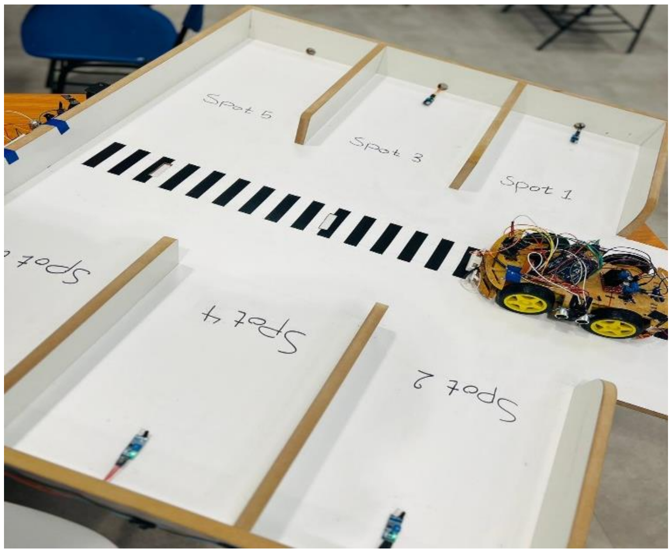
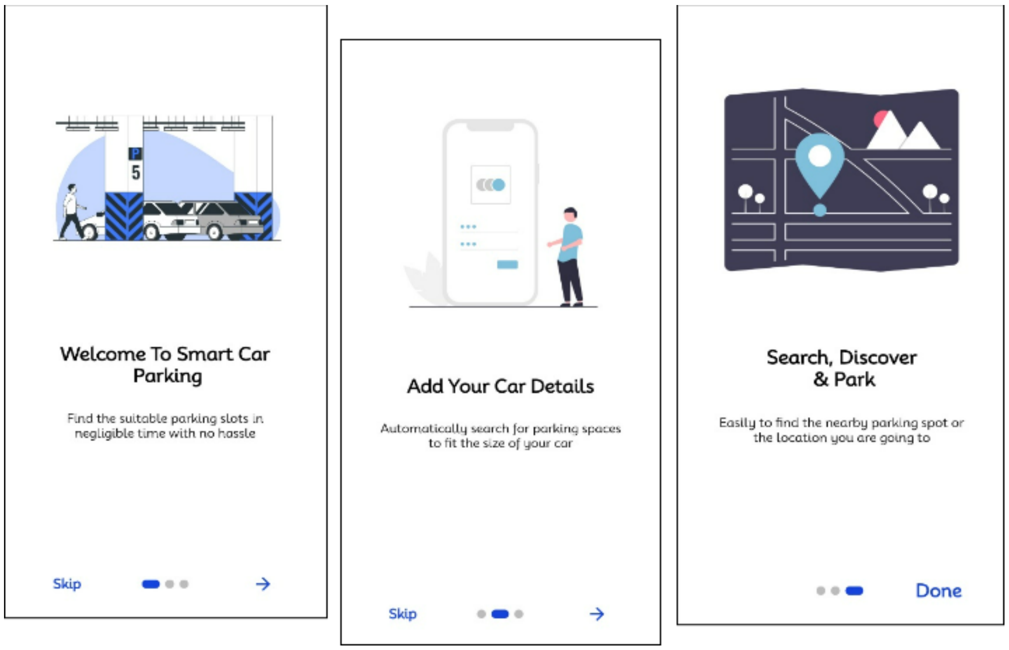


Figure 3. 16: Hardware Design

3.1.5 SOFTWARE DESIGN

In this section, we present the application design we have developed for the project, aimed at controlling the prototype car and assisting the driver in identifying the most suitable parking surrounding him.



Welcome To Smart Car Parking

Find the suitable parking slots in negligible time with no hassle

Add Your Car Details

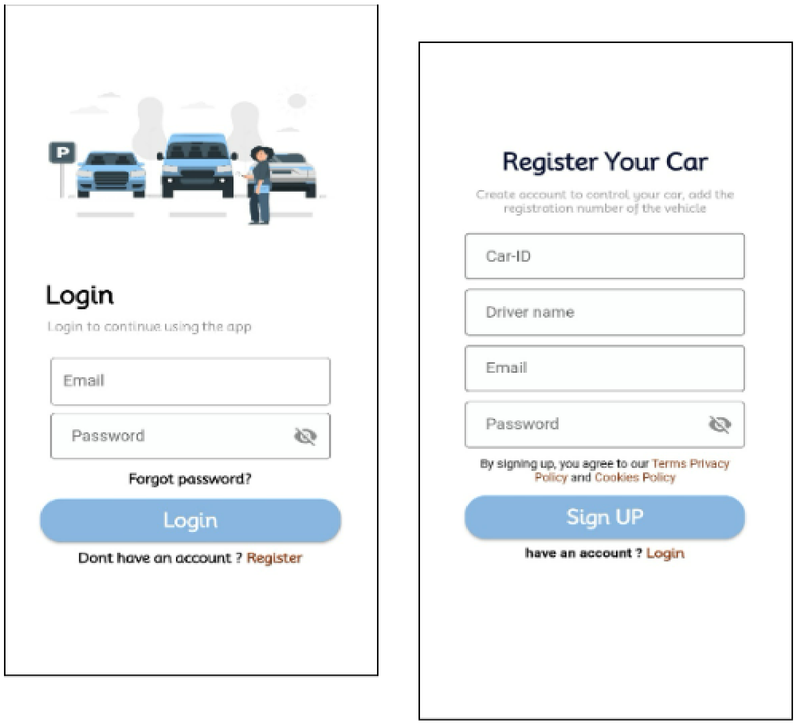
Automatically search for parking spaces to fit the size of your car

Search, Discover & Park

Easily to find the nearby parking spot or the location you are going to

Figure 3. 17: First Pages of the App

Following the introduction pages, users have the option to login to their accounts and commence using the app. In case a user does not have an existing account, he can create a new one by providing his information. As shown in figure 3.18.



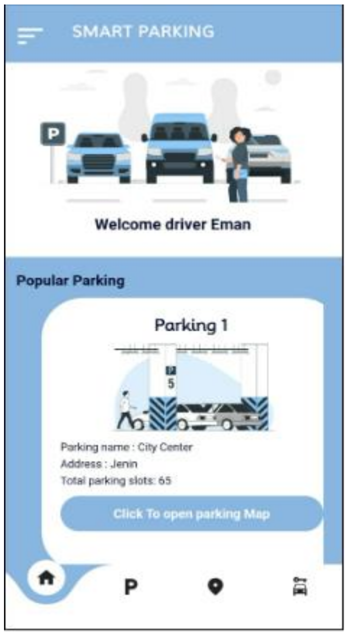
Login to continue using the app

Create account to control your car, add the registration number of the vehicle

By signing up, you agree to our Terms Privacy Policy and Cookies Policy

Figure 3. 18: Login and Signup Pages

The app's homepage exhibits parking information stored in the database, presenting it to the drivers. By tapping the "Parking Map" button, drivers can view a reflection for the parking spots. In our scenario, there are six spots according to the parking design, and their status is indicated by the infrared sensors' readings, whether they are available or vacant. As shown in figure 3.19.



Parking name : City Center
Address : Jenin
Total parking slots: 65

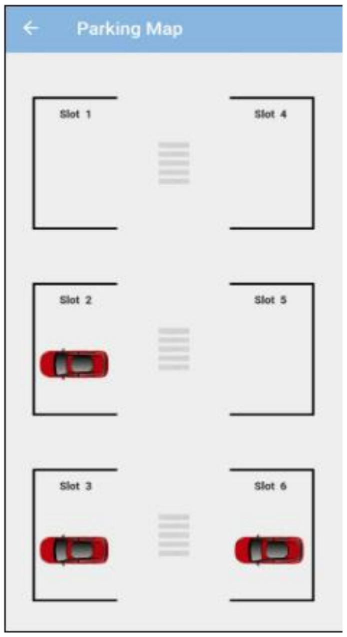


Figure 3. 19: Home Page and Parking Map

As figure 3.20, this page shows the most crucial feature, the "Auto Park" option. Upon pressing the button, the auto-parking process initiates and begins its operations. To confirm the initiation of the auto-park process, a notification will be send.

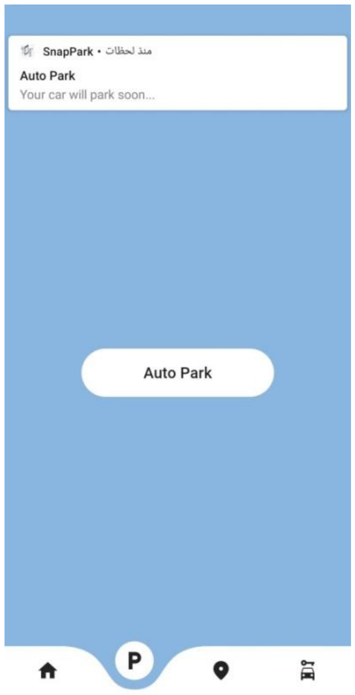

SnapPark • منذ لحظات
Auto Park
Your car will park soon...

Figure 3. 20: Auto Parking Option Page

The Google Map feature illustrates the count of available parking spots for our model at the specific parking location. This count computed based on real-time data obtained from the infrared sensors installed on each spot.

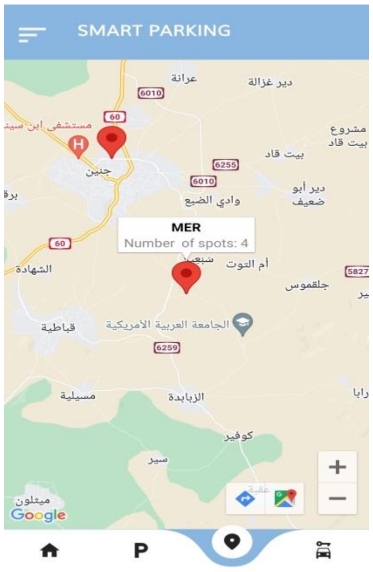


Figure 3. 21: Google Map of the App

The control car page of the application is equipped with five buttons, enabling users to send commands from the application to the car's microcontroller, facilitating movement in four directions. Additionally, the central button serves to halt the car's movements.

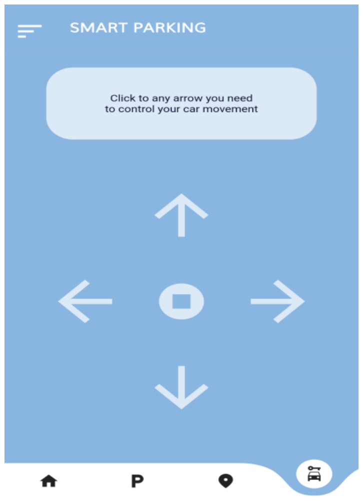
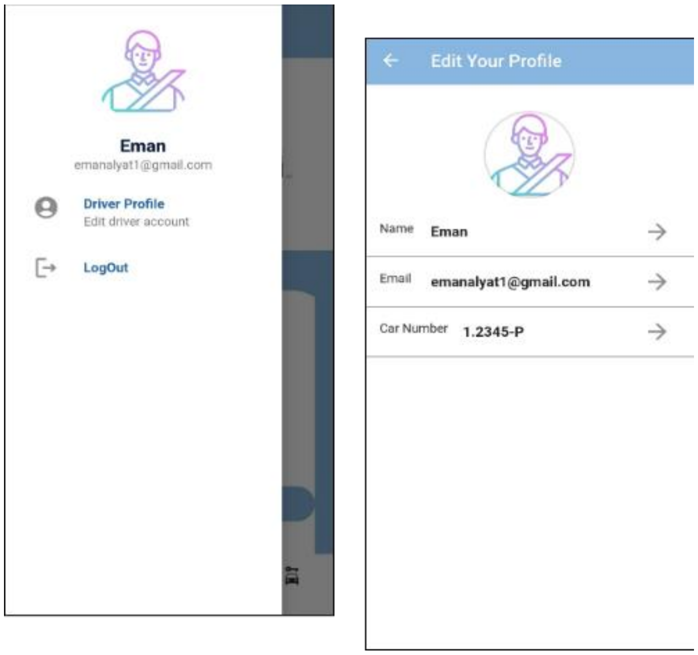


Figure 3. 22: Control Car Page

The driver has the capability to edit their personal information in the event of any changes after the initial sign-up process.


Eman
emanalyat1@gmail.com
Driver Profile
Edit driver account
LogOut

Edit Your Profile
Name Eman
Email emanalyat1@gmail.com
Car Number 1.2345-P

Figure 3. 23: Edit Driver Profile Pages

3.2 DESIGNSPECIFICATIONS AND CONSTRAINTS

3.2.1 SPECIFICATIONS

The functional specifications of the project defines the main features we implemented and the integration between prototype car, parking model and the application to provide a solution for the challenges associated with the parking.

- Provide cost effective, timesaving and well-organized way for selecting optimized parking by developing mobile application shows available parking in surrounding city.
- Allow the movement of the car in four directions using mobile control buttons.
- Allow the driver to activate auto-park option.
- Upon activation of the auto-park function, the prototype car will commence scanning for a suitable parking spot.
- Driver access to the application is restricted until their email verification is completed.
- The application displays information about the parking stored in the firestore database.

- The Google map shows the parking information, including its name, associated with a real-time update of empty spots.
- Allow the driver to edit his preferences. Updating his name, email and car number, and these updates change automatically.

3.2.2 CONSTRAINTS

- In the infrared sensor, real-time changes take 20 seconds to update the data in the real-time database.
- The prototype is not working unless the ESP is connected to the internet.
- The prototype car can't moving unless using the same network with the ESP.
- Continuous charging is required for the power supply.

3.3 DESIGN ALTERNATIVE:

In the alternatives section, we provide an explanation for the modifications made to certain components within our project, as well as introduce the new component that was incorporated.

- We considered utilizing a Bluetooth module for transmitting commands from the application to govern the car's movements. Simultaneously, we aimed to transfer data from the infrared sensors integrated into the parking design, processed by the Arduino microcontroller, to a real-time database. However, the use of Bluetooth resulted in confusion and limited functionality to a range of less than 10 meters. Consequently, we opted for a Wi-Fi strategy, employing the ESP8266-01, to avoid these issues.
- Initially, we intended to use the Arduino Mega as the microcontroller to control the car model. However, upon acquiring the car model and engaging with it, we discovered that its dimensions were incompatible with the Arduino Mega. Additionally, the Arduino Mega had numerous unused pins and required more power. Consequently, we opted to substitute it with the Arduino UNO, which has dimensions suitable for the prototype car and offers equivalent functionality to the Arduino Mega.
- We were compelled to incorporate a detection sensor for identifying the center of each row within the parking area. Consequently, we employed a reed switch that generates a magnetic field upon reaching the installed magnet at each center on the parking ground. This initiates a search for a suitable parking spot on both the left and right sides, facilitating a precise automated parking process.

3.4 SYSTEM ANALYSIS AND OPTIMIZATION

3.4.1 FLOW CHART:

Figure 3.24 shows the flow chart of the mobile application. After the driver opens the application, he can discover and move to the nearest available parking from the map in the application. Then, when the driver reaches the parking lot, he can activate the auto-parking option. The prototype car moved forward until it found the first reed switch. The ultrasonic sensor examines if the spot dimensions are suitable to park; the process is successful. Otherwise, the car will continue moving forward along all reed switches.

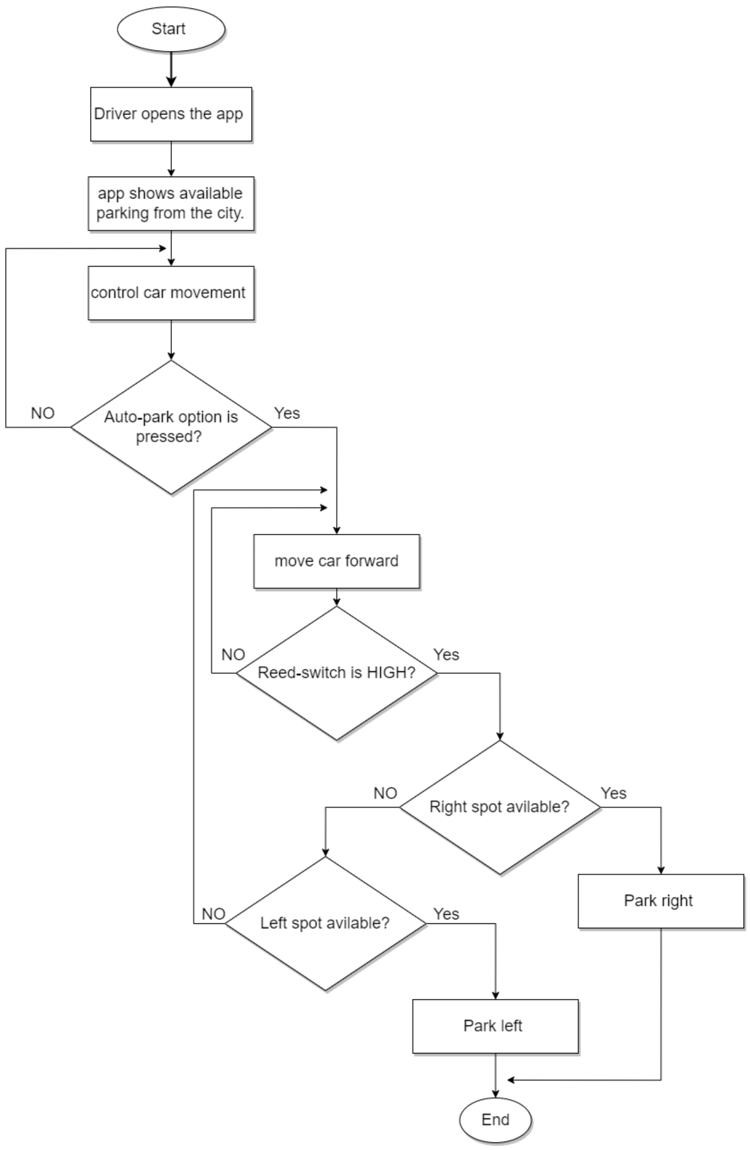


Figure 3. 24: Flow Chart for the System

Figure 3.25 shows the flow chart of smart parking. The infrared sensors detect the car's existence of each spot to send the data onto the real-time database in the firebase through the ESP8266-01, which is connected with an Arduino UNO to count the number of empty spots on each parking lot and update this information to make real-time updates.

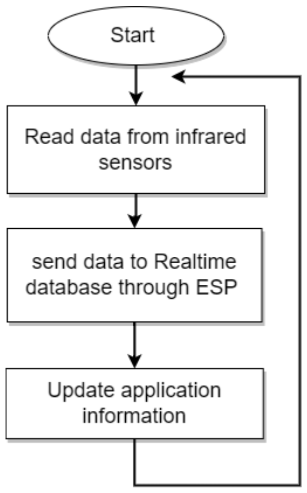

Start
Read data from infrared sensors
send data to Realtime database through ESP
Update application information

Figure 3. 25: Flow Chart for the Smart Parking

3.4.2 USE CASE:

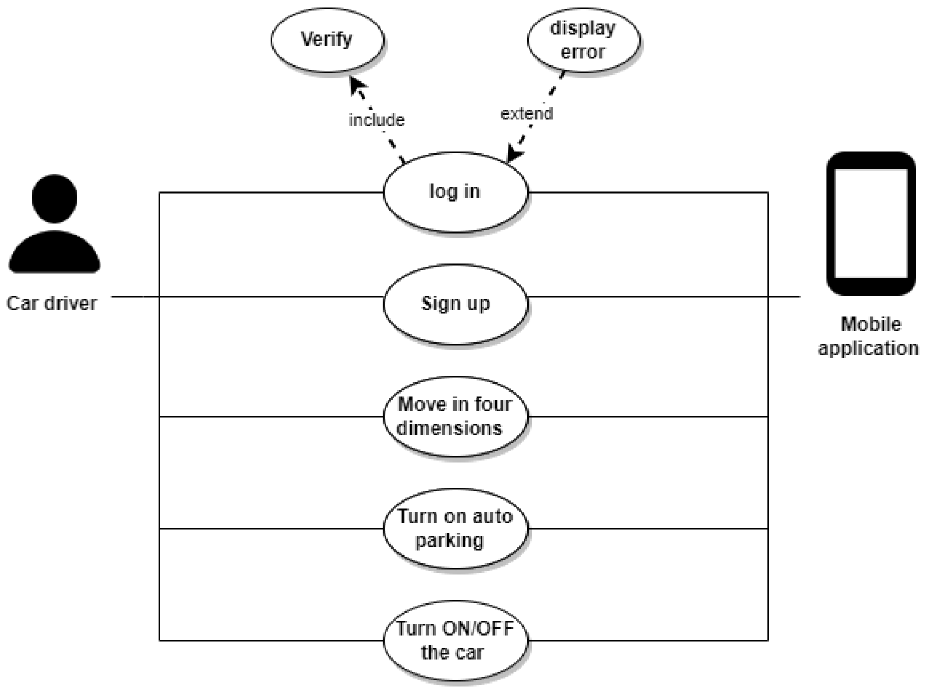


Figure 3. 26: Use Case for the Mobile Application

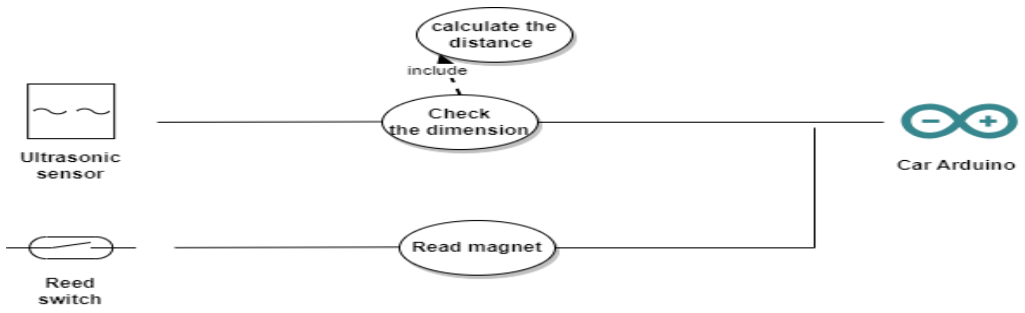


Figure 3. 27: Use Case for Auto Parking

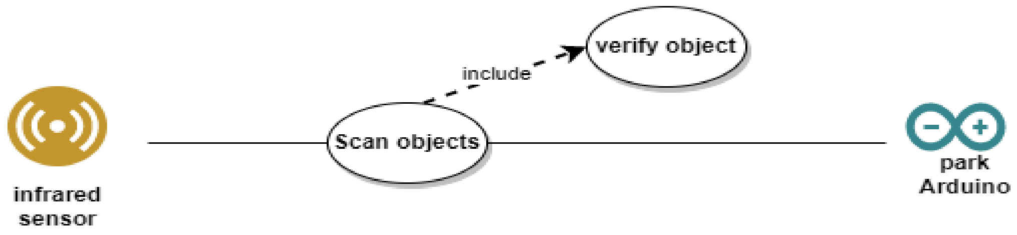


Figure 3. 28: Use Case for the Object Recognition

3.4.3 USE CASE TABLE:

Table 4, shows brief description for each use case in our project.

Table 4: Use Cases Description

Use case	Description
Sign-Up	The system allows the driver to sign up using the vehicle email, name, identification number (VIN) and password.
Log In	The database server makes authentication for the registered driver to log in his account.
Turn ON/OFF car	The system allows the driver to on and off the car prototype model using icons on the app.
Move	Using the application icons to control the movement of the car model in four directions.
Auto parking option	The mobile application supported by auto park option, which enables the drivers to park the car model automatically.
Read magnet	The reed switch has HIGH value until it reaches the center where there is installed magnet in the ground. Thus, the value changes to LOW and start looking up for an empty spot.

Check the dimension	The ultrasonic sensor takes accurate measurements for the spots' dimensions. If the right spot available, then it will park right, else it will park left. Otherwise, the can will move to the next center where the reed in installed.
Scan objects	The infrared sensor sends and receives infrared radiation to sense the car existence or not.

3.4.4 SEQUENCE DIAGRAM:

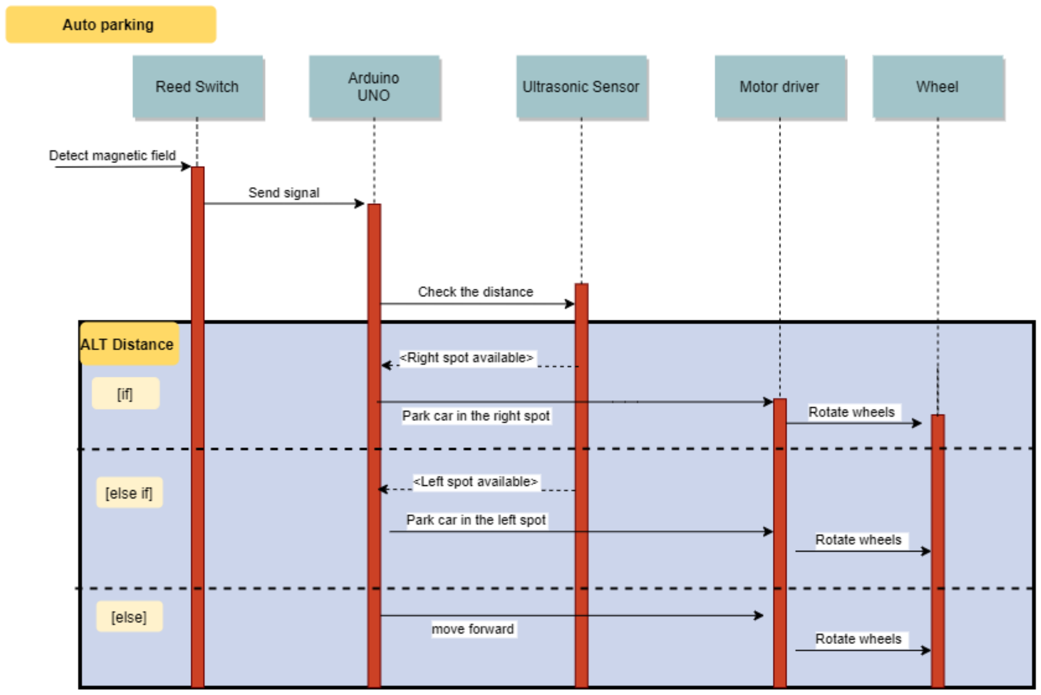


Figure 3. 29: Auto-Parking Sequence Diagram

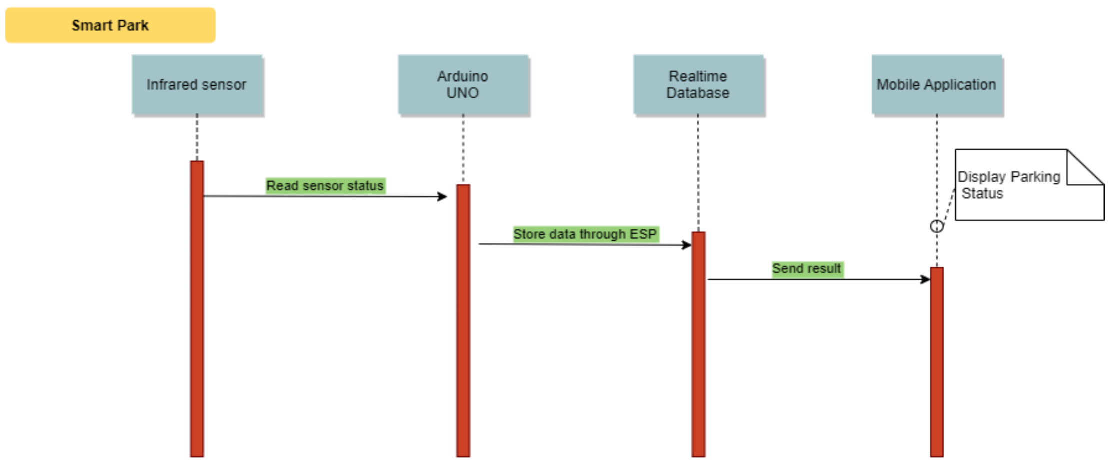


Figure 3. 30: Object Recognition Sequence Diagram

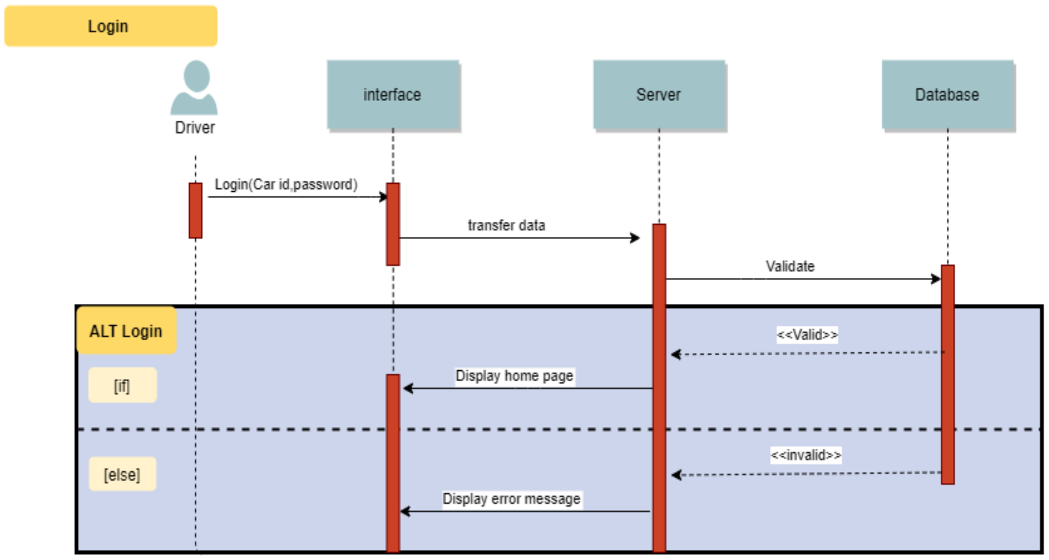


Figure 3. 31: Login Sequence Diagram

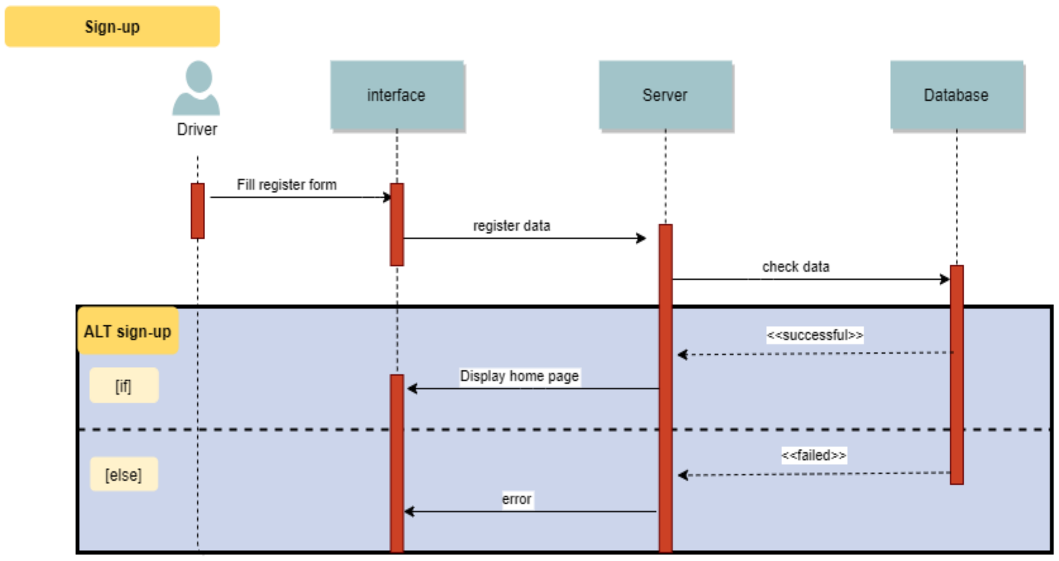


Figure 3. 32: Signup Sequence Diagram

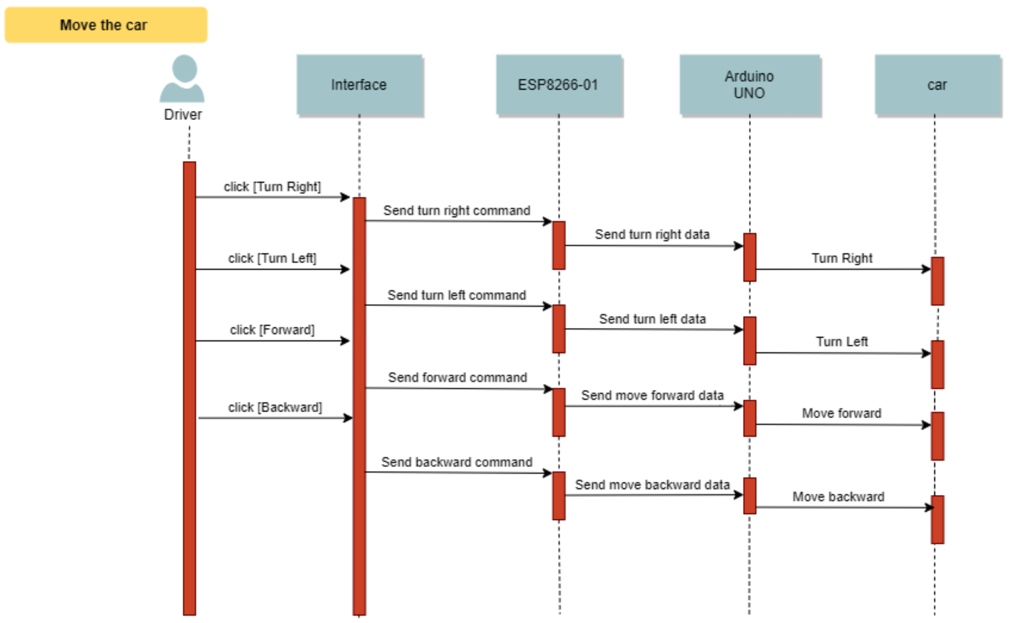
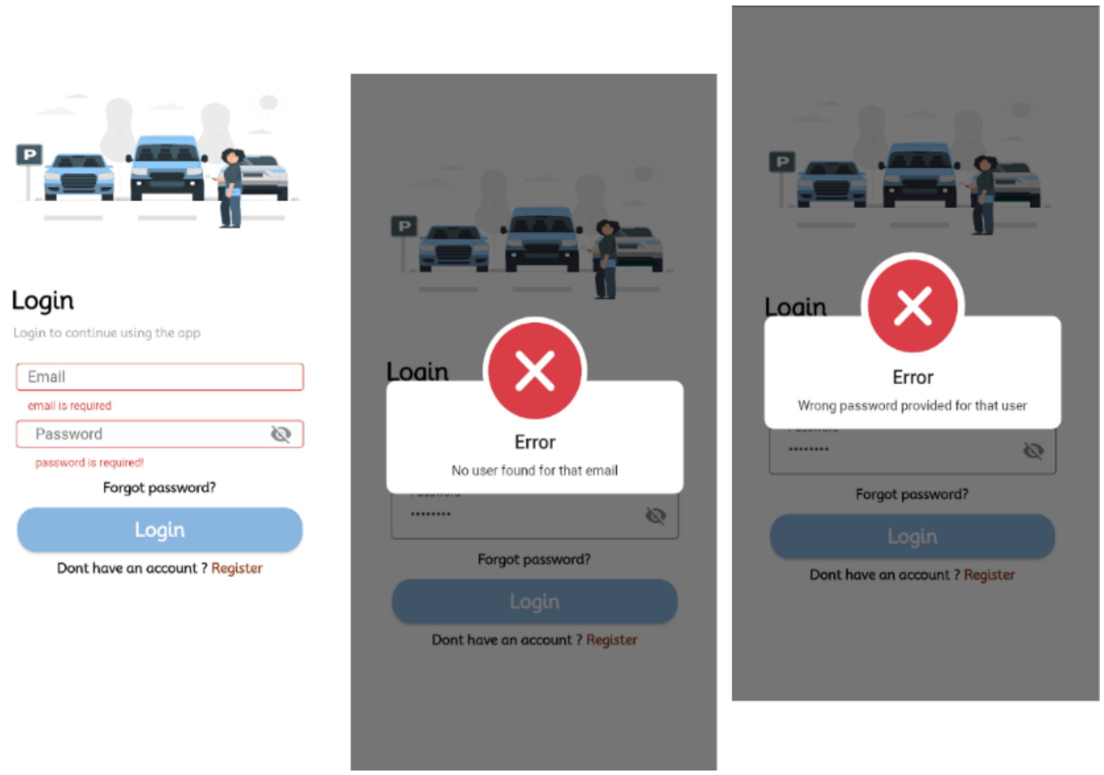


Figure 3. 33: Move the Car in Four Directions Sequence Diagram

3.5 SIMULATION AND/OR EXPERIMENTAL TEST

3.5.1 TEST LOGIN VALIDATION AND VERIFICATION

A user's login is verified if they are not found or if their password is incorrect. Therefore, both cases display an alarm.


Login
Login to continue using the app
Email
Password
Forgot password?
Login
Dont have an account ? Register
Login
Forgot password?
Login
Dont have an account ? Register
Login
Forgot password?
Login
Dont have an account ? Register

email is required

password is required!

Error

No user found for that email

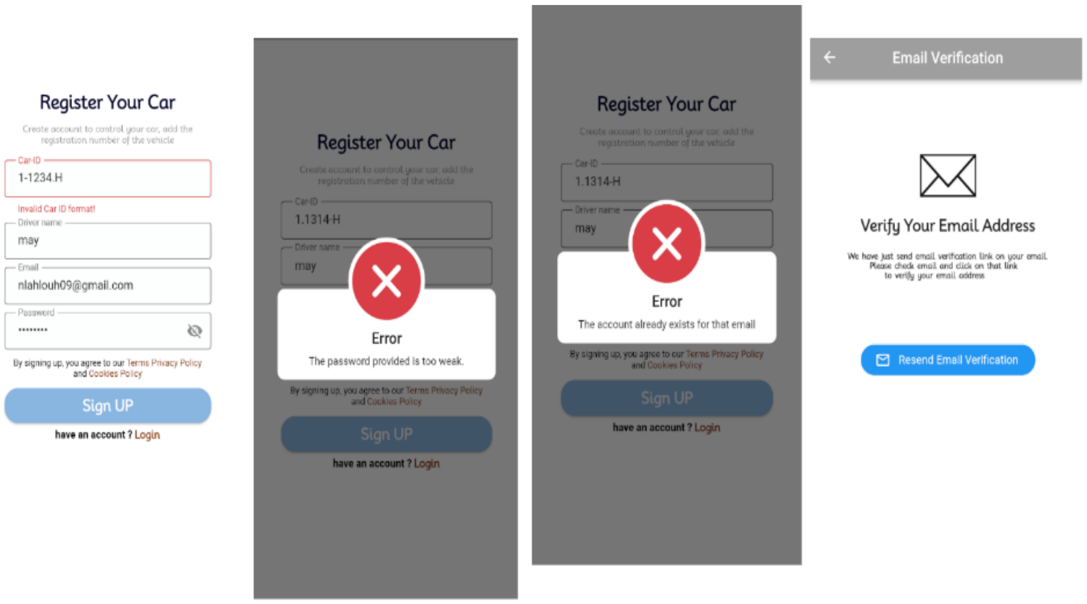
Error

Wrong password provided for that user

Figure 3. 34: Login Verification

3.5.2 TEST SIGNUP VALIDATION AND VERIFICATION

A weak password or an existing email in Firebase is considered as part of the registration verification process. In both cases, an alarm is displayed. Additionally, users cannot use the application until they verify their email by using the URL they receive by email.



Create account to control your car, add the registration number of the vehicle

Car-ID: 1-1234.H

Invalid Car ID format!

Driver name: may

Email: nlahlouh09@gmail.com

Password: ••••••••

By signing up, you agree to our Terms Privacy Policy and Cookies Policy

Car-ID: 1.1314-H

Driver name: may

Error

The password provided is too weak.

Error

The account already exists for that email

We have just send email verification link on your email. Please check email and click on that link to verify your email address

Figure 3. 35: Signup Verification

3.5.3 SYSTEM CONNECTIONS USING THE FRITZING SIMULATOR

Figure 3.26 illustrates the hardware design, detailing the connections between components responsible for controlling the car's behavior. The process unfolds as follows: Upon receiving commands via ESP8266-01, the Arduino UNO processes the commands through its serial interface. Subsequently, the motor driver takes charge of directing the movements of the DC motors based on the received commands (forward, backward, left, and right), allowing the car to move in four directions. In the case of an "auto-park" command, the motor driver controls the DC motors forward until the reed switch detects the magnet. Following this, ultrasonic sensors measure the height to determine the suitability or vacancy of the spot. If both spots left and right are unsuitable, the motor driver guides the DC motors to move forward until the next magnet is detected, repeating the process to achieve automatic parking.

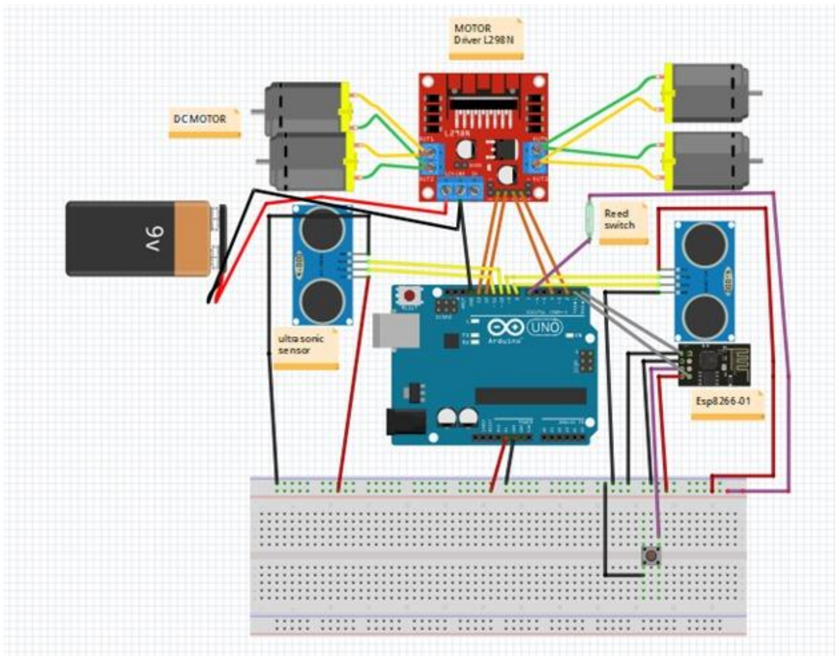


Figure 3. 36: Car's Components Simulation

Figure 3.27 illustrates the process of transmitting the readings from each spot's infrared detector. The Arduino UNO and ESP8266-01 chip utilize the same serial interface to transfer the infrared data for each spot, subsequently storing this information in the real-time database.

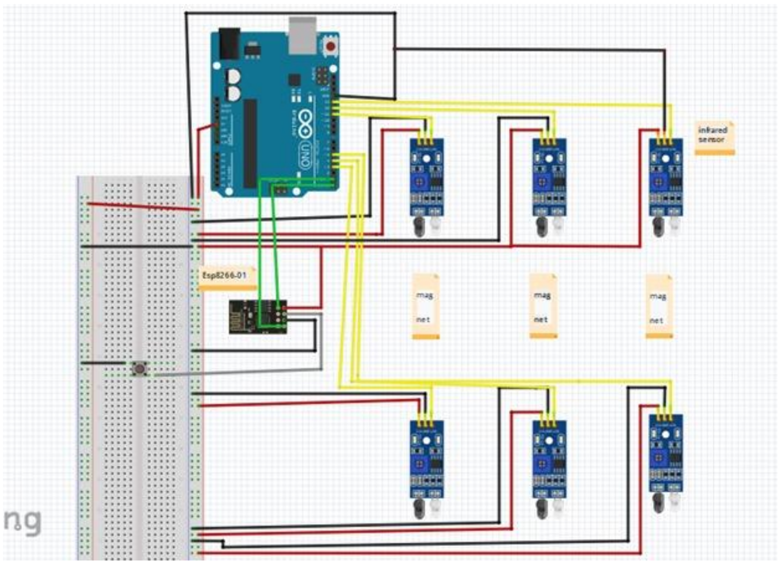


Figure 3. 37: Smart Parking Infrastructure Simulation

Table 5: Ultrasonic Sensor Pins and their Function

Pin	Function
VCC	This pin used for input supply. At this pin, we provide an input voltage to Ultrasonic Sensor.
GND	This pin used for ground.
Echo	The ultrasound receiver.
Trig	The ultrasound transmitter.

Table 6: Reed Switch Pins and their Function

Pin	Function
VCC	This pin used for input voltage.
Input pin	Active high pin, declaring the magnet existence.

Table 7: Infrared Sensor Pins and their Function

Pin	Function
VCC	This pin used for input voltage.
GND	This pin used for ground**.**
Output pin	Active high output, declaring the object existence.

Table 8: ESP8266-01 Pins and their Function

Pin	Function
3V	This pin used for input voltage.
RST	This pin used while programming to reset the PC.
EN	This pin to active the chip.
TX	Connect to RX with Arduino.
RX	Connect to TX with Arduino.
IO0	General Purpose Input/output pin 0. used to declare programming mode while uploading code on it.

IO2	General Purpose Input/output pin 2.
GND	This pin used for ground.

Table 9: Motor Driver Pins and their Function

Pin	**Function**
MOTORA	Motor1, controlling the right side of DC motors.
MOTORB	Motor2, controlling the left side of DC motors.
12V	Input power.
GND	Ground.
5V	Supply 5V output.
ENA	The enable pin for motor1.
ENB	The enable pin for motor2.
IN1	The pin for motor1 control.
IN2	The pin for motor1 control.
IN3	The pin for motor2 control.
IN4	The pin for motor2 control.

4 CHAPTER 4: RESULTS AND DISCUSSIONS

4.1 RESULTS

4.1.1 HARDWARE RESULT

- **PROTOTYPE CAR:**

Before the mobile application can communicate with the ESP module. The ESP8266-01 establishes a connection to the Wi-Fi network and is assigned a unique IP address. Upon establishing the connection, the application gains the capability to communicate with the prototype parking system. HTTP requests used to send commands to the prototype car. These commands, responsible for controlling the car's movements, transmitted through the ESP serial communication to the Arduino. Subsequently, the microcontroller interprets these commands (forward, backward, left, and right) and control of the DC motors using the motor driver. In addition to other functionalities, a crucial feature is the "Auto Park" command sent from the application to the microcontroller. Upon receiving this command, the DC motors are engaged to move the prototype car forward until the reed switch detected. If the magnet identified by the reed switch, the ultrasonic system activates to assess spot availability. Subsequently, the car automatically parks either left or right. In cases where the parking spots are unoccupied, the prototype car continues its forward movement, reaching the switch again and repeating the auto-parking process.

- **PARKING MODEL:**

The parking system comprises six fixed-dimensional spots, each equipped with an infrared sensor to detect the presence of an object. The data captured by the infrared sensors transmitted through the Arduino-ESP serial communication and promptly stored in a real-time database. The real-time data fetched in flutter to be used on application pages. In the Google map page, the number of empty spots associated within the parking name on its location. In the parking page, it illustrates the availability by associating a car for each vacant spot.

4.1.2 SOFTWARE RESULT

- **APPLICATION LOGO:**

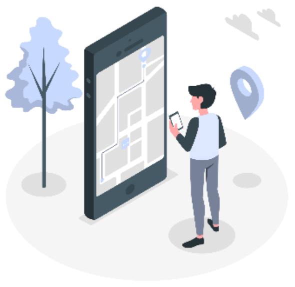
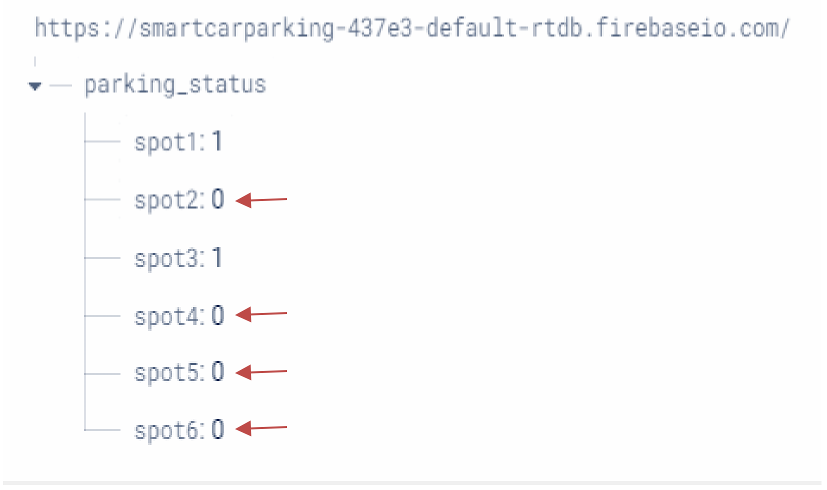


Figure 4. 1: Application Logo

- **REAL-TIME PARKING SYSTEM:**

In the parking model, the status of each parking spot is updated in real-time based on the input from infrared sensors. These updates are stored in a real-time database, with values set to one if the spot is vacant and zero if the spot is occupied as shown in figure 4.2. These values are utilized within the application through two methods to streamline the parking process, both on Google Maps and on a dedicated parking map. This approach enables drivers to easily find available parking spots by consulting the real-time data provided by the application, enhancing the efficiency and convenience of parking in the designated area.



```
https://smartcarparking-437e3-default-rtdb.firebaseio.com/
▾ — parking_status
      spot1: 1
      spot2: 0
      spot3: 1
      spot4: 0
      spot5: 0
      spot6: 0
```

Figure 4. 2: Real-Time Database for Parking Spots

Google Map : The Google Map page has been designed to show the number of available parking spots at each parking location by calculating the number of zeros in the parking status data. This functionality allows users to view, in real-time, how many spaces are empty in a specific parking area directly on its location on the map, making it easier for drivers to identify and navigate to the nearest available parking spot.

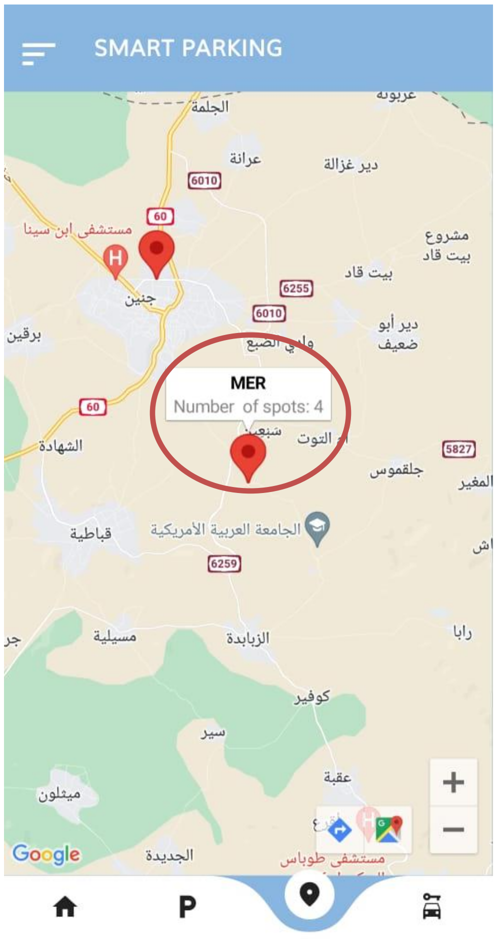


Figure 4. 3: Google Map of the Application Reflects Real-Time Data

Parking Map: The parking map acts as a real-time visual representation of the occupancy status of each parking spot. If the infrared sensor detects an object in a spot, that spot is marked on the map with a car icon, indicating it's occupied. Conversely, if no object is detected, the spot is shown as empty, and no icon is displayed. This user-friendly page allows users to quickly assess the availability of parking spot.
As shown in Figure 4.4, spots 1 and 3 are marked with a car icon, indicating that they are vacant, while the other spots are shown as empty according to the data in the real-time database.

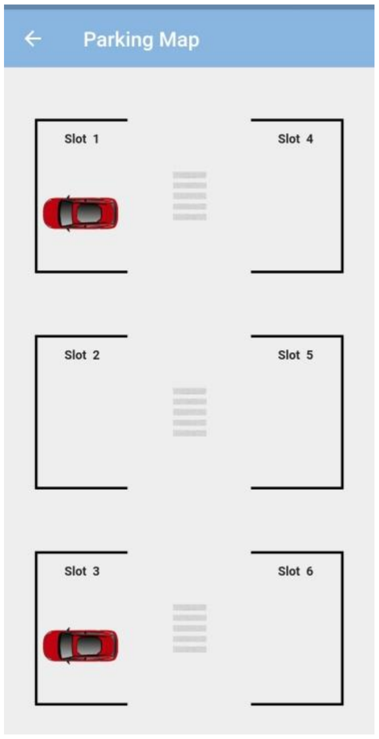


Figure 4. 4: Parking Map of the Application Reflects Real-Time Data

4.2 DISCUSSION

In this section, we looked at examples of the prototype car's behavior when we activated the auto-park option in the application.

- **Case1: Prototype Car Parks on the First Left Spot**

In this scenario, the prototype car moves forward until it identifies the initial magnet. Upon detection, the ultrasonic sensor gauges the availability of parking spots by measuring distances greater than 40 cm within the dimensions of our parking area.). Measurements calculated by both sensors positioned on the left and right sides of the prototype car. If the measured distance on the right side exceeds 40 cm, the car will park on the right. However, in this scenario, a specific object is present on the right spot, indicating that the distance is less than 40 cm. Therefore, the car will adjust its parking location and choose the left spot, which is unoccupied and free from any obstacles as shown in figure 4.5.

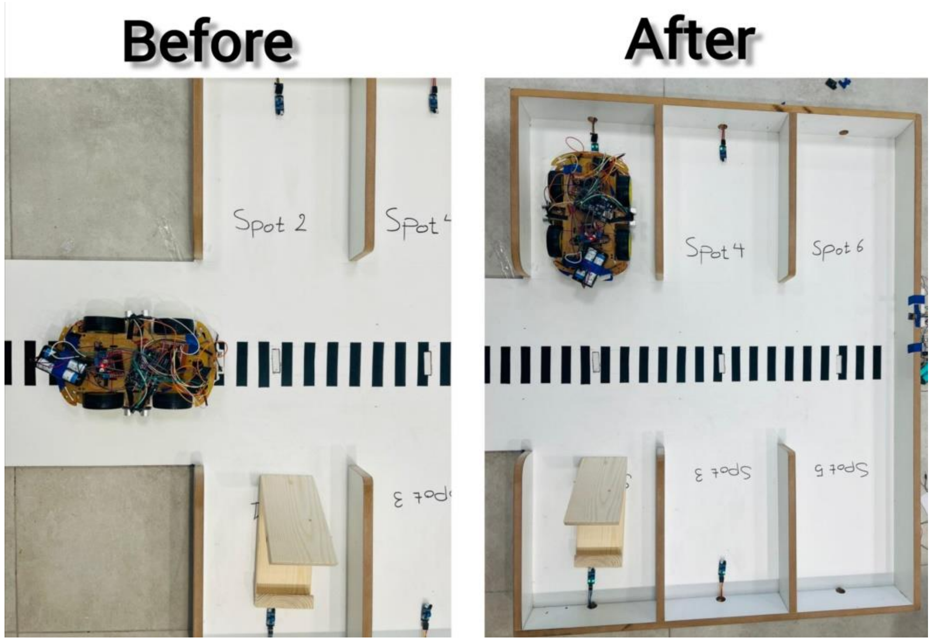


Figure 4. 5: Case1, Prototype Car Parks on First Left Spot

- **Case2: Prototype Car Parks on the Second Right Spot**

In this scenario, when the first magnet detected, the distances measured by both the right and left ultrasonic sensors are less than 40 cm; this is due to the obstacles present on both spots. Therefore, the car will continue moving until it reaches the second magnet. Upon reaching the center of the second spots, where both spots are available, and the calculated distances are more than 40 cm, the car will default to parking on the right as shown in figure 4.6.

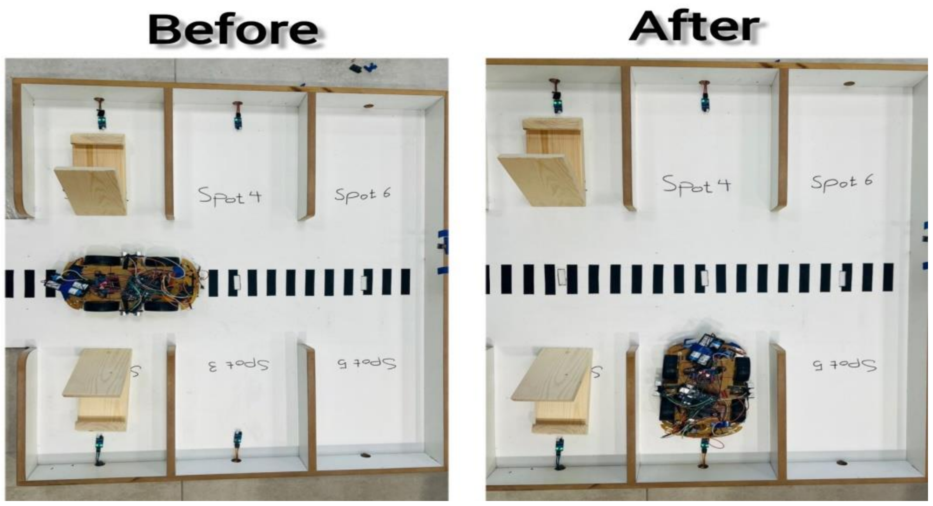


Figure 4. 6: Case2, Prototype Car Parks on the Second Right Spot

- **Case3: Prototype Car Parks on the Last Right Spot**

In this scenario, when the first and second magnet detected, the distances measured by both the right and left ultrasonic sensors are less than 40 cm in the first and second rows of spots; this is due to the obstacles present on all spots. Therefore, the car will continue moving until it reaches the last magnet. Upon reaching the center of the last spots, where both spots are available, and the calculated distances are more than 40 cm, the car will default to parking on the right as shown in figure 4.7.

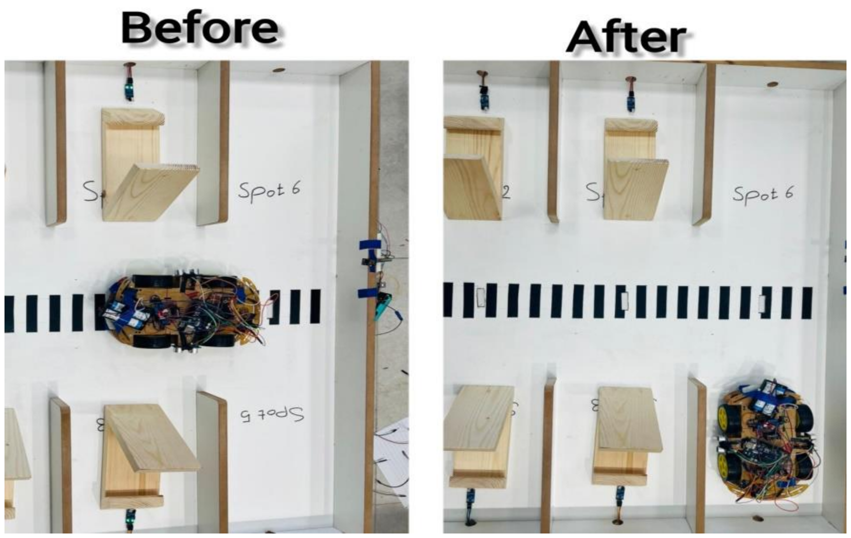


Figure 4. 7: Case3, Prototype Car Parks on the Last Right Spot

4.2.1 PARKING MODEL

As shown in figure 4.8, the infrared sensor detects the presence of a car when it reaches a parking spot and updates this information in real-time within the database.

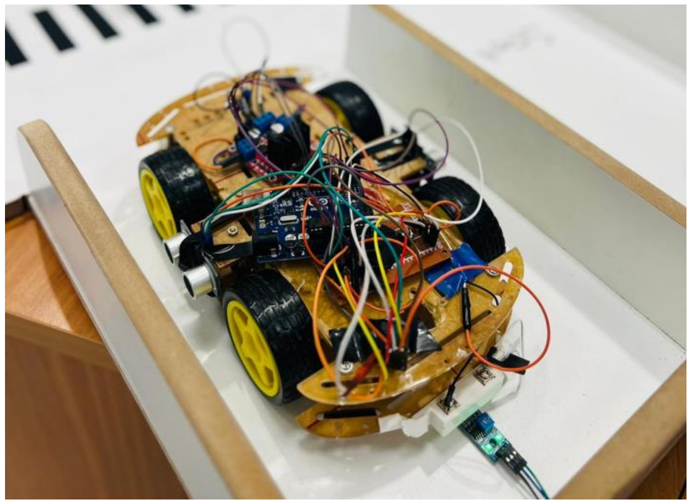
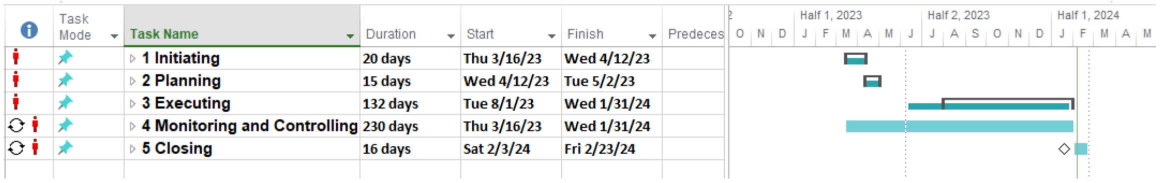


Figure 4. 8: Infrared Detection for the Prototype Car

5 CHAPTER 5: Project Management

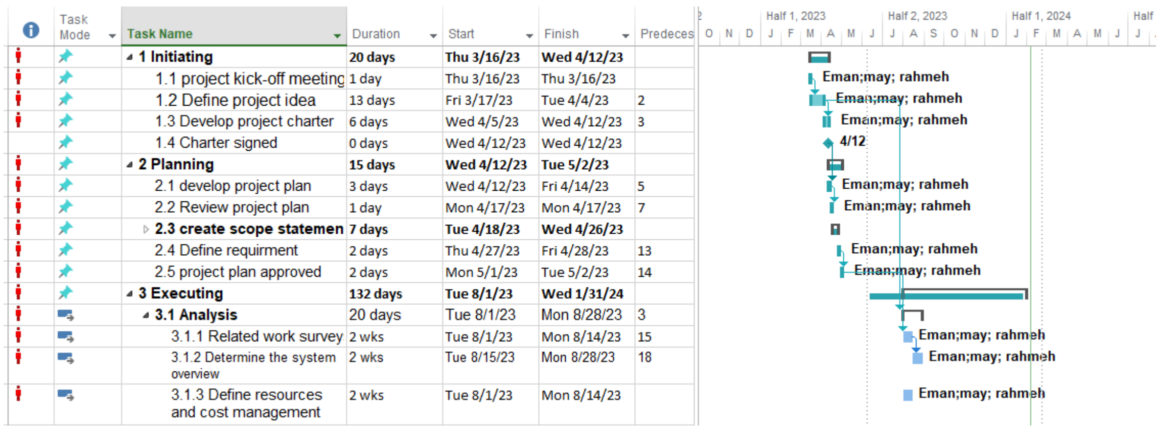
5.1 TASKS, SCHEDULE AND MILESTONES

The "Tasks, Schedule, and Milestones" section lists what needs to be done, when it's due, and who is doing it. It is like a roadmap for the project, breaking things down into manageable steps with clear deadlines. It helps us stay organized and on track toward reaching our goals.



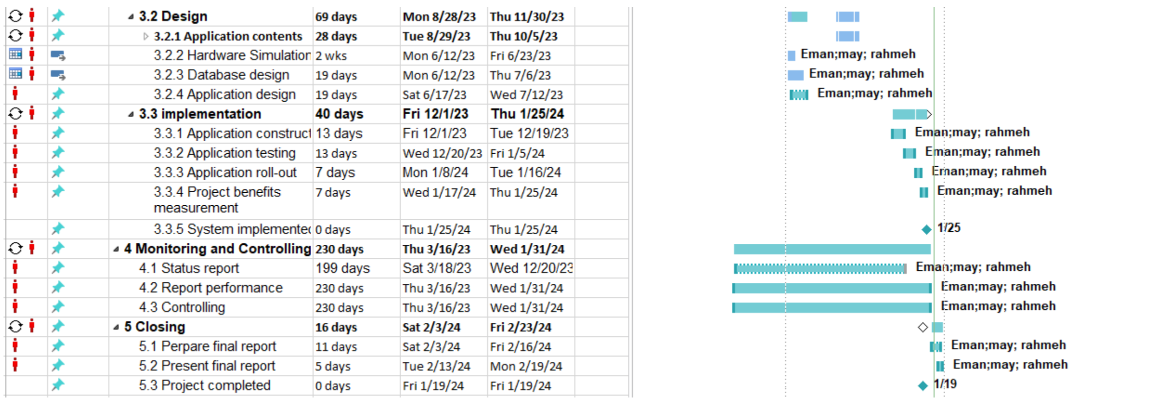
Task Name	Duration	Start	Finish	Predeces
1 Initiating	**20 days**	**Thu 3/16/23**	**Wed 4/12/23**	
2 Planning	**15 days**	**Wed 4/12/23**	**Tue 5/2/23**	
3 Executing	**132 days**	**Tue 8/1/23**	**Wed 1/31/24**	
4 Monitoring and Controlling	**230 days**	**Thu 3/16/23**	**Wed 1/31/24**	
5 Closing	**16 days**	**Sat 2/3/24**	**Fri 2/23/24**	

Figure 5. 1: Gantt Chart 1



Task Name	Duration	Start	Finish	Predeces
1 Initiating	**20 days**	**Thu 3/16/23**	**Wed 4/12/23**	
1.1 project kick-off meeting	1 day	Thu 3/16/23	Thu 3/16/23	
1.2 Define project idea	13 days	Fri 3/17/23	Tue 4/4/23	2
1.3 Develop project charter	6 days	Wed 4/5/23	Wed 4/12/23	3
1.4 Charter signed	0 days	Wed 4/12/23	Wed 4/12/23	
2 Planning	**15 days**	**Wed 4/12/23**	**Tue 5/2/23**	
2.1 develop project plan	3 days	Wed 4/12/23	Fri 4/14/23	5
2.2 Review project plan	1 day	Mon 4/17/23	Mon 4/17/23	7
2.3 create scope statemen	**7 days**	**Tue 4/18/23**	**Wed 4/26/23**	
2.4 Define requirment	2 days	Thu 4/27/23	Fri 4/28/23	13
2.5 project plan approved	2 days	Mon 5/1/23	Tue 5/2/23	14
3 Executing	**132 days**	**Tue 8/1/23**	**Wed 1/31/24**	
3.1 Analysis	20 days	Tue 8/1/23	Mon 8/28/23	3
3.1.1 Related work survey	2 wks	Tue 8/1/23	Mon 8/14/23	15
3.1.2 Determine the system overview	2 wks	Tue 8/15/23	Mon 8/28/23	18
3.1.3 Define resources and cost management	2 wks	Tue 8/1/23	Mon 8/14/23	

Figure 5. 2: Gantt Chart 2



3.2 Design	**69 days**	**Mon 8/28/23**	**Thu 11/30/23**	
3.2.1 Application contents	**28 days**	**Tue 8/29/23**	**Thu 10/5/23**	
3.2.2 Hardware Simulation	2 wks	Mon 6/12/23	Fri 6/23/23	
3.2.3 Database design	19 days	Mon 6/12/23	Thu 7/6/23	
3.2.4 Application design	19 days	Sat 6/17/23	Wed 7/12/23	
3.3 implementation	**40 days**	**Fri 12/1/23**	**Thu 1/25/24**	
3.3.1 Application construct	13 days	Fri 12/1/23	Tue 12/19/23	
3.3.2 Application testing	13 days	Wed 12/20/23	Fri 1/5/24	
3.3.3 Application roll-out	7 days	Mon 1/8/24	Tue 1/16/24	
3.3.4 Project benefits measurement	7 days	Wed 1/17/24	Thu 1/25/24	
3.3.5 System implemente	0 days	Thu 1/25/24	Thu 1/25/24	
4 Monitoring and Controlling	**230 days**	**Thu 3/16/23**	**Wed 1/31/24**	
4.1 Status report	199 days	Sat 3/18/23	Wed 12/20/23	
4.2 Report performance	230 days	Thu 3/16/23	Wed 1/31/24	
4.3 Controlling	230 days	Thu 3/16/23	Wed 1/31/24	
5 Closing	**16 days**	**Sat 2/3/24**	**Fri 2/23/24**	
5.1 Perpare final report	11 days	Sat 2/3/24	Fri 2/16/24	
5.2 Present final report	5 days	Tue 2/13/24	Mon 2/19/24	
5.3 Project completed	0 days	Fri 1/19/24	Fri 1/19/24	

Figure 5. 3: Gantt Chart 3

5.2 RESOURCES AND COST MANAGEMENT

Table 10 shows the calculations of the total price for the tools used to implement the prototype car and the parking model.

Table 10: Resources Management

	Voltage (v)	Current(mA)	Price(₪)	#unit	subtotals
Arduino UNO	7-12	40	35	2	70
Ultrasonic sensor	5	2	25	2	50
Motor driver	5	0-36	45	1	45
Infrared sensor	3.3-5	20	22	4	88
ESP8266-01	4- 6	30	35	2	70
wires	-	-	30	-	30
Car Model	-	-	110	1	110
Power supply	3.7	2	15	6	90
PCB Board	-	-	10	1	10
Parking Model	-	-	150	1	150
Total cost	713				

5.3 LESSONS LEARNED

Throughout our project work, we gained valuable insights into various aspects, notably the significance of teamwork, effective time management, and organizational skills. Pre-project planning and a thorough examination of previous projects, particularly those similar to ours, proved crucial for success. Additionally, studying other projects provided valuable lessons, enabling us to learn from their mistakes and make necessary corrections.

Our learning journey extended to encompass crucial aspects of both software and hardware. On the software side, we learned the Dart language and Flutter, which is programming language for mobile applications. Exploring Firebase databases became integral, and improving our skills in storing and retrieving data between the database and the application. On the hardware side, we gained experience in Arduino programming and data exchange among sensors, ESP, and Arduino UNO. Additionally, simulating these interactions using versatile programs like Proteus and Fritzing.

6 CHAPTER 6: IMPACT OF THE ENGINEERING SOLUTION

6.1 ECONOMICAL, SOCIETAL AND GLOBAL

The smart parking system designed to have features affect some aspects by showing real-time updates of parking spots and options to make safe park car. From an economic perspective, the system reduces the searching time for parking around the crowded city and thus economized the consumed fuel and increase and the overall productivity. Societally, having features for auto parking reduces stress with people having difficulties to make safe car park and individuals with disabilities. Furthermore, enhancing the overall safety in the roads. Globally, these systems reduce the traffic congestion in the roads and thus reflects the views of smarter and cleaner city.

6.2 ENVIRONMENTAL AND ETHICAL

The system have both environmental and ethical effects on people life. From environmental point of view, minimizing the searching time for available parking spaces reduces the carbon emissions and fuel consumption, thus leads to pure air and decreasing traffic congestion to achieve the environmental goals for reaching healthy and cleaner city. Ethically, the system can be a chance to who have difficulties in making safe car park if the prototype will developed to be a real system.

7 Chapter 7: Conclusion and Recommendation

7.1 SUMMARY OF ACHIEVEMENTS OF THE PROJECT OBJECTIVES

The primary objective for the project is to develop a prototype system encompassing both software and hardware requirements. The implemented application allow each driver to have an account. The application is designed for presenting real time information about the number of parking spots in the parking prototype by utilizing Google Maps to indicate the number of available spots on the location of our parking, which has installed infrared sensor in each spot to detect if it is vacant or not. Likewise, this parking prototype has an illustrating page to the status of each spot. Moreover, there is an option for autonomous park car, when this clicked the car starts searching by itself about the suitable spot through the usage of both reed switch at the center of each row of spots and installed ultrasonic sensors, which check the distances availability. These hardware components establish a connection with the application via Wi-Fi by using ESP chip. At the end of this semester, the project met the basic requirements and some enhancements to refine the prototype system.

7.2 NEW SKILLS AND EXPERIENCES LEARNT

At the end of senior project, there was a valuable experiences and knowledge acquired. Due to the combination of hardware and software elements in the project, we had an opportunity to have experiences in both sides. From the hardware side, the system had engagement with technologies such as Arduino microcontroller, ESP chips and various sensors. Starting from learning the connection between them, programming them using Arduino IDE and integrating them with the application using Wi-Fi protocols. In addition to that, we learnt Dart and Flutter language for the application development and the incorporation of Firebase, which is a tool to store and exchange data between the hardware components and the app. Finally, it was a year full of time management and collaborative teamwork.

7.3 RECOMMENDATION FOR FUTURE WORK

Looking forward, our hope is to implement our prototype system to be real system in our cities. Collaborating largest possible number of parking, to have more options of parking in the Google Map to the drivers registered with the application and increase the efficiency of the overall parking work. Moreover, we want to have more options in the application such as payment and reservation options to enhance utility. As engineers, we have a goal by using the experiences and advanced technologies with global companies to make our cities more managed, smarter and healthy.

PREFERENCES

[1] "مركز الاحصاء الفلسطيني," [Online]. Available: https://www.pcbs.gov.ps/statisticsIndicatorsTables.aspx?lang=ar&table_id=460. [Accessed 5 4 2023].

[2] "ResearchGate," [Online]. Available: https://www.researchgate.net/publication/230701092_Smart_Parking_System_SPS_Architecture_Using_Ultrasonic_Detector. [Accessed 5 5 2023].

[3] "AUTODESK Instructables," [Online]. Available: https://www.instructables.com/Autonomous-Parallel-Parking-Car-Making-Using-Ardui/. [Accessed 5 5 2023].

[4] "ResearchGate," [Online]. Available: https://www.researchgate.net/publication/344411337_The_Smart_Parking_Management_System. [Accessed 7 5 2023].

[5] "CIRCUIT DIGEST," [Online]. Available: https://circuitdigest.com/article/different-types-of-arduino-boards. [Accessed 29 5 2023].

[6] "Arduino Documentation," [Online]. Available: https://docs.arduino.cc/hardware/uno-rev3. [Accessed 5 29 2023].

[7] "MaxBotix," [Online]. Available: https://maxbotix.com/blogs/blog/how-ultrasonic-sensors-work. [Accessed 31 5 2023].

[8] "CIRCUIT DIGEST," [Online]. Available: https://circuitdigest.com/microcontroller-projects/interfacing-ir-sensor-module-with-arduino. [Accessed 1 6 2023].

[9] "Smart PROTOTYPING," [Online]. Available: https://www.smart-prototyping.com/L298N-Dual-H-bridge-Motor-Driver-Board. [Accessed 1 6 2023].

[10] "BYJU'S," [Online]. Available: https://byjus.com/physics/dc-motor/. [Accessed 1 6 2023].

[11] "Last Minute ENGINEERS," [Online]. Available: https://lastminuteengineers.com/reed-switch-arduino-tutorial/?fbclid=IwAR2D7ozIcZ1OAKFirZfhMSnHHuni12dhjJ_YeT8gjDY9e3MrBvaddc9Qv_s#google_vignette. [Accessed 26 9 2023].

[1 2] "AUTODESK Instructables," [Online]. Available: https://www.instructables.com/Getting-Started-With-the-ESP8266-ESP-01/. [Accessed 7 10 2023].

[1 3] "Real-Time Database of Smart Parking System," [Online]. Available: https://console.firebase.google.com/u/0/project/smartcarparking-437e3/database/smartcarparking-437e3-default-rtdb/data. [Accessed 7 2 2023].

APPENDICES

Appendix A:

ARDUINO CODE FOR THE PROTOTYPE CAR:

```
 const int in5 = 4;  ////right
const int in6 = 2;  // right
const int in7 = 12; // left
const int in8 = 13;//left
const int pinSwitch =7;
int StatoSwitch = 0;
const int trigPin = 8;
const int echoPin = 9;
const int trigPin2 = 10;
const int echoPin2 = 11;


void setup() {
  Serial.begin(9600);
  pinMode(pinSwitch, INPUT_PULLUP);
  pinMode(in5, OUTPUT);
  pinMode(in6, OUTPUT);
  pinMode(in7, OUTPUT);
  pinMode(in8, OUTPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);

}
void park(){

        while(StatoSwitch == HIGH){
        StatoSwitch = digitalRead(pinSwitch);
        Serial.println(StatoSwitch);
        moveForward();
        delay(20);
         if(StatoSwitch == LOW){
          moveForward();
          delay(120);
          stopMotors();
          delay(1000);
          int d = measureDistance(trigPin,echoPin);//distance ultra right
          int d2=measureDistance(trigPin2,echoPin2);//distance ultra left
          Serial.print("distance = ");
          Serial.print(d);
          Serial.println("cm\n");
          Serial.print("distance2 = ");
          Serial.print(d2);
          Serial.println("cm\n");
          Serial.println("============================================");

          if(d>=40 && d2<40){
            Serial.println("Park Right");
            parkRight();

          }  else if(d2>=40 && d<40){
            Serial.println("Park left");
            parkLeft();
          }  else if(d>=40 && d2>=40){
            Serial.println("Park Right");
            parkRight();

          } else if(d<=40 && d2<=40){
            StatoSwitch = HIGH ;
            park();
          }
          }
          }
}
void moveBackward() {
  digitalWrite(in5, HIGH);
  digitalWrite(in6, LOW);
  digitalWrite(in7, LOW);
  digitalWrite(in8, HIGH);

}
void moveForward() {
  digitalWrite(in5, LOW);
  digitalWrite(in6, HIGH);
  digitalWrite(in7, HIGH);
  digitalWrite(in8, LOW);

}
void turnleft() {
  digitalWrite(in5, LOW);
  digitalWrite(in6, HIGH);
  digitalWrite(in7, LOW);
  digitalWrite(in8, LOW);

}
void turnRight() {
    digitalWrite(in5, LOW);
  digitalWrite(in6, LOW);
  digitalWrite(in7, HIGH);
  digitalWrite(in8, LOW);
```

```
}

void stopMotors() {
  digitalWrite(in7, LOW);
  digitalWrite(in8, LOW);
  digitalWrite(in5, LOW);
  digitalWrite(in6, LOW);
}
int measureDistance(int t ,int e) {
  digitalWrite(t, LOW);
  delayMicroseconds(2);
  digitalWrite(t, HIGH);
  delayMicroseconds(10);
  digitalWrite(t, LOW);

  long duration = pulseIn(e, HIGH);
  int distance = duration / 58;

  return distance;
}
void parkRight(){
  moveBackward();
  delay(70);
  stopMotors();
  delay(200);
  turnRight();
  delay(600);
  stopMotors();
  delay(200);
  moveForward();
  delay(400);
  stopMotors();
}
void parkLeft(){
  moveBackward();
  delay(120);
  stopMotors();
  delay(200);
  turnleft();
  delay(650);
  stopMotors();
  delay(200);
  moveForward();
  delay(400);
  stopMotors();
}

void loop(){
     if (Serial.available() > 0) {
    // Read the incoming byte

    String command = Serial.readStringUntil('\n');

    // Check received command and perform actions
    if (command == "forward") {
      moveForward(); // move DC forward
      delay(1000);
      stopMotors();
      Serial.println("forward");
    } else if( command == "stop") {
      stopMotors(); // move DC backward

      Serial.println("stop");
    }else if( command == "backward") {
      moveBackward() ;// move DC backward
      delay(1000);
      stopMotors();
      Serial.println("backward");
    }else if( command == "left") {
      turnleft(); // move DC backward
      delay(800);
      stopMotors();
      Serial.println("left");

    }else if( command == "right") {
      turnRight(); // move DC backward
      delay(700);
      stopMotors();
      Serial.println("right");

    }
    else if( command == "Autopark") {
       StatoSwitch = digitalRead(pinSwitch);
         Serial.print("===StatoSwitch ===");
         Serial.println(StatoSwitch);
         park();
  }

  }
  }
```

ESP CODE FOR THE PROTOTYPE CAR:

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>

const char* ssid = "HUAWEI Y9 2019";
const char* password = "123456789";
```

```
ESP8266WebServer server(80);

void setup() {
 Serial.begin(9600);
 delay(10);

 // Connect to Wi-Fi
 WiFi.begin(ssid, password);

 while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.println("Connecting to WiFi...");
 }

 Serial.print("Connected to WiFi : ");
 Serial.println(WiFi.localIP());
 // Setup the web server


 server.on("/forward", HTTP_GET, [](){
  // Pass the command to Arduino over the serial connection
  Serial1.println("forward\n");
  Serial.println("forward\n");
  delay(1000);
  server.send(200, "text/plain", "Forward command sent to Arduino");
 });

 server.on("/backward", HTTP_GET, [](){
  // Pass the command to Arduino over the serial connection
  Serial1.println("backward\n");
  Serial.println("backward\n");
  delay(1000);
  server.send(200, "text/plain", "Backward command sent to Arduino");
 });

 server.on("/right", HTTP_GET, [](){
  // Pass the command to Arduino over the serial connection
  Serial.println("right\n");
  Serial1.println("right\n");
  delay(1000);
  server.send(200, "text/plain", "Right command sent to Arduino");
 });

 server.on("/left", HTTP_GET, [](){
  // Pass the command to Arduino over the serial connection
  Serial1.println("left\n");
  Serial.println("left\n");
  delay(1000);
  server.send(200, "text/plain", "Left command sent to Arduino");
 });

 server.on("/stop", HTTP_GET, [](){

  Serial1.println("stop\n");
  // Pass the command to Arduino over the serial connection
  Serial.println("stop\n");
  delay(1000);
  server.send(200, "text/plain", "Stop command sent to Arduino");
 });

 server.on("/Autopark", HTTP_GET, [](){

  Serial1.println("Autopark\n");
  // Pass the command to Arduino over the serial connection
  Serial.println("Autopark\n");
  delay(1000);
  server.send(200, "text/plain", "Autopark command sent to Arduino");
 });
 server.begin();
}

void loop() {
 server.handleClient();
 // Other code here (optional)
}
```

Appendix B:

ARDUINO CODE FOR PARKING MODEL:

```
const int irSensor1 = 2;
const int irSensor2 = 3;
const int irSensor3 = 4;
const int irSensor4 = 5;
 // Replace with your actual pin number
int s= 0;
void setup() {
 pinMode(irSensor1, INPUT);
 pinMode(irSensor2, INPUT);
 pinMode(irSensor3, INPUT);
 pinMode(irSensor4, INPUT);
 Serial.begin(9600);
}

void loop() {
 int irSensorValue1 = digitalRead(irSensor1);
 int irSensorValue2 = digitalRead(irSensor2);
 int irSensorValue3 = digitalRead(irSensor3);
 int irSensorValue4 = digitalRead(irSensor4);
 if(irSensorValue1 == 0){
   Serial.print("valid1\n");
   delay(50);
 }else{
   Serial.print("Invalid1\n");
   delay(50);
 }
  if(irSensorValue2 == 0){
   Serial.print("valid2\n");
   delay(50);
 }else{
   Serial.print("Invalid2\n");
   delay(50);
 }

  if(irSensorValue3 == 0){
   Serial.print("valid3\n");
   delay(50);
 }else{
   Serial.print("Invalid3\n");
   delay(50);
 }

  if(irSensorValue4 == 0){
   Serial.print("valid4\n");
   delay(50);
 }else{
   Serial.print("Invalid4\n");
   delay(50);
 }
 }
```

ESP CODE FOR PARKING MODEL:

ESP code with Arduino and firebase,see more details here [13]:

```
#include <ESP8266Firebase.h>
#include <ESP8266WiFi.h>

#define _SSID "HUAWEI Y9 2019"          // Your WiFi SSID
#define _PASSWORD "123456789"
#define REFERENCE_URL "https://smartcarparking-437e3-default-rtdb.firebaseio.com/"  //
Your Firebase project reference url

Firebase firebase(REFERENCE_URL);

void setup() {


  Serial.begin(9600);
```

```
 Serial.print("Connecting to: ");
 Serial.println(_SSID);
 WiFi.begin(_SSID,_PASSWORD);

 while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print("-");
 }
 Serial.println("");
 Serial.println("WiFi Connected");

 // Print the IP address
 Serial.print("IP Address: ");
 Serial.print("http://");
 Serial.print(WiFi.localIP());
 Serial.println("/");
}
void loop() {

 if (Serial.available() > 0) {
   // Read the incoming byte
   String command = Serial.readStringUntil('\n');
   // Check received command and perform actions
   if (command == "valid1") {
     Serial.println("valid from Arduino .. ");
     firebase.setInt("parking_status/spot1",1);
   } else if( command == "Invalid1") {
     Serial.println("Invalid from Arduino .. ");
     firebase.setInt("parking_status/spot1",0);
   }if (command == "valid2") {
     Serial.println("valid from Arduino .. ");
     firebase.setInt("parking_status/spot2",1);
   } else if( command == "Invalid2") {
     Serial.println("Invalid from Arduino .. ");
     firebase.setInt("parking_status/spot2",0);
   }if (command == "valid3") {
     Serial.println("valid from Arduino .. ");
     firebase.setInt("parking_status/spot3",1);
   } else if( command == "Invalid3") {
     Serial.println("Invalid from Arduino .. ");
     firebase.setInt("parking_status/spot3",0);
   }if (command == "valid4") {
     Serial.println("valid from Arduino .. ");
     firebase.setInt("parking_status/spot4",1);
   } else if( command == "Invalid4") {
     Serial.println("Invalid from Arduino .. ");
     firebase.setInt("parking_status/spot4",0);
   }
 }}
```