UNIVERSITY OF TURKISH AERONAUTICAL

ASSOCIATIONFACULTY OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND ELECTRONICSENGINEERING

FOLDABLE AUTONOMOUS VEHICLE

Ahmet Necati Uysal-Furkan Batuhan Aslan

180441024-180441053

Senior Design Project I

EEE 495

Project Supervisor: ASSIST. PROF. ÖZGÜR KELEKÇİ

January, 2023

ABSTRACT

FOLDABLE AUTONOMOUS VEHICLE

Ahmet Necati Uysal

Furkan Batuhan Aslan

Department of Electrical and Electronics Engineering

Project Supervisor: Assist. Prof. Özgür KELEKÇİ

The general operating principle of the foldable autonomous vehicle, following the data coming from the sensors connected to the central control unit (length of the parking space and the distance to the obstacle), the length of the vehicle changes, and it performs the autonomous parking process by reaching the length that it can park. So,people will be able to find parking space easily.

We used Arduino Uno, servo motor, an ultrasonic distance sensor, GPS module and DC motor in our project. Servo motors helped fold the vehicle. Thus, in places where the vehicle does not fit, the vehicle has reached the size where it can be folded and parked. With the data received from the GPS, the location of the vehicle is instantly displayed on the application and the autonomous parking process is carried out in line with this data. C++ and necessary libraries are used in the construction of the tool.

TABLE OF CONTENTS

ABSTRACT .... **i**
TABLE OF CONTENTS.. **ii**
LIST OF FIGURES .... **iii**
1. INTRODUCTION.. **1**
1.1 What is Autonomous Park.. 1
1.2 How does it contribute to the autonomous parking technology of foldable vehicles?.. 1
1.3 Project Structure 2
1.4 Purpose of The Project 3
2. CONSTRUCTION OF THE PROJECT.. **3**
2.1 Hardware 3
2.1.1 Project Materials.. 4
2.1.2 Why did we use servo motor? 5
2.1.3 Why did we use NodeMcu Ch-340?.. 5
2.1.4 Why did we use GPRS 5
2.2 Software 6
2.2.1 Project Schema 8
2.2.2 Project Code 9
2.2.4 Mapping With GSM Shield and Arduino UNO.. 12
2.2.5 Mapping with Nodemcu and Bylnk App.. 14
3. APPLICATION RESULTS.. **16**
4. CONCLUSION.. **16**
5. REFERENCES..**17**

LIST OF FIGURES

Figure 1.1: Autonomous Parking Operation).. 1
Figure 1.2: Beginning of Measurements for Autonomous Parking... 2
Figure 1.3: Completion of Measurements for Autonomous Parking....................................... 2
Figure 1.4: Parking Maneuvers 2
Figure 1.5: Completing the Autonomous Parking Process.. 2
Figure 1.6: Unfolded Vehicle.. 3
Figure 1.7: Coated Vehicle with the Help of Servos.. 3
Figure 2.1: Block Diagram of Autonomous Foldable Vehicle.. 4
Figure 2.2: MG995 Servo Motor.. 5
Figure 2.3: NodeMcu ESP8266... 5
Figure 2.4: GY-NEO6MV2 and GPS Module... 6
Figure 2.5: HCSR-04 Distance Measurement.. 7
Figure 2.6: GPRS Module Connected to GSM Shield.. 13
Figure 2.7: Getting Instant Location as Message Using GSM Shield................................... 13
Figure 2.8: GPRS Module Connected to Nodemcu... 15
Figure 2.9: Seeing Instant Location with BLYNK application..15

1. INTRODUCTION

Since the first automobile equipped with an internal combustion engine powered by gasoline was built by German engineer Karl Benz in 1885, cars have undergone a great evolution. The increasing number of vehicles in use has brought various problems, one of which is the problem of parking cars. Today, the parking problem is solved by human power and some car park sensors and rearview cameras. In addition, autonomous technologies are developing in today's cars. For example, (semi-autonomous driving, lane tracking sensor, fatigue detection sensor, and camera-assisted park view). However, if the resources are provided, these technologies can be used. In this project, thanks to the servo motors used, parking can also be made in areas where there are no facilities. In the study, the kinematic model was used and the controller was designed under certain constraints. Before the parking process, the kinematics of the vehicle was used according to the standing angle of the vehicle and the size of the parking area to try to automatically perform the parking process. Different situations have been defined based on the data collected from the sensors. In situations where the empty parking area is narrower than the length of the vehicle, it is aimed to park the vehicle by folding it at the front and rear parts with the motors located there. In his study, Zengin E. developed a mobile vehicle that can find a parking space with smooth ground, flat surface obstacles and park in a parallel position with the least maneuvering.

1.1 What is Autonomous Park ?

It is the parking of the vehicle in the parking place it deems appropriate without the need for human power. It aims to increase driving comfort and safety in constrained environments where a lot of attention and experience are required to steer the vehicle. The parking maneuver is performed by controlling the steering angle and vehicle speed, which takes into account the surrounding situation and obstacles to ensure collision-free movement within the available space. The autonomous parking in our project is done in line with the data from the ultrasonic distance sensors on the vehicle.

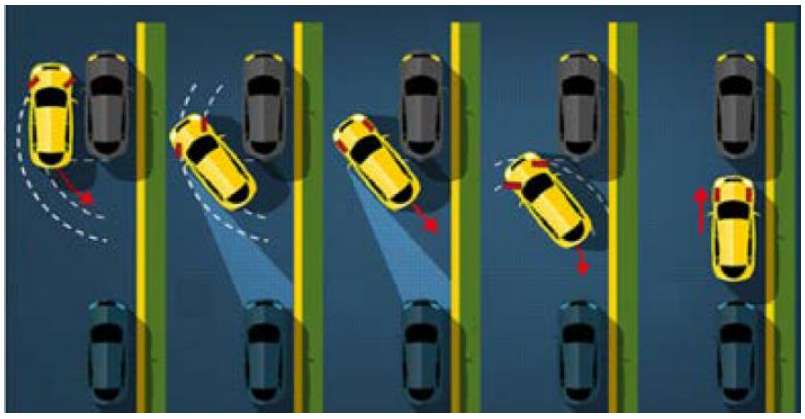


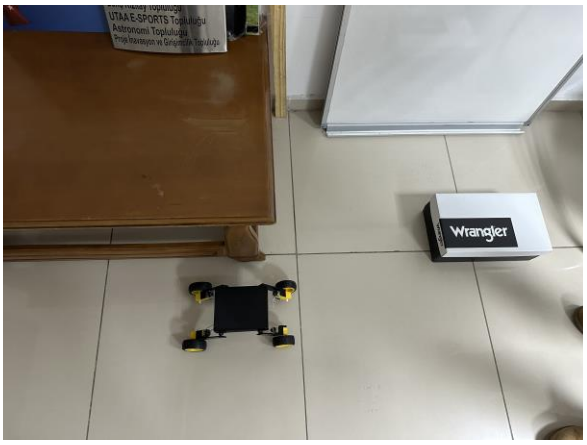
Figure 1.1 Autonomous Parking Operation)

1.2. How does it contribute to the autonomous parking technology of foldable vehicles?

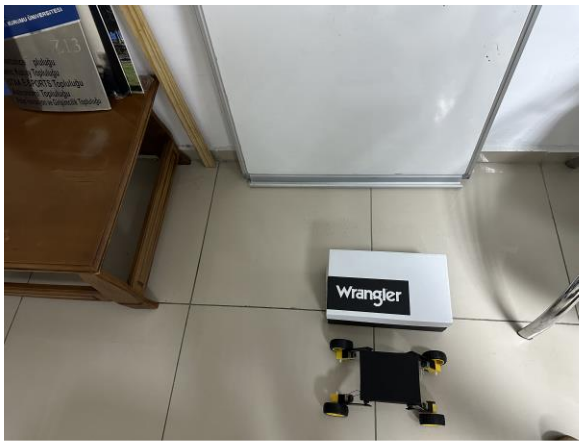
Since normal vehicles are connected to a single axle, they cannot be folded. However, with the help of the servo motor located between the body and wheels of the vehicle we have developed, can reach smaller dimensions he vehicle than its own size by folding. In this way, it can easily park in tight spaces.

1.3. Project Structure

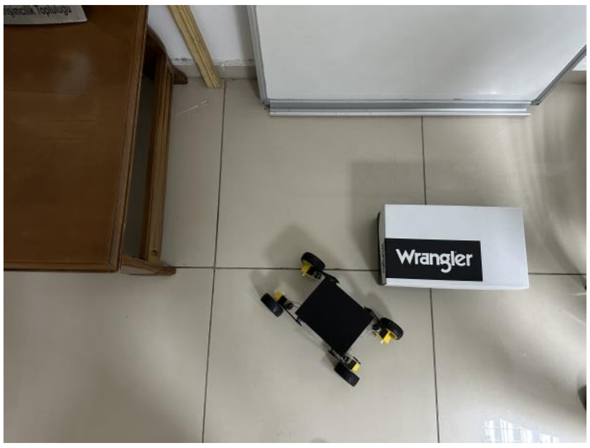
The information from the sensors enables the Arduino to park the vehicle in empty spaces below a certain distance through codes written in C++. Different conditions are defined for the parking process and the parking process is performed according to these conditions. In cases where the vehicle cannot fit into the spaces in the parking space, the system changes the size of the vehicle by rotating the servo motors approximately 90 degrees. In addition, with the GPS module in the vehicle, the location of the vehicle can be determined from within the application and from the map as a message. In Figure 1.2, a photograph taken at the beginning of the parking lot determination during the application is seen. Figure 1.3 shows the implementation of autonomous parking steps. Some photos of the folding and autonomous parking of the developed prototype vehicle are given in Figure 1.6 and Figure 1.7 during the application. It has been examined in 2 cases: the case where the vehicle parking space is smaller than the vehicle size and the situation where there is a suitable parking space between the two vehicles.



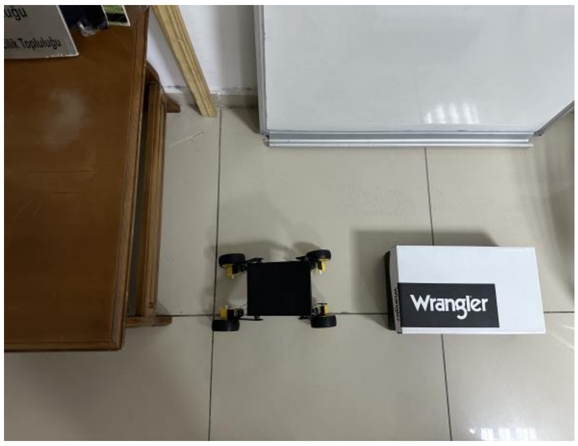
(Figure 1.2 Beginning of Measurements for Autonomous Parking)



(Figure 1.3 Completion of Measurements for Autonomous Parking)



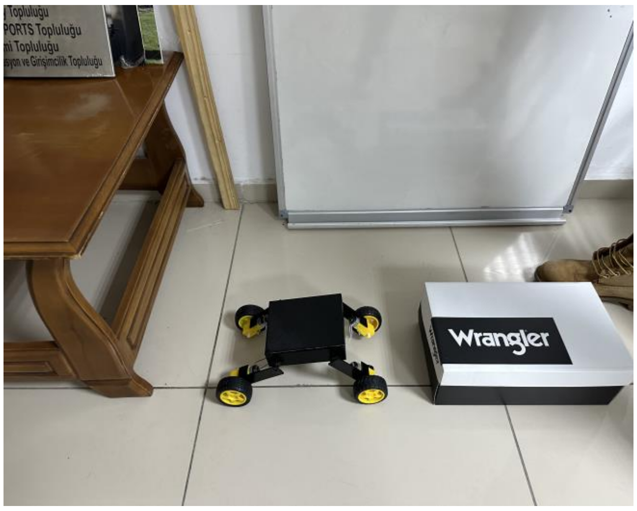
(Figure 1.4 Parking Maneuvers)



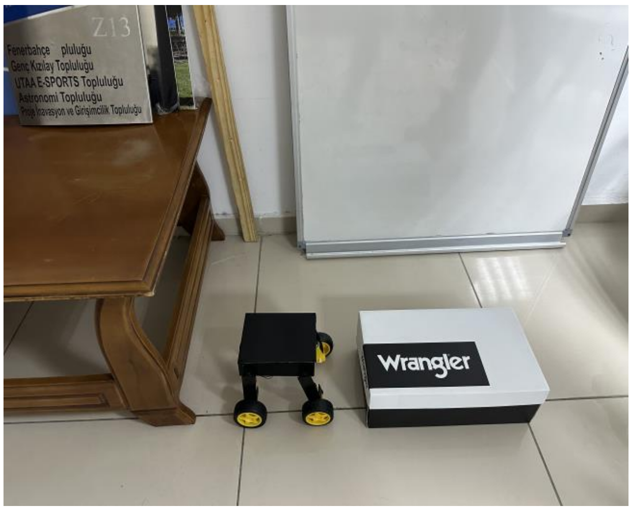
(Figure 1.5 Completing the Autonomous Parking Process)

1.4 Purpose of the Project

In this study, the problem of automatically parking cars and similar vehicles was addressed. The aim is to enable drivers to park their vehicles more easily, especially in crowded and hard-to-find parking areas, with a system that allows the vehicle to fold and park itself. The developed hardware and software help calculate whether there is enough space to park, and if so, the vehicle can be automatically parked with the help of sensors on the vehicle without the driver's intervention. If there is not enough space, the servo motors on the vehicle can fold and park in that area.



(Figure 1.6 Unfolded Vehicle)



(Figure 1.7 Coated Vehicle with the Help of Servos)

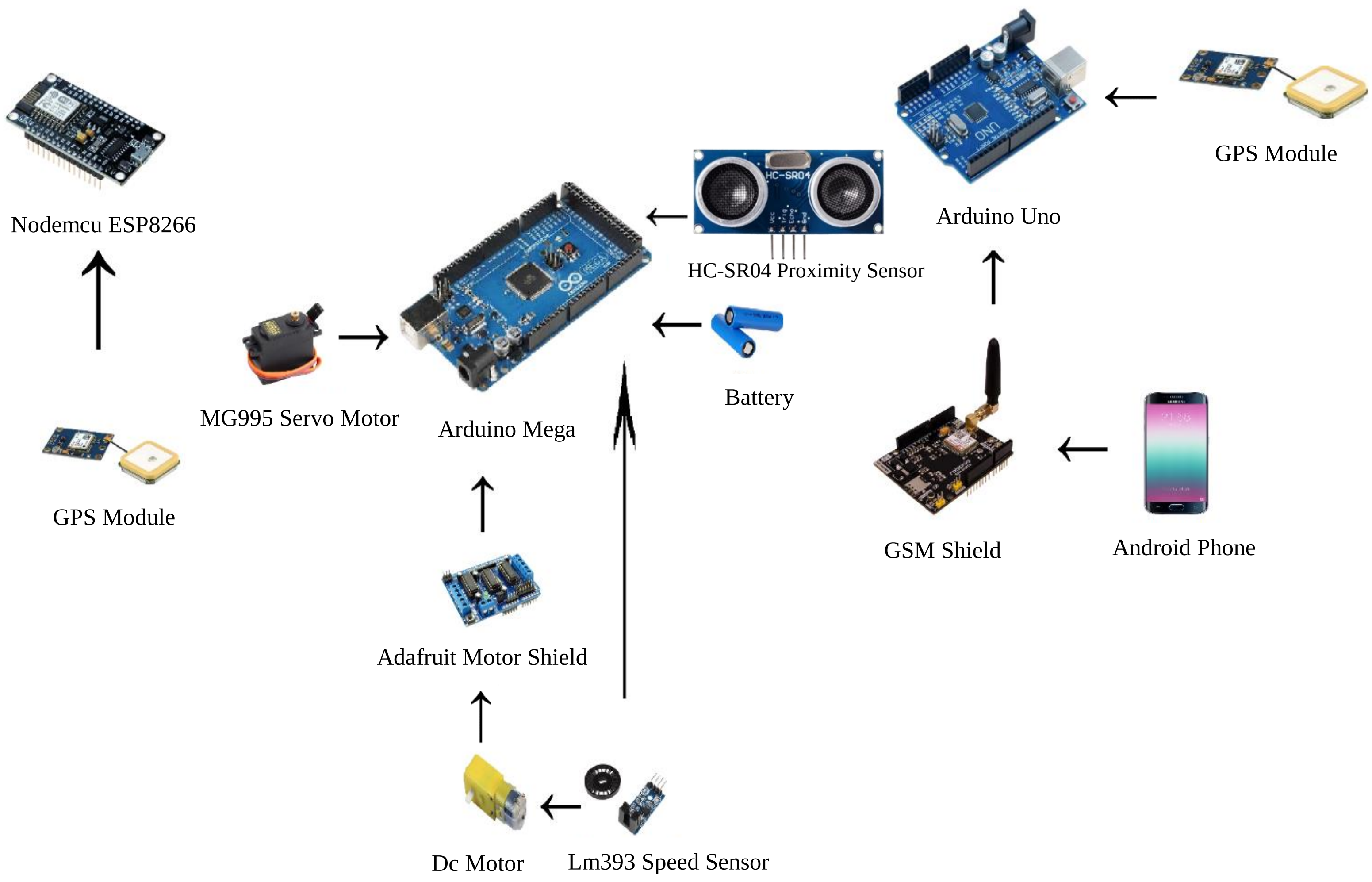
2. CONSTRUCTION OF THE PROJECT

2.1 HARDWARE

In the scope of this study, the chassis of the vehicle was printed using a 3D printer, and an Arduino Mega board was used. The DC motors were connected to the Adafruit motor driver board, 4 ultrasonic distance sensors were placed inside the vehicle and connected to the Arduino. The ultrasonic sensors (HC-SR04) were mounted on the front, back and right rear of the vehicle at different angles. The location of the vehicle can be seen through the GPRS module installed in the vehicle. To be able to see the location of the vehicle on a mobile application, the vehicle needs to be connected to the internet, and for this purpose, a NodeMcu ESP8266 was used. The Bylnk application was used to track the location of the vehicle on the map. In addition, the location of the vehicle can be obtained as a message thanks to the GSM Module used in the vehicle.

2.1.1 Project Materials

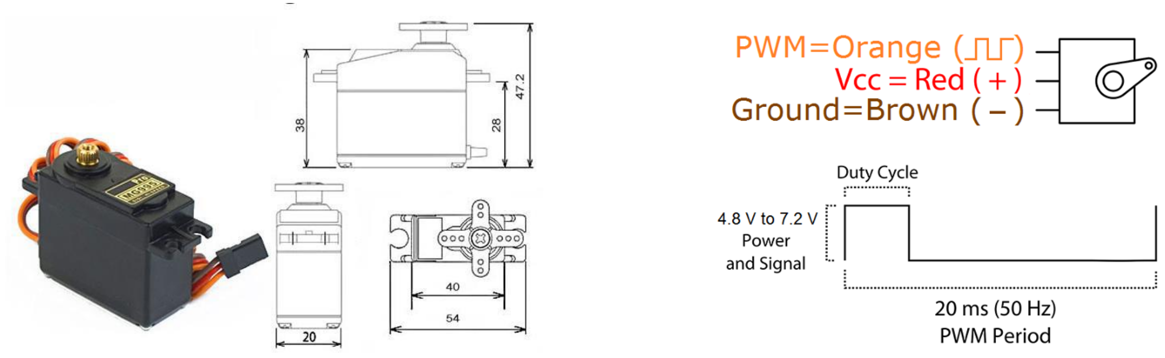
-Arduino Mega
-Arduino UNO
-Adafruit Motor Shield
-HC-SR04 Ultrasonic Proximity Sensor
-MG995 12 kg Servo Motor
-Lm393 Speed Sensor
-Lipo Battery
-GY-NEO6MV2 GPS Module
-Nodemcu ESP8266
-GSM Shield
-6V 250RPM DC motor



(Figure 2.1 Block Diagram of Autonomous Foldable Vehicle)

2.1.2 Why did we use servo motor ?

People may have difficulty parking their vehicles in gaps close to the length of their vehicle. In this project, thanks to the servo motors placed at the connection points of the vehicle, the length of the vehicle can be shortened and the vehicle will be able to park comfortably in places where it will be more difficult to fit.



(Figure 2.2 MG995 Servo Motor)

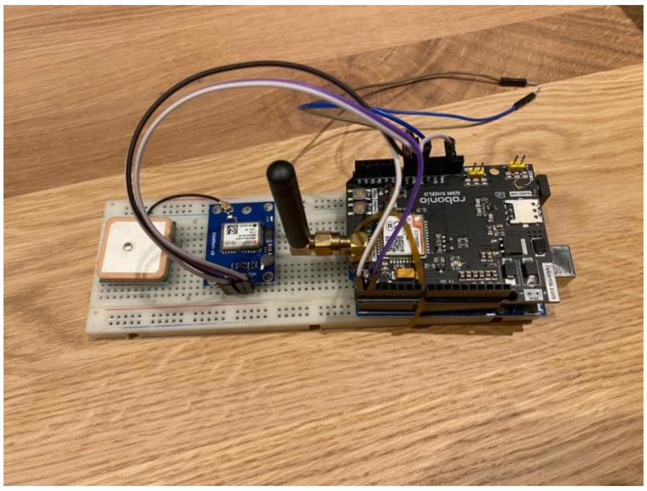
2.1.3 Why did we use NodeMcu ESP8266 ?

In order to be able to see the location of our vehicle on the internet, we use a circuit board in addition to the GPRS. With this card, we can connect our Arduino to the internet and obtain the location information on the internet. The NodeMcu ESP8266 works with low voltage energy and has a large number of connection points. Using these connection points, we can control other electronic components that you will connect. Its built-in WiFi enables us to easily create IOT devices, also known as internet of things.

(Figure 2.3 NodeMcu ESP8266)

2.1.4 Why did we use GPRS ?

We used the GPS module to learn the location of our vehicle. The GY-NEO6MV2 module on this card is a product that can be used in many projects, including flight control systems, to control and track location. This module is of high quality and has high accuracy. It has an accuracy of about 5 meters

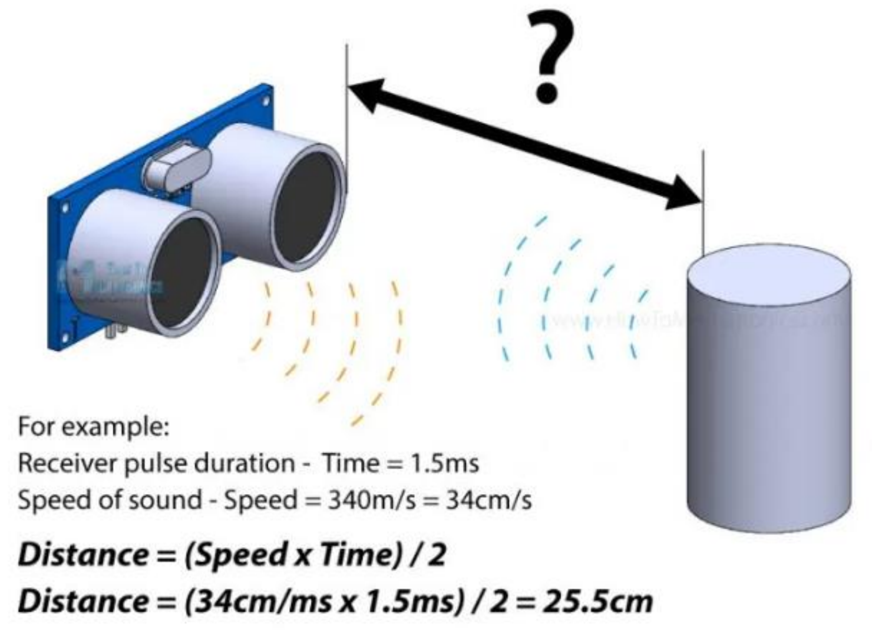

(Figure 2.4 GY-NEO6MV2 and GPS Module)

2.2 SOFTWARE

An Arduino Mega 2560 microcontroller board has been used to collect and interpret sensor information, generate input signals for the motor driver, and calculate the necessary maneuvers for parking in the prototype vehicle. The algorithm developed for the general operating principle allows the vehicle to be controlled wirelessly (via Bluetooth) and to park autonomously. The autonomous parking embedded software algorithm is divided into two parts: finding a suitable parking spot and performing the parking maneuver. When the driver arrives at the area where they want to park parallel, they give the autonomous parking command through the mobile application. After detecting this command sent via the wireless communication module, the vehicle starts to look for a suitable parking spot. Two different situations are considered in the determination of the parallel parking spot. In the first case, the situation of parking between two vehicles is considered, while in the second case, the situation of parking in front of a vehicle is considered. Using the ultrasonic sensors placed around the vehicle, the algorithm searches for suitable parking spots according to the distance the vehicle can park. If the distance at which parking is desired is not suitable for the vehicle to park, the vehicle folds using the servo motors and then starts the parking process. If the distance at which parking is desired is suitable for the vehicle to park, the algorithm directs the vehicle to park in that area. If an obstacle is detected in front of the vehicle during the parking spot determination process, the automatic braking system of the vehicle is activated. While parking autonomously, the ultrasonic sensor in front of the vehicle tries to prevent possible accidents by activating the automatic braking system if any obstacle is detected. There are two driving modes for the vehicle: the normal remote control mode and the autonomous parking mode. If the autonomous parking mode is selected for the vehicle, the algorithm first checks that the distance between the vehicle and the obstacle on the side where it will be parked is at least 32. If the distance to the right of the vehicle to be parked is greater than 32 cm, the second distance measurement is taken. If the second measured distance is greater than the first value, it means that the vehicle has moved from the obstacle to the space between the vehicles. After this, the counter is started and the next distance measurement is calculated. For the prototype vehicle used in this study, the acceptable parking distance was determined as 35 cm and the algorithm was run according to this value.

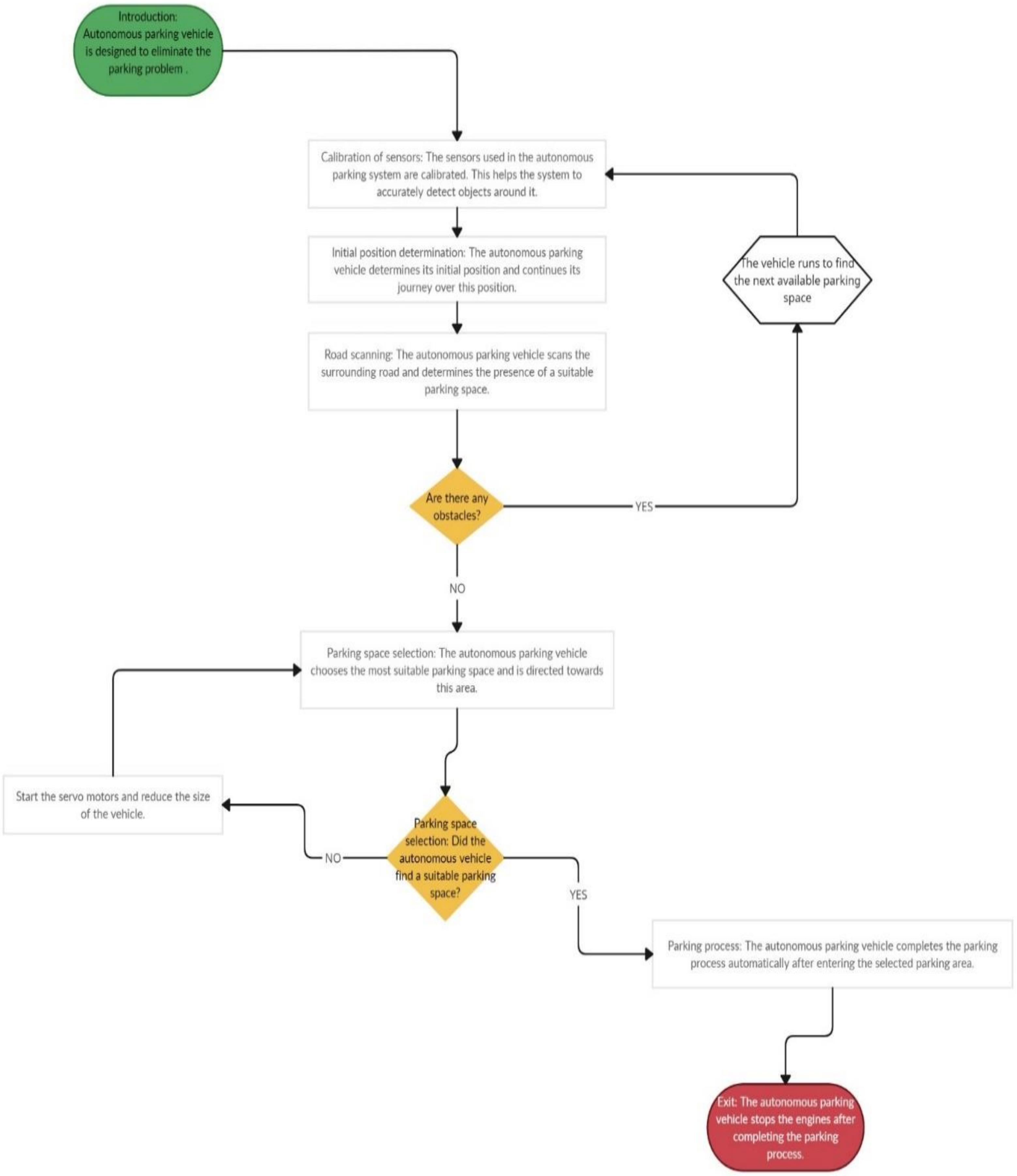
If the measured distance value is greater than the previously determined parking distance of the vehicle, the algorithm looks at two scenarios for finding a parking spot and starts the autonomous parking process, if the measured distance is small, the vehicle folds and shrinks and starts the autonomous parking process.

In this study, the HC-SR04 ultrasonic sensors are used to send a sound wave (echo) to the environment and detect the reflected sound wave. The microcontroller's timer is used to calculate the time difference between the two waves in microseconds.
The speed of sound in air is approximately 343 m/s and the distance between the obstacle and the vehicle is measured using the calculated time and the product of the speed of sound, as shown in Equation 2.2.1

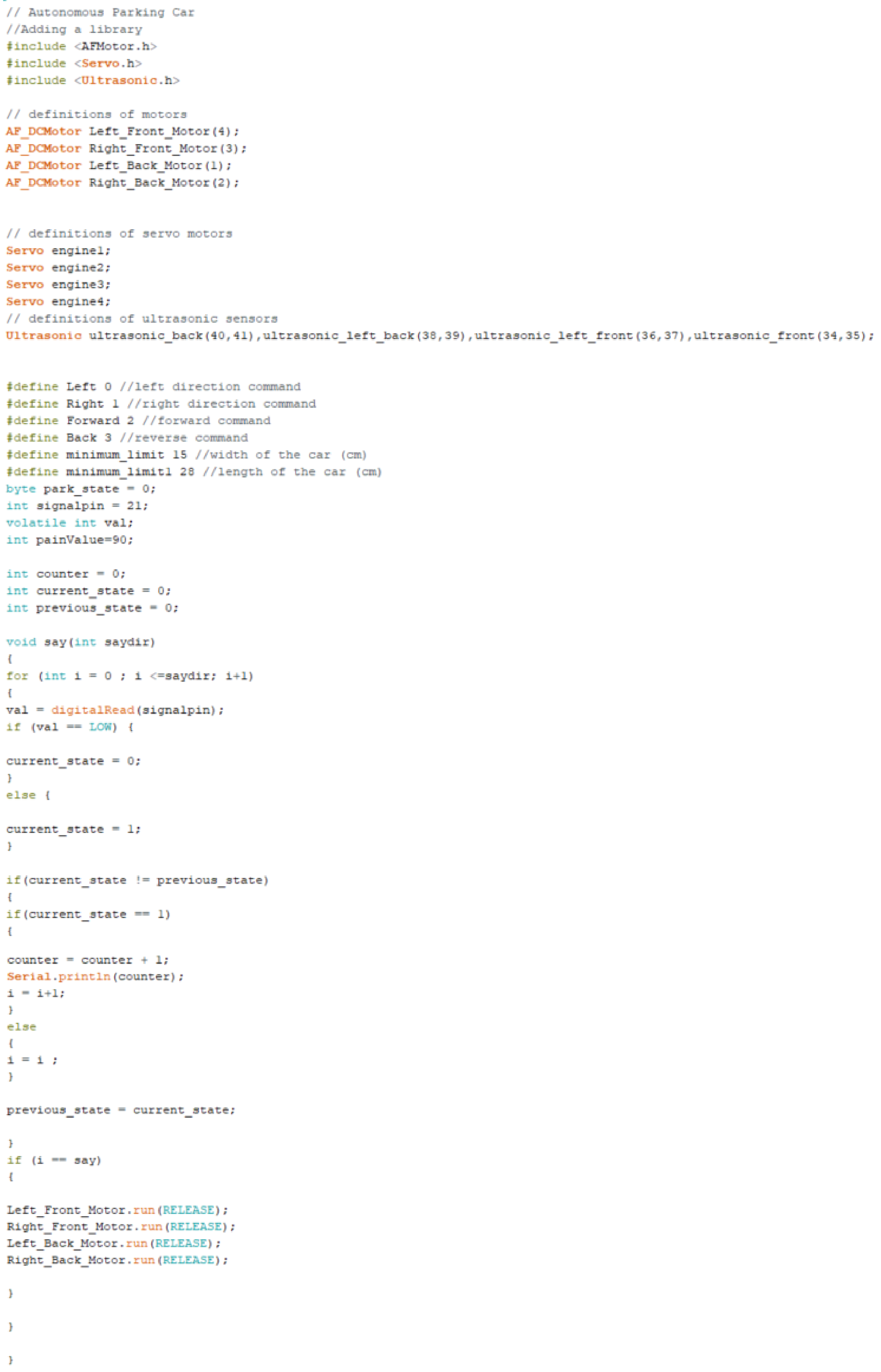

?
For example:
Receiver pulse duration - Time = 1.5ms
Speed of sound - Speed = 340m/s = 34cm/s
Distance = (Speed x Time) / 2
Distance = (34cm/ms x 1.5ms) / 2 = 25.5cm

(Figure 2.5 HCSR-04 Distance Measurement)

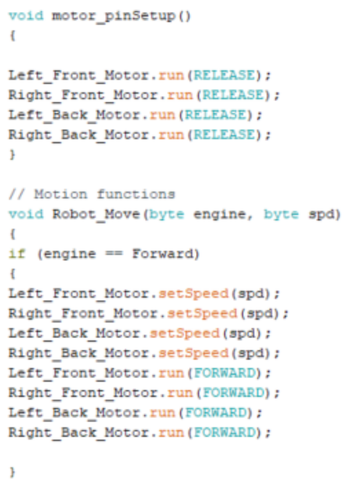
2.2.1 Project Scheme (Flow Chart)


Introduction: Autonomous parking vehicle is designed to eliminate the parking problem .
Calibration of sensors: The sensors used in the autonomous parking system are calibrated. This helps the system to accurately detect objects around it.
Initial position determination: The autonomous parking vehicle determines its initial position and continues its journey over this position.
Road scanning: The autonomous parking vehicle scans the surrounding road and determines the presence of a suitable parking space.
Are there any obstacles?
YES
The vehicle runs to find the next available parking space
NO
Parking space selection: The autonomous parking vehicle chooses the most suitable parking space and is directed towards this area.
Start the servo motors and reduce the size of the vehicle.
NO
Parking space selection: Did the autonomous vehicle find a suitable parking space?
YES
Parking process: The autonomous parking vehicle completes the parking process automatically after entering the selected parking area.
Exit: The autonomous parking vehicle stops the engines after completing the parking process.

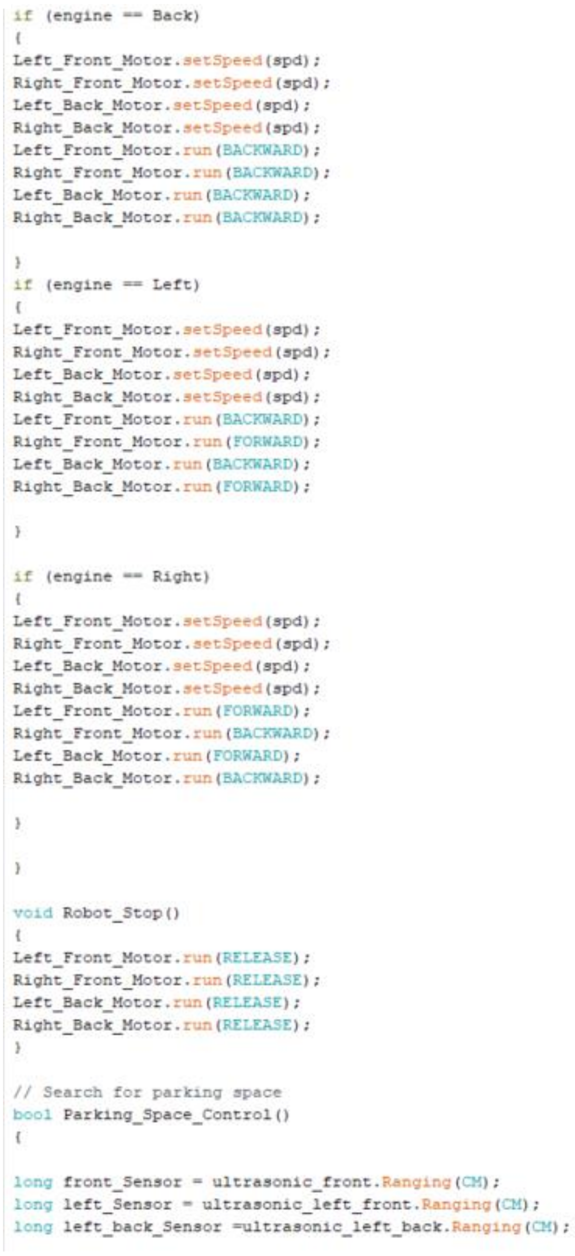
2.2.2 Project Code



```
// Autonomous Parking Car
//Adding a library
#include <AFMotor.h>
#include <Servo.h>
#include <Ultrasonic.h>

// definitions of motors
AF_DCMotor Left_Front_Motor(4);
AF_DCMotor Right_Front_Motor(3);
AF_DCMotor Left_Back_Motor(1);
AF_DCMotor Right_Back_Motor(2);


// definitions of servo motors
Servo engine1;
Servo engine2;
Servo engine3;
Servo engine4;
// definitions of ultrasonic sensors
Ultrasonic ultrasonic_back(40,41),ultrasonic_left_back(38,39),ultrasonic_left_front(36,37),ultrasonic_front(34,35);


#define Left 0 //left direction command
#define Right 1 //right direction command
#define Forward 2 //forward command
#define Back 3 //reverse command
#define minimum_limit 15 //width of the car (cm)
#define minimum_limit1 28 //length of the car (cm)
byte park_state = 0;
int signalpin = 21;
volatile int val;
int painValue=90;

int counter = 0;
int current_state = 0;
int previous_state = 0;

void say(int saydir)
{
for (int i = 0 ; i <=saydir; i+1)
{
val = digitalRead(signalpin);
if (val == LOW) {

current_state = 0;
}
else {

current_state = 1;
}

if(current_state != previous_state)
{
if(current_state == 1)
{

counter = counter + 1;
Serial.println(counter);
i = i+1;
}
else
{
i = i ;
}

previous_state = current_state;

}
if (i == say)
{

Left_Front_Motor.run(RELEASE);
Right_Front_Motor.run(RELEASE);
Left_Back_Motor.run(RELEASE);
Right_Back_Motor.run(RELEASE);

}

}

}
```



```
void motor_pinSetup()
{

Left_Front_Motor.run(RELEASE);
Right_Front_Motor.run(RELEASE);
Left_Back_Motor.run(RELEASE);
Right_Back_Motor.run(RELEASE);
}

// Motion functions
void Robot_Move(byte engine, byte spd)
{
if (engine == Forward)
{
Left_Front_Motor.setSpeed(spd);
Right_Front_Motor.setSpeed(spd);
Left_Back_Motor.setSpeed(spd);
Right_Back_Motor.setSpeed(spd);
Left_Front_Motor.run(FORWARD);
Right_Front_Motor.run(FORWARD);
Left_Back_Motor.run(FORWARD);
Right_Back_Motor.run(FORWARD);

}
```



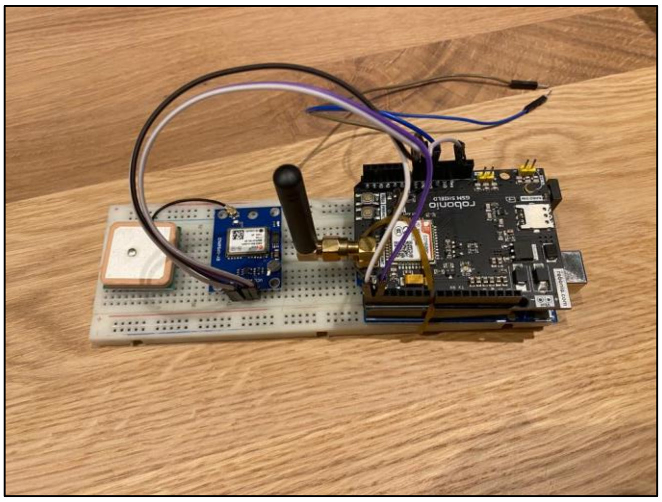
```
if (engine == Back)
{
Left_Front_Motor.setSpeed(spd);
Right_Front_Motor.setSpeed(spd);
Left_Back_Motor.setSpeed(spd);
Right_Back_Motor.setSpeed(spd);
Left_Front_Motor.run(BACKWARD);
Right_Front_Motor.run(BACKWARD);
Left_Back_Motor.run(BACKWARD);
Right_Back_Motor.run(BACKWARD);

}
if (engine == Left)
{
Left_Front_Motor.setSpeed(spd);
Right_Front_Motor.setSpeed(spd);
Left_Back_Motor.setSpeed(spd);
Right_Back_Motor.setSpeed(spd);
Left_Front_Motor.run(BACKWARD);
Right_Front_Motor.run(FORWARD);
Left_Back_Motor.run(BACKWARD);
Right_Back_Motor.run(FORWARD);

}

if (engine == Right)
{
Left_Front_Motor.setSpeed(spd);
Right_Front_Motor.setSpeed(spd);
Left_Back_Motor.setSpeed(spd);
Right_Back_Motor.setSpeed(spd);
Left_Front_Motor.run(FORWARD);
Right_Front_Motor.run(BACKWARD);
Left_Back_Motor.run(FORWARD);
Right_Back_Motor.run(BACKWARD);

}

}

void Robot_Stop()
{
Left_Front_Motor.run(RELEASE);
Right_Front_Motor.run(RELEASE);
Left_Back_Motor.run(RELEASE);
Right_Back_Motor.run(RELEASE);
}

// Search for parking space
bool Parking_Space_Control()
{

long front_Sensor = ultrasonic_front.Ranging(CM);
long left_Sensor = ultrasonic_left_front.Ranging(CM);
long left_back_Sensor =ultrasonic_left_back.Ranging(CM);
```

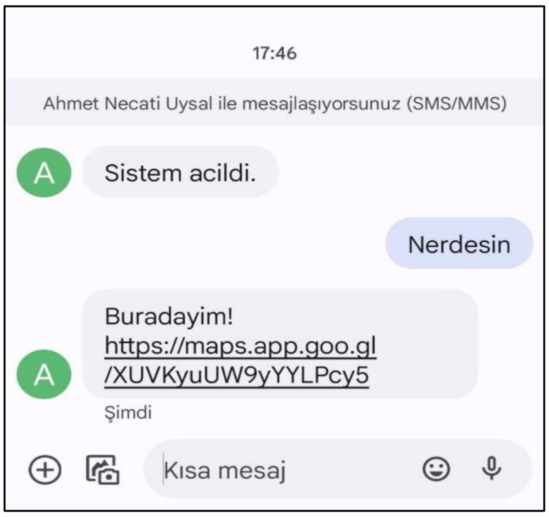
```
if( (left_Sensor <= minimum_limit)&&(left_back_Sensor <= minimum_limit)&&(park_state == 0))
{
Robot_Move(Forward, 100);
park_state = 1; Serial.println(park_state);
}

if((left_Sensor > minimum_limit)&&(left_Sensor < minimum_limit1)&&(left_back_Sensor > minimum_limit)&&(left_back_Sensor < minimum_limit1)&&(park_state == 1))
{
Robot_Move(Forward, 100);
park_state = 2;Serial.println(park_state);
}

if((left_Sensor >= minimum_limit1)&&(left_back_Sensor >= minimum_limit1)&&(park_state == 1))
{
/* Vertical Parking Decision */
Robot_Stop() ;
delay(500);
engine1.write(0);
engine2.write(0);
engine3.write(180);
engine4.write(180);
park_state = 3;Serial.println(park_state);
}

if((left_Sensor <= minimum_limit)&&(left_back_Sensor <= minimum_limit)&&(park_state == 2))
{
/* Parallel Parking Decision */
park_state = 3; Serial.println(park_state);
}

return park_state;
}

void Park_find()
{
Parking_Space_Control();
if(park_state == 3 )
{
Robot_Stop();Serial.println(park_state);
delay(400);
park_state = 4;
}
if(park_state == 4 )
{

Robot_Move(Back,120);
say(18);
Robot_Stop();Serial.println(park_state);
delay(500);
Robot_Move(Right,150);
say(9);
Robot_Stop();
delay(500);
park_state = 5;
}
if(park_state == 5)
{

Robot_Move(Back,120);
long back_Sensor = ultrasonic_back.Ranging(CM);Serial.println(back_Sensor);

if(back_Sensor>0 && back_Sensor <= 13)
{
Robot_Stop();
delay(400);
park_state = 6;
}
return ultrasonic_back;
}

if(park_state == 6)
{
Robot_Move(Left,150);
long left_Sensor = ultrasonic_left_front.Ranging(CM); Serial.println(left_Sensor);
long left_Back_Sensor = ultrasonic_left_back.Ranging(CM); Serial.println(left_Back_Sensor);

if(left_Sensor == left_Back_Sensor)
{
Robot_Stop();
park_state = 7;
}

return left_Sensor,left_Back_Sensor;
}
if(park_state == 7)
{
long front_Sensor = ultrasonic_front.Ranging(CM);

if(front_Sensor<=6)
{
Robot_Stop();
park_state = 8;
}
else
{
Robot_Move(Forward,100);
}
return ultrasonic_front;
}
if (park_state ==10)
{
```

```
Robot_Move(Left,180);
say(14);
Robot_Stop();
delay(500);
park_state = 7;

}

}

void setup()
{
Serial.begin(9600);
attachInterrupt(5, say, CHANGE);
pinMode (signalpin, INPUT) ;
engine1.attach(26);
engine2.attach(28);
engine3.attach(30);
engine4.attach(32);
motor_pinSetup();
}

void loop()
{
Park_find();
}
```

2.2.4 Mapping With GSM Shield and Arduino UNO

```
#include <RobonioGSM.h>
RobonioGSM Robonio;
String smstext, smsNumber;
/* We define the number to which the address will be sent */
#define TargetNumber  "+90"
#define LED_pin 13

/*We add the GPS library*/
#define GPS Serial
#include <TinyGPS.h>
TinyGPS gps;

double enl;
double boy;
String location;
char location2[255];

void setup() {
  Robonio.basla();
  delay(2000);
  pinMode(LED_pin, OUTPUT);
  GPS.begin(9600);
  delay(2000);
  Robonio.tumSmsSil();
  delay(2000);
  Robonio.smsGonder(TargetNumber, "Sistem acildi.");
}

void loop()
/*Defining commands*/
{
  smstext = Robonio.smsOku(1);
  if (smstext.indexOf("OK") != -1) {
    if (smstext.length() > 7) {
      smsNumber = Robonio.numaraliSmsOku(1);
      smstext.toUpperCase();

      if (smstext.indexOf("LEDAC") != -1) {
        digitalWrite(LED_pin, HIGH);
        Robonio.smsGonder(TargetNumber, "LED Acildi");
      }
      else if (smstext.indexOf("LEDKAPAT") != -1) {
        digitalWrite(LED_pin, LOW);
        Robonio.smsGonder(TargetNumber, "LED Kapatildi");
      }
      else if (smstext.indexOf("NERDESIN") != -1 || smstext.indexOf("NEREDESIN") != -1 || smstext.indexOf("KONUM") != -1)
        delay(1000);

        bool yeniVeri = false;
        for (unsigned long start = millis(); millis() - start < 1000;) {
          while (GPS.available()) {
            char c = GPS.read();
            if (gps.encode(c))
              yeniVeri = true;
          }
        }
if (yeniVeri) {
  float flat, flon;
  unsigned long age;
  gps.f_get_position(&flat, &flon, &age);
  enl = flat;
  boy = flon;
  location = "www.google.com.tr/maps/place/" + String(enl, 6) + "," + String(boy, 6);
  location.toCharArray(location2, 100);
  delay(1000);
}
delay(1000);
        char location2[255];
        konum.toCharArray(konum2, 100);
        Robonio.smsGonder(TargetNumber, konum2);
      }
      else {
      }
      Robonio.tumSmsSil();
    }
  }
  }
```


(Figure 2.6 GPRS Module Connected to GSM Shield)



17:46

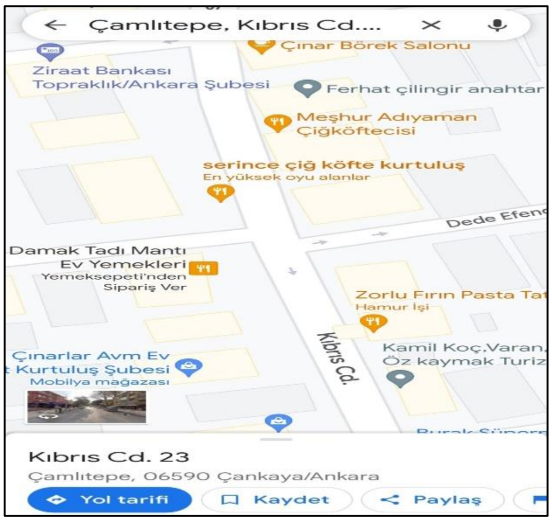
Ahmet Necati Uysal ile mesajlaşıyorsunuz (SMS/MMS)

A: Sistem acildi.

Nerdesin

A: Buradayim! https://maps.app.goo.gl/XUVKyuUW9yYYLPcy5

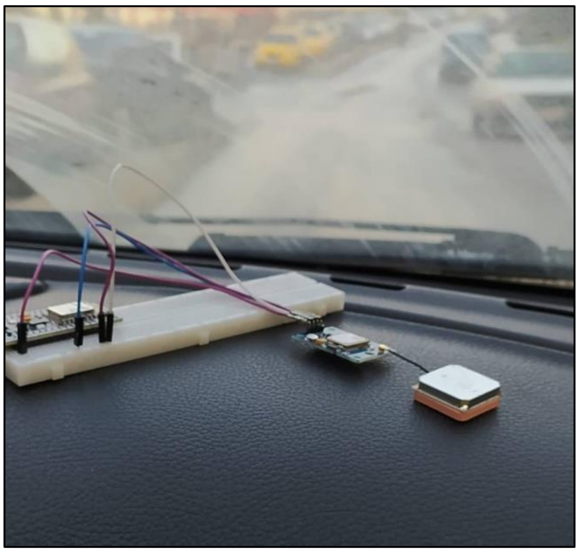
Şimdi

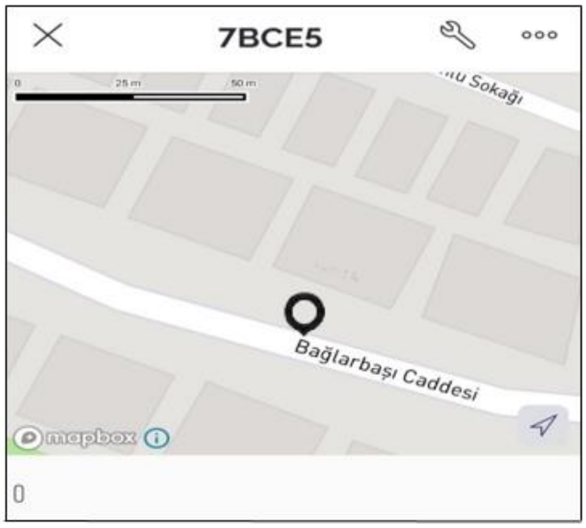


(Figure 2.7 Getting Instant Location as Message Using GSM Shield)

2.2.5 Mapping with Nodemcu and Bylnk App

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

static const int RXPin = 4, TXPin = 5;   // GPIO 4=D2(conneect Tx of GPS) and GPIO 5=D1(Connect Rx of GPS
static const uint32_t GPSBaud = 9600; //if Baud rate 9600 didn't work in your case then use 4800

TinyGPSPlus gps;      // The TinyGPS++ object
WidgetMap myMap(V0);  // V0 for virtual pin of Map Widget

SoftwareSerial mygps(RXPin, TXPin);  // The serial connection to the GPS device

BlynkTimer timer;

float latitude;     //Storing the Latitude
float longitude;    //Storing the Longitude
float velocity;     //Variable  to store the velocity
float sats;         //Variable to store no. of satellites response
String bearing;     //Variable to store orientation or direction of GPS

char auth[] = "********";            //Blynk Authentication Token
char ssid[] = "********";            // WiFi SSID
char pass[] = "********";            // WiFi Password

//unsigned int move_index;          // moving index, to be used later
unsigned int move_index = 1;        // fixed location for now
void checkGPS()
{
  if (gps.charsProcessed() < 10)
  {
    Serial.println(F("No GPS detected: check wiring."));
    Blynk.virtualWrite(V3, "GPS ERROR");  // Value Display widget  on V3 if GPS not detected
  }
}

void loop()
{
  while (mygps.available() > 0)
  {
    // sketch displays information every time a new sentence is correctly encoded.
    if (gps.encode(mygps.read()))
      displayInfo();
  }
  Blynk.run();
  timer.run();
}


void displayInfo()
{
  if (gps.location.isValid() )
  {
    sats = gps.satellites.value();       //get number of satellites
    latitude = (gps.location.lat());     //Storing the Lat. and Lon.
    longitude = (gps.location.lng());
    velocity = gps.speed.kmph();         //get velocity
    bearing = TinyGPSPlus::cardinal(gps.course.value());     // get the direction
    Serial.print("SATS:  ");
    Serial.println(sats);  // float to x decimal places
    Serial.print("LATITUDE:  ");
    Serial.println(latitude, 6);  // float to x decimal places
    Serial.print("LONGITUDE: ");
    Serial.println(longitude, 6);
    Serial.print("SPEED: ");
    Serial.print(velocity);
    Serial.println("kmph");
    Serial.print("DIRECTION: ");
    Serial.println(bearing);

    Blynk.virtualWrite(V1, String(latitude, 6));
    Blynk.virtualWrite(V2, String(longitude, 6));
    Blynk.virtualWrite(V3, sats);
    Blynk.virtualWrite(V4, velocity);
    Blynk.virtualWrite(V5, bearing);
    myMap.location(move_index, latitude, longitude, "GPS_Location");
  }
  Serial.println();
}
```


(Figure 2.8 GPRS Module Connected to Nodemcu)



(Fig 2.9 Seeing Instant Location with BLYNK application)

3. APPLICATION RESULTS

A general block diagram of the study is shown in title 2.2.1. The most important difference between this study and similar studies is that if the vehicle cannot find a suitable parking space, we adapt to the parking area by folding and we can follow the location of our vehicle on the map. In order to carry out the study, microcontroller board, motor driver board, four ultrasonic sensors, 4 servo motors, Nodemcu ESP8266, GPS and battery were mounted in the middle of the vehicle. Based on the information from a sensor, an instant distance measurement was made and a suitable parking place was determined. The location of the vehicle can be tracked in real time via the mobile application with the Nodemcu ESP8266 card and via the GSM shield as a message.

4. CONCLUSION

In this study, a prototype has been developed that enables vehicles such as automobiles to automatically find suitable parking spaces and park autonomously. With the developed hardware and software, the prototype detects the parking space and, with the help of sensors on the vehicle, automatically parks the vehicle without the need for driver assistance when there is enough space, and can fold and reduce it. The autonomous parking part is divided into two: finding a suitable parking space and autonomous parking. Different scenarios were tested in the application studies and the prototype developed for these scenarios was successful. In future studies, it is planned to find automatic parking space based on image processing on the map and autonomous parking in the marked area.

5. REFERENCES

[1] Zengin E., "Otomatik park eden araba", Yıldız Teknik Üniversitesi, Mekatronik Mühendisliği Bölümü, Lisans Bitirme Tezi, 2013

[2] Şanlı, Erdem**.** Yapay Sinir Ağı Kontrollü Otonom RC Araç Uygulaması, İstanbul Gelişim Üniversitesi, 2018.

[3] Bektaş, Özgür. Ardunio Uno RC Car, İstanbul Üniversitesi, 2015.

[4] Robot Sitem. ‘’DC Motorlar’’. http://www.robotiksistem.com/dc_motor_ozellikleri.html

[5] IEEExplore, ‘’A sensor guided autonomous parking system for nonholonomic mobile robots’’**.** https://ieeexplore.ieee.org/abstract/document/769997

[6] IEEExplore, ‘’GATA: GPS-Arduino based Tracking and Alarm system for protection of wildlife animals. https://ieeexplore.ieee.org/abstract/document/8035325

[7] Khin, M.M.J., “Real-Time Vehicle Tracking System Using Arduino, GPS, GSM and Web-Based Technologies”, 2018

[8] Bühler O, Wegener J.,” Automatic Testing of an Autonomous Parking System using Evolutionary Computation”, 2004

[9] Demirli, K., & Khoshnejad, M., “Autonomous parallel parking of a car-like mobile robot by a neuro-fuzzy sensor-based controller”, 2009