ME 417: Mechatronics

Project Report

Self-Parking Car

Submitted to: Assistant Professor Barkan UĞURLU

Submitted by: Nezih Sunman - Ayhan Alp Aydeniz

Student ID: S009292 - S009740

Date: 06.05.2019

ABSTRACT

This report considers a course project conducted under the course of ME 417 (Mechatronics). In this project, we have built a self parking mobile robot which have the ability to park through several parking techniques. The robot can be controlled via an Android App using WIFI. To control the robot, the hardware consists of an Arduino Mega, an ESP8266, sensors, and DC motors. The whole system is designed to simulate real scenarios, for example, a driver drives a car by himself/herself, but the parking is achieved autonomously. The robot has been tested in indoor environment. Hereinafter, the robot is referred as self-parking car.

1. INTRODUCTION

Cars started to be used globally in the world in the 20th century under the brand name of Benz and Ford [1]. With the developments in the world economy, it has become a product for the upper segment to the lower segment. With the technological developments in recent years, cars became a technology center with the first electric cars in 2008 [2]. Firstly, computer systems were added to cars and these systems were called electronic control units [2]. It became a system that controlled all the controls of a car from the brake system to the gassing system. That's why some technology companies have developed a number of self-driving lane tracking vehicles in cars.

The project which is tried to be done in the mechatronics course is an exemplary system. The purpose of this project was to integrate the state machine logic into a vehicle parking system and how to use this system more easily and with less sensors.

The logic of the State Machine is based on a geometrical movement. In line with this movement logic, the engine systems provide a complete return and driving force. Basically the parking logic is based on parking in a certain area, while the project helps the user to find any docking or parking space in the system. In this way, without the use of any camera system, only the sensors, the parking space, the distance and the gap is measuring. In this way, the logic of the electric system can be parked or not found.

2. CHALLENGES

The basic method of designing the self-parking car system is to determine the driving system of a vehicle and determine which control system will be more suitable. As a result of these determinations, the engine and plastic case without any encoder system will be used. The necessity to choose these systems is to limit our budget and to increase the budget for a wide range of applications. Therefore, we do not know the acceleration speed and positional values while driving. As a result of these limitations, it is necessary to get a continuous change of information from the sensor systems to get feedback on where and how the vehicle behaves, and to decide where the vehicle itself is in line with this information.

Another challenging condition is that when the design of the vehicle requires a plastic casing and the plexi material is very susceptible to flex, the 4 yellow motors give unusual reactions in every second and driving and motors behave extraordinary from all the parameters of the coded system. First of all, there is a need for a ground test that can get an average value for a general ground where the vehicle will go. To read the values that will change during this test, we need a reliable speed, angle, and accurate distance measurement tools to move the certain ratio of vehicle in each step of Arduino mega void loop.

Another important step is how to implement the parking strategy. There are many logical and sensory solution methods in the parking process. One of the methods that can be considered is that a computer vision system can be installed and the system can be parked on the way to remote scan and decide on the park. The second example is that the vehicle is constantly moving and moving to the situation and the result with a logic of the state machine by means of trial and error. The basic convenience of this orientation is the use of sensor systems in mechatronics. any additional computer, such as a computer vision system, does not require an additionally image processing and communication system. In this way, real time data can be interpreted instantaneously and the system to be designed in a specific and optimal budget.

Another compelling factor is the constant reduction and change of the input voltage of the motors. As it is known from data sheets, the main reason for this change is the continuous and excessive current of the motors and the ESP8266

system. The parallel connection of the 1.5 Volt Big Batteries used in these systems in addition to all previously mentioned factors adds an uncertainty and condition to the system. This problem can be solved easily by using the Li-Po battery, which is based on long lasting and high current. But since this is a requirement above our budget, the last decision was made after trying all the necessary means.

3. SYSTEM ARCHITECTURE

The system is built to simulate different cases in which a driver drives manually; however, the parking is achieved autonomously. To implement the algorithms an Arduino Mega is used, because it enables a suitable environment to program the interaction between different parts of the hardware.

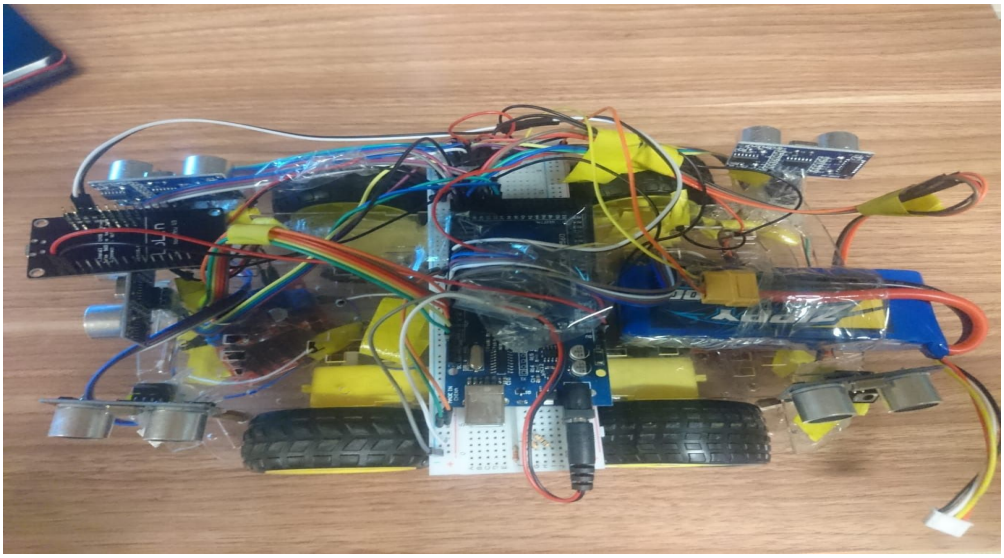

Figure 1 - Self-parking car

3.1 Hardware Design

To implement the whole system, we have used an Arduino Mega, an ESP8266, 4 DC motors, 2 motor drivers, and 6 ultrasonic sensors.

Arduino Mega: Arduino is used to combine the hardware and the software parts of self-parking car. We have connected the all branches through the Arduino. Each sensor and each motor connected to the Arduino pins and their relations are defined in Arduino Integrated Development Environment (IDE).

ESP8266: To transmit the commands to the self-parking car, we have used ESP8266. The required commands are entered via an Android App and sent from a mobile phone to the self-parking car over WIFI. ESP8266 enables the communication between microcontroller and the mobile phone through behaving as a WIFI Hotspot.

LM298N (Motor Driver): The motor drivers are used to amplify the current from the Arduino to the DC motors. We have used 2 LM298N motor drivers to drive 4 DC motors separately. LM298N drivers have the ability to drive 2 separate DC motors and we have used 3 different pins as variables in the Arduino through its in1, in2, en pins of it per DC motor.

HC-SR04 (Ultrasonic Sensor): These ultrasonic sensors uses ultrasonic sound waves to measure the distance between the sensors and obstacles. These sensors have four pins used to interface with the sensors, which are Vcc (Voltage), Trig (signal output pin), Echo (signal input pin), and GND (Ground). Vcc is connected to 5V, Trig and Echo to digital pins of the Arduino, GND to pin for GND.

6V 250 RPM DC Motor: These motors enables the mobility of the self-parking car. These motor are connected to the Arduino through LM298N motor drivers.

3.2 Algorithm Design

We have implemented algorithm to define the parking techniques. The self-parking car have the ability to park with 6 different types of parking. These types differ from each other with maneuvering. We have used FSMs to implement the algorithms of each type of parking. The codes for the algorithms implemented on Arduino.

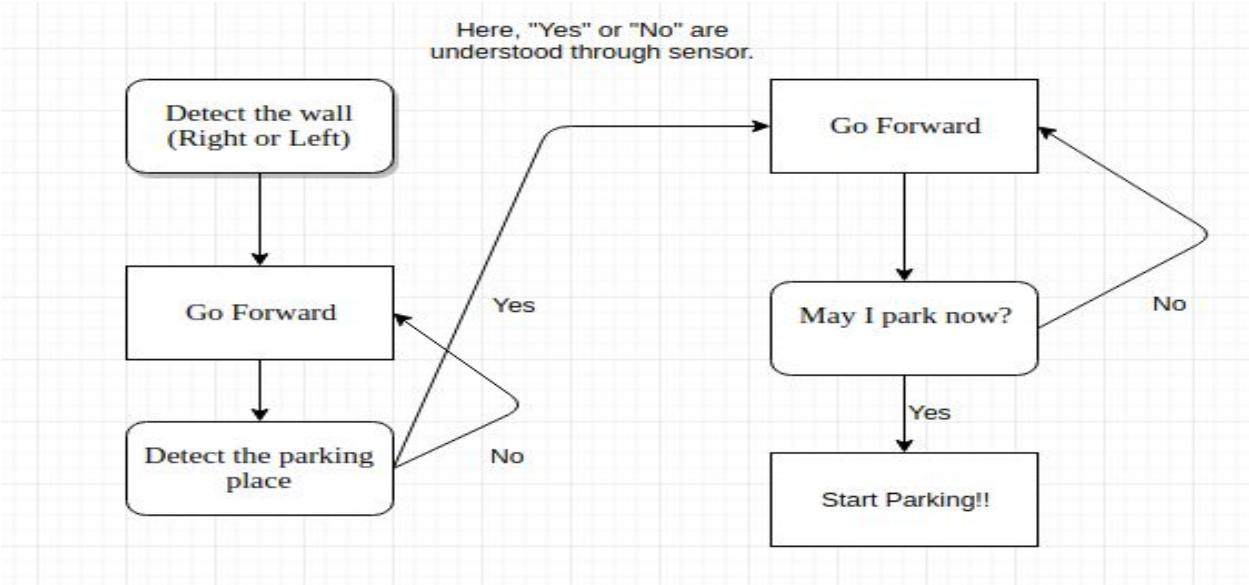

Figure 2 - General FSM logic of self-parking

3.3 Manipulation

The manipulation of the self-parking car is done via WIFI. ESP8266 has been used to enable WIFI control, then I2C (Inter-integrated Circuit) is used as the protocol for the communication between ESP8266 and the Arduino. The code written in Appendix A clearly shows that ESP8266WiFi, WiFiClient, ESP8266WebServer libraries were used. The commands are sent via an Android App.

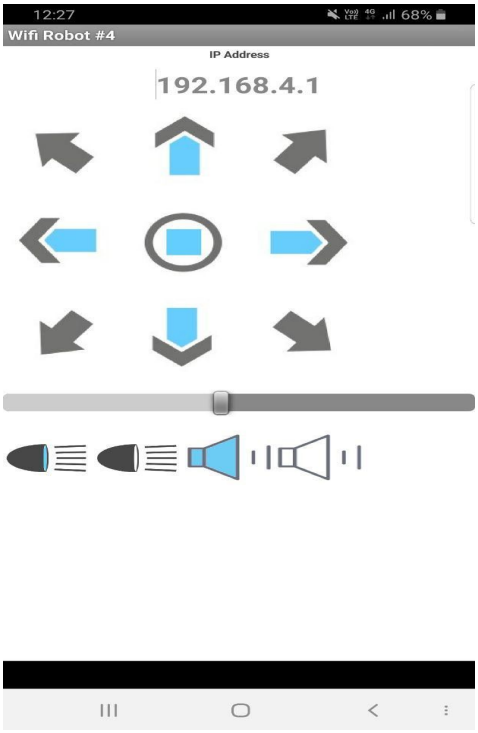

Figure 1 - The Android App through which remote control is done

The mobile application above found through the work done in [3].

4. DEMONSTRATION AND DISCUSSION

We have driven the self-parking car for many times as demonstration and we have observed that the robot is stabilized, when the robot is controlled via WIFI.

The self-parking car is tested in indoor environment on different floors. Because the self-parking car does not use a PID controller, we need to tune the robot, when we changed the floor. However, as a future work, we can use sensors to get feedback from the wheels, so that we would be able to use a PID controller.

5. CONCLUSIONS

The Self-Parking car system is basically an algorithmic structure with a state machine logic. A lot of add-ons and enhancements can be made so that the basic logic of this structure remains the same. These additions can be systems like machine learning algor. With the help of these systems, we can make thousands of

basic tests in a very short time. In these tests, the data can be analyzed autonomously and the data can be analyzed in the database system and analyzed retrospectively. In this way, errors caused by the system itself and the errors due to the algorithm can be different. Experiments can be carried out with modified codes and a more advanced system can be obtained.

The algorithm is simple as can be seen in Appendix B. This can be done using more states of the algorithm, but the result is still the same. Because, although there are many different situations, the absence of any conditions or emergency stops between those situations simplifies the system. Written codes and the margins of error from the electrical system were reduced by "IF", which is the control statements within the system itself. To summarize briefly, the written system algorithm works properly and controllable.

References

[1]

- URL:http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvQ2Fy
- Website Title: Wikipedia
- Article Title: road vehicle powered by a motor to carry driver and small number of passengers
- Date Published: June 02, 2019
- Date Accessed: June 03, 2019

[2]

- URL: https://www.popularmechanics.com/cars/how-to/a7386/how-it-works-the-computer-inside-your-car/
- Website Title: Popular Mechanics
- Article Title: How it Works: The Computer Inside Your Car
- Date Published: February 15, 2018
- Date Accessed: June 03, 2019

[3] URL:

https://create.arduino.cc/projecthub/andriy-baranov/from-bt-to-wifi-creating-wifi-controlled-arduino-robot-car-09b7c1

- Website Title: Arduino Project Hub
- Article Title: From BT To WiFi: Creating WiFi Controlled Arduino Robot Car
- Date Accessed: June 03, 2019

Appendix A

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
const char* host = "WiFi_Robot4_NodeMCU";
const char* ssid = "nezih";
//Connection Name WithOut Password
ESP8266WebServer server(80);
void setup() {
  Serial.begin(115200);
  // Connecting WiFi
  WiFi.mode(WIFI_AP);
  WiFi.softAP(ssid);
// Starting WEB-server
     server.on ( "/", HTTP_handleRoot );
     server.onNotFound ( HTTP_handleRoot );
     server.begin();
     }
void loop() {
  server.handleClient();
   delay(50);
}
void HTTP_handleRoot(void) {
if( server.hasArg("State") ){
       Serial.println(server.arg("State"));
  }
  server.send ( 200, "text/html", "" );
}
```

Appendix B

```
int count = 0;
int speeds = 85;
int speedl = 72;
int d = 100;
int yanascount = 0;
int distancevalue = 19;
int command;

bool parking = false;
bool yanas = false;
bool onpark = false;
bool leftpark = false;

// connect motor controller pins to Arduino digital pins
// motor Sol Ön
int enA = 8;
int in1 = 9;
int in2 = 10;

// motor Sol Arka
int enB = 13;
int in3 = 11;
int in4 = 12;

// motor Sağ Ön
int enB2 = 2;
int in32 = 4;
int in42 = 3;

// motor Sağ Arka
int enA2 = 7;
int in12 = 6;
int in22 = 5;

//Sol Ön sensör
const int trigPin_sol_on = 31; // Pin numaraları
const int echoPin_sol_on = 30; // Pin numaraları

long duration_sol_on; //variables
int distance_sol_on; //variables


//Sağ Arka sensör
const int trigPin_sag_arka =  25; // Pin numaraları
const int echoPin_sag_arka = 24; // Pin numaraları
```

```
long duration_sag_arka; //variables
int distance_sag_arka; //variables

//Sağ Ön sensör
const int trigPin_sag_on =  23; // Pin numaraları
const int echoPin_sag_on = 22; // Pin numaraları

long duration_sag_on; //variables
int distance_sag_on; //variables


//Sol Arka sensör
const int trigPin_sol_arka =  27; // Pin numaraları
const int echoPin_sol_arka = 26; // Pin numaraları

long duration_sol_arka; //variables
int distance_sol_arka; //variables

//Ön sensör
const int trigPin_on =  33; // Pin numaraları
const int echoPin_on = 32; // Pin numaraları

long duration_on; //variables
int distance_on; //variables

void setup()
{
  // set all the motor control pins to outputs
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);

  pinMode(enA2, OUTPUT);
  pinMode(enB2, OUTPUT);
  pinMode(in12, OUTPUT);
  pinMode(in22, OUTPUT);
  pinMode(in32, OUTPUT);
  pinMode(in42, OUTPUT);

  //Sağ Ön  Sensor
  pinMode(trigPin_sag_on, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin_sag_on, INPUT); // Sets the echoPin as an Input

  //Sol Arka Sensor
  pinMode(trigPin_sol_arka, OUTPUT); // Sets the trigPin as an Output
```

```
  pinMode(echoPin_sol_arka, INPUT); // Sets the echoPin as an Input

  //Sağ Arka Sensor
  pinMode(trigPin_sag_arka, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin_sag_arka, INPUT); // Sets the echoPin as an Input

  //Sol Ön Sensor
  pinMode(trigPin_sol_on, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin_sol_on, INPUT); // Sets the echoPin as an Input

  //Ön Sensor
  pinMode(trigPin_on, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin_on, INPUT); // Sets the echoPin as an Input

  Serial.begin(115200);
}
```
//Receive data from sensor code
```
int sensorDistance(const int echoPin, const int trigPin, long duration, int distance) {
  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);

  // Calculating the distance
  distance = duration * 0.034 / 2;

  // Returns the distance on the Serial Monitor
  return distance;
}

void rotateRight() {
```
//Bu kısım için maksimum hız limiti 255
```

  //Sol ön motor saat yönünün tersi yönde döner
  Serial.print("arac saat yönğnde yonde dönüyor");
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  analogWrite(enA, speeds);

  //Sol arka motor saat yönünün tersi yönde döner
  digitalWrite(in3, HIGH);
```

```
  digitalWrite(in4, LOW);
  analogWrite(enB, speeds);

  //Sağ arka motor saat yönünde döner
  digitalWrite(in12, LOW);
  digitalWrite(in22, HIGH);
  analogWrite(enA2, speedl);

  //Sağ ön motor saat yönünde döner
  digitalWrite(in32, LOW);
  digitalWrite(in42, HIGH);
  analogWrite(enB2, speedl);

}

void rotateLeft() {
  //Bu kısım için maksimum hız limiti 255

  //Sol ön motor saat yönünün tersi yönde döner
  Serial.print("arac saat yönğnde yonde dönüyor");
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  analogWrite(enA, speeds);

  //Sol arka motor saat yönünün tersi yönde döner
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  analogWrite(enB, speeds);

  //Sağ arka motor saat yönünde döner
  digitalWrite(in12, HIGH);
  digitalWrite(in22, LOW);
  analogWrite(enA2, speedl);

  //Sağ ön motor saat yönünde döner
  digitalWrite(in32, HIGH);
  digitalWrite(in42, LOW);
  analogWrite(enB2, speedl);

}

void Return() {
  //Bu kısım için maksimum hız limiti 255
  //Araç geri geri gider
  Serial.print("arac geri yonde gidiyor");
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  analogWrite(enA, speeds);
```

```
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  analogWrite(enB, speeds);

  digitalWrite(in12, LOW);
  digitalWrite(in22, HIGH);
  analogWrite(enA2, speedl);

  digitalWrite(in32, LOW);
  digitalWrite(in42, HIGH);
  analogWrite(enB2, speedl);

}

void Drive() {
  //Bu kısım için maksimum hız limiti 255
  //Araç ileri yönde gider
  Serial.print("arac ileri yonde gidiyor");
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  analogWrite(enA, speeds);

  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  analogWrite(enB, speeds);

  digitalWrite(in12, HIGH);
  digitalWrite(in22, LOW);
  analogWrite(enA2, speedl);

  digitalWrite(in32, HIGH);
  digitalWrite(in42, LOW);
  analogWrite(enB2, speedl);
}
void Stop() {

  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  analogWrite(enA, speeds);

  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  analogWrite(enB, speeds);

  digitalWrite(in12, LOW);
  digitalWrite(in22, LOW);
  analogWrite(enA2, speedl);
```

```
  digitalWrite(in32, LOW);
  digitalWrite(in42, LOW);
  analogWrite(enB2, speedl);
  Serial.println("DURRRRRR");

}

void loop()
{


  if (Serial.available() > 0) {

    command = Serial.read();
   /* Serial.print("Command: ");
    Serial.println(command);
    Serial.println(parking);
    Serial.println(yanas);*/
    switch (command) {
      case 'F': Drive(); break;
      case 'B': Return(); break;
      case 'L': rotateLeft(); break;
      case 'R': rotateRight(); break;
      case 'S': Stop(); break;
      case 'V': parking = true; count = 0; break;
      case 'v': parking = false; break;
      case 'W': yanas = true; break;
      case 'w': yanas = false; break;
      case 'G': leftpark = true; break;
      case 'I': leftpark = false; break;
      case 'H': onpark = true; break;
      case 'J': onpark = false; break;
    }
  }

  if (onpark == true) {
    // ON PARK
    yanas = false;
    parking = false;
    distance_sag_on = sensorDistance(echoPin_sag_on, trigPin_sag_on, duration_sag_on, distance_sag_on);
    Serial.println("Sağ Ön Sensör Değeri: ");
    Serial.println(distance_sag_on);
    delay(4);

    distance_sol_arka = sensorDistance(echoPin_sol_arka, trigPin_sol_arka, duration_sol_arka, distance_sol_arka);
    Serial.println("Sol Arka Sensör Değeri: ");
    Serial.println(distance_sol_arka);
```

```
delay(4);

distance_sag_arka = sensorDistance(echoPin_sag_arka, trigPin_sag_arka, duration_sag_arka, distance_sag_arka);
Serial.println("Sağ Arka Sensör Değeri: ");
Serial.println(distance_sag_arka);
delay(4);

distance_sol_on = sensorDistance(echoPin_sol_on, trigPin_sol_on, duration_sol_on, distance_sol_on);
Serial.println("Sol Ön Sensör Değeri: ");
Serial.println(distance_sol_on);
delay(4);

distance_on = sensorDistance(echoPin_on, trigPin_on, duration_on, distance_on);
Serial.println("Ön Sensör Değeri: ");
Serial.println(distance_on);
delay(4);

if (distance_on < 3) {
  Stop();
}

else if (distance_sol_on < distancevalue && distance_sol_arka < distancevalue && distance_on > 10 && count
   == 0) {
  Drive();
  delay(d);
  count  = 0;
  Stop();
}

else if (distance_sol_on > distancevalue && distance_sol_arka < distancevalue && distance_on > 10 && (count
   == 0 || count == 1)) {
  Drive();
  delay(d);
  count = 1;
}
else if (distance_sol_on > distancevalue && distance_sol_arka > distancevalue && distance_on > 10 && (count
   == 1)) {

  Return();
  delay(100);
  count = 2;
  Stop();
  rotateLeft();
  delay(600);
}

else if ((distance_on > 5) && (count == 2 || count == 3)) {
  Drive();
```

```
  delay(d/3);
  Stop();
  rotateLeft();
  delay(d / 9);
  Stop();
  //Serial.println("CASE 4 - GERİ GİDİYOR");
  count = 3;
 }


 else if (distance_on <= 5 && distance_on >  3 && (count == 3)) {
  rotateRight();
  delay(720);
  Stop();
  //Serial.println("CASE 4 - GERİ GİDİYOR");
  count = 4;
 }
 else if (distance_on >  4 && (count == 4)) {
  Drive();
  delay(40);
  Stop();
  //Serial.println("CASE 4 - GERİ GİDİYOR");
  count = 4;
 }

 Stop();

}

if (parking == true && leftpark == true) {
  yanas = false;
  onpark = false;
  distance_sag_on = sensorDistance(echoPin_sag_on, trigPin_sag_on, duration_sag_on, distance_sag_on);
  Serial.println("Sağ Ön Sensör Değeri: ");
  Serial.println(distance_sag_on);
  delay(4);

  distance_sol_arka = sensorDistance(echoPin_sol_arka, trigPin_sol_arka, duration_sol_arka, distance_sol_arka);
  Serial.println("Sol Arka Sensör Değeri: ");
  Serial.println(distance_sol_arka);
  delay(4);

  distance_sag_arka = sensorDistance(echoPin_sag_arka, trigPin_sag_arka, duration_sag_arka, distance_sag_arka);
  Serial.println("Sağ Arka Sensör Değeri: ");
  Serial.println(distance_sag_arka);
  delay(4);

  distance_sol_on = sensorDistance(echoPin_sol_on, trigPin_sol_on, duration_sol_on, distance_sol_on);
```

```
Serial.println("Sol Ön Sensör Değeri: ");
Serial.println(distance_sol_on);
delay(4);

distance_on = sensorDistance(echoPin_on, trigPin_on, duration_on, distance_on);
Serial.println("Ön Sensör Değeri: ");
Serial.println(distance_on);
delay(4);

if (distance_on < 3
  ) {
 Stop();
}

else if (distance_sol_on < distancevalue && distance_sol_arka < distancevalue && distance_on > 10 && count
  == 0) {
 Drive();
 delay(d);
 count  = 0;
}

else if (distance_sol_on > distancevalue && distance_sol_arka < distancevalue && distance_on > 10 && (count
  == 0 || count == 1)) {
 Drive();
 delay(d);
 count = 1;
}
else if (distance_sol_on > distancevalue && distance_sol_arka > distancevalue && distance_on > 10 && (count
  == 1 || count == 2)) {

 Drive();
 delay(d);
 count = 2;
}

else if (distance_sol_on < distancevalue && distance_sol_arka > 14 && distance_on > distancevalue && (count
  == 2 || count == 3)) {
 Drive();
 delay(d);
 Stop();
 count = 3;
}

else if (distance_sol_arka < distancevalue && distance_sol_arka > 2 && (count == 3 || count == 4)) {
 Return();
 delay(d);
 Stop();
 //Serial.println("CASE 4 - GERİ GİDİYOR");
```

```
    count = 4;
  }

  else if (distance_sol_arka > distancevalue && distance_sol_on < distancevalue && distance_sol_on > 5 &&
     (count == 4 || count == 5)) {
    Return();
    delay(350);

    rotateRight();
    delay(600);

    Return();
    delay(500);

    rotateLeft();
    delay(670);

    Stop();
    count = 6;
    parking = false;
  }

  Stop();

}

if (parking == true && leftpark == false) { // SAG PARK
  yanas = false;
  distance_sag_on = sensorDistance(echoPin_sag_on, trigPin_sag_on, duration_sag_on, distance_sag_on);
  Serial.println("Sağ Ön Sensör Değeri: ");
  Serial.println(distance_sag_on);
  delay(4);

  distance_sol_arka = sensorDistance(echoPin_sol_arka, trigPin_sol_arka, duration_sol_arka, distance_sol_arka);
  Serial.println("Sol Arka Sensör Değeri: ");
  Serial.println(distance_sol_arka);
  delay(4);

  distance_sag_arka = sensorDistance(echoPin_sag_arka, trigPin_sag_arka, duration_sag_arka, distance_sag_arka);
  Serial.println("Sağ Arka Sensör Değeri: ");
  Serial.println(distance_sag_arka);
  delay(4);

  distance_sol_on = sensorDistance(echoPin_sol_on, trigPin_sol_on, duration_sol_on, distance_sol_on);
  Serial.println("Sol Ön Sensör Değeri: ");
  Serial.println(distance_sol_on);
  delay(4);
```

```
distance_on = sensorDistance(echoPin_on, trigPin_on, duration_on, distance_on);
Serial.println("Ön Sensör Değeri: ");
Serial.println(distance_on);
delay(4);

if (distance_on < 3
  ) {
 Stop();
}

else if (distance_sag_on < distancevalue && distance_sag_arka < distancevalue && distance_on > 10 && count
  == 0) {
 Drive();
 delay(d);
 count  = 0;
}

else if (distance_sag_on > distancevalue && distance_sag_arka < distancevalue && distance_on > 10 && (count
  == 0 || count == 1)) {
 Drive();
 delay(d);
 count = 1;
}
else if (distance_sag_on > distancevalue && distance_sag_arka > distancevalue && distance_on > 10 && (count
  == 1 || count == 2)) {

 Drive();
 delay(d);
 count = 2;
}

else if (distance_sag_on < distancevalue && distance_sag_arka > distancevalue && distance_on > 14 && (count
  == 2 || count == 3)) {
 Drive();
 delay(d);
 Stop();
 count = 3;
}

else if (distance_sag_arka < distancevalue && distance_sag_arka > 2 && (count == 3 || count == 4)) {
 Return();
 delay(d);
 Stop();
 //Serial.println("CASE 4 - GERİ GİDİYOR");
 count = 4;
}
```

```
  else if (distance_sag_arka > distancevalue && distance_sag_on < distancevalue && distance_sag_on > 5 &&
     (count == 4 || count == 5)) {
   Return();
   delay(350);

   rotateLeft();
   delay(650);

   Return();
   delay(500);

   rotateRight();
   delay(600);

   Stop();
   count = 6;
   parking = false;
  }

  Stop();
 }
 Serial.print ("YANAS: ");
 Serial.println (yanas);

 Serial.print ("LEFTPARK: ");
 Serial.println (yanas);
 if (yanas == true && leftpark == true) { // SOLA YANAS
   Serial.print ("MESAJ: ");
   Serial.println ("MEKANDAYIM");
   parking = false;

   distance_sag_on = sensorDistance(echoPin_sag_on, trigPin_sag_on, duration_sag_on, distance_sag_on);
   Serial.println("Sağ Ön Sensör Değeri: ");
   Serial.println(distance_sag_on);
   delay(4);

   distance_sol_arka = sensorDistance(echoPin_sol_arka, trigPin_sol_arka, duration_sol_arka, distance_sol_arka);
   Serial.println("Sol Arka Sensör Değeri: ");
   Serial.println(distance_sol_arka);
   delay(4);

   distance_sag_arka = sensorDistance(echoPin_sag_arka, trigPin_sag_arka, duration_sag_arka, distance_sag_arka);
   Serial.println("Sağ Arka Sensör Değeri: ");
   Serial.println(distance_sag_arka);
   delay(4);

   distance_sol_on = sensorDistance(echoPin_sol_on, trigPin_sol_on, duration_sol_on, distance_sol_on);
   Serial.println("Sol Ön Sensör Değeri: ");
```

```
  Serial.println(distance_sol_on);
  delay(4);

  distance_on = sensorDistance(echoPin_on, trigPin_on, duration_on, distance_on);
  Serial.println("Ön Sensör Değeri: ");
  Serial.println(distance_on);
  delay(4);


  //ilk şart eğer araba açıktaysa ve sol tarafında park edebileceği bir yer var mı diye bakıyor
  if (distance_sol_on > distancevalue &&  distance_sol_arka > distancevalue && distance_on > 10 &&
     (yanascount == 0 || yanascount == 1)) {
   // araba duvara doğru yaklaşıyor mesefe 20 a düşene kadar
   rotateLeft();
   delay(170);
   Stop();
   Drive();
   delay(200);
   rotateRight();
   delay(160);
   Stop();
   yanascount = 1;
  }

  else if ( (distance_sol_on + 3 < distance_sol_arka) && (yanascount == 1 || yanascount == 2)) {
   rotateRight();
   delay(10);
   Stop();
   yanascount = 2;
  }
  else if ( (distance_sol_on > distance_sol_arka + 3) && (yanascount == 1 || yanascount == 2)) {
   rotateLeft();
   delay(10);
   Stop();
   yanascount = 2;

  }

  else if ((distance_sol_on - distance_sol_arka <3) && (distance_sol_arka - distance_sol_on  < 3)  && (yanascount
     == 1 || yanascount == 2)) {
   yanascount = 3;
   yanas = false;

  }
 }

}
```