Autonomous Parallel Parking Using Arduino

DESIGN PROJECT REPORT
submitted by

Abhishek Mohan (Reg No: MAC18EE005)
to

the APJ Abdul Kalam Technological University
in partial fulfilment of the requirements for the award of the degree

of

Bachelor of Technology
In
Electrical and Electronics Engineering

Department of Electrical and Electronics Engineering

Mar Athanasius College of Engineering
Kothamangalam,Kerala,India 686666
DECEMBER 2020

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
MAR ATHANASIUS COLLEGE OF ENGINEERING
KOTHAMANGALAM

CERTIFICATE

This is to certify that the design project report entitled **Autonomous Parallel Parking Using Arduino** submitted by **Abhishek Mohan (Reg No: MAC18EE005)** to the APJ Abdul Kalam Technological University in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Electrical & Electronics Engineering is a bonafide record of the design project carried out by them under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Prof. Ninu Joy	Prof. Geethu James	Prof. Beena M Varghese
Project Coordinator	Project Coordinator	Head of the Dept.

Dept.seal

06-12-2020

Acknowledgement

It is a great pleasure to acknowledge all those who have assisted and supported me for successfully completing this project.

First of all, We thank God Almighty for his blessings as it is only through his grace that I was able to complete this project successfully.

I take this opportunity to extend my sincere thanks to the project coordinator and faculty advisor Prof. Ninu Joy, Assistant Professor, Department of Electrical & Electronics Engineering for her constant support and immense contribution for the success of this project.

I also extend my sincere thanks to Prof. Geethu James, Assistant Professor, Electrical & Electronics Engineering Department and all other members of the Department of Electrical & Electronics Engineering for sharing their valuable comments during the preparation of this project.

I am also grateful to Prof.Beena M Varghese, Head of Electrical & Electronics Engineering Department, for the valuable guidance as well as timely advice which helped a lot during the preparation of the project.

I am deeply indebted to Dr. Mathew K, Principal, Mar Athanasius College of Engineering for his encouragement and support.

I whole - heartedly thank all my classmates, for their valuable suggestions and for the spirit of healthy competition that existed between us.

Abstract

Parallel parking can be challenging for many drivers. Many modern day cars do not have any systems in place to make parking a lot easier.And the vehicles that has autonomous parking facilities are not affordable for common people.So, the purpose of this project is to design a simple prototype parking system that can perform parallel parking manoeuvres without much input from the driver . The system can be configured to be compatible with any commercially available vehicle at an affordable price.The system works with the help of a series of sensors and is monitored and controlled by an arduino micro-controller. The system is currently implemented in a scaled down model of a vehicle but can be implemented with a few modifications onto a real one. This project explores how sensor inputs, algorithms can be used for practical applications. By further improving the algorithms the program loaded in the arduino can be updated to include other kinds of parking like angle parking, perpendicular parking etc.... without changing the system components.

Table Of Contents

List of Figures

Page

Chapter 1

Introduction

Parallel parking can be challenging for drivers, including myself. The typical modern day car does not contain any systems in place to make parking easier. The main goal of this project is to design a simple prototype parking system that can perform parallel parking manoeuvres autonomously. This system would include a series of proximity sensors as well as a central microprocessor that controls the car. This system would ideally work on both a scale model of a car as well as a life-sized car. This project explores how sensor input and an algorithm can be used for practical applications. This paper covers the various components used to create the parking system, including an Arduino microprocessor,ultrasonic and infra-red sensors, H-bridges, and servo motors. It also includes the parking algorithm implemented within the system.

As the population increased in the cities, the usage of vehicles got increased. It causes problem for parking which leads to traffic congestion, driver frustration, and air pollution. When we visit the various public places like shopping malls, multiplex cinema hall and hotels during the festival time or weekends it creates more parking problem. Recent research shows that a driver takes nearly 8 minutes to park his vehicle because he/she spends more time searching for a parking space which the person feels confident enough to park. The searching leads to 30 to 40 percent to traffic congestion. Here we going to see how to reduce the parking problem automatic car parking using offerings are transforming cities by improving infrastructure, creating more efficient and cost effective municipal services, enhancing public transportation, reducing traffic congestion, and keeping citizens safe and more engaged in the community. Car parking is an issue of significance both at the local and at the strategic level of planning. This project's main purpose is to produce a real life solution to the car parking problem which the whole world is facing frequently. People usually roam around in the parking lots trying to find a suitable place to park in to solve that problem we have created the automatic car parking system, using an open source hardware, programmable sensors and the use of computers to provide an interface to understand the digital output produced.

Chapter 2

Hardware Required

Arduino is basically an open-source computer hardware/software platform for building digital devices and interactive objects that can sense and control the physical world around them. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments. It comes with an open supply hardware feature that permits users to develop their own kit. The software of the Arduino is well-suited to all kinds of operation systems like Linux, Windows, and Macintosh, etc. It also comes with open supply software system feature that permits tough software system developers to use the Arduino code to merge with the prevailing programming language libraries and may be extended and changed. For beginners, it is very simple to use and also cheap. It can be used to create such devices that can interact with the environment using sensors and actuators. Some common examples include robot, thermostats and motion detectors. This paper introduces the use of Arduino in one such area, that is, Automatic car parking, so that people can become aware of where free parking place is available and save time, while avoiding traffic congestion

2.1 Ultrasonic sensors

Figure 2.1: Ultrasonic sensor

The Arduino Ultrasonic Range Detection Sensor is used with Arduino in order to calculate distances from objects. So, if we start with the Arduino Ultrasonic Range Detection Sensor, it's an IC that works by sending an ultrasound pulse at around 40 KHz. It then waits and listens for the pulse to echo back, calculating the time taken in microseconds (1 microsecond = 1.0 × 10-6 seconds). We can trigger a pulse as fast as 20 times a second and it can determine objects up to 10 meters away and as near as 4cm.It needs a 5Vpower supply to run. And then it waits and listens for the pulse to echo back,

by calculating the time taken in microseconds. Adding the Arduino Ultrasonic Range Detection Sensor to the Arduino is very easy, only 4 pins to worry about. Power, Ground, Trigger and Echo. Since it needs 5V and Arduino provides 5V we are obviously going to use this to power it.

2.2 Arduino Microcontroller

Figure 2.2: Arduino Uno

Figure 2.3: Arduino mega 2560R3

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 6 analog inputs, 14 digital input/output pins (of which 6 can be used as PWM outputs), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; all we have to do is simply connect in to PC with a USB cable or power it with an AC-to-DC adapter or battery to get started. The Uno board is the first in a series of USB Arduino board and the reference model for the Arduino platform. Although we had used Arduino Uno for our project, due to lesser number of pins it provides, Arduino Mega is preferred as it has much more number of pins.

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila. For our prototype we had used Arduino Uno, for real world application, Arduino MEGA is preferred since it can recieve data from more sensors and has much more processing power.

2.3 The Arduino motor Shield

Figure 2.4: Arduino motor shield

The Arduino motor Shield is a device that lets you control various loads that a typical Arduino pin cannot drive. The motor shield has quite a few features such as current measuring and the ability to drive a single stepper motor. At the heart of this shield is the L298P dual full bridge driver that can handle up to 3 amps for very short durations or 2 amps continuously per channel. Thorough example code is available for all the sections in the attached zipped folder.

2.4 Servo motor

Figure 2.5: Servo motor

The servo motor is an integral part of the final model because it allows the car to have the same front-wheel steering as a real car. A servo can be programmed to rotate at very precise degrees, making it perfect for a self-parallel parking car application.

Chapter 3

Circuit Implementation

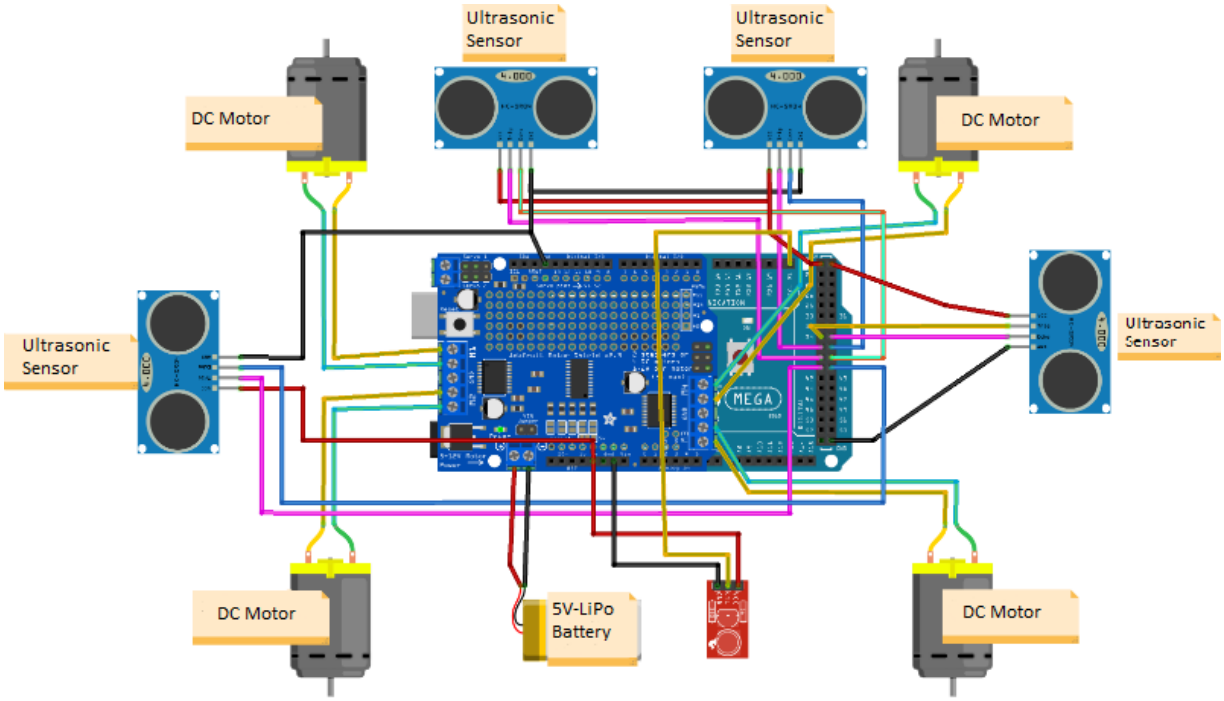

Figure 3.1: Circuit Diagram

The mentioned hardware were configured according to the above circuit diagram to form the prototype model as below.

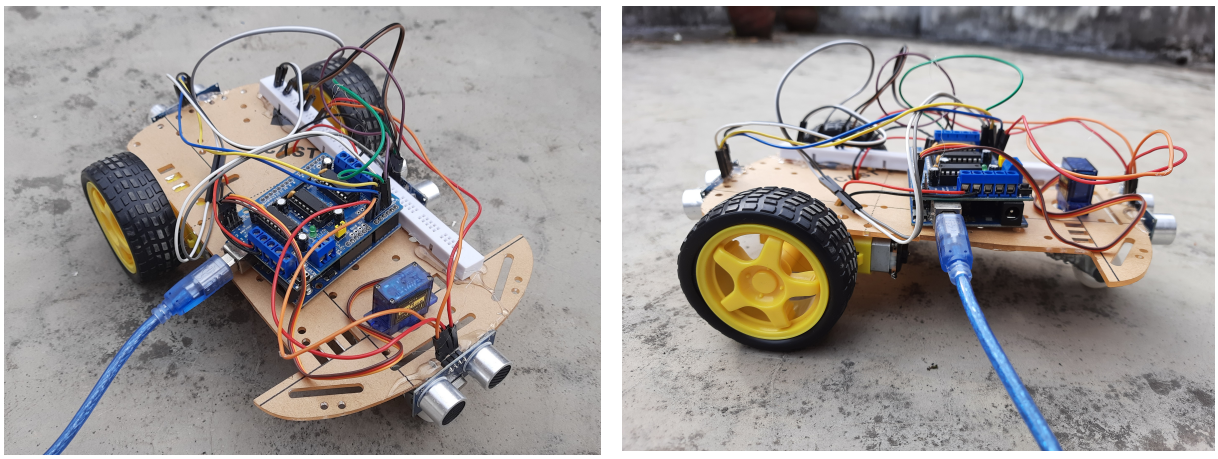

Figure 3.2: Prototype Model

Chapter 4

Algorithm & Program Code

4.1 Algorithm

- **Case 1:** If the measured value is bigger than the car and smaller than the length of the car, the parallel parking system will operate otherwise it will not initiate the parking program. In this case, the car crosses the parking area and the car stops when two sensors on the side senses the wall again. It reverses a little and turns left 45 degrees w.r.t the parking space. While reversing, the rear sensor assess the park area by measuring the distance from the car or obstacle behind it and starts to turn right. During the right turn, the sensors at the edges measure continuously and the car continue to turn left until the measured value equals each other. Stop when those are equal. The front sensor measures and the car forward until reaches a predetermined value. Parking is over.
- **Case 2:** If the measured value is shorter than the expected values car will not park.

4.2 Program Code

```
#include <AFMotor.h>
    #include <Servo.h>
    Servo servo1;
AF_DCMotor motor1(1, MOTOR12_64KHZ); // create motor #2, 64KHz pwm
    AF_DCMotor motor2(4, MOTOR12_64KHZ);
int distance;
   long duration;
   #define echoPin A2 // attach pin D2 Arduino to pin Echo of HC-SR04
  #define trigPin A3
  int distance1;
   long duration1;
   #define echoPin1 A5 // attach pin D2 Arduino to pin Echo of HC-SR04
  #define trigPin1 A4
  int distance2;
  long duration2;
   #define echoPin2 A1 // attach pin D2 Arduino to pin Echo of HC-SR04
  #define trigPin2 A0
  double  parkingspace;
   double  sidespace;
void setup() {
  servo1.write(90);
   servo1.attach(10);

  // put your setup code here, to run once:
pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin, INPUT);
  pinMode(trigPin1, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin1, INPUT);
  pinMode(trigPin2, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin2, INPUT);
  motor1.setSpeed(250);
      motor2.setSpeed(250);
      spacefind();
      Serial.begin(9600);
   }

void loop() {
  // put your main code here, to run repeatedly:
}

double backdistance()
{ digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  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; // Speed of sound wave divided by 2 (go and back)
  Serial.print("Distance: ");
  Serial.print(distance);
  return distance;
}
```

```
double sidedistance()
{ digitalWrite(trigPin1, LOW);
  delayMicroseconds(2);

  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin1, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin1, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration1 = pulseIn(echoPin1, HIGH);

  // Calculating the distance
  distance1 = duration1 * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
  Serial.print("Distance: ");
  Serial.print(distance1);
  return distance1;
}

double frontdistance()
{ digitalWrite(trigPin2, LOW);
  delayMicroseconds(2);

  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin2, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin2, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration2 = pulseIn(echoPin2, HIGH);

  // Calculating the distance
  distance2 = duration2 * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
  Serial.print("Distance: ");
  Serial.print(distance2);
  return distance2;
}
void spacefind()
{
  sidespace=sidedistance();
      servo1.write(90);
      delay(900);

 while(sidespace>10)
  {
  sidespace=sidedistance();

       motor1.setSpeed(250);
      motor2.setSpeed(250);
      motor1.run(FORWARD);

      motor2.run(FORWARD);
    }

    motor1.run(RELEASE);
    motor2.run(RELEASE);
    delay(600);
    reversedrive();
```

```
  }
void reversedrive()
{
      parkingspace = sidedistance();
servo1.write(10);
delay(900);
 while(parkingspace<12)
  {     parkingspace=sidedistance();

        motor1.setSpeed(254);
      motor2.setSpeed(150);
      motor1.run(BACKWARD);

      motor2.run(BACKWARD);
      delay(300);
   }

    motor1.run(RELEASE);
    motor2.run(RELEASE);
    delay(600);
    servo1.write(90);
    delay(900);
     parkingspace = backdistance();
    while(parkingspace>14)
  {     parkingspace = backdistance();

        motor1.setSpeed(250);
      motor2.setSpeed(200);
      motor1.run(BACKWARD);

      motor2.run(BACKWARD);
  }
 motor1.run(RELEASE);
    motor2.run(RELEASE);
    delay(600);

    sidedrive();
  }
void sidedrive()
{
      parkingspace = backdistance();
      sidespace=sidedistance();
      servo1.write(150);
      delay(900);

 while(parkingspace>5 && sidespace>2)
  {

parkingspace = backdistance();
  sidespace=sidedistance();

        motor1.setSpeed(180);
      motor2.setSpeed(250);
      motor1.run(BACKWARD);
      motor2.run(BACKWARD);
  }
    motor1.run(RELEASE);
    motor2.run(RELEASE);
    delay(600);
    servo1.write(90);
    frwddrive();



  }
void frwddrive()
{
      parkingspace = frontdistance();
      sidespace=sidedistance();
      servo1.write(80);
      delay(900);

 while(parkingspace >5)
  {     parkingspace = frontdistance();
  sidespace=sidedistance();

        motor1.setSpeed(250);
      motor2.setSpeed(250);
      motor1.run(FORWARD);
      motor2.run(FORWARD);
  }
    motor1.run(RELEASE);
    motor2.run(RELEASE);
    delay(600);
    servo1.write(90);
  }
```

Chapter 5

Working

Figure 5.1: Initializing Parking

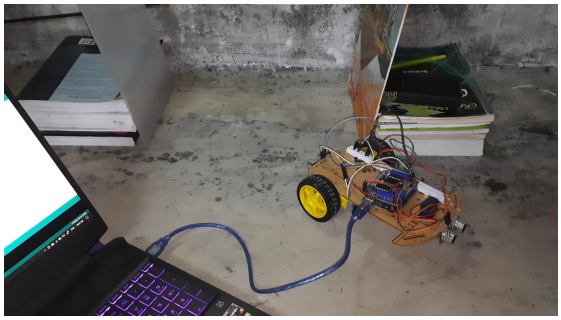

Figure 5.3: Aligning for Entry

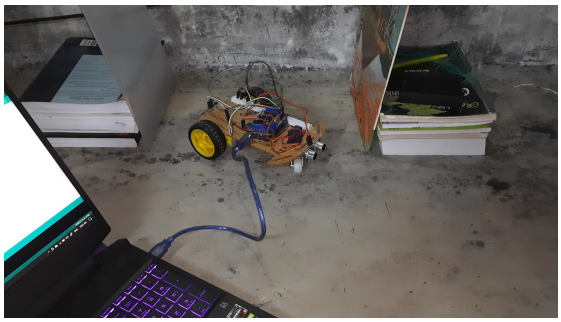

Figure 5.2: Entering Parking Space

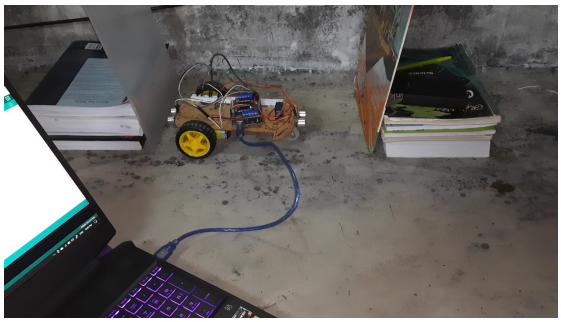

Figure 5.4: Aligning Parallel

1. Driver alignes the car for parking and then initializes the autonomous parking system.
2. The system microcontroller asses the position of the car with respect to the space available using the ultrasonic sensors.
3. Analyzing the fetched data, it alignes the car towards the parking space for entry with the help of the servo for steering control
4. After aligning the car for entry the microcontroller initiates the reversing while monitoring the distances around the car using its sensors.
5. The microcontroller stops the reversing at a preconfigured distance from the nearest obstacle suffficient enough for the car to make a left turn.
6. The car takes a left all while tyhe microcontroller assessing its position w.r.t the parking space until the car becomes parallel w.r.t the parking space
7. The car finally moves forward so that it is at a preconfigured distance from the front obstacle, sufficient enough for the car to exit the space.

Chapter 6

Advantages & Limitations

6.1 Advantages

1. Time saving.
2. Safer parking.
3. More efficient utilisation of parking spaces.
4. Lesser chances of accidents being occured.
5. Cost effective.
6. Modular and easier to install.

6.2 Limitations of Current Model

1. Not suitable for extremely precise applications (Less than a few mm).
2. Cooling is needed as the windings may heat up quickly.
3. Not suitable for movements beyond a certain angle (170°).
4. Currently limited to parallel parking only.
5. Lesser accuracy due to less number of pins in Arduino Uno for more sensors.

Chapter 7

Conclusion

Automatic car parking is one of the important factors in today's motor world. Car parking system that is discussed here is automated and will aid the driver in some tedious parking procedures.As a conclusion, this development of car robot by using Fuzzy Logic (FL) gives the best performances in term of efficiency, flexibility and robustness. Based on the result obtained, the objectives of this project are achieved. The parking system successfully assess and analyses the available parking space and will perform the necessary parking manoeuvres all while without human intervention. The time taken by the system to park the vehicle is much less than that performed by a normal human.

The parking accuracy for the current model can be further improved by extensive use of sensors across the vehicle (ideal being 3 sensors per side making a total of 12 sensors for the vehicle) all those sensors require more pin sets and the program monitoring all these sensor input demands higher processing power. Hence in future models we prefer Arduino Due or Arduino Mega with a motor driver. In future further programs containing algorithms to perform more varieties of parking procedures can be added such as vertical parking, angular parking etc.

Automated car parking is very useful in this modern world, where finding adequate parking spaces as well as performing tight parkings have became a bigger problem. This system is being modular could transform any car to a self parking car.

References

1. Pavitra,Apoorva J Shet,Ankit kumar,Avinash killikyatar - Autonomous self parking robot ,International conference on Design Innovations(IEEE ICDI3C),2018

2. M. A. Jeffril and N. Sariff, "The integration of fuzzy logic and artificial neural network methods for mobile robot obstacle avoidance in a static environment," in System Engineering and Technology (ICSET), 2013 IEEE 3rd International Conference on, 2013, pp. 325-330.

3. C.Patel, M.Swami, P.Saikia, S.Shah, —Rotary Automated Car Parking system‖,International Journal of EngineeringScience and Innovative Technology (IJESIT) Volume 4, Issue 2, March 2015.

4. Fei Frank and Kevin Hsiue-Self Parking car,Project report(MIT)

5. Lyon, D., "Parallel parking with curvature and nonholonomic constraints." Proceedings of the Intelligent Vehicles '92 Symposium, Jul. 341-346, 1992