A Self-Parking Car Graduation Project

Mechatronics Department

Kareem Abdelkader, Omar Ehab, Shehab Gomaa, Ayman Hamdy, Abdelrahman Emad, Daniel George, Peter Sameh, Abdelrahman Abdelmoez, Youssef Yacoub

Super-visor Dr. Ahmed Abdelaziz

2024

Abstract

The project is focused on achieving a single task (automatic parking) through the integration of sensors and actuators controlled by a microcontroller and a strategy (planning and coding). We will discuss the stages of project completion, from research and selection to design to implementation and testing. The car is an example of a nonholonomic system where the number of control commands available is less than the number of coordinates that represent its position and orientation, according to non-holonomic constraints, a shortest path algorithm for the parallel parking problem in a certain condition is proposed and proven. The appropriate components of the project were selected to increase efficiency and accuracy and will be explained. We will discuss the stages of developing and writing the codes required to implement the project.

Keywords

Development – Design – Components – Path Planning Algorithm – Software – Testing and Results.

1. Introduction

Advanced driver-assistance systems (ADAS) are electronic systems in a vehicle that use advanced technologies to assist the driver. They can include many active safety features, such as self-parking, driver alertness, and adaptive cruise control. For decades, parallel parking and navigating tight spaces have been the bane of many a driver's existence. This report delves into the world of self-parking cars. The problem we are working to solve is parking, and we are using embedded systems technologies to help drivers find parking spots and maneuver into them. These systems aim to improve parking efficiency, reduce driver fatigue, and reduce the risk of accidents.

Automatic parking is an autonomous car-maneuvering system that moves a vehicle from a traffic lane into a parking spot to perform parallel, perpendicular, or angle parking. The goal of the automated parking system is to improve driving comfort and safety in tight spaces where careful attention to detail and skill are needed to maneuver the vehicle. To ensure collision-free mobility within the available space, the parking maneuver is accomplished through coordinated management of the steering angle and speed, taking into consideration the real situation in the environment.

An automatic parking system uses various methods to detect objects around the vehicle. Sensors installed on the car can act as both a transmitter and a receiver, and a servo motor will control the steering angle of the car. A DC motor will drive the force to the wheels. It will all be under the control of the microcontroller.

2. Development

We began researching the importance of self-parking systems for cars, and with the development of technology and the use of electronics in all fields, including the field of cars, we made sure to seek the application of these systems in cars. We searched about how to make this project and what are the right sequences and algorithms, we have reviewed implemented projects in the field of self-parking to learn and gain experience.

Then we moved on to search for the appropriate algorithm to implement the project objectives accurately and efficiently by reviewing references and scientific research dealing with the same topic, then comparing the results with the expected results of the project and the extent of the possibility of implementing it through simulation programs such as MATLAB. After considering problems, we propose one kind of shortest collision-free path in a certain condition to approach the car to the goal following a stable trajectory while avoiding the obstacles. And an iterative path-planning algorithm is presented to park in the narrow space, which is not big enough to operate the parking operation at one time.

We were working on creating a car design based on the dimensions and design of real cars to achieve realism for our program, and a car design from BMW was chosen.

In parallel with the design, we were working on choosing the appropriate components for the project, such as SOC, sensors, communication buses, and actuators. The selection was made based on criteria for each component that suited our project and from other projects that were like our project.

3. Design

the journey of designing and manufacturing a groundbreaking prototype car inspired by the iconic BMW 2 Series. Our team meticulously crafted each component, using SolidWorks.

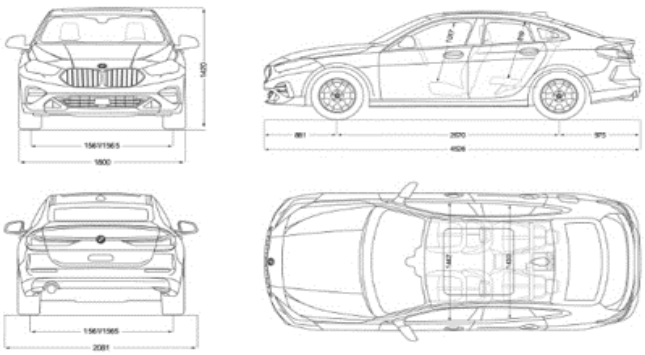

Fig 1. BWM 2 Series Car’s Dimensions

In the design phase, every part of the prototype car was drawn from scratch using SolidWorks. The dimensions of the BMW 2 Series

were inspiring to us, and the precision was exemplified in the detailed 3D models we created, carefully integrating each element, including a steering system and a power-driven system.

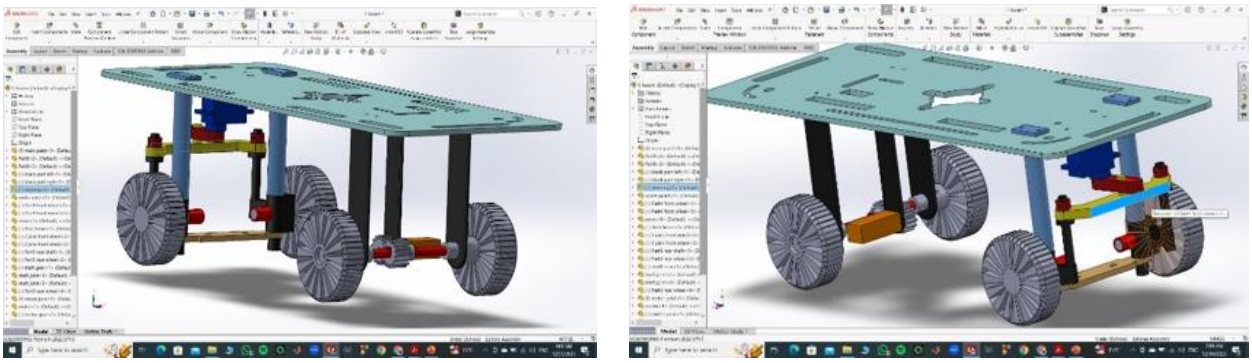

Fig 2. Car's Desing in SOLIDWORKS

A key innovation was the implementation of a power-driven system incorporating two gears with a 1:1 ratio. One gear was connected to the motor, while the other seamlessly transmitted power to the rear wheels via a shaft.

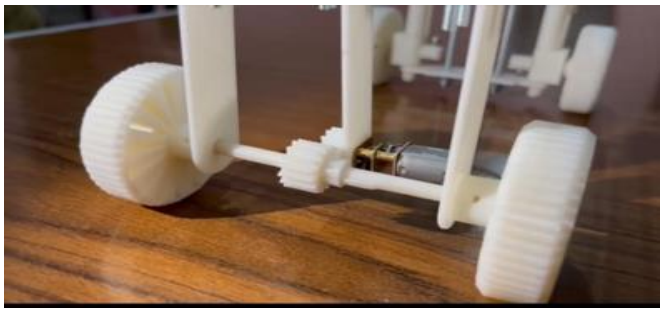

Fig 3. Power-driven System

Assembly and 3D printing, With the design phase completed, we meticulously assembled all the parts into a cohesive 3D model. The intricate components were then 3D printed, adding a layer of modern manufacturing technology to our project. The decision to 3D print

allowed for the creation of complex and customized parts, facilitating a more efficient assembly process.

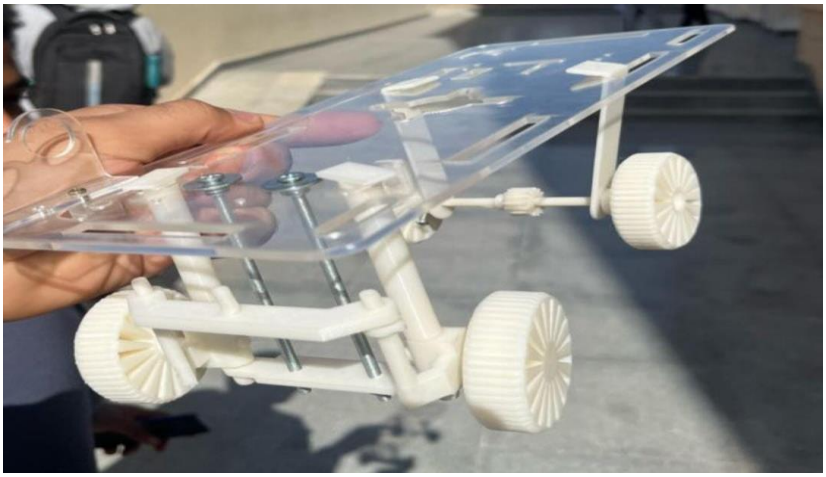

Fig 4. The Final Shape of the Car

4. Components

To reach the final goal of the project, which is to self-park the car, the appropriate components must be used to work together in terms of the possibility of interface between them, speed, accuracy, and price. Therefore, each component was studied and compared to its counterpart through practical experiments and nominations.

4.1 SOC; Blue pill board with STM32F103c8 Arm® Microcontroller.

This board was selected to ensure that the selected hardware can run algorithms in real-time and can handle the computational requirements of sensors and actuators.

STM32F103c8 Arm® 32-bit Cortex®-M3 CPU core 72 MHz maximum frequency, 1.25 DMIPS/MHz . 64 or 128 Kbytes of Flash

memory. 20 Kbytes of SRAM. Up to 40 fast I/O ports. Seven timers. Up to nine communication interfaces (I2C, USART, CAN, LIN, USB, and SPI).

Fig 5. Bluepill Board

4.2 Distance measurement sensor; Ultrasonic sensor.

Ultrasonic was selected because it is more accrued, detects a range of materials: Ultrasonic position sensors can detect and measure objects irrespective of their surface or color. and needed it to know the distance (in a different phases) to uses this distance in the mathematical modeling. It uses sound waves to measure the distance. It consists of 2 transducers; one acts as transmitter and other acts as receiver. The transmission emits an ultrasound wave at 40 KHz and if it there an object it will bounce back to the receiver.

Ultrasonic Sensor Vs IR Sensor:

	Ultrasonic Sensor	**IR SENSOR**
DETECTION TYPE	continuous or discrete	Presence/absence
RANGE	Generally, longer (up to 10 meters)	Generally shorter (up to 5 meters)
ACCURACY	More accurate for larger distances	Less accurate, affected by object shape and material

4 sensors were used, according to the standards (park on right hand), so 2 sensors were placed on the right side to determine the lateral

distance. A sensor was placed at the front and rear of the car to avoid collisions when moving.

Fig 6. Ultrasonic Sensor

4.3 DC Motor

One DC motor with a gear box is used to provide power to the wheels and cause forward and backward motion, it is installed with the rear wheels.

DC motor Vs Stepper motor:

	DC MOTOR	**STEPPER MOTOR**
CONTROL	Offer variable speed control, allowing smooth acceleration and deceleration	Require a precise sequence of pulses to move a specific number of steps.
MOTION	Have continuous motion	Have incremental motion
WEIGHT AND SIZE	A smaller size and weight which makes them more suitable	Big size and weight
NOISE AND HEAT	Often quieter and generate less heat	More noise and heat generation

Fig 7. DC Geared Motor

4.4 L298N Dual H-Bridge Motor Driver

Used to provide suitable power to the motor and to protect MCU from high voltage or current issues.

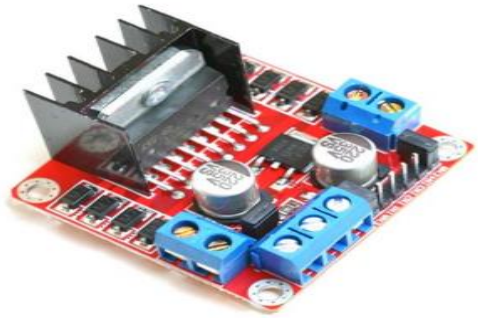

Fig 8. L298N Motor Driver

4.5 Servo Motor

It is a closed-loop system that uses position feedback to control its motion and final position. Consists of DC motor, gear box, potentiometer, and control unit. Potentiometer provides position feedback to the CU which compares current position to the target one.

Fig 9. Servo Motor SG90

4.6 Power Supply; Rechargeable lithium batteries 3.7V. We used 3 batteries.

Fig 10. Batteries

4.7 Battery Managment System (BMS)

Is technology dedicated to the oversight of a battery pack, which is an assembly of battery cells electrically organized in a row-x-column matrix configuration to enable delivery of a targeted range of voltage and current for a duration of time against expected load scenarios.

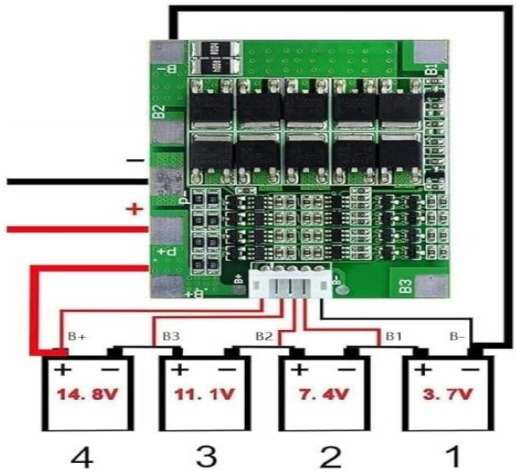

Fig 11. BMS

5. Path Planning Algorithm

Path-planning is a key issue in the automatic parking assist system. It depends on studying the kinematics of the vehicle and its surrounding environment to determine the most important parameters and then designing the scenario that includes the shortest and safest path.

Our case study will follow non-holonomic constraints for vehicles and use a low-speed method. System simulation and verification done using the MATLAB program.

5.1 Holonomic and Non-holonomic Systems

A holonomic system refers to a system that can independently control its position and orientation in all degrees of freedom. In a holonomic system, each degree of freedom is controllable. This means that a holonomic robot can move laterally, rotate, and change

orientation without needing to navigate in arcs or make complex maneuvers.

Non-holonomic Systems that have constraints on their motion and cannot move independently in all degrees of freedom. The number of control inputs (such as velocities or steering angles) is fewer than the number of degrees of freedom. As a result, the system cannot instantaneously change its position or orientation in any direction. Instead, it must follow certain constraints and specific motion patterns to achieve the desired movements.

5.2 Vehicle Kinematic Model

It is assumed that the vehicle moves with the non-sliding method in the parking process because of the low speed. In the reference coordinate system, r is the midpoint of the rear wheel, f is the midpoint of the front wheel, x=x(t) and y=y(t) are the coordinates of r, θ=θ(t) is the course angle of the car with respect to the global coordinate system, ϕ=ϕ(t) is the steering angle, v=v(t) is the velocity of f, l is the wheel base, and R is the turning radius of r.

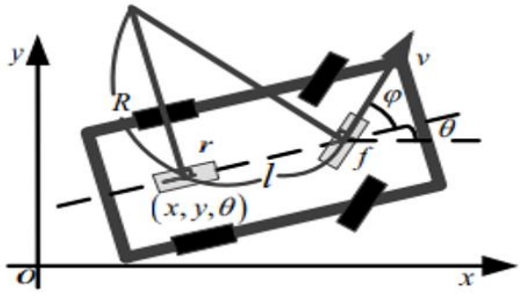

Fig 12. Kinematic model of front wheel drive car

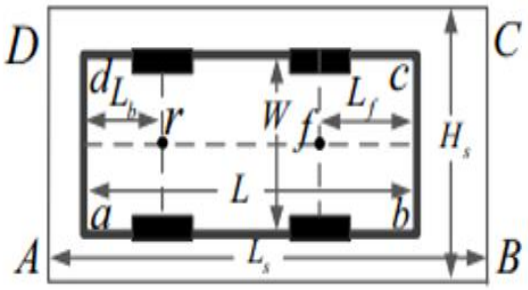

Fig 13. Description of car and parking space

The parking environment can be constructed with information from sensors. In which abcd represents the four corners of the car, while ABCD represents the four corners of parking space, let the length of the rear and front overhang be Lb and Lf, L and H are the length and width

of the car, and the length and width of parking space, which dominate the difficulty of parking, are defined as Ls and Hs.

5.3 Planning SHORTEST PATH OF PARALLEL PARKING

The working procedure of a path planner is partitioned into two parts:

- Firstly, the system decides whether the space is big enough to park by analyzing the information from the ultrasonic sensors.
- Secondly, the planner generates a feasible collision-free parking path with the consideration of choosing the appropriate start and end positions if the space meets the parking requirement.

5.4 Circle-Straight Line-Circle (CSC)

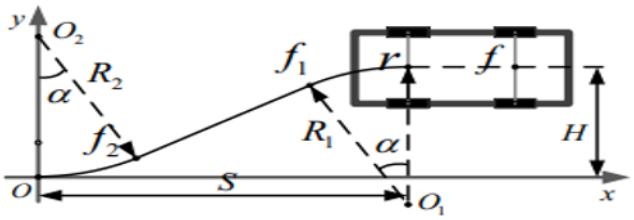

Fig 14. CSC path

Where O1 and O2 are the circle centers of the first and final stage, R1 and R2 are the radius of the circles O1 and O2, f1 and f2 are tangency points between lines and circles O1 and O2. αis the angle of the arc path, O the origin of the coordinate, and S and H are the horizontal and vertical coordinates of the start position. From Fig.3 the coordinates of O1 and O2 are (x1,y1) = (S,H-R1) and (x2,y2) = (0,R2). The parking space is defined as a narrow space when the car fails to park by one operation. For narrow parking spaces, drivers may move forward and backward several times to park; the shortest parking path is improved to adjust the narrow space circumstances to a certain extent.

The path can be described as the equations

$$\begin{cases} r \to f_1 : (x-S)^2 + \left[y - (H - R_{\min}) \right]^2 = R_{\min}^2 \\ f_1 \to f_2 : kx - y + m = 0 \\ f_2 \to O : x^2 + (y - R_{\min})^2 = R_{\min}^2 \end{cases} .$$

5.5 Planning the Parking Space Environment

The start and end positions are given. But the path planner in the parking assist system should also be concerned about the collision avoidance problem based on the planned path.

R_min occurs when ϕ is max,

R_min = l / ϕ_max

The side clearance between ab and AB is

$$d_2 = \sqrt{(R_{\min} + W/2)^2 + L_b^{\,2}} - R_{\min} - W/2$$

The front clearance between b and C is

$$R_b = \sqrt{(l + L_f)^2 + (R_{\min} + W/2)^2}$$

So, the minimum length of parking space is

$$\min L_s = \sqrt{R_b^{\,2} - (R_{\min} + W/2 + d_2 - H_s)^2} + L_b$$

The minimum width the of parking space is

Hs = W + d2

The minimum parallel distance between the car and an obstacle is

$$d_3 = (R_{\min} - W/2) - \sqrt{(R_{\min} - W/2)^2 - (S - L_s + L_b)^2} \quad (16)$$

The width of the track is

$$d_4 = R_b + (R\_min + W/2)$$

5.6 Model Verification and Simulation

To evaluate the proposed geometric path planning and steering controls for parallel reverse parking in one trial, simulation results using MATLAB were conducted.

The actual dimensions of the vehicle should be considered, the parameters of the test vehicle are:

- The car's length (L_Car) = 30 cm (about 11.81 in)
- The car's width (W_Car) = 12 cm (about 4.72 in)
- The wheelbase (L_axi) = 18 cm (about 7.09 in)
- The front overhang (Lf) = 5 cm (about 1.97 in)
- The rear overhang (Lb) = 7 cm (about 2.76 in)
- The maximum turning angle for steering wheel is within a range of (40: 35) for each side, The maximum turning angle (phi_max) = 40
- The ratio between wheelbase and cross base, the coefficients average values are gained is (0.55, 0.56, 0.66), The ratio between wheelbase and cross base (kb) = W_Car / L_axi = 0.66
- The turning ratio (R_min) = L_axi / Phi_max = 21.45 cm (about 8.44 in)

To specify the parking space environment, the following consideration parameters should be taken:

- Minimum side clearance (d2) = 0.87 cm (about 0.34 in)
- Minimum front clearance (Rb) = 35.81 cm (about 1.17 ft)
- Minimum length of parking space (Ls) = 39.30 cm (about 1.29 ft)
- Minimum Width of parking space (Hs) = 15.50 cm (about 6.1 in)

- The coordinate points of the parking space

Initial position and orientation of the vehicle:

- The horizontal coordinate of start position (S) = Ls
- The minimum parallel distance between the car and the obstacle d3
- The vertical coordinate of start position (H) = W_Car + d3

We have three phases of movement,

- From start position r to f1
- From f1 to f2
- From f2 to end position O

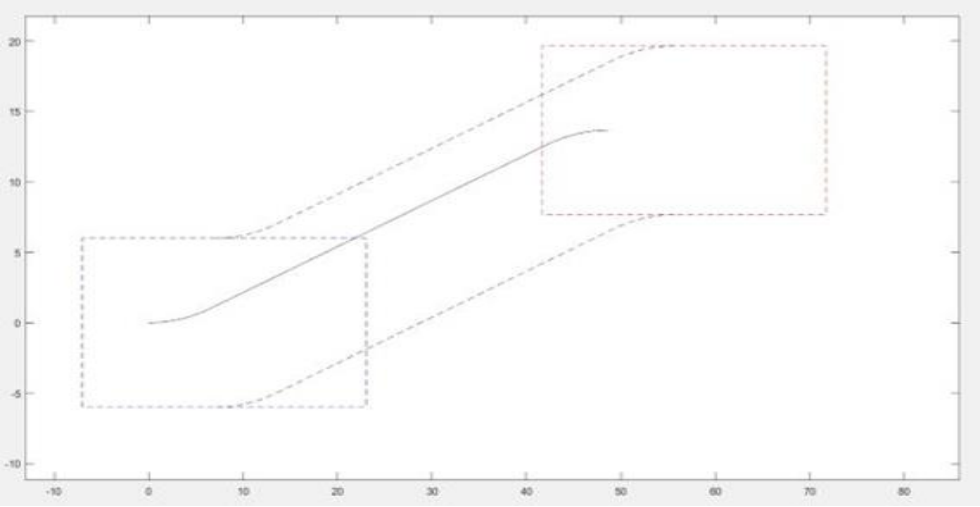

Fig 15. Path Generation

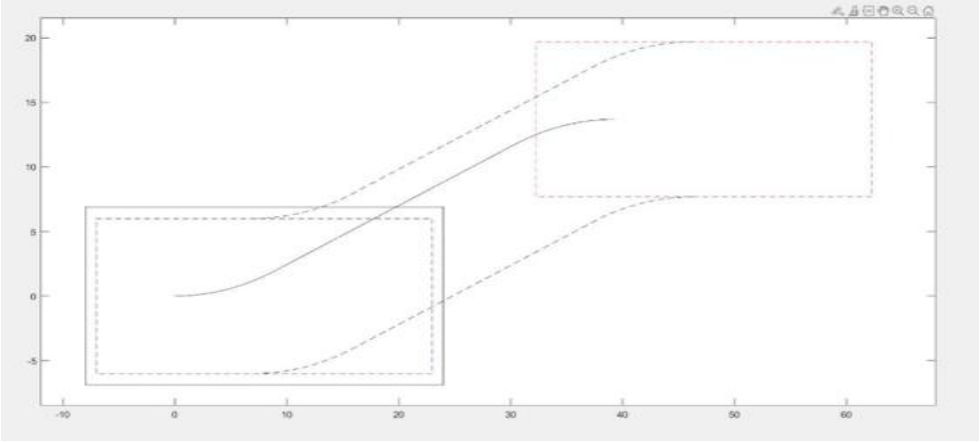

Fig 16. The Car Follows the Path

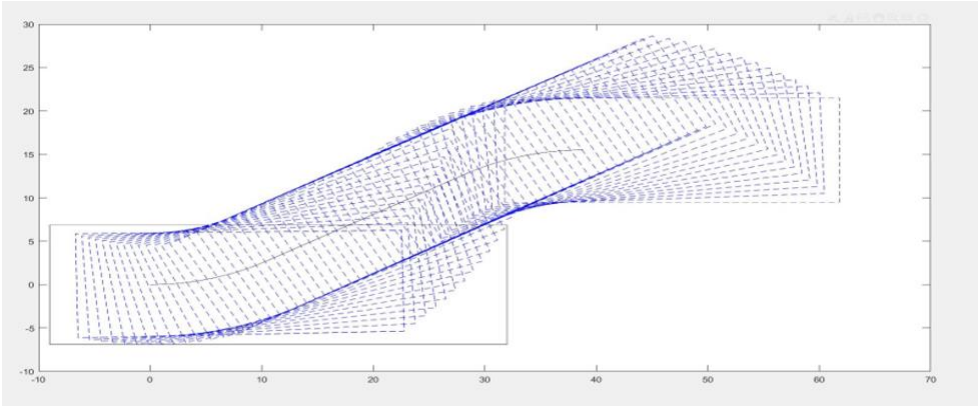

Fig 17. Model Verification

6. Software

To use any of the features of the controller, sensors, and actuators, their registers must be activated through the CPU. Therefore, we used the C and Embedded C programming language with the STMCube IDE program. SW implementation depends on two stages, SW configuration and testing/debugging.

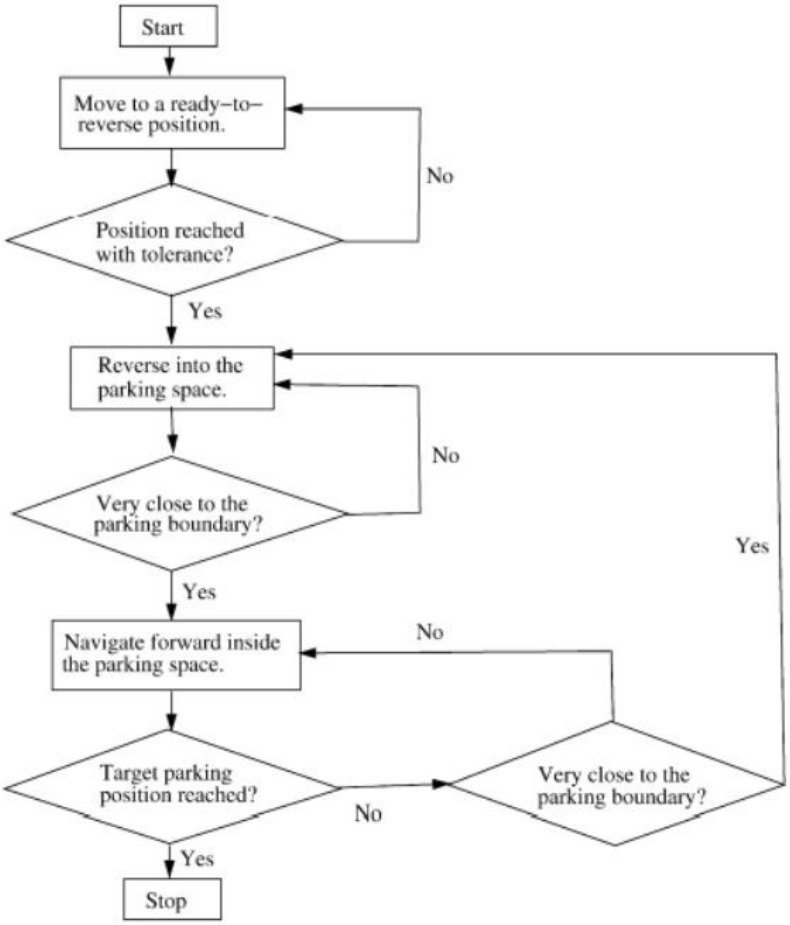

Fig 18. System Flowchart

6.1 SW Configuration Implement

To program the controller and use its peripherals, you must read its Technical Reference Manual (TRM) and all its features, as well as the data sheet for the used board.

Technical Reference Manual (TRM) shows the following, this provides that how the peripheral work and how to use it:

- Peripheral Introduction.
- Peripheral Function Description.
- Peripheral Registers.

Data sheet of the used board shows the following, this provides that the steps need for configuration the peripheral:

- Block Diagram.
- Clock Tree.
- Memory Mapping.
- Pin Definitions.

For each peripheral two configuration files. Header file (file.h) that includes peripheral configurations, user definitions, and APIs functions, Source file (file.c) that includes the implementation of APIs functions that meet our needs from the peripheral.

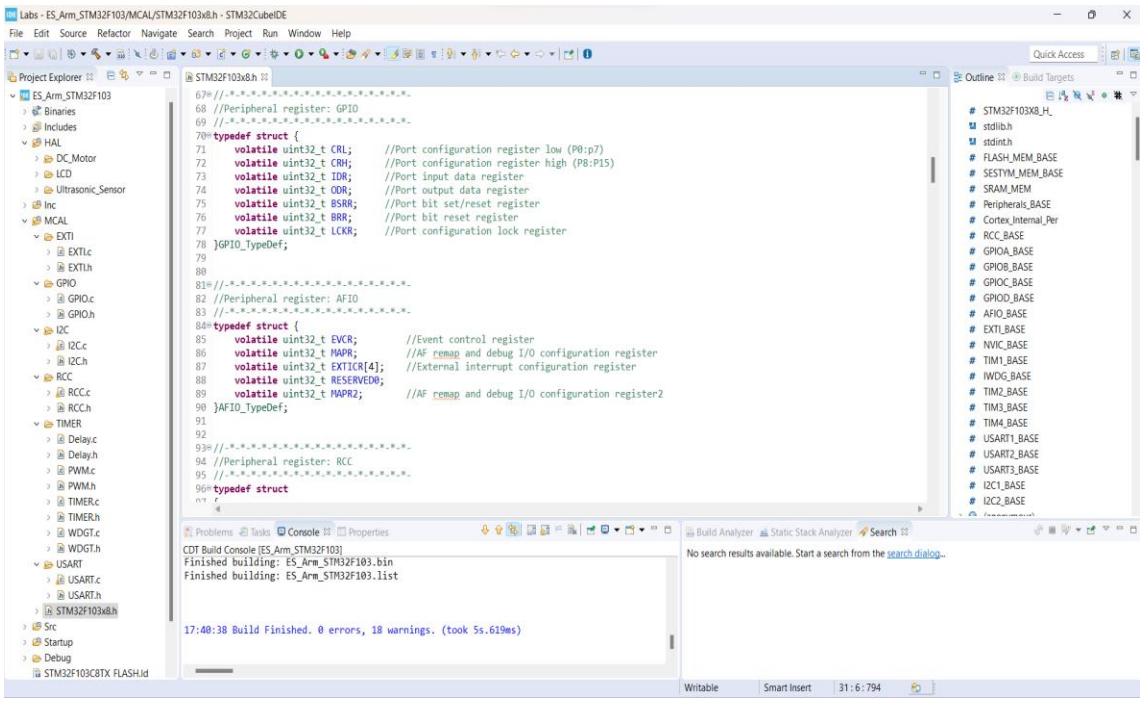

Fig 19. Software IDE and Files

6.2 Testing and debugging.

This stage contains the testing of each microcontroller peripheral, sensors, and actuators drivers and the logic of interfacing them together.

6.3 System Features and Functionalities

Features used by the controller:

- General Purpose Input/Output (GPIO). It is used to configure the controller's pins to interface with sensors and actuators by setting them as either input or output, can be routed to other peripheral to serve some alternate functions.
- Interrupt Service Routen (ISR) and Nested vectored interrupt controller (NVIC).
- Timers and Counters. They used to make action at a specific time, work by counting clk cycles from the system clock.
- Universal Asynchronous Serial Receiver and Transmitter (UART). Is a standard serial communication protocol that can transmit data between devices.
- Inter Integrated Circuit (I2C). Is a serial communication protocol used to connect multiple low speed devices to an MCU.

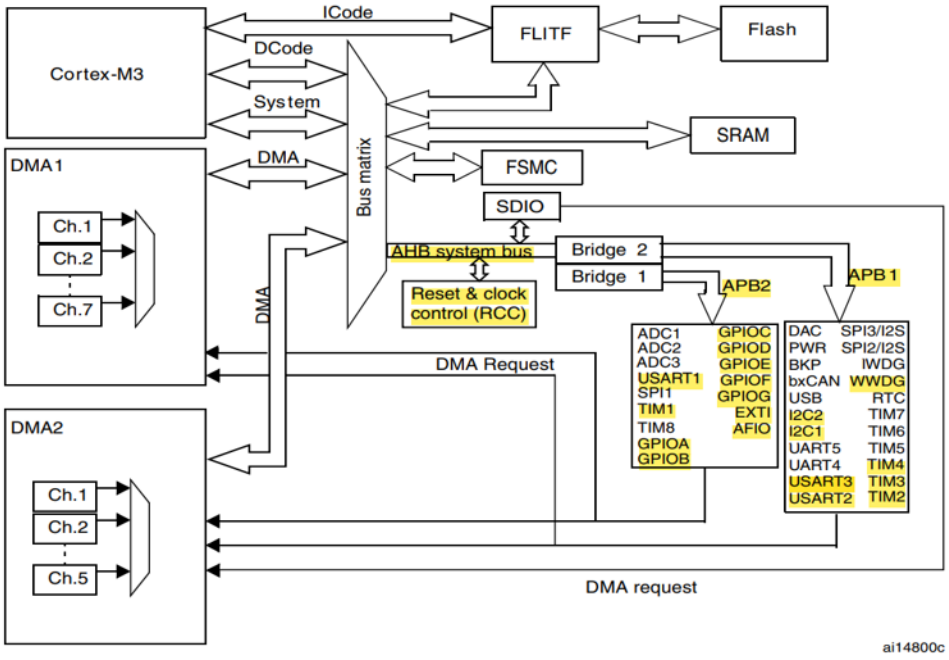

Fig 20. MCU System Architecture

We are seeking to create an SW system to determine the empty location in which the car can park. The microcontroller can detect that an empty spot has been marked through ultrasonic sensor readings. We will use ultrasonic sensor to make a space mapping and measure the distances. Our system uses low-speed RPM that is provided from the DC motor. Servo motor will control the steering angle. Through the system, the parking process will begin.

We can use path planning algorithm and link it with software in two stages:

First stage: empty slot detection

- Make a counter for front side ultrasonic (U1). When the value of U1 is 15.5cm or more, start counting. When the value of U1 is less than 15.5cm, stop counting. The counter will return time, if time is equal 10.25 seconds or more, stop the DC motor and start parking.

Second stage: parking motion and it consists of three phases,

- Phase 1: from start point (r) to the first point of the straight line (f1). Servo motor will change its angle to 50 and DC motor will work for time (t1).
- Phase 2: from the first point of the straight line (f1) to the last point of the straight line (f2). Servo motor will change its angle to 90 and DC motor will work for time (t2).
- Phase 3: from the last point of the straight line (f2) to the end point. Servo motor will change its angle to 130 and DC motor will work for time (t1).

To achieve Advanced Driver Assistance Systems (ADAS), we work on making Real Time Operating System (RTOS). That will help our project improve its functionalities in real time and make the right decisions for

each problem. This is done by determining the number of tasks, their priority, their duration, and whether the task is dependent on another task or not, then distributing them to the CPU to extract the program.

7. Testing and Resulting

We have adopted a specific approach in this part, which is Upon completion of a certain part (studying, implementing, and testing), one moves to the next part. It's called Waterfall Model.

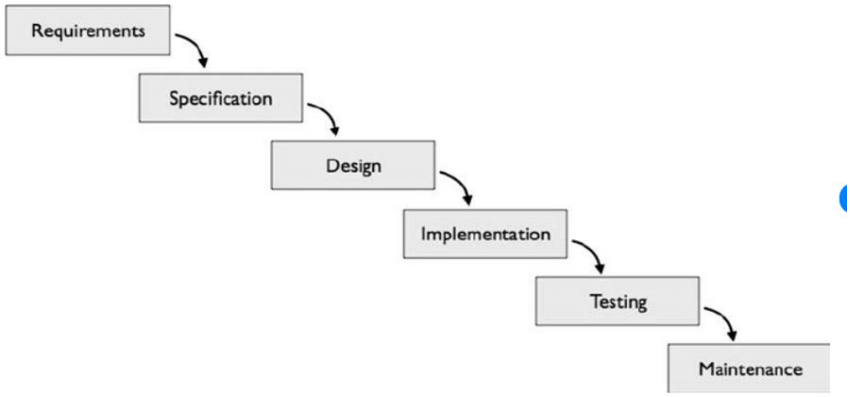

Fig 21. Waterfall Model

A HW/SW partitioning algorithm implements a specification on some sort of multiprocessor architecture. Each part is verified through practical experiments, simulations and evaluations.

Software Testing Life Cycle (STLC) consists of a series of activities carried out methodically by Testers methodologically to test your software product. The software part had the largest part of testing and verification, and this was done by testing each drive for each microcontroller feature, sensors, and motors by using Keil Micro Vision program and physical components.

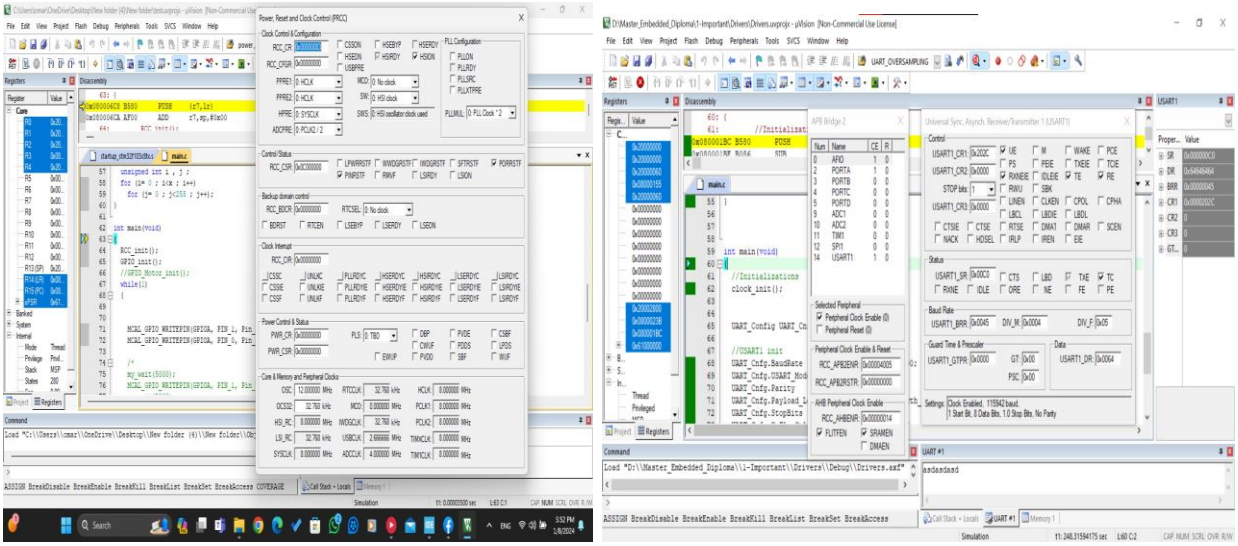

Fig 22. Code Debugging in Kiel Program

Conclusion

This paper explores the realm of self-parking automobiles, which assist drivers in locating and parking spaces through embedded systems technology. The devices are designed to lessen driver fatigue, increase parking efficiency, and lower the chance of collisions. We created car design, parts, and algorithms. To reach the true experience of self-parking and come closer to reality, we used actual automobile measurements. We used the SOLIDWORKS program to design every part, and 3D printing was used to make every item. The project needed thorough component evaluation and selection based on speed, accuracy, cost, interface compatibility, and other considerations. In order to accomplish the goal in constrained settings, the research used a low-speed approach with non-holonomic limitations for vehicles to accomplish the aim in confined locations. MATLAB was used for system modeling and verification. The dimensions of the parking area and the automobile itself determined the length and width of the parking

environment, which was built using sensor data. Equations were used to characterize the parking path, and the car's width, wheelbase, turning angle, and clearance requirements were taken into consideration for determining the minimum size of the parking space. We are developing a software system that uses a servo motor, DC motor, and ultrasonic sensor to identify an empty parking space.

References

[1] Zhenji Lv, Linhui Zhao, Zhiyuan Liu, “A path-planning algorithm for parallel automatic parking”, Third International Conference on Instrumentation, Measurement, Computer, Communication and Control 2013
[2] Dainis Berjova, “RESEARCH IN KINEMATICS OF TURN FOR VEHICLES AND SEMITRAILERS”, Latvia University of Agriculture, Faculty of Engineering 2008
[3] American International Journal of Sciences and Engineering Research, “Autonomous 4WD Smart Car Parallel Self-Parking System by Using Fuzzy Logic Controller”, American Center of Science and Education 2019
[4] STM32F10Xxx Technical Reference Manual
[5] STM32F103c8 Datasheet
[6] HCSR04(Ultrasonic sensor) Datasheet
[7] User guide L298N Dual H-Bridge Motor Driver