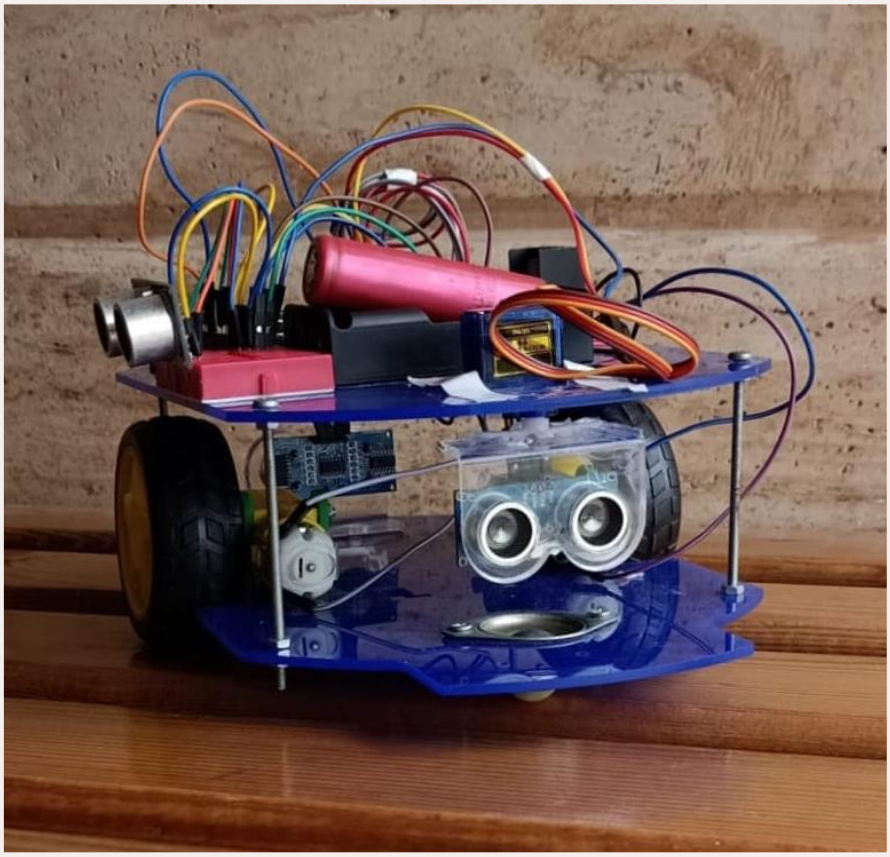
جامعة الإسكندرية
ALEXANDRIA UNIVERSITY

OBSTACLE AVOIDING AND AUTO-PARKING CAR



Team Members

Amany Fathy	2022565968
Hagar Tarek	20221310699
Loreen Mohamed	20221377256
Nada Mohamed	20221377458
Sama Ahmed	20221377562

Content

1 Problem Description

- With technology advancement, every industry is aiming to turn their operations to be automated and best-case scenario is to achieve full automation. The car industry is one of the leading industries that has changed due to automation, with automation levels ranging from level 0 (no automation) to level 5 (full autonomous car).
- Our objective was to design and implement an autonomous robot that can avoid obstacles and auto-park itself, making it a level 2 automation, partially automated.
- The problem here was how would we implement such idea using non-industrial components, like Arduino uno board and simple h-bridge motor driver. Also, apart from the system design we thought about challenges facing each required task:

1. **Obstacle avoidance:**
    - Choosing best sensor to detect the presence of obstacle without adding unnecessary cost.
    - When obstacle is found, then choose the best path to continue driving in which is the path with greater distance as it means that it's a free path.

2. **Auto-parking:**
    - The number of sensors needed for optimal parking.
    - Challenges of parallel parking differ from perpendicular parking. As in perpendicular parking, once free area was found then we'd control the car to move 90° then return backward. But parallel parking should put into consideration front and back object and to not hit them and well as not getting too close to the parallel surface as to not damage the car.

2 System Design

The autonomous car project utilizes an Arduino Uno board, an H-bridge, a breadboard, jumper wires, and two lithium batteries to create a self-driving vehicle capable of object avoidance and parallel parking. The car body is equipped with the car base, two motors, and two wheels, as well as three ultrasonic sensors - one in the front, one in the back, and one on the side.

Components and Functionality:

1. Arduino Uno Board:
    - The Arduino Uno is the main microcontroller that coordinates the operation of the entire system.
    - It receives input from the various sensors, processes the data, and sends appropriate control signals to the motors and servo to navigate the car.

2. H-bridge:
    - The H-bridge is an integrated circuit that allows the Arduino to control the direction and speed of the two motors independently.
    - It enables the car to move forward, backward, and perform turns by controlling the rotation of the motors.

3. Breadboard and Jumper Wires:
    - The breadboard provides a temporary and reconfigurable platform for connecting the various components of the system.
    - The jumper wires are used to establish the necessary electrical connections between the components.

4. Resistor:
    - Resistors lower voltage across components, ensuring they operate within safe limits.

5. Lithium Batteries:
    - The two lithium batteries provide the power supply for the entire system, ensuring the car can operate autonomously without being tied to a power source.

6. Car Body, Motors, and Wheels:
    - The car’s body provides the physical structure for the electronic components and the two motors.
    - The two motors are responsible for driving the two wheels, allowing the car to move in different directions.

7. Ultrasonic Sensors:
    - The front ultrasonic sensor is attached to a servo, enabling it to rotate 180 degrees to detect obstacles in the car's path.
    - The front and back ultrasonic sensors are mounted on holders to maintain their position and orientation, providing a stable and reliable detection of obstacles in front and behind the car.
    - The side ultrasonic sensor helps the car detect objects or obstacles during parallel parking.
    - The sensor data is processed by the Arduino, which determines the appropriate course of action, such as slowing down, turning, or stopping the car to avoid obstacles and find the optimal parking strategy.

8. Mode Switching Button and LED:
    - A button has been added to allow the user to switch between the obstacle avoidance and parallel parking modes.
    - A LED has been included to indicate when the parallel parking mode is active.
    - The button and LED are connected to the Arduino using pull-up resistor circuits to ensure reliable operation.

3 Solution Implementation

Obstacle Avoidance

The obstacle avoidance solution uses a simple yet effective approach. It leverages the ultrasonic sensor to detect obstacles in front of the car and then determines the appropriate action based on the relative distances to the left and right sides. This allows the car to navigate through its environment and avoid collisions with obstacles.

1. Sensor Integration and Obstacle Detection:
    - The code utilizes an ultrasonic sensor to measure the distance to obstacles. The ultrasonic is attached to a servo in the front side of the car to be able to turn around 180 degrees to check for any obstacles and find the clear path to move through.
    - We used the NewPing library in Arduino which is designed for interfacing with ultrasonic distance sensors and other compatible sensors. This library provides a simplified way to measure distances using ultrasonic sensors and offers features like reducing the number of sensor connections, simplifying code, and improving measurement accuracy.
    - The `readPing()` function is responsible for reading the distance from the ultrasonic sensor and handling cases where the sensor returns a value of 0 cm (interpreted as a very close obstacle).

2. Autonomous Navigation and Path Planning:
    - The obstacle avoidance logic is integrated into the car's movement, allowing it to dynamically adjust its path based on the detected obstacles.
    - The `lookLeft()` and `lookRight()` functions are used to calculate the distance from the left and right sides to check the path available for the car to move.
    - The `forward()`, `backward()`, `right()`, and `left()` functions are used to control the car's movement and execute the appropriate actions (stop, move forward, turn right, or turn left) based on the obstacle detection.

3. Obstacle Avoidance:
    - The `obstacle_avoid()` function is responsible for handling the obstacle avoidance logic.
    - It starts by reading the distance from the ultrasonic sensor using the `readPing()` function.
    - If the distance is less than 20 cm, the car stops, moves backward for 400 milliseconds, and then calculates the distance to the left and right sides using the `lookLeft()` and `lookRight()` functions.
    - Based on the comparison of the left and right distances, the car turns either left or right to avoid the obstacle.
    - If the distance is greater than 20 cm, the car continues to move forward.

Auto Parallel Parking

The auto parallel parking solution is a well-structured and comprehensive approach to implementing an auto parallel parking feature. The combination of sensor-based detection, stepwise parking strategies, and precise movement control allows the car to navigate and park in a parallel parking spot effectively.

1. Sensor Integration:
    - The solution relies on the ultrasonic sensor readings to make informed decisions at each stage of the parallel parking process. This ensures that the car can adapt to the specific conditions of the parking spot and make appropriate adjustments as needed.
    - We are using 3 ultrasonic sensors:
        - Front sensor to calculate the distance from the front to check if the car can move forward.
        - Back sensor to calculate the distance from the back to check if the car can move backward.
        - Side sensor to calculate the distance of the parking slot to detect a parking slot and check If it fits the car.
    - We used the Ultrasonic library in Arduino which is designed for working with ultrasonic distance sensors. This library simplifies the process of interfacing with these sensors, allowing us to measure distances accurately and efficiently.

2. Parking Spot Detection:
    - The `Check_Parking_Spot()` function is responsible for detecting a potential parking spot and confirming its dimensions.
    - It uses front and side ultrasonic sensors to gather distance information.
    - The function follows a series of conditional checks to determine the parking spot status:
        - If the side sensor distance is less than or equal to 15 (width of the car) and the parking status is 0 (no parking spot detected), the car is instructed to move forward, and the parking status is set to 1 (potential parking spot detected).
        - If the side sensor distance is greater than 15 (width of the car) but less than 28 (length of the car), and the parking status is 1, the car is instructed to move forward, and the parking status is set to 2 (confirmed parking spot dimensions).
        - If the side sensor distance is less than or equal to 15 (width of the car) and the parking status is 2, the parking status is set to 3 (decide to park parallel).

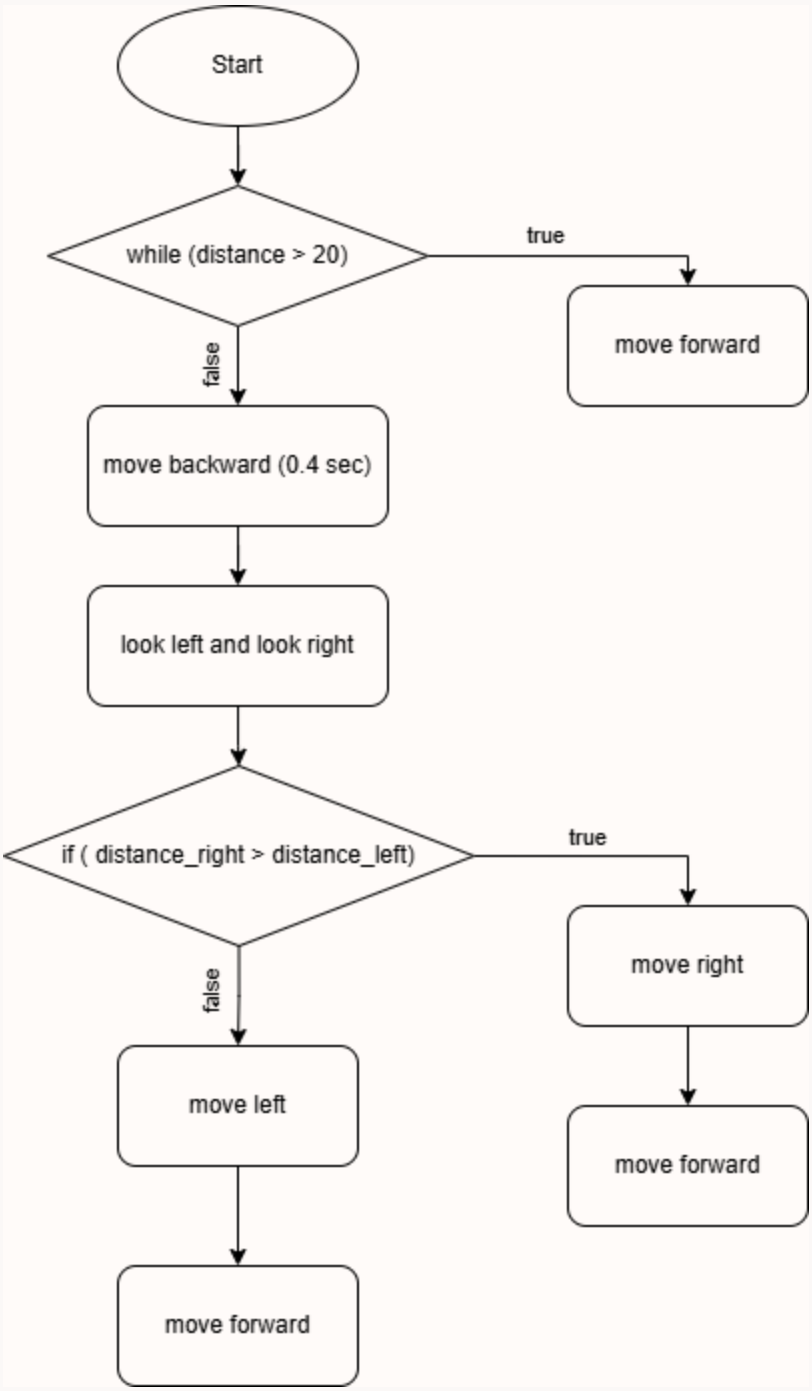
3. Parallel Parking Strategy:
    - The `Find_Park()` function coordinates the parallel parking strategy.
    - When the parking status is 3 meaning that the parking spot is available, so the car is stopped, and the parking status is set to 4 (start parallel parking).
    - In the parking status 4, the car performs the first step of the parallel parking process:
        - It moves backward for 200 milliseconds, stops, and then turns right for 400 milliseconds.
    - The parking status is then set to 5 (adjust position).
    - In the parking status 5, the car adjusts its position to align with the parking spot:
        - It moves backward for 100 milliseconds and checks the distance from the back sensor.
        - If the back sensor distance is greater than 7 cm and less than or equal to 15 cm, the car stops, and the parking status is set to 6 (final alignment).
    - In the parking status 6, the car performs the final alignment:
        - It turns left for 200 milliseconds and checks the distance from the side and back sensors.
        - If the side sensor distance is less than or equal to 8 cm and the back sensor distance is less than or equal to 9 cm, the car stops, and the parking status is set to 7 (final check and adjustments).

3. Parallel Parking Strategy:

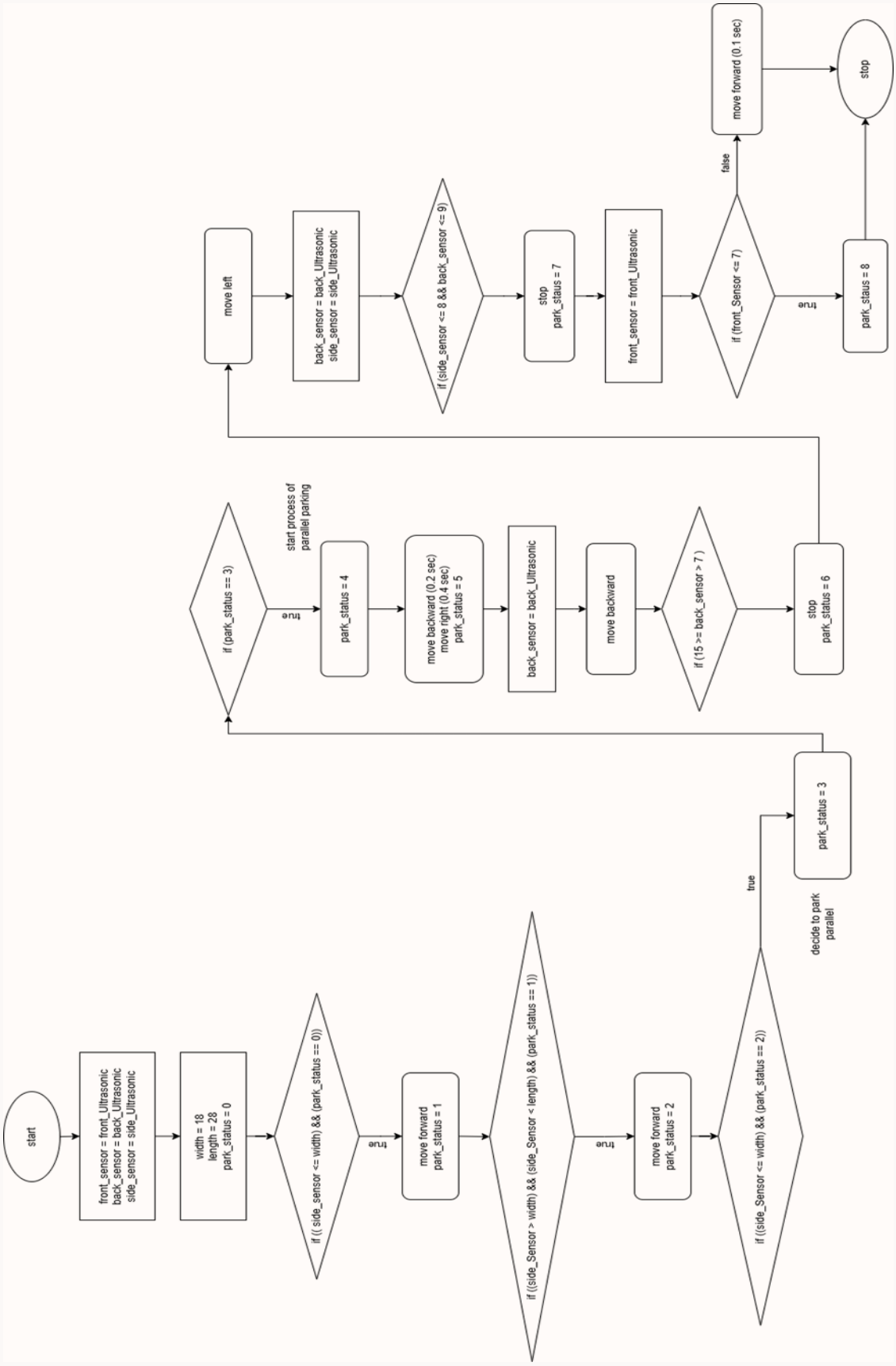
- In the parking status 7, the car performs a final check and adjustment:
  - It checks the distance from the front sensor.
  - If the front sensor distance is less than or equal to 7 cm, the car stops, and the parking status is set to 8 (parked).
  - If the front sensor distance is greater than 7 cm, the car moves forward for 100 milliseconds.

4 Detailed flowchart

Obstacle Avoidance



Auto Parallel Parking



5 Cost analysis

Component Name	Units	Unit price	Total price
Car body	1	200 EGP	200 EGP
Arduino UNO	1	465 EGP	465 EGP
Bread board	1	15 EGP	15 EGP
Jumpers	32	1 EGP	32 EGP
Ultrasonic sensor	3	45 EGP	135 EGP
Ultrasonic sensor holder	2	15 EGP	30 EGP
Lithium Battery	2	65 EGP	130 EGP
Battery holder	1	21 EGP	21 EGP
H Bridge	1	90 EGP	90 EGP
Servo	1	110 EGP	110 EGP
Resistor	1	0.5 EGP	0.5 EGP
Button	1	2.5 EGP	2.5 EGP
Free wheel	1	35 EGP	35 EGP
Led	1	1 EGP	1 EGP

The total cost of 1,267 EGP for the autonomous car system is an appropriate and justified cost for the following reasons:

- The selection of each component was carefully considered to balance functionality, performance, and cost-effectiveness.
- By choosing cost-effective yet reliable components, such as the Arduino Uno, ultrasonic sensors, and basic electronic parts, the project was able to keep the overall cost at a reasonable level.
- The cost of 1,267 EGP reflects the level of complexity and the range of features included in the system, which would be difficult to achieve at a significantly lower cost.

Budget Constraints:

The design process for the autonomous car system aimed to minimize costs without compromising the overall functionality and performance of the vehicle. By selecting cost-effective components, such as the ultrasonic sensors and the Arduino Uno, the project was able to stay within a reasonable budget while still achieving the desired capabilities.

In this project, the decision was made to use three ultrasonic sensors instead of a camera-based system. The key reasons for this choice are:

- **Cost:** Ultrasonic sensors are generally more affordable than high-quality camera modules, making them a more cost-effective solution for this project. The camera's cost is estimated to be 800 EGP, while the three ultrasonic sensors together cost 135 EGP, so we were able to save about 665 EGP.
- **Simplicity:** Ultrasonic sensors provide direct distance measurements, which can be easily integrated into the Arduino-based control system. Camera-based systems often require more complex image processing and algorithms, which can increase the overall complexity and cost of the project.

6 Conclusion

In conclusion, our autonomous car project, which focused on object avoidance and parallel parking, successfully met its goals and objectives through the effective use of embedded systems. By integrating ultrasonic sensors, we enabled the car to accurately perceive its surroundings, ensuring reliable detection of obstacles and precise execution of parallel parking strategies. The seamless interaction between hardware components and software algorithms demonstrated the viability of ultrasonic sensor technology in enhancing the capabilities of autonomous vehicles.