An-Najah National University

Faculty Of Engineering And Information Technology

Computer Engineering Department

Solar Panel Cleaner

By:
Mohamad Jondi
Ahmad Amira

Supervisor:
Dr. Anas Toma

Acknowledgment

We would like to express our heartfelt gratitude to Dr. Anas toma, our supervisor, for his continuous support throughout our project.We would also like to thank the academics in the Department of Computer Engineering for their expertise and assistance for the whole 5 years of studying engineering that lead us to this moment, as well as our friends and family for their belief in our abilities. We are deeply grateful for their contributions, support, and belief in our capabilities.

Disclaimer

The following report has been authored by students from the Computer Engineering Department, Faculty of Engineering, An-Najah National University. The report has undergone minimal modifications, limited to editorial corrections, and may still contain errors in language and content. It is important to note that the opinions expressed within the report, including any conclusions and recommendations, solely belong to the students. An-Najah National University bears no responsibility or liability for any consequences arising from the utilization of this report for purposes other than its intended commission.

Abstract

Solar panels are a popular and environmentally friendly method of generating electricity. However, over time, solar panels can get dirty and dusty, which reduces their efficiency. Regular cleaning is essential to ensure optimal performance. However, the problems with traditional cleaning methods are mainly the high cost of manual labor and the potential for human error. Moreover, there are risks that come with cleaning solar panels in high places.

This project aims to develop a solar panel cleaner robot. By automating the cleaning process, the robot offers a cost-effective solution that can improve energy generation and extend the lifetime of solar panels. The project aims to cover important aspects such as autonomous navigation, effective cleaning mechanisms, and safety measures to prevent harm to the solar cells.

To achieve these objectives, the robot will incorporate a range of sensors and utilize hardware components. For autonomous navigation, edge detection sensors like ultrasonic or infrared sensors will enable the robot to detect and remain within the frame of the cell. To ensure mobility, a robust system consisting of wheels or tracks will be employed to navigate the solar panel array. Additionally, elastic materials for the wheels will be used to provide stability and prevent slipping, especially considering the different angles of the cells. The robot will also include a sensor for monitoring the water level in the tank, providing alerts when it is empty. Furthermore, the addition of a solar panel to the robot itself is being considered.

While similar projects have been done before, this project aims to enhance the existing designs by adding more features and improving stability. The intention is to create a solar panel cleaner robot that is more efficient and reliable.

Contents

Chapter 1

Introduction

Solar panels are devices that harness the power of sunlight and convert it into electricity, making them among the top choices for renewable energy. However, as time passes, the build-up of dust and dirt can gradually reduce the efficiency of these solar cells. Manual cleaning, on the other hand, poses significant challenges, it can be very expensive, sometimes costing as much as 500 NIS per worker per day in certain locations. Furthermore, it can be dangerous due to the placement of solar panels in high and difficult-to-access locations with varying angles.

1.1 Problem Statement

In short, we need a better solution for cleaning the solar panels that is better than the traditional Manuel labor that is cheap and reliable

1.2 Scope of Work

1. Movement and Cleaning System: This includes implementing the body of the robot, which have many components such as the water pump, brush, and motors responsible for the robot's movements.

2. Control System: Developing the control system that consists of Arduino microcontrollers, various sensors for navigation and obstacle detection, and an ESP32 for wireless communication.

3. Water System: Designing a separate water supply system, connected using another ESP32, to ensure the robot can access water when its internal water tank is empty during cleaning operations.

4. Web Application: Creating a web application that allows manual control of the robot.

1.3 Significance

Our project is significant because it offers a cost-effective and safe solution for maintaining solar panels, which is not only good for the environment but also for reducing electricity costs.

Chapter 2

Constraints and Earlier Coursework

2.1 Constraints & Limitations

1. The angle of the solar panel cells was the biggest problem since we want the robot to move on the cells horizontally and vertically without slipping or falling.
2. Finding the appropriate type of motors that has enough torque to move the robot.
3. The ultrasonic sensors give inaccurate readings.
4. Making the two servo motors work in sync with each other to carry the brush.

2.2 Earlier Coursework

1. Learn about electronic systems and technologies in an electronics course.
2. Understand microcontrollers and their practical uses in a microcontrollers course.
3. Get hands-on with Arduino and learn how to control stepper motors in the microcontrollers lab.
4. Develop critical thinking and research skills, like reading science papers and using tools like LaTeX to write research papers.

Chapter 3

Literature Review

3.1 history

The story of solar panel cleaning robots starts about 15 years ago when people began to use solar panels more. Back then, we didn't have robots like we do now. People had to clean the panels by hand using water and brushes, which wasn't very efficient. But in 2008, a company called Kärcher made a robot called the SolarBrush, one of the first ones. After that, more and more robots were developed, like the Engineers for Change robot in 2013, which ran on solar power. These early robots helped set the stage for the better robots we have today. These robots are really important because they help keep solar panels working well. Over time, dust, dirt, and other stuff can cover solar panels, making them less effective. Studies have shown that dirty panels can lose up to 30Our robot is pretty special. It can work in different places, like on your roof or in a big solar field. It's smart too, using sensors and computer brains to figure out the best way to clean the panels. It's also good for the environment because it doesn't waste water or energy. Our robot is all about making sure your solar panels work great while being friendly to the planet and cost effective.

Chapter 4

Methodology

4.1 Design

To begin our project, we started with an initial design, as shown in Figure 4.1. This design served as the foundation for the making of the body for our solar panel cleaning robot.

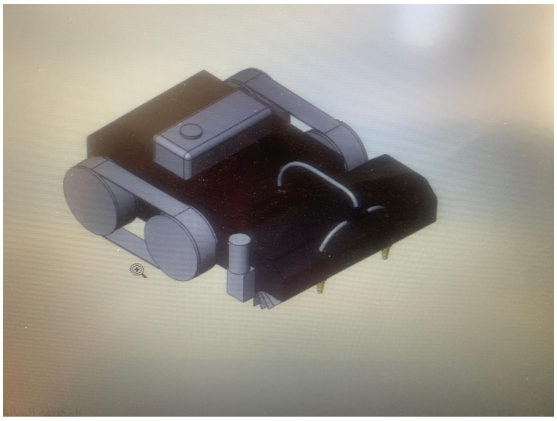

Figure 4.1: Initial Design of the Solar Panel Cleaning Robot

4.2 DC Motors

4.2.1 Four 5 RPM DC Motors

These motors are responsible for the movement of our robot. although low in RPM (Revolutions Per Minute) they make up for it with their high torque to withstand the weight of the robot , but still giving the robot some limitation in the department of speed .

Figure 4.2: 4 RPM DC Motor

4.2.2 Two 250 RPM Yellow DC Motors

The 250 RPM Yellow DC Motors provide enough power to carry the brush and make it spin to achieve the necessary cleaning

Figure 4.3: 250 RPM Yellow DC Motor

4.3 Servo Motors

4.3.1 Two Servo Motors

These servo motors carry the attached body of the brush and motor to control whether to put it on the ground or to lift it to smoothing the turning of the robot.

Figure 4.4: Servo Motor

4.4 Microcontrollers and Controllers

4.4.1 Two ESP32 Microcontrollers

The ESP32 microcontrollers serve as helper and driver for the main robot brain which is the Arduino mega in the first system and helps it receive manual commands and when to start the system , in the second system it acts like a receiver to just open the water tank .

Figure 4.5: ESP32 Microcontroller

4.4.2 Arduino Mega

The Arduino Mega is used for overall system control and coordination, making all the readings from the sensors and deciding what to do next

Figure 4.6: Arduino Mega

4.5 Water Pumps

connect to the internal and external water tanks , these two pumps provide the system with continues water supply to make efficient cleaning possible.

Figure 4.7: Water Pump

4.6 Sensors

4.6.1 Ultrasonic Sensors

An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal, by connecting 5 of them on the robot body , one on the front and one on the far front , one on each side and one on the back facing downwards the robot can navigate through the solar panel no matter the size of it and even detect if there is a solar panel next to it that it can move to , using the far forward sensor.

Figure 4.8: Ultrasonic Sensor

4.6.2 Puck Converters

Puck converters are essential for efficiently managing power distribution within the robot, ensuring optimal performance during cleaning operations , we used two in this project so it can power the Arduino and esp safely from the battery .

Figure 4.9: Puck Converter

4.7 Additional Components

4.7.1 Water Level Sensor

The water level sensor helps monitor the water tank's status, to help the first system to notify the second of needing refilling .

4.7.2 12V9A Battery

The 12V battery serves as the power source for the robot, ensuring it has sufficient energy for cleaning operations.

Figure 4.10: Battery

4.7.3 Suction Cups

Suction cups along side the belt of provide more elastic and a wider surface to increase friction and to navigate the glass surface of the solar panel safely without slipping.

Figure 4.11: Suction Cups

4.7.4 Wooden Wheels

Wooden wheels provide stability and traction to the robot while moving on the solar panels.

4.7.5 H-Bridge Motor Driver

The H-bridge motor driver is crucial for controlling the direction and speed of the DC motors for movment, brush and water pump.

4.8 how the system works

this a flow chart of how the system really boots

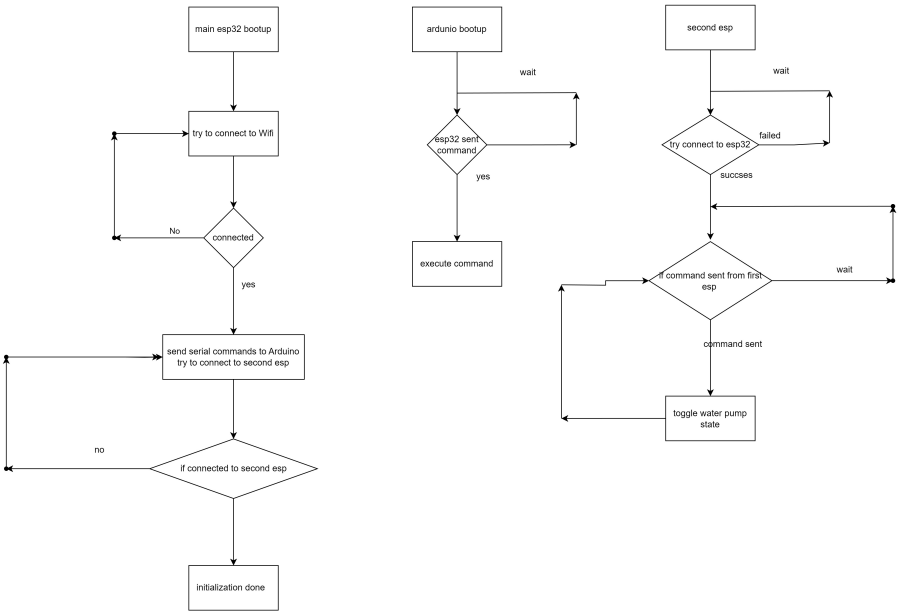

Figure 4.12: booting of the system

after booting up the system the Arduino waits for commands from the esp32 and the esp32 tries to connect to the WiFi till successful, when connected the esp will act as a server and get commands from the HTML page that is shown below

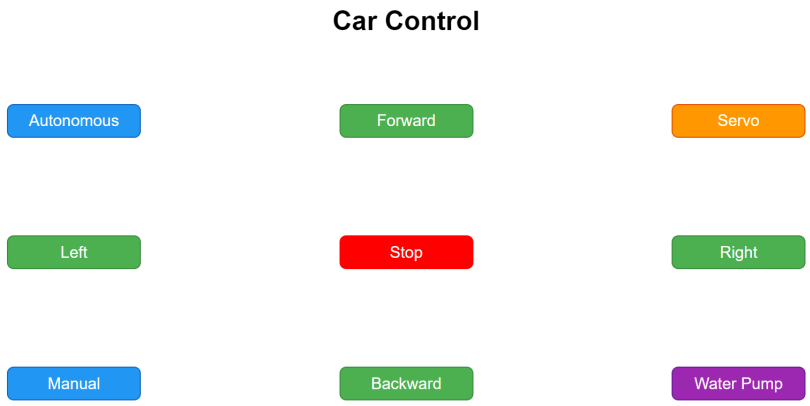

Figure 4.13: HTML page

when the html page presses Automate
the process is called according to this State Digram

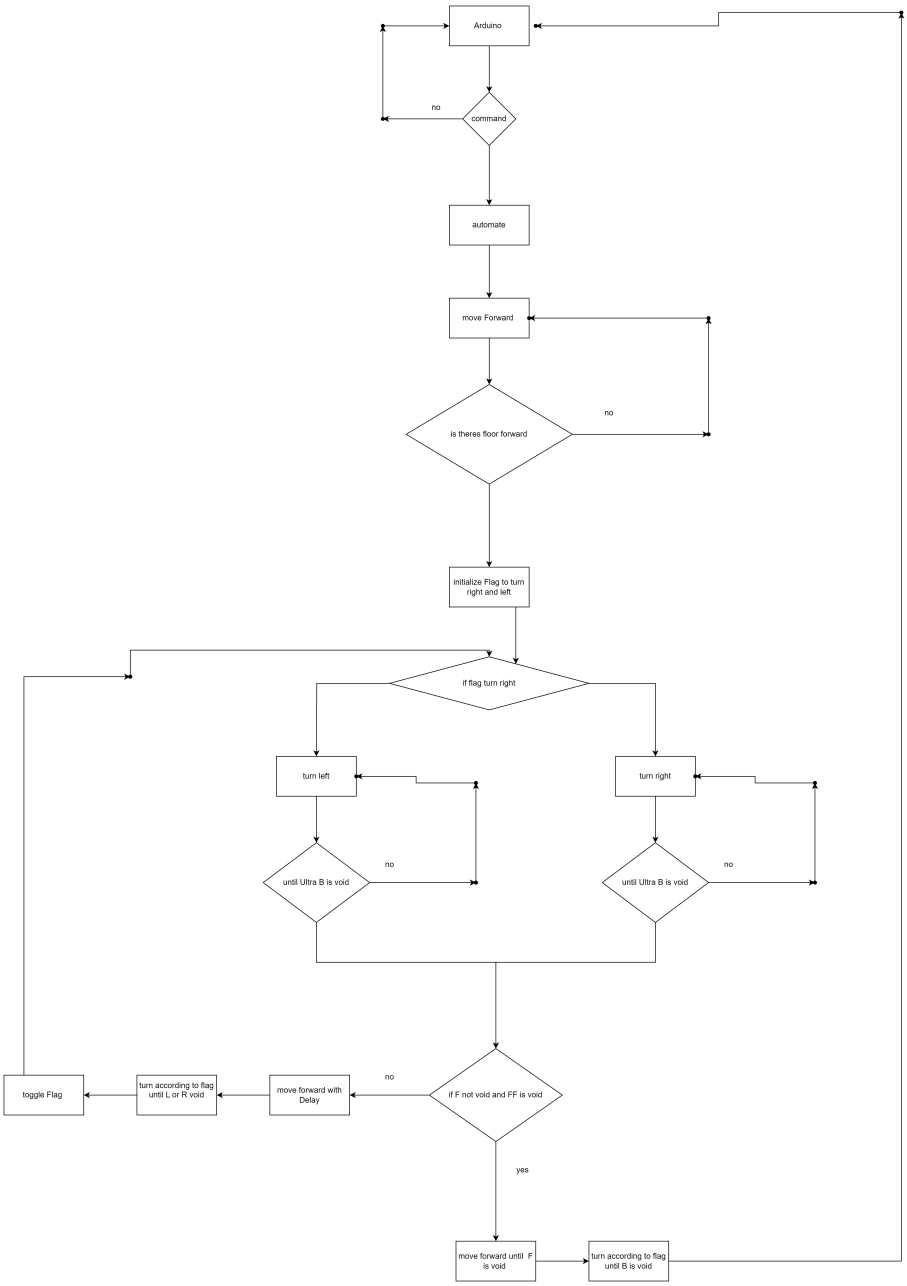

Figure 4.14: State Digram

4.9 Final Design

After several iterations and refinements, we arrived at the final design of the solar panel cleaning robot .

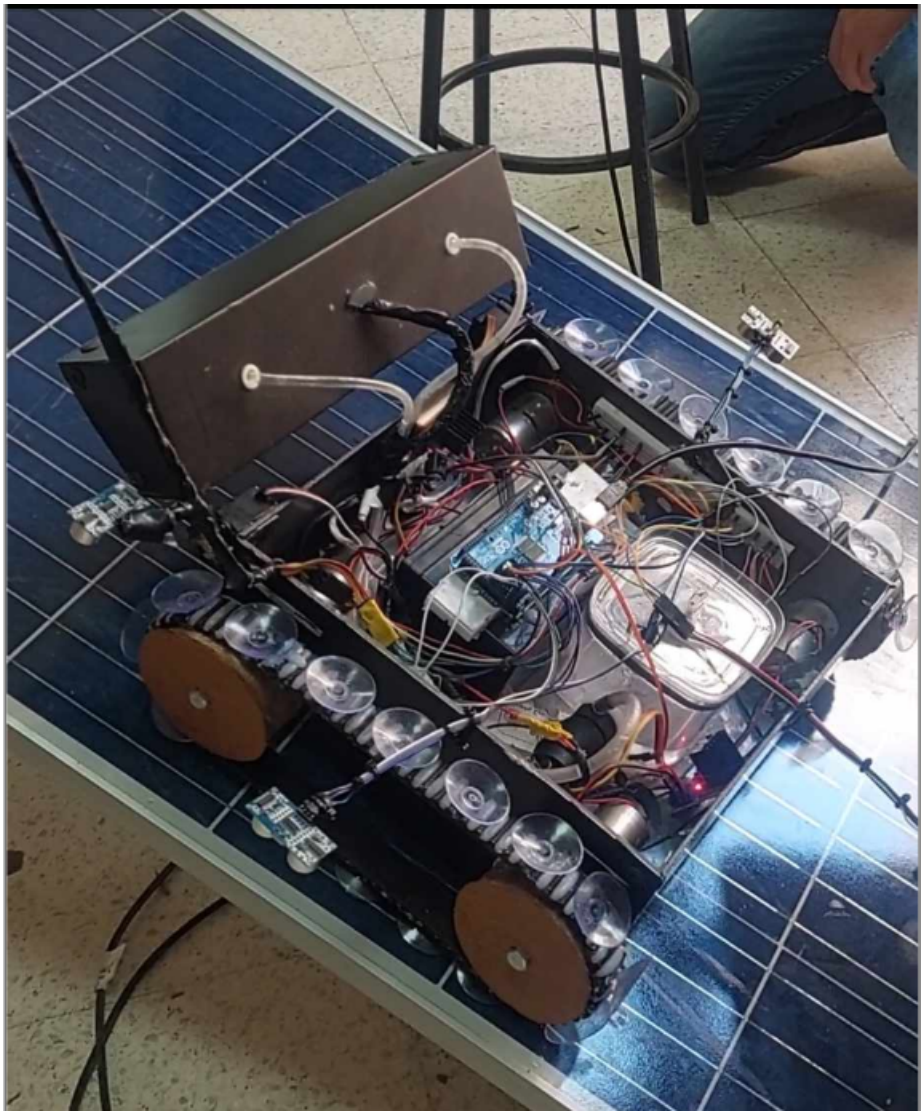

Figure 4.15: Final Design

Chapter 5

Results and Discussion

5.1 Results

The outcomes of our experiments, while promising for a prototype, reveal several areas for potential optimization. It is evident that employing suction cups as a means of adhering to glass or solar panels is a cost-effective solution. However, there is room for improvement by exploring alternative methods, such as incorporating an air pump and spider-like mechanisms for enhanced mobility and efficiency.

5.2 Discussion

While our solar panel cleaning robot is still in the production phase, we firmly believe that we've achieved a significant breakthrough in the field of solar panel maintenance. Our innovative solution has the potential to revolutionize the solar panel cleaning industry, offering an efficient and cost-effective alternative to manual labor. By harnessing the power of automation, we contribute to a brighter future powered by sustainable and renewable energy sources.

Chapter 6

Conclusion And future work

6.1 Conclusion

In conclusion, while our product is not yet tuned to perfection, it represents a valuable contribution to the engineering world. It serves as an important step towards creating the next generation of commercial products that can be even more refined and efficient in the field of solar panel cleaning.

6.2 Future Work

Some of the potential areas for enhancement include:

- **Optimization:** Further refining the robot's design and cleaning mechanisms to increase efficiency and coverage.
- **Mobility:** Exploring advanced mobility solutions, such as spider-like or air-pump-based mechanisms, to enhance the robot's ability to navigate different types of solar panels with different angels.
- **image processing and AI:** adding image processing would be an improvement and giving the robot more adaptive field of "vision".

Chapter 7

Listing 7.1: Arduino Code for Solar Panel Cleaner

```
#include <Servo.h>


Servo servoMotor1;
Servo servoMotor2;

const int trigPin = 30;
const int echoPin = 31;
const int trigPin2 = 32;
const int echoPin2 = 33;


const int trigPin3 = 34;
const int echoPin3 = 35;
const int trigPin4 = 36;
const int echoPin4 = 37;

const int trigPin5 = 38;
const int echoPin5 = 39;



const int waterPumbEnablePin = 2;
const int brushMotorEnablePin = 3;

const int waterPumbForwardPin = 4;
const int waterPumbBackwardPin = 5;
const int brushMotorForwardPin = 6;
const int brushMotorBackwardPin = 7;

const int servoPin1 = 8;
const int servoPin2 = 9;
```

```
const int motor1ForwardPin = 24;
const int motor1BackwardPin = 25;
const int motor2ForwardPin = 26;
const int motor2BackwardPin = 27;

int globalFlag = 1;
// void is zero for ultrasonic
int UltraF = 2; // trigPin
int UltraFF = 2; // trigPin2
int UltraL = 2; //trigPin3
int UltraR = 2;// trigPin4
int UltraB =2; // trigPin5

// left is 1 and right is 2
int LeftRightFlag=0;




int pumb = 0;
int servoState = 0;
int firstTime = 0 ;
void setup() {

  pinMode(motor1ForwardPin, OUTPUT);
  pinMode(motor1BackwardPin, OUTPUT);
  pinMode(motor2ForwardPin, OUTPUT);
  pinMode(motor2BackwardPin, OUTPUT);
  pinMode(waterPumbEnablePin, OUTPUT);
  pinMode(brushMotorEnablePin, OUTPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);
  pinMode(trigPin3, OUTPUT);
  pinMode(echoPin3, INPUT);
  pinMode(trigPin4, OUTPUT);
  pinMode(echoPin4, INPUT);
  pinMode(trigPin5, OUTPUT);
  pinMode(echoPin5, INPUT);
  servoMotor1.attach(servoPin1);
  servoMotor2.attach(servoPin2);
  stopMotors();
  Serial.begin(115200);
  stopMotors();
}
```

```
void loop() {
  if (Serial.available()) {
    String command = Serial.readStringUntil('\n');
    command.trim();
    if (command == "automate") {
      globalFlag = 0;
    } else if (command == "manual") {
      globalFlag = 1;
    }
    if (globalFlag == 1) {
      executeCommand(command);
    }
  }
  if (globalFlag == 0) {

    ultraSonic();
    autonomus();
  }
}



void TestOverrideFunction(){
   if (Serial.available()) {
    String command = Serial.readStringUntil('\n');
    command.trim();
    if (command == "automate") {
      globalFlag = 0;
    } else if (command == "manual") {
      globalFlag = 1;
    }
  }

}
void ultraSonic() {

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  long duration = pulseIn(echoPin, HIGH);
  int distance_cm = duration * 34300L / 2 / 1000000L;

  digitalWrite(trigPin2, LOW);
```

```
delayMicroseconds(2);
digitalWrite(trigPin2, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin2, LOW);
long duration2 = pulseIn(echoPin2, HIGH);
int distance_cm2 = duration2 * 34300L / 2 / 1000000L;

 digitalWrite(trigPin3, LOW);
delayMicroseconds(2);
digitalWrite(trigPin3, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin3, LOW);
long duration3 = pulseIn(echoPin3, HIGH);
int distance_cm3 = duration3 * 34300L / 2 / 1000000L;

  digitalWrite(trigPin4, LOW);
delayMicroseconds(2);
digitalWrite(trigPin4, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin4, LOW);
long duration4 = pulseIn(echoPin4, HIGH);
int distance_cm4 = duration4 * 34300L / 2 / 1000000L;

  digitalWrite(trigPin5, LOW);
delayMicroseconds(2);
digitalWrite(trigPin5, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin5, LOW);
long duration5 = pulseIn(echoPin5, HIGH);
int distance_cm5 = duration5 * 34300L / 2 / 1000000L;


if (distance_cm > 20) {
  UltraF = 0;
} else {
  UltraF = 1;
}
if (distance_cm2 > 40) {
  UltraFF = 0;
} else {
  UltraFF = 1;
}

 if (distance_cm3 > 20) {
  UltraL = 0;
} else {
```

```
    UltraL = 1;
  }

  if (distance_cm4 > 20) {
    UltraR = 0;
  } else {
    UltraR = 1;
  }

if (distance_cm5 > 20) {
    UltraB = 0;
  } else {
    UltraB = 1;
  }

}



void ultraSonicSeiral() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  long duration = pulseIn(echoPin, HIGH);
  int distance_cm = duration * 34300L / 2 / 1000000L;

  digitalWrite(trigPin2, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin2, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin2, LOW);
  long duration2 = pulseIn(echoPin2, HIGH);
  int distance_cm2 = duration2 * 34300L / 2 / 1000000L;

   digitalWrite(trigPin3, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin3, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin3, LOW);
  long duration3 = pulseIn(echoPin3, HIGH);
  int distance_cm3 = duration3 * 34300L / 2 / 1000000L;

    digitalWrite(trigPin4, LOW);
  delayMicroseconds(2);
```

```
  digitalWrite(trigPin4, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin4, LOW);
  long duration4 = pulseIn(echoPin4, HIGH);
  int distance_cm4 = duration4 * 34300L / 2 / 1000000L;

    digitalWrite(trigPin5, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin5, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin5, LOW);
  long duration5 = pulseIn(echoPin5, HIGH);
  int distance_cm5 = duration5 * 34300L / 2 / 1000000L;


  if (distance_cm > 20) {
    UltraF = 0;
  } else {
    UltraF = 1;
  }
  if (distance_cm2 > 40) {
    UltraFF = 0;
  } else {
    UltraFF = 1;
  }

   if (distance_cm3 > 20) {
    UltraL = 0;
  } else {
    UltraL = 1;
  }

  if (distance_cm4 > 20) {
    UltraR = 0;
  } else {
    UltraR = 1;
  }

if (distance_cm5 > 20) {
    UltraB = 0;
  } else {
    UltraB = 1;
  }

  Serial.print("ultraSonicSeiral");
  Serial.print(UltraF);
```

```
  Serial.print(" ");
  Serial.print(UltraFF);
   Serial.print(" ");
  Serial.print(UltraL);
     Serial.print(" ");
  Serial.print(UltraR);
  Serial.print(" ");
  Serial.print(UltraB);
  Serial.println("");

}

void autonomus() {
  moveForwardAutonomus();


  //code for first time
  if (LeftRightFlag ==0 ){
    ultraSonic();
    if (UltraL==0 && UltraR ==1){
      LeftRightFlag =1;
    }
    else if (UltraL==1 && UltraR ==0) LeftRightFlag =2;
  }

  if (firstTime ==0){
    if (LeftRightFlag == 2){
    TurnLeftAutonomusFirstTime();
    ultraSonic();

    if (UltraF ==0 && UltraFF != 0 )moveForwardAutonomus();
    else moveForwardAutonomusWithDelay();
    TurnLeftAutonomusSecondTime();
  }
    else if (LeftRightFlag == 1){
    TurnRightAutonomusFirstTime();
    ultraSonic();
    if (UltraF ==0 && UltraFF != 0 )moveForwardAutonomus();
    else moveForwardAutonomusWithDelay();
    TurnRightAutonomusSecondTime();
  }
    firstTime=1;

  }
  else{
  if (LeftRightFlag == 2){
```

```
    TurnLeftAutonomus();
    ultraSonic();
    if (UltraF ==0  && UltraFF != 0 )moveForwardAutonomus();
    else moveForwardAutonomusWithDelay();
    TurnLeftAutonomusSecondTime();
  }

  else if (LeftRightFlag == 1){
    TurnRightAutonomus();
    ultraSonic();
    if (UltraF ==0  && UltraFF != 0 )moveForwardAutonomus();
    else moveForwardAutonomusWithDelay();
    TurnRightAutonomusSecondTime();
  }

   ultraSonic();
  }
}

void TurnLeftAutonomusSecondTime(){
   Serial.println("TurnLeftAutonomusSecondTime");
  ultraSonic();
  while (UltraB !=0 ){
     TestOverrideFunction();
  if (globalFlag ==1)return;
ultraSonic();
turnLeft();
}
DelayForTime();

}


void TurnRightAutonomusSecondTime(){
  Serial.println("TurnRightAutonomusSecondTime");
  ultraSonic();
  while (UltraB !=0 ){
     TestOverrideFunction();
  if (globalFlag ==1)return;
ultraSonic();
turnRight();
}
DelayForTime();

}
```

```
void moveForwardAutonomus(){
  Serial.println("moveForwardAutonomus");
  ultraSonic();
  while (UltraF ==1 ){
    TestOverrideFunction();
    if (globalFlag ==1)return;


    BrushAndPumbOn();
    ServoDown();
    moveForward();
    ultraSonic();
  }
  StopAutonomus();

}

void moveForwardAutonomusWithDelay(){
   Serial.println("moveForwardAutonomusWithDelay");
   TestOverrideFunction();
  if (globalFlag ==1)return;
 BrushAndPumbOn();
    ServoDown();
    moveForward();
    delay(4000);
}

void StopAutonomus(){
  BrushAndPumbOff();
    ServoUp();
    stopMotors();
     TestOverrideFunction();
  if (globalFlag ==1)return;
}

void TurnLeftAutonomus(){
  Serial.println("TurnLeftAutonomus");
  ultraSonic();
while (UltraR !=0 ){
   TestOverrideFunction();
  if (globalFlag ==1)return;
ultraSonic();
turnLeft();
}
DelayForTime();
LeftRightFlag=1;
```

```
}

void TurnRightAutonomus(){
  Serial.println("TurnRightAutonomus");
  ultraSonic();
while (UltraL !=0 ){
   TestOverrideFunction();
  if (globalFlag ==1)return;
ultraSonic();
turnRight();
}
DelayForTime();
LeftRightFlag=2;


}


void TurnLeftAutonomusFirstTime(){
  Serial.println("TurnLeftAutonomusFirstTime");
ultraSonic();
while (UltraB !=0 ){
   TestOverrideFunction();
  if (globalFlag ==1)return;
ultraSonic();
turnLeft();
}
DelayForTime();
LeftRightFlag=1;
}

void TurnRightAutonomusFirstTime(){
  Serial.println("TurnRightAutonomusFirstTime");
ultraSonic();
while (UltraB !=0 ){

   TestOverrideFunction();
  if (globalFlag ==1)return;
ultraSonic();
turnRight();
}
DelayForTime();

LeftRightFlag=2;
```

```
}


void executeCommand(const String &command) {
  if (command == "stop") {
    BrushAndPumbOff();
    ServoUp();
    stopMotors();
  } else if (command == "forward") {
    moveForward();
    delay(1000);
  } else if (command == "backward") {
    moveBackward();
    delay(1000);
  } else if (command == "left") {
    turnLeft();
    delay(1000);
  } else if (command == "right") {
    turnRight();
    delay(1000);
  }else if (command == "stop") {
    stopMotors();
    delay(1000);
  }else if (command == "ultraSonicSeiral") {
    ultraSonicSeiral();
    delay(1000);
  }else if (command == "pump") {
    pumb = !pumb;
    if (pumb == 0) {
      BrushAndPumbOff();
    } else {
      BrushAndPumbOn();
    }
    delay(500);
  } else if (command == "servo") {
    servoState = !servoState;
    if (servoState == 0) {
      ServoDown();
    } else {
      ServoUp();
    }
    delay(500);
  }
}

void ServoDown() {
```

```
  servoMotor1.write(130);
  servoMotor2.write(30);
}

void ServoUp() {
  servoMotor1.write(30);
  servoMotor2.write(130);
}

void BrushAndPumbOn() {
  digitalWrite(waterPumbForwardPin, HIGH);
  digitalWrite(waterPumbBackwardPin, LOW);
  digitalWrite(brushMotorForwardPin, HIGH);
  digitalWrite(brushMotorBackwardPin, LOW);




  analogWrite(waterPumbEnablePin,150);
  analogWrite(brushMotorEnablePin, 255);
}

void BrushAndPumbOff() {
  digitalWrite(waterPumbForwardPin, LOW);
  digitalWrite(waterPumbBackwardPin, LOW);
  digitalWrite(brushMotorForwardPin, LOW);
  digitalWrite(brushMotorBackwardPin, LOW);

  analogWrite(waterPumbEnablePin, 0);
  analogWrite(brushMotorEnablePin, 0);
}

void moveBackward() {
  digitalWrite(motor1ForwardPin, LOW);
  digitalWrite(motor1BackwardPin, HIGH);
  digitalWrite(motor2ForwardPin, LOW);
  digitalWrite(motor2BackwardPin, HIGH);
}

void turnRight() {
   digitalWrite(motor1ForwardPin, HIGH);
  digitalWrite(motor1BackwardPin, LOW);
  digitalWrite(motor2ForwardPin, LOW);
  digitalWrite(motor2BackwardPin, HIGH);
```

```
}

void turnLeft() {

  digitalWrite(motor1ForwardPin, LOW);
  digitalWrite(motor1BackwardPin, HIGH);
  digitalWrite(motor2ForwardPin, HIGH);
  digitalWrite(motor2BackwardPin, LOW);

}

void moveForward () {
  digitalWrite(motor1ForwardPin, HIGH);
  digitalWrite(motor1BackwardPin, LOW);
  digitalWrite(motor2ForwardPin, HIGH);
  digitalWrite(motor2BackwardPin, LOW);
}

void stopMotors() {
  digitalWrite(motor1ForwardPin, LOW);
  digitalWrite(motor1BackwardPin, LOW);
  digitalWrite(motor2ForwardPin, LOW);
  digitalWrite(motor2BackwardPin, LOW);
}

void DelayForTime(){
    delay(5500);
}
```