ARDUINO BASED SMART VACUUM CLEANER

ECS1001:ENGINEERING CLINIC'S

(ARDUINO USING EMBEDDED C)

FALL SEM 2022-23

VIT-AP UNIVERSITY

GUDIED BY:

PROF. ESWARAIAH RAYACHOTI

SUBMITTED BY:

20BCE7369 - YALAMARTHI KOUSHIK SAI – CSE
20BCE7347 - PAVULURI KARUN SAI – CSE
20BCE7331 - VAKA RAJASIMHA REDDY – CSE
20BCE7586 - LOKESH VALLURI – CSE
20BES7056 - KARNATI NAGA PRASANTH REDDY - ECE (SPEC. IN EMBEDDED SYSTEMS)
20BEV7029 - TIRUVEEDHULA CHANDRA MOULI - ECE (SPEC. IN VLSI)

AGENDA

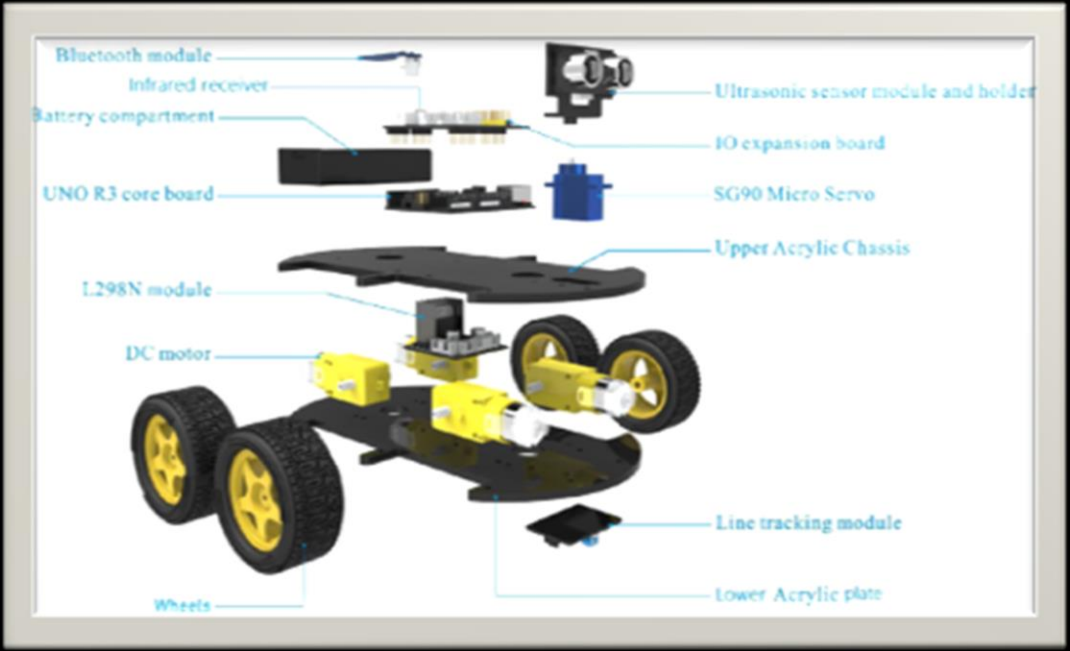
- Introduction
- Parts
- Problem Definition
- Circuit Diagram
- Codes in Appendix
- Obstacle-Avoidance Mode

This project is aims at developing Arduino based smart vacuum cleaner to provide a better solution to society.

In this modern digital world, everyone is moving towards automation Robotics allows automation where machines perform a well-defined step safely and productively, in autonomous or partial autonomous manners.

A number of vacuum cleaner bases are available for just such a project. These inexpensive bases are generally made of acrylic and come complete with a set of small DC motors.

PARTS



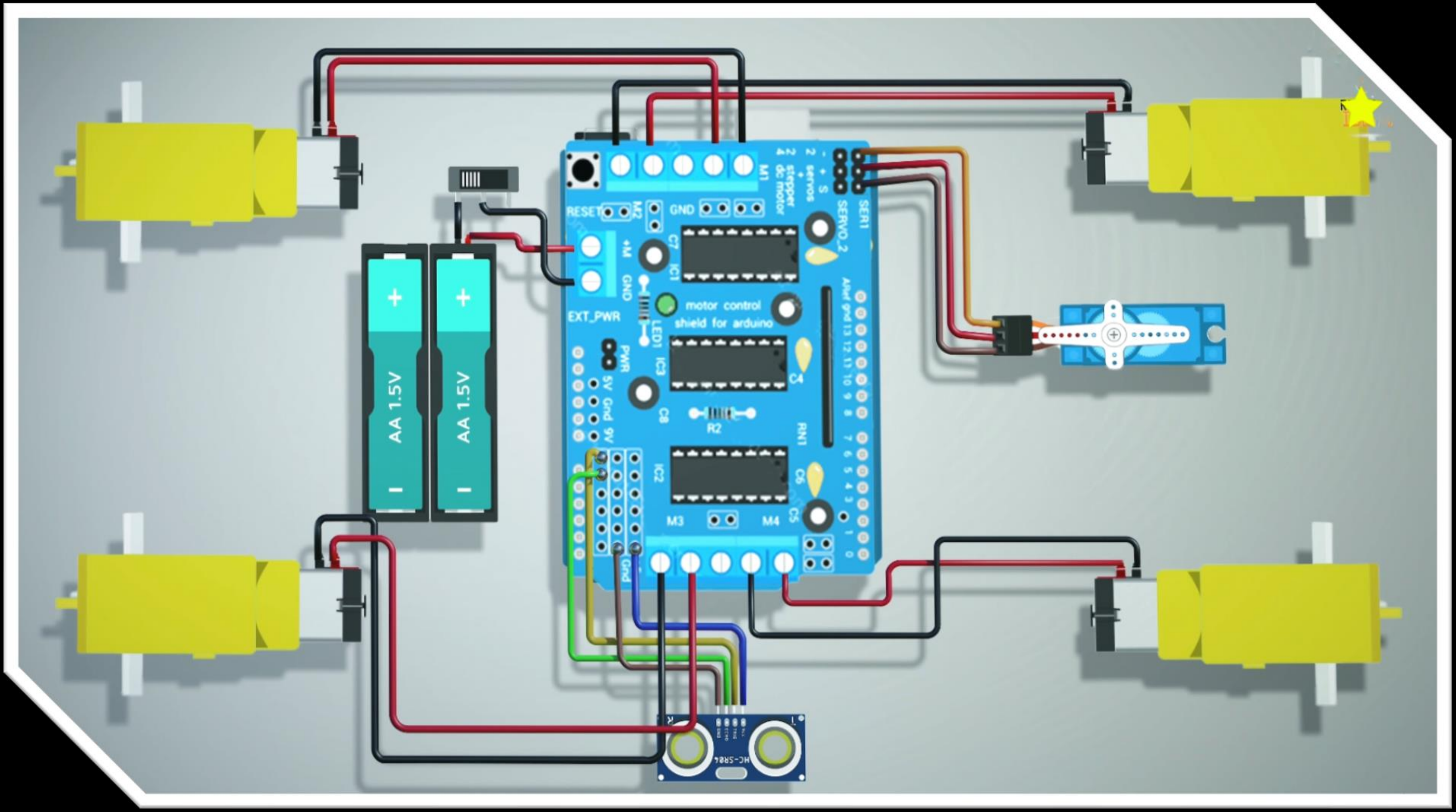
- ➢DC MOTORS WITH WHEELS
- ➢ARDUNIO UNO
- ➢L293D SHIELD DRIVER BOARD
- ➢SERVO MOTOR
- ➢ULTRASONIC SENSOR
- ➢LITHIUM-ION BATTERY
- ➢CONNECTING WIRES

Problem Definition

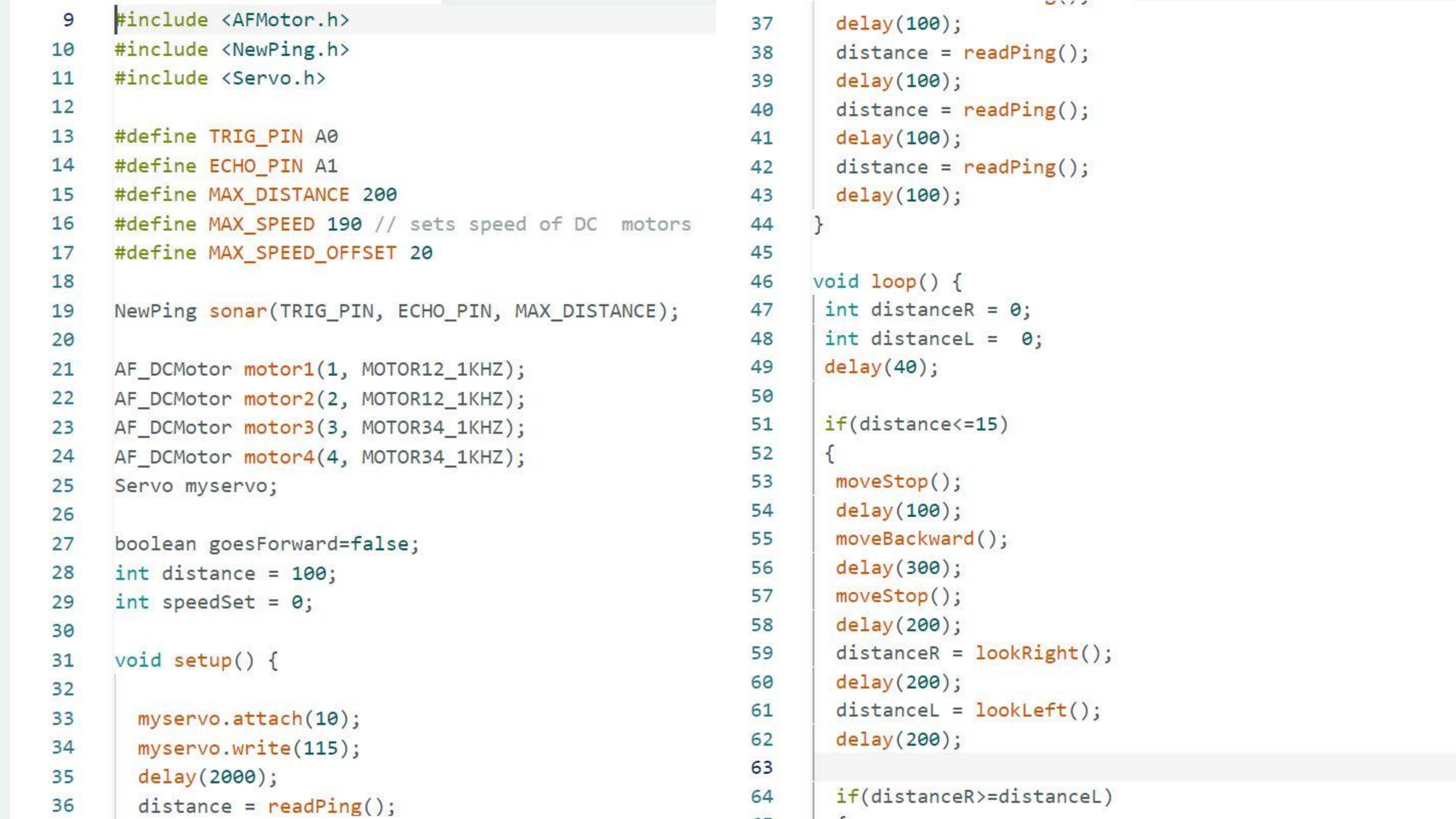
- IDEA:-
- • In a present-day scenario, we all are so busy with our work that we don't have the time for cleaning our house properly. The solution to the problem is very simple, you just need to buy a domestic vacuum cleaner
- • The purpose of this project is to clean a room floor from small debris and dust/dirt autonomously.

- WORKING AND IMPLIMENTATION:-
- • So today, we decided to make a simple Floor cleaner robot, which is not only simple to make but costs very less compared to commercial products available in the market .
- • The new Arduino Vacuum Cleaner we are going to build here will be compact and more practical.
- • On top of that, this robot will have ultrasonic sensors and an IR proximity sensor. The ultrasonic sensor will allow the robot to avoid obstacles so that it can move freely until the room is properly cleaned.

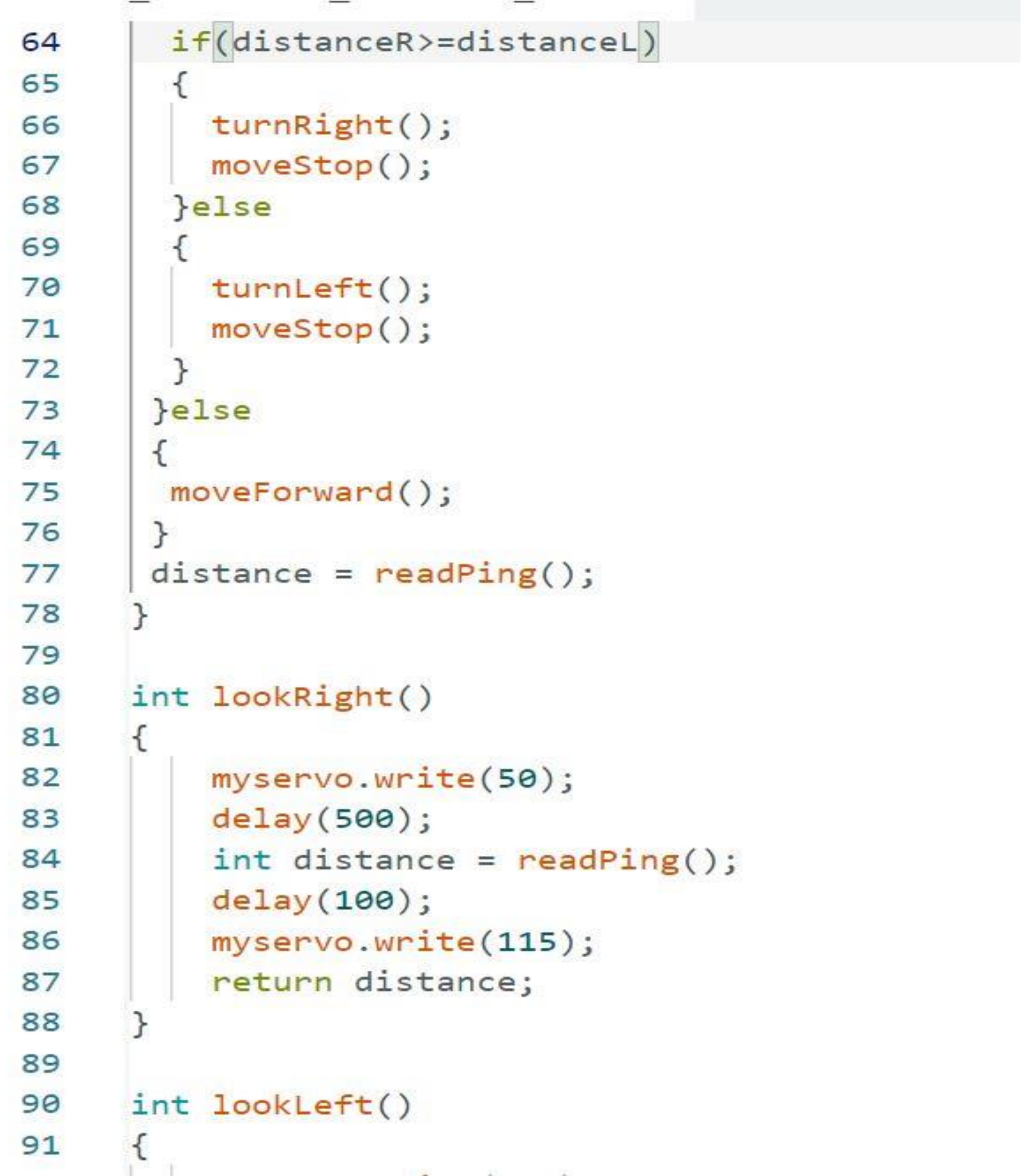
CIRCUIT DIAGRAM



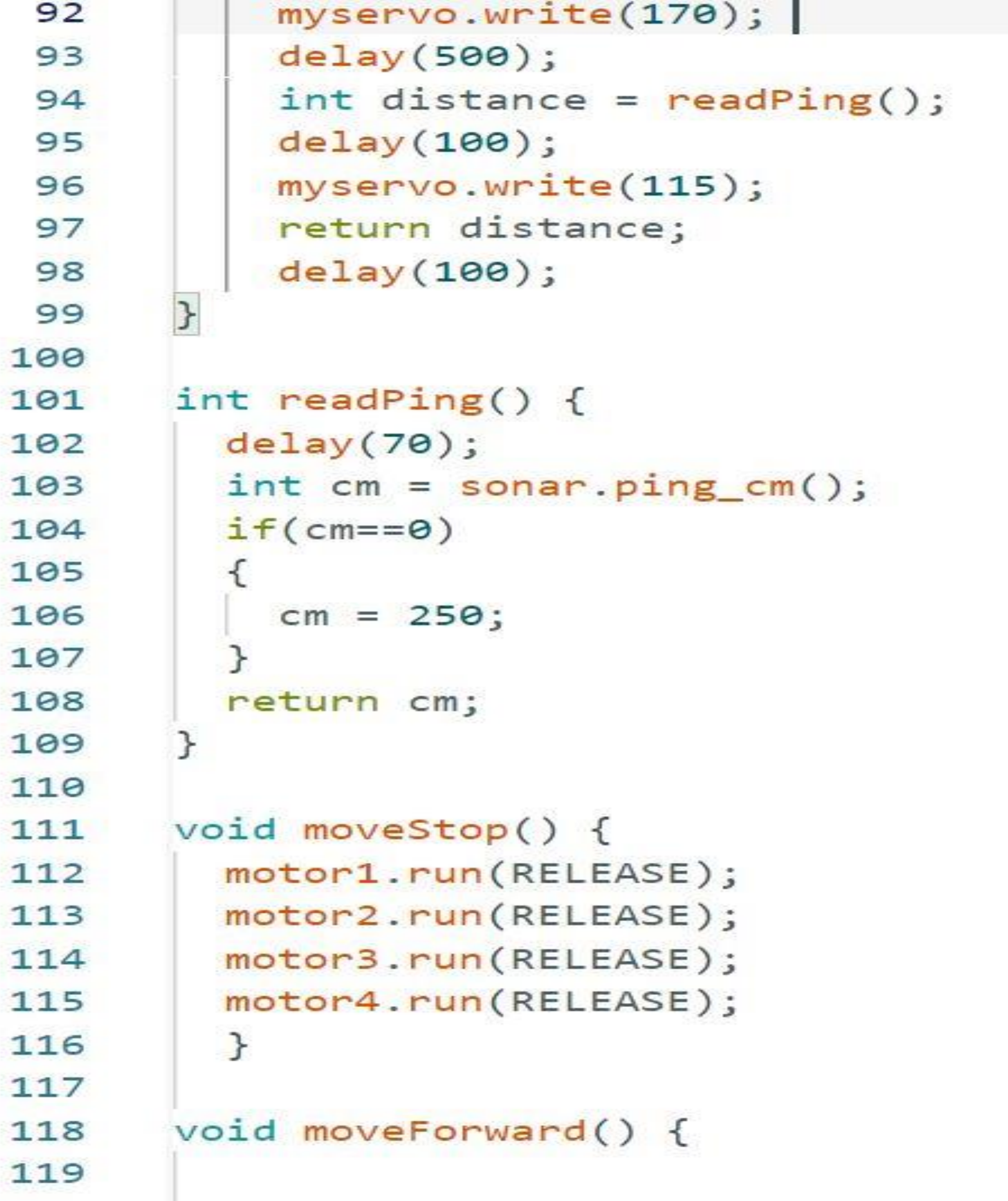
CODES IN APPENDIX



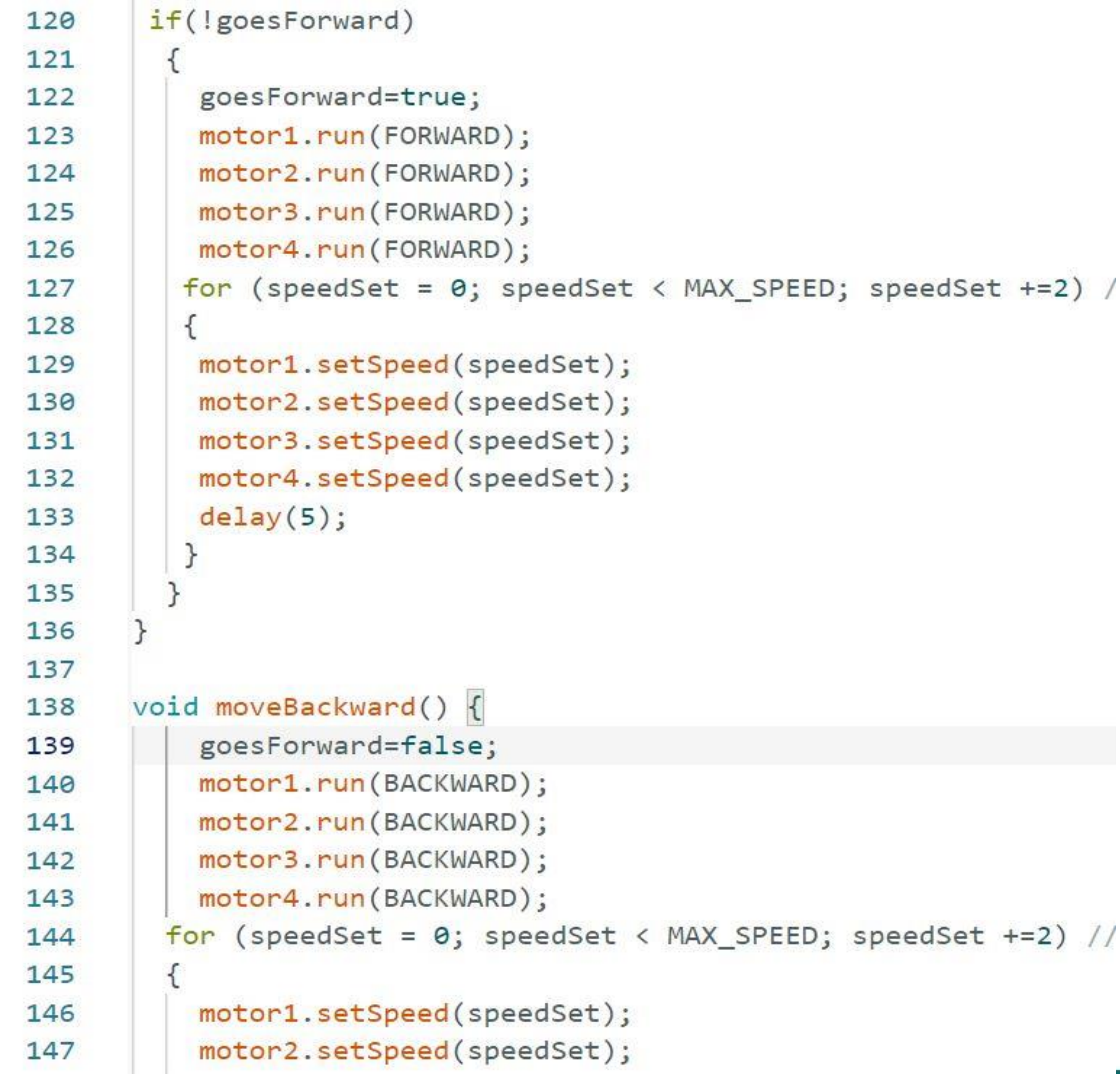
```
#include <AFMotor.h>
#include <NewPing.h>
#include <Servo.h>

#define TRIG_PIN A0
#define ECHO_PIN A1
#define MAX_DISTANCE 200
#define MAX_SPEED 190 // sets speed of DC  motors
#define MAX_SPEED_OFFSET 20

NewPing sonar(TRIG_PIN, ECHO_PIN, MAX_DISTANCE);

AF_DCMotor motor1(1, MOTOR12_1KHZ);
AF_DCMotor motor2(2, MOTOR12_1KHZ);
AF_DCMotor motor3(3, MOTOR34_1KHZ);
AF_DCMotor motor4(4, MOTOR34_1KHZ);
Servo myservo;

boolean goesForward=false;
int distance = 100;
int speedSet = 0;

void setup() {

  myservo.attach(10);
  myservo.write(115);
  delay(2000);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
}

void loop() {
 int distanceR = 0;
 int distanceL =  0;
 delay(40);

 if(distance<=15)
 {
  moveStop();
  delay(100);
  moveBackward();
  delay(300);
  moveStop();
  delay(200);
  distanceR = lookRight();
  delay(200);
  distanceL = lookLeft();
  delay(200);

  if(distanceR>=distanceL)
```



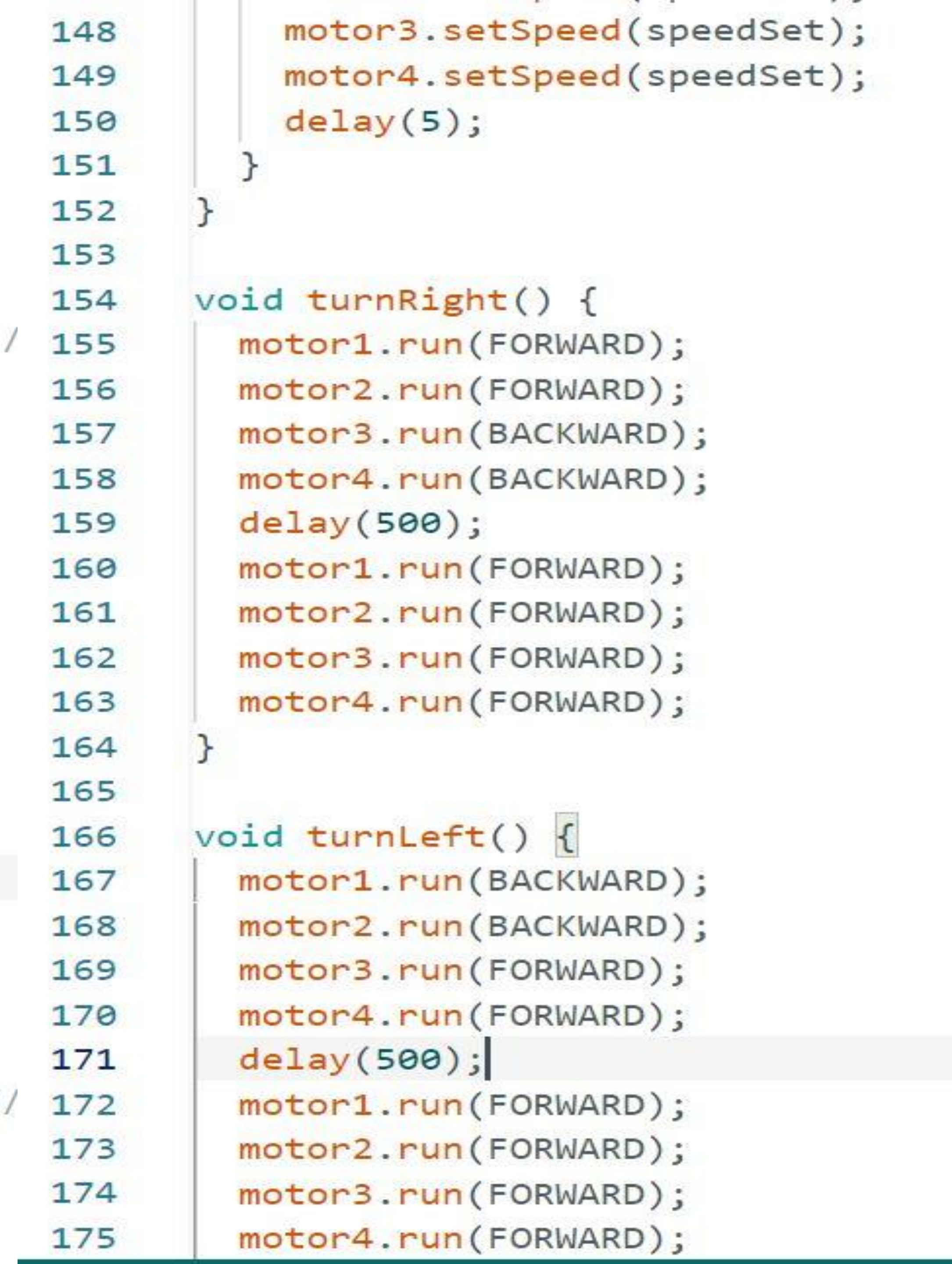
```
  if(distanceR>=distanceL)
  {
    turnRight();
    moveStop();
  }else
  {
    turnLeft();
    moveStop();
  }
 }else
 {
  moveForward();
 }
 distance = readPing();
}

int lookRight()
{
    myservo.write(50);
    delay(500);
    int distance = readPing();
    delay(100);
    myservo.write(115);
    return distance;
}

int lookLeft()
{
```



```
    myservo.write(170);
    delay(500);
    int distance = readPing();
    delay(100);
    myservo.write(115);
    return distance;
    delay(100);
}

int readPing() {
  delay(70);
  int cm = sonar.ping_cm();
  if(cm==0)
  {
    cm = 250;
  }
  return cm;
}

void moveStop() {
  motor1.run(RELEASE);
  motor2.run(RELEASE);
  motor3.run(RELEASE);
  motor4.run(RELEASE);
  }

void moveForward() {

```



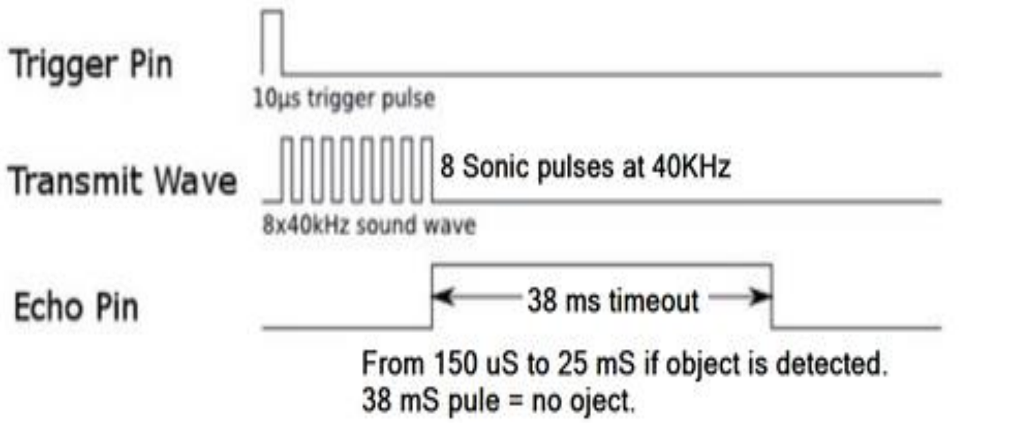
```
  if(!goesForward)
   {
     goesForward=true;
     motor1.run(FORWARD);
     motor2.run(FORWARD);
     motor3.run(FORWARD);
     motor4.run(FORWARD);
    for (speedSet = 0; speedSet < MAX_SPEED; speedSet +=2) /
    {
     motor1.setSpeed(speedSet);
     motor2.setSpeed(speedSet);
     motor3.setSpeed(speedSet);
     motor4.setSpeed(speedSet);
     delay(5);
    }
   }
}

void moveBackward() {
    goesForward=false;
    motor1.run(BACKWARD);
    motor2.run(BACKWARD);
    motor3.run(BACKWARD);
    motor4.run(BACKWARD);
   for (speedSet = 0; speedSet < MAX_SPEED; speedSet +=2) //
   {
    motor1.setSpeed(speedSet);
    motor2.setSpeed(speedSet);
```



```
    motor3.setSpeed(speedSet);
    motor4.setSpeed(speedSet);
    delay(5);
  }
}

void turnRight() {
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);
  delay(500);
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
}

void turnLeft() {
  motor1.run(BACKWARD);
  motor2.run(BACKWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
  delay(500);
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
```

Obstacle-Avoidance Mode

The ultrasonic sensor has four pins:
VCC – This is the 5-volt positive power connection.
TRIG – This is the “Trigger”, an input for the pulse we will be sending from the ultrasonic transmitter.
ECHO – This is an output that sends back the received pulse.
GND – The Ground connection.



To obtain the distance, measure the width (Ton) of Echo pin.
Time = Width of Echo pulse, in uS (micro second)
- Distance in centimeters = Time / 58
- Distance in inches = Time / 148
- Or you can utilize the speed of sound, which is 340m/s
The ultrasonic module can be tested using Arduino microcontroller and the following sketch function.

```
int ultrasonic_test(){
  digitalWrite(Trig, LOW);
  delayMicroseconds(2);
  digitalWrite(Trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(Trig, LOW);
  float distance = pulseIn(Echo, HIGH);
  distance = distance / 58;
}
```

Obstacle Avoidance – Operation

The principle of obstacle or collision avoidance is as simple as "if – else if – else" statement in C++ or any other programing languages. The ultrasonic sensor module will detect the distance between the car and an obstacle in front of it and sending the data to the microcontroller. Then, the microcontroller sends a corrective action to the smart car and this process continues repeatedly.

The algorithm follows the following sequence.Ultrasonic sensor measures the distance to the nearest object until obstacle detected.

- Stop the car.
- Measure the distance to the right and left of the Smart Car.
- Turn the car in the direction that you measure the longest distance.
- Move forward.
- Upload the downloaded Arduino sketch, "Obstacle_Avoidance_Car.ino" to your Smart Car.
- Disconnect the programming cable.
- Reinstall the Bluetooth module.
- Position the Smart Car on a flat surface, turn the power switch to ON
- Open the "BLE Tool" App to your smartphone, if you haven't done it on the previous section.
- Pair the Smart Car Bluetooth with the App.
- Inside the "Rocker Control Panel" of the App, select obstacle avoidance and enjoy the autonomous navigation in action.

THANK YOU