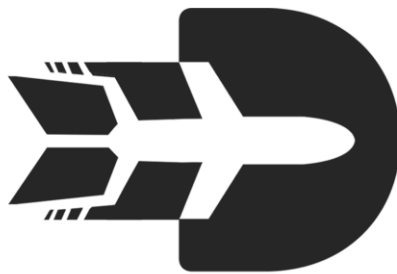


# Pursuitron Documentation

---



## THE 3<sup>RD</sup> DIMENSION

AEROMODELLING CLUB OF NITT

## Table of Contents-

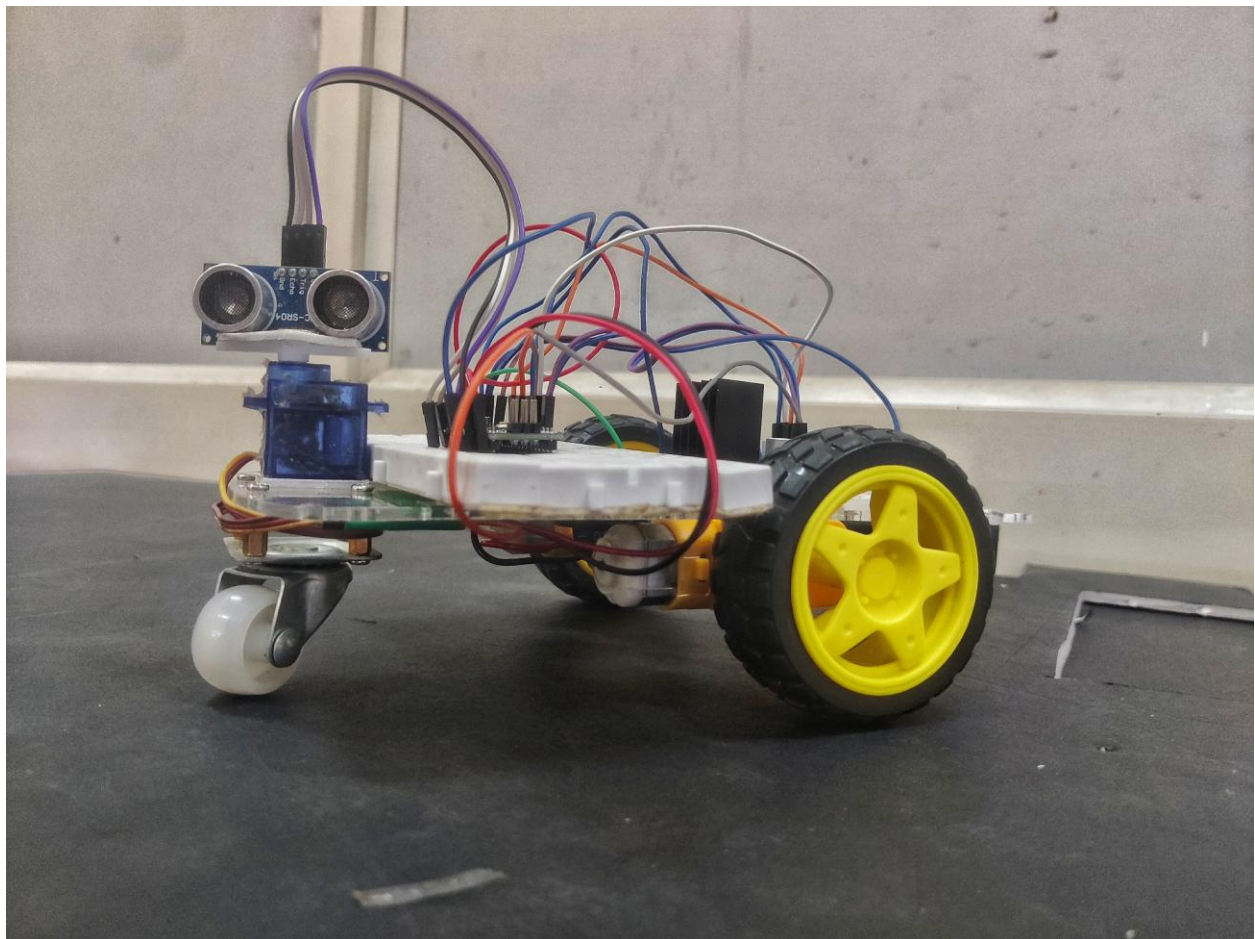
1. Objective
2. Components list and Specification
3. About the Components
4. General Architecture
5. Links Repositories
6. Circuit Diagram
7. Procedure
8. Code
9. Working of the Bot
10. Conclusion
11. Safety Guidelines
12. Doubts and Assistance

## Objective-

This documentation report outlines the construction and assembly process of an Arduino-based robot titled “Pursuitron” by The Third Dimension Aeromodelling Club of NITT, designed for object detection and pursuit missions as part of the Aeroatrix Workshop ‘25. The robot utilizes ultrasonic sensors for accurate object detection, a servo for rotating the ultrasonic sensor, an Arduino Nano microcontroller for processing, a L298N driver for motor control, and various other components.

This report provides a detailed component list, theory of operation for Arduino and the ultrasonic sensors, step-by-step procedures for assembling the robot, and general safety guidelines. It encourages participants to seek volunteer assistance in case of doubts or difficulties.

## Pursuitron-



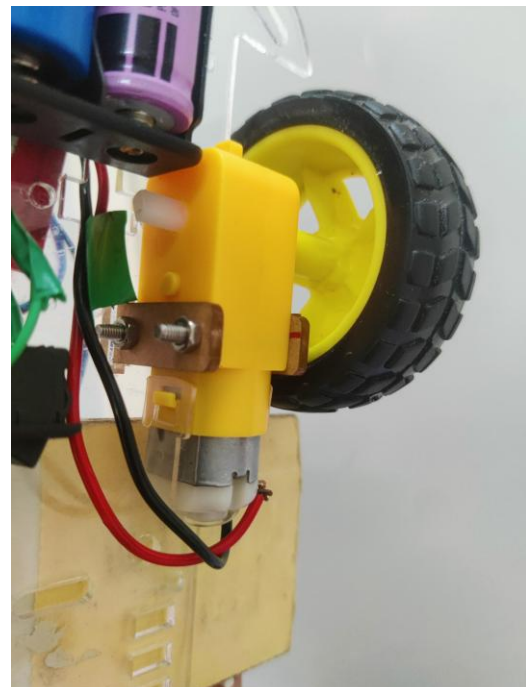
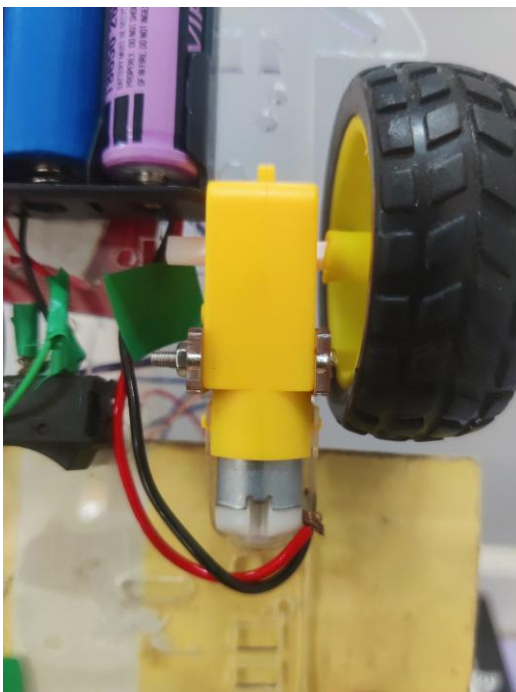
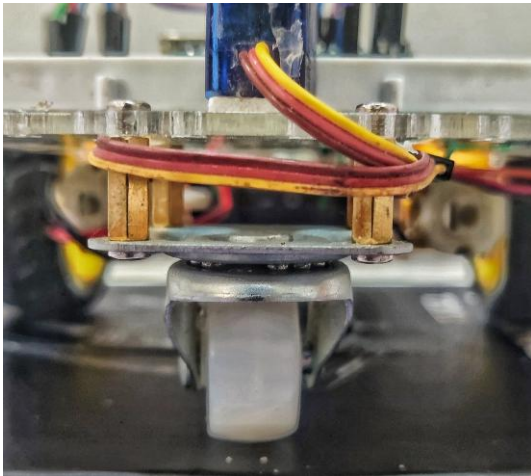
## Components List and Specification-

To build Pursuitron, the following components are required:

1.	Arduino Nano microcontroller board	-
2.	L298N motor driver	Max motor supply voltage/current- 46V/2A
3.	Ultrasonic sensors HC-SR04	-
4.	Micro Servo SG90	Torque- 1.2 kg-cm, 180-degree rotation, Vdc- 3.0~7.2V
5.	Transparent plastic chassis	-
6.	DC BO motors (2)	
7.	Wheels (2)	-
8.	Jumper wires (Male-Male, Female-Male)	-
9.	3.7V 18650 Li-ion Cells	
10.	Various mechanical fasteners	Nuts and Bolts
11.	USB cable for Arduino programming	-
12.	Tools	Screwdriver, plier, and wire stripper
13.	Castor Wheel	-

## Procedure-

- Start by gathering all the required components and ensuring their functionality.
- Begin assembling the robot by securing the DC motors to the chassis using screws and nuts.
- Attach the wheels to the motor shafts securely.
- Place the Arduino Nano microcontroller board on the chassis, ensuring the pins' correct orientation and alignment.



- Connect the L298N motor driver IC to the Arduino and make necessary connections between the motor driver and the Arduino Nano by ensuring correct pin alignment.
- Connect the ultrasonic sensors to the driver + controller unit and establish connections between the sensors and the Arduino Nano.
- Double-check all the connections and ensure there are no loose or short-circuited wires.
- Once the circuit connections are complete, upload the provided code to the Arduino Nano using the Arduino IDE and a USB cable.
- Use mechanical fasteners to securely mount the electronics (Arduino, motor driver, and ultrasonic sensors) onto the chassis.
- As shown, power the Arduino and motor driver by connecting the battery to the appropriate pins.
- Perform a final inspection to ensure everything is properly connected and secured.
- Test Pursuitron's functionality by powering it on and observing its response to different objects and obstacles.
- If the robot functions as expected, the assembly is complete. If not, review the connections, code, and troubleshooting as necessary.

## About the Components-

**Ultrasonic Sensors:** The HC-SR04 ultrasonic sensors emit high-frequency sound waves and measure the time it takes for the waves to bounce back after hitting an object. The robot can detect objects in its surroundings by calculating the distance based on the time taken.

**Arduino Nano:** The Arduino Nano is a microcontroller board based on the ATmega328P. The Arduino board is a specially designed circuit board for programming and prototyping with Atmel microcontrollers.

- It offers several advantages over other development boards, including its affordability, ease of setup, and use.
- The board has an easy USB interface, allowing it to connect directly to a computer's USB port and be recognized as a virtual serial port.
- This USB interface simplifies communication with the Arduino using the widely used and reliable serial communication protocol.
- Arduino provides convenient power management and built-in voltage regulation. It can be powered by an external power source or a USB port.

- The Arduino Nano uses the ATmega328 microcontroller, which is readily available and affordable.
- The ATmega328 chip offers various hardware features such as timers, PWM pins, interrupts, and sleep modes, making it versatile for various projects.
- With a 16MHz clock speed, the Arduino Nano is sufficiently fast for most applications.
- The Arduino Nano includes 20 digital and 8 analog pins, which can be used to connect external hardware and sensors, expanding its capabilities.
- The board features an ICSP connector, which allows direct interfacing with the microcontroller as a serial device without using the USB port.
- Overall, the Arduino board offers a user-friendly and versatile platform for programming and prototyping with microcontrollers, making it accessible to beginners and experienced users. Its features and affordability have contributed to its popularity and widespread adoption in the maker and electronics communities.

**Servo:** A servo motor is defined as an electric motor that allows for precise control of angular or linear position, speed, and torque. It consists of a suitable motor coupled to a sensor for position feedback and a controller that regulates the motor's movement according to a desired setpoint.

## General Architecture-

- The Arduino-based robot is designed to interact with its surroundings using an ultrasonic sensor and servo and perform specific actions based on detected objects. The robot has two powered wheels and two motors attached to the chassis for movement control.
- When an object, such as your hand, is brought near the ultrasonic sensor, the robot's detection mechanism is activated. The servo stops rotating and the ultrasonic sensor detects the object's presence and relays this information to the Arduino microcontroller.
- By observing the position of your hand, the Arduino interprets your gestures and generates corresponding instructions for the motor driver. For example, if you turn your hand to the left, the Arduino instructs the robot to move in the left direction. Similarly, if you move your hand to the right, the robot responds by moving in the right direction.
- To ensure accurate detection, the Arduino establishes a predefined distance threshold. If the object, such as your hand, is too far from the ultrasonic sensor, it

will not be registered. Conversely, when the object is near the sensor, it will be detected and registered by the Arduino.

- Once the Arduino recognizes the presence of an object in front of the sensor, it sends appropriate instructions to the motor driver. The motor driver, in turn, triggers the motors to initiate movement. In this case, the robot will start moving forward by activating the motors to rotate in the forward direction.
- It's important to note that the specific movements and actions of the robot can be further customized and expanded by modifying the Arduino code to include additional instructions and behaviors.
- By combining the ultrasonic sensor's object detection capabilities, the servo's rotational ability, the Arduino's decision-making process, and the motor driver's control over the robot's movement, the Arduino-based robot can effectively interact with its environment and respond to user gestures.

## Links Repositories-

Download link for Arduino IDE 1.8.19:

The screenshot shows the Arduino.cc website's software page for the Legacy IDE (1.8.X). The page features a teal header with navigation links: PROFESSIONAL, EDUCATION, STORE, and a search bar. Below the header, the 'SOFTWARE' tab is selected. The main content area displays the Arduino IDE 1.8.19 download options. On the left, there's a description of the IDE as open-source software for writing and uploading code to any Arduino board, with a link to the 'Getting Started' page for installation instructions. Below this, it mentions that active development is hosted on GitHub and provides links for building the code and verifying source code archives. On the right, a 'DOWNLOAD OPTIONS' section lists download links for Windows (Win 7 and newer, ZIP file), Windows app (Win 8.1 or 10), Linux (32 bits, 64 bits, ARM 32 bits, ARM 64 bits), and Mac OS X (10.10 or newer). A 'Help' button is visible in the bottom right corner.

Legacy IDE (1.8.X)

**Arduino IDE 1.8.19**

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the [Getting Started](#) page for Installation instructions.

**SOURCE CODE**

Active development of the Arduino software is [hosted by GitHub](#). See the instructions for [building the code](#). Latest release source code archives are available [here](#). The archives are PGP-signed so they can be verified using [this](#) gpg key.

**DOWNLOAD OPTIONS**

- Windows** Win 7 and newer
- Windows** ZIP file
- Windows app** Win 8.1 or 10 [Get](#)
- Linux** 32 bits
- Linux** 64 bits
- Linux** ARM 32 bits
- Linux** ARM 64 bits
- Mac OS X** 10.10 or newer

[Release Notes](#)

[Checksums \(sha512\)](#)

**Previous Releases**

Download the previous version of the current release, the classic 1.0.x, or old beta releases.

[Help](#)

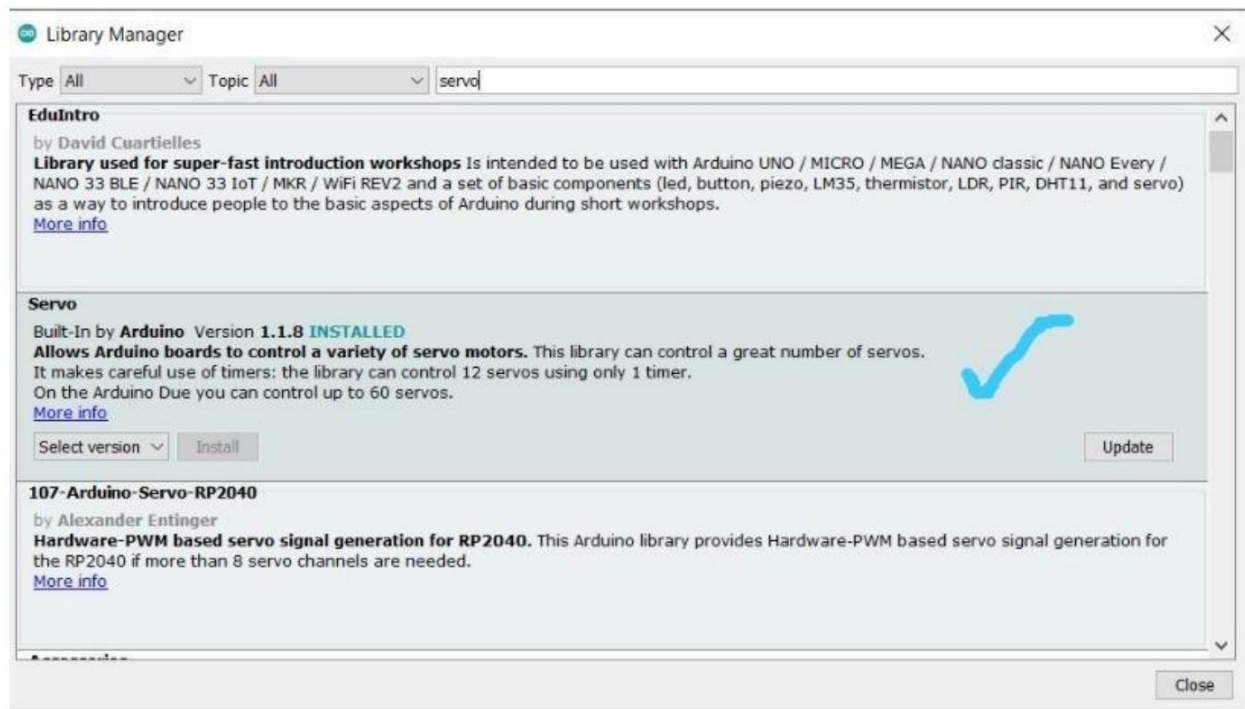
<https://www.arduino.cc/en/software>

For Driver issues (Manually Install Drivers on Windows), follow this guide

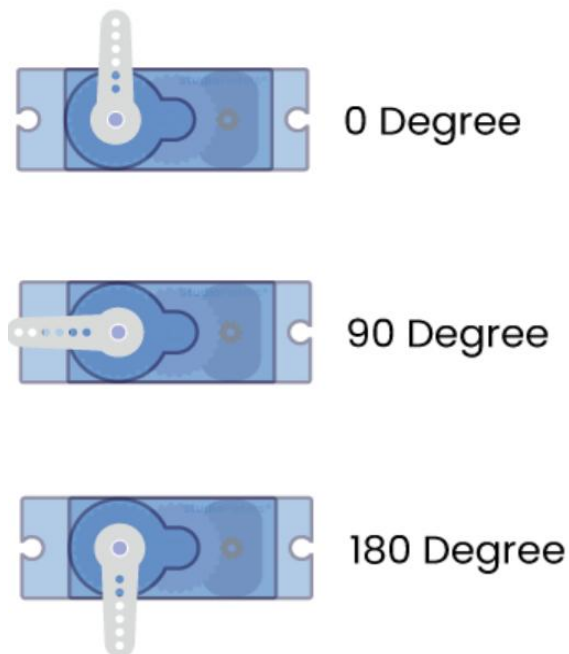
<https://docs.arduino.cc/tutorials/generic/DriverInstallation>



Servo library to be downloaded via Sketch > Include libraries> Manage libraries >



Before fixing the ultrasonic on the servo blades, We have to ensure that the initial servo blade is at 90 degrees like the figure shown below-



Initially, it can be at any position as shown below-



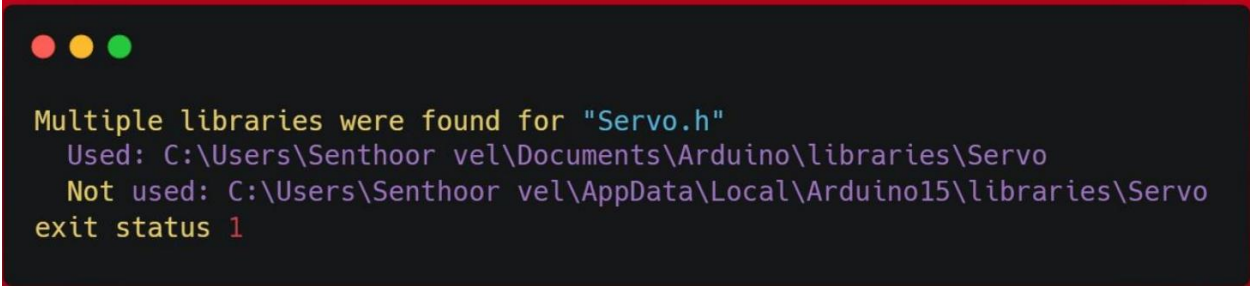
www.potoku.com

To bring the rotor to zero deg write the following code (below) to get the rotor (of servo) to zero position. After this mount the blade along the servo body and then mount the ultrasonic sensor.

### Code-

```
1  #include <Servo.h>
2
3  Servo servo; // creates a servo object to control a servo
4
5  void setup() {
6      servo.attach(10); //attaches the servo on pin 9 to the servo object
7  }
8
9  void loop() {
10     servo.write(0); //sets the servo position to zero degrees
11     delay(1000); //waits for the servo to get there
12 }
13
```

In case of issues like-



```
Multiple libraries were found for "Servo.h"
Used: C:\Users\Senthoo vel\Documents\Arduino\libraries\Servo
Not used: C:\Users\Senthoo vel\AppData\Local\Arduino15\libraries\Servo
exit status 1
```

**Solution:** Delete the servo library in Arduino 15\ libraries\ Servo.

**Reason for error:** Servo library installed in multiple paths, then Arduino gets confused on which one to use upon being called. In the above case, it used the Servo library from Documents and hence there is a clash. The solution is to just choose to keep one.

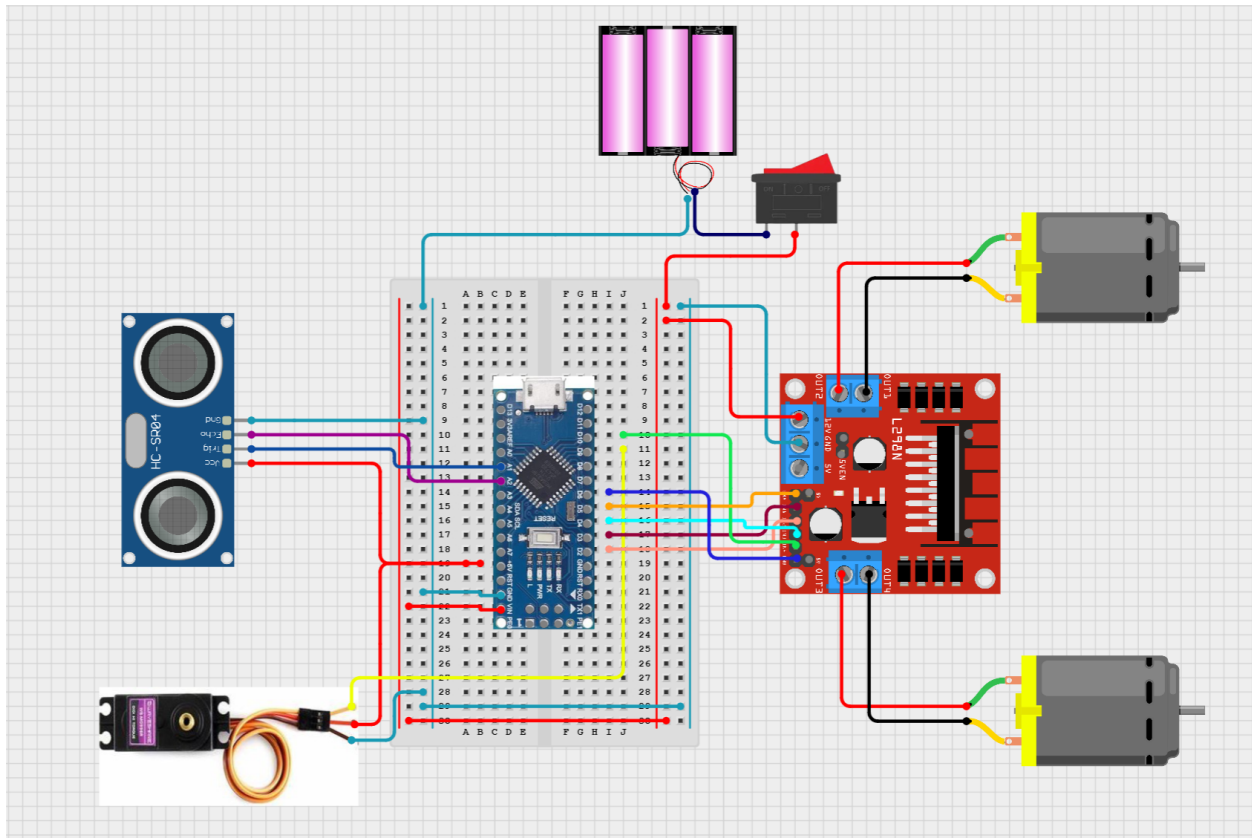
Code for checking the working of the DC BO motor-

```
1 // Define motor control pins
2 const int in1 = 2; // Control pin 1 for Motor A
3 const int in2 = 3; // Control pin 2 for Motor A
4
5 const int in3 = 4; // Control pin 1 for Motor B
6 const int in4 = 5; // Control pin 2 for Motor B
7
8 void setup() {
9     // Set all motor control pins as OUTPUT
10    pinMode(in1, OUTPUT);
11    pinMode(in2, OUTPUT);
12    pinMode(in3, OUTPUT);
13    pinMode(in4, OUTPUT);
14 }
15
16 void loop() {
17     // Test Motor A forward
18     Serial.println("Motor A: Forward");
19     digitalWrite(in1, HIGH);
20     digitalWrite(in2, LOW);
21     delay(3000); // Run for 3 seconds
22
23     // Stop Motor A
24     Serial.println("Motor A: Stop");
25     digitalWrite(in1, LOW);
26     digitalWrite(in2, LOW);
27     delay(2000); // Wait for 2 seconds
28
29     // Test Motor A reverse
30     Serial.println("Motor A: Reverse");
31     digitalWrite(in1, LOW);
32     digitalWrite(in2, HIGH);
33     delay(3000); // Run for 3 seconds
34
35     // Stop Motor A
36     Serial.println("Motor A: Stop");
37     digitalWrite(in1, LOW);
38     digitalWrite(in2, LOW);
39     delay(2000); // Wait for 2 seconds
40
41     // Test Motor B forward
42     Serial.println("Motor B: Forward");
43     digitalWrite(in3, HIGH);
44     digitalWrite(in4, LOW);
45     delay(3000); // Run for 3 seconds
46
47     // Stop Motor B
48     Serial.println("Motor B: Stop");
49     digitalWrite(in3, LOW);
50     digitalWrite(in4, LOW);
51     delay(2000); // Wait for 2 seconds
52
53     // Test Motor B reverse
54     Serial.println("Motor B: Reverse");
55     digitalWrite(in3, LOW);
56     digitalWrite(in4, HIGH);
57     delay(3000); // Run for 3 seconds
58
59     // Stop Motor B
60     Serial.println("Motor B: Stop");
61     digitalWrite(in3, LOW);
62     digitalWrite(in4, LOW);
63     delay(2000); // Wait for 2 seconds
64 }
```

Code for checking the working of the Servo motor and the Ultrasonic distance sensor-

```
1  #include <Servo.h>
2
3  // Define pins for the ultrasonic sensor
4  const int trigPin = A1; // Trig pin connected to digital pin 9
5  const int echoPin = A2; // Echo pin connected to digital pin 10
6
7  // Define servo pin
8  const int servoPin = 9; // Servo signal pin connected to digital pin 6
9
10 Servo myServo; // Create a servo object
11
12 void setup() {
13     Serial.begin(9600); // Initialize serial communication
14
15     // Set up ultrasonic sensor pins
16     pinMode(trigPin, OUTPUT);
17     pinMode(echoPin, INPUT);
18
19     // Attach the servo to the servo pin
20     myServo.attach(servoPin);
21 }
22
23 void loop() {
24     // Move servo to 45° and measure distance
25     myServo.write(45);
26     delay(2000); // Wait for 2 seconds
27     int distance45 = measureDistance();
28     Serial.print("Distance at 45°: ");
29     Serial.print(distance45);
30     Serial.println(" cm");
31
32     // Move servo to 90° and measure distance
33     myServo.write(90);
34     delay(2000); // Wait for 2 seconds
35     int distance90 = measureDistance();
36     Serial.print("Distance at 90°: ");
37     Serial.print(distance90);
38     Serial.println(" cm");
39
40     // Move servo to 135° and measure distance
41     myServo.write(135);
42     delay(2000); // Wait for 2 seconds
43     int distance135 = measureDistance();
44     Serial.print("Distance at 135°: ");
45     Serial.print(distance135);
46     Serial.println(" cm");
47 }
48
49 // Function to measure distance using the ultrasonic sensor
50 int measureDistance() {
51     long duration;
52
53     // Send a short pulse to the Trig pin to start the measurement
54     digitalWrite(trigPin, LOW);
55     delayMicroseconds(2);
56     digitalWrite(trigPin, HIGH);
57     delayMicroseconds(10);
58     digitalWrite(trigPin, LOW);
59
60     // Read the Echo pin to get the pulse duration
61     duration = pulseIn(echoPin, HIGH);
62
63     // Calculate the distance in centimeters
64     int distance = duration * 0.034 / 2; // Speed of sound is 0.034 cm/μs
65
66     return distance;
67 }
```

## Circuit Diagram-



## Hardware Connections-

Component	Pin on Arduino Nano	Pin on Module	Function/Description
L298N Motor Driver			Motor Control
Left Motor Forward	D3	IN1	Controls left motor forward rotation
Left Motor Backward	D2	IN2	Controls left motor backward rotation
Right Motor Forward	D4	IN3	Controls right motor forward rotation
Right Motor Backward	D10	IN4	Controls right motor backward rotation
Left Motor Enable	D5 (PWM)	ENA	Speed control of left motor
Right Motor Enable	D6 (PWM)	ENB	Speed control of right motor
Ultrasonic Sensor (HC-SR04)			Distance Measurement
Trigger (Trig)	A1	TRIG	Sends ultrasonic pulse
Echo	A2	ECHO	Receives reflected pulse
Micro Servo SG90	D9	Signal (PWM)	Object Scanning
Power Connections			Power Supply
L298N Motor Driver (VCC)	External Battery	VCC	Power supply for motors
Arduino Nano (Vin)	External Battery	VIN	Main power for microcontroller
Ultrasonic Sensor (VCC)	5V	VCC	Power from Arduino
Servo Motor (VCC)	5V	VCC	Power from Arduino
All GND Connections	GND (Arduino & Motor Driver)	GND	Common ground for all components

## Code- (Code for Pursuitron)

```
1  #include <Servo.h>
2
3  // Motor control pins
4  const int LeftMotorForward = 2;   // D2
5  const int LeftMotorBackward = 3;  // D3
6  const int RightMotorForward = 4;  // D4
7  const int RightMotorBackward = 10; // D5
8
9  // Enable pins for motor speed control
10 const int LeftMotorEnable = 6;    // D6 (PWM)
11 const int RightMotorEnable = 5;   // D5 (PWM)
12
13 // Ultrasonic sensor pins
14 #define trig_pin A1 // A1 (analog pin 1)
15 #define echo_pin A2 // A2 (analog pin 2)
16
17 // Servo motor pin
18 #define servo_pin 9 // D9
19
20 // Threshold distances
21 const int pursuitThreshold = 30; // Pursue objects within 30 cm
22 const int tooCloseDistance = 10; // Move back if closer than 10 cm
23
24 // Servo angles for scanning (including 90 degrees for front)
25 const int scanAngles[] = {30, 60, 90, 120, 150};
26
27 Servo servo_motor; // Servo motor object
28
29 void setup() {
30     // Initialize motor control pins as outputs
31     pinMode(LeftMotorForward, OUTPUT);
32     pinMode(LeftMotorBackward, OUTPUT);
33     pinMode(RightMotorForward, OUTPUT);
34     pinMode(RightMotorBackward, OUTPUT);
35     pinMode(LeftMotorEnable, OUTPUT);
36     pinMode(RightMotorEnable, OUTPUT);
37
38     // Initialize ultrasonic sensor pins
39     pinMode(trig_pin, OUTPUT);
40     pinMode(echo_pin, INPUT);
41
42     // Attach servo to pin D9 and set it to 90 degrees (center position)
43     servo_motor.attach(servo_pin);
44     servo_motor.write(90); // Center position
45     delay(1000);
46
47     // Set motor speed to 100/255 using PWM
48     analogWrite(LeftMotorEnable, 100);
49     analogWrite(RightMotorEnable, 100);
50 }
```

```

51
52 void loop() {
53     int distance = measureDistance(); // Measure distance in front
54
55     // If an object is within 30 cm, pursue it
56     if (distance <= pursuitThreshold && distance > 0) {
57         // If too close, move back until 10 cm
58         if (distance < tooCloseDistance) {
59             moveBackward();
60             while (measureDistance() < tooCloseDistance) {
61                 delay(10); // Keep moving back until 10 cm is reached
62             }
63             moveStop();
64         } else {
65             moveForward(); // Otherwise, move forward
66         }
67     } else {
68         servo_motor.write(90);
69         moveStop();
70         // If no object is detected, scan left and right
71         int detectedAngle = scanForObject();
72         if (detectedAngle != -1) {
73             // Turn towards the detected angle and align with the object
74             alignWithObject(detectedAngle);
75         }
76     }
77     delay(50); // Small delay for stability
78 }
79 // Function to measure distance using ultrasonic sensor
80 int measureDistance() {
81     digitalWrite(trig_pin, LOW);
82     delayMicroseconds(2);
83     digitalWrite(trig_pin, HIGH);
84     delayMicroseconds(10);
85     digitalWrite(trig_pin, LOW);
86     return pulseIn(echo_pin, HIGH) * 0.0344 / 2; // Convert to cm
87 }
88
89 // Function to scan for objects at 30°, 60°, 90°, 120°, and 150°
90 int scanForObject() {
91     for (int i = 0; i < 5; i++) {
92         servo_motor.write(scanAngles[i]); // Move servo to the current angle
93         delay(500); // Wait for the servo to stabilize
94         int distance = measureDistance(); // Measure distance at this angle
95         if (distance <= pursuitThreshold && distance > 0) {
96             return scanAngles[i]; // Return the angle where an object is detected
97         }
98     }
99     servo_motor.write(90); // Return servo to center position
100     return -1; // No object detected
101 }

```



```

102
103 // Function to align the bot with the detected object
104 void alignWithObject(int targetAngle) {
105     while (true) {
106         // Turn towards the detected angle
107         if (targetAngle < 90) {
108             turnRight(); // Turn right if the target angle is less than 90 degrees
109         } else if (targetAngle > 90) {
110             turnLeft(); // Turn left if the target angle is greater than 90 degrees
111         }
112
113         // Move forward for 500 milliseconds
114         moveForward();
115         delay(500);
116         moveStop();
117
118         // Scan again to check if the object is now at 90 degrees
119         int newDistance = measureDistance();
120         if (newDistance <= pursuitThreshold && newDistance > 0) {
121             // If the object is now centered, break the loop
122             if (servo_motor.read() == 90) {
123                 break;
124             }
125         }
126
127         // Update the target angle based on the new scan
128         targetAngle = scanForObject();
129         if (targetAngle == -1) {
130             break; // Exit if no object is detected
131         }
132     }
133 }
134 // Motor control functions
135 void moveForward() {
136     digitalWrite(LeftMotorForward, HIGH);
137     digitalWrite(LeftMotorBackward, LOW);
138     digitalWrite(RightMotorForward, HIGH);
139     digitalWrite(RightMotorBackward, LOW);
140 }
141 void moveBackward() {
142     digitalWrite(LeftMotorForward, LOW);
143     digitalWrite(LeftMotorBackward, HIGH);
144     digitalWrite(RightMotorForward, LOW);
145     digitalWrite(RightMotorBackward, HIGH);
146 }
147 void moveStop() {
148     digitalWrite(LeftMotorForward, LOW);
149     digitalWrite(LeftMotorBackward, LOW);
150     digitalWrite(RightMotorForward, LOW);
151     digitalWrite(RightMotorBackward, LOW);
152 }
153 void turnLeft() {
154     digitalWrite(LeftMotorForward, LOW);
155     digitalWrite(LeftMotorBackward, HIGH);
156     digitalWrite(RightMotorForward, HIGH);
157     digitalWrite(RightMotorBackward, LOW);
158     delay(250); // Adjust delay for turning angle
159     moveStop();
160 }
161 void turnRight() {
162     digitalWrite(LeftMotorForward, HIGH);
163     digitalWrite(LeftMotorBackward, LOW);
164     digitalWrite(RightMotorForward, LOW);
165     digitalWrite(RightMotorBackward, HIGH);
166     delay(250); // Adjust delay for turning angle
167     moveStop();
168 }

```



## Working of the Bot-

The bot measures the distance using the ultrasonic sensor at servo angles of 30, 60, 90, 120, and 150 degrees. (Here 90 degrees is the position at which the sensor is facing forward)

If the target is at less than a 90-degree angle, then it turns right and scans for the target again.

If the target is at more than 90-degree angle, then it turns left and scans for the target again.

If the target is at 90 degrees, it will move forward provided the distance between the target and the bot is not less than 10 cm.

If the distance between the target and the bot is less than 10 cm, then the bot will move slightly backwards and continue scanning.

## Safety and Guidelines-

- Ensure the robot is powered off during electrical connections or adjustments.
- Be cautious while handling tools and sharp objects during the assembly process to prevent injuries.
- Avoid placing the robot near water or any other liquid, as it may cause damage or electrical hazards.
- Do not touch any exposed wires or connections when the robot is powered on.
- Follow proper battery safety guidelines and use appropriate battery types and ratings.

## Doubts and Assistance-

If you encounter any difficulties or have doubts during the fabrication process, do not hesitate to seek assistance from the workshop volunteers or instructors. They will be available to answer questions, provide guidance, and ensure the successful completion of the project.