



Gurukul Technicals Institute

Computer education · Robotics · Science projects · Delhi

FREE EBOOK

ROBOTICS AND ELECTRONICS

Build an Obstacle-Avoiding Robot

An ultrasonic sensor robot that finds its own way, step by step

INSIDE THIS BOOK

- How an obstacle-avoiding robot works
- Parts, tools and safety
- Building the chassis and the sensor mount
- Measuring distance with sound
- Wiring the robot
- ...and 6 more chapters, practice questions and a glossary

Intermediate

Students of class 7 to 12 and robotics clubs

engineergurukul.com

+91 97116 99511

Build an Obstacle-Avoiding Robot

Build a robot that drives on its own, spots walls and furniture with an ultrasonic sensor, looks left and right on a servo, and turns towards the clearer path. You will learn the speed-of-sound maths, wire and code the whole robot, tune it, add lights, a buzzer and Bluetooth control, and present it with confidence.

WHO IT IS FOR	Students of class 7 to 12 and robotics clubs
LEVEL	Intermediate
SUBJECT AREA	Robotics and electronics
WHAT IS INSIDE	11 chapters, practice questions with answers, and a glossary

HOW TO USE IT

Read one chapter at a time and try the examples as you go: on a computer, at your workbench or in your notebook. The boxes give tips, safety points and short activities. When you finish, test yourself with the practice questions and check your answers at the end. Words you may not know are explained in the glossary.



Gurukul Technicals Institute

A-10, Chief Commissioner Colony, opposite Rana Pratap Bagh, Delhi 110007
Phone 011 2721 1512 · Mobile and WhatsApp +91 97116 99511
engineergurukul@yahoo.co.in · engineergurukul.com

Free ebook. You may read, print and share this ebook for learning. It may not be sold or changed.

© 2026 Gurukul Technicals Institute. First edition, October 2026. The information was checked when it was written; software, prices and rules change, so for anything official always check the latest source.

What you will find inside

1	How an obstacle-avoiding robot works	4
2	Parts, tools and safety	5
3	Building the chassis and the sensor mount	7
4	Measuring distance with sound	8
5	Wiring the robot	10
6	Testing each part on its own	12
7	The complete Arduino code	14
8	Testing and tuning thresholds	18
9	Power supply and troubleshooting	19
10	Extensions: buzzer, LEDs and Bluetooth override	21
11	Presenting at an exhibition	23
	Practice questions	25
	Answers	26
	Glossary	27

How an obstacle-avoiding robot works

An obstacle-avoiding robot drives around a room by itself. When something blocks its path, it stops, looks left and right, and turns towards the side with more space. Nobody steers it. It makes every decision on its own.

Seeing with sound

Instead of eyes, this robot uses an **ultrasonic sensor** called the **HC-SR04**. Ultrasonic means sound so high-pitched that humans cannot hear it. The sensor has two round "eyes" on the front: one is a tiny speaker that sends out a short burst of sound, and the other is a tiny microphone that listens for the echo.

When the sound hits a wall, it bounces back. The sensor measures how long the echo takes to return. A short time means the wall is close; a long time means it is far away. The Arduino turns that time into a distance in centimetres.

Looking around on a servo

The sensor sits on top of a small **servo motor**: a motor that turns to an exact angle and holds it. When the path ahead is blocked, the servo turns the sensor to the left, then to the right, so the robot can measure both sides without moving its wheels. Then it compares the two distances and chooses.

The robot's decision loop

1. Look straight ahead and measure the distance.
2. If the path is clear, drive forward and go back to step 1.
3. If something is too close, stop and back up a little.
4. Turn the sensor left and measure. Turn it right and measure.
5. Turn the robot towards the side with more space. If both sides are blocked, turn round.
6. Point the sensor ahead again and go back to step 1.

DID YOU KNOW

Bats find insects in the dark using the same idea, called **echolocation**. They make high-pitched calls and listen for the echoes. The reversing sensors in many cars' bumpers work on the echo idea too.

REMEMBER

The robot senses, decides and acts, again and again. Every part you build in this book does one of those three jobs.

TRY THIS

Stand facing a big wall in an open ground, about 50 metres away, and clap once, loudly. Listen for the echo. Sound takes a moment to travel there and back. The HC-SR04 measures the same kind of delay, but in millionths of a second.

CHAPTER 2

Parts, tools and safety

Gather everything before you begin. A missing part halfway through is the most common reason a project stalls.

Parts list

Part	Quantity	What it does
Arduino Uno with USB cable	1	The brain
HC-SR04 ultrasonic sensor	1	Measures distance
SG90 micro servo, with its arms and screws	1	Turns the sensor left and right
Holder or bracket for the HC-SR04	1	Joins the sensor to the servo arm
L298N motor driver module	1	Powers and controls the motors
BO motors (100 to 200 RPM) with wheels	2	Drive the robot

Part	Quantity	What it does
Ball caster	1	Supports the front
2WD chassis plate with motor clamps and screws	1	The body
Battery holder for 2 × 18650 Li-ion cells, with cells, or a 6 × AA holder	1	Power
On/off switch	1	Cuts the battery safely
Mini breadboard	1	Shares 5 V and GND
Jumper wires (male-to-male and male-to-female)	about 20	Connections
Spacers, double-sided foam tape, cable ties	a few	Mounting and tidying

Tools

- Small screwdrivers (cross-head and flat)
- A wire stripper and small pliers
- A multimeter
- A measuring tape or a metre scale
- A computer with the Arduino IDE (free from arduino.cc)

Meet the HC-SR04

Pin	What it does
VCC	Power: 5 V
Trig	Trigger: a short pulse from the Arduino tells it to send a sound burst
Echo	Goes HIGH for exactly as long as the sound took to go and come back
GND	Ground

The sensor sends its burst at 40,000 vibrations per second (40 kHz). Human hearing stops at about 20 kHz, so you hear nothing. It can measure from about 2 cm to a few metres.

SAFETY

All power comes from batteries under 12 V; never connect anything to mains. Li-ion cells need a proper Li-ion charger and an adult's supervision, and must never be shorted, crushed or left charging unattended. Younger students should use the 6 × AA holder. Soldering, hot glue and cutting need an adult.

TIP

The servo arms come in a small packet with two tiny screws. Keep the packet taped inside your project box: those screws are very easy to lose.

CHAPTER 3

Building the chassis and the sensor mount

Most 2WD acrylic kits go together in the same order. Look at all your pieces first and plan where each one will sit.

The chassis

1. Peel the protective paper off both sides of the acrylic plate.
2. Fix each BO motor between its two acrylic clamps with two long screws and nuts. Point the shafts outward and tighten until the motors cannot wiggle.
3. If the motors have no wires, ask an adult to solder two wires to each one.
4. Screw the ball caster to the other end of the plate, with spacers if needed, so the plate sits level.
5. Fix the battery holder low and near the middle. The robot must not tip backwards.
6. Mount the Arduino on spacers, and put the L298N close to the motors.
7. Fit the switch where you can reach it, in the battery's positive wire.
8. Push the wheels fully onto the flat-sided shafts.

The caster end becomes the **front**. The sensor goes there, so it sees obstacles before the wheels reach them.

Centring the servo first

This step matters more than any other in the build. If you fix the sensor to the servo at a random angle, your robot will think "straight ahead" is somewhere off to one side.

1. Connect the servo: brown wire to GND, red to 5V, orange to Arduino pin 9.
2. Upload the centring sketch in the chapter on testing (it sets the servo to 90 degrees).
3. With the servo now held at 90 degrees, push the servo arm onto the shaft at the angle that will make the sensor face straight forward once its holder is fixed on. Fix the arm with its small screw.
4. Attach the sensor holder to the arm, with the two round "eyes" facing straight forward.

Mounting the servo

Fix the servo near the front edge with its shaft pointing up, using screws through its mounting tabs or strong double-sided foam tape. The sensor should sit high enough that the chassis does not block its view, and its front should stick out slightly past the robot's front edge.

SAFETY

If you use hot glue to fix the servo or bracket, an adult must handle the glue gun. The nozzle and fresh glue can burn skin.

TIP

Check that the sensor can swing fully left and right without hitting the wires, the Arduino or the chassis. Leave the sensor's wires a little slack so they do not pull as it turns.

EXAMPLE

Kabir from Bhopal fixed his sensor before centring the servo. His robot kept stopping in an empty corridor, because its "straight ahead" really pointed 30 degrees to the left, at the corridor wall. Centring the servo and refitting the arm fixed it in five minutes.

CHAPTER 4

Measuring distance with sound

This chapter explains the maths inside the robot. It is simple, and knowing it will help you answer viva questions with confidence.

The speed of sound

Sound travels through air at about **343 metres per second** when the air is at 20 °C. In warm air it goes a little faster: about 0.6 m/s faster for every 1 °C rise.

Speed of sound $\approx 331 + 0.6 \times \text{temperature in } ^\circ\text{C}$ (in metres per second)

The sensor measures time in **microseconds** (μs). A microsecond is one millionth of a second. So let us change the speed into centimetres per microsecond:

- 343 metres per second = 34,300 centimetres per second
- $34,300 \div 1,000,000 = 0.0343$ centimetres per microsecond

There and back

The sound travels to the obstacle and back again. The echo time covers twice the distance, so we divide by 2:

$$\text{Distance (cm)} = \text{echo time } (\mu\text{s}) \times 0.0343 \div 2$$

Multiplying by 0.0343 and then dividing by 2 is the same as dividing by about 58, so a handy shortcut is:

$$\text{Distance (cm)} \approx \text{echo time } (\mu\text{s}) \div 58$$

EXAMPLE

The Echo pin stays HIGH for 1,166 μs . Distance = $1,166 \times 0.0343 \div 2 \approx 20$ cm. With the shortcut: $1,166 \div 58 \approx 20$ cm. The obstacle is 20 cm away.

EXAMPLE

A chair leg stands 50 cm away. The sound must travel 100 cm in total. Time = $100 \div 0.0343 \approx 2,915$ μs , or under 3 thousandths of a second.

Does temperature matter?

EXAMPLE

Meera tests her robot in Delhi in May, when the room is 40 °C. Sound now travels at about $331 + 0.6 \times 40 = 355$ m/s. A wall 100 cm away sends back an echo after about 5,634 μs . Her code still uses 0.0343, so it calculates $5,634 \times 0.0343 \div 2 \approx 96.6$ cm. The error is about 3 cm, which does not matter for avoiding walls. It would matter for an accurate measuring tape.

How the reading is taken

1. The Arduino holds Trig LOW for a moment, then sets it HIGH for 10 μs , then LOW again.
2. The sensor sends out eight quick pulses of 40 kHz sound.
3. It sets Echo HIGH and waits for the echo to come back.
4. When the echo arrives, Echo goes LOW.
5. The Arduino's `pulseIn()` function measures how long Echo stayed HIGH.

If no echo comes back, because nothing is in range, `pulseIn()` gives up after the time limit you set and returns 0. Our code treats 0 as "far away".

What the sensor finds hard to see

- **Soft things** such as cushions, curtains and clothes absorb sound, so the echo is weak.
- **Surfaces that face the sensor at a slant** bounce the sound away, like a ball thrown at a slanting wall.
- **Thin things** such as chair legs and table wires may reflect too little sound.
- **Very close things** under about 2 cm, because the echo returns before the sensor is ready.

The sound spreads out in a cone about 15 degrees wide, so the sensor sees a patch in front of it, not a thin line.

REMEMBER

Distance = speed × time. Divide by 2, because the sound makes a round trip.

CHAPTER 5

Wiring the robot

Work with the battery off and the USB cable unplugged. Tick off each row as you connect it.

Sharing 5 V and ground

The Uno has only one 5V pin on its header, but both the sensor and the servo need 5 V. Run one wire from the Uno's 5V pin and one from its GND to the two rails of a mini breadboard, and connect the sensor and servo to those rails.

Wiring table

From	To
HC-SR04 VCC	Breadboard 5 V rail
HC-SR04 GND	Breadboard GND rail
HC-SR04 Trig	Uno pin 12
HC-SR04 Echo	Uno pin 11
Servo red	Breadboard 5 V rail
Servo brown	Breadboard GND rail

From	To
Servo orange	Uno pin 9
L298N ENA (jumper cap removed)	Uno pin 5
L298N IN1, IN2	Uno pins 7, 8
L298N IN3, IN4	Uno pins 4, 3
L298N ENB (jumper cap removed)	Uno pin 6
Left motor	L298N OUT1 and OUT2
Right motor	L298N OUT3 and OUT4
Battery positive	Switch, then L298N 12V terminal and Uno VIN
Battery negative	L298N GND terminal
Uno GND	L298N GND terminal

Leave the L298N's 5V terminal empty, and keep its regulator jumper (beside the power terminals) in place.

Why these pins?

Motor speed is set by **PWM** (pulse width modulation): the Arduino switches the enable pin on and off very fast, and a longer on-time means more speed. On an Uno, the Servo library stops PWM from working on pins 9 and 10. That is why the motor speed wires (ENA and ENB) go to pins 5 and 6, which keep their PWM. The servo itself can use pin 9 without any problem.

Under each ENA and ENB jumper cap, one pin is 5 V and the other is the real enable input. Read the label on your board, or power it and measure each pin to GND with a multimeter. Connect your wire to the pin that does **not** show 5 V.

Why the Uno goes on VIN

The Uno's VIN pin feeds its own regulator, which makes a steady 5 V for the board, the sensor and the servo. VIN works best with 7 to 12 V, so a 2 × 18650 pack (7.4 V) or six AA cells suit it.

REMEMBER

The Uno GND to L298N GND wire creates a **common ground**. Without it the motor driver cannot read the Arduino's signals.

TIP

Twist each motor's two wires together and keep them away from the sensor wires. Motors make electrical noise that can upset readings.

SAFETY

Check the battery's polarity twice before switching on. Positive and negative swapped can destroy the driver and the Arduino instantly.

CHAPTER 6

Testing each part on its own

Never test the whole robot first. If it fails, you will not know which part is at fault. Test each part alone, then put them together.

Test 1: centre the servo

C++

```
// Centre the servo at 90 degrees, then do nothing
#include <Servo.h>

Servo scanner;

void setup() {
  scanner.attach(9);
  scanner.write(90);
}

void loop() {
}
```

The servo should turn to its middle position and stay there. Use this moment to fix the servo arm and sensor facing straight ahead, as described in the chapter on building.

Test 2: measure distance

CPP

```
// Print the echo time and the distance from the HC-SR04
const int TRIG_PIN = 12;
const int ECHO_PIN = 11;

void setup() {
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  Serial.begin(9600);
}

void loop() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  unsigned long duration = pulseIn(ECHO_PIN, HIGH, 30000UL);
  Serial.print(duration);
  Serial.print(" us ");
  Serial.print(duration * 0.0343 / 2);
  Serial.println(" cm");
  delay(200);
}
```

Open the Serial Monitor at 9600 baud. Hold a book in front of the sensor at 10, 20 and 50 cm, measured with a scale, and compare.

Real distance	Reading should be about
10 cm	9 to 11 cm
20 cm	19 to 21 cm
50 cm	48 to 52 cm
Nothing near (open space)	A large number from a far wall, or 0 if no echo returns in time

Test 3: the motors

With the wheels lifted off the table, check each motor turns forward when it should. A simple way is to load the complete program from the next chapter with the sensor pointing at open space: both wheels should spin forward. Put your hand in front of the sensor: they should stop, reverse briefly, and then the servo should look left and right.

TRY THIS

Point the sensor at a pillow, then at a steel almirah, from the same distance. Watch how the readings jump about for the pillow. Soft things are hard to see with sound.

TIP

If any wheel turns backward, swap that motor's two wires at the L298N. Do it now, before moving on.

CHAPTER 7

The complete Arduino code

Here is the whole program. Read the explanation after it, so you understand every part.

```
// Obstacle-avoiding robot: HC-SR04 on an SG90 servo, L298N, Arduino Uno
#include <Servo.h>
```

```
// ----- Pins -----
```

```
const int TRIG_PIN = 12;
const int ECHO_PIN = 11;
const int SERVO_PIN = 9;
const int ENA = 5; // left motor speed (PWM)
const int IN1 = 7; // left motor direction
const int IN2 = 8;
const int ENB = 6; // right motor speed (PWM)
const int IN3 = 4; // right motor direction
const int IN4 = 3;
```

```
// ----- Settings to tune -----
```

```
const int STOP_DISTANCE = 25; // cm: anything closer is an obstacle
const int LOOK_LEFT = 160; // servo angles: swap 160 and 20 if
const int LOOK_AHEAD = 90; // your servo looks the other way
const int LOOK_RIGHT = 20;
int driveSpeed = 150; // 0 to 255
int turnSpeed = 160;
int turnTime = 400; // ms for roughly a quarter turn
```

```
Servo scanner;
```

```
void setup() {
    pinMode(TRIG_PIN, OUTPUT);
    pinMode(ECHO_PIN, INPUT);
    pinMode(ENA, OUTPUT);
    pinMode(IN1, OUTPUT);
    pinMode(IN2, OUTPUT);
    pinMode(ENB, OUTPUT);
    pinMode(IN3, OUTPUT);
    pinMode(IN4, OUTPUT);
    scanner.attach(SERVO_PIN);
    scanner.write(LOOK_AHEAD);
    Serial.begin(9600);
    drive(0, 0);
    delay(3000); // time to put the robot down
}
```

```
void loop() {
    int ahead = readDistance();
    Serial.println(ahead);

    if (ahead > STOP_DISTANCE) {
        drive(driveSpeed, driveSpeed); // path clear: keep going
        delay(60); // the sensor needs a short rest
    }
}
```

```

    return;
}

// Something is in the way: stop and back off a little
drive(0, 0);
delay(200);
drive(-driveSpeed, -driveSpeed);
delay(300);
drive(0, 0);

// Look both ways
int leftDist = lookAndMeasure(LOOK_LEFT);
int rightDist = lookAndMeasure(LOOK_RIGHT);
scanner.write(LOOK_AHEAD);
delay(300);

// Decide
if (leftDist <= STOP_DISTANCE && rightDist <= STOP_DISTANCE) {
    drive(turnSpeed, -turnSpeed);    // both blocked: turn round
    delay(turnTime * 2);
} else if (leftDist > rightDist) {
    drive(-turnSpeed, turnSpeed);    // more room on the left
    delay(turnTime);
} else {
    drive(turnSpeed, -turnSpeed);    // more room on the right
    delay(turnTime);
}
drive(0, 0);
delay(100);
}

int lookAndMeasure(int angle) {
    scanner.write(angle);
    delay(400);                      // let the servo arrive and settle
    return readDistance();
}

int readDistance() {
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
    unsigned long duration = pulseIn(ECHO_PIN, HIGH, 30000UL);
    if (duration == 0) return 500;    // no echo: treat as far away
    return (int)(duration * 0.0343 / 2);
}

// Positive speed = forward, negative = backward, 0 = stop
void drive(int leftSpeed, int rightSpeed) {

```

```

    setMotor(ENA, IN1, IN2, leftSpeed);
    setMotor(ENB, IN3, IN4, rightSpeed);
}

void setMotor(int enPin, int inA, int inB, int speed) {
    if (speed > 0) {
        digitalWrite(inA, HIGH);
        digitalWrite(inB, LOW);
    } else if (speed < 0) {
        digitalWrite(inA, LOW);
        digitalWrite(inB, HIGH);
        speed = -speed;
    } else {
        digitalWrite(inA, LOW);
        digitalWrite(inB, LOW);
    }
    analogWrite(enPin, constrain(speed, 0, 255));
}

```

How the code works

- `** #include <Servo.h> **` brings in the Servo library, and `Servo scanner;` creates a servo object named scanner.
- **Settings.** All the numbers you will tune sit together near the top: the stop distance, the three servo angles, the speeds and the turn time.
- **setup()** sets the pins, points the sensor ahead, starts the Serial Monitor and waits three seconds.
- **loop()** measures the distance ahead. If it is more than `STOP_DISTANCE`, the robot drives forward and `return` starts the loop again. Otherwise it stops, backs off, looks both ways and turns towards the bigger distance.
- **lookAndMeasure()** turns the servo, waits 400 ms for it to arrive and stop shaking, then measures.
- **readDistance()** sends the 10 μ s trigger, times the echo with `pulseIn()` (giving up after 30,000 μ s), and converts the time to centimetres. A time of 0 means no echo, so it returns 500, meaning "far away".
- **drive()** and **setMotor()** set each wheel's direction and speed. A negative number means backward.

REMEMBER

The robot turns by timing. `turnTime` is how long it spins, not an angle. You will set it by testing, in the next chapter.

Testing and tuning thresholds

A new robot rarely behaves perfectly on its first run. Tuning means adjusting the settings, one at a time, until it moves smoothly and safely.

Your first runs

1. Choose an open floor with a few large, hard obstacles: a wall, a box, a steel cupboard.
2. Switch on, place the robot, and step back during the three-second wait.
3. Watch it drive, stop, look, and turn. Note anything odd in your notebook.
4. Switch off and adjust one setting at a time.

Tuning the stop distance

`STOP_DISTANCE` decides how close the robot gets before reacting. The robot keeps moving between readings and needs a little distance to stop, so a faster robot needs a larger stop distance.

EXAMPLE

Farhan's robot moves at about 0.5 m/s, which is 50 cm per second. One pass through the loop takes about 100 ms (a tenth of a second), so between readings it travels about $50 \times 0.1 = 5$ cm. Stopping takes a few centimetres more. With `STOP_DISTANCE` at 25 cm, it always halts with room to spare. When he tried 10 cm, it bumped into walls.

STOP_DISTANCE	What usually happens
Too small (under 15 cm)	Bumps into things, or cannot turn without hitting them
About 20 to 30 cm	A good start for most small robots
Too large (over 50 cm)	Turns away from everything and gets stuck in corners

Tuning the turn time

Spin the robot with `turnTime` at 400 and see how far it turns. You want roughly a quarter turn (90 degrees). If it turns too little, raise the number; too much, lower it. Batteries matter here: fresh cells turn the robot further in the same time.

Tuning the servo angles

If the robot keeps turning towards the more crowded side, the servo probably looks right when the code thinks it looks left. Swap the numbers for `LOOK_LEFT` and `LOOK_RIGHT`. If the sensor hits the chassis at the edges, use angles closer to 90, such as 150 and 30.

Tuning the speed

Slow robots are easier to tune. Start with `driveSpeed` at 150. When everything works, try higher speeds, and raise `STOP_DISTANCE` a little each time.

TIP

Keep a tuning table: setting, value, and what happened. Evidence beats memory, especially on exhibition morning.

TRY THIS

Build a small maze from cardboard boxes on the floor. Count how many times your robot touches a wall in two minutes. Tune one setting, then count again. You now have a fair test.

CHAPTER 9

Power supply and troubleshooting

Most problems with this robot come from power. Motors, a servo and a sensor all share one battery, and the motors are greedy.

Power tips

- **Use strong batteries.** A 2 × 18650 Li-ion pack or six AA cells suits this robot. Never use a 9 V PP3 battery: it cannot supply the current the motors need, and the Arduino keeps resetting.
- **Feed the Uno through VIN,** never straight into its 5V pin.
- **Add a capacitor.** A large electrolytic capacitor, such as 470 µF rated at 16 V or more, across the servo's 5 V and GND helps stop resets when the servo jerks. Its stripe marks the negative leg; fitted backwards it can swell or burst.
- **Keep power wires short,** and tighten every screw terminal.

EXAMPLE

Suppose a set of six AA alkaline cells costs ! 150 and lasts three test sessions, while two 18650 cells and a proper charger cost ! 700 and last for hundreds of charges. After about fifteen sessions the alkaline cells have cost ! 750, and the rechargeable set is ahead.

Brown-outs

If the robot freezes or the servo snaps back to the centre every time the motors start, the battery voltage is dipping and the Arduino is restarting. This is called a **brown-out**. You will notice the three-second wait happening again. Charge the batteries, check for loose connections, and add the capacitor above.

Troubleshooting table

Problem	Likely cause	What to do
Robot spins in circles	One motor wired backward	Swap that motor's wires at the L298N
Reading is always 0	Trig and Echo swapped, or no 5 V to the sensor	Check pins 12 and 11 and the 5 V rail
Readings jump wildly	Loose wires, motor noise, or soft targets	Tighten wires, twist motor wires, test on hard objects
Robot crashes into chair legs	Thin objects reflect little sound	Accept it, or add a second sensor lower down
Always turns the same way	Servo angles swapped, or servo not centred	Swap <code>LOOK_LEFT</code> and <code>LOOK_RIGHT</code> ; re-centre the servo
Servo jitters or buzzes	Weak 5 V, or pressing against its end stop	Add the capacitor; use angles nearer 90
Motors do nothing	Battery off, enable jumpers on, no common ground	Check the switch, ENA and ENB, and the GND wire
Gets stuck in corners	Stop distance too large or turn too short	Lower <code>STOP_DISTANCE</code> ; raise <code>turnTime</code>
Upload fails	Wrong board or port	Check the Tools menu and the cable

REMEMBER

Change one thing, then test. Two changes at once teach you nothing.

Extensions: buzzer, LEDs and Bluetooth override

Once your robot works, make it more fun and more useful. Add each extension one at a time, and test after each.

A buzzer and two LEDs

Part	Connect to
Active buzzer, positive leg	Uno A0
Active buzzer, negative leg	GND
Green LED	Uno A1, then a 220 Ω resistor, then the LED's long leg; short leg to GND
Red LED	Uno A2, then a 220 Ω resistor, then the LED's long leg; short leg to GND

An **active buzzer** makes its own tone when given power, so you just switch it on and off. Choose a small 5 V one that draws less than 20 mA; a louder one needs a transistor. The Uno's analog pins A0 to A5 also work as ordinary digital outputs.

Add these lines near the top of the program:

CPP

```
const int BUZZER = A0;
const int GREEN_LED = A1;
const int RED_LED = A2;
```

Add these to `setup()`:

CPP

```
pinMode(BUZZER, OUTPUT);
pinMode(GREEN_LED, OUTPUT);
pinMode(RED_LED, OUTPUT);
```

In `loop()`, inside the "path clear" block, just before `drive(driveSpeed, driveSpeed);`, add:

CPP

```
digitalWrite(GREEN_LED, HIGH);  
digitalWrite(RED_LED, LOW);
```

And straight after the comment "Something is in the way", add:

CPP

```
digitalWrite(GREEN_LED, LOW);  
digitalWrite(RED_LED, HIGH);  
digitalWrite(BUZZER, HIGH);    // short beep  
delay(150);  
digitalWrite(BUZZER, LOW);
```

Bluetooth override

An **HC-05** Bluetooth module lets you take control from an Android phone with a Bluetooth terminal app. Send **M** for manual mode, then **F**, **B**, **L**, **R** or **S** to drive; send **A** to hand control back to the robot.

HC-05	Connect to
VCC	5 V rail
GND	GND rail
TXD	Uno pin 2
RXD	Leave unconnected (this robot only listens)

If you later want the robot to send messages back, connect Uno pin 10 to RXD through a **voltage divider**: 1 k Ω from pin 10 to RXD, and 2 k Ω from RXD to GND. This brings 5 V down to about 3.3 V, which the HC-05 expects. Pin 10 is fine as a plain digital pin even though the Servo library has taken its PWM.

Add at the top:

CPP

```
#include <SoftwareSerial.h>  
SoftwareSerial bt(2, 10);    // RX on pin 2, TX on pin 10  
bool manualMode = false;
```

Add `bt.begin(9600);` to `setup()`. Then put this at the very start of `loop()`:

```

if (bt.available() > 0) {
    char c = bt.read();
    if (c == 'M') manualMode = true;
    else if (c == 'A') manualMode = false;
    else if (manualMode) {
        if (c == 'F') drive(driveSpeed, driveSpeed);
        else if (c == 'B') drive(-driveSpeed, -driveSpeed);
        else if (c == 'L') drive(-turnSpeed, turnSpeed);
        else if (c == 'R') drive(turnSpeed, -turnSpeed);
        else if (c == 'S') drive(0, 0);
    }
}
if (manualMode) return;    // skip the automatic driving

```

Pair the phone with the HC-05 in its Bluetooth settings (the PIN is often 1234 or 0000), then connect from the app. The servo may twitch slightly when a command arrives; that is normal when SoftwareSerial and the Servo library run together.

TRY THIS

Make manual mode safer: in manual mode, measure the distance ahead and stop the robot if it is closer than `STOP_DISTANCE`, even if the phone says go.

CHAPTER 11

Presenting at an exhibition

Judges and visitors want to see the robot work, and they want to hear that you understand it. Prepare for both.

Before the day

- Charge every battery and pack a spare set.
- Bring a few hard obstacles of your own, such as shoe boxes, in case the hall has none.
- Save your code on the laptop and a pen drive.
- Make a chart with a block diagram (sensor and servo, Arduino, driver, motors, battery), the distance formula with one worked example, and your tuning table.

At the venue

1. Choose a corner of your table or floor and build a small arena with boxes.

2. Run the distance test sketch once to check the sensor.
3. Do a few practice runs. Check `turnTime` again, because floors differ.

A one-minute explanation

"This robot avoids obstacles by itself. The ultrasonic sensor sends out sound too high for us to hear and times the echo. Sound travels at about 343 metres per second, so from the time we calculate the distance, dividing by two because the sound goes there and back. When something is closer than 25 centimetres, the robot stops, the servo turns the sensor left and right, and the robot turns towards the side with more space."

EXAMPLE

At an inter-school fair in Kochi, Ananya let visitors walk in front of her robot. Each time, it stopped, looked both ways and turned. Visitors remembered the "looking" more than anything else she said.

Viva questions and model answers

1. **How does the sensor measure distance?** It sends a burst of ultrasonic sound, times the echo, and the Arduino calculates $\text{distance} = \text{speed} \times \text{time} \div 2$.
2. **Why divide by 2?** The sound travels to the obstacle and back, so the time covers twice the distance.
3. **Why can't we hear the sensor?** It uses 40 kHz sound, and human hearing stops at about 20 kHz.
4. **What objects are hard to detect?** Soft things that absorb sound, surfaces at a slant, and very thin things like chair legs.
5. **Why is the sensor on a servo?** So the robot can look left and right without turning its whole body.
6. **Why are the motor speed wires on pins 5 and 6?** The Servo library stops PWM on pins 9 and 10 of an Uno.
7. **Does temperature change the reading?** Yes, slightly. Sound is faster in warm air, so at 40 °C the code reads about 3% short.
8. **Where is this idea used in real life?** In car reversing sensors, and in robots that must avoid bumping into things.

REMEMBER

Speak slowly, show the robot working, and point to your chart as you explain. A calm, clear explanation impresses judges more than a fast robot.

Practice questions

1. What does "ultrasonic" mean?
2. Name the four pins of the HC-SR04 and say what each does.
3. Why is the sensor mounted on a servo?
4. Why must you centre the servo before fixing the sensor to it?
5. What is the speed of sound in air at 20 °C?
6. Change 343 m/s into centimetres per microsecond.
7. Write the formula for distance from echo time, and explain the " $\div 2$ ".
8. The echo lasts 2,320 μ s. How far away is the obstacle?
9. An obstacle is 30 cm away. About how long is the echo?
10. About how fast is sound at 30 °C, using the formula in this book?
11. Name three kinds of objects the HC-SR04 finds hard to detect.
12. Why do the motor speed wires use pins 5 and 6 and not 9 or 10?
13. Why do the sensor and servo connect through a mini breadboard?
14. What does `pulseIn()` return if no echo arrives in time, and how does our code treat it?
15. What does `turnTime` control?
16. A robot keeps bumping into walls. Which setting would you raise first?
17. A robot always turns towards the more crowded side. What should you change?
18. Why should the robot never run on a 9 V PP3 battery?
19. How can you tell that your robot is suffering brown-outs?
20. What is an active buzzer?
21. Why does the HC-05's RXD pin need a voltage divider if it is connected?
22. Which characters switch the Bluetooth version into manual and automatic modes?

Answers

1. Sound too high-pitched for humans to hear, above about 20 kHz.
2. VCC (5 V power), Trig (starts a measurement), Echo (goes HIGH for the echo time) and GND (ground).
3. So the robot can measure distances to the left and right without turning its body.
4. Otherwise "straight ahead" in the code would point off to one side, and the robot would make wrong decisions.
5. About 343 metres per second.
6. $343 \text{ m/s} = 34,300 \text{ cm/s}$, and $34,300 \div 1,000,000 = 0.0343 \text{ cm per } \mu\text{s}$.
7. $\text{Distance (cm)} = \text{time } (\mu\text{s}) \times 0.0343 \div 2$. We divide by 2 because the sound goes there and back.
8. $2,320 \times 0.0343 \div 2 \approx 39.8 \text{ cm}$, so about 40 cm.
9. The sound travels 60 cm, so $60 \div 0.0343 \approx 1,749 \mu\text{s}$.
10. $331 + 0.6 \times 30 = 349 \text{ m/s}$.
11. Soft objects such as cushions and curtains, surfaces that face the sensor at a slant, and thin objects such as chair legs.
12. On an Uno, the Servo library stops PWM on pins 9 and 10, so speed control there would not work.
13. The Uno has only one 5V pin on its header, so the breadboard rails share 5 V and GND.
14. It returns 0. The code treats 0 as "far away" and returns 500.
15. How long the robot spins when it turns, which sets roughly how far it turns.
16. `STOP_DISTANCE`.
17. Swap the values of `LOOK_LEFT` and `LOOK_RIGHT`, and check the servo is centred.
18. It cannot supply the current the motors need, so the voltage drops and the Arduino keeps resetting.
19. The robot freezes or restarts, the servo snaps back to the centre, and the three-second wait happens again when motors start.
20. A buzzer that makes its own tone when powered, so it only needs switching on and off.
21. The HC-05's RXD expects about 3.3 V, and the divider brings the Arduino's 5 V signal down to that level.
22. `M` for manual and `A` for automatic.

Glossary

Active buzzer

A buzzer that makes its own tone when power is applied.

Brown-out

A dip in battery voltage that makes the Arduino restart.

Common ground

A shared GND connection between the Arduino, driver and battery.

Echo

Sound that bounces back from an object.

Echolocation

Finding objects by sending sounds and listening for echoes, as bats do.

HC-05

A Bluetooth module that acts as a wireless serial link.

HC-SR04

A low-cost ultrasonic distance sensor with Trig and Echo pins.

kHz

Kilohertz: thousands of vibrations per second.

Microsecond (μ s)

One millionth of a second.

pulseIn()

An Arduino function that measures how long a pin stays HIGH or LOW.

PWM

Pulse width modulation: switching power on and off fast to control motor speed.

Servo motor

A motor that turns to a chosen angle and holds it.

SoftwareSerial

An Arduino library that makes a serial port on ordinary digital pins.

Speed of sound

How fast sound travels: about 343 m/s in air at 20 °C.

Threshold

A set value that decides between two actions, such as the stop distance.

Trigger

The short pulse that tells the sensor to send a sound burst.

Ultrasonic

Sound above about 20 kHz, too high for humans to hear.

Voltage divider

Two resistors that bring a voltage down to a lower level.



Gurukul Technicals Institute

Computer education institute in Delhi, with authorised member centres, robotics classes and science projects, working models and DIY kits.

COURSES

Computer & Office Skills · Accounting & Business Software · Programming & Web Development · Design & Digital Skills · Hardware & Networking · Robotics & Coding · Other Programmes

PROJECTS AND KITS

Ready-made school projects · DIY and STEM kits · Robotics and coding · Components and spares · Engineering projects

CONTACT

Head office: A-10, Chief Commissioner Colony, opposite Rana Pratap Bagh, Delhi 110007

Phone: 011 2721 1512 · **Mobile and WhatsApp:** +91 97116 99511

Email: engineergurukul@yahoo.co.in

Website: engineergurukul.com · **Instagram:** [@science_projects_club](https://www.instagram.com/science_projects_club)

More free ebooks for computer courses, robotics and science projects: engineergurukul.com/resources/ebooks/

Free to read, print and share. Not for sale.