Build an **obstacle-avoiding robot** using an **ESP32 DevKit V1** and an **HC-SR04 Ultrasonic Distance Sensor**, you will need a few more components so the robot can **move and change direction** when it detects an obstacle.

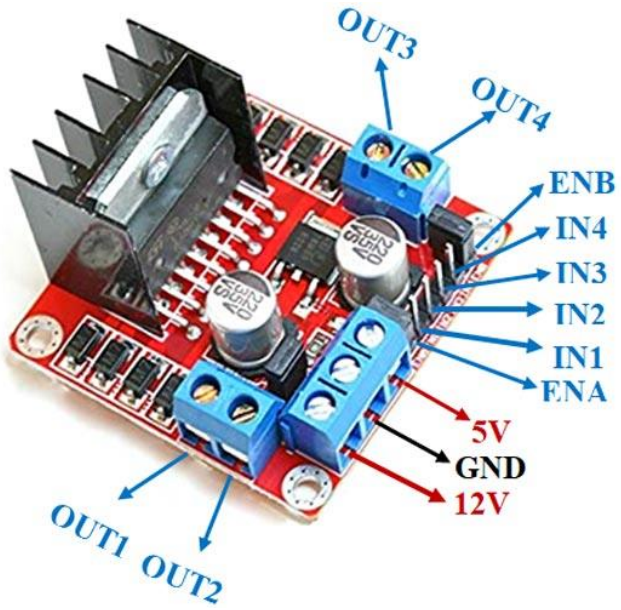
---

1 Components Required

- **ESP32 DevKit V1**

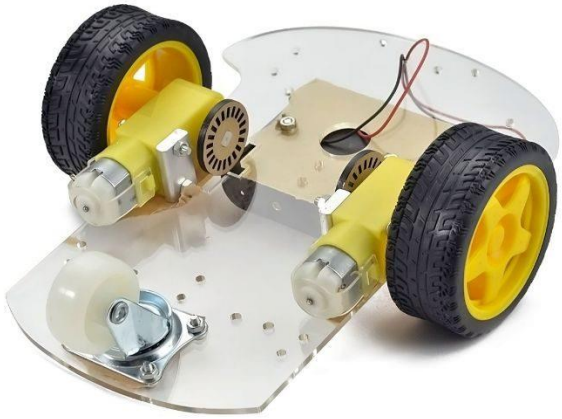
- **HC-SR04 Ultrasonic Distance Sensor**

- **L298N Motor Driver Module**



- **DC Gear Motors** (2 for left and right wheels)

Robot chassis with wheels (**2WD Two Wheel Drive DIY Kit)**



- Battery pack (7.4V–12V)
- Jumper wires

---

2 Basic Working Principle

The robot follows this logic:

1. **Ultrasonic sensor measures distance**
2. If **distance > 20 cm** → robot moves **forward**
3. If **distance < 20 cm** → robot **stops**
4. Robot **turns left or right**
5. Then continues moving forward again

So the robot **automatically avoids obstacles**.

3️⃣ Connection Overview

Ultrasonic Sensor → ESP32

Sensor Pin	ESP32
VCC	5V
GND	GND
TRIG	GPIO 5
ECHO	GPIO 18

Motor Driver → ESP32

L298N Pin	ESP32
IN1	GPIO 26
IN2	GPIO 27
IN3	GPIO 14
IN4	GPIO 12

Motors connect to **OUT1, OUT2, OUT3, OUT4** of the driver.

4️⃣ Basic Obstacle Avoiding Robot Code

```
#define TRIG_PIN 5
#define ECHO_PIN 18

#define IN1 26
#define IN2 27
#define IN3 14
#define IN4 12

long duration;
float distance;

void setup() {
  Serial.begin(115200);

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
```

```
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
}

void loop() {

  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);

  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  duration = pulseIn(ECHO_PIN, HIGH);
  distance = duration * 0.034 / 2;

  Serial.println(distance);

  if(distance > 20) {
    forward();
  }
  else {
    stopRobot();
    delay(500);
    turnRight();
    delay(500);
  }
}

void forward() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}

void stopRobot() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
}

void turnRight() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
}
```

---

5 Robot Behavior Example

Distance	Robot Action

Distance	Robot Action
40 cm	Move Forward
25 cm	Move Forward
15 cm	Stop
15 cm	Turn Right
30 cm	Move Forward again

---

6 How It Works Internally

1. **HC-SR04 Ultrasonic Distance Sensor measures distance**
2. **ESP32 DevKit V1 processes the data**
3. **L298N Motor Driver Module controls the motors**
4. Motors move the robot in different directions

Understanding **ESP32 pins** becomes easy if you group them by **purpose instead of memorizing all numbers**. I'll explain the important pins of the **ESP32 DevKit V1** and give you **simple tricks to remember them**.

---

1 Power Pins

These supply power to the board.

Pin	Meaning	Use
3V3	3.3 Volt output	Power sensors
5V / VIN	5 Volt input	Power ESP32 from battery
GND	Ground	Common return path

Easy memory trick:
"3-5-G" → 3.3V, 5V, Ground = Power group

---

2 GPIO Pins (General Purpose Pins)

GPIO means **General Purpose Input Output**.
These pins can **read sensors or control devices**.

Examples:

Pin	Use Example
GPIO2	LED
GPIO4	Sensor input
GPIO5	Trigger pin
GPIO18	Echo pin
GPIO23	Output

Example:

```
pinMode(23, OUTPUT);
digitalWrite(23, HIGH);
```

Memory trick:
Think **GPIO = "General Pins for Everything"**

---

3 UART Pins (Serial Communication)

Pin	Function
TX (GPIO1)	Transmit data
RX (GPIO3)	Receive data

Used for:

- Serial monitor
- Bluetooth modules
- Communication between devices

Memory trick:
T → Talk (Transmit)
R → Receive

---

4 PWM Pins (Motor / LED control)

PWM = **Pulse Width Modulation**
Used for:

- Motor speed control
- LED brightness
- Servo motors

Common PWM pins:

```
GPIO 2, 4, 5, 18, 19, 21, 22, 23
```

Memory trick:
“2-4-5 group and 18-23 group”

These are **safe pins for outputs**.

---

5 I2C Communication Pins

Used for:

- LCD
- OLED
- Sensors

Pin	Name
GPIO21	SDA
GPIO22	SCL

 Memory trick:
21 → SDA
22 → SCL

Just remember **21-22 pair**.

---

6 SPI Communication Pins

Used for:

- Displays
- SD cards
- Fast sensors

Pin	Function
GPIO23	MOSI
GPIO19	MISO
GPIO18	SCK
GPIO5	CS

 Memory trick:
“23-19-18-5” = SPI group

---

7 Analog Pins (ADC)

Used to read **analog values** like sensors.

Examples:

```
GPIO 32
GPIO 33
GPIO 34
GPIO 35
GPIO 36
GPIO 39
```

Example:

```
analogRead(34);
```

 Memory trick:
“32–39 range = Analog sensors”

---

8 Pins to Avoid

Some pins cause **boot problems**.

Avoid using for outputs:

```
GPIO 0
GPIO 2
GPIO 15
```

 Memory trick:
“0-2-15 = Boot pins”

---

9 Simple Pin Group Memory Map

Think of ESP32 like this:

```
Power → 3V3, 5V, GND

Communication
UART → 1,3
I2C → 21,22
SPI → 23,19,18,5

Safe Output Pins
2,4,5,18,19,21,22,23

Analog
32–39
```

□ Quick Trick Robotics Students Use □

For most **ESP32 projects**, remember just this:

```
Trigger sensor → GPIO5
Echo sensor → GPIO18
LED → GPIO23
I2C devices → 21,22
```

These pins work **in 90% of beginner projects**.