

Topic	Parking Assist	
Class Description	Students will learn how to create an ESP32 program that helps park safely using distance measuring HC-SR04 Ultrasonic Sensors.	
Class	PRO C263	
Class time	50 mins	
Goal	<ul style="list-style-type: none"> • Understand HC-SR04 Ultrasonic Sensor. • Connecting a HC-SR04 with ESP32. • Learn to write an adruino program using the HC-SR04 Ultrasonic Sensor, LED Bar graph, Buzzer and Resistor. 	
Resources Required	<ul style="list-style-type: none"> • Teacher Resources: <ul style="list-style-type: none"> ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Smartphone • Student Resources: <ul style="list-style-type: none"> ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen 	
Class structure	Warm-Up Teacher-Led Activity Student-Led Activity Wrap-Up	10 mins 15 mins 15 mins 10 mins
WARM-UP SESSION - 10 mins		
Teacher Action		Student Action
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today?		ESR: Hi, thanks! Yes, I am excited about it!

<p>Following are the WARM-UP session deliverables:</p> <ul style="list-style-type: none"> • Greet the student. • Revision of previous class activities. • Quizzes. 	<p>Click on the slide show tab and present the slides</p>
<p align="center">WARM-UP QUIZ Click on In-Class Quiz</p>	
<p>Activity Details</p> <p>Following are the session deliverables:</p> <ul style="list-style-type: none"> • Appreciate the student. • Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students. 	
<p align="center">TEACHER-LED ACTIVITY 15 mins</p>	
<p align="center">Teacher Initiates Screen Share</p>	
<ul style="list-style-type: none"> • Understand HC-SR04 Ultrasonic Sensor. • Connecting a HC-SR04 with ESP32. • Reading data from and writing data to the HC-SR04 sensor. 	
Teacher Action	Student Action
<p>Do you remember what we learnt in the previous class?</p> <p>Can you tell me how we achieved it?</p> <p>Great. You are revising very well.</p> <p>Do you have any questions from the previous class?</p> <p><i>Note: If the student has any doubts, clarify the doubts.</i></p>	<p>ESR: Yes.</p> <p>ESR: Varied.</p> <p>ESR: Varied</p>

Have you seen any vehicles parked?	ESR: Yes/No.
Do you know how it works?	ESR: Varied.
Yes. A sensor placed at the back of the vehicle gets activated as soon as the vehicle is in reverse gear. It checks the distance between the vehicle and the obstacle (like a wall in many cases) to assist the driver to park.	
How does it indicate the distance to the driver?	ESR: Graphic Display or Varying buzzer/ any sound.
Perfect. Let's learn today to create such a program.	
Before that, what does a sensor do?	ESR: Sensor detects the changes in the environment and transfers the signal to the electronic device where it can be read and managed easily by humans.
What kind of sensor will be suitable for parking?	ESR: A distance measuring sensor.
Bingo! We have Ultrasonic sensors that can measure the distance to a wide range of objects regardless of shape, color or surface texture. They are also able to measure an approaching or receding object. By using "non-contact" ultrasonic sensors, distances can be measured without damage to the object.	
It uses ultrasonic waves.	

Do you know any real life examples of ultrasonic waves?

The working is very similar to the bats flying in the dark. Do you know how bats fly in the dark?

Note: Teacher can use the below gif to explain how bats fly.



<https://s3-whjr-curriculum-uploads.whjr.online/7a46679d-f4dd-4513-9e89-6954db431101.gif>

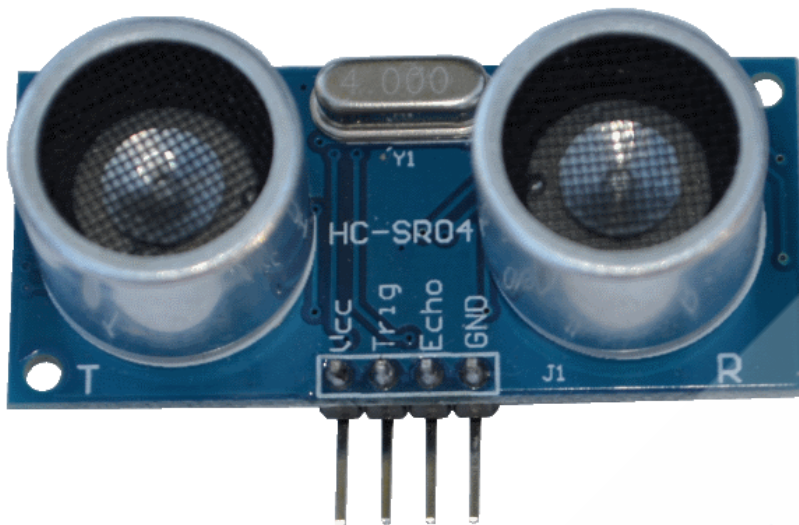
Ultrasonic sensors work exactly the same.

One such sensor is HC-SR04. It has a range from 2cm to 400cm (about an inch to 13 feet).

Below is what the sensor looks like.

ESR: Bats/ Submarine.

ESR: Varied.



<https://s3-whjr-curriculum-uploads.whjr.online/5965a4b6-e37d-4278-b084-f838dad9684d.png>

On either side the circular shapes represent the T-Transmitter and R- Receiver. Transmitter emits the ultrasonic waves and the Receiver captures these ultrasonic waves.

It has 4 pins.

1. TRIG
2. ECHO
3. VCC
4. GND

We have two pins other than **VCC** and **GND** on the HC-SR04. They are **TRIG** and **ECHO** pins. Let's understand and use them one by one.

TRIG pin is used to send the signal to the sensor that it can start transmitting the Ultrasonic wave.

The ECHO pin is used for reception of the wave. As soon as the wave is received by the receiver, the ECHO pin sends the signal.

Let's have a look at the working of the sensor.

The gif can be shown to understand the working of HC-SR04 and the TRIG and ECHO pins.

<https://s3-whjr-curriculum-uploads.whjr.online/835baf56-250c-43c8-af9a-0f02086601d6.gif>

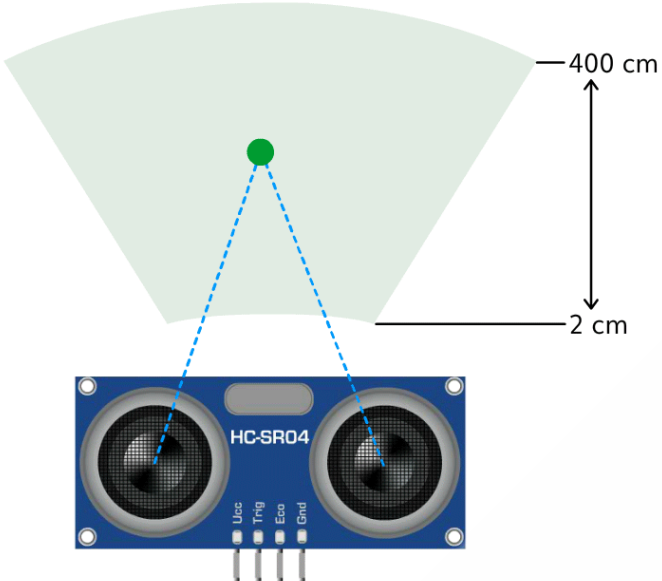
Let's also learn about blind zone. Can you tell me what is behind you without turning your back?

Why so?

Exactly. That's the Blind Zone. "The Blind Zone" is the shortest permissible sensing range. This means that no objects or targets are permitted within the minimum working area ("Blind Zone") as this would falsely trigger the sensor.

ESR: No.

ESR: I cannot see and sense it. Blind spots are areas or zones that cannot be seen by a viewer while looking at the rear view. The person must turn his or her head in order to see it.

<p>Sensing zone</p>  <p>https://s3-whjr-curriculum-uploads.whjr.online/ab36b7b8-3433-40bb-8412-350a582eee41.gif</p> <p><i>Note: Teacher can read more about the working of HC-SR04 and its application here.</i></p> <p>Let's see how this sensor works with ESP32. Are you excited?</p>	<p>ESR: Yes.</p>
<p>Open the wokwi simulator and create a new ESP32 project.</p> <p>How do we get started?</p> <p>Correct.</p> <p><i>Note: Teacher follows the below steps and involves the</i></p>	<p>ESR: We create the circuit diagram first.</p>

student as well while doing so.

Step 1: Select the components:

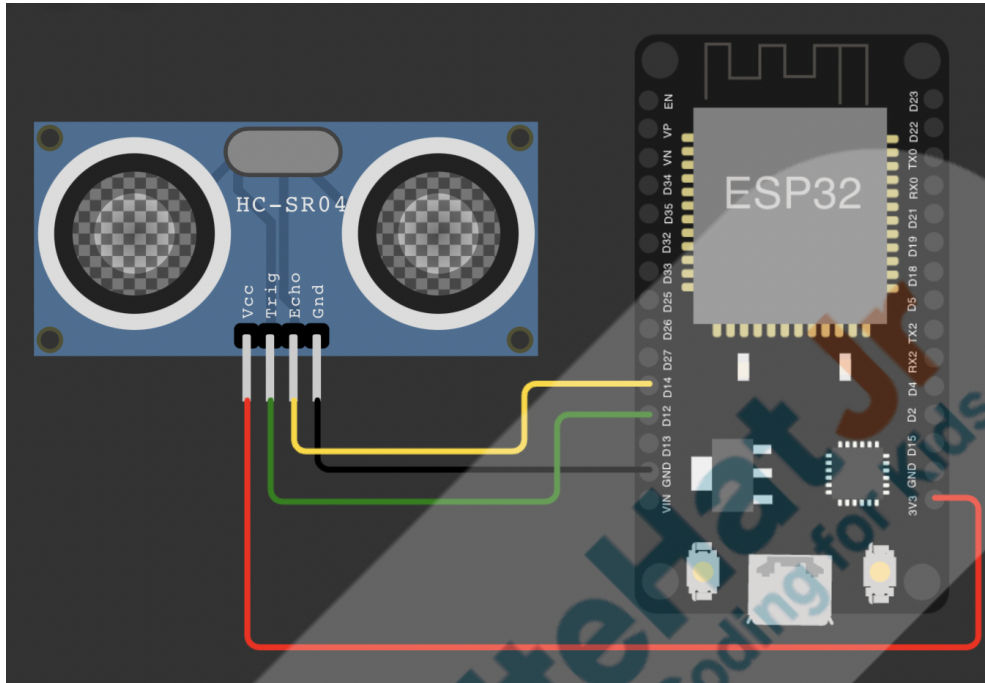
- 1 x **ESP32**
- 1 x **HC-SR04 Ultrasonic Distance Sensor**

Step 2: Make the connections:

The circuit of this project consists of an **ESP32** Controller, and a **HC-SR04 Ultrasonic Distance Sensor**.

HC-SR04	ESP32 PIN
VCC	VCC 3V
GND	GND
TRIG	GPIO 12
ECHO	GPIO 14

*Note: Wire color can be changed by **clicking over it** and selecting the color, or via the `diagram.json` file. Go to the `diagram.json` wire and change the color of the wire. Any design changes or color changes can be done via the `diagram.json` file. Keep track of the components and change the design settings.*



Next, what do we do?

How do we get started when we add a component and connect it to the controller?

1. Define the **TRIG** and **ECHO** pins. In this case they are the pins number 12 and 14 on the ESP32 Board and they are named trig_pin and echo_pin.

```
const byte trig_pin = 12;  
const byte echo_pin = 14;
```

2. Next, in the **setup()** we have to define the trig_pin as an output and the echo_pin as an Input and also start the serial communication for showing the results on the serial monitor.

ESR: Let's code.

ESR: We have to define the pins and set their mode of IO.

```
pinMode(trig_pin , OUTPUT);  
pinMode(echo_pin , INPUT);
```

3. Next, we create a function to calculate the distance to ease the work. How should it be defined?

Can you help me here?

```
int get_distance() {  
  
    return distance;  
}
```

How can the distance be calculated?

Correct. So how do we transmit the wave?

4. Let's set the trig_pin on the HIGH State for 10 μ s. What command do we use to set HIGH/LOW for a pin?

```
// sending trigger burst  
digitalWrite(trig_pin , HIGH);  
delayMicroseconds(10);  
digitalWrite(trig_pin , LOW);
```

pulseIn(): Reads a pulse (either HIGH or LOW) on a pin. For example, if the value is HIGH, pulseIn() waits for

ESR: It will not take any input but return the distance as a response. The distance is a number and hence int data type.

ESR: Student helps the teacher write the syntax.

ESR: We can calculate the difference of time taken by the ultrasonic wave to travel back and forth.

ESR: The TRIG pin will be used to do so.

ESR: digitalWrite()

the pin to go HIGH, starts timing, then waits for the pin to go LOW and stops timing. Returns the length of the pulse in microseconds or 0 if no complete pulse is received within the timeout.

5. Using the **pulseIn()** function we read the travel time and put that value into the variable "duration". This function has 2 parameters, the first one is the name of the Echo pin and the second is the state of the pulse we are reading, either HIGH or LOW.

```
// waiting for response : checking HIGH duration of echo pin
long int duration = pulseIn(echo_pin , HIGH);
long int distance = 0;
```

In this case, we need this set to HIGH, as the HC-SR04 sensor sets the Echo pin to High after sending the 8 cycle ultrasonic burst from the transmitter. This actually starts the timing and once we receive the reflected sound wave the Echo pin will go to Low which stops the timing. At the end the function will return the length of the pulse in microseconds.

6. Also clear the noise before sending the burst

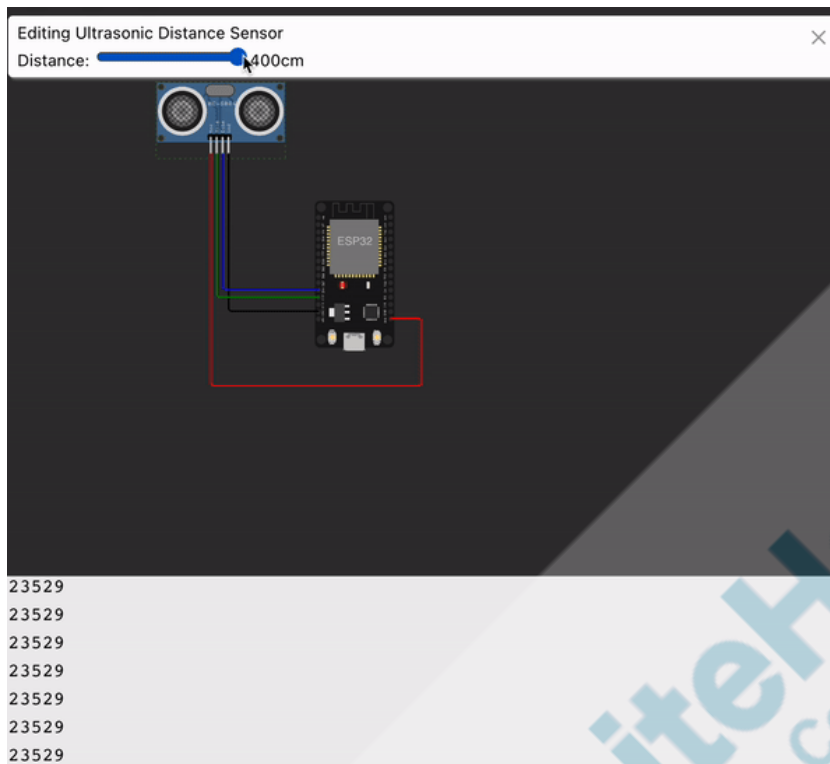
```
// getting ready before sending trigger burst
digitalWrite(trig_pin , LOW);
delayMicroseconds(2);
```

7. Next, we need to call the function that is calculating the distance in the loop().

```
int distance = get_distance();
```

We can check the output using Serial.println(). We can see the output in duration(microseconds).

Reference Output:



<https://s3-whjr-curriculum-uploads.whjr.online/54aa09d8-6907-49e7-aca2-b247c8f02bbf.gif>

How can we calculate the distance when we have the time?

Correct. What is the speed of Ultrasonic waves?

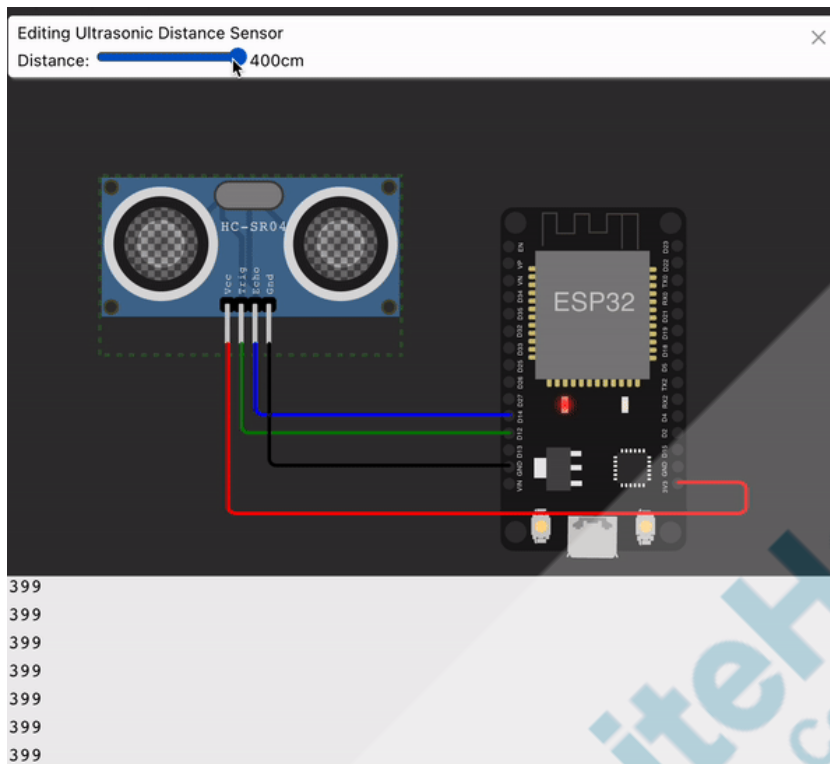
Great. For getting the distance we will multiply the duration by 0.034(speed of Ultrasonic wave) and divide it by 2 as the time known is for both to and fro.

```
float sound_speed = 0.034;
long int distance = (duration * sound_speed) / 2;
```

Output Reference:

ESR: Using the
Distance=Time * Speed
formula.

ESR: 340 milliseconds.



<https://s3-whjr-curriculum-uploads.whjr.online/45a0379c-cc60-48cc-9a4e-f7398e3b3205.gif>

Now we have the distance in cm. What should we do next?

Superb! Are you excited to do this now?

ESR: Check the distance and notify the driver using a buzzer and display if it's less than 150 cm.

ESR: Yes.

Teacher Stops Screen Share

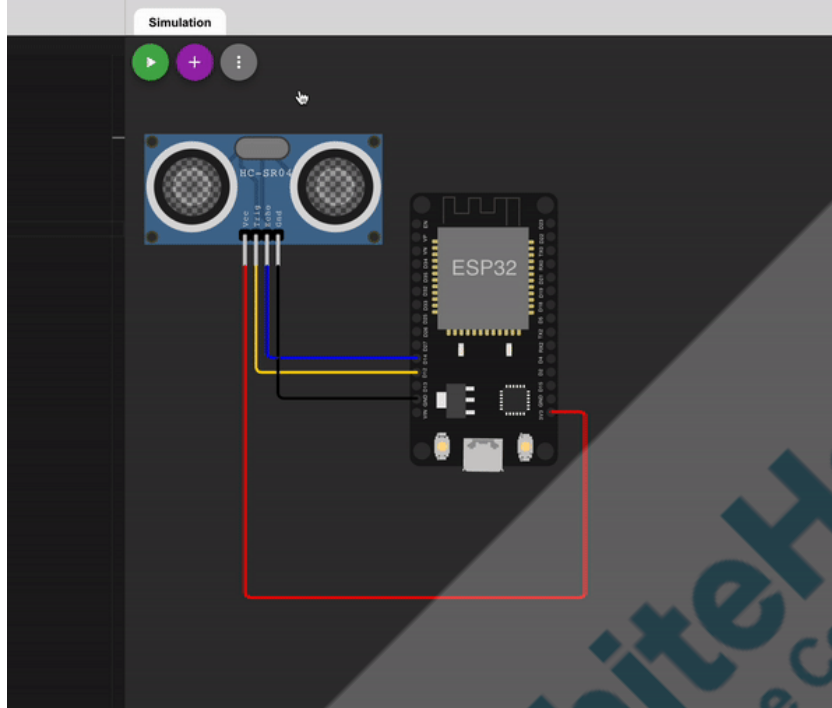
STUDENT-LED ACTIVITY- 15 mins

- Ask the student to press the ESC key to come back to the panel.
- Guide the student to start Screen Share.
- The teacher gets into Full Screen.

Student Initiates Screen Share	
<p style="text-align: center;"><u>ACTIVITY</u></p> <ul style="list-style-type: none"> • Learn about LED Bar graphs and Resistors. • Learn to use Buzzer 	
Teacher Action	Student Action
<p>Let's try. I will guide you through it.</p> <p><i>Note: Teacher helps the student.</i></p>	<p>Students download the code from Student Template 1.</p>
<p>What concept can we use for checking and deciding on the response?</p> <p>Correct. Which approach is better to use functions or do everything in a loop()?</p> <p>Great. Let's start.</p> <p>We first add the LED Bar graph and Resistor.</p> <p>Do you know what these components are and what are they used for?</p> <p><i>Note: Students have already learnt these below steps and hence involve them using the inquiry approach.</i></p> <p>LED Bar Graph: The bar graph - a series of LEDs in a line is made up of a series of LEDs in a row to allow usage of multiple LEDs together controlled by a little code. Read more about it here.</p> <p>Resistors: Resistors are used to limit the amount of current going to certain components in the circuit, such as LEDs and integrated circuits.</p> <p>The video below can be used for adding the component</p>	<p>ESR: Conditionals</p> <p>ESR: Using functions.</p> <p>ESR: Varied.</p>

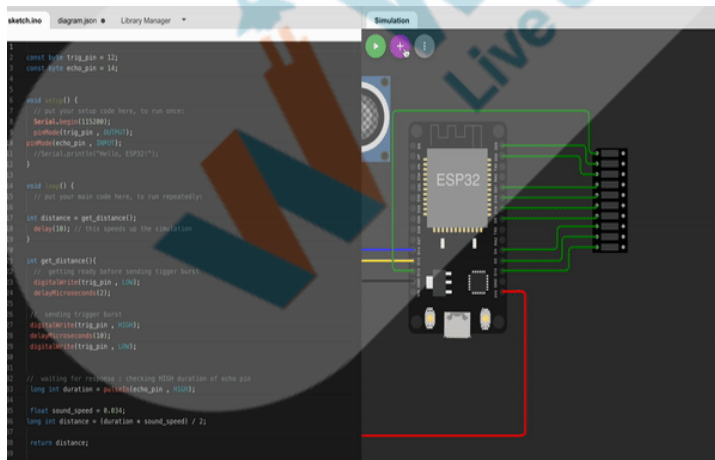
and making connections.

- **Add the LED Bar graph and make connections:**



<https://s3-whjr-curriculum-uploads.whjr.online/2c029c10-2975-4039-a2f6-3329108f944f.gif>

- **Resistor and its connection:**



<https://s3-whjr-curriculum-uploads.whjr.online/b3f36ddd-fb9a-4b2a-913a-862ba2111618.gif>

- Resistor value to update:

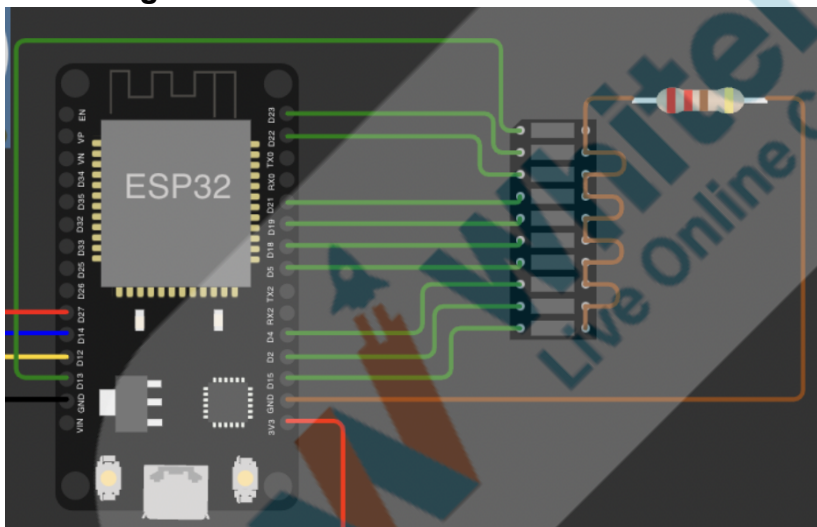
```

sketch.ino  diagram.json  Library Manager
1
2 const byte trig_pin = 12;
3 const byte echo_pin = 14;
4
5
6 void setup() {
7   // put your setup code here, to run once:
8   Serial.begin(115200);
9   pinMode(trig_pin, OUTPUT);
10  pinMode(echo_pin, INPUT);
11  //Serial.println("Hello, ESP32!");
12 }
13
14 void loop() {
15   // put your main code here, to run repeatedly:
16
17   int distance = get_distance();
18   delay(10); // this speeds up the simulation
19 }
20
21 int get_distance(){
22   // position needs to be before sending trigger burst

```

<https://s3-whjr-curriculum-uploads.whjr.online/5557e0b5-a28b-44d7-9b06-2d74f80719f2.gif>

Final image for reference:



Teacher guides the student to write the below code.
What do we do after we add the component?

How many pins are connected to the LED Bar Graph?

So how do we avoid repetition of code?

ESR: register the pins.

ESR: many/10.

ESR: Using Loop.

Let's do so.

1. Declare an array of led_pins to register the pin numbers.

```
// array of led_pins
byte led_pins[] = {13, 15, 2, 4, 5, 18, 19, 21, 22, 23};
```

Note: The numbers correspond to the GPIO pins they are connected to from bottom to top approach.

Next, what should be done?

ESR: Select the mode of the LED pins.

Correct. Let's do so.

2. We assign the pin mode for each of these LED pins. As we need to for each of them, we use loops.

```
// declaring all led pins as output
for (int i = 0; i < sizeof(led_pins); i++){
    pinMode(led_pins[i] , OUTPUT);
}
```

3. Next, we need to check if these LED pins are working or not. So, let's create a function to do so. And where will it be called?

```
indicator_testing();
```

ESR: setup() as the check needs to be done only once.

4. Let's define this method. What instruction can be used to turn on the LED?

Correct. And we again need it to be repeated for all the LEDs. Hence we use Loops.

Teacher can explain the sizeof() instruction. It can be used to get the number of elements in an array.

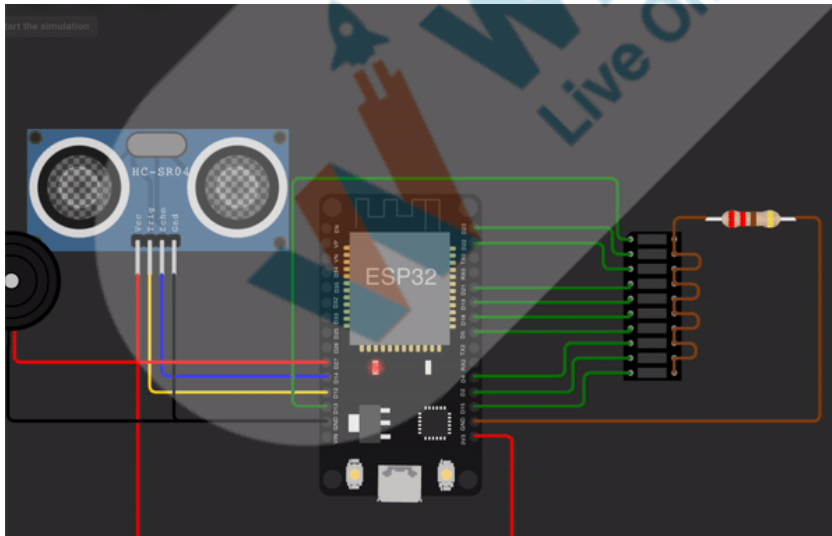
ESR: digitalWrite()

Also, starting from bottom to top, we will turn it ON

and OFF one by one. So, the first loop will turn it ON and the next loop will turn it OFF. But turning OFF will happen in reverse order i.e. top to bottom.

```
void indicator_testing(){  
    // turning all leds on  
    for (int i = 0; i < sizeof(led_pins);  
    i++){  
        digitalWrite(led_pins[i] , HIGH);  
        delay(70);  
    }  
    // all leds off  
    for (int i = sizeof(led_pins) - 1; i >= 0;  
    i--){  
        digitalWrite(led_pins[i] , LOW);  
        delay(70);  
    }  
}
```

Reference output:



<https://s3-whjr-curriculum-uploads.whjr.online/0e403c05-8bb9-4cc1-b503-9b15eaea1242.gif>

Next, what should be done?

ESR: Turn on the LEDs based on the distance.

Perfect. Let's do so by creating a function to turn on the desired number of LEDs.

5. It accepts an integer/byte as an input for the number of LEDs.

Now, depending on the led_count, we will run a loop to start lighting the LEDs from the bottom.

And also at the same time, we make sure the other LEDs are turned OFF in case they were turned on previously based on the distance.

```
void led_on(byte led_count){
    // total led which are going to be on
    for (int i = 0; i < led_count; i++){
        digitalWrite(led_pins[i] , HIGH);
    }

    // total leds which are going to be off
    for (int i = led_count; i <
sizeof(led_pins); i++){
        digitalWrite(led_pins[i] , LOW);
    }
}
```

Next, we create a function indicator() that will check the distance using conditionals and turn on the desired number of LEDs.

6. indicator()-

```
void indicator(int distance){

    if (distance <= 150 && distance >
135) led_on(1);
    else if (distance <= 135 && distance >
120) led_on(2);
    else if (distance <= 120 && distance >
105) led_on(3);
```

```

else if (distance <= 105 && distance >
90) led_on(4);
else if (distance <= 90 && distance >
75) led_on(5);
else if (distance <= 75 && distance >
60) led_on(6);
else if (distance <= 60 && distance >
45) led_on(7);
else if (distance <= 45 && distance >
30) led_on(8);
else if (distance <= 30 && distance >
15) led_on(9);
else if (distance <= 15 && distance >
0) led_on(10);
else led_on(0);
}

```

Where should we call this indicator() function?

```
indicator(distance);
```

Great work!!

Do you feel something is missing?

Bingo! And how do we add that?

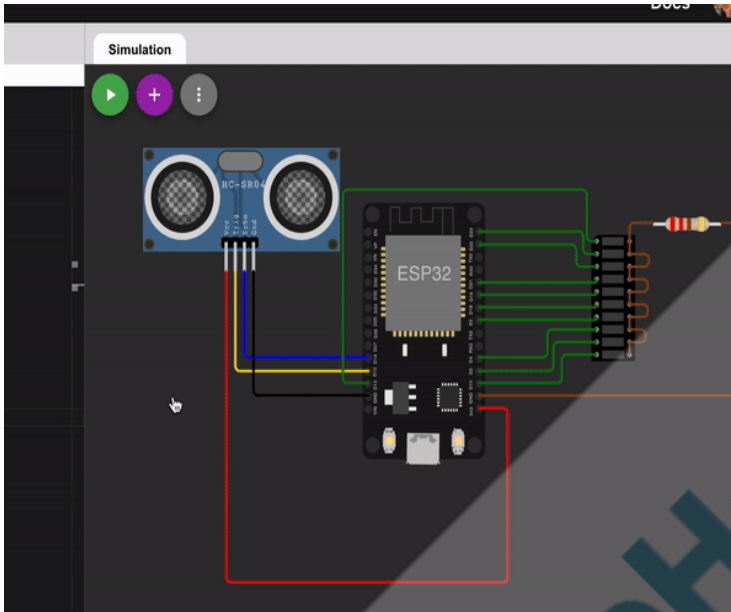
Perfect! Let's start.

- Teacher guides the student to add a new component(buzzer) and make connections.

ESR: loop() as it needs to keep checking for the distance all the time.

ESR: Yes. Sound would be amazing.

ESR: Using buzzer.



<https://s3-whjr-curriculum-uploads.whjr.online/72003ba6-7d10-4ca6-9050-35df6f60c02f.gif>

Also, add code for pin configuration at the correct places.

```
int buzzerPin = 26;
```

```
pinMode(buzzerPin, OUTPUT);
```

8. Create a function to buzz the sound:

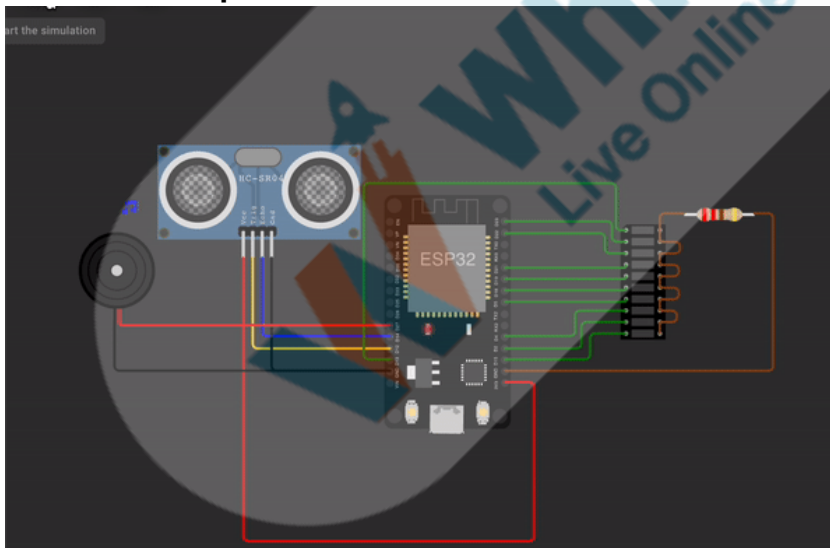
```
void buzzer(){
  for (int j = 0; j < 3; j++)
  {
    digitalWrite(buzzerPin, HIGH);
    delay(2);
    digitalWrite(buzzerPin, LOW);
    delay(2);
  }
}
```

9. In **indicator()** method, we add the below code to

play the buzzer if the distance is less than 150cm. Also, we have added the buzzer interval. buzzer_interval is the time gap between the beeps. It should be less when the gap between the vehicle and the obstacle is less, thereby beeping very fast to indicate to the user. Hence it is mapped as per the distance using map instructions. The distance from 0 to 150 cm is mapped between 0 to 1000 milliseconds.

```
int buzzer_interval = 0;
if(distance <= 150){
    // calculating buzzer interval, mapping with distance
    buzzer_interval = map(distance , 0 , 150 , 0 , 1000);
    buzzer();
    delay(buzzer_interval);
}
```

Reference Output:



<https://s3-whjr-curriculum-uploads.whjr.online/c558c213-eaf3-45df-99cc-4eeb4c61c551.gif>

<p><i>Note: While actually performing this activity with proper hardware, please make sure you give proper power to the sensor and other components. There is a chance that your sensor and other components might need 5 volts, but ESP32 can give a maximum of 3.3 volts.m</i></p>	
Teacher Guides Student to Stop Screen Share	
WRAP-UP SESSION - 10 mins	
Activity details Following are the WRAP-UP session deliverables: <ul style="list-style-type: none"> • Appreciate the student. • Revise the current class activities. • Discuss the quizzes. 	
WRAP-UP QUIZ Click on In-Class Quiz	
Activity Details Following are the session deliverables: <ul style="list-style-type: none"> • Explain the facts and trivia • Next class challenge • Project for the day • Additional Activity (Optional) 	
<u>FEEDBACK</u> <ul style="list-style-type: none"> • Appreciate and compliment the student for trying to learn a difficult concept. • Get to know how they are feeling after the session. • Review and check their understanding. 	
Teacher Action	Student Action
You get “hats-off” for your excellent work!	<p><i>Make sure you have given at least 2 hats-off during the class for:</i></p>

<p>In the next class, we will learn how microcontrollers talk to each other using different communication protocols.</p>	<div data-bbox="1019 348 1312 453">Creatively Solved Activities +10</div> <div data-bbox="1019 470 1312 575">Great Question +10</div> <div data-bbox="1019 592 1312 697">Strong Concentration +10</div>
<p align="center">PROJECT OVERVIEW DISCUSSION Refer the document below in Activity Links Sections</p>	
<p align="center">Teacher Clicks</p>	<div data-bbox="747 842 1060 932">✕ End Class</div>
<p align="center">ADDITIONAL ACTIVITIES (Optional)</p>	
<p>Additional Activities</p>	

ACTIVITY LINKS		
Activity Name	Description	Links
Teacher Activity 1	Simulator	https://wokwi.com/
Teacher Activity 2	HC-SR04	https://docs.wokwi.com/parts/wokwi-hc-sr04

Teacher Reference 1	Teacher Activity Reference Code	https://github.com/procodingclass/PRO-C263-Student-Template
Teacher Reference 2	Reference Code	https://github.com/procodingclass/PRO-C263-Teacher-Activity
Teacher Reference 3	Project	https://s3-whjr-curriculum-uploads.whjr.online/0de3c1f0-ed6a-4d3a-894b-6454a0feb9a7.pdf
Teacher Reference 4	Project Solution	https://github.com/procodingclass/PRO-C263-Project-Solution
Teacher Reference 5	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/76295bc4-c710-4bb2-a94f-ed65a9436435.pdf
Student Activity 1	Simulator	https://wokwi.com/
Student Activity 2	HC-SR04	https://docs.wokwi.com/parts/wokwi-hc-sr04
Student Activity 3	Student Activity Template Code	https://github.com/procodingclass/PRO-C263-Student-Template