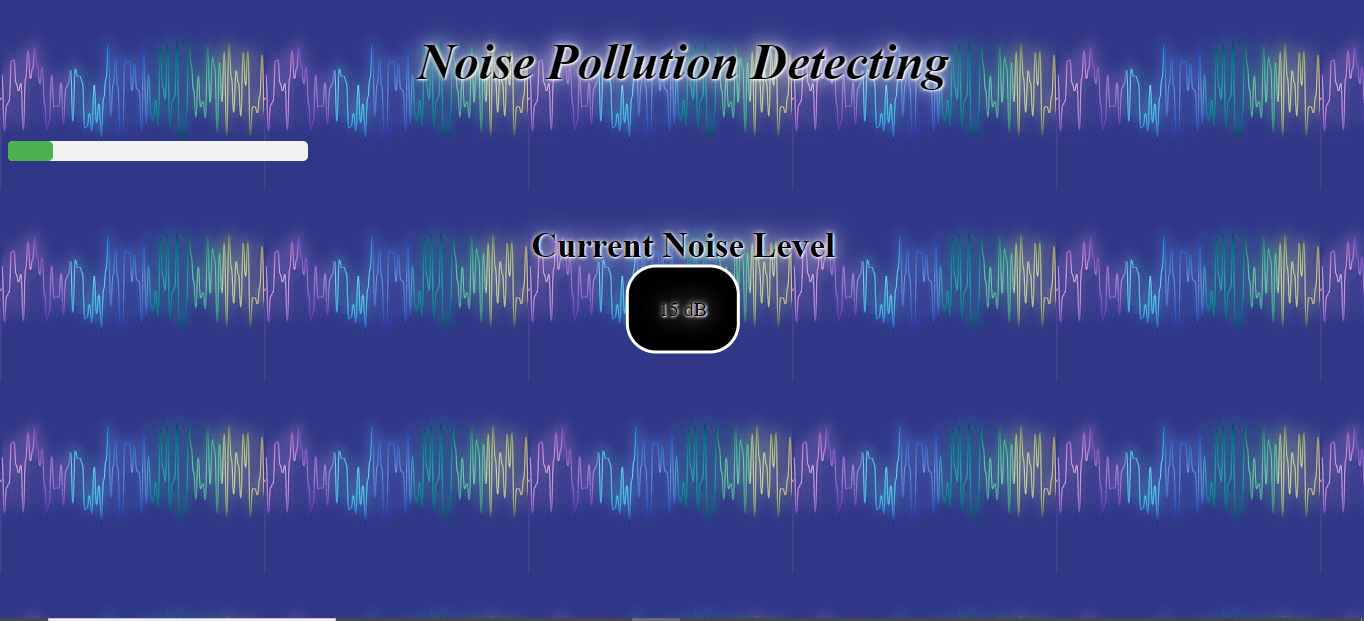
**Noise Pollution Detecting**

**Phase 4-Development Part-2:**

****

**Coding:**

**-------HTML CODE-------**

<!DOCTYPE html>

<html>

<head>

<title>Noise Level Monitor</title>

<link rel="stylesheet" href="noise.css">

<script src="noise.js"></script>

</head>

<header class="header">

<h1>Noise Pollution Detecting</h1>

</header>

<body>

<div id="noiseLevel">

<div id="noiseIndicator"></div>

</div>

<div class="txt">

<h3>Current Noise Level</h3>

<div id="noiseText">0 dB</div>

</div>

</body>

</html>

**-------css-------**

#noiseLevel {

width: 300px;

height: 20px;

background-color: #F2F2F2;

border-radius: 5px;

position: relative;

margin-top: 50px;

}

#noiseIndicator {

width: 0;

height: 100%;

background-color: #4CAF50;

border-radius: 5px;

transition: width 0.5s ease;

}

#noiseText {

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

font-size: 20px;

border-style: solid;

padding: 30px;

border-radius: 30px;

font-size: 20px;

background-color: black;

border-color: white;

}

.txt{

text-align: center;

padding-top: 30px;

color: black;

font-size: 30px;

text-shadow: 1px 1px 2px white, 0 0 25px white, 0 0 5px white;

}

h1{

text-align: center;

font-size: 50px;

color: black;

font-style: italic;

font-family: serif;

text-shadow: 1px 1px 2px white, 0 0 25px white, 0 0 5px white;

}

body{

background-image: url("img2.jpg");

}

**--------Javascript-----------**

function updateNoiseLevel(noiseLevel) {

var indicator = document.getElementById("noiseIndicator");

var text = document.getElementById("noiseText");

indicator.style.width = noiseLevel + "%";

text.innerText = noiseLevel + " dB";

}

// Sample function to update noise level every second

setInterval(function() {

var randomNoiseLevel = Math.floor(Math.random() \* 101);

updateNoiseLevel(randomNoiseLevel);

}, 1000);

**Description:**

The above project shows the current Noise level in the surrounding.The project is the Website that shows the current noise level in the surrounding by using sensors in the our iot project “Noise Pollution Monitoring System”.The sensors used in our project sences current noise level in the surrounding which displays on the websites.This happens by connecting our websites to our noise pollution monitoring system via wires.

**ESP32 Simulation**

The ESP32 is a popular WiFi and Bluetooth-enabled microcontroller, widely used for IoT Projects. Wokwi simulates the ESP32, ESP32-C3, ESP32-S2, ESP32-S3, ESP32-C6 (beta), and ESP32-H2 (alpha).

VIN GND D13 D12 D14 D27 D26 D25 D33 D32 D35 D34 VN VP EN 3V3 GND D15 D2 D4 RX2 TX2 D5 D18 D19 D21 RX0 TX0 D22 D23ESP32

**Getting Started**

You can use the ESP32 simulator to run different kinds of applications:

1. ESP32 Arduino Core projects (including ESP-IDF projects)

2. MicroPython and CircuitPython projects

3. Rust projects

4. Custom application firmware files (e.g. applications built using the ESP-IDF)

**Arduino Core**

Start from the Arduino-ESP32 Project Template, or from the ESP32 Blink Example.If you want to use third-party Arduino libraries, add a libraries.txt file with the list of libraries that you use.

**MicroPython**

Start from the MicroPython ESP32 Project Template, or from the MicroPython ESP32 Blink Example.

Note: While the simulation is running, press Ctrl+C inside the Serial Terminal to get into the MicroPython REPL. Alternatively, you can edit the Blink Example code and remove the while loop. For more information, check out the MicroPython Guide.

**Custom Application Firmware**

Open the ESP32 custom application project template, and press "F1" in the code editor. Then choose "Upload Firmware and Start Simulation…". Choose any .bin, .elf or .uf2 file from your computer and the simulation will start.

**wokwi-hc-sr04 Reference**

HC-SR04 Ultrasonic Distance Sensor

**Pin names**

|  |  |
| --- | --- |
| **Name** | **Description** |
| VCC | Voltage supply (5V) |
| TRIG | Pulse to start the measurement |
| ECHO | Measure the high pulse length to get the distance |
| GND | Ground |

**Attributes**

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Default value** |
| Distance Initial distance value, in centimeters | "400" |  |

**Operation**

To start a new distance measurement set the TRIG pin to high for 10uS or more. Then wait until the ECHO pin goes high, and count the time it stays high (pulse length). The length of the ECHO high pulse is proportional to the distance. Use the following table to convert the ECHO pulse length in microseconds into centimeters / inches:

|  |  |
| --- | --- |
| **Unit** | **Distance** |
| Centimeters | PulseMicros / 58 |
| Inches | PulseMicros / 148 |

**Setting the distance**

To change the distance while the simulation is running, click on the HC-SR04 drawing in the diagram and use the slider to set the distance value. You can choose any value between 2cm and 400cm.