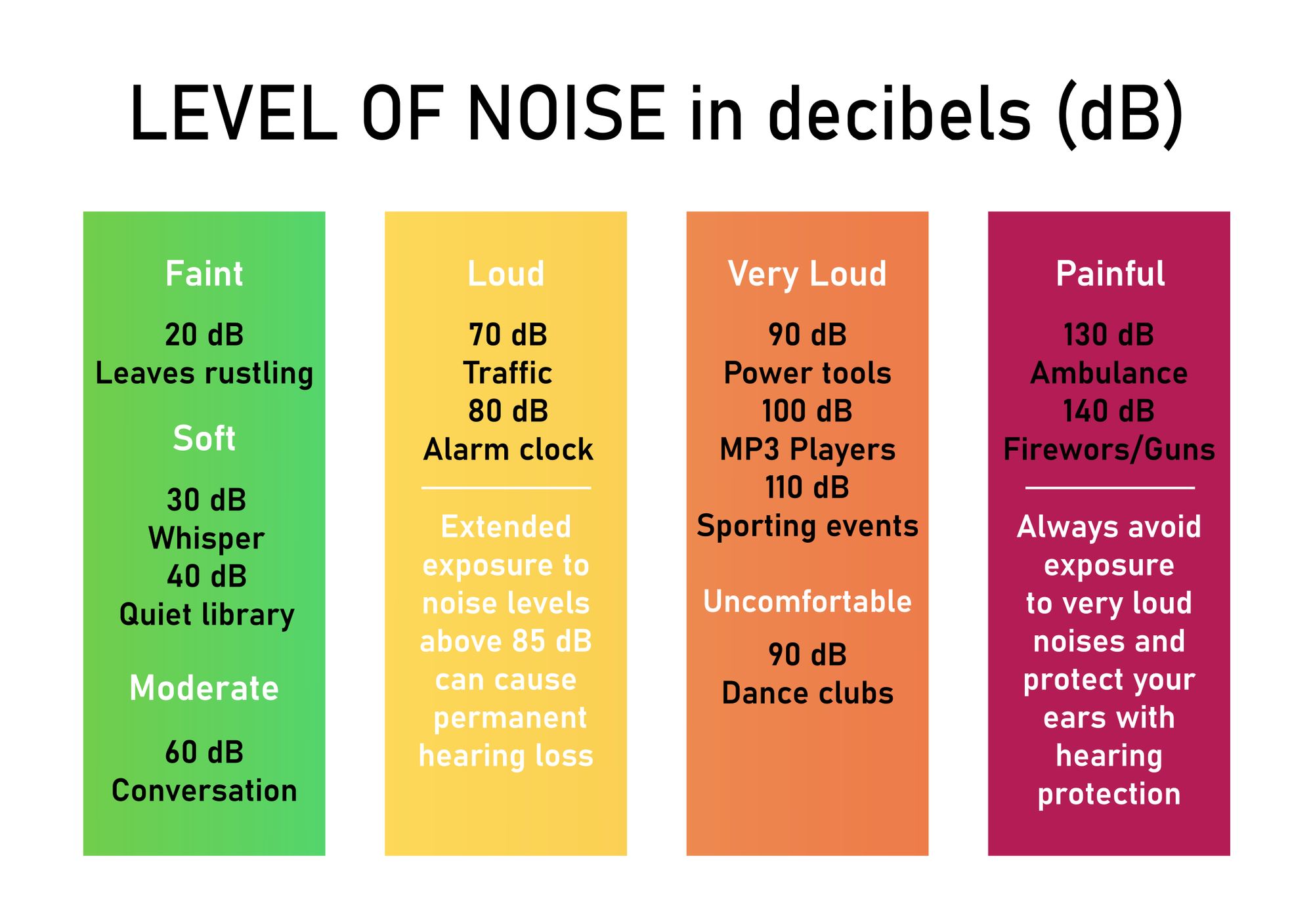
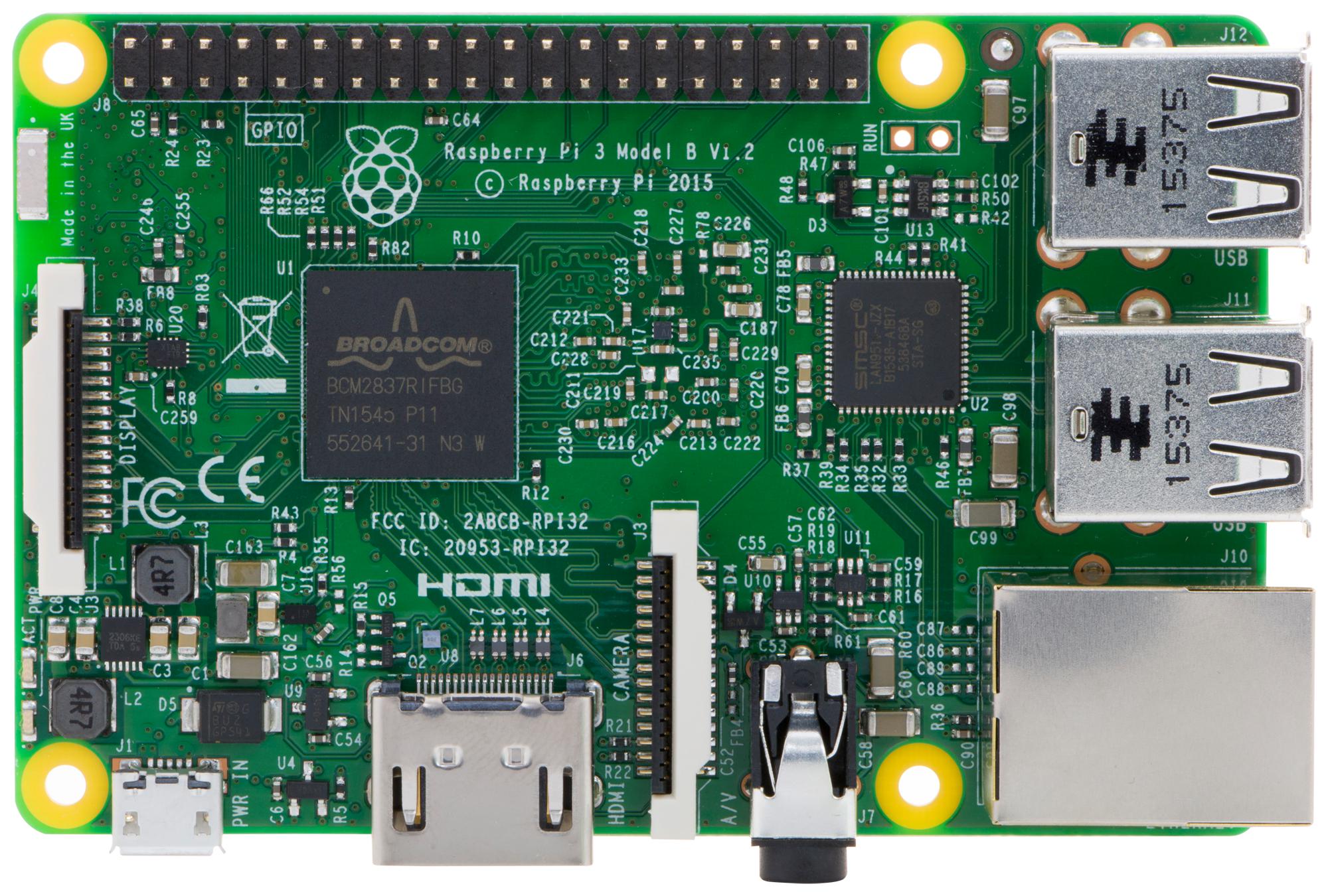
# **Noise pollution monitoring system**

**BUILDING THE PROJECT:**

The noise pollution is measured by the presence of sound pollutants present in the environment. In the initial stage, we use noise pollution sensor connected in the **Raspberry pi** .Then, a Web platform will be created in the next step. For building noise pollution monitoring system, first the type of sensor should be selected. After selecting the appropriate sensor that meet the project requirement. Choose communication protocols for connecting sensor to sever or cloud platform. MQTT and CoAP are the popular choices for IoT application due to their lightweight nature.

Write embedded software which measures the noise level and transmit to the level for the web platform. Depending on the sensor the code will be vary. Set up a server to receive the data and to store the received data. The embedded program used in the project is **Python Script**. **HTML, CSS, JavaScript** are use to develop the web platform. The data can be stored by means of SQL or NoSQL. The lowest decibel level is 20db to 50db, 60db is moderate level, 70db to 80db is loud,90db to 110db is very loud,140db is painful.





DEPLOYEMENT OF NOISE SENSOR:

In deploying the noise sensor, first define the purpose of the deploying of noise sensor, Determine what kind of data you want to collect. Select the appropriate sensor that meet your requirement in term of accuracy, measurement range, power consumption and connectivity options. Choose communication protocol based on sensor specifications and location requirements. MQTT, CoAP, and cellular network are common protocols for IoT communication. Determine the power supply for your sensor, Depending on the locations. The sensors can be placed in the Public areas such as parks, schools, Roads, industries . The sensors should be placed in the correct locations.

PYTHON PROGRAM:

import machine

import time

import urequests

import ujson

import network

import math

# Define your Wi-Fi credentials

wifi\_ssid = 'Wokwi-GUEST'

wifi\_password = ''  # Replace with the actual Wi-Fi password

# Connect to Wi-Fi

wifi = network.WLAN(network.STA\_IF)

wifi.active(True)

wifi.connect(wifi\_ssid, wifi\_password)

# Wait for Wi-Fi connection

while not wifi.isconnected():

    pass

# Define ultrasonic sensor pins (Trig and Echo pins)

ultrasonic\_trig = machine.Pin(15, machine.Pin.OUT)

ultrasonic\_echo = machine.Pin(4, machine.Pin.IN)

# Define microphone pin

microphone = machine.ADC(2)

calibration\_constant = 2.0

noise\_threshold = 60  # Set your desired noise threshold in dB

# Firebase Realtime Database URL and secret

firebase\_url = 'https://noise-pollution-bd0ab-default-rtdb.asia-southeast1.firebasedatabase.app/'  # Replace with your Firebase URL

firebase\_secret = 'nBsgyQFTqHUe4qExlaZX6VL3mpf5gn6BlpnMiuR0'  # Replace with your Firebase secret

def measure\_distance():

    # Trigger the ultrasonic sensor

    ultrasonic\_trig.value(1)

    time.sleep\_us(10)

    ultrasonic\_trig.value(0)

    # Measure the pulse width of the echo signal

    pulse\_time = machine.time\_pulse\_us(ultrasonic\_echo, 1, 30000)

    # Calculate distance in centimeters

    distance\_cm = (pulse\_time / 2) / 29.1

    return distance\_cm

def measure\_noise\_level():

    # Read analog value from the microphone

    noise\_level = microphone.read()

    noise\_level\_db = 20 \* math.log10(noise\_level / calibration\_constant)

    return noise\_level, noise\_level\_db

# Function to send data to Firebase

def send\_data\_to\_firebase(distance, noise\_level\_db):

    data = {

        "Distance": distance,

        "NoiseLevelDB": noise\_level\_db

    }

    url = f'{firebase\_url}/sensor\_data.json?auth={firebase\_secret}'

    try:

        response = urequests.patch(url, json=data)  # Use 'patch' instead of 'put'

        if response.status\_code == 200:

            print("Data sent to Firebase")

        else:

            print(f"Failed to send data to Firebase. Status code: {response.status\_code}")

    except Exception as e:

        print(f"Error sending data to Firebase: {str(e)}")

try:

    while True:

        distance = measure\_distance()

        noise\_level, noise\_level\_db = measure\_noise\_level()

        print("Distance: {} cm, Noise Level: {:.2f} dB".format(distance, noise\_level\_db))

        if noise\_level\_db > noise\_threshold:

            print("Warning: Noise pollution exceeds threshold!")

        # Send data to Firebase

        send\_data\_to\_firebase(distance, noise\_level\_db)

        time.sleep(1)  # Adjust the sleep duration as needed

except KeyboardInterrupt:

    print("Monitoring stopped")

JSON:

{

  "version": 1,

  "author": "BBGS",

  "editor": "wokwi",

  "parts": [

    {

      "type": "wokwi-esp32-devkit-v1",

      "id": "esp",

      "top": -72.1,

      "left": 52.6,

      "attrs": { "env": "micropython-20231005-v1.21.0" }

    },

    { "type": "wokwi-microphone", "id": "mic", "top": -16.98, "left": 263.79, "attrs": {} },

    {

      "type": "wokwi-hc-sr04",

      "id": "ultrasonic1",

      "top": -190.5,

      "left": 274.3,

      "attrs": { "distance": "88" }

    }

  ],

  "connections": [

    [ "esp:TX0", "$serialMonitor:RX", "", [] ],

    [ "esp:RX0", "$serialMonitor:TX", "", [] ],

    [ "mic:1", "esp:D2", "purple", [ "v0" ] ],

    [ "mic:2", "esp:GND.1", "black", [ "v0" ] ],

    [ "ultrasonic1:VCC", "esp:3V3", "red", [ "v0" ] ],

    [ "ultrasonic1:TRIG", "esp:D15", "yellow", [ "v0" ] ],

    [ "ultrasonic1:ECHO", "esp:D4", "green", [ "v0" ] ],

    [ "ultrasonic1:GND", "esp:GND.1", "black", [ "v0" ] ]

  ],

  "serialMonitor": { "display": "plotter" },

  "dependencies": {}

}