**NOISE POLLUTION MONITORING**

**PHASE : 05 Project Documentation & Submission**

**S.Gokul(210521205017)**

**Project Objectives:**

The core objective of this noise pollution monitoring project is to address the pervasive issue of

noise pollution in urban environments by deploying state-of-the-art IoT sensors to gather real-

time noise level data across various public areas. Through the development of an accessible and

user-friendly noise pollution information platform or mobile application, the project aims to

bridge the gap between data collection and public awareness. By processing and visualizing this

data using tools like Python, it enables individuals and communities to make data-driven

decisions regarding noise exposure, facilitating the identification of noise pollution trends and

areas requiring mitigation efforts. Additionally, the project strives to promote active engagement

within the community, fostering a collective commitment to noise reduction and improved urban

living conditions through education and awareness initiatives.

**IOT Sensor Deployment:**

Deploying sensors for noise pollution monitoring involves the several steps are following as,

**Sensor Selection:** Choose appropriate sensors for the task. Microphones or sound level meters are commonly used for noise monitoring. Select sensors that can capture data accurately and reliably.

**Sensor Placement:** Position sensors strategically. Consider factors such as distance from potential noise sources, height, and obstructions. Place them in locations representative of the area being monitored.

**Data Collection:** Set up the sensors to collect data continuously or at specific intervals. Ensure the sensors are calibrated correctly and have sufficient power or connectivity for data transmission.

**Data Processing and Analysis:** Once data is collected, process and analyze it. This can involve identifying patterns, high noise events, or correlating noise levels with specific activities or times of day.

**Python script :**

Import smtplib

From email.mime.text import MIMEText

From email.mime.multipart import MIMEMultipart

Import struct

Import math

Import numpy as np

Import http.client

From scipy.fftpack import fftfreq, fft

Import urllib.request

Import scipy.io.wavfile

Import json

Import sys

Import getopt

Def print\_usage(name):

Print(“usage: python”, name, “-l number.number.number.number:port”)

Def get\_streaming\_link(argv):

If len(argv) == 1:

Print\_usage(name = argv[0])

Sys.exit(2)

Try:

Opt\_vals, args = getopt.getopt(argv[1:], ‘l:’, [‘link=’])

Except getopt.GetoptError:

Print\_usage(argv[0])

Sys.exit(2)

For opt, val in opt\_vals:

If opt in (‘-l’, ‘—link’):

Return ‘http://’ + val

Print\_usage(argv[0])

Sys.exit(2)

Streaming\_time = 4 # in seconds

Streaming\_length = streaming\_time \* 4

Frame\_rate = 22500

Number\_of\_frames = streaming\_length \* frame\_rate // 4

File\_location = “soundtemp.mp3”

IoT\_platform\_url = “uBeac’s IoT platform address” #for example

‘askini.hub.ubeac.io’

IoT\_platform\_gateway = “uBeac’s gateway” # for example ‘/lanmic’

Def get\_audio®:

Audio = r.read(frame\_rate)

For I in range(streaming\_length – 1):

Audio += r.read(frame\_rate)

Return audio

Def get\_rate\_and\_data(audio):

Format\_float = ‘&lt;’ + str(number\_of\_frames) + ‘I’

Result = struct.unpack(format\_float, audio)

Abs\_numbers = np.abs(np.array(result))

Max\_number = np.max(abs\_numbers)

Audio\_data = (abs\_numbers/max\_number).astype(np.float32)

Scipy.io.wavfile.write(file\_location, frame\_rate, audio\_data)

Rate, data = scipy.io.wavfile.read(file\_location)

Return rate, data

Def get\_amplitude(data):

Rms\_amplitude = np.sqrt(np.mean(np.square(data)))

Log\_of\_rms\_amp = 20 \* math.log10(rms\_amplitude)

Return -1 \* log\_of\_rms\_amp

Def prep\_sensor\_data(id, data):

Sensor\_data = {

‘id’: id,

‘data’: data

}

Return sensor\_data

Def get\_max\_freq(data, rate):

Frequencies = fftfreq(data.shape[0], 1/rate)

Freqspos = frequencies[:frequencies.size // 2]

Fft\_of\_data = fft(data)

Fftabs = abs(fft\_of\_data)[:frequencies.size // 2]

#peakfreq = np.max(fftabs) #NOTE: may produce bug

Max\_idx = np.argmax(fftabs)

Max\_freq = freqspos[max\_idx]

Return max\_freq, max\_idx

Def send\_data\_to\_IoT\_platform(sensors):

Device = [{

‘id’: “Android Microphone”,

‘sensors’: sensors

}]

Connection = http.client.HTTPSConnection(IoT\_platform\_url)

Connection.request(‘POST’, IoT\_platform\_gateway, json.dumps(device))

Response = connection.getresponse()

Print(response.read().decode())

Import time

Time\_previous = 0

Def send\_notification(amplitude):

Global time\_previous

Time\_now = time.time()

If time\_now – time\_previous &lt; 60:

Return

Elif float(amplitude[‘data’][‘Amplitude’]) &lt; 22:

Return

Time\_previous = time\_now

Mail\_content = ‘’’Hello,

Noise detected!!!!

Cheers

#The mail addresses and password

Sender\_address = ‘your@email.address’

With open(‘pass.txt’, ‘r’) as f:

Sender\_pass = f.read()

Receiver\_address = ‘your@email.address’

#Setup the MIME

Message = MIMEMultipart()

Message[‘From’] = sender\_address

Message[‘To’] = receiver\_address

Message[‘Subject’] = ‘Noise Detection Notification’ #The subject line

#The body and the attachments for the mail

Message.attach(MIMEText(mail\_content, ‘plain’))

#Create SMTP session for sending the mail

Session = smtplib.SMTP(‘smtp.gmail.com’, 587) #use gmail with port

#session = smtplib.SMTP\_SSL(‘smtp.gmail.com’)

Session.starttls() #enable security

Session.login(sender\_address, sender\_pass) #login with mail\_id and

password

Text = message.as\_string()

Session.sendmail(sender\_address, receiver\_address, text)

Session.quit()

Print(‘email notification sent’)

Link = get\_streaming\_link(sys.argv)

With urllib.request.urlopen(link) as r:

Print(‘the program is running’)

\_ = r.read(44) # skip header

While True:

Audio = get\_audio®

Rate, data = get\_rate\_and\_data(audio)

Log\_of\_rms\_amp = get\_amplitude(data)

Amplitude = prep\_sensor\_data(“Average Amplitude”,

{“Amplitude”:str(log\_of\_rms\_amp)})

Send\_notification(amplitude) # sends an email when apmplitude is high

Max\_freq, max\_freq\_idx = get\_max\_freq(data, rate)

Frequency = prep\_sensor\_data(“Frequency”, {“Max Frequency” :

str(max\_freq)})

Peak = prep\_sensor\_data(“Max Peak”, {“Amplitude” : str(max\_freq\_idx)})

Sensors = [amplitude, frequency, peak]

Send\_data\_to\_IoT\_platform(sensors)

**Platform:**

Here we are creating a webpage for indicating the noise as sound as using html, css and javascript.

**HTML, CSS, and JavaScript:**

**HTML:** Create the structure of your web page, including headers, content areas, and

placeholders for noise level data.

**CSS:** Style your web page for a visually appealing and user-friendly design. Consider responsive

design to ensure it works on different devices.

**JavaScript:** Use JavaScript to fetch and display real-time noise level data. You may need to use

technologies like WebSockets or AJAX for this purpose.

**Code:**

**Index.html**

!DOCTYPE html&gt;

&lt;html&gt;

&lt;head&gt;

&lt;title&gt;Real-Time Noise Monitor&lt;/title&gt;

&lt;link rel=&quot;stylesheet&quot; type=&quot;text/css&quot; href=&quot;styles.css&quot;&gt;

&lt;/head&gt;

&lt;body&gt;

&lt;div class=&quot;container&quot;&gt;

&lt;h1&gt;Noise Level Monitor&lt;/h1&gt;

&lt;div class=&quot;noise-display&quot;&gt;

&lt;h2&gt;Current Noise Level:&lt;/h2&gt;

&lt;div id=&quot;noise-level&quot;&gt;Loading...&lt;/div&gt;

&lt;/div&gt;

&lt;div class=&quot;controls&quot;&gt;

&lt;label for=&quot;threshold&quot;&gt;Set Alert Threshold (dB):&lt;/label&gt;

&lt;input type=&quot;number&quot; id=&quot;threshold&quot; placeholder=&quot;e.g., 70&quot;&gt;

&lt;/div&gt;

&lt;/div&gt;

&lt;script src=&quot;script.js&quot;&gt;&lt;/script&gt;

&lt;/body&gt;

&lt;/html&gt;

**Styles.css**

body {

font-family: Arial, sans-serif;

background-color: #f2f2f2;

}

.container {

max-width: 800px;

margin: 0 auto;

padding: 20px;

background-color: #fff;

border-radius: 5px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);

}

.noise-display {

text-align: center;

}

.controls {

margin-top: 20px;

}

input {

width: 100px;

}

**Script.js**

document.addEventListener(&quot;DOMContentLoaded&quot;, function () {

const noiseLevelElement = document.getElementById(&quot;noise-level&quot;);

const thresholdInput = document.getElementById(&quot;threshold&quot;);

// Simulate real-time noise data updates (replace with actual data source).

const updateNoiseData = () =&gt; {

const randomNoiseLevel = Math.floor(Math.random() \* 101); // Random value between 0

and 100 dB.

noiseLevelElement.textContent = randomNoiseLevel + &quot; dB&quot;;

// Check if the noise level exceeds the user-defined threshold.

const threshold = parseFloat(thresholdInput.value);

if (!isNaN(threshold) &amp;&amp; randomNoiseLevel &gt; threshold) {

alert(&quot;Noise level exceeds the threshold!&quot;);

}

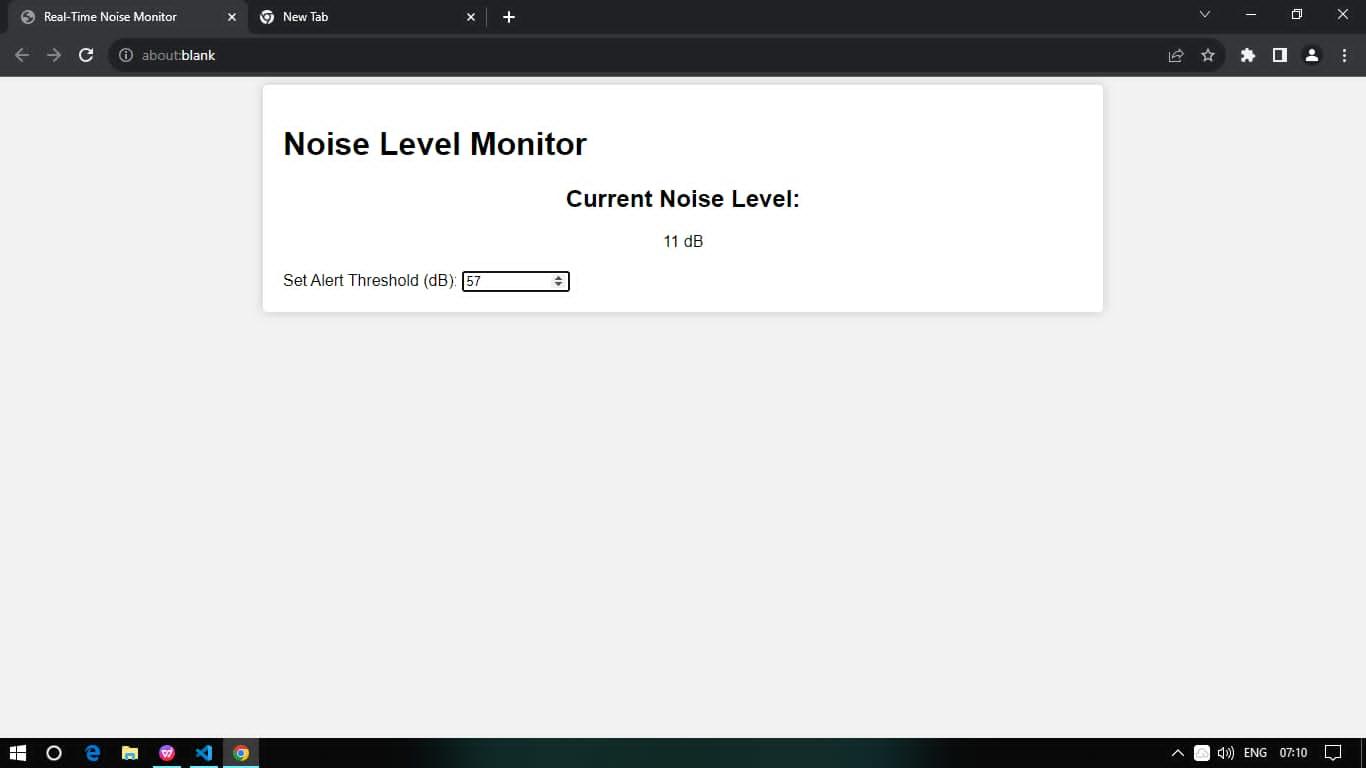
};

// Update noise data every 5 seconds (adjust as needed).

setInterval(updateNoiseData, 5000);

});

**OUTPUT**



**Schematic Diagram**

**Components & Sensors Selection :**

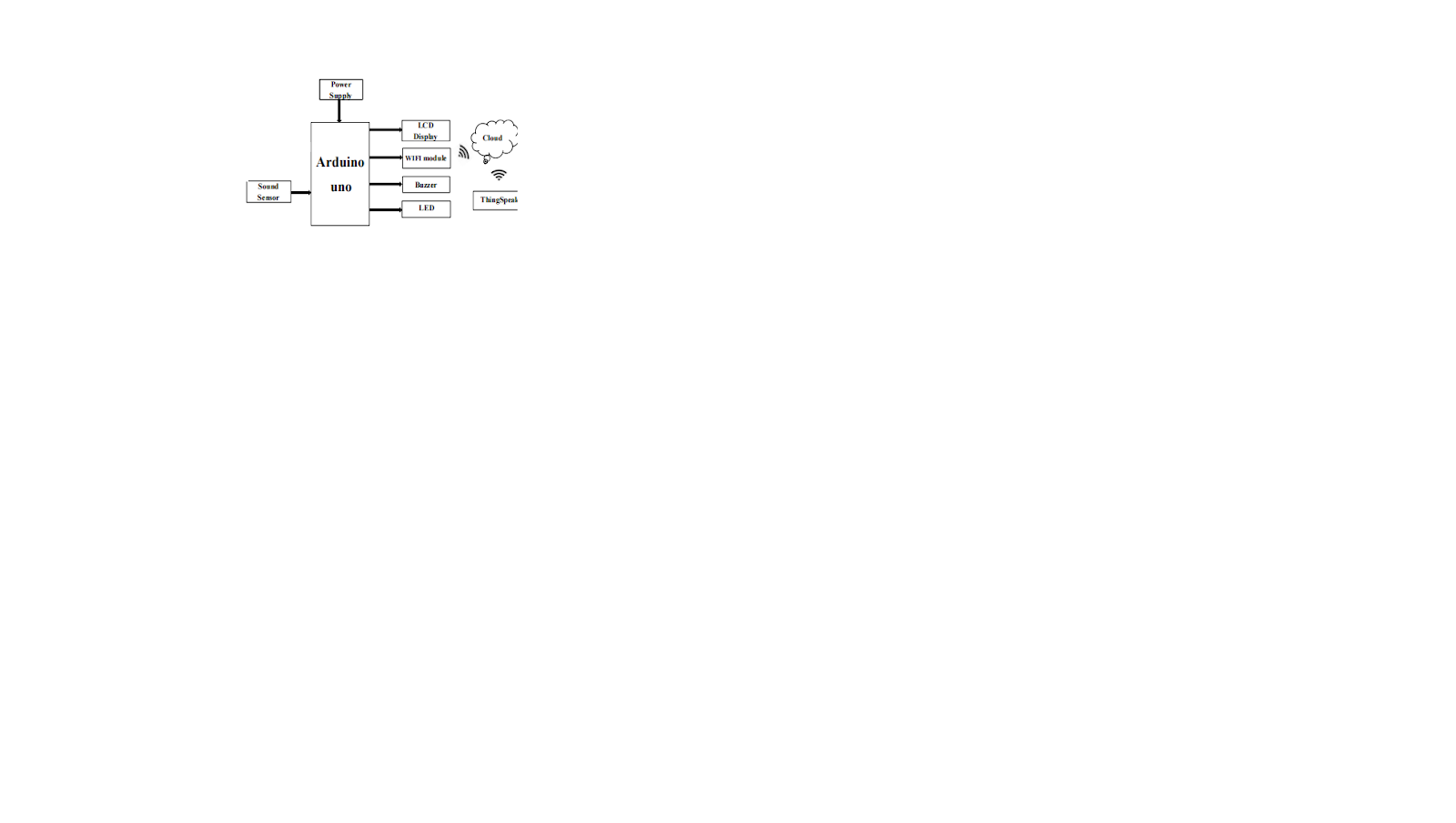
* Microphone /Sound sensor
* EP8266 Node MCU
* 16\*2 LED module
* Bread boards
* Connecting wires

**Used sensors :**

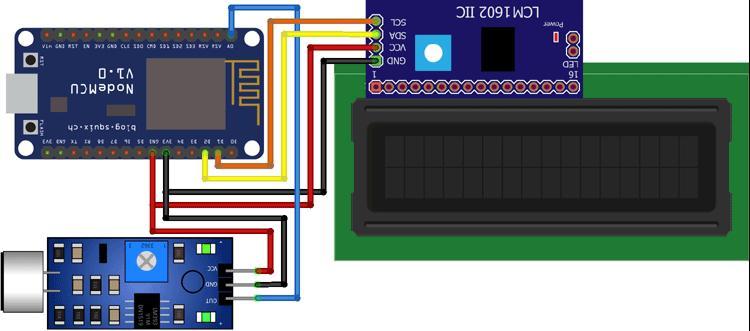
Microphone/Sound sensor : The microphone based sound sensor is used to detect sound. It gives a measurement of how loud a sound is.(50HZ-10khz) sound sensor is defined as a module that detects sound waves through its intensity and converts them to electrical signals.



**Block Daigram:**

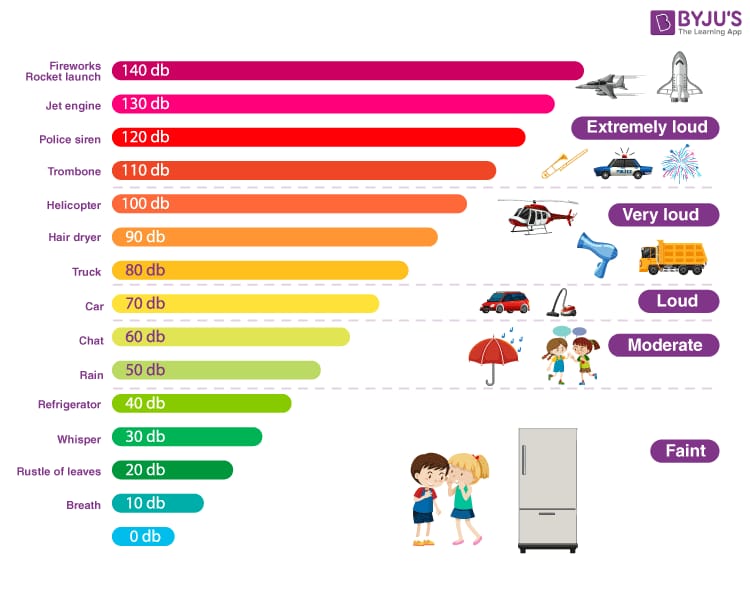
\

**Circuit Diagram:**



**Conclusion**

A real-time noise level monitoring system helps promote public awareness by providing immediate, accurate data on noise levels in specific areas. This information can be accessible to the public, raising awareness about noise pollution and its impact on health. By understanding the current noise levels, individuals and authorities can take proactive measures to mitigate noise pollution, such as identifying high-noise areas for potential regulation, adjusting infrastructure or city planning, and educating the public on noise-reducing behaviors. This system's real-time data empowers individuals and communities to make informed decisions, ultimately contributing to more effective noise pollution mitigation efforts.



**Fig: Types of Noise in Environment**