**NOISE POLLUTION MONITORING**

**PHASE 5: DOCUMENT SUBMISSION**

**810021106068:M.SARAVANA GANESAN**

**PROJECT DEFINITION:**

This project involves installing IoT sensors in various public areas. These sensors are designed to measure and record noise levels in their respective locations. The primary goal is to provide real time noise level data. The recorded data will be available to the public through a mobile app or any platform. Each sensor is capable of detecting and measuring sound and noise levels in its vicinity. These sound meters are connected to Arduino boards, which act as microcontrollers to gather data from the sensors. These data are uploaded to the cloud and are analyzed for noise pollution. This analysis involves processing and interpreting the raw sound level measurements to determine the degree of noise pollution in the area. The results of the data analysis, which represent noise pollution levels, are made available through a mobile app. Users can access this app to check the noise levels in different public areas.

**DESIGN THINKING:**

**Hardware setup:**

1. Select components:
   1. Choose IoT sensors with sound level measurement like decibelmeter.
   2. Acquire arduino board and ESP8266 module.
   3. Gather power supplies, cables and protective cover for the hardware
2. Connect sound sensor to arduino:
   1. Connect the sensors to the Arduino board.
   2. Write arduino code to read sound level from the sensors and transmit this data to the ESP8266 module.
3. ESP8266 integration:
   1. Connect the ESP8266 module to connect to wifi networks.
   2. Write Arduino code to establish a wifi connection and sensor data to a cloud server.

**Cloud Setup:**

1. Choosing microsoft azure or google cloud for data storage and analysis.
2. Set up an account and create a new project or environment.
3. Create a cloud based database to store the incoming noise level data.
4. Design a data ingestion system that can receive data from the ESP8266 module and store it in the cloud.
5. Writing scripts or functions to analyze the data stored in the cloud.
6. Calculate noise pollution level based on collected sound data.

**Mobile app development:**

1. Create the app’s user interface. Design screens to display noise pollution data in real-time.
2. Connect the app to the cloud service where the noise data is stored.
3. Write code to fetch real time noise pollution data from the cloud and display it.
4. Ensure the app works correctly and debug any issues.

**Deployment and accessibility:**

1. Install the IoT sensors in the selected public areas. Ensuring that they are securely mounted and protected from environmental factors.
2. Publish the mobile app on app stores to make it accessible to the public.
3. Ensuring that the data collected and analyzed in the cloud is accessible to the public through the mobile app or web platform.
4. Regularly monitor the system’s performance and update the app as needed.

**COMPONENTS REQUIRED:**

ESP8266 NodeMCU Board

Microphone sensor

16\*2 LCD Module

Breadboard

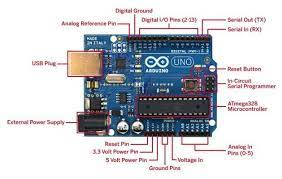
Connecting wires

**MICROPHONE BASED SOUND DETECTOR:**

The microphone based sound sensor is used to detect sound. It gives a measurement of how loud a sound is. The sound sensor module is a small board that mixes a microphone (50Hz-10kHz) and a few processing circuits to convert sound waves into electrical signals. This electrical signal is fed to on-board LM393 High Precision Comparator to digitise it and is made available at the OUT pin.

The module features a built-in potentiometer for sensitivity adjustment of the OUT signal. We will set a threshold by employing a potentiometer. So that when the amplitude of the sound exceeds the edge value, the module will output LOW, otherwise, HIGH. Apart from this, the module has two LEDs. The facility LED will illuminate when the module is powered. The Status LED will illuminate when the digital output goes LOW.

**ARDUINO UNO:**

****

Arduino is an 8 bit microcontroller board based on the ATmega328P. The operating voltage is 5V. It has 14 pins digital input output pins (Of which can be used 6 asPWM output)Oscillator frequency is 16 MHz It contains everything needed to support the microcontroller simply connect it to a computer with USB cable. It has 6 analog input pins.

**FEATURE:**

• Operating voltage is5v.

• DC current per input pin is 40mA.

• Clock speed16MHz.

• DC current for the 3.3v pin is 50mA.

• SPAM 2 KB

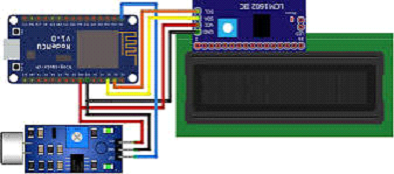
• EEPROM 1KB

**SOUND SENSOR MODULE:**

****

The sound sensor only has three pins: VCC, GND & OUT. VCC pin supplies power for the sensor & works on 3.3V to 5V. OUT pin outputs HIGH when conditions are quiet and goes LOW when sound is detected.

**CIRCUIT DIAGRAM:**

****

**PYTHON CODE FOR CREATING APPLICATION:**

import sys

from phant import Phant

from twilio.rest import TwilioRestClient

from time import sleep

account\_sid = ""

auth\_token = ""

client = TwilioRestClient(account\_sid, auth\_token)

p = Phant(public\_key='q5JMKnDJKXCMnjbYr0lG', fields=['temp'], private\_key='')

while(True):

data = p.get()

print("Latest Loudness Value is: {}".format(data[0]['temp']))

if float(data[0]['temp']) > 1500:

message = client.messages.create(body="Loudness: {0}\nD313

Room is making noise, please take action".format(data[0]['temp']),

to="+919650055244", # Replace with your phone number

from\_="+12018905183") # Replace with your Twilio number

print (message.sid)

sleep(15)

**ARDUINO CODE FOR DECIBEL METER AND LCD DISPLAY:**

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

LiquidCrystal\_I2C lcd = LiquidCrystal\_I2C(0x27, 16, 2);

const int sampleWindow = 50;

unsigned int sample;

#define SENSOR\_PIN A0

#define PIN\_QUIET 3

#define PIN\_MODERATE 4

#define PIN\_LOUD 5

void setup ()

{

pinMode (SENSOR\_PIN, INPUT);

pinMode(PIN\_QUIET, OUTPUT);

pinMode(PIN\_MODERATE, OUTPUT);

pinMode(PIN\_LOUD, OUTPUT);

digitalWrite(PIN\_QUIET, LOW);

digitalWrite(PIN\_MODERATE, LOW);

digitalWrite(PIN\_LOUD, LOW);

Serial.begin(115200);

lcd.begin();

lcd.backlight();

lcd.clear();

}

void loop ()

{

unsigned long startMillis= millis();

float peakToPeak = 0;

unsigned int signalMax = 0;

unsigned int signalMin = 1024;

while (millis() - startMillis < sampleWindow)

{

sample = analogRead(SENSOR\_PIN);

if (sample < 1024)

{

if (sample > signalMax)

{

signalMax = sample;

}

else if (sample < signalMin)

{

signalMin = sample;

}

}

}

peakToPeak = signalMax - signalMin;

int db = map(peakToPeak,20,900,49.5,90);

lcd.setCursor(0, 0);

lcd.print("Loudness: ");

lcd.print(db);

lcd.print("dB");

if (db <= 60)

{

lcd.setCursor(0, 1);

lcd.print("Level: Quite");

digitalWrite(PIN\_QUIET, HIGH);

digitalWrite(PIN\_MODERATE, LOW);

digitalWrite(PIN\_LOUD, LOW);

}

else if (db > 60 && db<85)

{

lcd.setCursor(0, 1);

lcd.print("Level: Moderate");

digitalWrite(PIN\_QUIET, LOW);

digitalWrite(PIN\_MODERATE, HIGH);

digitalWrite(PIN\_LOUD, LOW);

}

else if (db>=85)

{

lcd.setCursor(0, 1);

lcd.print("Level: High");

digitalWrite(PIN\_QUIET, LOW);

digitalWrite(PIN\_MODERATE, LOW);

digitalWrite(PIN\_LOUD, HIGH);

}

delay(200);

lcd.clear();

}

**HTML CODE FOR WEB PAGE DESIGN**

<!DOCTYPE html>

<html>

<head>

<!-- EXTERNAL LIBS-->

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>

<script src="https://www.google.com/jsapi"></script>

<!-- EXAMPLE SCRIPT -->

<script>

// onload callback

function drawChart() {

var public\_key = 'q5JMKnDJKXCMnjbYr0lG';

// JSONP request

var jsonData = $.ajax({

url: 'https://data.sparkfun.com/output/' + public\_key + '.json',

data: {page: 1},

dataType: 'jsonp',

}).done(function (results) {

var data = new google.visualization.DataTable();

data.addColumn('datetime', 'Time');

data.addColumn('number', 'Loudness');

$.each(results.slice(0,80), function (i, row) {

data.addRow([

(new Date(row.timestamp)),

parseFloat(row.temp) ]);

});

var chart = new google.visualization.LineChart($('#chart').get(0));

chart.draw(data, {

title: 'SNU Library Noise Level Monitor'

});

});

}

// load chart lib

google.load('visualization', '1', {

packages: ['corechart']

});

// call drawChart once google charts is loaded

google.setOnLoadCallback(drawChart);

</script>

</head>

<body>

<div id="chart" style="width: 100%;"></div>

</body>

</html>

**CONTRIBUTION TO PUBLIC AWARENSS:**

Real-time noise level monitoring systems raise public awareness of noise pollution by providing immediate feedback, promoting accountability for noise sources, and enabling data-driven behavior changes. They facilitate efficient complaint resolution and support policy advocacy. The continuous data collected aids in research, noise map creation, and education on noise pollution's impacts. Overall, these systems empower communities to address and reduce noise pollution, benefiting public health and the environment.