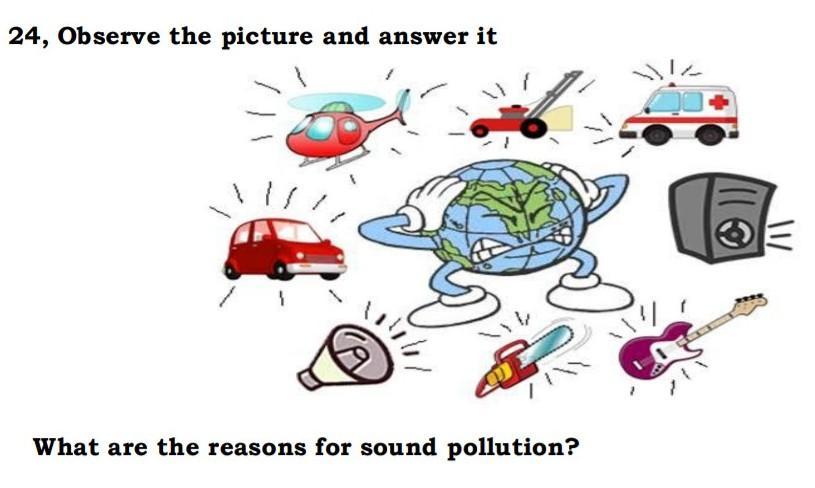
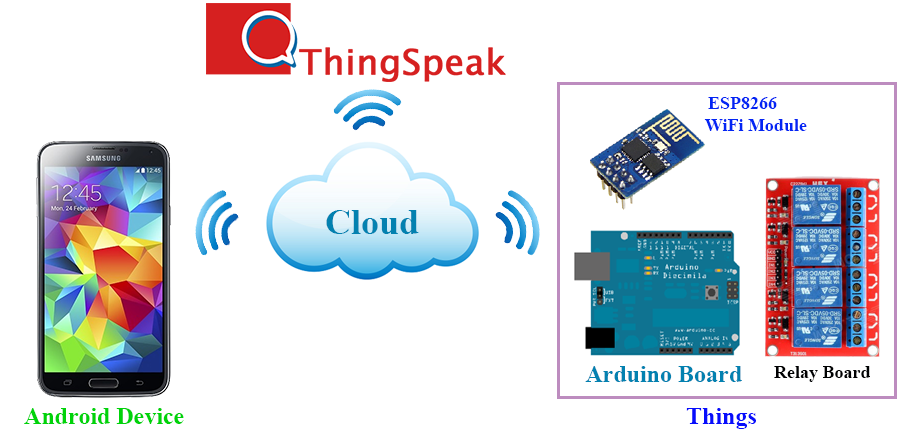
**NOISE POLLUTION MONITORING SYSTEM USING IOT**

**PROJECT DESCRIPTION:**

Noise pollution has become a very big issue around the world. The adverse effects of this pollution include hearing impairment, negative social behavior annoyance, sleep disturbance and intelligibility to understand people’s speech. In learning context, noise can affect understanding and behavior of people and places with high noise level are not suitable for learning and teaching process. Internet of Things (IoT) technology is one of the best choices to monitor the noise or sound intensity in the environment for the safety of human being.



**IOT IMPLEMENTATION:**

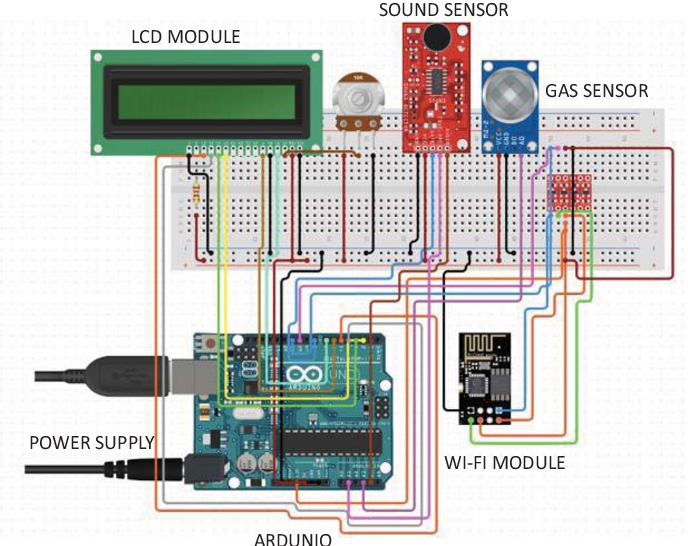


Sound sensor or mic sensor provides digital output and it detects sound from atmosphere. A WiFi module is also connected to Arduino and it is used to transfer data from the sensors to cloud server. ESP8266 WiFi module is used to store the data to online server. The data from sensor are basically analog signal so analog to digital converter (ADC) is used to convert the data. 16 x 2 Liquid crystal display (LCD) is used to display the measured value from the sensors. It can display two lines and each line has 16 characters.

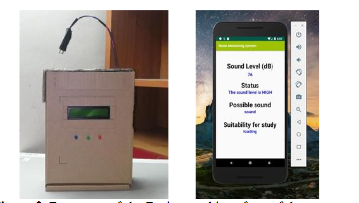
**METHODOLOGY:**

LM 393 sound sensor is used to read the readings of the sound level from the environment. The reading of sound sensor is calibrated using the real sound level meter to ge t the accurate readings of the sound level. The 16x2 LCD will show the values of sound level at that researched area and give the warning that says the level of sound is high when the measurement exceeds the set value. If the users could not read the readings due to poor eyesight, they can know the level of sound by using the light emitting diodes (LED) which in red, blue and green colour placed below the LCD. LED acts as an indicator to indicate when the noise is very high. It will turn to red, blue for low noise while green for intermediate level. All these components such as sound sensor, LCD, and LEDs will be connected to the ESP8266 NodeMCU.

**Block Diagram:**



**Prototype of Noise Pollution Monitoring:**



**App Development :**

As the app was created by using Visual Studio Code, the app will display the data taken from the sound sensor. Visual Studio Code is a software to create app use Python language to design an Android development. The app has four features which are the reading of sound level in dBA, the level of warning based on the reading of sound intensity, the possible sound that contributes to the sound level and the suitability for students to study. The app gives different level of warning such as “low”, “normal”, “high” and “very high”

**APP CODING:**

import SwiftUI

struct ContentView: View {

@State private var isMonitoring = false

@State private var noiseLevel = 0.0

var body: some View {

VStack {

Text("Current Noise Level: \(noiseLevel) dB")

.font(.largeTitle)

Button(action: {

// Start/stop monitoring code here

self.isMonitoring.toggle()

}) {

Text(isMonitoring ? "Stop Monitoring" : "Start Monitoring")

.padding()

.background(Color.blue)

.foregroundColor(.white)

.cornerRadius(10)

}

}

}

}

@main

struct NoiseApp: App {

var body: some Scene {

WindowGroup {

ContentView()

}

}

}

xml

<!-- activity\_main.xml layout file -->

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:gravity="center">

<TextView

android:id="@+id/noiseLevelTextView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Current Noise Level: 0 dB"

android:textSize="24sp" />

<Button

android:id="@+id/startStopButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Start Monitoring" />

</LinearLayout>

kotlin

// MainActivity.kt

import android.os.Bundle

import android.view.View

import android.widget.Button

import android.widget.TextView

import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

private lateinit var noiseLevelTextView: TextView

private lateinit var startStopButton: Button

private var isMonitoring = false

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

noiseLevelTextView = findViewById(R.id.noiseLevelTextView)

startStopButton = findViewById(R.id.startStopButton)

startStopButton.setOnClickListener {

// Start/stop monitoring code here

isMonitoring = !isMonitoring

updateUI()

}

}

private fun updateUI() {

if (isMonitoring) {

startStopButton.text = "Stop Monitoring"

} else {

startStopButton.text = "Start Monitoring"

}

}

}

**IOT CODING FOR ESP32 :**

import sounddevice as sd

import numpy as np

import paho.mqtt.client as mqtt

import time

# MQTT settings

mqtt\_broker\_address = "your\_broker\_address"

mqtt\_port = 1883

mqtt\_topic = "noise\_level"

# Sampling parameters

sample\_rate = 44100 # Samples per second

duration = 10 # Recording duration in seconds

# Function to calculate dB from audio data

def calculate\_db(audio\_data):

rms = np.sqrt(np.mean(np.square(audio\_data))) # Root Mean Square

db = 20 \* np.log10(rms / 0.0002) # Reference sound pressure level

return db

# MQTT callback functions

def on\_connect(client, userdata, flags, rc):

if rc == 0:

print("Connected to MQTT broker")

else:

print("Connection to MQTT broker failed")

def on\_publish(client, userdata, mid):

print("Data published to MQTT")

# Create MQTT client

client = mqtt.Client()

client.on\_connect = on\_connect

client.on\_publish = on\_publish

# Connect to MQTT broker

client.connect(mqtt\_broker\_address, mqtt\_port)

# Start recording and publishing

with sd.InputStream(callback=calculate\_db, channels=1, samplerate=sample\_rate):

print("Monitoring noise level...")

while True:

db = calculate\_db(np.random.random\_sample(44100)) # Simulated audio data for testing

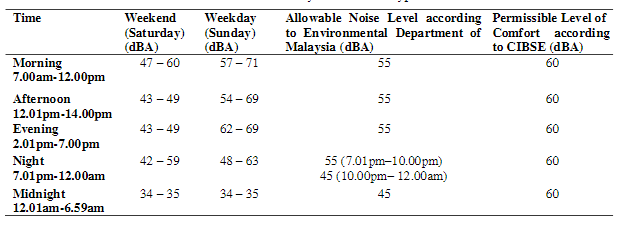
client.publish(mqtt\_topic, str(db))

time.sleep(60) # Publish data every minute (adjust as needed)

**Working Principle of the Prototype:**

The sound sensor will record the readings of sound level at the researched area. Then, the data is sent over to the cloud server called Firebase. Firebase is a development platform that is specialized for web application and mobile developed by Firebase Incorporation . Firebase is chosen as the cloud server for this system because it can easily be connected to Android Studio. The data is stored in Firebase real-time database which the user can also access via web browser. Then, data from Firebase is transferred to the app. The users can use the app to know the reading of sound level and they also can know which time is suitable to study and what factors contribute to the high noise based on the value of sound level

**DATA ANALYSIS FROM THE PROTOTYPE:**



**CONCLUSION :**

By using this project each and every variation we can analyze and inform nearby people in time. We can also analyze data form home using thingspeak. The most important factor of this system is that it is small, cost efficient and portable. Sensors are available easily anywhere. This system fully helpful to save the lives and overcome all the problem related to environment.