**IoT Based Noise Pollution Monitoring System**

**Phase 2**

**Abstract**

The increasing sound pollution is one of the significant issue now days. As the pollution increasing it is giving rise number of diseases so, it has become essential to control the pollution for better future and healthy life .here we propose an air quality as well as sound pollution monitoring system that allows us to monitor and check live sound pollution monitoring in particular areas through IOT . System keeps measure sound level and report it to the online server over IOT. The user friendly and easy handling of the system technology is such that it can be installed in houses, schools and in smallplaces.

**INTRODUCTION**

The main objective of IOT based noise pollution monitoring system is that the noise pollution is a rising issue these days.. Pollution is very harmful for those people who have any kind of internal diseases on this type of people pollution affect veryfastly. some are harmful for environment for certain level are good for human, animals, plants but beyond certain level these created problem for services to overcome these problem system is useful because of this we can analyse the noise pollution means how many pollution level in atmosphere in three different levels.We use Internet Of Things (IOT). In this we use thingspeak we can analyse previous data also using this platform in graphical form.

**PROBLEM STATEMENT**

An effective natural observing framework is essential to screen and estimate the conditions in the event of surpassing endorsed level of parameter . At the point whenthe items like condition furnished with sensor gadgets, smaller scale controller and different programming application turn into a self-securing and self-observing condition.

**OBJECTIVES**

1. To study the existing system.

2. To design the block diagram.

3. To decide the components specification &device in system.

4. To design the circuit diagram and simulate it using suitable software.

5. To design the PCB and implement hardware.

6. To test the circuit and observe the result.

7. To prepare report.

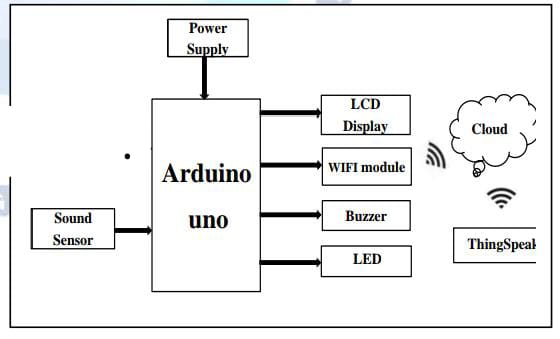
**RELATED WORK**

There are numerous works that have been done related to IOT based Projects. Using node mcu , mic and ultrasonic sensor to monitoring noise which present in atmosphere and giving reading to node mcu processor and data to the rends. using OLED display, to display the pollution condition. All sensors will sends data to node mcu or ESP32 and and the node mcu or ESP32 send to think speak then graph shows on think speak and also data show on OLED.

**WORKING & BLOCK SCHEMATIC**

In system we use arduino as main controller. In system we use sound sensor LM393 module for detect the sound pollution. Sensed data of sensor given to analog pin of the arduino then digital output pin are connected to LCD, buzzer and LED. If sound pollution is there then LED will glow. All condition of pollution display on LED and we can also analyze past data using thingspeak in graphical form. Arduino an open source prototype. Software will operate in Arduino IDE Computer code can be written and upload to the physical board. Arduino board is a board that can be functioned via Arduino IDE by sending a set of instructions to the microcontroller on it. For controlling Sensors. For arduino programming we are going to use Embedded C. We are going to build project in Embedded C and for monitoring that project

we are using Cloud connected with ESP32 which was connected in the circuit.



**COMPONENTS**

1. ArduinoUNO

2. LM393 (Noisesensor)

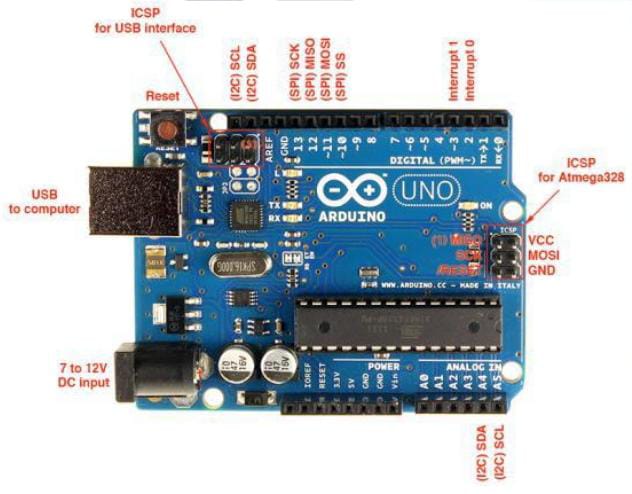
3. ESP8266 WIFIModule

4. 16\*2 LCDDisplay

5. LED

6. Buzzer

1. **ArduinoUNO**



Arduino is 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 as PWM 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.

**Features**

• Operating voltage is5v.

• DC current per input pin is 40mA.

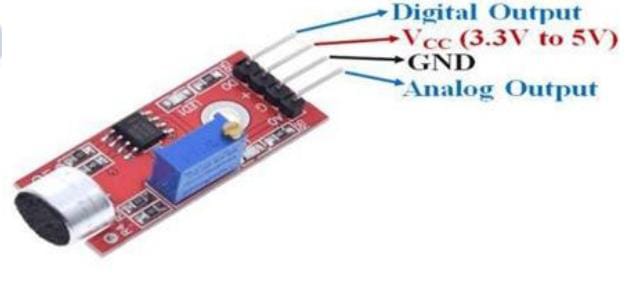
• Clock speed16MHz.

• DC current for 3.3v pin is 50mA.

• SPAM 2 KB

• EEPROM 1KB

1. **LM393 (Noise sensor)**



The sound sensor module provide an easy way to detect sound and it generally used for detecting sound

intensity. Module detect the sound has exceeded a threshold value. Sound is detected via microphone and fed into an LM393 opamp. The sound level adjust through pot. The sound increases set value output is low.These module work on DC 3.3-5 voltage.

**Features**

• Operating voltage 3.3V-5V

• Output model: digital switch outputs (0 and 1, high or low level)

• Voltage Gain 26dB

• Microphone Impedance 2.2kΏ

• Microphone Frequency 16.20 kHz

1. **ESP32 or ESP8266 WIFIModule**



The ESP32 or esp8266 WIFI module is a self containedsoc with integrated TCP/IP protocol stack that can give any microcontroller access to your WIFI network. The ESP32 or esp8266 is capable of either hosting an application or offloading all WIFI networking functions from another application processor.

**Features**

• 2.4 GHz Wi-Fi (802.11 b/g/n supporting WPA/WPA2).

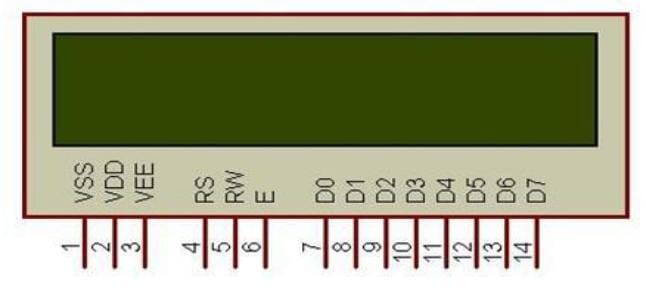
• General-purpose input/output (16 GPIO).

• Inter-Integrated Circuit (I²C) serial communication protocol.

• Analog-to-digital conversion (10-bit ADC).

• Serial Peripheral Interface (SPI) serial communication

1. **16\*2 LCDDisplay**



LCD is used for to display the condition there are three conditions in air pollution and three conditions in noise pollution means air and sound is clear, moderately polluted or highly polluted that is displayed on LED.

**Features**

• Operating Voltage is 4.7V to 5.3V

• Current consumption is 1mA without backlight

• Alphanumeric LCD display module, meaning can display alphabets and numbers

• Consists of two rows and each row can print 16 characters.

• Each character is build by a 5×8 pixel box

• Can work on both 8-bit and 4-bit mode

1. **LED**



light emitting diode (a device that produces a light on electrical and electronic equipment).

1. **Buzzer**



An Active Buzzer Alarm Module easily interfaced with an Arduino or other microcontrollers is an audio signaling device, which may be mechanical, electromechanical, or piezoelectric. Just like what you are viewing now, it is 3.3V to 5V DC power supply Electronic Part Active Buzzer Module.

**ALGORITHM**

1. Start Arduino UNO.

2. initialize LCD, gas sensor & noise sensor

3. Establish WITI connections.

4. If connection successful. next step else go to step 1

5. Read sensor values.

6. If Sensor value available establish TCP connection else read values again.

7. Establish TCP connection

8. If TCP connections successful. send data to server (Thing speak). else set go to step 7

9. If TCP connections successful. send data to server (Thing Speak). else set gob to step 7

10. Check for acknowledgement.

11. If acknowledgement received. go to step some time & else wait for go to step no 5.

**ADVANTAGES**

1. Sensors are easily available.

2. Sensors are effortlessly accessible.

3. Simple, compact and easy to handle.

4. Sensors have long life time.

5. Low cost

6. Data can be used to control pollution.

**APPLICATIONS**

1. To estimate the pollution.

2. To design server and upload data on that server with date and time.

4. We can use it at industrial area as there is lot of noise pollution

5. In city roads traffic noise.

6. Automation Automation.

**FUTURE SCOPE**

In future we modify the system to notify a user about the noise level it reaches beyond permissible level through sms or app. We can monitor sound pollution level at any place of the world.

**PROGRAM**

import time

import sys

import struct

import getopt

import threading

import usb1

class Bafx3608:

"""BAFX3608 Sound Pressure Level meter USB driver"""

USBVendorID = 0x64bd

USBProductID = 0x74e3

OUT\_ENDPOINT = 0x02

IN\_ENDPOINT = 0x81

RANGE = ['30-130', '30-80', '50-100', '60-110', '80-130', 'invalid', 'invalid', 'invalid']

WEIGHT = ['A', 'C']

MAXMODE = [' ', 'Max']

FASTMODE = ['Slow', 'Fast']

def \_\_init\_\_(self, fast=True, maxmode=False, weightC=False, dBrange=0):

self.options = (fast << 6) | (maxmode << 5) | (weightC << 4) | dBrange

self.thread\_id = -1

self.cb\_on\_reading = None

self.cb\_on\_reading\_str = None

self.cb\_on\_reading\_raw = None

self.handle = None

def set\_config\_usb(self):

"""

set\_config\_usb() writes BAFX3600 meter options over USB

"""

# Write meter options

command = bytearray([0x56, self.options, 0, 0, 0, 0, 0, 0])

while True:

try:

self.handle.interruptWrite(Bafx3608.OUT\_ENDPOINT, command, 10)

break

except usb1.USBErrorTimeout:

pass

def set\_config(self, fast=True, maxmode=False, weight\_c=False, db\_range=0):

"""

set\_config(fast=True, maxmode=False, bool weight\_c=False, db\_range=0)

"""

self.options = (fast << 6) | (maxmode << 5) | (weight\_c << 4) | db\_range

self.set\_config\_usb()

def get\_config(self):

"""

get\_config() return BAFX3608 options as bitmap

"""

return self.options

def loop\_start(self):

"""

Start thread reading from meter

"""

if self.thread\_id == -1:

self.thread\_id = threading.Thread(target=self.usb\_poll\_thread)

self.thread\_id.start()

def loop\_forever(self):

"""

Read from meter blocking. Does not return.

"""

self.usb\_poll\_thread()

def \_eval\_data(self, usbdata, fast\_mode):

if usbdata is not None and len(usbdata) > 2:

decibels, options = struct.unpack\_from('>HB', usbdata)

decibels = decibels / 10

if decibels <= 130.0:

fast\_mode = (options & (1<<6)) != 0

max\_mode = (options & (1<<5)) != 0

ac\_mode = (options & (1<<4)) != 0

inrange = options & 0x07

if self.cb\_on\_reading\_raw is not None:

self.cb\_on\_reading\_raw(usbdata[:3])

if self.cb\_on\_reading is not None:

self.cb\_on\_reading(decibels, fast\_mode, max\_mode, ac\_mode,\

inrange)

if self.cb\_on\_reading\_str is not None:

db\_str = f"{decibels:5.1f}"

fast\_mode\_str = f"{self.FASTMODE[fast\_mode]}"

max\_mode\_str = f"{self.MAXMODE[max\_mode]}"

self.cb\_on\_reading\_str(db\_str, fast\_mode\_str, max\_mode\_str, \

f"{self.WEIGHT[ac\_mode]}", f"{self.RANGE[inrange]}")

return fast\_mode

def usb\_poll\_thread(self):

"""

Meter reading thread

"""

self.handle = usb1.USBContext().openByVendorIDAndProductID(

Bafx3608.USBVendorID,

Bafx3608.USBProductID,

skip\_on\_error=True,

)

if self.handle is None:

# Device not present, or user is not allowed to access device.

print('Device not present')

sys.exit(1)

if self.handle.kernelDriverActive(0):

self.handle.detachKernelDriver(0)

self.handle.claimInterface(0)

try:

self.set\_config\_usb()

# Read sound levels from meter

command = bytearray([0xb3, 0xaa, 0xbb, 0xcc, 0x00, 0x00, 0x00, 0x00])

fast\_mode = True

while True:

try:

self.handle.interruptWrite(Bafx3608.OUT\_ENDPOINT, command, 20)

data = self.handle.interruptRead(Bafx3608.IN\_ENDPOINT, 8, 10)

fast\_mode = self.\_eval\_data(data, fast\_mode)

except usb1.USBErrorTimeout:

pass

except usb1.USBErrorOverflow:

pass

except usb1.USBErrorPipe:

pass

except usb1.USBErrorNoDevice:

break

if fast\_mode:

time.sleep(0.250)

else:

time.sleep(1.0)

finally:

try:

self.handle.releaseInterface(0)

self.handle.close()

except usb1.USBErrorNoDevice:

pass

def main():

"""

main program

"""

help\_cli = f'{sys.argv[0]} --range=[0..4] --fast=[0,1] --max=[0,1] --weight=[A,C]'

out\_range = 0

out\_fast = 1

out\_weight = 0

out\_max = 0

try:

opts, \_args = getopt.getopt(sys.argv[1:], '', ['range=', 'fast=', 'weight=', 'max='])

except getopt.GetoptError:

print(help\_cli)

sys.exit(2)

try:

for opt, arg in opts:

if opt == '--range':

out\_range = int(arg) & 0x07

elif opt == '--fast':

out\_fast = int(arg) & 0x01

elif opt == '--weight':

out\_weight = 0

if arg in ('C', 'c'):

out\_weight = 1

elif opt == '--max':

out\_max = int(arg) & 0x01

except ValueError:

print(help\_cli)

sys.exit(2)

def reading\_callback\_str(decibels, \_fast, maxmode, weight\_c, db\_range):

"""

callback

"""

print(decibels, "dB" + weight\_c, maxmode, db\_range)

meter = Bafx3608(out\_fast, out\_max, out\_weight, out\_range)

meter.cb\_on\_reading\_str = reading\_callback\_str

meter.cb\_on\_reading = None

meter.cb\_on\_reading\_raw = None

meter.loop\_forever()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**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.