IoT Based Noise Pollution Monitoring System

Components used in this project

- ESP8266 NodeMCU Board
- Microphone sensor
- 16*2 LCD Module
- Breadboard
- Connecting wires

ESP8266 NodeMCU Board

The ESP8266 is a **low-cost Wi-Fi microchip, with built-in TCP/IP networking software, and micro controller capability.**

In our project we use this micro controller for network conection.

ESP8266 NodeMCU Board

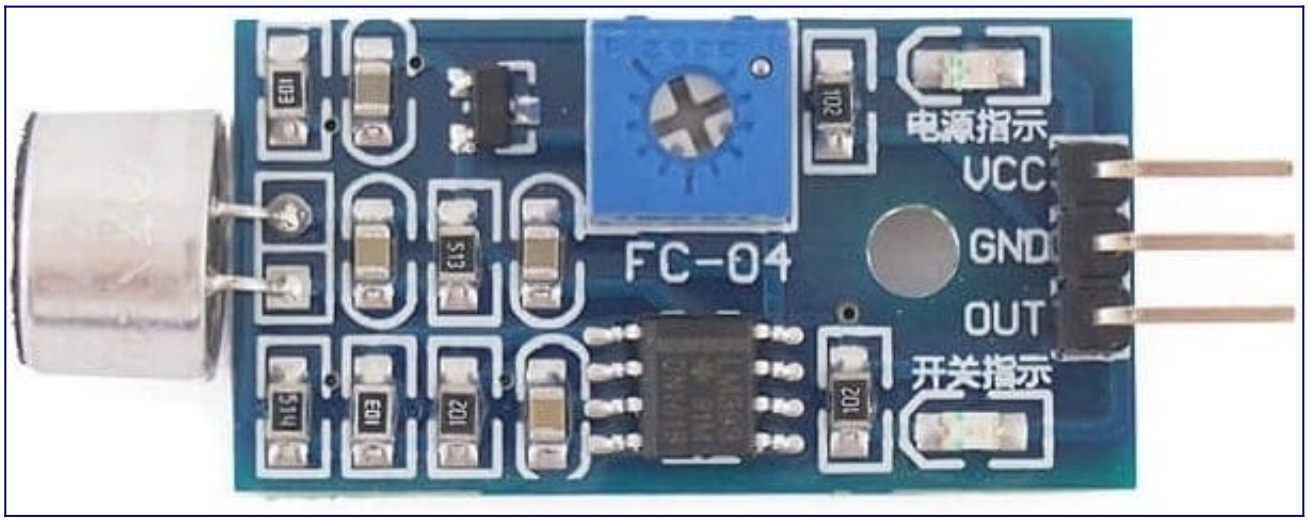
Microphone sensor

The microphone sound sensor, as the name says, detects sound. It gives a measurement of how loud a sound is. The sound sensor is a small board that combines a microphone (50Hz-10kHz) and some processing circuitry to convert sound waves into electrical signals. This electrical signal is fed to on-board LM393 High Precision Comparator to digitize it and is made available at OUT pin.

The module has a built-in potentiometer for sensitivity adjustment of the OUT signal. We can set a threshold by using a potentiometer. So that when the amplitude of the sound exceeds the threshold value, the module will output LOW otherwise HIGH.

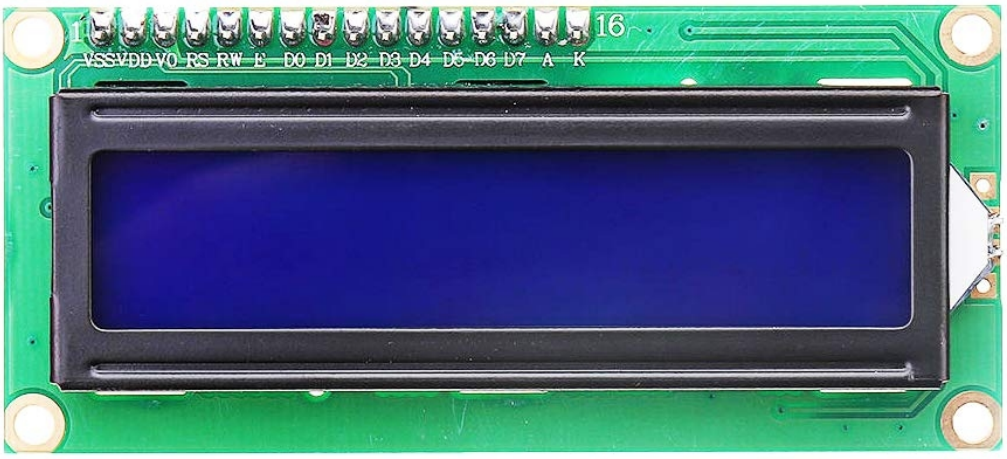
Apart from this, the module has two LEDs. The Power LED will light up when the module is powered. The Status LED will light up when the digital output goes LOW.

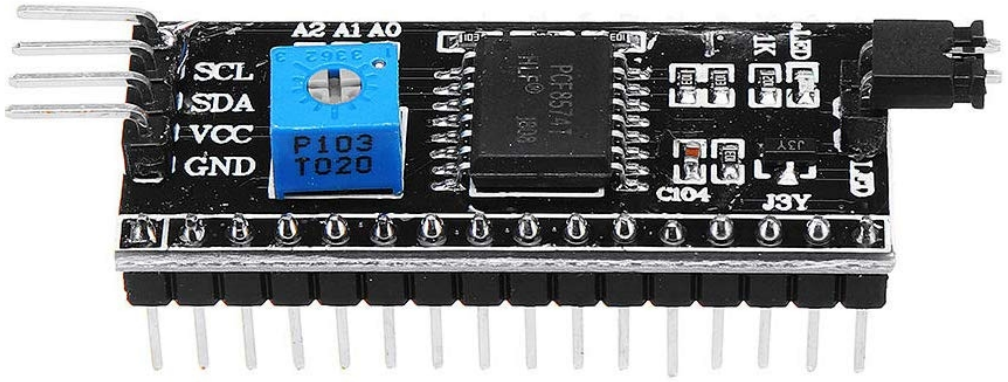
The sound sensor only has three pins: VCC, GND & OUT. VCC pin supplies power for the sensor & works on 3.3V to 5V. OUT pin outputs HIGH when conditions are quiet and goes LOW when sound is detected.



Microphone sensor

16x2 LCD Display with I2C Module with Controller

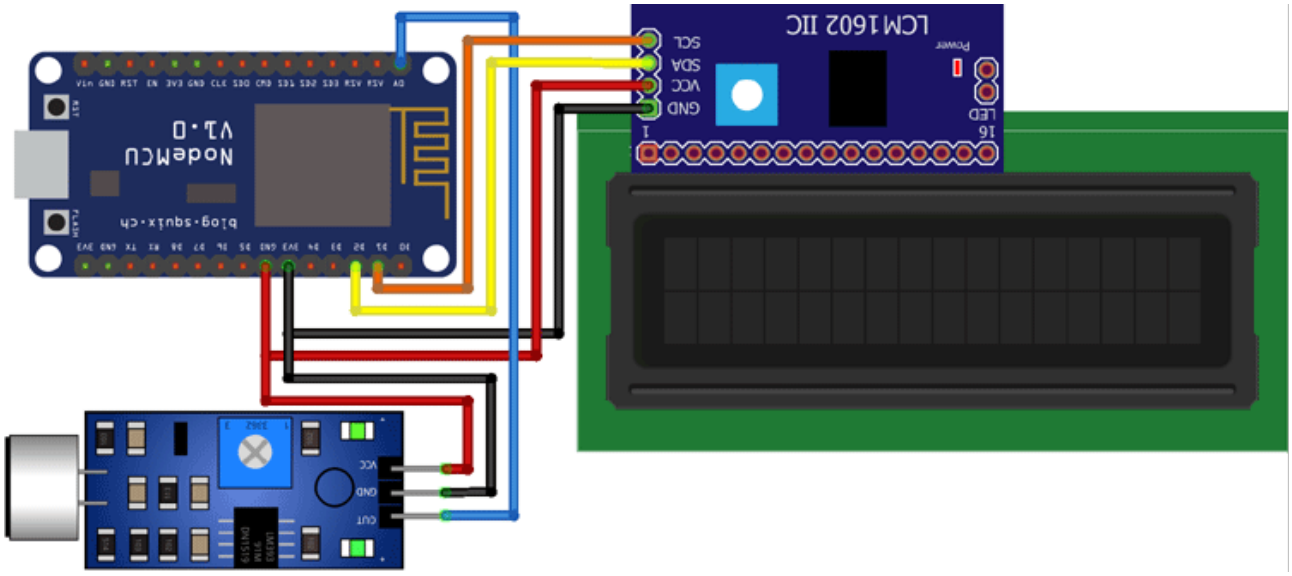




Software and online services:

- Python IDE
- Arduino IDE
- SQLite
- Vue js
- Think space
-

Circuit Diagram

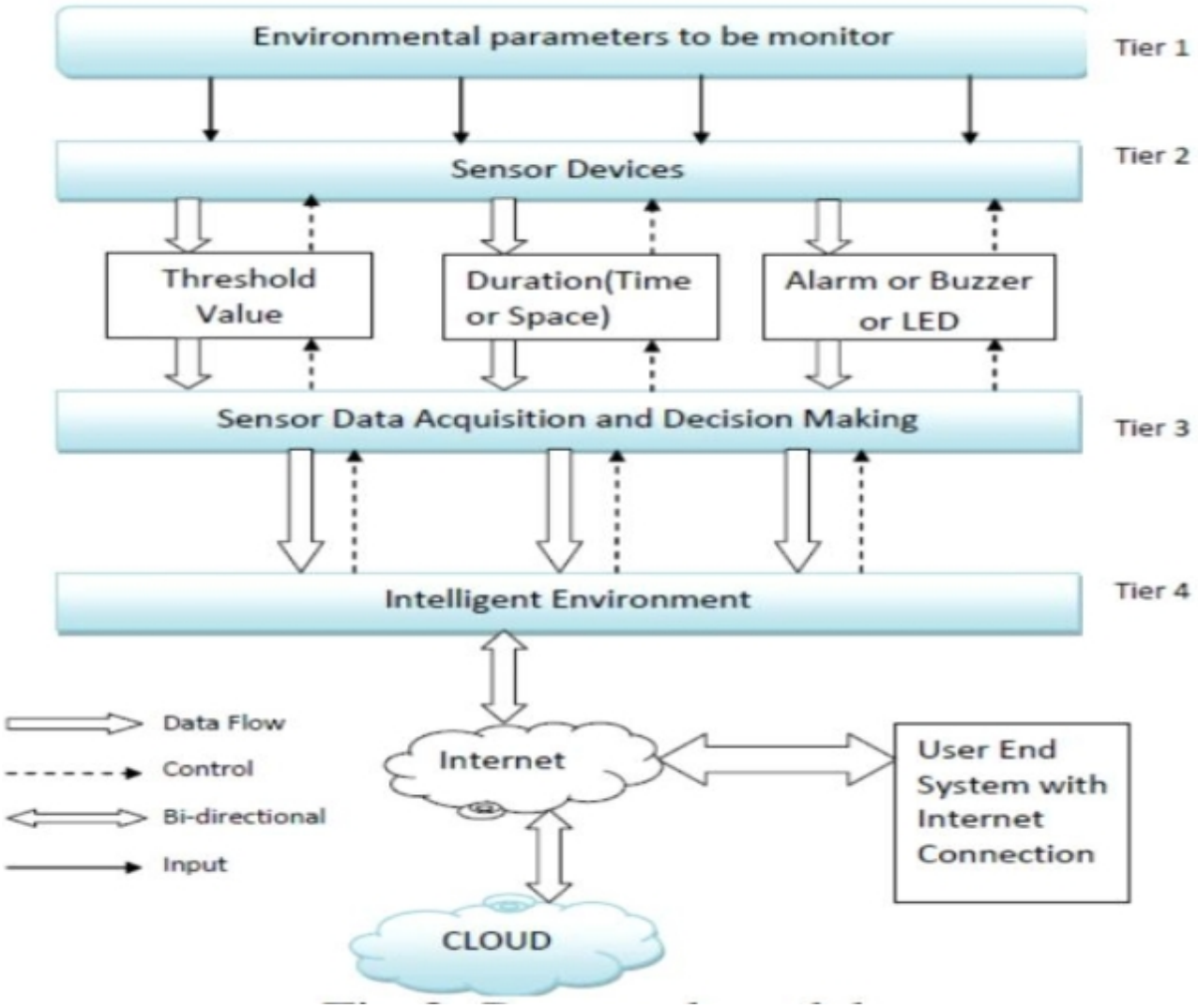


IoT Based Noise Pollution Monitoring System

The connections are pretty simple, we just have to connect the sound sensor to one of the Analog pin and the LCD to the I2C pins.

n the above diagram, we have connected the power pins of the sound sensor and LCD display to 3v3 and GND pin of NodeMCU. Along with that, we have also connected the SCL and SDA pins of the module to D1 and D2 respectively, and the OUT pin of the sound sensor to A0 pin.

Proposed model for this project



Program use in the project

Here, we have to develop a code that takes input from the sound sensor and maps it value to decibels and after comparing the loudness, it should not only print it to the 16*2 LCD display.

In the very first part of the code, we have included all the necessary libraries and definitions. Also, we have defined the necessary variables and objects for further programming.

```
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <LiquidCrystal_I2C.h>
#define SENSOR_PIN A0
LiquidCrystal_I2C lcd(0x3F, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
const int sampleWindow = 50;
unsigned int sample;
int db;
char auth[] = "IEu1xT825VDt6hNfrcFgdJ6InJ1QUfsA";
char ssid[] = "YourSSID";
char pass[] = "YourPass";
```

Further ahead, we have created a Blynk function to handle the virtual pin that our gauge is connected to. We are simply sending the values stored in the dB variable to the V0 pin.

```
BLYNK_READ(V0)
{
  Blynk.virtualWrite(V0, db);
}
```

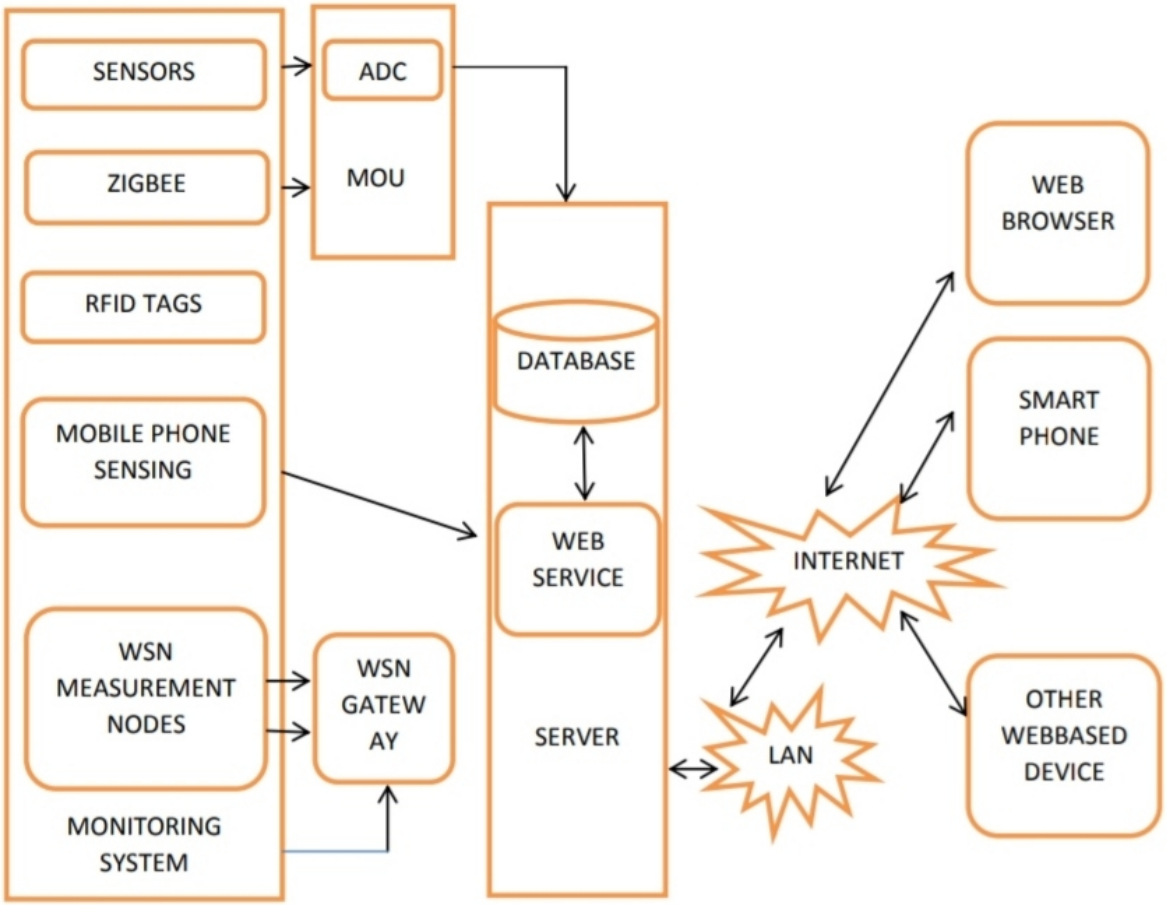
In the setup part of the code, we are defining the pin mode as input and beginning the LCD display as well as the Blynk function.

```
void setup() {
  pinMode (SENSOR_PIN, INPUT);
  lcd.begin(16, 2);
  lcd.backlight();
  lcd.clear();
  Blynk.begin(auth, ssid, pass);
}
```

In the loop part, we are doing all the processing tasks like comparison and value assignment along with running the Blynk function.

```
void loop() {
  Blynk.run();
  unsigned long startMillis = millis(); // Start of sample window
  float peakToPeak = 0; //peak-to-peak level
  unsigned int signalMax = 0; //minimum value
  unsigned int signalMin = 1024; //maximum value
  // collect data for 50 mS
  while (millis() - startMillis < sampleWindow)
  {
    sample = analogRead(SENSOR_PIN); //get reading from microphone
    if (sample < 1024) // toss out spurious readings
    {
      if (sample > signalMax)
      {
        signalMax = sample; // save just the max levels
      }
      else if (sample < signalMin)
      {
        signalMin = sample; // save just the min levels
      }
    }
  }
  peakToPeak = signalMax - signalMin; // max - min = peak-peak amplitude
  Serial.println(peakToPeak);
  db = map(peakToPeak, 20, 900, 49.5, 90); //calibrate for deciBels
  lcd.setCursor(0, 0);
  lcd.print("Loudness: ");
  lcd.print(db);
  lcd.print("dB");
  if (db <= 50)
  {
    lcd.setCursor(0, 1);
    lcd.print("Level: Quite");
  }
  else if (db > 50 && db < 75)
  {
    lcd.setCursor(0, 1);
    lcd.print("Level: Moderate");
  }
  else if (db >= 75)
  {
    lcd.setCursor(0, 1);
    lcd.print("Level: High");
  }
  delay(600);
  lcd.clear();
}
```

Existing system model



Backup program(other method)

```
 #include <ESP8266WiFi.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1       /
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

int num_Measure = 128 ;
int pinSignal = A0;
long Sound_signal;
long sum = 0 ;
long level = 0 ;
int soundlow = 40;
int soundmedium = 500;
int error = 33;

String apiKey = "14K8UL2QEK8BTHN6";
const char *ssid = "Alexahome";
const char *pass = "12345678";
const char* server = "api.thingspeak.com";

WiFiClient client;

void setup ()
{
  pinMode (pinSignal, INPUT);
  Serial.begin (115200);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  delay(10);

  Serial.println("Connecting to ");
  Serial.println(ssid);

  display.clearDisplay();
  display.setCursor(0,0);
```

```
  display.setTextSize(1);
  display.setTextColor(WHITE);

 display.println("Connecting to ");
  display.setTextSize(2);
  display.print(ssid);
  display.display();

  WiFi.begin(ssid, pass);

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
    Serial.println("");
    Serial.println("WiFi connected");

    display.clearDisplay();
    display.setCursor(0,0);
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.print("WiFi connected");
    display.display();
    delay(4000);
}


void loop ()
{

  // Performs 128 signal readings
  for ( int i = 0 ; i <num_Measure; i ++)
  {
   Sound_signal = analogRead (pinSignal);
    sum =sum + Sound_signal;
  }

  level = sum / num_Measure;

  Serial.print("Sound Level: ");
  Serial.println (level-error);

    display.clearDisplay();
    display.setCursor(0,0);  //oled display
    display.setTextSize(1);
```

```
   display.setTextColor(WHITE);
   display.println("--- Decibelmeter ---");



  if( (level-error) < soundlow)
  {

   Serial.print("Intensity= Low");
   display.setCursor(0,20);
   display.setTextSize(1);
   display.setTextColor(WHITE);
   display.print("Sound Level: ");
   display.println(level-error);

   display.setCursor(0,40);
   display.print("Intensity: LOW");

   display.display();
//    {
//     if( (level-error) < 0)
//     Serial.print("Intensity= Low");
//     display.setCursor(0,20);
//     display.setTextSize(1);
//     display.setTextColor(WHITE);
//     display.print("Sound Level: 0");
//
//     display.setCursor(0,40);
//     display.print("Intensity: LOW");
//
//     display.display();
//    }

  }
  if( ( (level-error) > soundlow ) && ( (level-error) < soundmedium )  )
  {

   Serial.print("Intensity=Medium");
   display.setCursor(0,20);  //oled display
   display.setTextSize(1);
   display.setTextColor(WHITE);
   display.print("Sound Level: ");
   display.println(level-error);

   display.setCursor(0,40);
   display.print("Intensity: MEDIUM");
```

```
  display.display();

}
if( (level-error) > soundmedium )
{

  Serial.print("Intensity= High");
  display.setCursor(0,20);  //oled display
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.print("Sound Level: ");
  display.println(level-error);

  display.setCursor(0,40);
  display.print("Intensity: HIGH");
  display.display();

}
sum = 0 ;
delay(200);

if (client.connect(server, 80))
{
  String postStr = apiKey;
  postStr += "&field1=";
  postStr += String(level-error);
  postStr += "r\n";

  client.print("POST /update HTTP/1.1\n");
  client.print("Host: api.thingspeak.com\n");
  client.print("Connection: close\n");
  client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");
  client.print("Content-Type: application/x-www-form-urlencoded\n");
  client.print("Content-Length: ");
  client.print(postStr.length());
  client.print("\n\n");
  client.print(postStr);

}
  client.stop();
```

APPLICATIONS OF THE PROJECT

- Roadside pollution Monitoring.
- Industrial Perimeter Monitoring.
- Site selection for reference monitoring stations.
- Indoor Air Quality Monitoring.
- Design server using IoT and upload data on that
- server with date and time.
- To make this data available to the common man.
- To set a danger limit on that server and inform
- authorities to take future actions for wellbeing.

CONCLUSION

To implement this need to deploy the sensor devices in theenvironment for collecting the data and analysis. Bydeploying sensor devices in the environment, we can bring theenvironment into real life i.e. it can interact with other objectsthrough the network. Then the collected data and analysisresults will be available to the end user through the Wi-Fi.

This data will be helpful for future analysis and it can beeasily shared to other end users. Thismodel can be furtherexpanded to monitor the developing cities and industrial zonesfor pollution monitoring. To protect the public health frompollution, this model provides an efficient and low costsolution for continuous monitoring of environment.