PROECT TITLE : NOISE POLLUTION MONITORING SYSTEM

PHASE 4 : DEVELOPMENT PART – II

OBJECTIVE:

The objective of a noise monitor is to provide data regarding the level of noise in a location so that it may be compared to the established noise limits.

It helps identify work locations where there are noise problems, employees who may be exposed to noise levels that can cause hearing loss, and where additional noise measurements need to be made. This information also helps determine appropriate noise control measures that need to be put in place.

To regulate and control noise producing and generating sources. Maintain the ambient air quality standards in respect of noise.

WOKWI DOCS:

Wokwi is an online Electronics simulator. You can use it to simulate Arduino, ESP32, STM32, and many other popular boards, parts and sensors.

Start right now.

No waiting for components, or downloading large software. Your browser has everything you need to start coding your next IoT project in seconds.

Mistakes are okay.

You can't destroy the virtual hardware. Trust us, we tried. So don't worry about frying your precious components. And unlike real hardware, you can always undo.

Easy to get help and feedback.

Sharing a link to your Wokwi project is all you need.

Gain confidence in your code.

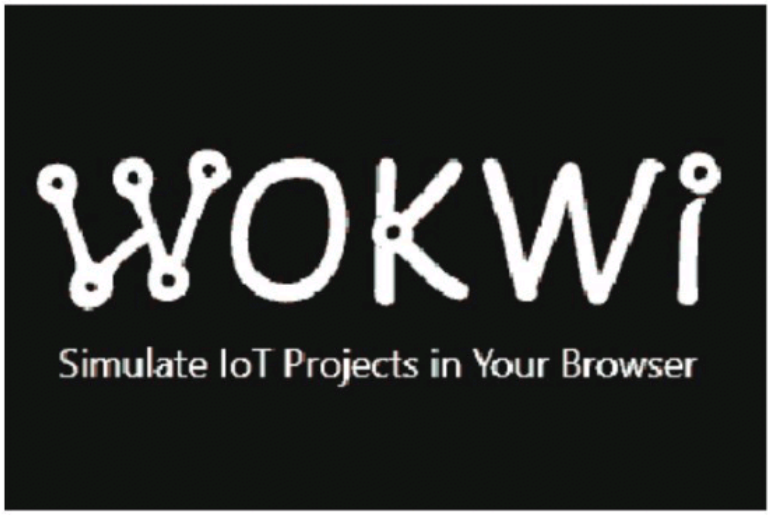
Separate hardware and software issues.

Unlimited hardware.

No need to scavenge parts from old projects. Use as many parts as you need, without worrying about project price and stock.

Maker-friendly community.

A place for you to share your projects, ask for help, and get inspiration.



CODING WITH EXPLANATION:

Allows communication with alphanumerical liquid crystal displays (LCDs).
This library allows an Arduino/Genuino board to control LiquidCrystal displays (LCDs) based on the Hitachi HD44780 (or a compatible) chipset, which is found on most text-based LCDs. The library works with in either 4 or 8 bit mode (i.e. using 4 or 8 data lines in addition to the rs, enable, and, optionally, the rw control lines).

```
#include <LiquidCrystal.h> // include the LiquidCrystal
library

const int micPin1 = A0; // define the pin for the first
microphone
const int micPin2 = A1; // define the pin for the second
microphone
const int micPin3 = A2; // define the pin for the third
microphone
const int buzzerPin = 9; // define the pin for the buzzer
const int ledPin = 6; // define the pin for the LED
const int contrast = 50; // define the LCD contrast
LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // initialize the
LCD display
```

Configures the specified pin to behave either as an input or an output. See the Digital Pins page for details on the functionality of the pins.

As of Arduino 1.0.1, it is possible to enable the internal pullup resistors with the mode `INPUT_PULLUP`. Additionally, the `INPUT` mode explicitly disables the internal pullups.

Syntax

```
pinMode(pin, mode)
```

```
void setup() {
  pinMode(buzzerPin, OUTPUT); // set the buzzer pin as
output
  pinMode(ledPin, OUTPUT); // set the LED pin as output
  lcd.begin(16, 2); // initialize the LCD display
  analogWrite(6,contrast); // set the LCD contrast
  Serial.begin(9600); // initialize the serial monitor
}
```

Reads the value from the specified analog pin. Arduino boards contain a multichannel, 10-bit analog to digital converter. This means that it will map input voltages between 0 and the operating voltage(5V or 3.3V) into integer values between 0 and 1023. On an Arduino UNO, for example, this yields a resolution between readings of: 5 volts / 1024 units or, 0.0049 volts (4.9 mV) per unit. See the table below for the usable pins, operating voltage and maximum resolution for some Arduino boards.

```
void loop() {
  // read the values from the microphones
  int micValue1 = analogRead(micPin1);
  int micValue2 = analogRead(micPin2);
  int micValue3 = analogRead(micPin3);

  // calculate the sound levels in dB for each microphone
  float voltage1 = micValue1 * 5.0 / 1024.0; // convert
the first microphone value to voltage (5V reference)
  float voltage2 = micValue2 * 5.0 / 1024.0; // convert
the second microphone value to voltage (5V reference)
```

```
  float voltage3 = micValue3 * 5.0 / 1024.0; // convert
the third microphone value to voltage (5V reference)
  float dB1 = 20 * log10(voltage1/0.0063); // calculate
the sound level in dB for the first microphone
  float dB2 = 20 * log10(voltage2/0.0063); // calculate
the sound level in dB for the second microphone
  float dB3 = 20 * log10(voltage3/0.0063); // calculate
the sound level in dB for the third microphone

  // calculate the average sound level in dB for all
microphones
  float averageDB = (dB1 + dB2 + dB3) / 3;
```

lcd.setCursor()

This function places the cursor (and any printed text) at any position on the screen. It can be used in the void setup() or void loop() section of your program. The cursor position is defined with lcd.setCursor(column, row) . The column and row coordinates start from zero (0-15 and 0-1 respectively).

```
  // display the sound level on the LCD display and the
serial monitor
  lcd.setCursor(0, 0); // set the cursor to the first row
of the LCD display
  lcd.print("Sound Level: "); // print the text "Sound
Level: " on the LCD display
  lcd.setCursor(0, 1); // set the cursor to the second row
of the LCD display
  lcd.print(averageDB); // print the average sound level
on the LCD display
  Serial.print("Sound Level: "); // print the text "Sound
Level: " on the serial monitor
  Serial.println(averageDB); // print the average sound
level on the serial monitor
```

If you do not set the `pinMode()` to `OUTPUT`, and connect an LED to a pin, when calling `digitalWrite(HIGH)`, the LED may appear dim. Without explicitly setting `pinMode()`, `digitalWrite()` will have enabled

the internal pull-up resistor, which acts like a large current-limiting resistor.

Syntax

```
digitalWrite(pin, value)
```

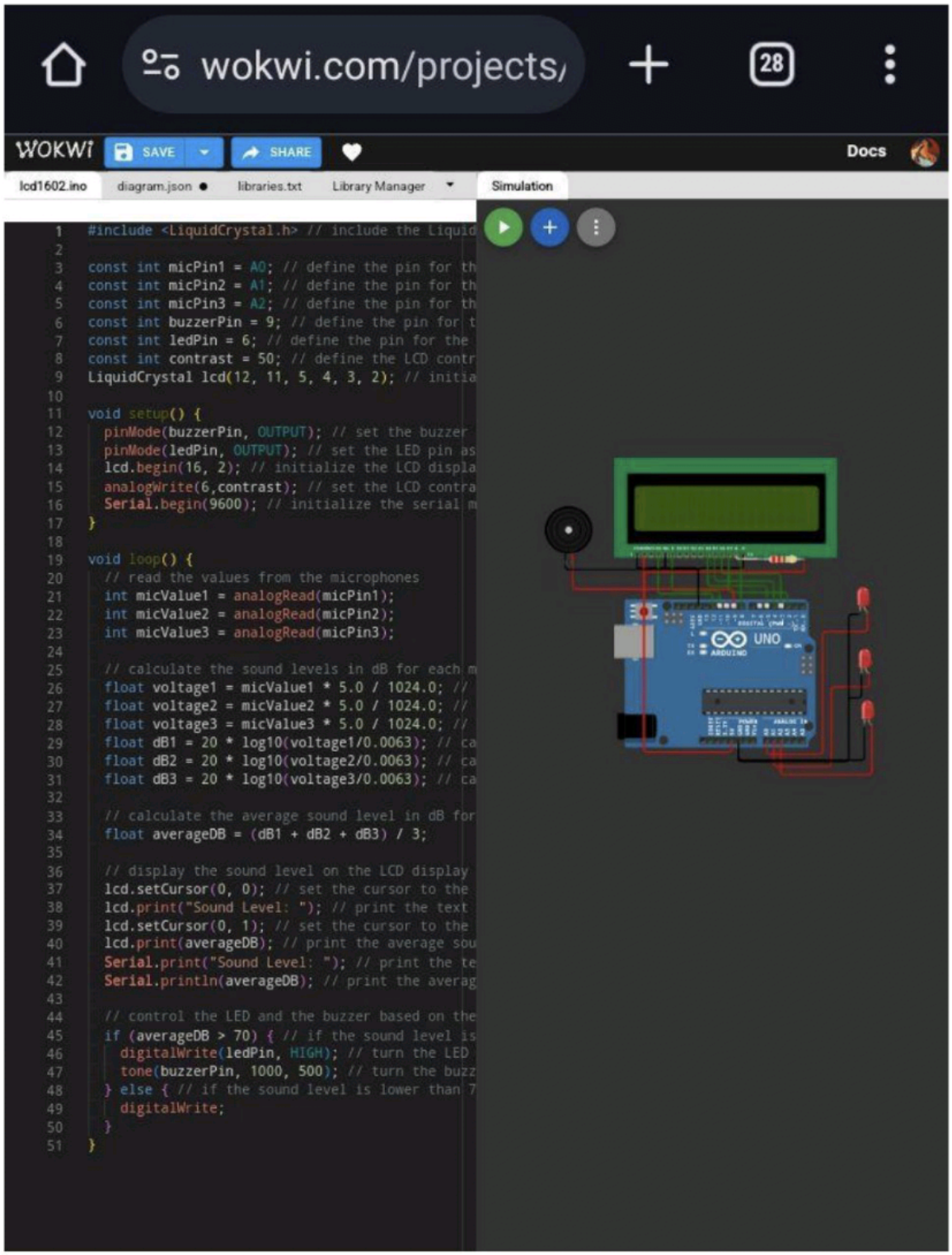
```
  // control the LED and the buzzer based on the sound
level
  if (averageDB > 70) { // if the sound level is higher
than 70 dB
    digitalWrite(ledPin, HIGH); // turn the LED on
    tone(buzzerPin, 1000, 500); // turn the buzzer on
  } else { // if the sound level is lower than 70 dB
    digitalWrite;
  }
}
```

DIAGRAM DESIGN:

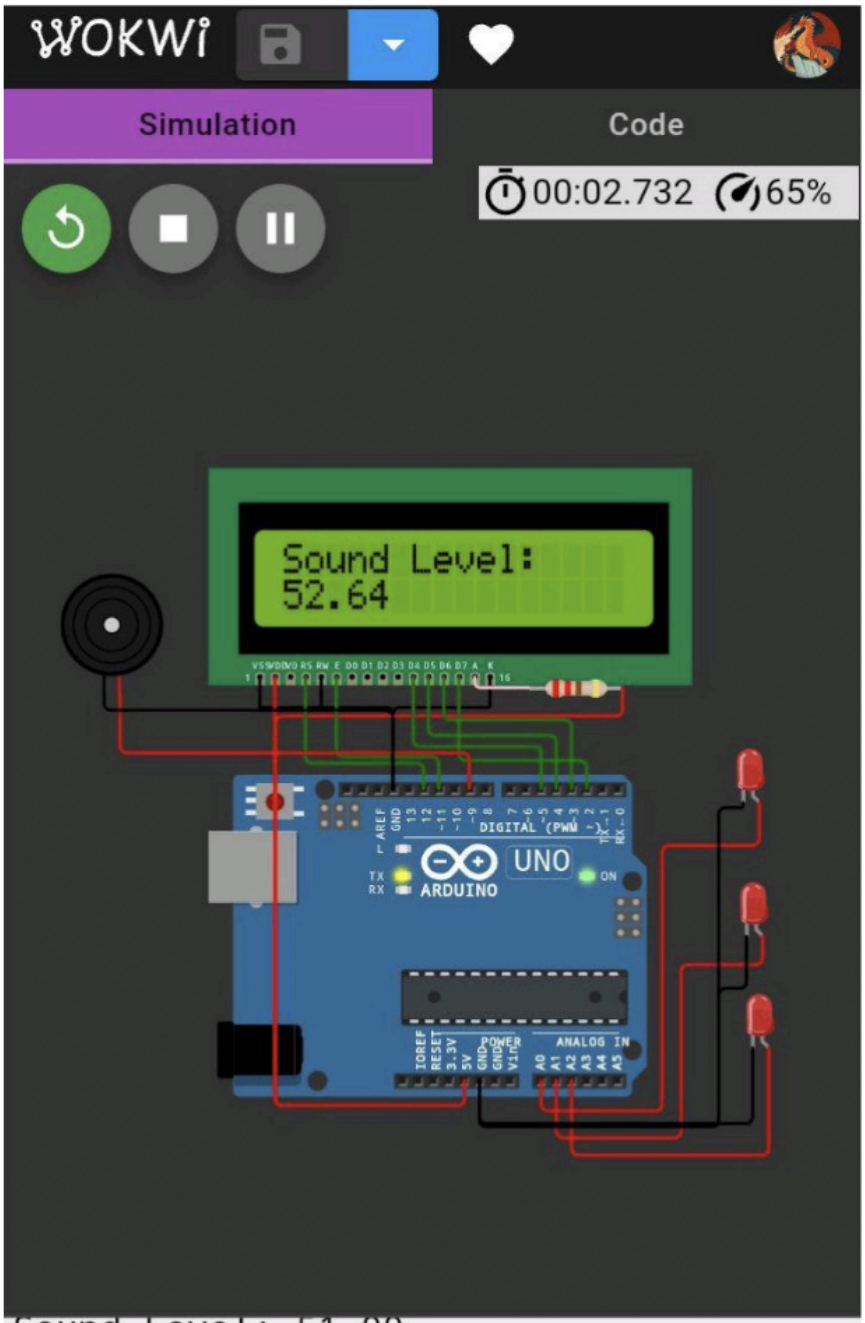
```
{
  "version": 1,
  "author": "Uri Shaked",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-arduino-uno", "id": "uno", "top": 200, "left":
20, "attrs": {} },
    { "type": "wokwi-lcd1602", "id": "lcd", "top": 8, "left": 20,
"attrs": {} },
    { "type": "wokwi-resistor", "id": "r1", "top": 140, "left": 220,
"attrs": { "value": "220" } },
    {
      "type": "wokwi-buzzer",
      "id": "bz1",
      "top": 66.16,
      "left": -72.28,
      "attrs": { "volume": "0.1" }
    },
    {
      "type": "wokwi-led",
      "id": "led1",
      "top": 175.61,
```

```
      "left": 339.36,
      "attrs": { "color": "red" }
    },
    {
      "type": "wokwi-led",
      "id": "led2",
      "top": 259.31,
      "left": 341.33,
      "attrs": { "color": "red" }
    },
    {
      "type": "wokwi-led",
      "id": "led3",
      "top": 329.23,
      "left": 345.27,
      "attrs": { "color": "red" }
    }
  ],
  "connections": [
    [ "uno:GND.1", "lcd:VSS", "black", [ "v-51", "*", "h0", "v18" ]
],
    [ "uno:GND.1", "lcd:K", "black", [ "v-51", "*", "h0", "v18" ] ],
    [ "uno:GND.1", "lcd:RW", "black", [ "v-51", "*", "h0", "v18" ]
],
    [ "uno:5V", "lcd:VDD", "red", [ "v16", "h-16" ] ],
    [ "uno:5V", "r1:2", "red", [ "v16", "h-118", "v-244", "h50" ] ],
    [ "r1:1", "lcd:A", "pink", [] ],
    [ "uno:12", "lcd:RS", "green", [ "v-16", "*", "h0", "v20" ] ],
    [ "uno:11", "lcd:E", "green", [ "v-20", "*", "h0", "v20" ] ],
    [ "lcd:D4", "uno:5", "green", [ "v43.53", "h76.86" ] ],
    [ "lcd:D5", "uno:4", "green", [ "v36.63", "h75.24" ] ],
    [ "lcd:D6", "uno:3", "green", [ "v26.79", "h78.54" ] ],
    [ "lcd:D7", "uno:2", "green", [ "v52.39", "h79.87" ] ],
    [ "bz1:2", "uno:9", "red", [ "v36.28", "h220.75" ] ],
    [ "bz1:1", "uno:GND.1", "black", [ "v9.69", "h180.53", "v54.16"
] ],
    [ "led1:A", "uno:A0", "red", [ "v26.92", "h-60.72", "v166.42",
"h-77.79" ] ],
    [ "led2:A", "uno:A1", "red", [ "v18.06", "h-50.87", "v108.32",
"h-81.73" ] ],
    [ "led3:A", "uno:A2", "red", [ "v67.3", "h-125.71" ] ],
    [ "led3:C", "uno:GND.2", "black", [ "v47.6", "h-169.87" ] ],
    [ "led2:C", "uno:GND.2", "black", [ "v32.84", "h-17.24",
"v86.66", "h-147.71" ] ],
    [ "led1:C", "uno:GND.2", "black", [ "v3.29", "h-15.27",
"v198.92", "h-150.66" ] ]
  ],
  "dependencies": {}
```

}


wokwi.com/projects/
WOKWI
SAVE
SHARE
Docs
lcd1602.ino
diagram.json
libraries.txt
Library Manager
Simulation
#include <LiquidCrystal.h> // include the Liquid
const int micPin1 = A0; // define the pin for th
const int micPin2 = A1; // define the pin for th
const int micPin3 = A2; // define the pin for th
const int buzzerPin = 9; // define the pin for t
const int ledPin = 6; // define the pin for the
const int contrast = 50; // define the LCD contr
LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // initia
void setup() {
pinMode(buzzerPin, OUTPUT); // set the buzzer
pinMode(ledPin, OUTPUT); // set the LED pin as
lcd.begin(16, 2); // initialize the LCD displa
analogWrite(6,contrast); // set the LCD contra
Serial.begin(9600); // initialize the serial m
}
void loop() {
// read the values from the microphones
int micValue1 = analogRead(micPin1);
int micValue2 = analogRead(micPin2);
int micValue3 = analogRead(micPin3);
// calculate the sound levels in dB for each m
float voltage1 = micValue1 * 5.0 / 1024.0; //
float voltage2 = micValue2 * 5.0 / 1024.0; //
float voltage3 = micValue3 * 5.0 / 1024.0; //
float dB1 = 20 * log10(voltage1/0.0063); // ca
float dB2 = 20 * log10(voltage2/0.0063); // ca
float dB3 = 20 * log10(voltage3/0.0063); // ca
// calculate the average sound level in dB for
float averageDB = (dB1 + dB2 + dB3) / 3;
// display the sound level on the LCD display
lcd.setCursor(0, 0); // set the cursor to the
lcd.print("Sound Level: "); // print the text
lcd.setCursor(0, 1); // set the cursor to the
lcd.print(averageDB); // print the average sou
Serial.print("Sound Level: "); // print the te
Serial.println(averageDB); // print the averag
// control the LED and the buzzer based on the
if (averageDB > 70) { // if the sound level is
digitalWrite(ledPin, HIGH); // turn the LED
tone(buzzerPin, 1000, 500); // turn the buzz
} else { // if the sound level is lower than 7
digitalWrite;
}
}
UNO
ARDUINO

RESULT:




```
Sound Level: 51.99
Sound Level: 52.09
Sound Level: 51.81
Sound Level: 51.99
Sound Level: 52.07
Sound Level: 51.84
```

Conclusion:

In wokwi, By deploying a network of smart sensors equipped with noise detectors, this system continuously gathers real-time acoustic data from various locations. Here we propose an air quality as well as sound pollution monitoring system that allows us to monitor and check live air quality as well as sound pollution in a particular areas through IOT. A sound level meter (SLM) can measure sound at different frequencies (called octave band analysis) and record sound clips to determine the source of noise pollution. Noise monitoring safeguards employees' hearing from any excessive noise in the workplace that leads to hearing problems, insomnia, hypertension, heart disease, ear injuries, and the ringing and buzzing in the ear called tinnitus