**IOT\_Phase3**

**(****Noise Pollution Monitoring)**

**Introduction:**

A noise monitoring system is designed to capture, analyze, and manage environmental noise levels. It helps measure and control noise pollution in various settings, including urban areas, industrial sites, and public spaces.

**Hardware Components:**

First Code Snippet (MicroPython for ESP8266):

* An analog-to-digital converter (ADC) connected to Pin 2, which is used to read analog data.
* The code snippet reads a value from the ADC and prints it.

Second Code Snippet (MicroPython for ESP32):

* The component information is defined in the JSON configuration provided. Here are the components specified in the JSON configuration:
* An ESP32 development kit (type: "wokwi-esp32-devkit-v1") with the ID "esp."This represents the ESP32 microcontroller.
* A microphone component with the ID "mic." This represents a microphone sensor.

**Software Components:**

* Arduino IDE:

The Arduino Integrated Development Environment is used for writing and uploading code to MicroPython for ESP8266 and MicroPython for ESP32.

**System Design:**

ESP8266/ESP32:

Low-cost microcontrollers with Wi-Fi capabilities, ideal for IoT applications.

IoT Connectivity:

Implementing connectivity to transmit noise data to Azure IoT Hub.

**Options include:**

Wi-Fi:

Use onboard Wi-Fi capabilities of microcontrollers.

Cellular:

If remote locations are involved, consider using cellular IoT modules.

LoRa:

For long-range communication in low-power scenarios, LoRaWAN can be suitable.

Power Source:

Deployment scenario, power your IoT device with a suitable power source. This could be a standard power outlet, battery, or energy harvesting solutions for low-power, remote deployments.

Traffic Data Integration :

Incorporate traffic data sources such as traffic cameras, GPS, or traffic flow sensors to obtain information about vehicle counts, speeds, and congestion in the same areas.

Data analytics is crucial for noise monitoring as it helps in extracting valuable insights from the collected noise data.

Alert Response:

Plan for appropriate responses to alerts, such as dispatching noise control teams, notifying residents of possible disturbances, or making real-time traffic management adjustments.

**Simulation Code:**

*{*

*"version": 1,*

*"author": "Kishore M",*

*"editor": "wokwi",*

*"parts": [*

*{*

*"type": "wokwi-esp32-devkit-v1",*

*"id": "esp",*

*"top": -52.9,*

*"left": 62.2,*

*"attrs": { "env": "micropython-20231005-v1.21.0" }*

*},*

*{ "type": "wokwi-microphone", "id": "mic", "top": -16.98, "left": 263.79, "attrs": {} }*

*],*

*"connections": [*

*[ "esp:TX0", "$serialMonitor:RX", "", [] ],*

*[ "esp:RX0", "$serialMonitor:TX", "", [] ],*

*[ "mic:1", "esp:D2", "green", [ "v0" ] ],*

*[ "mic:2", "esp:GND.1", "green", [ "v0" ] ]*

*],*

*"serialMonitor": { "display": "plotter" },*

*"dependencies": {}*

*}*

'''

*from machine import Pin, ADC*

*from time import sleep*

*pot = ADC(Pin(2))*

*pot.atten(ADC.ATTN\_11DB) #Full range: 3.3v*

*#ADC.ATTN\_0DB: Maximum voltage of 1.2V*

*#ADC.ATTN\_2\_5DB: Maximum voltage of 1.5V*

*#ADC.ATTN\_6DB: Maximum voltage of 2.0V*

*#ADC.ATTN\_11DB: Maximum voltage of 3.3V*

*while True:*

*pot\_value = pot.read()*

*print(pot\_value)*

*sleep(0.1)*

*'''*

*import machine, time*

*a = machine.ADC(machine.Pin(32))*

*while True:*

*sample = a.read() # we want 16 bits, a.read() returns 10 bits*

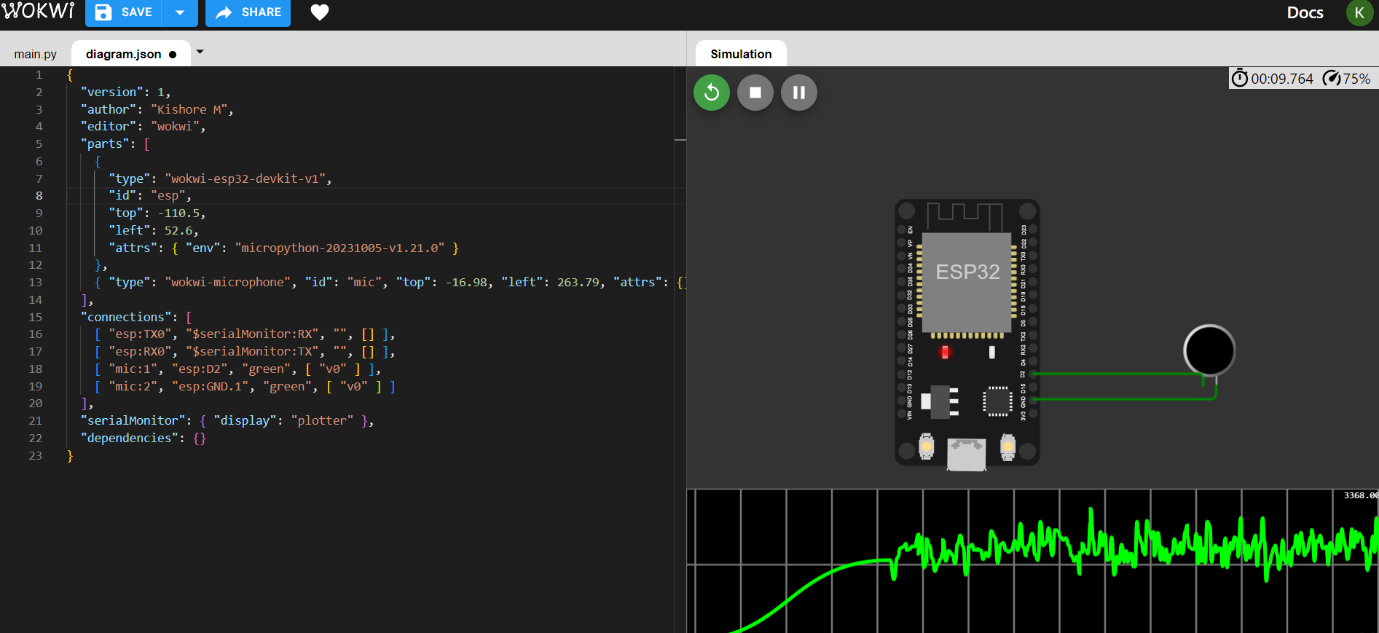
*print(sample)*

*time.sleep(1/44100)*

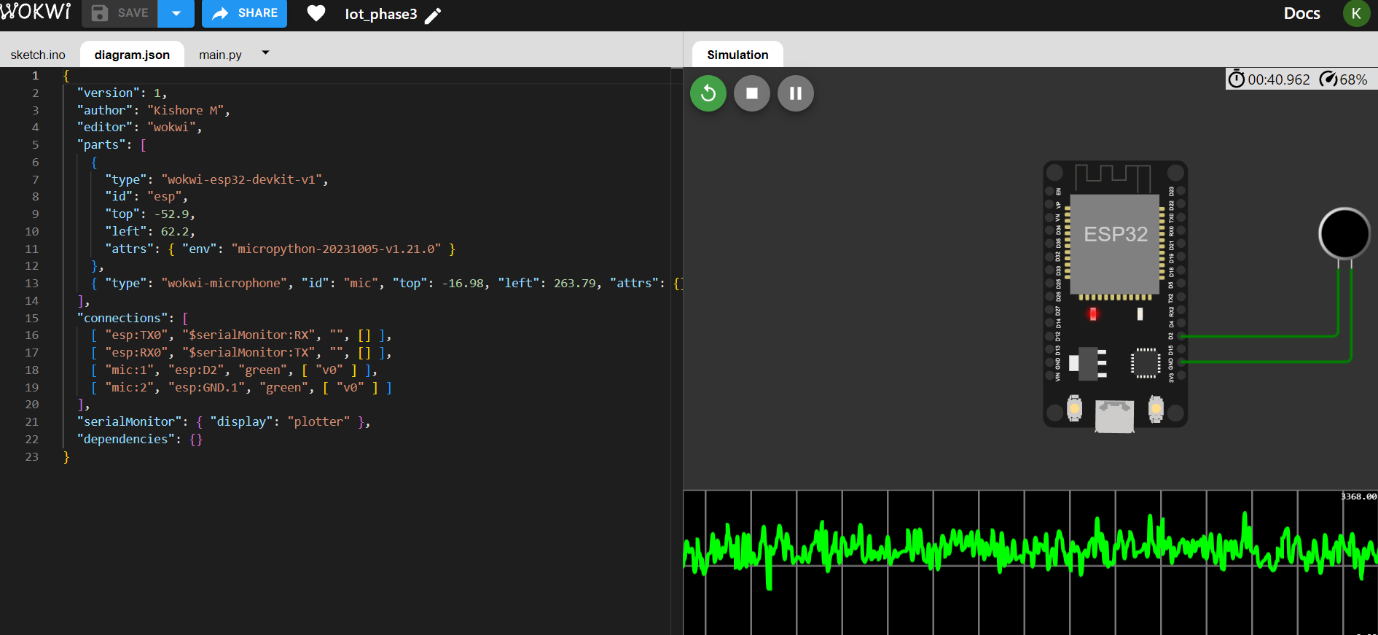
The first snippet is for reading analog input from a potentiometer using an ESP8266 (MicroPython), and the second snippet appears to be for reading analog input from an ADC (Analog-to-Digital Converter) on a MicroPython compatible board.

The second snippet reads analog data using the machine.ADC class and prints the value

The first snippet is designed for the ESP8266 with specific pin and voltage range settings. The second snippet is a more general example of reading analog data using the MicroPython **machine.ADC** class and waiting for a specified time interval between readings. Please make sure to use the appropriate code snippet for your hardware and requirements.



Sample Ouput :



**Simulation link:**

https://wokwi.com/projects/378851393730889729

**Conclusion:**

The conclusion of such a project would typically involve further development, testing, and interaction between various virtual components to achieve a specific goal or test various electronic circuits.

It represents a virtual circuit with an ESP32 and a microphone sensor. It reads analog data from the microphone and prints it to the console.