NOISE POLLUTION MONITORING

A.Meera Benasir _952621106009

S.Veerasamy chettiar college of engineering and technology - 9526, Puliyangudi

PHASE 3:

Development part 1:

To build the IoT Noise pollution monitoring system.

Requirements :

- Wokwi
- Thingspeak
- ESP32
- Relay
- GND
- Vcc
- Register
- Mic
- Microphone
- Noise pollution sensor library

Wokwi :

Wokwi is an embedded systems and IoT simulator supporting ESP32, Arduino, and the Raspberry Pi Pico. Your code never leaves your computer - Wokwi runs the

simulation inside VS Code, using the firmware binaries from your project.

Thinkspeak :

ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize, and analyze live data streams in the cloud. You can send data to ThingSpeak™ from your devices, create instant visualizations of live data, and send alerts using web services like Twitter® and Twilio®

PROCEDURE:

Step 1: Create a New Wokwi Project:

1. Open your web browser and go to the Wokwi website.
2. Sign in with your Wokwi account or create a new account if you don't have one.
3. Click on the "Create New Project" button to create a new project.

Step 2: Select the ESP32 Microcontroller:

1. In the Wokwi editor, click on the "Components" tab on the left sidebar.
2. Search for "ESP32" in the components search bar.
3. Drag and drop the ESP32 microcontroller

component onto the canvas.

Step 3: Add a Virtual Microphone Sensor:

1. In the Wokwi editor, click on the "Components" tab.

2. Create a virtual microphone sensor by combining an analog input component (e.g., potentiometer) and a label.

3. Drag and drop an analog input component onto the canvas.

4. Drag and drop a label component and position it next to the analog input component.

5 .Connect the analog input's output pin to the label's input pin.

6. Double-click the label component to edit its label text, e.g., "Microphone Sensor."

Step 4: Write Code for Noise Pollution Monitoring:

1.Click on the "Code" tab on the left sidebar to access the code editor.

2.Write the code for your ESP32 to read the analog value from the virtual microphone sensor

Step 5: Simulate Noise Pollution Monitoring:

1.The Wokwi simulator will display the virtual ESP32 with the code you wrote.

2.You can interact with the virtual components in real-time.

3.Adjust the potentiometer (virtual microphone sensor) to simulate different sound levels.

4.Observe the sound level readings in the serial monitor.

Step 6 :

Create a ThingSpeak Channel:

Log in to your ThingSpeak account.

Create a new ThingSpeak channel.

Note the Write API Key for this channel.

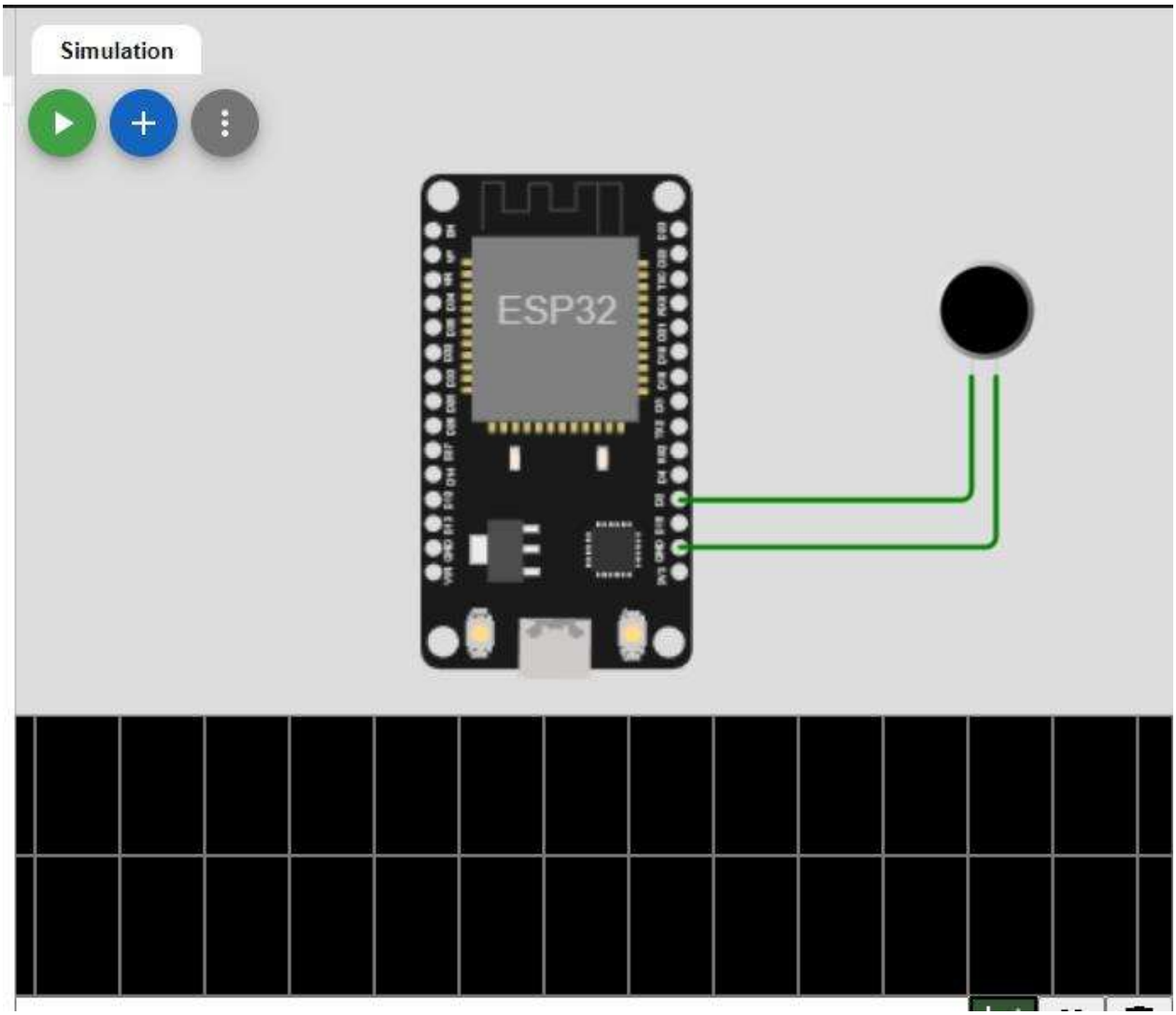
Step 7: Analyze and Extend:

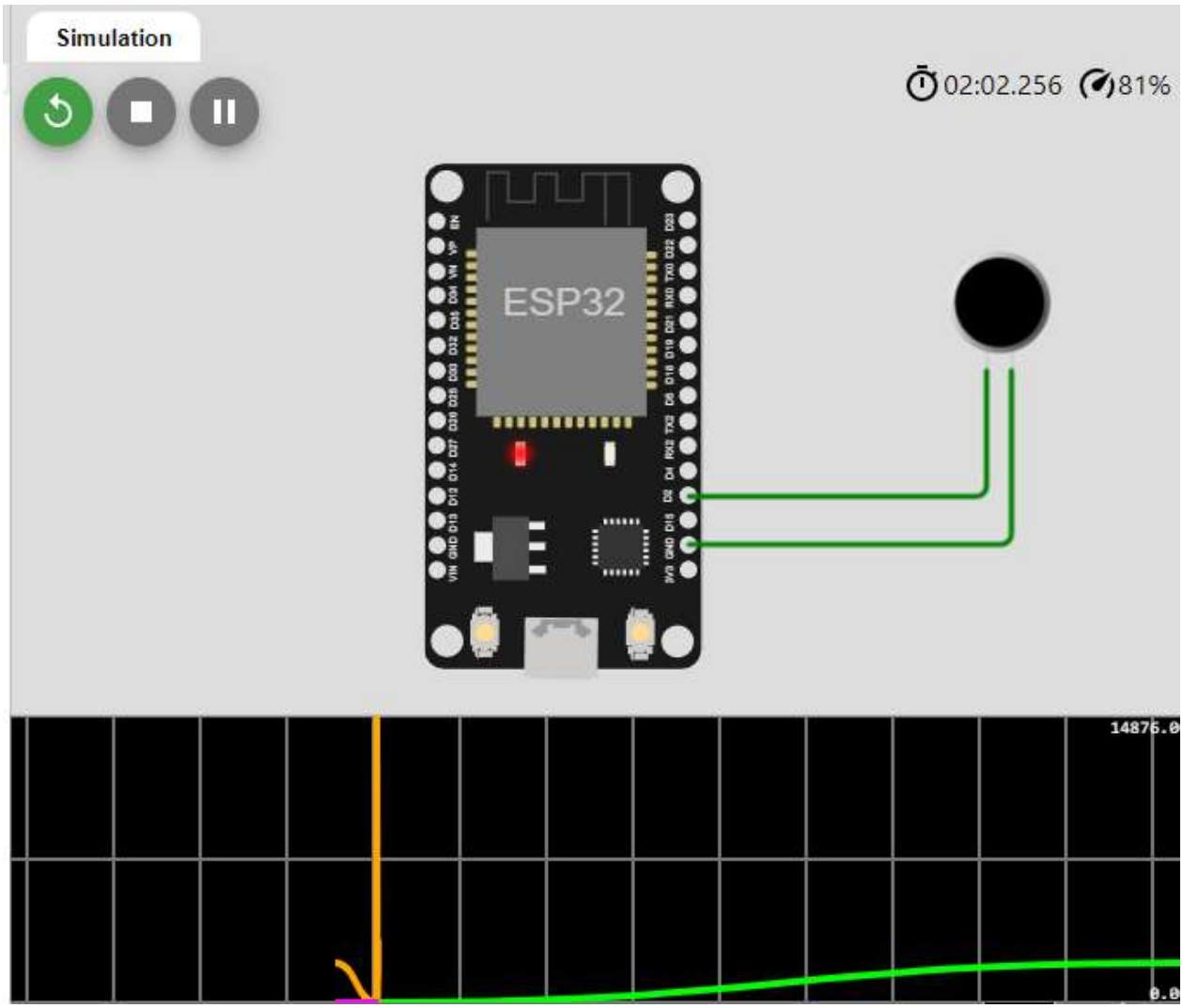
1.Use the simulated data to test and develop noise pollution monitoring algorithms.

2.Extend the simulation by adding features like data logging, real-time sound level calculations, and data visualization.

3.Experiment with different scenarios and configurations to fine-tune your noise pollution monitoring system.

Simulation :


Simulation
ESP32



Source code :

```
#include <WiFi.h>
#include <ThingSpeak.h>
#include "DHTesp.h"
#include "stm32f4xx_hal.h"
#include "stdio.h"

char ssid[] = "Wokwi-GUEST";
```

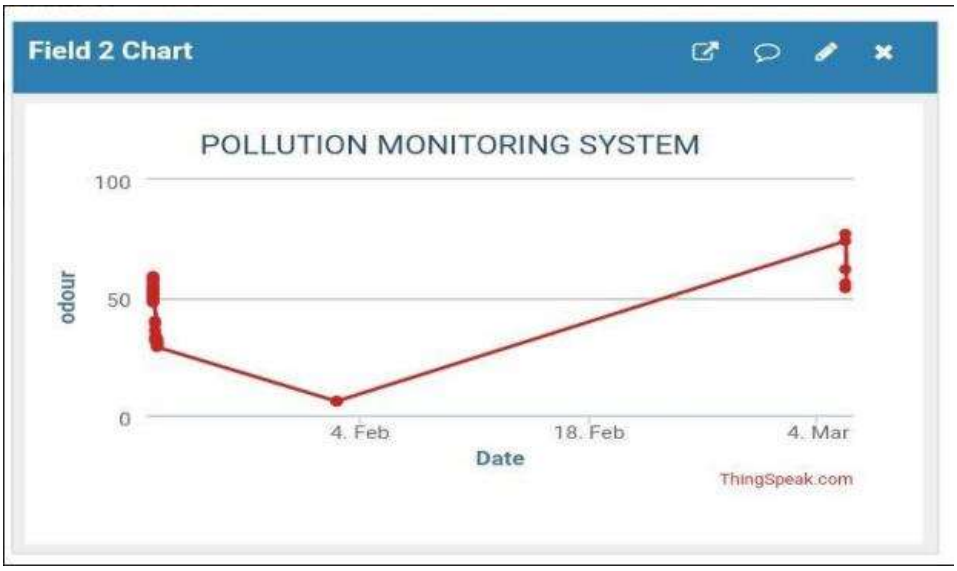
```
char pass[] = "";
WiFiClient client;

unsigned long myChannelNumber = 2066744;
const char *myWriteAPIKey =
"DVRJHBHU6XIIN7UY";
int statusCode;

ADC_HandleTypeDef hadc1;

void Error_Handler(void) {
  while (1) {
    // An error occurred, stay in this loop.
  }
}

void SystemClock_Config(void) {
  // Configure the system clock here.
}

void ADC_Config(void) {
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_ADC1_CLK_ENABLE();

  GPIO_InitTypeDef GPIO_InitStruct;
  GPIO_InitStruct.Pin = GPIO_PIN_0; // Assuming you
are using PA0 for ADC
  GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
```

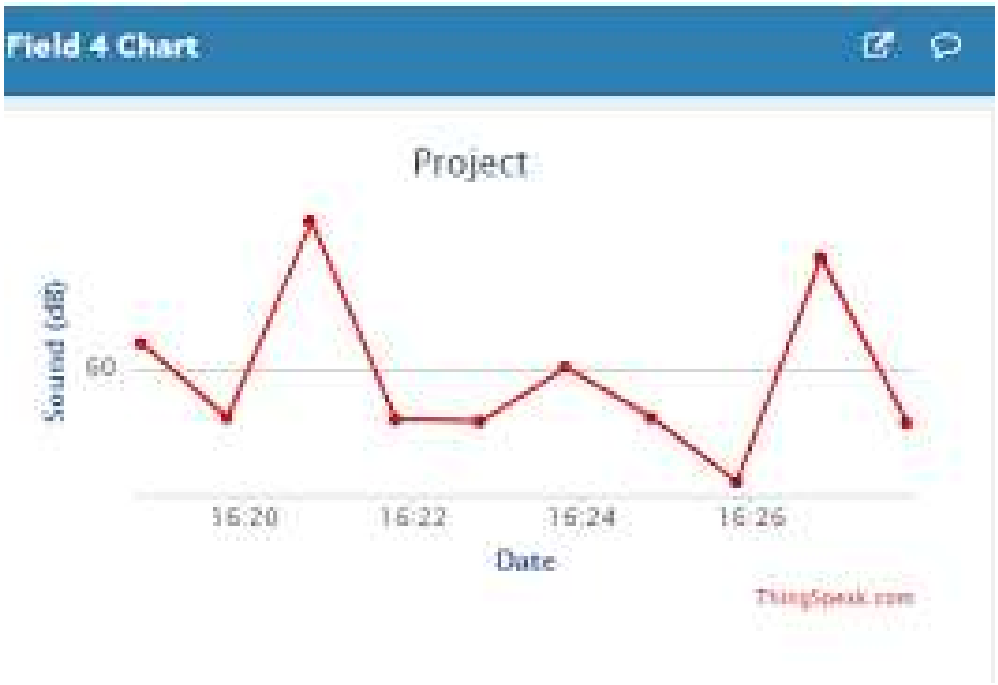
```
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  hadc1.Instance = ADC1;
  hadc1.Init.Resolution = ADC_RESOLUTION_12B;
  hadc1.Init.ScanConvMode = DISABLE;
  hadc1.Init.ContinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConv =
ADC_SOFTWARE_START;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 1;
  hadc1.Init.DMAContinuousRequests = DISABLE;
  hadc1.Init.EOCSelection =
ADC_EOC_SINGLE_CONV;

  if (HAL_ADC_Init(&hadc1) != HAL_OK) {
    Error_Handler();
  }
}

void connectToCloud() {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.print("Attempting to connect");
    while (WiFi.status() != WL_CONNECTED) {
      WiFi.begin(ssid, pass);
      for (int i = 0; i < 5; i++) {
        Serial.print(".");
        delay(1000);
      }
```

```
      Serial.println("\nConnected.");
    }
  }
}

void writeData() {
  // Replace 'Noise' with your actual data
  int Noise = 42; // Example value

  ThingSpeak.setField(1, Noise);
   statusCode =
ThingSpeak.writeFields(myChannelNumber,
myWriteAPIKey);

  if (statusCode == 200) {
    Serial.println("Channel update successful.");
  } else {
    Serial.println("Problem Writing data. HTTP error code:
" + String(statusCode));
  }
  delay(15000); // Data to be uploaded every 15 seconds
}

void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);
  SystemClock_Config();
```

```
  ADC_Config();
}

void loop() {
  connectToCloud();

  uint16_t pot_value;
  if (HAL_ADC_Start(&hadc1) == HAL_OK) {
    if (HAL_ADC_PollForConversion(&hadc1,
HAL_MAX_DELAY) == HAL_OK) {
      pot_value = HAL_ADC_GetValue(&hadc1);
      Serial.printf("ADC Value: %d\n", pot_value);
    }
  }

  writeData();
  delay(1000);
}
```

Real -time data output :


Field 2 Chart
POLLUTION MONITORING SYSTEM
100
50
0
odour
4. Feb
18. Feb
4. Mar
Date
ThingSpeak.com


Field 4 Chart
Project
Sound (dB)
60
16:20
16:22
16:24
16:26
Date
ThingSpeak.com

Source code output :

```
ets Jul 29 2019 12:21:46

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0
x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:4728
load:0x40078000,len:14876
ho 0 tail 12 room 4
load:0x40080400,len:3368
entry 0x400805cc
1573
1712
1894
2129
2199
```