3. Source Code Alat

```
#include <Wire.h>
#include <DFRobot_ADS1115.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266HTTPClient.h>

const int oneWireBus = D7;
const int pinPompa = D6;
const int pinBuzz = D5;
const int pinOtomatis = D3;
const int analogInPin = A0;

OneWire oneWire(oneWireBus);
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -
1);
DallasTemperature sensors(&oneWire);
DFRobot_ADS1115 ads(&Wire);

const char *ssid = "MI 8 Lite";
const char *password = "Persada15";

WiFiClient wifiClient;

const int NUM_SAMPLES = 10;
int buf[NUM_SAMPLES];

const float PH_SENSOR_VOLTAGE_OFFSET = 3.3;
const float PH_SENSOR_SCALE = 1024.0;
const float PH_FORMULA_COEFF = -5.769;
const float PH_FORMULA_OFFSET = 22.17;

const unsigned long DISPLAY_INTERVAL = 2000;

float tb, hasiltb;
boolean Disp = false;
String user = "user1";
unsigned long lastDisplayUpdate = 0;
```

```
void setup(void) {
    Serial.begin(9600);
    if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
        Serial.println(F("SSD1306 allocation failed"));
        while (true);
    }
    display.setTextColor(WHITE);
    connectWiFi();
    initSensors();
}

void loop(void) {
    readSensors();
    displayData();
    sendDataToServer();
    delay(2000);  //
}

void connectWiFi() {
    display.clearDisplay();
    display.setTextSize(2);
    display.setCursor(0, 20);
    display.print("CONNECT..");
    display.display();

    WiFi.mode(WIFI_OFF);
    delay(1000);
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);
    Serial.print("Connecting");

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

    Serial.println("");
    Serial.print("Connected to ");
    Serial.println(ssid);
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());
}

void initSensors() {
    pinMode(pinPompa, OUTPUT);
    digitalWrite(pinPompa, HIGH);
    pinMode(pinOtomatis, INPUT);
```

```
    pinMode(pinBuzz, OUTPUT);

    ads.setAddr_ADS1115(0x48);
    ads.setGain(eGAIN_TWOTHIRDS);
    ads.setMode(eMODE_SINGLE);
    ads.setRate(eRATE_128);
    ads.setOSMode(eOSMODE_SINGLE);
    ads.init();

    sensors.begin();
}

void readSensors() {
    sensors.requestTemperatures();
    float temperatureC = sensors.getTempCByIndex(0);
    temperatureC = max(temperatureC, 15.0f);  // Pastikan suhu
tidak kurang dari 15°C

    if (ads.checkADS1115()) {
        int16_t adc0 = ads.readVoltage(0);
        tb = f_map(adc0, 2500, 4200, 3000, 0);
        tb = constrain(tb, 0, 3000);
        tb = max(tb, 0.0f);  // Pastikan tb tidak kurang dari 0
        Serial.print("Sensor voltage = ");
        Serial.print(adc0);
        Serial.print("mV, tb value = ");
        Serial.println(tb);
    } else {
        Serial.println("ADS1115 Disconnected!");
        tb = 0;  // Setel tb ke 0
    }

    for (int i = 0; i < NUM_SAMPLES; i++) {
        buf[i] = analogRead(analogInPin);
        delay(10);
    }
    std::sort(buf, buf + NUM_SAMPLES);
    float avgValue = 0;
    for (int i = 2; i < 8; i++) {
        avgValue += buf[i];
    }
    avgValue /= 6;
    float pHVoltage = avgValue * PH_SENSOR_VOLTAGE_OFFSET /
PH_SENSOR_SCALE;
    float pHValue = PH_FORMULA_COEFF * pHVoltage +
PH_FORMULA_OFFSET;
```

```
    pHValue = max(pHValue, 5.0f);  // Pastikan pHValue tidak
kurang dari 5
    Serial.print("Sensor voltage = ");
    Serial.print(pHVoltage);
    Serial.print("V, pH value = ");
    Serial.println(pHValue);
}

void displayData() {
    if (millis() - lastDisplayUpdate < DISPLAY_INTERVAL) {
        return;
    }
    lastDisplayUpdate = millis();

    Disp = !Disp;
    display.clearDisplay();
    display.setTextSize(1);

    if (Disp) {
        float temperatureC = sensors.getTempCByIndex(0);
        temperatureC = max(temperatureC, 15.0f);  // Pastikan
suhu tidak kurang dari 15°C

        display.setCursor(0, 0);
        display.print("Temperature: ");
        display.setTextSize(2);
        display.setCursor(0, 10);
        display.print(temperatureC);
        display.print(" ");
        display.setTextSize(1);
        display.cp437(true);
        display.write(167);  // Degree symbol
        display.setTextSize(2);
        display.print("C");

        float pHValue = getpHValue();
        display.setTextSize(1);
        display.setCursor(0, 35);
        display.print("pH: ");
        display.setTextSize(2);
        display.setCursor(0, 45);
        display.print(pHValue);
    } else {
        display.setCursor(0, 0);
        display.print("Turbidity: ");
        display.setTextSize(2);
        display.setCursor(0, 10);
```

```
        hasiltb = max(tb - 2185, 0.0f);  // Pastikan hasiltb
tidak kurang dari 0
        display.print(hasiltb);
    }

    display.display();
}

float getpHValue() {
    // Hitung dan kembalikan nilai pH berdasarkan saat pembacaan
sensor
    float avgValue = 0;
    for (int i = 2; i < 8; i++) {
        avgValue += buf[i];
    }
    avgValue /= 6;
    float pHVoltage = avgValue * PH_SENSOR_VOLTAGE_OFFSET /
PH_SENSOR_SCALE;
    return max(PH_FORMULA_COEFF * pHVoltage + PH_FORMULA_OFFSET,
5.0f);  // Pastikan nilai pH tidak kurang dari 5
}

void sendDataToServer() {
    HTTPClient http;
    String getData, Link, Status, modealat;
    int buttonState = digitalRead(pinOtomatis);

    modealat = (buttonState == LOW) ? "Otomatis" : "Manual";
    Status = (hasiltb < 100) ? "Normal" : "Keruh";

    getData = "?user='" + String(user) + "'&data1='" +
String(getpHValue()) + "'&data2='" + String(hasiltb) +
"'&data3='" + String(sensors.getTempCByIndex(0)) + "'&data4='" +
String(Status) + "'&data5='" + String(modealat) + "'";
    Link = "http://kualitasair.my.id/sync.php" + getData;
    http.begin(wifiClient, Link);
    Serial.print(Link);

    int httpCode = http.GET();
    if (httpCode == 200) {
        Serial.println("Send OK");
        String payload = http.getString();
        int relay[3];
        for (int i = 0; i < 3; i++) {
            relay[i] = char_to_digit(payload[i]);
        }
        if (buttonState == HIGH) {
```

```
            // Manual mode
            digitalWrite(pinPompa, relay[1] ? LOW : HIGH);
            digitalWrite(pinBuzz, relay[0] ? HIGH : LOW);
        }
    } else {
        Serial.println("Send Failed");
    }
    http.end();
}

float f_map(float x, float x1, float x2, float y1, float y2) {
    return (x - x1) * (y2 - y1) / (x2 - x1) + y1;
}

int char_to_digit(char c) {
    return c - '0';
}
```