LAMPIRAN

1. Arduino.ino

```
#include <Wire.h>
#include <SPI.h>
#include "MICS6814.h"
#include <Adafruit_BME280.h>
#include <WiFi.h>
#include <ArduinoJson.h>
#include <PubSubClient.h>

#define WIFI_SSID "4G-UFI-B0B"
#define WIFI_PASSWORD "1234567890"

const char* mqtt_server = "test.mosquitto.org";
const char* mqtt_topic = "node1/udara";
const char* mqtt_will_status_topic = "node1/status";
const char* mqtt_err_status = "node1/err";
const char* mqtt_username = "";
const char* mqtt_password = "";
const int port = 1883;

const int MICS6814_PIN_CO = 4; //A1 32
const int MICS6814_PIN_NH3 = 3; //A2 34
const int MICS6814_PIN_NO2 = 2; //A3 35

WiFiClient espClient;
PubSubClient client(espClient);

MICS6814 gas(MICS6814_PIN_CO, MICS6814_PIN_NO2, MICS6814_PIN_NH3);
Adafruit_BME280 bme; // use I2C interface
#define SEALEVELPRESSURE_HPA (1013.25)

void setup() {
  Serial.begin(115200);
  bool wifi = initWifi();
  pinMode(7, OUTPUT);

  if(wifi){
  digitalWrite(7, 30);
  Serial.print("Calibrating Sensor");
  delay(30);
  gas.calibrate()
  client.setServer(mqtt_server,port);
  mqttCheckConnection();
  Serial.println(" OK!");
  Serial.println(F("BME280 Sensor event test"));
  if (!bme.begin(0x76)) {
    Serial.println(F("Could not find a valid BME280 sensor, check
wiring!"));
    while (1) delay(10);
  }}}

void loop() {
```

```
if(WiFi.status() != WL_CONNECTED){
  digitalWrite(7, LOW);
  Serial.println("Wifi Disconnect");

  initWifi();

  client.setServer(mqtt_server,port);
  mqttCheckConnection();

}else{
  digitalWrite(7, 30);
}

client.loop();
client.setServer(mqtt_server,port);
mqttCheckConnection();
float ppmCO = gas.measure(CO);
float getDataCO = (ppmCO + 0.95)/1.3;
int getDataNH3 = gas.measure(NH3);
double getDataNO2 = gas.measure(NO2);
float getDataTemperature = bme.readTemperature();
float getDataHumidity = bme.readHumidity();
float getDataPressure = bme.readPressure() / 100.0F;

Serial.print("NH3: ");
Serial.print(gas.getResistance(CH_NH3));
Serial.print(getDataNH3);
Serial.println(" ppm");
delay(50);

Serial.print("CO: ");
Serial.print(getDataCO);
Serial.println(" ppm");
delay(50);

Serial.print("NO2: ");
Serial.print(getDataNO2);
Serial.println(" ppm");
delay(50);
Serial.println();

Serial.print("Temperature = ");
Serial.print(getDataTemperature);
Serial.println(" °C");

Serial.print("Humidity = ");
Serial.print(getDataHumidity);
Serial.println(" %");

Serial.print("Pressure = ");
Serial.print(getDataPressure);
Serial.println(" hPa");
```

```
    Serial.print("Approx. Altitude = ");
    Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
    Serial.println(" m");
    Serial.println();

   // convert ppm into microgram in here
      double coMicroGram = (getDataCO * 28.01) / (24.5 * 0.001);
      double Rawno2MicroGram = ((getDataNO2 / 10)  * 46.0055) / (24.5
* 0.001);
      double no2MicroGram = (Rawno2MicroGram + 35.226)/5.3488;
      // check microgram result in here
      if(coMicroGram == 0.000 || no2MicroGram == 0.000){
        // set default
        coMicroGram = 6.000;
        no2MicroGram = 6.000;
      }

      Serial.print("CO Data (microgram/metercubic) :");
      Serial.println(coMicroGram);
      Serial.print("NO2 Data (microgram/metercubic) :");
      Serial.println(no2MicroGram);

      String dataJsonPackage =
jsonDataPackage(coMicroGram,no2MicroGram, getDataTemperature,
getDataHumidity, getDataPressure);
      mqttPublisher(dataJsonPackage,mqtt_topic);


     delay(1000);

}

// json data package in here
String jsonDataPackage(float valueCO, float valueNO2, float ValueTemp,
float ValueHum, float ValuePress){
  const int capacity = JSON_OBJECT_SIZE(5);

  // make json object
  StaticJsonDocument<capacity> jsonData;

  // add a data into json object
  jsonData["CO"] = valueCO;
  jsonData["NO2"] = valueNO2;
  jsonData["Temperature"] = ValueTemp;
  jsonData["Humidity"] = ValueHum;
  jsonData["Pressure"] = ValuePress;

  // convert into string and return string
  String jsonString;
  serializeJson(jsonData,jsonString);
  return jsonString;
}

void mqttCheckConnection(){
```

```
  while(!client.connected()){
    String client_Id = "ESP32-C3-Client-";
    Serial.printf("The client %s connects to the public MQTT
broker\n",client_Id.c_str());

if(client.connect(client_Id.c_str(),mqtt_username,mqtt_password,mqtt_w
ill_status_topic,0,true,"Online")){
      Serial.println("MQTT Broker connected now");
    } else {
      Serial.print("Fail with state");
      Serial.println(client.state());
      initWifi();
      delay(2000);
    }
  }
}

// mqtt publisher in here
void mqttPublisher(String jsonData,String topic){
  client.publish(topic.c_str(),jsonData.c_str());
}

// mqtt err publish in here
void mqttErrDevicePublisher(String message){
  client.publish(mqtt_err_status,message.c_str());
}

bool initWifi(){
  bool isConnect = false;
  int i = 0;
  WiFi.begin(WIFI_SSID,WIFI_PASSWORD);

  while(WiFi.status() != WL_CONNECTED){
    if(i == 60){
      ESP.restart();
    }
    delay(500);
    Serial.print("Connecting to Wifi ");
    Serial.println(i);
    i++;
  }

  Serial.println("Connected to Wifi");


  isConnect = true;
  return isConnect;
}
```

2. Server.js

```
const express = require("express");
const app = express();
const http = require("http");
const cors = require("cors");
```

```
const mqtt = require("mqtt");

function addHours(date, hours) {
  const hoursToAdd = hours * 60 * 60 * 1000;
  date.setTime(date.getTime() + hoursToAdd);
  return date;
}
const nDate = new Date();
const nDate1 = addHours(nDate, 7);

const server = http.createServer(app);
const { Server } = require("socket.io");

app.use(cors());
const SVR = new Server(server, {
  cors: {
    origin: "http://localhost:3000",
    methods: ["GET", "POST"],
  },
});
const mongoose = require("mongoose");

// console.log(nDate);
// mqtt setup
const mqttProtocol = "mqtt";
const mqttHost = "test.mosquitto.org";
const mqttPort = "1883";
const mqttHostUrl = `${mqttProtocol}://${mqttHost}:${mqttPort}`;

// Connect to MongoDB
const uriMongoDB =

"mongodb+srv://rdhrhmtllah:Companel01@cluster0.dsqyq5j.mongodb.net/ISP
Udb?retryWrites=true&w=majority&appName=Cluster0";
mongoose
  .connect(uriMongoDB, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
  })
  .then(() => console.log("mongoDB Connection established"))
  .catch((error) => console.log("MongoDB Connection failed:",
error.message));

const { Schema } = mongoose;
const collecISPU = new Schema({
  id: {
    type: String,
    trim: true,
    lowercase: false,
  },
  PM25: {
    type: Number,
    default: 0,
  },
```

```
  CO: {
    type: Number,
    default: 0,
  },
  NO2: {
    type: Number,
    default: 0,
  },
  Temperature: {
    type: Number,
    default: 0,
  },
  Humidity: {
    type: Number,
    default: 0,
  },
  Pressure: {
    type: Number,
    default: 0,
  },
  created: { type: Date, default: nDate1 },
});
var node1 = mongoose.model("node1", collecISPU);
var node2 = mongoose.model("node2", collecISPU);

let previous_ISPU;
let current_ISPU;

function handleISPU(item, index, arr) {
  if (item <= 4000) {
    Ia = 50;
    Ib = 0;
    xb = 0;
    xa = 4000;
  } else if (item > 4000 && item <= 8000) {
    Ia = 100;
    Ib = 50;
    xb = 4000;
    xa = 8000;
  } else if (item > 8000 && item <= 15000) {
    Ia = 200;
    Ib = 100;
    xb = 8000;
    xa = 15000;
  } else if (item > 15000 && item <= 30000) {
    Ia = 300;
    Ib = 200;
    xb = 15000;
    xa = 30000;
  } else if (item > 30000 && item <= 45000) {
    Ia = 400;
    Ib = 300;
    xb = 30000;
    xa = 45000;
```

```
  }

  arr[index] = ((Ia - Ib) / (xa - xb)) * (item - xb) + Ib;
}

function handleISPUNO(item, index, arr) {
  if (item <= 80) {
    Ia = 50;
    Ib = 0;
    xb = 0;
    xa = 80;
  } else if (item > 80 && item <= 200) {
    Ia = 100;
    Ib = 50;
    xb = 80;
    xa = 200;
  } else if (item > 200 && item <= 1130) {
    Ia = 200;
    Ib = 100;
    xb = 200;
    xa = 1130;
  } else if (item > 1130 && item <= 2260) {
    Ia = 300;
    Ib = 200;
    xb = 1130;
    xa = 2260;
  } else if (item > 2260 && item <= 3000) {
    Ia = 400;
    Ib = 300;
    xb = 2260;
    xa = 3000;
  }

  arr[index] = ((Ia - Ib) / (xa - xb)) * (item - xb) + Ib;
}

function isJsonString(str) {
  try {
    JSON.parse(str);
  } catch (e) {
    return false;
  }
  return true;
}

SVR.on("connection", (socket) => {
  console.log(`socket-${socket.id} connected`);

  const mqttOptions = {
    keepalive: 0,
    clientID: socket.id,
    protocolId: "MQTT",
    protocolVersion: 4,
    clean: true,
```

```
    reconnectPeriod: 1000,
    connectTimeout: 30 * 1000,
  };

  socket.mqttClient = {};
  socket.mqttClient = mqtt.connect(mqttHostUrl, mqttOptions);
  // connect to node
  socket.mqttClient.subscribe(`node1/status`, { qos: 1 });
  socket.mqttClient.subscribe(`node1/udara`, { qos: 1 }, () => {
    console.log(`Terhubung Dengan Node 1`);
  });
  socket.mqttClient.subscribe(`node2/status`, { qos: 1 });
  socket.mqttClient.subscribe(`node2/udara`, { qos: 1 }, () => {
    console.log("Terhubung dengan Node 2");
  });

  // as room in socket.io and as a topic in mqtt
  socket.on("trade_ISPU", (payload) => {
    previous_ISPU = payload.topic[0];
    current_ISPU = payload.topic[1];

    socket.leave(`${previous_ISPU}/status`);
    socket.leave(`${previous_ISPU}/udara`);

    console.log(previous_ISPU, current_ISPU);

    socket.join(`${current_ISPU}/status`);
    socket.join(`${current_ISPU}/udara`);

    if (`${current_ISPU}/udara` == "node1/udara") {
      const baru = node1
        .aggregate(pipeline)
        .limit(1)
        .then((doc) => {
          console.log(doc);
          SVR.to(`${current_ISPU}/udara`).emit("24",
`${JSON.stringify(doc)}`);
        });
      socket.on("datalogDate", function (date) {
        console.log(date.datadate);
        let dateGt = new Date(`${date.datadate} 00:00:00`);
        let dateLt = new Date(`${date.datadate} 23:59:00`);
        let DateGtUTC = addHours(dateGt, 7);
        let DateLtUTC = addHours(dateLt, 7);
        // console.log(DateLtUTC);

        node1
          .aggregate([
            {
              $match: {
                created: {
                  $gt: DateGtUTC,
                  $lt: DateLtUTC,
                },
```

```
              },
            },

            {
              $group: {
                _id: { hour: { $hour: "$created" } }, // Group by hour
in timestamp
                averageValueCO: { $avg: "$CO" },
                averageValueNO: { $avg: "$NO2" },
              },
            },
          ])
          .then((doc) => {
            console.log(doc);
            let ispuCO = [];
            let ispuNO = [];
            let hour = [];

            doc.map((item, index) => {
              ispuCO.push(item.averageValueCO);
              ispuNO.push(item.averageValueNO);
              hour.push(item._id.hour);
            });
            ispuCO.forEach(handleISPU);
            ispuNO.forEach(handleISPUNO);

            SVR.to(`${current_ISPU}/udara`).emit(
              "datalogCO",
              `${JSON.stringify(ispuCO)}`
            );
            SVR.to(`${current_ISPU}/udara`).emit(
              "datalogNO",
              `${JSON.stringify(ispuNO)}`
            );
            SVR.to(`${current_ISPU}/udara`).emit(
              "hour",
              `${JSON.stringify(hour)}`
            );
      });
    } else if (`${current_ISPU}/udara` == "node2/udara") {
      const baru = node2
        .aggregate(pipeline)
        .limit(5)
        .then((doc) => {
          console.log(doc);
          SVR.to(`${current_ISPU}/udara`).emit("24",
`${JSON.stringify(doc)}`);
        });

      socket.on("datalogDate2", function (date) {
        console.log(date.datadate);
        let dateGt = new Date(`${date.datadate} 00:00:00`);
        let dateLt = new Date(`${date.datadate} 23:59:00`);
        let DateGtUTC = addHours(dateGt, 7);
```

```
        let DateLtUTC = addHours(dateLt, 7);
        node2
          .aggregate([
            {
              $match: {
                created: {
                  $gt: DateGtUTC,
                  $lt: DateLtUTC,
                },
              },
            },
            {
              $group: {
                _id: { hour: { $hour: "$created" } }, // Group by hour
in timestamp
                averageValueCO: { $avg: "$CO" },
                averageValueNO: { $avg: "$NO2" },
              },
            },
          ])
          .then((doc) => {
            console.log(doc);
            let ispuCO = [];
            let ispuNO = [];
            let hour = [];

            doc.map((item, index) => {
              ispuCO.push(item.averageValueCO);
              ispuNO.push(item.averageValueNO);
              hour.push(item._id.hour);
            });
            ispuCO.forEach(handleISPU);
            ispuNO.forEach(handleISPUNO);

            SVR.to(`${current_ISPU}/udara`).emit(
              "datalogCO",
              `${JSON.stringify(ispuCO)}`
            );
            SVR.to(`${current_ISPU}/udara`).emit(
              "datalogNO",
              `${JSON.stringify(ispuNO)}`
            );
            SVR.to(`${current_ISPU}/udara`).emit(
              "hour",
              `${JSON.stringify(hour)}`
            );
          });
      });
    }
  });

  socket.on("24Dinamis", (dinamis) => {
    previous_ISPU = dinamis.topic[0];
    current_ISPU = dinamis.topic[1];
```

```
    if (`${current_ISPU}/udara` == "node1/udara") {
      const baru = node1
        .aggregate(pipeline)
        .limit(1)
        .then((doc) => {
          console.log(doc);
          SVR.to(`${current_ISPU}/udara`).emit("24",
`${JSON.stringify(doc)}`);
        });
    }
  });
  socket.mqttClient.on("connect", () => {
    console.log(`mqtt-${socket.id} Connected`);
  });

  socket.mqttClient.on("reconnect", () => {
    console.log(`mqtt-${socket.id} close`);
  });

  socket.mqttClient.on("close", function () {
    console.log(`mqtt-${socket.id} close`);
  });

  // fungsi ambil rata2 24 jam dari database
  const pipeline = [
    {
      $match: {
        created: {
          $gt: new Date(nDate1 - 24 * 60 * 60 * 1000),
        },
      },
    },
    {
      $group: {
        _id: null,
        count: {
          $avg: "$CO",
        },
        countNO: {
          $avg: "$NO2",
        },
      },
    },
  ];


  socket.mqttClient.on("message", function (topic, payload) {
    console.log(`Node: ${topic}, Payload: ${payload}`);

    if (isJsonString(payload)) {
      if (topic == "node1/udara") {
        console.log(`${payload}`);
        var doc = new node1(JSON.parse(payload));
        node1
```

```
          .find({})
          .sort({ created: -1 })
          .limit(1)
          .then((doc) => {
            SVR.to(`${topic}`).emit("message",
`${JSON.stringify(doc)}`);
          })
          .catch((err) => {
            console.log(`Error on fetach document: ${err}`);
          });
      } else if (topic == "node2/udara") {
        console.log(`${payload}`);
        var doc = new node2(JSON.parse(payload));
        node2
          .find({})
          .sort({ created: -1 })
          .limit(1)
          .then((doc) => {
            SVR.to(`${topic}`).emit("message",
`${JSON.stringify(doc)}`);
          })
          .catch((err) => {
            console.log(`Error on fetach document: ${err}`);
          });
      }
    }
  });

  socket.on("disconnect", () => {
    console.log(`socket-${socket.id} disconnected`);
    socket.mqttClient.end();
    delete socket;
  });
});
function datatoMongo() {
  const clientId = "client" + Math.random().toString(36).substring(7);
  const mqttOptions = {
    keepalive: 0,
    clientID: clientId,
    protocolId: "MQTT",
    protocolVersion: 4,
    clean: true,
    reconnectPeriod: 1000,
    connectTimeout: 30 * 1000,
  };

  mqttClient = {};
  mqttClient = mqtt.connect(mqttHostUrl, mqttOptions);

  // telebot

  const telegramBot = require("node-telegram-bot-api");
  const tokenTele = "7006462367:AAGaxTDidp-MCkz-BuGWGlTdR2gOERUI5S0";
  const options = {
```

```
  polling: true,
};
const pipeline = [
  {
    $match: {
      created: {
        $gt: new Date(nDate1 - 24 * 60 * 60 * 1000),
      },
    },
  },
  {
    $group: {
      _id: null,
      count: {
        $avg: "$CO",
      },
      countNO: {
        $avg: "$NO2",
      },
    },
  },
];

const SVRbot = new telegramBot(tokenTele, options);

function sendtele(doc, t4, id) {
  const { createCanvas, loadImage } = require("canvas");
  const fs = require("fs");
  doc = doc;
  t4 = t4;
  id = id;
  mikro = doc[0].count;
  mikroNO = doc[0].countNO;

  let Ia;
  let Ib;
  let xb;
  let xa;

  if (mikro <= 4000) {
    Ia = 50;
    Ib = 0;
    xb = 0;
    xa = 4000;
  } else if (mikro > 4000 && mikro <= 8000) {
    Ia = 100;
    Ib = 50;
    xb = 4000;
    xa = 8000;
  } else if (mikro > 8000 && mikro <= 15000) {
    Ia = 200;
    Ib = 100;
    xb = 8000;
    xa = 15000;
```

```
    } else if (mikro > 15000 && mikro <= 30000) {
      Ia = 300;
      Ib = 200;
      xb = 15000;
      xa = 30000;
    } else if (mikro > 30000 && mikro <= 45000) {
      Ia = 400;
      Ib = 300;
      xb = 30000;
      xa = 45000;
    }

    let IaNO;
    let IbNO;
    let xbNO;
    let xaNO;

    if (mikroNO <= 80) {
      IaNO = 50;
      IbNO = 0;
      xbNO = 0;
      xaNO = 80;
    } else if (mikroNO > 80 && mikroNO <= 200) {
      IaNO = 100;
      IbNO = 50;
      xbNO = 80;
      xaNO = 200;
    } else if (mikroNO > 200 && mikroNO <= 1130) {
      IaNO = 200;
      IbNO = 100;
      xbNO = 200;
      xaNO = 1130;
    } else if (mikroNO > 1130 && mikroNO <= 2260) {
      IaNO = 300;
      IbNO = 200;
      xbNO = 1130;
      xaNO = 2260;
    } else if (mikroNO > 2260 && mikroNO <= 3000) {
      IaNO = 400;
      IbNO = 300;
      xbNO = 2260;
      xaNO = 3000;
    }

    // substitusi hasil klasifikasi konsentrasi senyawa

    let ISPU = ((Ia - Ib) / (xa - xb)) * (mikro - xb) + Ib;

    let ISPUNO = ((IaNO - IbNO) / (xaNO - xbNO)) * (mikroNO - xbNO) +
IbNO;

    const width = 1025;
    const height = 768;
    const canvas = createCanvas(width, height);
```

```
    const ctx = canvas.getContext("2d");
    // Set the coordinates for the image position.
    const imagePosition = {
      w: 1025,
      h: 768,
      x: 0,
      y: 0,
    };
    // Because we are putting the image near the top (y: 75)

    // Bring up the author's Y value as well to make it all
    // fit together nicely.
    let info;

    class bar {
      constructor(ypos, radius, w, loc) {
        this.xpos = 157;
        this.ypos = ypos;
        this.radius = radius;
        this.w = w;
        this.loc = loc;

        if (this.w >= 0 && this.w <= 160) {
          this.color = "#79ac78";
          this.status =
            "Kualitas udara baik,<br>bagus untuk berkegiatan<br>diluar
ruangan.";
        } else if (this.w > 160 && this.w <= 320) {
          this.color = "#59c6ff";
          this.status =
            "Kualitas udara cukup baik,<br>bagus untuk
berkegiatan<br>diluar ruangan.";
        } else if (this.w > 320 && this.w <= 480) {
          this.color = "#ffce59";
          this.status =
            "Kualitas udara kurang baik, <br>gunakan masker
untuk<br>berkegiatan diluar ruangan.";
        } else if (this.w > 480 && this.w <= 640) {
          this.color = "#ff5959";
          this.status =
            "Kualitas udara tidak sehat, <br>tidak baik untuk
berkegiatan <br>diluar ruangan.";
        } else if (this.w > 640) {
          this.color = "#3a3a3a";
          this.status =
            "Kualitas udara berbahaya, <br>tidak disarankan untuk
<br>berkegiatan diluar ruangan.";
        }
        info = this.status.replace(/<br\s*\/?>/gi, " ");
        this.title = this.status.split("<br>");
        this.date = new Date();
      }

      draw(ctx) {
```

```
        ctx.beginPath();

        ctx.shadowBlur = 10;

        ctx.shadowOffsetX = 1;
        ctx.shadowOffsetY = 15;
        ctx.shadowColor = "#b8bfb9";
        ctx.fillStyle = this.color;
        ctx.arc(
          this.xpos + this.w - 21,
          this.ypos,
          this.radius,
          1.5 * Math.PI,
          0.5 * Math.PI
        );

        ctx.fill();

        ctx.rect(
          this.xpos - this.radius,
          this.ypos - this.radius,
          this.w,
          this.radius * 2
        );
        ctx.fill();
      }
      text(ctx) {
        ctx.shadowBlur = 0;
        ctx.shadowOffsetX = 0;
        ctx.shadowOffsetY = 0;
        ctx.shadowColor = "#b8bfb9";
        ctx.fillStyle = "#3a3a3a";
        ctx.font = "40px Poppins";
        ctx.fillText(this.title[0], 410, 660);
        ctx.fillText(this.title[1], 410, 700);
        ctx.fillText(this.title[2], 410, 740);
        ctx.font = "30px Poppins";
        ctx.fillText(this.date.toDateString(), 740, 80);
        ctx.font = "bold 70px Poppins";
        ctx.fillText(this.loc, 160, 245);
      }
    }
    let date = new Date();
    loadImage("tmp.png").then((image) => {
      const { w, h, x, y } = imagePosition;
      ctx.drawImage(image, x, y, w, h);
      // position y, width rect, radius circle, color
      let wCO;
      let wNO;
      if (ISPU <= 50) {
        wCO = 160;
      } else if (ISPU > 50 && ISPU <= 100) {
        wCO = 320;
      } else if (ISPU > 100 && ISPU <= 200) {
```

```
        wCO = 480;
      } else if (ISPU > 200 && ISPU <= 300) {
        wCO = 640;
      } else wCO = 800;

      if (ISPUNO <= 50) {
        wNO = 160;
      } else if (ISPUNO > 50 && ISPUNO <= 100) {
        wNO = 320;
      } else if (ISPUNO > 100 && ISPUNO <= 200) {
        wNO = 480;
      } else if (ISPUNO > 200 && ISPUNO <= 300) {
        wNO = 640;
      } else wNO = 800;

      let loc = t4;
      let CO = new bar(410, 19, wCO, loc);
      let NO = new bar(500, 19, wNO, loc);
      CO.draw(ctx);
      NO.draw(ctx);
      if (wCO > wNO) {
        CO.text(ctx);
      } else NO.text(ctx);

      let caption = `

            Kondisi udara terkini ISPUnet ${date.toDateString()}

${info}
Konsentrasi CO : ${ISPU.toFixed(2)} PPM
Konsentrasi NO2 : ${ISPUNO.toFixed(2)} PPM

untuk info selengkapnya kunjungi http://192.3.113.195:3000/ `;
      const buffer = canvas.toBuffer("image/png");
      // fs.writeFileSync("./image.png", buffer);

      SVRbot.sendPhoto(id, buffer, {
        caption: caption,
      });
    });
  }
  const prefix = "/";
  // lokasi sungai sahang

  const udaraSungaiSahang = new RegExp(`^${prefix}sungai_sahang$`);

  SVRbot.onText(udaraSungaiSahang, async (callback) => {
    const baru = await node1
      .aggregate(pipeline)
      .limit(1)
      .then((doc) => {
        console.log(doc);
        if (doc.length == 0) {
          SVRbot.sendMessage(callback.from.id, "Data terakhir tidak
```

```
ada");
        } else {
          let t4 = "Sungai Sahang";
          let id = callback.from.id;
          sendtele(doc, t4, id);
        }
      });
  });

  // lokasi POLSRI

  const udaraPolsri = new RegExp(`^${prefix}politeknik_sriwijaya$`);

  SVRbot.onText(udaraPolsri, async (callback) => {
    const baru = await node1
      .aggregate(pipeline)
      .limit(1)
      .then((doc) => {
        console.log(doc);
        if (doc.length == 0) {
          SVRbot.sendMessage(callback.from.id, "Data terakhir tidak
ada");
        } else {
          let t4 = "Politeknik Sriwijaya";
          let id = callback.from.id;
          sendtele(doc, t4, id);
        }
      });
  });

  // lokasi SMAN10

  const udaraSma10 = new RegExp(`^${prefix}sma10$`);

  SVRbot.onText(udaraSma10, async (callback) => {
    const baru = await node1
      .aggregate(pipeline)
      .limit(1)
      .then((doc) => {
        console.log(doc);
        if (doc.length == 0) {
          SVRbot.sendMessage(callback.from.id, "Data terakhir tidak
ada");
        } else {
          let t4 = "SMA N 10";
          let id = callback.from.id;
          sendtele(doc, t4, id);
        }
      });
  });

  // lokasi Padang Selasa
  const udaraPadangSelasa = new RegExp(`^${prefix}padang_selasa$`);
```

```
  SVRbot.onText(udaraPadangSelasa, async (callback) => {
    const baru = await node2
      .aggregate(pipeline)
      .limit(1)
      .then((doc) => {
        console.log(doc);
        if (doc.length == 0) {
          SVRbot.sendMessage(callback.from.id, "Data terakhir tidak
ada");
        } else {
          let t4 = "Padang Selasa";
          let id = callback.from.id;
          sendtele(doc, t4, id);
        }
      });
  });

  // lokasi Kambang Iwak
  const udaraKambangiwak = new RegExp(`^${prefix}kambang_iwak$`);

  SVRbot.onText(udaraKambangiwak, async (callback) => {
    const baru = await node2
      .aggregate(pipeline)
      .limit(1)
      .then((doc) => {
        console.log(doc);
        if (doc.length == 0) {
          SVRbot.sendMessage(callback.from.id, "Data terakhir tidak
ada");
        } else {
          let t4 = "Kambang Iwak";
          let id = callback.from.id;
          sendtele(doc, t4, id);
        }
      });
  });

  // connect to node
  mqttClient.subscribe(`node1/udara`, { qos: 1 }, () => {});
  mqttClient.subscribe(`node2/udara`, { qos: 1 }, () => {});

  mqttClient.on("message", function (topic, payload) {
    console.log(`Node: ${topic}, Payload: ${payload}`);

    if (isJsonString(payload)) {
      if (topic == "node1/udara") {
        console.log(`${payload}`);
        var data = new node1(JSON.parse(payload));
        data.save();
      } else if (topic == "node2/udara") {
        console.log(`${payload}`);
        var data = new node2(JSON.parse(payload));
        data.save();
      }
```

```
    }
  });


}
datatoMongo();

server.listen(3001, () => console.log("SERVER IS RUNNING"));
```

3. App.jsx

```
/* eslint-disable no-unused-vars */
import { useEffect, useState, useRef } from "react";
import io from "socket.io-client";
import Navbar from "./components/navbar/navbar";
import Hero from "./components/Hero/hero";
import Map from "./components/map/map";
import Card from "./components/card/card";
import Deskripsi from "./components/deskripsi/deskripsi";
import Chart from "./components/chart/chart";
import Footer from "./components/footer/footer";
import Gotop from "./components/Gotop/goTop";

const socket = io.connect("http://localhost:3001");
function isValidJSON(str) {
  try {
    JSON.parse(str);
  } catch (e) {
    return false;
  }
  return true;
}
// Deklarasi Variable

let previous_ISPU = "";
let current_ISPU = "";

let data_CO;
const CO_BM = 28.01;
let Ia = 0;
let Ib = 0;
let xb = 0;
let xa = 0;
let IaNO = 0;
let IbNO = 0;
let xbNO = 0;
let xaNO = 0;
let kondisiUdara;
let kondisiUdaraNO;
let deskripsi;
let deskripsiNO;
let ISPU;
let ISPUNO;

export default function App() {
  const scrollRef = (useRef < HTMLDivElement) | (null > null);
```

```
  let [location, setLocation] = useState("node1");
  let [datadate, setDatadate] = useState();
  let [title, setTitle] = useState("SUNGAI SAHANG");
  let [displayCO, setDisplayCO] = useState(0);
  let [displayNO, setDisplayNO] = useState(0);
  let [displayTemp, setDisplayTemp] = useState(0);
  let [displayHum, setDisplayHum] = useState(0);
  let [displayPress, setDisplayPress] = useState(0);
  let [display24CO, setDisplay24CO] = useState(20);
  let [display24NO, setDisplay24NO] = useState(20);
  let [mikro, setMikro] = useState(0);
  let [mikroNO, setMikroNO] = useState(0);
  let [width, setWidth] = useState(0);
  let [widthNO, setWidthNO] = useState(0);
  let [datalogCO, setDatalogCO] = useState([]);
  let [hour, setHour] = useState([]);
  let [datalogNO, setDatalogNO] = useState([]);
  let [Warna, setWarna] = useState("#4caf50");
  let [WarnaNO, setWarnaNO] = useState("#4caf50");
  const [sementara, setSementara] = useState([]);
  let doc;

  function handleDataDate(date) {
    setDatadate(date);
  }

  useEffect(() => {
    let nodeDatalog = location == "node1" ? "datalogDate" :
"datalogDate2";
    // console.log(nodeDatalog);
    socket.emit(nodeDatalog, {
      datadate,
    });
    console.log("data tanggal terkirim");
  }, [display24CO]);

  function handleDatachild(Datalocation, title) {
    setLocation(Datalocation);
    setTitle(title);
  }
  const minute_ms = 60000;
  // fungsi connect to server
  useEffect(() => {
    previous_ISPU = current_ISPU;
    current_ISPU = location;
    socket.on("connect", () => {
      console.log(`Connect: ${socket.id}`);

      socket.emit("trade_ISPU", {
        topic: [`${previous_ISPU}`, `${current_ISPU}`],
      });
    });
    socket.io.on("reconnect", (attempt) => {
      previous_ISPU = current_ISPU;
```

```
      current_ISPU = location;
      socket.emit("trade_ISPU", {
        topic: [`${previous_ISPU}`, `${current_ISPU}`],
      });
    });
    socket.emit("trade_ISPU", {
      topic: [`${previous_ISPU}`, `${current_ISPU}`],
    });
    const interval = setInterval(() => {
      console.log("logs every minute");

      socket.emit("trade_ISPU", {
        topic: [`${previous_ISPU}`, `${current_ISPU}`],
      });
    }, minute_ms);
    return () => clearInterval(interval);
  }, []);

 useEffect(() => {
    setWidth(100 - (display24CO / 300) * 100);
     setWidthNO(100 - (display24NO / 300) * 100);
   });

  // akhir fungsi connect
  useEffect(() => {
    previous_ISPU = current_ISPU;
    current_ISPU = location;
    socket.emit("24", {
      topic: [`${previous_ISPU}`, `${current_ISPU}`],
    });
  }, [location]);

  //Pesan CO Dan NO Realtime
  useEffect(() => {
    socket.on("message", (message) => {
      let doc = JSON.parse(message);
      let data_PM;
      let data_NO;
      let data_Temp;
      let data_Hum;
      let data_Press;
      let data_created;

      doc
        .slice()
        .reverse()
        .forEach(function (item, index) {
          data_CO = item.CO;
          data_NO = item.NO2;
          data_PM = item.PM25;
          data_Temp = item.Temperature;
          data_Hum = item.Humidity;
          data_Press = item.Pressure;
```

```
          setDisplayCO((displayCO = data_CO));
          setDisplayNO((displayNO = data_NO));
          setDisplayTemp((displayTemp = data_Temp.toFixed()));
          setDisplayHum((displayHum = data_Hum.toFixed()));
          setDisplayPress((displayPress = data_Press.toFixed()));

          // if (Number(data_CO) >= 0) {
          // }
        });
    });
  }, []);
  // Pesan Kesimpulan 24 jam CO DAN NO
  useEffect(() => {
    socket.on("24", (data24) => {
      if (isValidJSON(data24)) {
        let hitung = JSON.parse(data24);
        let Rawco;
        let Rawno;
        hitung
          .slice()
          .reverse()
          .forEach(function (item, index) {
            Rawco = item.count;
            Rawno = item.countNO;
            setMikro((mikro = Rawco));
            setMikroNO((mikroNO = Rawno));
          });

        // Klasifikasi konsentrasi senyawa dama ug/m3
        if (mikro <= 4000) {
          Ia = 50;
          Ib = 0;
          xb = 0;
          xa = 4000;
        } else if (mikro > 4000 && mikro <= 8000) {
          Ia = 100;
          Ib = 50;
          xb = 4000;
          xa = 8000;
        } else if (mikro > 8000 && mikro <= 15000) {
          Ia = 200;
          Ib = 100;
          xb = 8000;
          xa = 15000;
        } else if (mikro > 15000 && mikro <= 30000) {
          Ia = 300;
          Ib = 200;
          xb = 15000;
          xa = 30000;
        } else if (mikro > 30000 && mikro <= 45000) {
          Ia = 400;
          Ib = 300;
          xb = 30000;
          xa = 45000;
```

```
        }

        if (mikroNO <= 80) {
          IaNO = 50;
          IbNO = 0;
          xbNO = 0;
          xaNO = 80;
        } else if (mikroNO > 80 && mikroNO <= 200) {
          IaNO = 100;
          IbNO = 50;
          xbNO = 80;
          xaNO = 200;
        } else if (mikroNO > 200 && mikroNO <= 1130) {
          IaNO = 200;
          IbNO = 100;
          xbNO = 200;
          xaNO = 1130;
        } else if (mikroNO > 1130 && mikroNO <= 2260) {
          IaNO = 300;
          IbNO = 200;
          xbNO = 1130;
          xaNO = 2260;
        } else if (mikroNO > 2260 && mikroNO <= 3000) {
          IaNO = 400;
          IbNO = 300;
          xbNO = 2260;
          xaNO = 3000;
        }

        // substitusi hasil klasifikasi konsentrasi senyawa

        ISPU = ((Ia - Ib) / (xa - xb)) * (mikro - xb) + Ib;
        setDisplay24CO((display24CO = ISPU.toFixed(2)));

        ISPUNO = ((IaNO - IbNO) / (xaNO - xbNO)) * (mikroNO - xbNO) +
IbNO;
        setDisplay24NO((display24NO = ISPUNO.toFixed(2)));

        // Pengkategorian Kesimpulan ISPU
        if (ISPU <= 50) {
          deskripsi =
            "Kualitas udara baik, bagus untuk berkegiatan diluar
ruangan.";
          kondisiUdara = "Baik";
          setWarna((Warna = "#4caf50"));
          setWidth(20);
        } else if (ISPU > 50 && ISPU <= 100) {
          deskripsi =
            "Kualitas udara cukup baik, bagus untuk berkegiatan diluar
ruangan.";
          kondisiUdara = "Sedang";
          setWarna((Warna = "#2196f3"));
          setWidth(40);
        } else if (ISPU > 100 && ISPU <= 200) {
```

```
          deskripsi =
            "Kualitas udara kurang baik, gunakan masker untuk
berkegiatan diluar ruangan.";
          kondisiUdara = "Tidak Sehat";
          setWarna((Warna = "#ffeb3b"));
          setWidth(80);
        } else if (ISPU > 200 && ISPU <= 300) {
          deskripsi =
            "Kualitas udara tidak sehat, tidak baik untuk berkegiatan
diluar ruangan.";
          kondisiUdara = "Sangat tidak sehat";
          setWarna((Warna = "#f44336"));
          setWidth(80);
        } else if (ISPU > 300) {
          deskripsi =
            "Kualitas udara berbahaya, tidak disarankan untuk
berkegiatan diluar ruangan.";
          kondisiUdara = "Berbahaya";
          setWarna((Warna = "#000000"));
          setWidth(100);
        } else {
          kondisiUdara = "UNDIFINED";
        }

        if (ISPUNO <= 50) {
          deskripsiNO =
            "Kualitas udara baik, bagus untuk berkegiatan diluar
ruangan.";
          kondisiUdaraNO = "Baik";
          setWarnaNO((WarnaNO = "#4caf50"));
          setWidthNO(20);
        } else if (ISPUNO > 50 && ISPUNO <= 100) {
          deskripsiNO =
            "Kualitas udara cukup baik, bagus untuk berkegiatan diluar
ruangan.";
          kondisiUdaraNO = "Sedang";
          setWarnaNO((WarnaNO = "#2196f3"));
          setWidthNO(40);
        } else if (ISPUNO > 100 && ISPUNO <= 200) {
          deskripsiNO =
            "Kualitas udara kurang baik, gunakan masker untuk
berkegiatan diluar ruangan.";
          kondisiUdaraNO = "Tidak Sehat";
          setWarnaNO((WarnaNO = "#ffeb3b"));
          setWidthNO(60);
        } else if (ISPUNO > 200 && ISPUNO <= 300) {
          deskripsiNO =
            "Kualitas udara tidak sehat, tidak baik untuk berkegiatan
diluar ruangan.";
          kondisiUdaraNO = "Sangat tidak sehat";
          setWarnaNO((WarnaNO = "#f44336"));
          setWidthNO(80);
        } else if (ISPUNO > 300) {
          deskripsiNO =
```

```
            "Kualitas udara berbahaya, tidak disarankan untuk
berkegiatan diluar ruangan.";
          kondisiUdaraNO = "Berbahaya";
          setWarnaNO((WarnaNO = "#000000"));
          setWidthNO(100);
        } else {
          kondisiUdaraNO = "UNDIFINED";
        }

        // CO2.textContent = `Kondisi Udara : ${kondisiUdara}, dengan
angka : ${ISPU.toFixed(
        //   1
        // )}`;
      }
    });
  }, [location]);

  useEffect(() => {
    socket.on("datalogCO", (payload) => {
      if (isValidJSON(payload)) {
        doc = JSON.parse(payload);
        console.log(doc);
        if (doc.leght == 0) {
          setDatalogNO();
        } else {
          doc.map((item, index) => {
            let dulu = [];
            const addArray = (prev) => [
              ...prev.slice(0, index),
              item.toFixed(2),
            ];
            dulu = addArray;
            setDatalogCO(dulu);
          });
        }
      }
    });

    socket.on("datalogNO", (payload) => {
      if (isValidJSON(payload)) {
        doc = JSON.parse(payload);
        console.log(doc);
        if (doc.length === 0) {
          console.log("NO kosong");
        } else {
          doc.map((item, index) => {
            let dulu = [];
            const addArray = (prev) => [
              ...prev.slice(0, index),
              item.toFixed(2),
            ];
            dulu = addArray;
            setDatalogNO(dulu);
          });
```

```
        }
      }
    });
    socket.on("hour", (payload) => {
      if (isValidJSON(payload)) {
        doc = JSON.parse(payload);
        console.log(doc);
        if (doc.leght === 0) {
          setDatalogNO();
        } else {
          doc.map((item, index) => {
            let dulu = [];
            const addArray = (prev) => [...prev.slice(0, index),
`${item}:00`];
            dulu = addArray;
            setHour(dulu);
          });
        }
      }
    });
  }, [location]);

  const panjangCO = {
    width: `${width}%`,

    backgroundColor: `${Warna}`,
  };
  const panjangNO = {
    width: `${widthNO}%`,

    backgroundColor: `${WarnaNO}`,
  };
  console.log(datalogCO);
  return (
    <div className="div">
      <Navbar />
      <Gotop />
      <Hero
        desc={deskripsi}
        CO={display24CO}
        panjangCo={panjangCO}
        panjangNo={panjangNO}
        NO={display24NO}
        Temp={displayTemp}
        Hum={displayHum}
        Press={displayPress}
        title={title}
      />
      {/* <Map /> */}
      <Card buttonData={handleDatachild} CO={display24CO}
NO={display24NO} />
      <Deskripsi />
      <Chart
        datalogCO={datalogCO}
```

```
        datalogNO={datalogNO}
        upDate={handleDataDate}
        dataHour={hour}
      />
      <Footer />
    </div>
  );
}
```