LAMPIRAN

1. Tampilan Alat
2. Spesifikasi sensor Turbidity
3. Spesifikasi sensor pH
4. Spesifikasi sensor suhu
5. Spesifikasi sensor TDS
6. Spesifikasi motor pompa
7. Spesifikasi dan *Software* ESP32 Devkit v1

Lampiran 1

Tampilan Alat

a. Tampilan Gateway



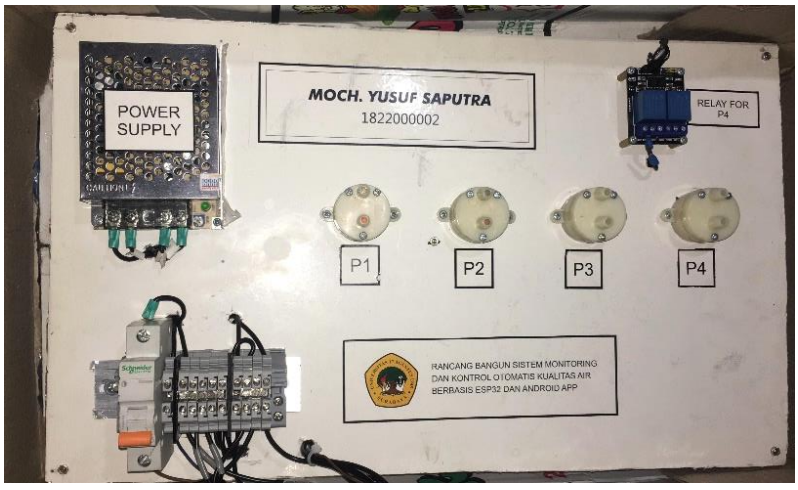
c. Tampilan bagian luar *Node* 1 dan *Node* 2



d. Tampilan bagian dalam *Node* 1 dan *Node* 2



e. Papan Kontrol Motor Pompa DC 12V



Lampiran 2

Spesifikasi sensor Turbidity :

- Tegangan operasional : 5V DC
- Operasi saat ini : 40 Ma (MAX)
- Waktu respon : <500ms
- Resistansi isolasi : 100M (min)
- Metode keluaran :
  - *Output* analog : 0-4 . 5V
  - Keluaran digital : sinyal level tinggi / rendah (dapat disesuaikan nilai ambang batas dengan menyesuaikan potensiometer)
- Suhu operasional : 5 °C ~ 90 °C
- Suhu penyimpanan : - 10 °C ~ 90 °C
- Berat : 30g

Dimensi adaptor : 38mm * 28mm * 10mm/ 1.5inci * 1.1inci *0.4 inci.

Lampiran 3

Spesifikasi sensor pH :

- Tegangan operasional : 5V DC
- Kisaran konsentrasi yang dapat dideteksi: PH0-14
- Kisaran Suhu deteksi: 0-80 ℃
- Waktu respons: 5s
- Waktu penetapan: 60s
- Daya komponen: 0.5W
- Suhu kerja: -10 ~ 50 ℃ (suhu nominal 20 ℃)
- Kelembaban: 95% RH (kelembaban nominal 65% RH)
- Ukuran Modul: 42mm x 32mm x 20mm
- Output: keluaran sinyal tegangan analog
- Dilengkapidengan 4 lubang dudukan M3

Lampiran 4

Spesifikasi sensor suhu :

- Power supply : 3V – 5.5 V
- Konsumsi arus : 1 mA
- Range suhu : -55 sampai 1250C
- Akurasi : ±0.5%
- Resolusi : 9 – 12 bit
- Waktu konversi : < 750 ms

Lampiran 5

Spesifikasi sensor TDS :

- Tegangan Input: 3.3 ~ 5.5V
- Tegangan Output: 0 ~ 2.3V
- Arus Kerja: 3 ~ 6mA
- Pengukuran TDS: 0 ~ 1000ppm
- Akurasi: ± 10% F.S. (25°C)
- Dimensi Modul: 42 x 32 mm
- Pajang Probe 83 cm
- Tipe Output: Tegangan Analog

Lampiran 6

Pada dasarnya. pompa mengubah energi mekanik motor menjadi energi aliran fluida. Energi yang diterima oleh fluida akan digunakan untuk menaikkan tekanan dan mengatasi tahanan – tahanan yang terdapat pada saluran yang dilalui. Pada penelitian ini. digunakan motor pompa 12V DC yang bertujuan untuk mengeluarkan sejumlah air untuk menstabilkan suhu. TDS. pH dan kekeruhan dalam sebuah ekosistem kolam.

Lampiran 7

ESP32 sudah terintegrasi antena .RF balun . *power amplifier* . *low-noise amplifier*. *filter* dan *modul power management* .

Beberapa *software* yang digunakan untuk pemrograman ESP 32 meliputi :

- Arduino IDE
- Espressif IoT Development Framework

ESP32 Devkit v1 hadir dengan *chip serial-to-usb on board* yang memungkinkanpemrograman dan membuka UART modul ESP32. Driver mungkin diperlukan tergantung pada apa sistemnya *(mac* atau *windows*) dan dapat diunduh dari halaman dokumentasi resmi espressif .

Lampiran 8

Program ESP32 pada Proyek Akhir ini terdiri dari beberapa program yang ditulis melalui *software* Arduino IDE yakni sebagai berikut

- Program ESP32 Node 1

```
#include <esp_now.h>
#include <WiFi.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <EEPROM.h>
#include "GravityTDS.h"

#define DS182B0SensorPin 33
#define TdsSensorPin 34
#define turbidiSensorPin 35
#define phSensorPin 36

#define Serial2_TX 12
#define Serial2_RX 36

#define EEPROM_SIZE 512

GravityTDS gravityTds;

float temperature = 25;
float sensorValue;
unsigned long prev_sendToGateway = 0;

OneWire oneWire(DS182B0SensorPin);
DallasTemperature sensorSuhu(&oneWire);
```

```
//ESPNOW PARAMETER
uint8_t broadcastAddress[] = {0xD4, 0xD4, 0xDA, 0x5E, 0x20, 0xC8}; //GATEWAY MAC ADDRESS
esp_now_peer_info_t peerInfo;

typedef struct struct_message
{
  int id;
  float dataSensorSuhu;
  float dataSensorTds;
  float dataSensorPh;
  float dataSensorTurbidi;

} struct_message;
struct_message myData;

void setup()
{
  Serial.begin(115200);  // monitoring
  // Serial2.begin(115200); //receive arduino ph data

  EEPROM.begin(EEPROM_SIZE);

  pinMode(DS182B0SensorPin, INPUT);
  pinMode(TdsSensorPin, INPUT);
  pinMode(phSensorPin, INPUT);
  pinMode(turbidiSensorPin, INPUT);

  sensorSuhu.begin(); // Inisialisasi sensor DS18B20

  gravityTds.setPin(TdsSensorPin);
  gravityTds.setAref(3.3);
  gravityTds.setAdcRange(4096);
  gravityTds.begin();

  WiFi.mode(WIFI_STA); // Set device as a Wi-Fi Station
  WiFi.disconnect();

  // Init ESP-NOW
  if (esp_now_init() != ESP_OK)
  {
    Serial.println("Gagal Memulai ESP-NOW!");
    return;
  }
```

```
 // register for Send CB to get the status of Trasnmitted packet
 esp_now_register_send_cb(OnDataSent);

 // Register peer
 memcpy(peerInfo.peer_addr, broadcastAddress, 6);
 peerInfo.channel = 0;
 peerInfo.encrypt = false;

 // Add peer
 if (esp_now_add_peer(&peerInfo) != ESP_OK)
 {
  Serial.println("Failed to add peer");
  return;
 }
 Serial2.begin(115200); //receive arduino ph data
}

void loop()
{
 sensorValue = analogRead(turbidiSensorPin);

 if (myData.dataSensorTurbidi > 200) {
  myData.dataSensorTurbidi = 0;
 }

 if (myData.dataSensorTurbidi < 0) {
  myData.dataSensorTurbidi = 0;
 }

 if ((millis() - prev_sendToGateway) > 2000)
 {
  myData.id = 1;
  myData.dataSensorSuhu = ds18b20_sensor();
  myData.dataSensorTds = tds_sensor();
  myData.dataSensorTurbidi = turbidiSensor();
  SendToGateway();
 }

 if (Serial2.available() > 0)
 {
  Serial.println("Data masuk");
  delay(20);
  String data = Serial2.readString();
  myData.dataSensorPh = parseData(data);
```

```
  }
}

float ds18b20_sensor()
{
  sensorSuhu.requestTemperatures();
  return sensorSuhu.getTempCByIndex(0);
}

float tds_sensor()
{
  analogRead(33);
  float analogValue = analogRead(34);
  analogRead(35);
  analogRead(36);
  gravityTds.setTemperature(temperature); // Set the temperature and execute temperature compensation
  gravityTds.update(analogValue);
  return gravityTds.getTdsValue(); // Then get the value
}

float turbidiSensor()
{
  //masih contoh program
  //return random(10, 20) / 1.4;
  return (-1.333 * analogRead(35) + 3693 + 60.0);
  //return -1.333 * sensorValue + 3693;

}

void SendToGateway()
{
  // Variabel yang dikirim menuju Gateway (variabel matengan)
  Serial.println("");
  Serial.print("Suhu:");
  Serial.print(myData.dataSensorSuhu); // Presisi 2 digit (satu angka dibelakang koma)
  Serial.println(" °C");

  Serial.print("TDS: ");
  Serial.print(myData.dataSensorTds);
  Serial.println(" ppm");

  Serial.print("PH: ");
  Serial.print(myData.dataSensorPh);
  Serial.println(" ppm");
```

```
  Serial.print("Turbidity: ");
  Serial.print(myData.dataSensorTurbidi);
  Serial.println(" NTU");
  Serial.println("");

  // Mengirim pembacaan data sensor melalui ESP-NOW
  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *)&myData, sizeof(myData));

  if (result == ESP_OK)
  {
    Serial.println("Data Berhasil Terkirim");
  }
  else
  {
    Serial.println("Gagal Mengirim Data");
  }
}

// void callback ketika data telah terkirim
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status)
{
  Serial.print("Status Pengiriman Data Terakhir Node 1:\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Sukses" : "Gagal");
}

float parseData(String data)
{
  // Find the last occurrence of '\n' character
  int lastIndex = data.lastIndexOf('\n');
  int lastIndex_prev = data.lastIndexOf('\n', lastIndex - 1);

  // Extract the substring after the last newline character
  String latestValueString = data.substring(lastIndex_prev, lastIndex);
  Serial.println(latestValueString);
  float latestValue = latestValueString.toFloat();

  return latestValue;
}
```

- Program ESP32 Node 2

```
#include <esp_now.h>
#include <WiFi.h>
#include <OneWire.h>
```

```
#include <DallasTemperature.h>
#include <EEPROM.h>
#include "GravityTDS.h"

#define DS182B0SensorPin 33
#define TdsSensorPin 34
#define turbidiSensorPin 35
#define phSensorPin 36

#define Serial2_TX 12
#define Serial2_RX 36

#define EEPROM_SIZE 512

GravityTDS gravityTds; // Deklarasi variabel penyimpanan data untuk sensor TDS

float temperature = 25;
unsigned long prev_sendToGateway = 0;

OneWire oneWire(DS182B0SensorPin);
DallasTemperature sensorSuhu(&oneWire);

//ESPNOW PARAMETER
uint8_t broadcastAddress[] = {0xD4, 0xD4, 0xDA, 0x5E, 0x20, 0xC8}; //GATEWAY MAC ADDRESS
esp_now_peer_info_t peerInfo;

typedef struct struct_message
{
  int id;
  float dataSensorSuhu;
  float dataSensorTds;
  float dataSensorPh;
  float dataSensorTurbidi;
} struct_message;
struct_message myData;

void setup()
{
  Serial.begin(115200);  // monitoring
  //Serial2.begin(115200); //receive arduino ph data

  EEPROM.begin(EEPROM_SIZE);

  pinMode(DS182B0SensorPin, INPUT);
```

```
 pinMode(TdsSensorPin, INPUT);
 pinMode(phSensorPin, INPUT);
 pinMode(turbidiSensorPin, INPUT);

 // Inisialisasi sensor DS18B20
 sensorSuhu.begin();

 // inisialisasi TDS
 gravityTds.setPin(TdsSensorPin);
 gravityTds.setAref(3.3);
 gravityTds.setAdcRange(4096);
 gravityTds.begin();

 WiFi.mode(WIFI_STA); // Set device as a Wi-Fi Station
 WiFi.disconnect();

 // Init ESP-NOW
 if (esp_now_init() != ESP_OK)
 {
  Serial.println("Gagal Memulai ESP-NOW!");
  return;
 }

 // register for Send CB to get the status of Trasnmitted packet
 esp_now_register_send_cb(OnDataSent);

 // Register peer
 memcpy(peerInfo.peer_addr, broadcastAddress, 6);
 peerInfo.channel = 0;
 peerInfo.encrypt = false;

 // Add peer
 if (esp_now_add_peer(&peerInfo) != ESP_OK)
 {
  Serial.println("Failed to add peer");
  return;
 }

Serial.println("test setup");
}



void loop()
```

```
{
  Serial.println("teststststst");
  if ((millis() - prev_sendToGateway) > 2000)
  {
    myData.id = 2;
    myData.dataSensorSuhu = ds18b20_sensor();
    myData.dataSensorTds = tds_sensor();
    myData.dataSensorTurbidi = turbidiSensor();
    SendToGateway();
  }

  if (Serial2.available() >= 0)
  {
    delay(20);
    String data = Serial2.readString();
    myData.dataSensorPh = parseData(data);
  }
}

float ds18b20_sensor()
{
  sensorSuhu.requestTemperatures();
  return sensorSuhu.getTempCByIndex(0);
}

float tds_sensor()
{
  analogRead(33);
  float analogValue = analogRead(34);
  analogRead(35);
  analogRead(36);
  gravityTds.setTemperature(temperature); // Set the temperature and execute temperature compensation
  gravityTds.update(analogValue);
  return gravityTds.getTdsValue();
}

float turbidiSensor()
{
  //masih contoh program
  return random(10, 20) / 1.4;
}

void SendToGateway()
{
```

```
  // Variabel yang dikirim menuju Gateway (variabel matengan)
  Serial.println("");
  Serial.print("Suhu:");
  Serial.print(myData.dataSensorSuhu); // Presisi 2 digit (satu angka dibelakang koma)
  Serial.println(" °C");

  Serial.print("TDS: ");
  Serial.print(myData.dataSensorTds);
  Serial.println(" ppm");

  Serial.print("PH: ");
  Serial.print(myData.dataSensorPh);
  Serial.println(" ppm");

  Serial.print("Turbidity: ");
  Serial.print(myData.dataSensorTurbidi);
  Serial.println(" NTU");
  Serial.println("");

  // Mengirim pembacaan data sensor melalui ESP-NOW
  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *)&myData, sizeof(myData));

  if (result == ESP_OK)
  {
    Serial.println("Data Berhasil Terkirim");
  }
  else
  {
    Serial.println("Gagal Mengirim Data");
  }
}

// void callback ketika data telah terkirim
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status)
{
  Serial.print("Status Pengiriman Data Terakhir Node 2:\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Sukses" : "Gagal");
}

float parseData(String data)
{
  // Find the last occurrence of '\n' character
  int lastIndex = data.lastIndexOf('\n');
  int lastIndex_prev = data.lastIndexOf('\n', lastIndex - 1);
```

```
  // Extract the substring after the last newline character
  String latestValueString = data.substring(lastIndex_prev, lastIndex);

  float latestValue = latestValueString.toFloat();

  return latestValue;
}
```

- Program ESP32 *Gateway*

```
#include <SoftwareSerial.h>
#include <esp_now.h>
#include <WiFi.h>

SoftwareSerial mySerial(23, 22); // RX, TX
String kirimData;

// use volatile if you are dealing with interrupt
volatile bool node1_received = false, node2_received = false;

typedef struct struct_message
{
  int id; // Id board ESP32 (1 and 2)
  float dataSensorSuhu;
  float dataSensorTds;
  float dataSensorPh;
  float dataSensorTurbidi;
} struct_message;

struct_message boardsStruct[2];
struct_message myData;

// callback function that will be executed when data is received
void OnDataRecv(const uint8_t *mac_addr, const uint8_t *incomingData, int len)
{
  // copy data to the same struct as struct sender format
  memcpy(&myData, incomingData, sizeof(myData));

  // notify monitor
  Serial.print("Message received: Node-");
  Serial.println(myData.id);

  // penyesuaian id dengan indeks penyimpanan struct
  int index = -1;
```

```
  if (myData.id == 1)
    index = 0;
  else if (myData.id == 2)
    index = 1;

  boardsStruct[index].dataSensorSuhu = myData.id;
  boardsStruct[index].dataSensorSuhu = myData.dataSensorSuhu;
  boardsStruct[index].dataSensorTds = myData.dataSensorTds;
  boardsStruct[index].dataSensorPh = myData.dataSensorPh;
  boardsStruct[index].dataSensorTurbidi = myData.dataSensorTurbidi;

  // update flag after data copied successfully
  if (myData.id == 1)
    node1_received = true;
  else if (myData.id == 2)
    node2_received = true;
}

void setup()
{
  //Initialize Serial Monitor
  Serial.begin(115200);
  mySerial.begin(9600);

  //Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);

  // Inisialisasi ESP-NOW
  if (esp_now_init() != ESP_OK)
  {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // register callback on received espnow message
  esp_now_register_recv_cb(OnDataRecv);
}

void loop()
{
  if (node1_received)
  {
    sendDataToMaster(1); // ID HARUS DIATUR
    node1_received = false;
```

```
  }
  else if (node2_received)
  {
    sendDataToMaster(2); // ID HARUS DIATUR
    node2_received = false;
  }
}

void sendDataToMaster(int id)
{
  // penyesuaian id dengan indeks penyimpanan struct
  int index = -1;
  if (id == 1)
    index = 0;
  else if (id == 2)
    index = 1;

  // send data to monitor
  Serial.print("send UART with data: ");
  Serial.print(id);
  Serial.print(",");
  Serial.print(boardsStruct[index].dataSensorSuhu, 2);
  Serial.print(",");
  Serial.print(boardsStruct[index].dataSensorTds, 2);
  Serial.print(",");
  Serial.print(boardsStruct[index].dataSensorPh, 2);
  Serial.print(",");
  Serial.println(boardsStruct[index].dataSensorTurbidi, 2);
  Serial.println();

  // send to master
  mySerial.print(id);
  mySerial.print(",");
  mySerial.print(boardsStruct[index].dataSensorSuhu, 2);
  mySerial.print(",");
  mySerial.print(boardsStruct[index].dataSensorTds, 2);
  mySerial.print(",");
  mySerial.print(boardsStruct[index].dataSensorPh, 2);
  mySerial.print(",");
  mySerial.println(boardsStruct[index].dataSensorTurbidi, 2);
  delay(300); //tunggu master memproses
}
```

- Program ESP32 Master

```
#include <SoftwareSerial.h>
#include <LiquidCrystal.h>


#define BLYNK_TEMPLATE_ID "TMPL6tSnqTjSN"
#define BLYNK_TEMPLATE_NAME "Water Monitoring IoT"
#define BLYNK_AUTH_TOKEN "BxW_8J1XZdkC-w5LfG9kokZ861itwYH9"

#include <BlynkSimpleEsp32.h>
#define pin_1 18 // ph down
#define pin_3 19 // ph up
#define pin_21 21 // disposal
#define pin_19 4 // pengisian

LiquidCrystal lcd(14, 27, 26, 25, 33, 32);

char ssid[] = "MDR 34";
char pass[] = "terserah";

unsigned long turbidi_time = 0;

typedef struct struct_message
{
  int id; // Id board ESP32 (1 and 2)
  float dataSensorSuhu;
  float dataSensorTds;
  float dataSensorPh;
  float dataSensorTurbidi;
} struct_message;
struct_message boardsStruct[2];

typedef enum
{
  STATE_INIT,
  STATE_DISPLAY_NODE1,
  STATE_DISPLAY_NODE2
} LCDState;
LCDState lcdState = STATE_INIT;

SoftwareSerial mySerial(23, 22); // RX, TX
unsigned long previousMillis = 0, LCDInterval = 1000;

void setup()
{
  Serial.begin(115200);
  mySerial.begin(9600);
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);  // Memulai Blynk IoT

  //timer.setInterval(3000L, sendToBlynk);

  pinMode(pin_3, OUTPUT);
  pinMode(pin_1, OUTPUT);
  pinMode(pin_21, OUTPUT);
  pinMode(pin_19, OUTPUT);

  digitalWrite(pin_19, HIGH);

  lcd.begin(20, 4);
  lcd.clear();
  lcd.print("Please wait..");
  delay(3000);
}
```

```
void loop()
{
 //Serial.println("masuk");

 Blynk.run();
 //timer.run();
 //Serial.println("masuk 2");

 /* *********
 // UPDATE DATA DARI GATEWAY
 ********* */
 if (mySerial.available() > 0)
 {
  Serial.println("DATA SERIAL MASUK");

  float turb1 = boardsStruct[0].dataSensorTurbidi;
  // Serial.print("turb1= ");
  // Serial.println (turb1);

  // wait until '\n character arive
  String receivedData = mySerial.readStringUntil('\n');
  Serial.println(receivedData);

  // Split the received data by ","
  int index = receivedData.indexOf(',');
  String idStr = receivedData.substring(0, index);
  receivedData.remove(0, index + 1);

  index = receivedData.indexOf(',');
  String suhuStr = receivedData.substring(0, index);
  receivedData.remove(0, index + 1);

  index = receivedData.indexOf(',');
  String tdsStr = receivedData.substring(0, index);
  receivedData.remove(0, index + 1);

  index = receivedData.indexOf(',');
  String phStr = receivedData.substring(0, index);
  receivedData.remove(0, index + 1);

  // identify id, used to access boardsStruct index
  int dataID = idStr.toInt();

  // penyesuaian id dengan indeks penyimpanan struct
  index = -1;
  if (dataID == 1)
    index = 0;
  else if (dataID == 2)
    index = 1;

  // save data to struct
  boardsStruct[index].id = dataID;
  boardsStruct[index].dataSensorSuhu = suhuStr.toFloat();
  boardsStruct[index].dataSensorTds = tdsStr.toFloat();
  boardsStruct[index].dataSensorPh = phStr.toFloat();
  boardsStruct[index].dataSensorTurbidi = receivedData.toFloat(); // Since it's the last value
  if (boardsStruct[index].dataSensorTurbidi >400)
  boardsStruct[index].dataSensorTurbidi=0;
 }

 /* *********
 // UPDATE LCD
 ********* */
```

```
 if (millis() - previousMillis >= LCDInterval)
 {
  printLCD();
  sendToBlynk();
  previousMillis = millis();
 }

 /* **********
 // PERIKSA PH
 ********** */
 // if (boardsStruct[0].dataSensorPh > 8 && boardsStruct[0].dataSensorPh > 8)
 if (boardsStruct[0].dataSensorPh > 8)
 {
  // pin 1 ON sampai ph = 7 //ph down
  digitalWrite(pin_1, HIGH);
  digitalWrite(pin_3, LOW);
 }
 else if (boardsStruct[0].dataSensorPh < 5)
 {
  // pin 3 ON sampai ph = 7 //ph up
  digitalWrite(pin_3, HIGH);
  digitalWrite(pin_1, LOW);
 }
 else
 {
  digitalWrite(pin_3, LOW);
  digitalWrite(pin_1, LOW);
 }

 /* **********
 // PERIKSA TURBIDI
 ********** */
 // if (boardsStruct[0].dataSensorTurbidi >= 100 && boardsStruct[0].dataSensorTurbidi >= 100)
 if (boardsStruct[0].dataSensorTurbidi > 100)
 {
  if ((millis() - turbidi_time) < 10000) //10 DETIK
  {
   digitalWrite(pin_19, HIGH); //ACTIVE LOW
   digitalWrite(pin_21, HIGH); //DISPOSAL ON
  }
  else if ((millis() - turbidi_time) < 20000) //20 DETIK
  {
   digitalWrite(pin_19, LOW); //ACTIVE LOW
   digitalWrite(pin_21, LOW); //DISPOSAL OFF
   Serial.println("PENGISIAN ON");
  }
  else
  {
   digitalWrite(pin_19, HIGH);
   digitalWrite(pin_21, LOW);
   turbidi_time = millis();
  }
 }
}

void sendToBlynk() {
   Blynk.virtualWrite(V0, boardsStruct[0].dataSensorSuhu);
   Blynk.virtualWrite(V8, boardsStruct[0].dataSensorTds);
   Blynk.virtualWrite(V2, boardsStruct[0].dataSensorPh);
   Blynk.virtualWrite(V9, boardsStruct[0].dataSensorTurbidi);

   Blynk.virtualWrite(V4, boardsStruct[1].dataSensorSuhu);
   Blynk.virtualWrite(V5, boardsStruct[1].dataSensorTds);
```

```
    Blynk.virtualWrite(V6, boardsStruct[1].dataSensorPh);
    Blynk.virtualWrite(V7, boardsStruct[1].dataSensorTurbidi);
}

void printLCD()
{
  switch (lcdState)
  {
  case STATE_INIT:
    lcd.clear();
    //lcd.print("master node...");
    lcdState = STATE_DISPLAY_NODE1;
    break;

  case STATE_DISPLAY_NODE1:
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("SUHU  N.1 = ");
    lcd.print(boardsStruct[1].dataSensorSuhu);
    lcd.setCursor(0, 1);
    lcd.print("TDS   N.1 = ");
    lcd.print(boardsStruct[0].dataSensorTds);
    lcd.setCursor(0, 2);
    lcd.print("pH    N.1 = ");
    lcd.print(boardsStruct[0].dataSensorPh);
    lcd.setCursor(0, 3);
    lcd.print("Tbdty N.1 = ");
    lcd.print(boardsStruct[0].dataSensorTurbidi);
    lcdState = STATE_DISPLAY_NODE2;
    break;

  case STATE_DISPLAY_NODE2:
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("SUHU  N.2 = ");
    lcd.print(boardsStruct[1].dataSensorSuhu);
    lcd.setCursor(0, 1);
    lcd.print("TDS   N.2 = ");
    lcd.print(boardsStruct[1].dataSensorTds);
    lcd.setCursor(0, 2);
    lcd.print("pH    N.2 = ");
    lcd.print(boardsStruct[0].dataSensorPh);
    lcd.setCursor(0, 3);
    lcd.print("Tbdty N.2 = ");
    lcd.print(boardsStruct[0].dataSensorTurbidi);
    lcdState = STATE_DISPLAY_NODE1;
    break;
  }
```

BIOGRAFI



Moch. Yusuf Saputra lahir di Surabaya. 16 September 1999. Berdomisili di Kecamatan Gunung Anyar. Kota Surabaya. Penulis telah mengenyam pendidikan formal tingkat dasar di SD Negeri 273 Surabaya dan lulus pada tahun 2012 dan dilanjutkan pendidikan di SMP Negeri 23 Surabaya dan SMK Negeri 3 Surabaya sehingga lulus pada tahun 2018. Selanjutnya penulis berhasil masuk sebagai mahasiswa Program Studi Teknologi Listrik di Universitas 17 Agustus 1945 Surabaya. Selama berada di bangku perkuliahan. penulis merupakan mahasiswa yang turut aktif dalam beberapa kegiatan kampus. Penulis mengucapkan rasa syukur yang sebesar-besarnya atas terselesainya proyek akhir yang berjudul **"Rancang Bangun Sistem Monitoring dan Kontrol Otomatis Kualitas Air Berbasis ESP32 dan Android App".**

Informasi Penulis

- Nama : Moch. Yusuf Saputra
- Motto : "*Dream big. work hard"*
- Phone : +6283865270313
- Alamat : Jalan Gunung Anyar Tengah 7/35 Surabaya

Latar Belakang Pendidikan Formal

- SD Negeri 273 Surabaya
- SMP Negeri 23 Surabaya
- SMK Negeri 3 Surabaya
- Universitas 17 Agustus 1945 Surabaya

8.