Lampiran 3 Source Code

```
#define BLYNK_TEMPLATE_ID "TMPL66Qw4M3x-"
#define BLYNK_TEMPLATE_NAME "Sistem Monitoring Air"
#define BLYNK_AUTH_TOKEN "z7IZkH19-z4lU3j0z46jZ-9gWUufEZwu"

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include "HTTPClient.h"
#include <OneWire.h>
  #include <DallasTemperature.h>

  // Definisi pin untuk sensor
  #define ONE_WIRE_BUS 33  // DS18B20 di pin GPIO 33
  #define PH_PIN 32        // Pin untuk sensor pH
  #define TURBIDITY_PIN 35 // Pin untuk sensor kekeruhan
  #define TDS_PIN 34       // Pin untuk sensor TDS

  // Konfigurasi Blynk
  char ssid[] = "SAD1"; //sesuaikan dengan WiFi yang digunakan
  char pass[] = "10092002";
  char auth[] = BLYNK_AUTH_TOKEN;

  // Pin yang terhubung dengan relay
  #define RELAY_PIN 23

  OneWire oneWire(ONE_WIRE_BUS);
  DallasTemperature sensors(&oneWire);

  // Global variables
  float phValue, temperatureC, turbidityValue, tdsLevel;
  float final_ph, final_tds, final_kekeruhan;

  float ph_asam, ph_normal, ph_basa;
  float tds_rendah, tds_sedang, tds_tinggi;
  float minr[10];
  String Rule[10];
  String kualitas_air_baik = "Kualitas Air Baik";
  String kualitas_air_kurang_baik = "Kualitas Air Kurang Baik";
  String kualitas_air_buruk = "Kualitas Air Buruk";

  const char* IPAddr = "tif-skripsi.my.id";

  WiFiClient NodeMCU;

  float readPH() {
  int phRaw = analogRead(PH_PIN);
  float phValue;
  if (phRaw >= 3151) {
  phValue = map(phRaw, 3151, 3808, 70, 43) / 10.0;
  } else {
  phValue = map(phRaw, 2699, 3151, 90, 70) / 10.0;
  }
  return phValue;
  }
```

```
55.   float readTemperature() {
56.   sensors.requestTemperatures();
57.   return sensors.getTempCByIndex(0);
58.   }
59.
60.   void get_kekeruhan() {
61.   float raw_kekeruhan = 0;
62.   int batas_looping_kekeruhan = 100;
63.   for (int i = 0; i < batas_looping_kekeruhan; i++) {
64.   raw_kekeruhan += analogRead(TURBIDITY_PIN);
65.   }
66.   int nilai_kekeruhan = int(raw_kekeruhan /
   batas_looping_kekeruhan);
67.   nilai_kekeruhan = max(nilai_kekeruhan, 800);
68.   nilai_kekeruhan = min(nilai_kekeruhan, 3400);
69.   nilai_kekeruhan = map(nilai_kekeruhan, 800, 3400,0, 1023);
70.   final_kekeruhan = nilai_kekeruhan;
71.   }
72.
73.   float readTDS() {
74.   int tdsValue = analogRead(TDS_PIN);
75.   float voltageTDS = tdsValue * (3.3 / 4095.0);
76.   float tds = (133.42 * voltageTDS * voltageTDS * voltageTDS
   - 255.86 * voltageTDS * voltageTDS + 857.39 * voltageTDS) *
   0.5;
77.   return tds;
78.   }
79.
80.   float func_ph_asam() {
81.   if (final_ph <= 6.5) {
82.   ph_asam = 1;
83.   } else if (final_ph > 6.0 && final_ph <= 6.5) {
84.   ph_asam = (6.5 - final_ph) / (6.5 - 6.0);
85.   } else {
86.   ph_asam = 0;
87.   }
88.   return ph_asam;
89.   }
90.
91.   float func_ph_normal() {
92.   if (final_ph <= 6.5) {
93.   ph_normal = 0;
94.   } else if (final_ph > 6.5 && final_ph <= 7.5) {
95.   ph_normal = (final_ph - 6.5) / (7.5 - 6.5);
96.   } else if (final_ph > 8.0 && final_ph <= 8.5) {
97.   ph_normal = (8.5 - final_ph) / (8.5 - 8.0);
98.   } else {
99.   ph_normal = 0;
100.  }
101.  return ph_normal;
102.  }
103.
104.  float func_ph_basa() {
105.  if (final_ph <= 8.0) {
106.  ph_basa = 0;
107.  } else if (final_ph > 8.0 && final_ph <= 8.5) {
108.  ph_basa = (final_ph - 8.0) / (8.5 - 8.0);
109.  } else {
```

```
110.  ph_basa = 1;
111.  }
112.  return ph_basa;
113.  }
114.
115.  float func_tds_rendah() {
116.  if (final_tds <= 0) {
117.  tds_rendah = 1;
118.  } else if (final_tds > 0 && final_tds <= 200) {
119.  tds_rendah = (200 - final_tds) / 200;
120.  } else {
121.  tds_rendah = 0;
122.  }
123.  return tds_rendah;
124.  }
125.
126.  float func_tds_sedang() {
127.  if (final_tds <= 200) {
128.  tds_sedang = 0;
129.  } else if (final_tds > 200 && final_tds <= 600) {
130.  tds_sedang = (final_tds - 200) / (600 - 200);
131.  } else if (final_tds > 600 && final_tds <= 1200) {
132.  tds_sedang = (1200 - final_tds) / (1200 - 600);
133.  } else {
134.  tds_sedang = 0;
135.  }
136.  return tds_sedang;
137.  }
138.
139.  float func_tds_tinggi() {
140.  if (final_tds <= 600) {
141.  tds_tinggi = 0;
142.  } else if (final_tds > 600 && final_tds <= 1200) {
143.  tds_tinggi = (final_tds - 600) / (1200 - 600);
144.  } else {
145.  tds_tinggi = 1;
146.  }
147.  return tds_tinggi;
148.  }
149.
150.  float Min(float a, float b) {
151.  return (a < b) ? a : b;
152.  }
153.
154.  String rule() {
155.  minr[1] = Min(func_ph_asam(), func_tds_tinggi());
156.  Rule[1] = kualitas_air_buruk;
157.
158.  minr[2] = Min(func_ph_asam(), func_tds_sedang());
159.  Rule[2] = kualitas_air_buruk;
160.
161.  minr[3] = Min(func_ph_asam(), func_tds_rendah());
162.  Rule[3] = kualitas_air_kurang_baik;
163.
164.  minr[4] = Min(func_ph_normal(), func_tds_tinggi());
165.  Rule[4] = kualitas_air_buruk;

166.  minr[5] = Min(func_ph_normal(), func_tds_sedang());
167.  Rule[5] = kualitas_air_kurang_baik;
```

```
168.
169.  minr[6] = Min(func_ph_normal(), func_tds_rendah());
170.  Rule[6] = kualitas_air_baik;
171.
172.  minr[7] = Min(func_ph_basa(), func_tds_tinggi());
173.  Rule[7] = kualitas_air_buruk;
174.
175.  minr[8] = Min(func_ph_basa(), func_tds_sedang());
176.  Rule[8] = kualitas_air_buruk;
177.
178.  minr[9] = Min(func_ph_basa(), func_tds_rendah());
179.  Rule[9] = kualitas_air_kurang_baik;
180.
181.  float max_val = minr[1];
182.  String final_rule = Rule[1];
183.  for (int i = 2; i <= 9; i++) {
184.  if (minr[i] > max_val) {
185.  max_val = minr[i];
186.  final_rule = Rule[i];
187.  }
188.  }
189.  return final_rule;
190.  }

191.  String urlEncode(String str) {
192.  String encodedString = "";
193.  char c;
194.  char code0;
195.  char code1;
196.  char code2;
197.  for (int i = 0; i < str.length(); i++) {
198.  c = str.charAt(i);
199.  if (c == ' ') {
200.  encodedString += '+';
201.  } else if (isalnum(c)) {
202.  encodedString += c;
203.  } else {
204.  code1 = (c & 0xf) + '0';
205.  if ((c & 0xf) > 9) {
a. code1 = (c & 0xf) - 10 + 'A';
206.  }
207.  c = (c >> 4) & 0xf;
208.  code0 = c + '0';
209.  if (c > 9) {
a. code0 = c - 10 + 'A';
210.  }
211.  code2 = '\0';
212.  encodedString += '%';
213.  encodedString += code0;
214.  encodedString += code1;
215.  }
216.  yield();
217.  }
218.  return encodedString;
219.  }
220.
221.  bool systemActive = true; // Global variable to track
   system state
222.
```

```
void setup() {
// Memulai Serial Monitor
Serial.begin(115200);

// Mengatur pin relay sebagai output
pinMode(RELAY_PIN, OUTPUT);

// Menyambungkan ke WiFi
Blynk.begin(auth, ssid, pass);
Serial.println(WiFi.localIP());

// Mematikan pompa air pada awalnya
digitalWrite(RELAY_PIN, HIGH);
Serial.println("Pompa air dalam keadaan mati.");

sensors.begin();
}

void loop() {
Blynk.run();

// Check for input from the serial monitor
if (Serial.available() > 0) {
String input = Serial.readString();
input.trim();

if (input.equalsIgnoreCase("START")) {
systemActive = true;
Serial.println("Sistem dimulai.");
} else if (input.equalsIgnoreCase("STOP")) {
systemActive = false;
Serial.println("Sistem dihentikan.");
// Ensure the pump is off when the system is stopped
digitalWrite(RELAY_PIN, HIGH);
Serial.println("Pompa air dalam keadaan mati.");
}
}

if (!systemActive) {
// Skip the rest of the loop if the system is stopped
return;
}

if(!NodeMCU.connect(IPAddr, 80)) {
Serial.println("Gagal koneksi ke webserver");
return;
}

phValue = readPH();
final_ph = phValue;
temperatureC = readTemperature();
get_kekeruhan();
turbidityValue = final_kekeruhan;
tdsLevel = readTDS();
final_tds = tdsLevel;

String finalKualitasAir = rule();

Serial.print("pH Level: ");
```

```
282.  Serial.print(phValue);
283.  Serial.print(" (");
284.  Serial.print(func_ph_asam() ? "asam" : (func_ph_normal() ?
   "normal" : "basa"));
285.  Serial.println(")");
286.
287.  Serial.print("Temperature: ");
288.  Serial.print(temperatureC);
289.  Serial.println(" °C");

290.  Serial.print("Turbidity: ");
291.  Serial.print(turbidityValue);
292.  Serial.println();
293.
294.  Serial.print("TDS Level: ");
295.  Serial.print(tdsLevel);
296.  Serial.print(" (");
297.  Serial.print(func_tds_rendah() ? "rendah" :
   (func_tds_sedang() ? "sedang" : "tinggi"));
298.  Serial.println(")");
299.
300.  Serial.print("Final Kualitas Air: ");
301.  Serial.println(finalKualitasAir);
302.
303.  // Kirim data ke Blynk
304.  Blynk.virtualWrite(V0, finalKualitasAir);
305.  Blynk.virtualWrite(V1, phValue);
306.  Blynk.virtualWrite(V2, final_tds);
307.  Blynk.virtualWrite(V3, temperatureC);
308.  Blynk.virtualWrite(V4, final_kekeruhan);
309.
310.  // Kirim data ke server
311.  String url1 = "http://" + String(IPAddr) +
   "/syifaalmahdhori.tif-
   skripsi.my.id/kirimdata_history.php?sensor_ph=" +
   String(phValue) + "&sensor_tds=" + String(tdsLevel) +
   "&sensor_kekeruhan=" + String(final_kekeruhan) +
   "&sensor_suhu=" + String(temperatureC);
312.  //error disini, String final kualitas air
313.  String url2 = "http://" + String(IPAddr) +
   "/syifaalmahdhori.tif-
   skripsi.my.id/kirimdata.php?sensor_ph=" + String(phValue) +
   "&sensor_tds=" + String(tdsLevel) + "&sensor_kekeruhan=" +
   String(final_kekeruhan) + "&sensor_suhu=" +
   String(temperatureC) + "&final_kualitas_air=" +
   urlEncode(finalKualitasAir);
314.  //String(finalKualitasAir);
315.  HTTPClient http;
316.
317.  // Send to first URL
318.  http.begin(url1);
319.  int httpCode1 = http.GET();
320.  if (httpCode1 > 0) {
321.  String payload1 = http.getString();
322.  Serial.println("Response from first URL: " + payload1);
323.  } else {
324.  Serial.print("Error on first HTTP request: ");
325.  Serial.println(httpCode1);
326.  }
```

```
http.end();

// Send to second URL
http.begin(url2);
int httpCode2 = http.GET();
if (httpCode2 > 0) {
String payload2 = http.getString();
Serial.println("Response from second URL: " + payload2);
} else {
Serial.print("Error on second HTTP request: ");
Serial.println(httpCode2);
}
http.end();

// Delay sebelum pembacaan berikutnya
delay(1000);
}

// Fungsi ini akan dipanggil ketika Virtual Pin V1 diubah
   di aplikasi Blynk
BLYNK_WRITE(V5) {
int pinValue = param.asInt(); // Mendapatkan nilai dari
aplikasi Blynk (0 atau 1)

if(pinValue == HIGH) {
// Menyalakan pompa air
digitalWrite(RELAY_PIN, LOW);
Serial.println("Pompa air menyala.");
} else {
// Mematikan pompa air
digitalWrite(RELAY_PIN, HIGH);
Serial.println("Pompa air mati.");
}
}
```