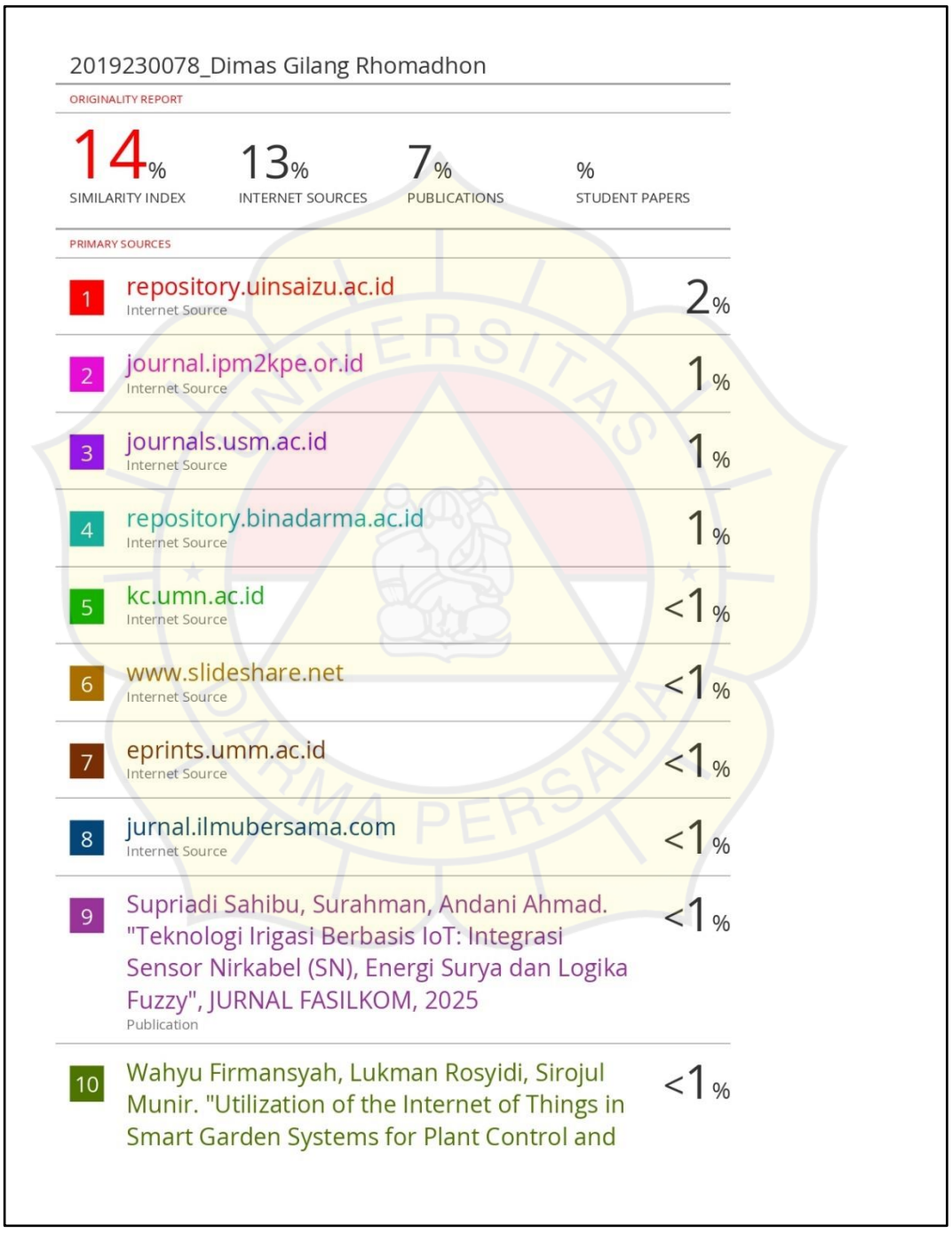
LAMPIRAN

Lampiran 1. Pengecekan Turnitin



2019230078_Dimas Gilang Rhomadhon

ORIGINALITY REPORT

14%	13%	7%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	repository.uinsaizu.ac.id Internet Source	2%
2	journal.ipm2kpe.or.id Internet Source	1%
3	journals.usm.ac.id Internet Source	1%
4	repository.binadarma.ac.id Internet Source	1%
5	kc.umn.ac.id Internet Source	<1%
6	www.slideshare.net Internet Source	<1%
7	eprints.umm.ac.id Internet Source	<1%
8	jurnal.ilmubersama.com Internet Source	<1%
9	Supriadi Sahibu, Surahman, Andani Ahmad. "Teknologi Irigasi Berbasis IoT: Integrasi Sensor Nirkabel (SN), Energi Surya dan Logika Fuzzy", JURNAL FASILKOM, 2025 Publication	<1%
10	Wahyu Firmansyah, Lukman Rosyidi, Sirojul Munir. "Utilization of the Internet of Things in Smart Garden Systems for Plant Control and	<1%

Lampiran 2. Kode Program Blynk

```
#define BLYNK_TEMPLATE_ID "TMPL6SnUsbCM5"
#define BLYNK_TEMPLATE_NAME "Microgreen"

#include <WiFi.h>
#include <EEPROM.h>
#include "GravityTDS.h"
#include <DHT.h>
#include "BlynkSimpleEsp32.h"

// WiFi & Blynk
char auth[] = "XPaBUS-eh3FuTCh3_myngs2vBIT7O5w-";
char ssid[] = "Dimas Home";
char pass[] = "DD772233";

// Pin Sensor & Relay
#define TDS_SENSOR_PIN 35
#define DHT_PIN 23
#define DHT_TYPE DHT22
#define SOIL_PIN 34

#define RELAY1_PIN 5
#define RELAY2_PIN 26
#define RELAY3_PIN 27
#define RELAY4_PIN 15

#define EEPROM_SIZE 512
```

```
// RS485 NPK Sensor (Modbus RTU)
#define RE_DE_PIN 4
#define RXD2 16
#define TXD2 17

// Flowmeter YF-S401
#define FLOW_AIR_PIN 18
#define FLOW_ABMIX_PIN 19
#define PULSES_PER_LITER 450
const float FLOW_FACTOR = 7.5; // datasheet

// Pulse counters, rates, process status
volatile unsigned long flowAirPulseCount = 0;
volatile unsigned long flowABMixPulseCount = 0;
portMUX_TYPE flowMux = portMUX_INITIALIZER_UNLOCKED;
float flowAirRate = 0.0, flowABMixRate = 0.0;
unsigned long lastFlowMillis = 0;

// Dosing state
bool airProcessActive = false;
bool abMixProcessActive = false;
unsigned long airPulseStart = 0, abMixPulseStart = 0;

// Modbus NPK sensor
const byte nitro[] = {0x01, 0x03, 0x00, 0x04, 0x00, 0x01, 0x60,
0x0B};
const byte phos[]  = {0x01, 0x03, 0x00, 0x05, 0x00, 0x01, 0xB1,
0xCB};
```

```
const byte pota[]  = {0x01, 0x03, 0x00, 0x06, 0x00, 0x01, 0x61,
0xCA};


// Sensor objects

DHT dht(DHT_PIN, DHT_TYPE);

GravityTDS gravityTds;

BlynkTimer timer;


// Sensor variables

float tdsValue = 0, temperature = 25, humidity = 0;

int soilValue = 0, soilPercent = 0;

uint16_t valN = 0, valP = 0, valK = 0;


// Success counters

unsigned long countDHT_ok = 0, countTDS_ok = 0, countSoil_ok = 0,
countNPK_ok = 0;


// Interrupt handlers

void IRAM_ATTR countFlowAir() { portENTER_CRITICAL_ISR(&flowMux);
if (airProcessActive) flowAirPulseCount++;
portEXIT_CRITICAL_ISR(&flowMux);}

void IRAM_ATTR countFlowABMix() {
portENTER_CRITICAL_ISR(&flowMux); if (abMixProcessActive)
flowABMixPulseCount++; portEXIT_CRITICAL_ISR(&flowMux);}


// Read NPK sensor via RS485

uint16_t readNPK(const byte* frame, const char* label) {

  byte response[16]; byte index = 0;

  digitalWrite(RE_DE_PIN, HIGH); delay(2);

  Serial2.write(frame, 8); Serial2.flush(); delay(2);

  digitalWrite(RE_DE_PIN, LOW);
```

```
  unsigned long start = millis();
  while (Serial2.available() < 7 && millis() - start < 500);
  while (Serial2.available() && index < 7) { response[index++] =
Serial2.read(); }
  Serial.print(label); Serial.print(" HEX: ");
  for (byte i = 0; i < index; i++) Serial.printf("%02X ",
response[i]);
  Serial.println();
  if (index == 7) return (response[3] << 8) | response[4];
  Serial.print(label); Serial.println(" Gagal baca."); return 0;
}

// Setup
void setup() {
  Serial.begin(115200); EEPROM.begin(EEPROM_SIZE);
  pinMode(RELAY1_PIN, OUTPUT); pinMode(RELAY2_PIN, OUTPUT);
  pinMode(RELAY3_PIN, OUTPUT); pinMode(RELAY4_PIN, OUTPUT);
  digitalWrite(RELAY1_PIN, HIGH); digitalWrite(RELAY2_PIN, HIGH);
  digitalWrite(RELAY3_PIN, HIGH); digitalWrite(RELAY4_PIN, HIGH);
  dht.begin();
  gravityTds.setPin(TDS_SENSOR_PIN); gravityTds.setAref(3.3);
gravityTds.setAdcRange(4096); gravityTds.begin();
  WiFi.begin(ssid, pass); Blynk.begin(auth, ssid, pass);
  pinMode(RE_DE_PIN, OUTPUT); digitalWrite(RE_DE_PIN, LOW);
  Serial2.begin(4800, SERIAL_8N1, RXD2, TXD2);
  pinMode(FLOW_AIR_PIN, INPUT_PULLUP); pinMode(FLOW_ABMIX_PIN,
INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(FLOW_AIR_PIN),
countFlowAir, RISING);
  attachInterrupt(digitalPinToInterrupt(FLOW_ABMIX_PIN),
countFlowABMix, RISING);
```

```
  timer.setInterval(2000L, sendAllSensorData);
}

// Send all sensor data to Blynk
void sendAllSensorData() {
  temperature = dht.readTemperature();
  humidity = dht.readHumidity();
  if (!isnan(temperature) && !isnan(humidity)) countDHT_ok++;
  if (isnan(temperature)) temperature = 25.0;

  gravityTds.setTemperature(temperature);
  gravityTds.update();
  tdsValue = gravityTds.getTdsValue();
  if (tdsValue > 0 && tdsValue < 3000) countTDS_ok++;

  soilValue = analogRead(SOIL_PIN);
  soilPercent = map(soilValue, 4095, 0, 0, 100);
  soilPercent = constrain(soilPercent, 0, 100);
  if (soilPercent >= 0 && soilPercent <= 100) countSoil_ok++;

  valN = readNPK(nitro, "Nitrogen"); delay(200);
  valP = readNPK(phos, "Phosphorous"); delay(200);
  valK = readNPK(pota, "Potassium"); delay(200);
  if (valN > 0 && valP > 0 && valK > 0) countNPK_ok++;

  unsigned long now = millis();
  if (now - lastFlowMillis >= 1000) {
    unsigned long airCount, abCount;
```

```
    portENTER_CRITICAL(&flowMux);
    airCount = flowAirPulseCount;
    abCount = flowABMixPulseCount;
    flowAirPulseCount = 0;
    flowABMixPulseCount = 0;
    portEXIT_CRITICAL(&flowMux);

    float airHz = airCount / 1.0, abHz = abCount / 1.0;
    flowAirRate = airHz / FLOW_FACTOR;
    flowABMixRate = abHz / FLOW_FACTOR;
    lastFlowMillis = now;

    Serial.printf("Flow Air: %.2f L/min, Flow AB Mix: %.2f
L/min\n", flowAirRate, flowABMixRate);
    Blynk.virtualWrite(V11, flowAirRate);
    Blynk.virtualWrite(V12, flowABMixRate);
  }

  // 100ml dosing, with interrupt safety
  if (airProcessActive) {
    unsigned long pulses;
    portENTER_CRITICAL(&flowMux);
    pulses = flowAirPulseCount - airPulseStart;
    portEXIT_CRITICAL(&flowMux);
    float airMl = (float)pulses / PULSES_PER_LITER * 1000.0;
    if (airMl >= 100.0) {
      digitalWrite(RELAY3_PIN, HIGH);
      airProcessActive = false;
```

```
      Blynk.virtualWrite(V20, 0);
    }
  }
  if (abMixProcessActive) {
    unsigned long pulses;
    portENTER_CRITICAL(&flowMux);
    pulses = flowABMixPulseCount - abMixPulseStart;
    portEXIT_CRITICAL(&flowMux);
    float abMl = (float)pulses / PULSES_PER_LITER * 1000.0;
    if (abMl >= 100.0) {
      digitalWrite(RELAY4_PIN, HIGH);
      abMixProcessActive = false;
      Blynk.virtualWrite(V21, 0);
    }
  }


  Serial.printf("Suhu: %.1f°C | Kelembapan: %.1f%% | TDS: %.0f ppm
| Soil: %d%% | N: %d | P: %d | K: %d\n",
    temperature, humidity, tdsValue, soilPercent, valN, valP,
valK);
  Serial.printf("Pembacaan BERHASIL: DHT=%lu, TDS=%lu, Soil=%lu,
NPK=%lu\n",
    countDHT_ok, countTDS_ok, countSoil_ok, countNPK_ok);


  Blynk.virtualWrite(V25, countDHT_ok);
  Blynk.virtualWrite(V26, countTDS_ok);
  Blynk.virtualWrite(V27, countSoil_ok);
  Blynk.virtualWrite(V28, countNPK_ok);
  Blynk.virtualWrite(V4, temperature);
```

```
  Blynk.virtualWrite(V5, humidity);
  Blynk.virtualWrite(V6, tdsValue);
  Blynk.virtualWrite(V7, soilPercent);
  Blynk.virtualWrite(V8, valN);
  Blynk.virtualWrite(V9, valP);
  Blynk.virtualWrite(V10, valK);
}

// Relay control via Blynk
BLYNK_WRITE(V0) { digitalWrite(RELAY1_PIN, !param.asInt()); }
BLYNK_WRITE(V1) { digitalWrite(RELAY2_PIN, !param.asInt()); }
BLYNK_WRITE(V2) { digitalWrite(RELAY3_PIN, !param.asInt()); }
BLYNK_WRITE(V3) { digitalWrite(RELAY4_PIN, !param.asInt()); }

// Button V20: Dosing air 100ml
BLYNK_WRITE(V20) {
  if (param.asInt() == 1 && !airProcessActive) {
    airProcessActive = true;
    portENTER_CRITICAL(&flowMux);
    airPulseStart = flowAirPulseCount;
    portEXIT_CRITICAL(&flowMux);
    digitalWrite(RELAY3_PIN, LOW);
  }
}
// Button V21: Dosing AB mix 100ml
BLYNK_WRITE(V21) {
  if (param.asInt() == 1 && !abMixProcessActive) {
    abMixProcessActive = true;
```

```
    portENTER_CRITICAL(&flowMux);
    abMixPulseStart = flowABMixPulseCount;
    portEXIT_CRITICAL(&flowMux);
    digitalWrite(RELAY4_PIN, LOW);
  }
}

void loop() {
  Blynk.run();
  timer.run();
    }
```