LAMPIRAN

UNIVERSITAS DARMA PERSADA
UPT PERPUSTAKAAN
Gedung Rektorat Lantai 3,
Jl.Taman Malaka Selatan, Pondok Kelapa – Jakarta Timur 13450

SURAT KETERANGAN
HASIL PENGECEKAN TURNITIN

UPT Perpustakaan Universitas Darma Persada menerangkan telah selesai melakukan pemeriksaan duplikasi/*similarity* menggunakan perangkat lunak Turnitin terhadap hasil karya sebagai berikut:

Judul : Rancang Bangun Sistem Monitoring Kualitas Air Pada Akuarium Ikan Mas Koki Berbasis Internet of Things Menggunakan Metode Fuzzy

Penulis : Muhammad Nihti Fauzi

NIM : 2018230077

Tgl pemeriksaan : 29 Juli 2025

Dengan hasil Tingkat Kesamaan (*similarity index*) 21%

Demikian Surat Keterangan kami buat, untuk dipergunakan sebagaimana mestinya.

Jakarta, 29 Juli 2025

Ka.UPT Perpustakaan Unsada

Yus Rusmiyati, SS., MM

Batas maksimal similarity 30% untuk Fakultas Sastra dan Ekonomi
Batas maksimal similarity 25% untuk Fakultas Teknik, Kelautan dan Pasca Sarjana

Lampiran 1. Surat Keterangan Hasil Turnitin

2018230077_Muhammad Nihti Fauzi_Rancang Bangun Sistem Monitoring Kualitas Air Pada Akuarium Ikan...

ORIGINALITY REPORT

21%	20%	8%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	conference.upnvj.ac.id Internet Source	2%
2	etheses.uin-malang.ac.id Internet Source	1%
3	ejurnal.seminar-id.com Internet Source	1%
4	Andres Rachman Duta. "RANCANG BANGUN SISTEM PEMANTAUAN DAN PENGENDALIAN KUALITAS AIR PADA AKUARIUM IKAN HIAS MAS KOKI MENGGUNAKAN METODE FUZZY LOGIC BERBASIS IoT (Internet of Things)", Jurnal Informatika dan Teknik Elektro Terapan, 2025 Publication	1%
5	repository.its.ac.id Internet Source	1%
6	docplayer.info Internet Source	<1%
7	ejournal.poltekharber.ac.id Internet Source	<1%
8	www.scribd.com Internet Source	<1%
9	tunasbangsa.ac.id Internet Source	<1%

Lampiran 2.1 Hasil Pengecekan Turnitin

- Lampiran 3. File Coding Arduino Mega

```
#define FIS_TYPE float
#define FIS_RESOLUSION 101
#define FIS_MIN -3.4028235E+38
#define FIS_MAX 3.4028235E+38
typedef FIS_TYPE(*_FIS_MF)(FIS_TYPE, FIS_TYPE*);
typedef FIS_TYPE(*_FIS_ARR_OP)(FIS_TYPE, FIS_TYPE);
typedef FIS_TYPE(*_FIS_ARR)(FIS_TYPE*, int, _FIS_ARR_OP);

#include <Wire.h>
#include<SoftwareSerial.h>

#include <LiquidCrystal_PCF8574.h>
//Inisialisasi LCD I2C (alamat umum 0x27, bisa 0x3F tergantung modul)
LiquidCrystal_PCF8574 lcd(0x27);

//relay
const int relay = 3;
//end relay

//Inisialisasi Pin Buzzer
const int buzzerPin = 4;

String kirimdata ;
String Status;
String pesan;

//Sensor PH
// Tentukan pin analog untuk sensor pH
const int ph_Pin  = A0;   // GPIO34 contoh, bisa ganti ke pin analog lain
float Po = 0;
float PH_step;
int nilai_analog_PH;
double TeganganPh;

//UNTUK KALIBRASI
float PHT = 3.35;
float PHE = 3.68;

//End Sensor PH

//Sensor TDS
int TdsSensorPin = A2;
float calibrationFactor = 1.0;
//End Sensor TDS

//Sensor Turbidity
int TurbiditySensorPin = A3;
int TurbiditySensorValue = 0;
```

```
//End Sensor Turbidity

//#include "fis_header.h"

// Number of inputs to the fuzzy inference system
const int fis_gcI = 3;
// Number of outputs to the fuzzy inference system
const int fis_gcO = 1;
// Number of rules to the fuzzy inference system
const int fis_gcR = 27;

FIS_TYPE g_fisInput[fis_gcI];
FIS_TYPE g_fisOutput[fis_gcO];

// Setup routine runs once when you press reset:

void setup() {
  Wire.begin();
  Serial.begin(115200);
  Serial1.begin(115200);
  pinMode(relay, OUTPUT);
  pinMode(buzzerPin, OUTPUT);
  pinMode (ph_Pin, INPUT);
  pinMode(TdsSensorPin,INPUT);
  pinMode(TurbiditySensorPin,INPUT);
  Serial.begin(115200);   // Baudrate umum ESP32
  lcd.begin(16, 2); // 16 baris, 2 kolom
  lcd.setBacklight(255); //menyalakan lampu latar
  lcd.clear();
}

void loop() {

 //Sensor TDS
int TdsSensorValue = analogRead(TdsSensorPin);
  float voltage = TdsSensorValue * (5.0 / 1023.0);
  float tdsValue = (133.42 * voltage * voltage * voltage - 255.86 * voltage * voltage + 857.39
* voltage) * calibrationFactor;
  //End Sensor TDS

  //Sensor PH
  int nilai_analog_PH = analogRead(ph_Pin);
  TeganganPh = 5 / 1024.0 * nilai_analog_PH;
  PH_step = (PHE - PHT) / 3;
  Po = 7.00 + ((PHT - TeganganPh) / PH_step);
  //End Sensor PH

  //Sensor Turbidity
```

```
  TurbiditySensorValue = analogRead(TurbiditySensorPin);
  float ntu = -0.5 * TurbiditySensorValue + 425;
  //End Sensor Turbidity

//  int ph = random(0,14);
//  int tds = random(0,1000);
//  int turbidity = random(0,400);


    // Read Input: Sensor_PH
    g_fisInput[0] = Po;
    // Read Input: Sensor_TDS
    g_fisInput[1] = tdsValue;
    // Read Input: Sensor_Turbidity
    g_fisInput[2] = ntu;

    g_fisOutput[0] = 0;

    fis_evaluate();

    // Set output vlaue: kualitas
    analogWrite(3 , g_fisOutput[0]);


  // Serial.print("Nilai ADC Ph: ");
  // Serial.println(nilai_analog_PH);
  // Serial.print("TeganganPh: ");
  // Serial.println(TeganganPh, 3);
  Serial.print("Nilai PH Cairan: ");
  Serial.println(Po, 2);

  Serial.print("TDS Value:");
  Serial.print(tdsValue,0);
  Serial.println("ppm");

  // Serial.print("ADC: ");
  // Serial.print(TurbiditySensorValue);
  Serial.print("NTU: ");
  Serial.println(ntu);

//  Serial.println(ph);
//  Serial.println(tds);
//  Serial.println(turbidity);
  Serial.print("Hasil ");
  Serial.println(g_fisOutput[0]);

   if (g_fisOutput[0] < 25)
  {
   Status = "Buruk";
```

```
  pesan = "Kualitas_Air_Akuarium_Buruk";
  Serial.print("Status : ");
  Serial.println(Status);
  Serial.print("Pesan : ");
  Serial.print(pesan);
  Serial.println("");
  Serial.println("");
  digitalWrite(buzzerPin, HIGH);
  digitalWrite(relay, HIGH);
}
else if(g_fisOutput[0] > 25 && g_fisOutput[0] < 74)
{
  Status = "Cukup";
  pesan = "-";
  Serial.print("Status : ");
  Serial.println(Status);
  Serial.print("Pesan : ");
  Serial.print(pesan);
  Serial.println("");
  Serial.println("");
  digitalWrite(buzzerPin, HIGH);
  delay(100);
  digitalWrite(buzzerPin, LOW);
  delay(100);
  digitalWrite(buzzerPin, HIGH);
  delay(100);
  digitalWrite(buzzerPin, LOW);
  delay(3000);
}
 else if(g_fisOutput[0] > 74)
{
  Status = "Baik";
  pesan = "-";
  Serial.print("Status : ");
  Serial.println(Status);
  Serial.print("Pesan : ");
  Serial.print(pesan);
  Serial.println("");
  Serial.println("");
  digitalWrite(buzzerPin, HIGH);
  delay(100);
  digitalWrite(buzzerPin, LOW);
  delay(3000);
}
else
{
  delay(3000);
}
```

```
  String kirimdata =
('#')+String(Po)+(',')+String(tdsValue)+(',')+String(ntu)+(',')+String(g_fisOutput[0])+(',')+Stri
ng(Status)+(',')+String(pesan);
  Serial1.println(kirimdata);

  delay(3000); // Delay untuk kestabilan pembacaan

  lcd.setCursor(0, 0);            // untuk menampilkan karakter dari kolom 0 dan baris 0
  lcd.print("Sensor PH: ");       // untuk menampilkan tulisan ke lcd
  lcd.setCursor(0, 1);            // untuk menampilkan karakter dari kolom 0 dan baris 1
  lcd.print(Po);
  lcd.print(" PH");
  delay (1200);                   // delay saat tulisan berjalan
  lcd.clear ();

  lcd.setCursor(0, 0);            // untuk menampilkan karakter dari kolom 0 dan baris 0
  lcd.print("Sensor TDS: ");      // untuk menampilkan tulisan ke lcd
  lcd.setCursor(0, 1);            // untuk menampilkan karakter dari kolom 0 dan baris 1
  lcd.print(tdsValue);
  lcd.print(" PPM");
  delay (1200);                   // delay saat tulisan berjalan
  lcd.clear ();

  lcd.setCursor(0, 0);            // untuk menampilkan karakter dari kolom 0 dan baris 0
  lcd.print("Sensor Turbidity: "); // untuk menampilkan tulisan ke lcd
  lcd.setCursor(0, 1);            // untuk menampilkan karakter dari kolom 0 dan baris 1
  lcd.print(ntu);
  lcd.print(" NTU");
  delay (1200);                   // delay saat tulisan berjalan
  lcd.clear ();

  lcd.setCursor(0, 0);            // untuk menampilkan karakter dari kolom 0 dan baris 0
  lcd.print("Status: ");          // untuk menampilkan tulisan ke lcd
  lcd.setCursor(0, 1);            // untuk menampilkan karakter dari kolom 0 dan baris 1
  lcd.print(g_fisOutput[0]);
  lcd.print(" %");
  lcd.print(Status);
  delay (1200);                   // delay saat tulisan berjalan
  lcd.clear ();
}


//***************************************************************************
// Support functions for Fuzzy Inference System
//***************************************************************************
// Trapezoidal Member Function
FIS_TYPE fis_trapmf(FIS_TYPE x, FIS_TYPE* p)
{
```

```
    FIS_TYPE a = p[0], b = p[1], c = p[2], d = p[3];
    FIS_TYPE t1 = ((x <= c) ? 1 : ((d < x) ? 0 : ((c != d) ? ((d - x) / (d - c)) : 0)));
    FIS_TYPE t2 = ((b <= x) ? 1 : ((x < a) ? 0 : ((a != b) ? ((x - a) / (b - a)) : 0)));
    return (FIS_TYPE) min(t1, t2);
}

FIS_TYPE fis_prod(FIS_TYPE a, FIS_TYPE b)
{
    return (a * b);
}

FIS_TYPE fis_probor(FIS_TYPE a, FIS_TYPE b)
{
    return (a + b - (a * b));
}

FIS_TYPE fis_sum(FIS_TYPE a, FIS_TYPE b)
{
    return (a + b);
}

FIS_TYPE fis_array_operation(FIS_TYPE *array, int size, _FIS_ARR_OP pfnOp)
{
    int i;
    FIS_TYPE ret = 0;

    if (size == 0) return ret;
    if (size == 1) return array[0];

    ret = array[0];
    for (i = 1; i < size; i++)
    {
        ret = (*pfnOp)(ret, array[i]);
    }

    return ret;
}


//************************************************************************
// Data for Fuzzy Inference System
//************************************************************************
// Pointers to the implementations of member functions
_FIS_MF fis_gMF[] =
{
    fis_trapmf
};

// Count of member function for each Input
```

```
int fis_gIMFCount[] = { 3, 3, 3 };

// Count of member function for each Output
int fis_gOMFCount[] = { 3 };

// Coefficients for the Input Member Functions
FIS_TYPE fis_gMFI0Coeff1[] = { 0, 0, 4, 6 };
FIS_TYPE fis_gMFI0Coeff2[] = { 5, 6.5, 7.5, 9 };
FIS_TYPE fis_gMFI0Coeff3[] = { 8, 10, 14, 14 };
FIS_TYPE* fis_gMFI0Coeff[] = { fis_gMFI0Coeff1, fis_gMFI0Coeff2, fis_gMFI0Coeff3 };
FIS_TYPE fis_gMFI1Coeff1[] = { 0, 0, 150, 300 };
FIS_TYPE fis_gMFI1Coeff2[] = { 200, 350, 650, 800 };
FIS_TYPE fis_gMFI1Coeff3[] = { 700, 850, 1000, 1000 };
FIS_TYPE* fis_gMFI1Coeff[] = { fis_gMFI1Coeff1, fis_gMFI1Coeff2, fis_gMFI1Coeff3 };
FIS_TYPE fis_gMFI2Coeff1[] = { 0, 0, 30, 50 };
FIS_TYPE fis_gMFI2Coeff2[] = { 30, 50, 150, 200 };
FIS_TYPE fis_gMFI2Coeff3[] = { 150, 200, 400, 400 };
FIS_TYPE* fis_gMFI2Coeff[] = { fis_gMFI2Coeff1, fis_gMFI2Coeff2, fis_gMFI2Coeff3 };
FIS_TYPE** fis_gMFICoeff[] = { fis_gMFI0Coeff, fis_gMFI1Coeff, fis_gMFI2Coeff };

// Coefficients for the Output Member Functions
FIS_TYPE fis_gMFO0Coeff1[] = { 0, 0, 0, 0 };
FIS_TYPE fis_gMFO0Coeff2[] = { 0, 0, 0, 50 };
FIS_TYPE fis_gMFO0Coeff3[] = { 0, 0, 0, 100 };
FIS_TYPE* fis_gMFO0Coeff[] = { fis_gMFO0Coeff1, fis_gMFO0Coeff2,
fis_gMFO0Coeff3 };
FIS_TYPE** fis_gMFOCoeff[] = { fis_gMFO0Coeff };

// Input membership function set
int fis_gMFI0[] = { 0, 0, 0 };
int fis_gMFI1[] = { 0, 0, 0 };
int fis_gMFI2[] = { 0, 0, 0 };
int* fis_gMFI[] = { fis_gMFI0, fis_gMFI1, fis_gMFI2};

// Output membership function set

int* fis_gMFO[] = {};

// Rule Weights
FIS_TYPE fis_gRWeight[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1 };

// Rule Type
int fis_gRType[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };

// Rule Inputs
int fis_gRI0[] = { 1, 3, 3 };
int fis_gRI1[] = { 1, 3, 2 };
```

```
int fis_gRI2[] = { 1, 3, 1 };
int fis_gRI3[] = { 1, 2, 3 };
int fis_gRI4[] = { 1, 2, 2 };
int fis_gRI5[] = { 1, 2, 1 };
int fis_gRI6[] = { 1, 1, 3 };
int fis_gRI7[] = { 1, 1, 2 };
int fis_gRI8[] = { 1, 1, 1 };
int fis_gRI9[] = { 1, 3, 3 };
int fis_gRI10[] = { 2, 3, 2 };
int fis_gRI11[] = { 2, 3, 1 };
int fis_gRI12[] = { 2, 2, 3 };
int fis_gRI13[] = { 2, 2, 2 };
int fis_gRI14[] = { 2, 2, 1 };
int fis_gRI15[] = { 2, 1, 3 };
int fis_gRI16[] = { 2, 1, 2 };
int fis_gRI17[] = { 2, 1, 1 };
int fis_gRI18[] = { 3, 3, 3 };
int fis_gRI19[] = { 3, 3, 2 };
int fis_gRI20[] = { 3, 3, 1 };
int fis_gRI21[] = { 3, 2, 3 };
int fis_gRI22[] = { 3, 2, 2 };
int fis_gRI23[] = { 3, 2, 1 };
int fis_gRI24[] = { 3, 1, 3 };
int fis_gRI25[] = { 3, 1, 2 };
int fis_gRI26[] = { 3, 1, 1 };
int* fis_gRI[] = { fis_gRI0, fis_gRI1, fis_gRI2, fis_gRI3, fis_gRI4, fis_gRI5, fis_gRI6,
fis_gRI7, fis_gRI8, fis_gRI9, fis_gRI10, fis_gRI11, fis_gRI12, fis_gRI13, fis_gRI14,
fis_gRI15, fis_gRI16, fis_gRI17, fis_gRI18, fis_gRI19, fis_gRI20, fis_gRI21, fis_gRI22,
fis_gRI23, fis_gRI24, fis_gRI25, fis_gRI26 };

// Rule Outputs
int fis_gRO0[] = { 1 };
int fis_gRO1[] = { 1 };
int fis_gRO2[] = { 1 };
int fis_gRO3[] = { 1 };
int fis_gRO4[] = { 1 };
int fis_gRO5[] = { 2 };
int fis_gRO6[] = { 1 };
int fis_gRO7[] = { 2 };
int fis_gRO8[] = { 2 };
int fis_gRO9[] = { 1 };
int fis_gRO10[] = { 1 };
int fis_gRO11[] = { 2 };
int fis_gRO12[] = { 1 };
int fis_gRO13[] = { 2 };
int fis_gRO14[] = { 3 };
int fis_gRO15[] = { 2 };
int fis_gRO16[] = { 3 };
```

```
int fis_gRO17[] = { 3 };
int fis_gRO18[] = { 1 };
int fis_gRO19[] = { 1 };
int fis_gRO20[] = { 2 };
int fis_gRO21[] = { 1 };
int fis_gRO22[] = { 2 };
int fis_gRO23[] = { 3 };
int fis_gRO24[] = { 2 };
int fis_gRO25[] = { 3 };
int fis_gRO26[] = { 3 };
int* fis_gRO[] = { fis_gRO0, fis_gRO1, fis_gRO2, fis_gRO3, fis_gRO4, fis_gRO5,
fis_gRO6, fis_gRO7, fis_gRO8, fis_gRO9, fis_gRO10, fis_gRO11, fis_gRO12, fis_gRO13,
fis_gRO14, fis_gRO15, fis_gRO16, fis_gRO17, fis_gRO18, fis_gRO19, fis_gRO20,
fis_gRO21, fis_gRO22, fis_gRO23, fis_gRO24, fis_gRO25, fis_gRO26 };

// Input range Min
FIS_TYPE fis_gIMin[] = { 0, 0, 0 };

// Input range Max
FIS_TYPE fis_gIMax[] = { 14, 1000, 400 };

// Output range Min
FIS_TYPE fis_gOMin[] = { 0 };

// Output range Max
FIS_TYPE fis_gOMax[] = { 100 };

//************************************************************************
// Data dependent support functions for Fuzzy Inference System
//************************************************************************
// None for Sugeno

//************************************************************************
// Fuzzy Inference System
//************************************************************************
void fis_evaluate()
{
  FIS_TYPE fuzzyInput0[] = { 0, 0, 0 };
  FIS_TYPE fuzzyInput1[] = { 0, 0, 0 };
  FIS_TYPE fuzzyInput2[] = { 0, 0, 0 };
  FIS_TYPE* fuzzyInput[fis_gcI] = { fuzzyInput0, fuzzyInput1, fuzzyInput2, };
  FIS_TYPE fuzzyOutput0[] = { 0, 0, 0 };
  FIS_TYPE* fuzzyOutput[fis_gcO] = { fuzzyOutput0, };
  FIS_TYPE fuzzyRules[fis_gcR] = { 0 };
  FIS_TYPE fuzzyFires[fis_gcR] = { 0 };
  FIS_TYPE* fuzzyRuleSet[] = { fuzzyRules, fuzzyFires };
  FIS_TYPE sW = 0;
```

```
// Transforming input to fuzzy Input
int i, j, r, o;
for (i = 0; i < fis_gcI; ++i)
{
   for (j = 0; j < fis_gIMFCount[i]; ++j)
   {
      fuzzyInput[i][j] =
         (fis_gMF[fis_gMFI[i][j]])(g_fisInput[i], fis_gMFICoeff[i][j]);
   }
}

int index = 0;
for (r = 0; r < fis_gcR; ++r)
{
   if (fis_gRType[r] == 1)
   {
      fuzzyFires[r] = 1;
      for (i = 0; i < fis_gcI; ++i)
      {
         index = fis_gRI[r][i];
         if (index > 0)
            fuzzyFires[r] = fis_prod(fuzzyFires[r], fuzzyInput[i][index - 1]);
         else if (index < 0)
            fuzzyFires[r] = fis_prod(fuzzyFires[r], 1 - fuzzyInput[i][-index - 1]);
         else
            fuzzyFires[r] = fis_prod(fuzzyFires[r], 1);
      }
   }
   else
   {
      fuzzyFires[r] = 0;
      for (i = 0; i < fis_gcI; ++i)
      {
         index = fis_gRI[r][i];
         if (index > 0)
            fuzzyFires[r] = fis_probor(fuzzyFires[r], fuzzyInput[i][index - 1]);
         else if (index < 0)
            fuzzyFires[r] = fis_probor(fuzzyFires[r], 1 - fuzzyInput[i][-index - 1]);
         else
            fuzzyFires[r] = fis_probor(fuzzyFires[r], 0);
      }
   }

   fuzzyFires[r] = fis_gRWeight[r] * fuzzyFires[r];
   sW += fuzzyFires[r];
}

if (sW == 0)
```

```
{
    for (o = 0; o < fis_gcO; ++o)
    {
        g_fisOutput[o] = ((fis_gOMax[o] + fis_gOMin[o]) / 2);
    }
}
else
{
    for (o = 0; o < fis_gcO; ++o)
    {
        FIS_TYPE sWI = 0.0;
        for (j = 0; j < fis_gOMFCount[o]; ++j)
        {
            fuzzyOutput[o][j] = fis_gMFOCoeff[o][j][fis_gcI];
            for (i = 0; i < fis_gcI; ++i)
            {
                fuzzyOutput[o][j] += g_fisInput[i] * fis_gMFOCoeff[o][j][i];
            }
        }

        for (r = 0; r < fis_gcR; ++r)
        {
            index = fis_gRO[r][o] - 1;
            sWI += fuzzyFires[r] * fuzzyOutput[o][index];
        }

        g_fisOutput[o] = sWI / sW;
    }
  }
}
```