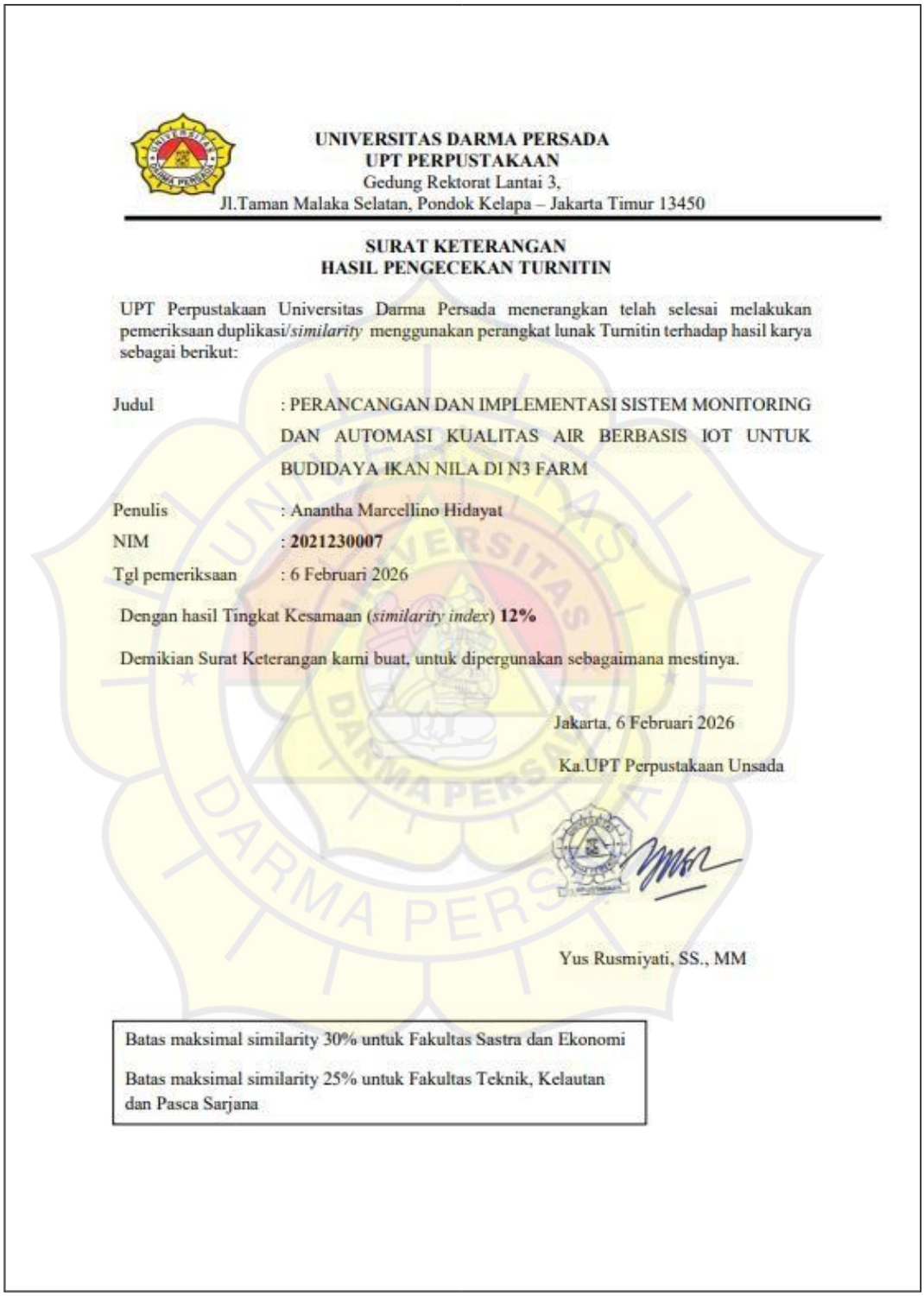
Lampiran 1. Hasil Turnitin



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 : PERANCANGAN DAN IMPLEMENTASI SISTEM MONITORING DAN AUTOMASI KUALITAS AIR BERBASIS IOT UNTUK BUDIDAYA IKAN NILA DI N3 FARM

Penulis : Anantha Marcellino Hidayat
NIM : **2021230007**
Tgl pemeriksaan : 6 Februari 2026

Dengan hasil Tingkat Kesamaan (*similarity index*) **12%**

Demikian Surat Keterangan kami buat, untuk dipergunakan sebagaimana mestinya.

Jakarta, 6 Februari 2026

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 2. Source Code Arduino

```
// ========== LIBRARY ==========
#include <Wire.h>
```

```
#include <RTClib.h>
#include <ESP32Servo.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <WiFi.h>
#include <FirebaseESP32.h>
#include "config.h"

// ========== KONFIGURASI PIN ==========
#define SDA_PIN 21
#define SCL_PIN 22
#define ONE_WIRE_BUS 4
#define TURBIDITY_PIN 35
#define PH_PIN 34
#define SERVO_PIN 18
#define RELAY1_PIN 25
#define RELAY2_PIN 26
#define LED_WIFI_PIN 2
#define LED_STATUS_PIN 15
// ========== PIN SENSOR HC-SR04 STOK PAKAN ==========
#define TRIG_PIN 13
#define ECHO_PIN 12
// ========== PIN SENSOR HC-SR04 STOK CAIRAN ==========
#define TRIG_ASAM_PIN 14
#define ECHO_ASAM_PIN 27
#define TRIG_BASA_PIN 33
#define ECHO_BASA_PIN 32

// ========== KONFIGURASI TANGKI PAKAN ==========
const int TOPLES_TINGGI = 18; const int
JARAK_KOSONG = 15; const int JARAK_PENUH = 5;
// ========== KONFIGURASI TANGKI CAIRAN ========== const int
TANGKI_TINGGI_ASAM = 16;   // Tinggi total tangki asam
(cm) const int JARAK_ASAM_PENUH = 4;      // Jarak sensor ke
cairan ketika PENUH (cm)
const int JARAK_ASAM_KOSONG = 11;    // Jarak sensor ke cairan
ketika KOSONG (cm)
```

```
const
      int TANGKI_TINGGI_BASA = 16;   // Tinggi total tangki basa
(cm) const int JARAK_BASA_PENUH = 4;      // Jarak sensor ke
cairan ketika PENUH (cm) const int JARAK_BASA_KOSONG = 11;
// Jarak sensor ke cairan ketika KOSONG (cm)
 const float SUHU_MIN = 25.0; const
float SUHU_MAX = 30.0; const float
SUHU_KRITIS_MIN = 20.0; const float
SUHU_KRITIS_MAX = 35.0;


// ═══════════════════════════════════════════════════════ //
KONFIGURASI TURBIDITY - HASIL UJI LAB PAM JAYA
// ═══════════════════════════════════════════════════════
const float CALIB_SLOPE = -303.79;          // Slope dari regresi
linear const float CALIB_INTERCEPT = 1000.24;       // Intercept
dari regresi linear const float CALIB_R_SQUARED = 0.9998;
// Akurasi kalibrasi


// Data kalibrasi untuk referensi const float CALIB_POINT_1_V =
3.300;         // Air Jernih const float CALIB_POINT_1_NTU =
4.40; const float CALIB_POINT_2_V = 2.948;         // Air Sedikit
Keruh const float CALIB_POINT_2_NTU = 95.90; const float
CALIB_POINT_3_V = 1.440;         // Air Keruh const float
CALIB_POINT_3_NTU = 564.00;


// Threshold untuk budidaya ikan nila
const float NTU_OPTIMAL = 25.0;              // < 25 NTU = Sangat
Baik const float NTU_GOOD = 50.0;              // 25-50 NTU =
Baik const float NTU_WARNING = 100.0;          // 50-100 NTU =
Warning
// > 100 NTU = Bahaya


// Sampling configuration const int TURBIDITY_SAMPLES = 20;
// Jumlah sampel untuk averaging
const int TURBIDITY_SAMPLE_DELAY = 50;       // Delay antar sampel
(ms)  const float PH_MIN
= 6.5; const float PH_MAX
= 8.0; float PH_IDEAL_MIN
= 7.0; const float
PH_IDEAL_MAX = 8.0;


// ========== KONFIGURASI pH CONTROL ========== const float
PH_TRIGGER_ASAM = 7.0; const float PH_TRIGGER_BASA = 8.0; const
unsigned long RELAY_ASAM_DURATION = 4000; const unsigned long
RELAY_BASA_DURATION = 1200; const unsigned long RELAY_COOLDOWN =
180000; // cooldown 3 menit const float ML_PER_TRIGGER_ASAM =
50.0; const float ML_PER_TRIGGER_BASA = 1.0;
```

```
const
 float adc_pH4  =
2558.71; float adc_pH6  =
1953.43; float adc_pH9  =
1452.71;

// ========== KONFIGURASI SERVO & PAKAN ==========
const int SERVO_TUTUP = 90; const int SERVO_BUKA =
0;
 const float WEIGHT_250MS = 35.4;
const float WEIGHT_500MS = 63.0;
const float WEIGHT_750MS = 95.6;
const float WEIGHT_1000MS = 121.8;
 struct FeedingSchedule
{   int hour;   int
minute;   float
weightGrams;   int
durationMs;   bool
enabled;
};

FeedingSchedule JADWAL_PAKAN[] = {
  {7, 0, 121.8, true},
  {12, 0, 121.8, true},
{18, 0, 121.8, true}
}; const int JUMLAH_JADWAL =
3;

const unsigned long INTERVAL_MONITORING = 5000; const
unsigned long INTERVAL_DISPLAY = 5000;

// ========== KONFIGURASI AUTO-RECONNECT ========== const
unsigned long FIREBASE_RECONNECT_INTERVAL = 30000;
unsigned long FIREBASE_RETRY_DELAY = 5000;
const int MAX_FIREBASE_RETRY = 3; const unsigned long
WIFI_CHECK_INTERVAL = 10000; const unsigned long
POWER_LOSS_CHECK_INTERVAL = 60000;

// ========== OBJEK GLOBAL ==========
RTC_DS3231 rtc;
Servo servoMotor;
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
```

```
const
FirebaseData firebaseData;
FirebaseAuth auth;
FirebaseConfig config;

// ========== VARIABEL GLOBAL ========== bool
sudahDiberiPakan[JUMLAH_JADWAL] = {false, false, false};
unsigned long lastDisplay = 0; unsigned long
lastFirebaseUpdate = 0; unsigned long lastScheduleCheck = 0;
unsigned long lastHistorySave = 0; unsigned long
wifiConnectTime = 0; unsigned long lastWiFiCheck = 0;
unsigned long lastFirebaseCheck = 0; unsigned long
lastPowerLossCheck = 0;
 int firebaseRetryCount = 0; bool
systemInitialized = false; unsigned long
lastSuccessfulFirebaseWrite = 0;


const unsigned long SCHEDULE_CHECK_INTERVAL = 25000;
 bool wifiConnected =
false; bool firebaseReady =
false; int wifiRetryCount =
0;


String firebaseBasePath = "/" + String(DEVICE_ID);


int levelMakanan = 0; String
statusMakanan = ""; bool
warningMakanan = false;


float temperature = 0.0;
String statusSuhu = ""; bool
warningSuhu = false;
```

```
 int turbidityRaw = 0; float
turbidityVoltage = 0.0; float
turbidityNTU = 0.0; String
statusTurbidity = ""; bool
warningTurbidity = false;

float phValue = 7.0;
String statusPH = ""; bool
warningPH = false;
 unsigned long lastRelayTrigger =
0; int relayAsamTriggerCount = 0;
int relayBasaTriggerCount = 0;
float relayAsamVolume = 0.0; float
relayBasaVolume = 0.0;
 unsigned long systemStartTime =
0;
 float totalFeedToday =
0.0; int feedingCountToday
= 0; int lastFeedingDay =
0;
 int levelStokAsam = 0;              // Percentage stok asam
(0-
100)
String statusStokAsam = "";         // Status stok asam bool
warningStokAsam = false;        // Flag warning stok asam

int levelStokBasa = 0;              // Percentage stok basa (0-
100)
String statusStokBasa = "";         // Status stok basa bool
warningStokBasa = false;        // Flag warning stok basa

// ═══════════════════════════════════════════════════ //
FUNGSI TURBIDITY SENSOR - UPDATED VERSION
// ═══════════════════════════════════════════════════
 float readAverageTurbidityVoltage()
{   float sum = 0;   int validCount =
0;   int saturatedCount = 0;
     for (int i = 0; i < TURBIDITY_SAMPLES;
i++) {     int adc = analogRead(TURBIDITY_PIN);
float voltage = adc * (3.3 / 4095.0);

    // Tracking saturasi     if (adc
>= 4090) saturatedCount++;
```

```
    // Hapus filter atas, hanya filter noise bawah
if (voltage >= 0.05) {        sum += voltage;
validCount++;
    }
delay(TURBIDITY_SAMPLE_DELAY);
  }

  // Warning saturasi (untuk debugging)    if
(saturatedCount > TURBIDITY_SAMPLES / 2) {
Serial.println("⚠️  Turbidity sensor saturasi!");
  }      if (validCount == 0) return -1;
// Error
     return sum /
validCount;
}  float voltageToNTU(float voltage) {    if
(voltage < 0) return -1;  // Error handling

  // ☑️ Persamaan kalibrasi dari Lab PAM Jaya
  // NTU = -303.79 × V + 1000.24
  float ntu = CALIB_SLOPE * voltage + CALIB_INTERCEPT;

  // Handle saturasi - set ke nilai minimum untuk air sangat
jernih    if (voltage >= 3.28) {      return constrain(ntu, 0,
10);
  }

  // Batasi range realistis
return constrain(ntu, 0, 1000);
}

String getTurbidityStatus(float ntu) {
if (ntu < 0) {      return "ERROR";
  } else if (ntu < NTU_OPTIMAL) {
  return "SANGAT BAIK";




  } else if (ntu < NTU_GOOD) {
return "BAIK";
  } else if (ntu < NTU_WARNING) {
return "KURANG BAIK";
  } else {
return "BURUK";
  }
}  void processTurbiditySensor() {    // Baca
voltage dengan averaging    turbidityVoltage =
```

```
readAverageTurbidityVoltage();   turbidityRaw =
analogRead(TURBIDITY_PIN);

  // Convert ke NTU menggunakan persamaan kalibrasi
turbidityNTU = voltageToNTU(turbidityVoltage);

  // Tentukan status berdasarkan threshold nila
statusTurbidity = getTurbidityStatus(turbidityNTU);

  // Set warning flag   warningTurbidity =
(turbidityNTU >= NTU_GOOD);
}

// ========== FUNGSI UTILITY TIMESTAMP ==========
String getTimestamp(DateTime dt) {   char
buffer[20];   sprintf(buffer, "%04d-%02d-%02d
%02d:%02d:%02d",            dt.year(), dt.month(),
dt.day(),           dt.hour(), dt.minute(),
dt.second());   return String(buffer);
}

String getFormattedDate(DateTime dt) {
char buffer[11];
  sprintf(buffer, "%04d-%02d-%02d", dt.year(), dt.month(),
dt.day());   return String(buffer);
}
unsigned long getEpochMillis(DateTime dt) {
return (unsigned long)dt.unixtime() * 1000UL;
}

// ========== FUNGSI UPDATE STATUS POMPA DOSING KE FIREBASE
========== void updateAcidPumpStatus(bool
isActive) {   if (!firebaseReady) {
    Serial.println("Firebase not ready, skip pump status update");
return;
  }

  String statusPath = firebaseBasePath +
"/dosingPump/acidPump/status";
  String lastActivatedPath = firebaseBasePath +
"/dosingPump/acidPump/lastActivated";

  // Update status
  if (Firebase.setBool(firebaseData, statusPath, isActive)) {
    Serial.print("✓ Acid Pump Status: ");
    Serial.println(isActive ? "ACTIVE" : "INACTIVE");
  } else {
```

```
    Serial.print("✗ Failed: ");
    Serial.println(firebaseData.errorReason());
  }      if
(isActive) {
    DateTime now = rtc.now();
    String timestamp = getTimestamp(now);
         if (Firebase.setString(firebaseData,
lastActivatedPath, timestamp)) {
      Serial.print("✓ Acid Pump LastActivated: ");
      Serial.println(timestamp);
    }
  }
}  void updateBasePumpStatus(bool
isActive) {   if (!firebaseReady) {
    Serial.println("Firebase not ready, skip pump status update");
return;
  }

  String statusPath = firebaseBasePath +
"/dosingPump/basePump/status";
  String lastActivatedPath = firebaseBasePath +
"/dosingPump/basePump/lastActivated";

 // Update status
  if (Firebase.setBool(firebaseData, statusPath, isActive)) {
    Serial.print("✓ Base Pump Status: ");
    Serial.println(isActive ? "ACTIVE" : "INACTIVE");
  } else {
    Serial.print("✗ Failed: ");
    Serial.println(firebaseData.errorReason());
  }      if
(isActive) {
    DateTime now = rtc.now();
    String timestamp = getTimestamp(now);
         if (Firebase.setString(firebaseData,
lastActivatedPath, timestamp)) {
      Serial.print("✓ Base Pump LastActivated: ");
      Serial.println(timestamp);
    }
  }
}
void initializeDosingPumpStatus() {
  Serial.println("Initializing dosing pump status...");
updateAcidPumpStatus(false);   updateBasePumpStatus(false);
  Serial.println("✓ Dosing Pump Status Initialized");
}
```

```
// ========== FUNGSI HISTORICAL DATA ==========
void saveHistoricalData() {   if
(!firebaseReady) {
    Serial.println("Firebase not ready, skip historical save");
return;
  }

  DateTime now = rtc.now();
  String currentDate = getFormattedDate(now);
unsigned long timestampKey = getEpochMillis(now);
  String recordedAt = getTimestamp(now);

  String historyPath = firebaseBasePath + "/waterQualityHistory/"
+ currentDate + "/" + String(timestampKey);

  Serial.println("Saving historical data...");
  Serial.print("  Date: ");
  Serial.print(currentDate);
  Serial.print(" | Time: ");
  Serial.println(recordedAt);
     // Save pH   bool phOK = Firebase.setFloat(firebaseData,
historyPath + "/ph", phValue);
  Firebase.setString(firebaseData, historyPath + "/phStatus",
statusPH);
  Firebase.setBool(firebaseData, historyPath + "/phWarning",
warningPH);

  // Save temperature   bool tempOK =
Firebase.setFloat(firebaseData, historyPath +
"/temp", temperature);
  Firebase.setString(firebaseData, historyPath + "/tempStatus",
statusSuhu);
  Firebase.setBool(firebaseData, historyPath + "/tempWarning",
warningSuhu);

  // Save turbidity   bool turbOK =
Firebase.setFloat(firebaseData, historyPath +
"/turbidity", turbidityNTU);
  Firebase.setInt(firebaseData, historyPath + "/turbidityNTU",
(int)turbidityNTU);
  Firebase.setString(firebaseData, historyPath +
"/turbidityStatus", statusTurbidity);
  Firebase.setBool(firebaseData, historyPath +
"/turbidityWarning", warningTurbidity);

  // Save timestamp
```

```
  Firebase.setString(firebaseData, historyPath + "/recordedAt",
recordedAt);
  if (phOK && tempOK && turbOK) {
    Serial.println("Historical data saved!");
    Serial.print("  pH: ");
    Serial.print(phValue, 2);
    Serial.print(" | Temp: ");
    Serial.print(temperature, 1);
  Serial.print("°C | NTU: ");
    Serial.println(turbidityNTU, 1);
  } else {
    Serial.println("Failed to save historical data");
  }
}

// ========== FUNGSI pH SENSOR ==========
float readADC_pH() {   int data[15];
for (int i = 0; i < 15; i++) {
data[i] = analogRead(PH_PIN);
delay(40);
  }    for (int i = 0; i < 14; i++) {
for (int j = i + 1; j < 15; j++) {
if (data[j] < data[i]) {         int
temp = data[i];          data[i] =
data[j];         data[j] = temp;
      }
    }
  }    float sum = 0;   for (int
i = 4; i <= 10; i++) {     sum +=
data[i];
  }      return sum
/ 7.0;
}  float readPH() {   float
adc = readADC_pH();
   if (adc >= adc_pH6) {     float m = (4.0 -
6.86) / (adc_pH4 - adc_pH6);     float b = 6.86 -
(m * adc_pH6);     return (m * adc) + b;   } else
if (adc >= adc_pH9) {     float m = (6.86 - 9.18)
/ (adc_pH6 - adc_pH9);     float b = 9.18 - (m *
adc_pH9);     return (m * adc) + b;
  } else {
return 9.18;
  }
}  void
processPhSensor() {
phValue = readPH();

  if (phValue < PH_MIN || phValue > PH_MAX) {
    statusPH = "ERROR";
warningPH = true;
```

```
  } else if (phValue >= PH_IDEAL_MIN && phValue <= PH_IDEAL_MAX) {
statusPH = "IDEAL";     warningPH = false;
  } else if (phValue < PH_IDEAL_MIN) {
statusPH = "TERLALU ASAM";
warningPH = true;   } else {
statusPH = "TERLALU BASA";
warningPH = true;
  }
}

// ========== FUNGSI TEMPERATURE SENSOR ==========
void processTemperatureSensor() {
sensors.requestTemperatures();   temperature =
sensors.getTempCByIndex(0);

  if (temperature < SUHU_KRITIS_MIN || temperature >
SUHU_KRITIS_MAX) {     statusSuhu = "KRITIS";
warningSuhu = true;
  } else if (temperature >= SUHU_MIN && temperature <= SUHU_MAX) {
statusSuhu = "IDEAL";     warningSuhu = false;
  } else {
    statusSuhu = "NORMAL";
warningSuhu = false;
  }
}

// ========== FUNGSI FOOD LEVEL SENSOR ==========
int bacaJarakUltrasonik() {
digitalWrite(TRIG_PIN, LOW);
delayMicroseconds(2);   digitalWrite(TRIG_PIN,
HIGH); delayMicroseconds(10);
digitalWrite(TRIG_PIN, LOW);
   long duration = pulseIn(ECHO_PIN, HIGH,
  30000); if (duration == 0) return -1;
   int jarak = duration * 0.034 /
  2; return jarak;
}  void processFoodLevelSensor() {
int jarak = bacaJarakUltrasonik();
     if (jarak < 0 || jarak >
TOPLES_TINGGI) {     levelMakanan = 0;
statusMakanan = "ERROR";     warningMakanan
= true;     return;
  }      if (jarak <= JARAK_PENUH) jarak =
JARAK_PENUH;   if (jarak >= JARAK_KOSONG) jarak =
JARAK_KOSONG;
     levelMakanan = map(jarak, JARAK_KOSONG, JARAK_PENUH, 0,
100);
     if (levelMakanan >= 70) {
statusMakanan = "PENUH";
```

```
warningMakanan = false;    } else
if (levelMakanan >= 30) {
statusMakanan = "CUKUP";
warningMakanan = false;    } else
if (levelMakanan >= 15) {
statusMakanan = "RENDAH";
warningMakanan = true;
  } else {
    statusMakanan = "KOSONG";
warningMakanan = true;
  }
}

// ========== FUNGSI STOK CAIRAN ASAM ==========
int bacaJarakAsamUltrasonik() {
digitalWrite(TRIG_ASAM_PIN, LOW);
delayMicroseconds(2);   digitalWrite(TRIG_ASAM_PIN,
HIGH); delayMicroseconds(10);
digitalWrite(TRIG_ASAM_PIN, LOW);
   long duration = pulseIn(ECHO_ASAM_PIN, HIGH,
  30000); if (duration == 0) return -1;
   int jarak = duration * 0.034 /
  2; return jarak;
}  void processStokAsamSensor() {   int
jarak = bacaJarakAsamUltrasonik();
     if (jarak < 0 || jarak >
TANGKI_TINGGI_ASAM) {     levelStokAsam = 0;
statusStokAsam = "ERROR";     warningStokAsam =
true;     return;
  }      if (jarak <= JARAK_ASAM_PENUH) jarak =
JARAK_ASAM_PENUH;   if (jarak >= JARAK_ASAM_KOSONG) jarak =
JARAK_ASAM_KOSONG;
     levelStokAsam = map(jarak, JARAK_ASAM_KOSONG,
JARAK_ASAM_PENUH,
0, 100);      if (levelStokAsam >=
70) {     statusStokAsam = "PENUH";
warningStokAsam = false;    } else
if (levelStokAsam >= 30) {
statusStokAsam = "CUKUP";
warningStokAsam = false;    } else
if (levelStokAsam >= 15) {
statusStokAsam = "RENDAH";
warningStokAsam = true;
  } else {     statusStokAsam
= "KOSONG";
warningStokAsam = true;
  }
}

// ========== FUNGSI STOK CAIRAN BASA ==========
int bacaJarakBasaUltrasonik() {
```

```
digitalWrite(TRIG_BASA_PIN, LOW);
delayMicroseconds(2); digitalWrite(TRIG_BASA_PIN,
HIGH); delayMicroseconds(10);
digitalWrite(TRIG_BASA_PIN, LOW);
   long duration = pulseIn(ECHO_BASA_PIN, HIGH,
  30000); if (duration == 0) return -1;
   int jarak = duration * 0.034 /
  2;
  return jarak;
}  void processStokBasaSensor() {   int
jarak = bacaJarakBasaUltrasonik();
     if (jarak < 0 || jarak >
TANGKI_TINGGI_BASA) {     levelStokBasa = 0;
statusStokBasa = "ERROR";     warningStokBasa =
true;     return;
  }      if (jarak <= JARAK_BASA_PENUH) jarak =
JARAK_BASA_PENUH;   if (jarak >= JARAK_BASA_KOSONG) jarak =
JARAK_BASA_KOSONG;
     levelStokBasa = map(jarak, JARAK_BASA_KOSONG,
JARAK_BASA_PENUH,
0, 100);      if (levelStokBasa >=
70) {     statusStokBasa = "PENUH";
warningStokBasa = false;    } else
if (levelStokBasa >= 30) {
statusStokBasa = "CUKUP";
warningStokBasa = false;    } else
if (levelStokBasa >= 15) {
statusStokBasa = "RENDAH";
warningStokBasa = true;
  } else {     statusStokBasa
= "KOSONG";
warningStokBasa = true;
  }
}

// ========== FUNGSI pH CONTROL ==========
void checkAndControlPH() {   unsigned long
currentMillis = millis();
   if (currentMillis - lastRelayTrigger < RELAY_COOLDOWN)
  {   return;
  }
  // ========== KONTROL pH TERLALU RENDAH (BUTUH BASA) ==========
  if (phValue < PH_TRIGGER_ASAM && !warningStokBasa) {
  Serial.println("=========================================");
    Serial.println("pH terlalu rendah! Trigger relay BASA");


    //Update status ke Firebase: POMPA BASA AKTIF
updateBasePumpStatus(true);
```

```
    // Aktifkan relay BASA
digitalWrite(RELAY2_PIN, LOW);
Serial.println("Relay BASA ON");
delay(RELAY_BASA_DURATION);
digitalWrite(RELAY2_PIN, HIGH);
    Serial.println("Relay BASA OFF");

    //Update status ke Firebase: POMPA BASA NONAKTIF
updateBasePumpStatus(false);
        relayBasaTriggerCount++;
relayBasaVolume += ML_PER_TRIGGER_BASA;
lastRelayTrigger = currentMillis;

    DateTime now = rtc.now();
    String logPath = firebaseBasePath + "/phControlLog/" +
String(getEpochMillis(now));

    Firebase.setString(firebaseData, logPath + "/type", "BASA");
Firebase.setFloat(firebaseData, logPath + "/phBefore", phValue);
    Firebase.setFloat(firebaseData, logPath + "/volumeMl",
ML_PER_TRIGGER_BASA);
    Firebase.setString(firebaseData, logPath + "/timestamp",
getTimestamp(now));

    Serial.println("✓ pH Control BASA Complete");
    Serial.println("═══════════════════════════════════════");
  }
  // ========== KONTROL pH TERLALU TINGGI (BUTUH ASAM) ==========
else if (phValue > PH_TRIGGER_BASA && !warningStokAsam) {
Serial.println("═══════════════════════════════════════");
    Serial.println("pH terlalu tinggi! Trigger relay ASAM");

    //Update status ke Firebase: POMPA ASAM AKTIF
    updateAcidPumpStatus(true);

    // Aktifkan relay ASAM
  digitalWrite(RELAY1_PIN, LOW);
    Serial.println("Relay ASAM ON");
delay(RELAY_ASAM_DURATION);     digitalWrite(RELAY1_PIN,
HIGH);
    Serial.println("Relay ASAM OFF");

    //Update status ke Firebase: POMPA ASAM NONAKTIF
updateAcidPumpStatus(false);
        relayAsamTriggerCount++;
relayAsamVolume += ML_PER_TRIGGER_ASAM;
lastRelayTrigger = currentMillis;

    DateTime now = rtc.now();
```

```
    String logPath = firebaseBasePath + "/phControlLog/" +
String(getEpochMillis(now));

    Firebase.setString(firebaseData, logPath + "/type", "ASAM");
Firebase.setFloat(firebaseData, logPath + "/phBefore", phValue);
    Firebase.setFloat(firebaseData, logPath + "/volumeMl",
ML_PER_TRIGGER_ASAM);
    Firebase.setString(firebaseData, logPath + "/timestamp",
getTimestamp(now));


    Serial.println("✓ pH Control ASAM Complete");
    Serial.println("═══════════════════════════════════════");
  }

  // Warning jika stok menipis
  if (warningStokAsam && phValue > PH_TRIGGER_BASA) {
    Serial.println("⚠  PERINGATAN: Stok asam menipis! Isi ulang
segera.");   }
  if (warningStokBasa && phValue < PH_TRIGGER_ASAM) {
    Serial.println("⚠  PERINGATAN: Stok basa menipis! Isi ulang
segera.");
  }
}

// ========== FUNGSI PEMBERIAN PAKAN ========== void
beriPakan(FeedingSchedule schedule) {
  if (levelMakanan < 10) {
    Serial.println("⚠ Level pakan terlalu rendah!");
    return;
  }

  // Langsung pakai durasi dari schedule (tidak perlu hitung
lagi!)   int duration = schedule.durationMs;

  // Validasi durasi (safety limits)
if (duration < 50) duration = 50;   if
(duration > 2000) duration = 2000;


  Serial.print("🍽 Memberi pakan: ");
  Serial.print(schedule.weightGrams, 1);
  Serial.print("g (");
  Serial.print(duration);
Serial.println("ms)");

servoMotor.write(SERVO_BUKA);
```

```
delay(duration);
servoMotor.write(SERVO_TUTUP);
     totalFeedToday +=
schedule.weightGrams;
feedingCountToday++;

  DateTime now = rtc.now();
  String feedingPath = firebaseBasePath + "/feedingLogs/" +
String(getEpochMillis(now));

  Firebase.setFloat(firebaseData, feedingPath + "/weightGrams",
schedule.weightGrams);
  Firebase.setInt(firebaseData, feedingPath + "/durationMs",
duration);
  Firebase.setString(firebaseData, feedingPath + "/timestamp",
getTimestamp(now));

  Serial.println("✓ Pakan berhasil diberikan");
}
void cekDanBeriPakan() {
DateTime now = rtc.now();
   if (now.day() != lastFeedingDay) { for
  (int i = 0; i < JUMLAH_JADWAL; i++) {
  sudahDiberiPakan[i] = false;
    }
    totalFeedToday = 0.0;
   feedingCountToday = 0;
  lastFeedingDay = now.day();
     relayAsamTriggerCount =
0;      relayBasaTriggerCount =
0;      relayAsamVolume = 0.0;
relayBasaVolume = 0.0;

    Serial.println("New day! Reset feeding flags");
  }      for (int i = 0; i < JUMLAH_JADWAL;
i++) {      if (!JADWAL_PAKAN[i].enabled)
continue;      if (sudahDiberiPakan[i])
continue;

    if (now.hour() == JADWAL_PAKAN[i].hour && now.minute() ==
JADWAL_PAKAN[i].minute) {       beriPakan(JADWAL_PAKAN[i]);
sudahDiberiPakan[i] = true;
    }
  }
}

// ========== WIFI & FIREBASE ========== void
setupWiFi() {
  Serial.print("Connecting WiFi");
```

```
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
     wifiConnectTime =
millis();    int dots = 0;
     while (WiFi.status() != WL_CONNECTED) {     if
(millis() - wifiConnectTime > WIFI_TIMEOUT) {
Serial.println("\nWiFi timeout!");
wifiRetryCount++;

      if (wifiRetryCount >= MAX_WIFI_RETRY) {
        Serial.println("Max WiFi retry reached. Continuing
offline...");       wifiConnected = false;       return;
      }

       Serial.println("Retrying WiFi...");
    delay(WIFI_RETRY_DELAY);
     WiFi.disconnect();
      WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
wifiConnectTime = millis();       continue;
    }

    Serial.print(".");     if (++dots % 20 == 0)
Serial.println();     digitalWrite(LED_WIFI_PIN,
!digitalRead(LED_WIFI_PIN));     delay(500);
  }

  Serial.println("\n✓ WiFi connected!");
  Serial.print("IP: ");
  Serial.println(WiFi.localIP());
     wifiConnected = true;
wifiRetryCount = 0;
digitalWrite(LED_WIFI_PIN, HIGH);
}  void setupFirebase()
{   if (!wifiConnected)
{
    Serial.println("WiFi not connected. Skipping Firebase
setup.");     return;
  }

  Serial.println("Setting up Firebase...");
     config.host = FIREBASE_HOST;
config.signer.tokens.legacy_token = FIREBASE_AUTH;

  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);

delay(2000);
  if (Firebase.ready()) {
```

```
    Serial.println("✓ Firebase connected!");
  firebaseReady = true;
    firebaseRetryCount = 0;

    Firebase.setString(firebaseData, firebaseBasePath +
"/device/status", "ONLINE");
    Firebase.setString(firebaseData, firebaseBasePath +
"/device/version", VERSION);
    Firebase.setString(firebaseData, firebaseBasePath +
"/device/id", DEVICE_ID);

    DateTime now = rtc.now();
    Firebase.setString(firebaseData, firebaseBasePath +
"/device/lastBoot", getTimestamp(now));
  } else {
    Serial.println("Firebase connection failed!");
firebaseReady = false;
  }
}  void
attemptFirebaseReconnect() {   if
(!wifiConnected) {
    Serial.println("Cannot reconnect Firebase: WiFi not
connected");     return;
  }
  if (firebaseRetryCount >= MAX_FIREBASE_RETRY) {
Serial.println("Max Firebase retry reached. Will retry
later...");     firebaseRetryCount = 0;     return;
  }

  Serial.print("Reconnecting Firebase (attempt ");
  Serial.print(firebaseRetryCount + 1);
  Serial.print("/");
  Serial.print(MAX_FIREBASE_RETRY);
  Serial.println(")...");

  Firebase.begin(&config, &auth);
delay(FIREBASE_RETRY_DELAY);

  if (Firebase.ready()) {
    Serial.println("Firebase
 reconnected!");    firebaseReady = true;
 firebaseRetryCount = 0;
 lastSuccessfulFirebaseWrite = millis();
  } else {
    Serial.println("Firebase reconnect failed");
firebaseRetryCount++;     firebaseReady = false;
  }
}
```

```
// ========== UPLOAD FIREBASE ==========
bool uploadAllSensors() {   if
(!firebaseReady) return false;
DateTime now = rtc.now();
  String timestamp = getTimestamp(now);

  String sensorPath = firebaseBasePath + "/sensors";
bool allSuccess = true;

  // Upload turbidity   if
(!Firebase.setFloat(firebaseData, sensorPath +
"/turbidity/ntu", turbidityNTU)) {
    Serial.println("  ✗ Turbidity upload failed");
allSuccess = false;
  }
  Firebase.setInt(firebaseData, sensorPath +
"/turbidity/turbidityNTU", (int)turbidityNTU);
  Firebase.setString(firebaseData, sensorPath +
"/turbidity/status", statusTurbidity);
  Firebase.setBool(firebaseData, sensorPath +
"/turbidity/warning", warningTurbidity);
  Firebase.setString(firebaseData, sensorPath +
"/turbidity/timestamp", timestamp);

  // Upload pH   if (!Firebase.setFloat(firebaseData, sensorPath
+ "/ph/value", phValue)) {
    Serial.println("  ✗ pH upload failed");
allSuccess = false;
  }
  Firebase.setString(firebaseData, sensorPath + "/ph/status",
statusPH);
  Firebase.setBool(firebaseData, sensorPath + "/ph/warning",
warningPH);
  Firebase.setString(firebaseData, sensorPath + "/ph/timestamp",
timestamp);

  // Upload temperature
  if (!Firebase.setFloat(firebaseData, sensorPath +
"/temperature/value", temperature)) {
    Serial.println("  ✗ Temperature upload failed");
allSuccess = false;
  }
  Firebase.setString(firebaseData, sensorPath +
"/temperature/status", statusSuhu);
  Firebase.setBool(firebaseData, sensorPath +
"/temperature/warning", warningSuhu);
  Firebase.setString(firebaseData, sensorPath +
```

```
"/temperature/timestamp", timestamp);

  // Upload food level   if
(!Firebase.setInt(firebaseData, sensorPath +
"/foodLevel/percentage", levelMakanan)) {
Serial.println("  ✗ Food level upload failed");
allSuccess = false;
  }
  Firebase.setString(firebaseData, sensorPath +
"/foodLevel/status", statusMakanan);
  Firebase.setBool(firebaseData, sensorPath +
"/foodLevel/warning", warningMakanan);

  // Upload Stok Cairan Asam   if
(!Firebase.setInt(firebaseData, sensorPath +
"/stokCairan/asam/percentage", levelStokAsam)) {
Serial.println("  ✗ Stok Cairan Asam upload failed");
allSuccess = false;
  }
  Firebase.setString(firebaseData, sensorPath +
"/stokCairan/asam/status", statusStokAsam);
  Firebase.setBool(firebaseData, sensorPath +
"/stokCairan/asam/warning", warningStokAsam);

  // Upload Stok Cairan Basa
  if (!Firebase.setInt(firebaseData, sensorPath +
"/stokCairan/basa/percentage", levelStokBasa)) {
Serial.println("  ✗ Stok Cairan Basa upload failed");
allSuccess = false;
  }
  Firebase.setString(firebaseData, sensorPath +
"/stokCairan/basa/status", statusStokBasa);
  Firebase.setBool(firebaseData, sensorPath +
"/stokCairan/basa/warning", warningStokBasa); return allSuccess;
}  void uploadSystemStatus()
{   if (!firebaseReady)
return;

  String statusPath = firebaseBasePath + "/systemStatus";

  Firebase.setString(firebaseData, statusPath + "/status",
"RUNNING");
  Firebase.setBool(firebaseData, statusPath + "/wifiConnected",
wifiConnected);

  if (wifiConnected) {
    Firebase.setInt(firebaseData, statusPath + "/wifiRSSI",
WiFi.RSSI());
```

```
  }
  unsigned long uptime = (millis() - systemStartTime) / 1000;
Firebase.setInt(firebaseData, statusPath + "/uptimeSeconds",
uptime);
}  void
syncFeedingSchedulesFromFirebase() {   if
(!firebaseReady) {
    Serial.println("⚠️  Sync skipped: Firebase not ready");
return;
  }


  Serial.println("🔄 Syncing schedules from Firebase...");

  String schedulePath = firebaseBasePath + "/feeding/schedules";

  for (int i = 0; i < JUMLAH_JADWAL; i++) {
    String path = schedulePath + "/schedule_" + String(i);

    Serial.print("  Checking schedule_");
    Serial.print(i);
    Serial.print("... ");

    // Coba baca hour dulu untuk cek apakah schedule exist
if (Firebase.getInt(firebaseData, path + "/hour")) {
int hour = firebaseData.intData();

     // Baca semua field
     Firebase.getInt(firebaseData, path + "/minute");
int minute = firebaseData.intData();

      Firebase.getFloat(firebaseData, path + "/weight_grams");  //
← snake_case       float weight =
firebaseData.floatData();

      Firebase.getInt(firebaseData, path + "/duration_ms");     //
← snake_case       int duration =
firebaseData.intData();

      Firebase.getBool(firebaseData, path + "/enabled");
bool enabled = firebaseData.boolData();

      // Cek apakah ada perubahan       bool changed =
(JADWAL_PAKAN[i].hour != hour) ||
(JADWAL_PAKAN[i].minute != minute) ||
                   (abs(JADWAL_PAKAN[i].weightGrams - weight) >
0.1) ||
```

```
                    (JADWAL_PAKAN[i].durationMs != duration) ||
                    (JADWAL_PAKAN[i].enabled != enabled);
            if
(changed) {
        JADWAL_PAKAN[i].hour = hour;
        JADWAL_PAKAN[i].minute = minute;
        JADWAL_PAKAN[i].weightGrams = weight;
        JADWAL_PAKAN[i].durationMs = duration;
        JADWAL_PAKAN[i].enabled = enabled;

        // Reset flag sudah diberi pakan jika jadwal berubah
sudahDiberiPakan[i] = false;

        Serial.print("✓ Updated: ");
        Serial.print(hour);
Serial.print(":");
        if (minute < 10) Serial.print("0");
        Serial.print(minute);
        Serial.print(" - ");
        Serial.print(weight, 1);
        Serial.print("g (");
        Serial.print(enabled ? "ENABLED" : "DISABLED");
  Serial.println(")");
      } else {
        Serial.println("No changes");
      }

    } else {
      Serial.println("Not found");

      // Jika schedule tidak ditemukan, disable
if          (JADWAL_PAKAN[i].enabled)          {
JADWAL_PAKAN[i].enabled          =          false;
Serial.println("  → Disabled");
      }
    }
  }

  Serial.println("✓ Sync complete");
}

// ========== FUNGSI HELPER UNTUK JADWAL ==========
String formatScheduleTime(int hour, int minute) {
char buffer[6];   sprintf(buffer, "%02d:%02d",
hour, minute);   return String(buffer);
}
```

```
String getTimeUntilSchedule(DateTime now, int scheduleIndex) {
if (scheduleIndex < 0 || scheduleIndex >= JUMLAH_JADWAL) {
return "--";
  }      int currentMinutes = now.hour() * 60 + now.minute();
int scheduleMinutes = JADWAL_PAKAN[scheduleIndex].hour * 60 +
JADWAL_PAKAN[scheduleIndex].minute;
     int minutesLeft = scheduleMinutes -
currentMinutes;   if (minutesLeft < 0) {
    minutesLeft += 1440; // Tambah 24 jam jika jadwal besok
  }
  int hoursLeft = minutesLeft / 60;
int minsLeft = minutesLeft % 60;
     char
buffer[10];
  sprintf(buffer, "%dh %dm", hoursLeft, minsLeft);
return String(buffer);
}

// ========== DISPLAY FUNCTIONS ==========
void printDateTime(DateTime dt) {
  Serial.print(dt.year());
  Serial.print('/');
  Serial.print(dt.month());
  Serial.print('/');
  Serial.print(dt.day());
  Serial.print(' ');
  Serial.print(dt.hour());
Serial.print(':');
  if (dt.minute() < 10) Serial.print('0');
  Serial.print(dt.minute());
Serial.print(':');
  if (dt.second() < 10) Serial.print('0');
  Serial.println(dt.second());
}

// ========== SETUP ==========
void setup() {
Serial.begin(115200);
delay(500);    systemStartTime
= millis();
   pinMode(LED_WIFI_PIN, OUTPUT);
pinMode(LED_STATUS_PIN, OUTPUT);
pinMode(TRIG_PIN, OUTPUT);
pinMode(ECHO_PIN, INPUT);
pinMode(TRIG_ASAM_PIN, OUTPUT);
pinMode(ECHO_ASAM_PIN, INPUT);
pinMode(TRIG_BASA_PIN, OUTPUT);
pinMode(ECHO_BASA_PIN, INPUT);
```

```
pinMode(RELAY1_PIN, OUTPUT);
pinMode(RELAY2_PIN, OUTPUT);
     digitalWrite(RELAY1_PIN,
HIGH);   digitalWrite(RELAY2_PIN,
HIGH);
  Serial.println("Relay initialized (Active LOW)");

  // ADC Configuration untuk turbidity
analogReadResolution(12);
  analogSetAttenuation(ADC_11db);
  Serial.println("ADC configured (12-bit, 0-3.3V)");

  Wire.begin(SDA_PIN, SCL_PIN);

  // ========== RTC SETUP ==========
  Serial.print("Initializing RTC... ");
  if (!rtc.begin()) {
    Serial.println("FAILED!");
  } else {
    Serial.println("OK");

    if (rtc.lostPower()) {
      Serial.println("POWER LOSS DETECTED!");
      Serial.println("System auto-recovering from power loss...");
      Serial.println("RTC time restored");
      Serial.println("System will reconnect WiFi and
Firebase...");
    }

    // Format: DateTime(YYYY, MM, DD, HH, MM, SS)
//rtc.adjust(DateTime(2026, 1, 17, 16, 47, 0));
    DateTime now = rtc.now();
Serial.print("Current RTC Time: ");
printDateTime(now);     lastFeedingDay
= now.day();
  }

  Serial.print("Servo... ");
servoMotor.attach(SERVO_PIN);
servoMotor.write(SERVO_TUTUP);
  Serial.println("OK");

  Serial.print("Temperature... ");
sensors.begin();    Serial.println("OK");

  // ========== CONNECTION SETUP ==========
setupWiFi();    setupFirebase();
```

```
  if (firebaseReady) {
initializeDosingPumpStatus();
  }
   systemInitialized = true;
  lastSuccessfulFirebaseWrite = millis();

  Serial.println("\nSystem ready!\n"); delay(2000);
}

// ========== DISPLAY ==========
void displayInfo() {   DateTime
now = rtc.now();

  Serial.println("\n═══════════════════════════════════════");
  Serial.println("        N3 FARM MONITORING");
  Serial.println("═══════════════════════════════════════");
Serial.print("Time: ");   printDateTime(now);

  Serial.print("Uptime: ");
  unsigned long uptime = (millis() - systemStartTime) / 1000;
  Serial.print(uptime / 3600);
  Serial.print("h ");
  Serial.print((uptime % 3600) / 60);
  Serial.println("m");

  Serial.println("─────────────────────────────────────");
  Serial.print("WiFi: ");
  Serial.print(wifiConnected ? "✓" : "✗");
if (wifiConnected) {     Serial.print("
(");
    Serial.print(WiFi.RSSI());
    Serial.print(" dBm)");
  }
  Serial.println();

  Serial.print("Firebase: ");
  Serial.println(firebaseReady ? "✓" : "✗");
   if (firebaseReady && lastSuccessfulFirebaseWrite > 0)
{     unsigned long timeSinceWrite = (millis() -
lastSuccessfulFirebaseWrite) / 1000;
Serial.print("Last Write: ");
    Serial.print(timeSinceWrite);
    Serial.println(" sec ago");
  }

  Serial.println("─────────────────────────────────────");
  Serial.print("Feed Today: ");
```

```
  Serial.print(totalFeedToday, 1); Serial.print("g
  (");
  Serial.print(feedingCountToday);
  Serial.println(" times)");

  Serial.println("──────────────────────────────────────");
  Serial.print("Temp: ");
  Serial.print(temperature, 1);
  Serial.print("°C (");
  Serial.print(statusSuhu);
  Serial.println(")");

  Serial.print("pH: ");
  Serial.print(phValue, 2);
  Serial.print(" (");
  Serial.print(statusPH);
  Serial.println(")");

  Serial.print("Turbidity: ");
  Serial.print(turbidityNTU, 1);
  Serial.print(" NTU | ");
  Serial.print(turbidityVoltage, 3);
  Serial.print("V (");
  Serial.print(statusTurbidity);
  Serial.println(")");

  Serial.print("Food: ");
  Serial.print(levelMakanan);
  Serial.print("% (");
  Serial.print(statusMakanan);
  Serial.println(")");

  Serial.print("Stok Asam: ");
  Serial.print(levelStokAsam);
  Serial.print("% (");
Serial.print(statusStokAsam);   if
(warningStokAsam) Serial.print(" ⚠");
  Serial.println(")");

  Serial.print("Stok Basa: ");
  Serial.print(levelStokBasa);
  Serial.print("% (");
  Serial.print(statusStokBasa); if
  (warningStokBasa) Serial.print(" ⚠");
  Serial.println(")");

  Serial.println("──────────────────────────────────────");
```

```
  Serial.println("FEEDING SCHEDULE:");

  // Tampilkan semua jadwal  for (int i =
 0; i < JUMLAH_JADWAL; i++) {
    Serial.print("  #");
    Serial.print(i + 1);
    Serial.print(": ");
    Serial.print(formatScheduleTime(JADWAL_PAKAN[i].hour,
JADWAL_PAKAN[i].minute));
    Serial.print(" - ");
    Serial.print(JADWAL_PAKAN[i].weightGrams, 1);
    Serial.print("g");

    // Status enabled/disabled
if (!JADWAL_PAKAN[i].enabled) {
Serial.print(" [DISABLED]");
    } else if (sudahDiberiPakan[i]) {
      Serial.print(" [✓ DONE]");
    } else {
      Serial.print(" [PENDING]");
    }

    Serial.println();
  }

  // Tampilkan jadwal berikutnya     int
nextSchedule    =    getNextSchedule(now);
Serial.print("      Next:   ");          if
(nextSchedule >= 0) {
    Serial.print("#");
    Serial.print(nextSchedule + 1);
    Serial.print(" at ");
    Serial.print(formatScheduleTime(JADWAL_PAKAN[nextSchedule].hou
r, JADWAL_PAKAN[nextSchedule].minute));
    Serial.print(" (in ");
    Serial.print(getTimeUntilSchedule(now, nextSchedule));
    Serial.println(")");
  } else {
    Serial.println("All schedules completed for today");
  }

  Serial.println("————————————————————————————————————————");
  Serial.print("pH Control - Asam: ");
  Serial.print(relayAsamTriggerCount);
  Serial.print(" (");
  Serial.print(relayAsamVolume, 1);
```

```
  Serial.println("ml)");

  Serial.print("pH Control - Basa: ");
  Serial.print(relayBasaTriggerCount);
  Serial.print(" (");
  Serial.print(relayBasaVolume, 1);
  Serial.println("ml)");
     unsigned long currentMillis = millis();   if (currentMillis
- lastRelayTrigger < RELAY_COOLDOWN) {     unsigned long
cooldownLeft = (RELAY_COOLDOWN - (currentMillis
- lastRelayTrigger)) / 1000;
    Serial.print("Cooldown: ");
    Serial.print(cooldownLeft / 60);
Serial.print("m ");
    Serial.print(cooldownLeft % 60);
    Serial.println("s");
  } else {
    Serial.println("Cooldown: Ready");
  }

  Serial.println("=====================================\n");
}

// ========== LOOP ========== void loop()
{   unsigned long currentMillis =
millis();

  // ========== WIFI HEALTH CHECK & AUTO-RECONNECT ==========
if (currentMillis - lastWiFiCheck >= WIFI_CHECK_INTERVAL) {
if (WiFi.status() != WL_CONNECTED && wifiConnected) {
Serial.println("WiFi disconnected! Auto-reconnecting...");
wifiConnected = false;       firebaseReady = false;
digitalWrite(LED_WIFI_PIN, LOW);

setupWiFi();

      if (wifiConnected) {
        Serial.println("✓ WiFi reconnected. Reconnecting
Firebase...");
  setupFirebase();
      } } lastWiFiCheck =
    currentMillis;
  }

  // ========== FIREBASE HEALTH CHECK & AUTO-RECONNECT ==========
if (currentMillis - lastFirebaseCheck >=
FIREBASE_RECONNECT_INTERVAL) {     if (wifiConnected &&
!firebaseReady) {       Serial.println("Firebase not ready.
Attempting reconnect...");       attemptFirebaseReconnect();
```

```
    }          if (firebaseReady &&
(currentMillis -
lastSuccessfulFirebaseWrite > 300000)) {
      Serial.println("No successful Firebase write for 5
minutes!");
      Serial.println("Force reconnecting Firebase...");
attemptFirebaseReconnect();
    }          lastFirebaseCheck =
currentMillis;
  }

  // Baca sensor
processTemperatureSensor();
processTurbiditySensor();
processFoodLevelSensor();
processStokAsamSensor();
processStokBasaSensor();
processPhSensor();

  // pH Control
checkAndControlPH();

  // Pemberian pakan
cekDanBeriPakan();

  // Upload realtime (5 detik)   if
(currentMillis - lastFirebaseUpdate >=
FIREBASE_UPDATE_INTERVAL) {     if
(uploadAllSensors()) {
      lastSuccessfulFirebaseWrite = currentMillis;
    }   uploadSystemStatus();
  lastFirebaseUpdate = currentMillis;
  }
  // Sync jadwal (25 detik) if
(currentMillis - lastScheduleCheck >=
SCHEDULE_CHECK_INTERVAL) {
syncFeedingSchedulesFromFirebase();
lastScheduleCheck = currentMillis;
  }

  // Save historical data (10 menit)   if (currentMillis -
lastHistorySave >= HISTORY_SAVE_INTERVAL) {
saveHistoricalData();     lastHistorySave = currentMillis;
  }

  // Display   if (currentMillis - lastDisplay >=
INTERVAL_DISPLAY) {     displayInfo();     lastDisplay =
currentMillis;
```

```
  }    digitalWrite(LED_STATUS_PIN,
!digitalRead(LED_STATUS_PIN));   delay(100);
}  int getNextSchedule(DateTime now) {   int
currentMinutes = now.hour() * 60 + now.minute();
     for (int i = 0; i < JUMLAH_JADWAL;
i++) {     if (!JADWAL_PAKAN[i].enabled)
continue;

    int scheduleMinutes = JADWAL_PAKAN[i].hour * 60 +
JADWAL_PAKAN[i].minute;     if (scheduleMinutes >
currentMinutes) {       return i;
    }   }
return -1;
}
void blinkLED(int pin, int times) {
  for (int i = 0; i < times; i++) {
  digitalWrite(pin, HIGH);
  delay(100); digitalWrite(pin,
  LOW); delay(100);
  }
}
```

Lampiran 3. Source Code Aplikasi Android

```
import 'dart:async'; import
'package:flutter/material.dart'; import
'package:syncfusion_flutter_gauges/gauges.dart'; import
'package:firebase_database/firebase_database.dart'; import
'notification_service.dart'; import
'ph_control_history_page.dart';
 class HomePage extends StatefulWidget {
const HomePage({Key? key}) : super(key: key);

  @override
  State<HomePage> createState() => _HomePageState();
}
StreamSubscription? _phControlLogListener;
String? _lastProcessedLogKey;
 class _HomePageState extends State<HomePage>
{
  // Firebase Database Reference
final DatabaseReference _database =
FirebaseDatabase.instance.ref();

  // Notification Service
```

```
  final NotificationService _notificationService =
NotificationService();

  // StreamSubscriptions untuk cancel listeners
  StreamSubscription? _tempListener;
  StreamSubscription? _turbidityListener;
  StreamSubscription? _phListener;
  StreamSubscription? _stokAsamListener;
  StreamSubscription? _stokBasaListener;
  StreamSubscription? _acidPumpListener;
  StreamSubscription? _basePumpListener;

  // Data sensor
  double phValue =
  7.0; double
  tempValue = 0.0;
  double
  turbidityValue =
  0.0;

  // Status
  String tempStatus = "Memuat...";
  String turbidityStatus = "Memuat...";
String phStatus = "Memuat...";   bool
tempWarning = false;   bool
turbidityWarning = false;   bool
phWarning = false;

  // Previous warning states untuk deteksi perubahan
bool _previousTempWarning = false;   bool
_previousTurbidityWarning = false;   bool
_previousPhWarning = false;

  // Stok Cairan Asam dan Basa   int
stokAsamPercentage = 0;   String
stokAsamStatus = "Memuat...";   bool
stokAsamWarning = false;   bool
_previousStokAsamWarning = false;

  int stokBasaPercentage = 0;   String
stokBasaStatus = "Memuat...";   bool
stokBasaWarning = false;   bool
_previousStokBasaWarning = false;   bool
acidPumpStatus = false;   bool
basePumpStatus = false;

  // Previous pump states untuk deteksi perubahan (menyala dari
mati)   bool _previousAcidPumpStatus = false;   bool
_previousBasePumpStatus = false;
```

```
  // Valve control   bool
isValveOpen = false;   bool
isLoading = false;

  // Connection status
bool isConnected = false;

  @override  void initState()
 { super.initState();
 _initializeNotifications();
 _setupFirebaseListeners();
    _setupPhControlLogListener();
  }

  Future<void> _initializeNotifications() async {
await _notificationService.initialize();
  }    void _setupFirebaseListeners() {
const String deviceId = 'ESP32_N3FARM_001';

    // Listener untuk Temperature
    _tempListener =
_database.child('$deviceId/sensors/temperature').onValue.listen((e
vent) {       if (!mounted) return;
       if (event.snapshot.value != null) {         final
data = event.snapshot.value as Map<dynamic, dynamic>;
setState(() {           tempValue = (data['value'] ??
0.0).toDouble();           tempStatus = data['status'] ??
'Unknown';           tempWarning = data['warning'] ??
false;           isConnected = true;
        });

        // 🔔 Cek dan kirim notifikasi temperature
        _checkTemperatureWarning();
      }
    }, onError: (error) {
print('Error temperature: $error');
if (mounted) {         setState(() {
isConnected = false;
        });
      }
    });

    // Listener untuk Turbidity
    _turbidityListener =
_database.child('$deviceId/sensors/turbidity').onValue.listen((eve
nt) {   if (!mounted) return;
      if (event.snapshot.value != null) {   final data =
      event.snapshot.value as Map<dynamic,
```

```
dynamic>;          setState(() {            turbidityValue =
(data['ntu'] ?? 0.0).toDouble();            turbidityStatus
= data['status'] ?? 'Unknown';           turbidityWarning =
data['warning'] ?? false;           isConnected = true;
        });

        // 🔔 Cek dan kirim notifikasi turbidity
        _checkTurbidityWarning();
      }
    }, onError: (error) {
print('Error turbidity: $error');
if (mounted) {          setState(() {
isConnected = false;
        });
      }
    });

    // Listener untuk pH
    _phListener =
_database.child('$deviceId/sensors/ph').onValue.listen((event) {
if (!mounted) return;
      if (event.snapshot.value != null) {          final
data = event.snapshot.value as Map<dynamic, dynamic>;
setState(() {           phValue = (data['value'] ??
7.0).toDouble();           phStatus = data['status'] ??
'Unknown';          phWarning = data['warning'] ??
false;          isConnected = true;
        });

        // 🔔 Cek dan kirim notifikasi pH
        _checkPhWarning();
      }
    }, onError: (error) {
 print('Error pH: $error');
 if (mounted) {
 setState(() {
 isConnected = false;   });
      }
    });

    // Listener untuk Stok Cairan Asam
    _stokAsamListener =
_database.child('$deviceId/sensors/stokCairan/asam').onValue.liste
n((event) {       if (!mounted) return;
      if (event.snapshot.value != null) {          final
data = event.snapshot.value as Map<dynamic, dynamic>;
setState(() {           stokAsamPercentage =
(data['percentage'] ?? 0);           stokAsamStatus =
```

```
data['status'] ?? 'Unknown';            stokAsamWarning =
data['warning'] ?? false;            isConnected = true;
        });

        // 🔔 Cek dan kirim notifikasi stok asam
        _checkStokAsamWarning();
      }
    }, onError: (error) {
print('Error stok asam: $error');
if (mounted) {          setState(() {
isConnected = false;
        });
      }
    });

    // Listener untuk Stok Cairan Basa
    _stokBasaListener =
_database.child('$deviceId/sensors/stokCairan/basa').onValue.liste
n((event) {       if (!mounted) return;
       if (event.snapshot.value != null) {         final
data = event.snapshot.value as Map<dynamic, dynamic>;
        setState(() {          stokBasaPercentage =
 (data['percentage'] ?? 0);          stokBasaStatus =
 data['status'] ?? 'Unknown';            stokBasaWarning =
 data['warning'] ?? false;     isConnected = true;
        });

        // 🔔 Cek dan kirim notifikasi stok basa
        _checkStokBasaWarning();
      }
    }, onError: (error) {
print('Error stok basa: $error');
if (mounted) {          setState(() {
isConnected = false;
        });
      }
    });

    // Listener untuk Status Pompa Asam
    _acidPumpListener =
_database.child('$deviceId/dosingPump/acidPump/status').onValue.li
sten((event) {       if (!mounted) return;       if
(event.snapshot.value != null) {         final bool newStatus =
event.snapshot.value as bool? ??
false;

        // Deteksi perubahan dari mati ke menyala
if (newStatus && !_previousAcidPumpStatus) {
// Pompa baru menyala, kirim notifikasi
```

```
_notificationService.showPumpDosingAlert(
pumpType: 'ASAM',              phValue: phValue,
            durationSeconds: 10, // Sesuaikan dengan durasi aktual
dari Firebase jika tersedia
          );
        }
        setState(() {
          acidPumpStatus = newStatus;
_previousAcidPumpStatus = newStatus;
        });
      }
    });

    // Listener untuk Status Pompa Basa
    _basePumpListener =
_database.child('$deviceId/dosingPump/basePump/status').onValue.li
sten((event) {       if (!mounted) return;       if
(event.snapshot.value != null) {         final bool newStatus =
event.snapshot.value as bool? ??
false;

        // Deteksi perubahan dari mati ke menyala         if
(newStatus && !_previousBasePumpStatus) {           // Pompa baru
menyala, kirim notifikasi
_notificationService.showPumpDosingAlert(           pumpType:
'BASA',             phValue: phValue,             durationSeconds:
10, // Sesuaikan dengan durasi aktual dari Firebase jika tersedia
          );
        }          setState(() {
basePumpStatus = newStatus;
_previousBasePumpStatus = newStatus;
        });
      }
    });
  }    void _setupPhControlLogListener() {
const String deviceId = 'ESP32_N3FARM_001';

    _phControlLogListener = _database
        .child('$deviceId/phControlLog')
        .orderByKey()
        .limitToLast(1) // Ambil log terbaru
        .onChildAdded
.listen((event) {       if
(!mounted) return;
       final logKey =
event.snapshot.key;

     // Skip jika log sudah diproses  if (logKey
    == _lastProcessedLogKey) return;
```

```
      if (event.snapshot.value != null) {
         final data = event.snapshot.value as Map<dynamic,
dynamic>;          final String type = data['type']
?? '';          final double phBefore =
(data['phBefore'] ??
7.0).toDouble();

         // Trigger notifikasi
if (type == 'ASAM') {
           _notificationService.showRelayAlert(
relayType: 'ASAM',              phValue:
phBefore,
           );
         } else if (type == 'BASA') {
_notificationService.showRelayAlert(
relayType: 'BASA',              phValue:
phBefore,
           );
         }

         _lastProcessedLogKey = logKey;
print('Relay $type detected');
       }
     });
   }


   // 🔔 Fungsi untuk cek pH warning
void _checkPhWarning() {
     // Hanya kirim notifikasi jika warning berubah dari false ke
true
     if (phWarning && !_previousPhWarning) {
if (phValue > 8.0) {
         _notificationService.showPhAlert(
phValue: phValue,          isHigh: true,
         );
       } else if (phValue < 7.0) {
_notificationService.showPhAlert(
phValue: phValue,          isHigh:
false,
         );
       }
     }
     _previousPhWarning = phWarning;
   }
   // 🔔 Fungsi untuk cek temperature warning
void _checkTemperatureWarning() {
     // Hanya kirim notifikasi jika warning berubah dari false ke
true     if (tempWarning && !_previousTempWarning) {       if
(tempValue > 30.0) {
```

```
        _notificationService.showTemperatureAlert(
tempValue: tempValue,           isHigh: true,
        );
      } else if (tempValue < 25.0) {
        _notificationService.showTemperatureAlert(
tempValue: tempValue,           isHigh: false,
        );
      }
    }
    _previousTempWarning = tempWarning;
  }


  // 🔔 Fungsi untuk cek turbidity warning
void _checkTurbidityWarning() {
    // Hanya kirim notifikasi jika warning berubah dari false ke
true      if (turbidityWarning && !_previousTurbidityWarning) {
if (turbidityValue > 50.0) {
        _notificationService.showTurbidityAlert(
ntuValue: turbidityValue,
        );
      }
    }
    _previousTurbidityWarning = turbidityWarning;
  }


  // 🔔 Fungsi untuk cek stok asam warning
void _checkStokAsamWarning() {
    // Hanya kirim notifikasi jika warning berubah dari false ke
true
    if (stokAsamWarning && !_previousStokAsamWarning) {
    _notificationService.showStokCairanAlert(
    cairanType: 'ASAM',    percentage:
    stokAsamPercentage,
     );
    }
    _previousStokAsamWarning = stokAsamWarning;
  }


  // 🔔 Fungsi untuk cek stok basa warning
void _checkStokBasaWarning() {
    // Hanya kirim notifikasi jika warning berubah dari false ke
true     if (stokBasaWarning && !_previousStokBasaWarning) {
_notificationService.showStokCairanAlert(         cairanType:
'BASA',         percentage: stokBasaPercentage,
      );
    }
    _previousStokBasaWarning = stokBasaWarning;
  }
```

```
  @override
  Widget build(BuildContext context) {     return
Scaffold(       backgroundColor: Colors.grey[100],
appBar: AppBar(         title: Row(
children: [             // Logo Aqualis
ClipRRect(               borderRadius:
BorderRadius.circular(8),               child:
Image.asset(
                'assets/images/Aqualis_Logo_with_Nila_Fish.png',
width: 32,                  height: 32,                 fit:
BoxFit.contain,
              ),
),
            const SizedBox(width: 12),
const Text(                'Home',
style: TextStyle(
color: Colors.black87,
fontSize: 24,
              ),
),
            const SizedBox(width: 12),            Container(
padding: const EdgeInsets.symmetric(horizontal: 8, vertical: 4),
              decoration: BoxDecoration(   color:
              isConnected ? Colors.green[100] :
Colors.red[100],                 borderRadius:
BorderRadius.circular(12),
              ),              child: Row(
mainAxisSize: MainAxisSize.min,                children:
[                   Container(                   width:
8,                     height: 8,
decoration: BoxDecoration(                    color:
isConnected ? Colors.green : Colors.red,
shape: BoxShape.circle,
                    ),
),
                  const SizedBox(width: 4),
Text(                     isConnected ? 'Online' : 'Offline',
style: TextStyle(                     fontSize: 11,
fontWeight: FontWeight.w600,                     color:
isConnected ? Colors.green[900] :
Colors.red[900],
                    ),
                  ),
                ],
              ),
            ),
          ],         ),         backgroundColor: Colors.white,
elevation: 1,          iconTheme: const IconThemeData(color:
Colors.black87),
      ),
```

```
      body: RefreshIndicator(
onRefresh: () async {
setState(() {});
          await Future.delayed(const Duration(seconds: 1));
        }, child:
        SingleChildScrollView(
          physics: const AlwaysScrollableScrollPhysics(),
          child: Padding(
            padding: const EdgeInsets.all(16.0),
            child: Column(   crossAxisAlignment:
CrossAxisAlignment.start,   children: [
Card(                    elevation: 3,
shape: RoundedRectangleBorder(
borderRadius: BorderRadius.circular(12),
                  ),                    child: Padding(
padding: const EdgeInsets.all(16.0),
child: Column(                          children: [
Row(                             mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
children: [
                                _buildGaugeIndicator(
label: 'pH',                                  value:
phValue,                                 minValue: 0,
maxValue: 14,                                  unit:
'',                                   status: phStatus,
hasWarning: phWarning,
                                ),
                                _buildGaugeIndicator(
label: 'Suhu',                                  value:
tempValue,                                minValue: 0,
maxValue: 50,                                  unit:
'°C',                                status:
tempStatus,                                   hasWarning:
tempWarning,
                                ),
                                _buildGaugeIndicator(
label: 'Turbidity',                                       value:
turbidityValue,                                 minValue: 0,
maxValue: 100,                                 unit: 'NTU',
status: turbidityStatus,       hasWarning:
turbidityWarning,
                              ),
                            ],
                          ),
                          const SizedBox(height: 20),
InkWell(   onTap: () {
Navigator.push(                                  context,
MaterialPageRoute(                                    builder:
(context) => const
PhControlHistoryPage(),
                                ),
```

```
                                );                                },
child: Container(                                width:
double.infinity,                                padding: const
EdgeInsets.all(16),                                decoration:
BoxDecoration(                                color:
Colors.grey[300],                                borderRadius:
BorderRadius.circular(8),
),                                child: Row(
mainAxisAlignment:
MainAxisAlignment.spaceBetween,
children: [
const Text(
                                  'Riwayat Kontrol pH',
style: TextStyle(
fontWeight: FontWeight.w600,
fontSize: 14,
                                  ),
                                ),
                                Icon(
                                  Icons.arrow_forward_ios,
size: 16,                                color:
Colors.grey[700],
                                ),
                              ],
                            ),
                          ),
                        ),
                      ],
                    ),
                  ),
                ),
                const SizedBox(height: 16),
                Card(
                  elevation: 3, shape:
                  RoundedRectangleBorder(   borderRadius:
                  BorderRadius.circular(12),
                  ),                    child: Container(
padding: const EdgeInsets.all(16),
decoration: BoxDecoration(                    borderRadius:
BorderRadius.circular(12),                    gradient:
LinearGradient(                    colors:
[Colors.blue[50]!, Colors.blue[100]!],
begin: Alignment.topLeft,                    end:
Alignment.bottomRight,
                  ),                    ),
child: Column(                    crossAxisAlignment:
CrossAxisAlignment.start,                    children: [
Row(                    children: [
Container(                    padding: const
EdgeInsets.all(8),                    decoration:
BoxDecoration(                    color:
Colors.blue[700],                    borderRadius:
```

```
BorderRadius.circular(8),
),                                  child: const Icon(
Icons.opacity,
color: Colors.white,
size: 24,
                                  ),
),                                 const SizedBox(width:
12),                                 const Text(
                                   'Kontrol pH Otomatis',
style: TextStyle(                                    fontSize:
16,                                      fontWeight:
FontWeight.bold,
                                    color: Colors.black87,
                                   ),
                                 ),
                               ],
                             ),
                             const SizedBox(height:
                             16), Row(
                             mainAxisAlignment:
MainAxisAlignment.spaceBetween,
children: [                                    Expanded(
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [                                          const Text(
'Pompa Dosing',                                           style:
TextStyle(                                           fontSize:
12,                                        color:
Colors.black54,
                                        ),
),
                                          const SizedBox(height: 6),
                Row(                                      children: [
      Container(                                           width: 10,
    height: 10,                                           decoration:
    BoxDecoration(                                             color:
                          acidPumpStatus ? Colors.green : Colors.red,
                                              shape: BoxShape.circle,
                                          ),
),
                                        const SizedBox(width: 6),
Flexible(                                            child: Text(
                                       'Asam: ${acidPumpStatus
? "Menyala" : "Mati"}',
style: TextStyle(
fontSize: 13,
fontWeight: FontWeight.w600,
color: acidPumpStatus ? Colors.green[700] : Colors.red[700],
                                            ),
                                          ),
                                        ),
```

```
                                    ],
                                  ),
                                  const SizedBox(height: 4),
                                  Row(
                                                          children: [
    Container(                                             width: 10,
  height: 10,                                             decoration:
  BoxDecoration(                                               color:
                      basePumpStatus ? Colors.green : Colors.red,
                                          shape: BoxShape.circle,
                                        ),
),
                                      const SizedBox(width: 6),
Flexible(                                        child: Text(
                                      'Basa: ${basePumpStatus
                                      ? "Menyala" : "Mati"}',
                                           style: TextStyle(
                                               fontSize: 13,
                                     fontWeight: FontWeight.w600,
                                           color: basePumpStatus
? Colors.green[700] : Colors.red[700],
                                                ),
                                              ),
                                            ),
                                          ],
                                        ),
                                      ],
                                    ),
                                  ),
                                  ElevatedButton.icon(
onPressed: isLoading ? null :
_toggleValve,                                 icon: isLoading
? const SizedBox(                                  width: 16,
height: 16,                                   child:
CircularProgressIndicator(
strokeWidth: 2,     valueColor:
                                  AlwaysStoppedAnimation<Color>(
                                      Colors.white),
                                ),
                              )
                                  : Icon(
                                isValveOpen
                                    ? Icons.lock_open
                                    : Icons.lock,
                              ),
label: Text(
isLoading
                                    ? 'Memproses...'
                                    : (isValveOpen ? 'Mati' :
'Aktif'),
```

```
                                ),
style: ElevatedButton.styleFrom(
backgroundColor: isValveOpen
? Colors.orange[700]                                        :
Colors.blue[700],                                    padding:
const EdgeInsets.symmetric(
horizontal: 20,                                      vertical:
12,
                                  ),
                                ),
                              ),
                            ],
),
                          const SizedBox(height: 12),
Container(                                  padding: const
EdgeInsets.all(12),                           decoration:
BoxDecoration(                              color:
Colors.blue[50],                              borderRadius:
BorderRadius.circular(8),
border: Border.all(                                  color:
Colors.blue[200]!,                                 width: 1,
                            ),
),                            child:
Row(
children: [
                              Icon(
                                Icons.info_outline,
                              size: 16,   color:
                              Colors.blue[700],
                              ),
                              const SizedBox(width: 8),
                              Expanded(
                                child: Text(
                                  'Pompa akan terbuka otomatis
jika pH < 7.0 atau > 8.0',
style: TextStyle(
fontSize: 11,                                        color:
Colors.blue[900],
                                    ),
                                  ),
                                ),
                              ],
                            ),
                          ),
                        ],
                      ),
                    ),                    ),
const SizedBox(height: 16),
                // Card untuk Stok Cairan Asam dan Basa
Card(                   elevation: 3,
```

```
shape: RoundedRectangleBorder(
borderRadius: BorderRadius.circular(12),
                  ),                    child: Container(
padding: const EdgeInsets.all(16),
decoration: BoxDecoration(                        borderRadius:
BorderRadius.circular(12),                        gradient:
LinearGradient(                           colors:
[Colors.purple[50]!, Colors.purple[100]!],
                        begin: Alignment.topLeft,
end: Alignment.bottomRight,
                      ),                      ),
child: Column(                          crossAxisAlignment:
CrossAxisAlignment.start,                           children: [
Row(                              children: [ Container( padding:
const EdgeInsets.all(8), decoration: BoxDecoration(   color:
Colors.purple[700],
                                  borderRadius:
BorderRadius.circular(8),
),                                 child: const Icon(
Icons.science,
color: Colors.white,
size: 24,
                                ),
),                                 const SizedBox(width: 12),
const Text(                                  'Stok Cairan pH',
style: TextStyle(                                   fontSize:
16,                                fontWeight:
FontWeight.bold,                                    color:
Colors.black87,
                                ),
                              ),
                            ],
),
                        const SizedBox(height: 20),
Row(                              mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
children: [
                              _buildStokCairanGauge(
label: 'Cairan Asam',
percentage: stokAsamPercentage,
status: stokAsamStatus,
hasWarning: stokAsamWarning,
color: Colors.red,                                  icon:
Icons.remove_circle,
                              ),
                              _buildStokCairanGauge(
label: 'Cairan Basa',
percentage: stokBasaPercentage,
status: stokBasaStatus,
hasWarning: stokBasaWarning,
color: Colors.blue,                                  icon:
Icons.add_circle,
                              ),
```

```
                        ],
                ),
                        const SizedBox(height: 16),
                        Container(
                          padding: const EdgeInsets.all(12),
                         decoration: BoxDecoration(
                color: Colors.purple[50],
                            borderRadius:
BorderRadius.circular(8),
border: Border.all(                                color:
Colors.purple[200]!,                                width:
1,
                            ),
),                           child:
Row(
children: [
                             Icon(
                               Icons.info_outline,
size: 16,                                color:
Colors.purple[700],
                             ),
                             const SizedBox(width: 8),
Expanded(                                child: Text(
                               'Sistem akan memberi notifikasi
jika stok cairan menipis atau kritis',
style: TextStyle(                                    fontSize:
11,                                color:
Colors.purple[900],
                                ),
                              ),
                             ),
                           ],
                         ),
                        ),
                      ],
                    ),
                  ),
),
                const SizedBox(height: 16),
Card(                    elevation: 2,
shape: RoundedRectangleBorder(
borderRadius: BorderRadius.circular(12),
                  ),     child: Padding(       padding:
                const EdgeInsets.all(16.0),       child:
                Row(         children: [
                          Icon(
                            Icons.lightbulb_outline,
                color: Colors.amber[700],
                            size: 24,
),                           const SizedBox(width:
```

```
12),                          const Expanded(
child: Text(
                              'Tips: Periksa kualitas air secara
berkala untuk menjaga kesehatan ikan',
style: TextStyle(                                   fontSize: 12,
color: Colors.black54,
                                ),
                              ),
                            ),
                          ],
                        ),
                      ),
                    ),
                  ],
                ),
              ),
            ),
          ),
        ),
      ),
    );
  }

  Widget _buildGaugeIndicator({
required String label,
required double value,
required double minValue,
required double maxValue,
required String unit,     required
String status,     required bool
hasWarning,     bool isStatic =
false,
  }) {     return
Column(
children: [
        Stack( alignment:
          Alignment.topRight, children:
          [
            SizedBox(
              width: 100,
              height: 100,
              child:
              SfRadialGauge
              (   axes:
              <RadialAxis>[
              RadialAxis(
              minimum:
              minValue,
              maximum:
              maxValue,
              showLabels:
              false,
              showTicks:
              false,
```

```
                axisLineStyle
                :
                AxisLineStyle
                (
                thickness:
                0.15,
                cornerStyle:
                CornerStyle.b
                othCurve,
                color:
                Colors.grey[3
                00],
                thicknessUnit
                :
                GaugeSizeUnit
                .factor,
                      ),
                      pointers: <GaugePointer>[
RangePointer(                           value: value,
cornerStyle: CornerStyle.bothCurve,
width: 0.15,                          sizeUnit:
GaugeSizeUnit.factor,                       enableAnimation:
true,                           animationDuration: 1200,
animationType: AnimationType.ease,
gradient: SweepGradient(                          colors:
_getGradientColors(label, hasWarning),
stops: const <double>[0.0, 0.5, 1.0],
                        ),
                    ),
],
                      annotations: <GaugeAnnotation>[
GaugeAnnotation(                          angle: 90,
positionFactor: 0.5,
widget: Column(
                          mainAxisSize: MainAxisSize.min,
children: [                               Text(
value.toStringAsFixed(1),
style: const TextStyle(                             fontSize:
16,
                                fontWeight: FontWeight.bold,
          color: Colors.black87,
                             ),
                            ),
                            if
                (unit.isNotEmpty)
                Text(                         unit,
                                  style: TextStyle(
fontSize: 10,                                       color:
Colors.grey[600],
                                ),
                              ),
```

```
                        ],
                      ),
                    ),
                  ],
                ),
              ],
            ),
          ),
          if (hasWarning && !isStatic)
            Container(
              width: 20,
              height: 20,
              decoration: BoxDecoration(
                color: Colors.red,
                shape: BoxShape.circle,
                border: Border.all(color: Colors.white, width: 2),
              ),
              child: const Icon(
                Icons.warning,
                size: 12,
                color: Colors.white,
              ),
            ),
          if (isStatic)
            Container(
              padding: const EdgeInsets.symmetric(horizontal: 4, vertical: 2),
              decoration: BoxDecoration(
                color: Colors.grey[400],
                borderRadius: BorderRadius.circular(4),
              ),
              child: const Text(
                'Static',
                style: TextStyle(
                  fontSize: 8,
                  color: Colors.white,
                  fontWeight: FontWeight.bold,
                ),
              ),
            ),
        ],
      ),
      const SizedBox(height: 8),
      Text(
        label,
        style: const TextStyle(
          fontSize: 14,
          fontWeight: FontWeight.w600,
          color: Colors.black87,
        ),
      ),
      if (status.isNotEmpty && !isStatic)
        Container(
          margin: const EdgeInsets.only(top: 4),
          padding: const EdgeInsets.symmetric(horizontal: 8, vertical: 2),
          decoration: BoxDecoration(
            color: hasWarning ? Colors.red[50] : Colors.green[50],
            borderRadius: BorderRadius.circular(12),
            border: Border.all(
              color: hasWarning ?
```

```
Colors.red[200]! : Colors.green[200]!,                 width:
1,
              ),             ),             child:
Text(               status,               style:
TextStyle(                 fontSize: 10,
fontWeight: FontWeight.w600,                  color:
hasWarning ? Colors.red[700] :
Colors.green[700],
              ),
            ),
          ),
      ],
    );
  }

  List<Color> _getGradientColors(String label, bool hasWarning) {
    if (hasWarning) {
    return const [
    Color(0xFFEF5350),
        Color(0xFFE53935),
        Color(0xFFD32F2F),
      ];
    }      switch (label) {
case 'pH':          return
const [
Color(0xFF4FC3F7),
          Color(0xFF2196F3),
          Color(0xFF1976D2),
];       case 'Suhu':
return const [
Color(0xFFFF9800),
          Color(0xFFF57C00),
          Color(0xFFE65100),
];       case 'Turbidity':
return const [
Color(0xFF66BB6A),
          Color(0xFF4CAF50),
          Color(0xFF388E3C),
];       default:
return const [
Color(0xFF9E9E9E),
          Color(0xFF757575),
          Color(0xFF616161),
        ];
    }
  }

  // Widget untuk gauge stok cairan
Widget _buildStokCairanGauge({
```

```
required String label,     required
int percentage,   required String
status,   required bool hasWarning,
required Color color,   required
IconData icon,
  }) {
    // Tentukan warna berdasarkan
    percentage Color gaugeColor; if
    (hasWarning) {
      if (percentage <= 10) {
gaugeColor = Colors.red; // Kritis       }
else if (percentage <= 30) {
gaugeColor = Colors.orange; // Rendah
      } else {         gaugeColor =
Colors.yellow; // Menipis
      }     } else {       gaugeColor = color; // Normal (merah
untuk asam, biru untuk basa)     }      return Column(
children: [         Stack(           alignment:
Alignment.topRight,           children: [            SizedBox(
width: 120,               height: 120,              child:
SfRadialGauge(                axes: <RadialAxis>[
RadialAxis(                 minimum: 0,
maximum: 100,                   showLabels: false,
showTicks: false,                   axisLineStyle:
AxisLineStyle(                    thickness: 0.15,
cornerStyle: CornerStyle.bothCurve,                        color:
Colors.grey[300],                   thicknessUnit:
GaugeSizeUnit.factor,
                ),
                pointers: <GaugePointer>[
RangePointer(                    value:
percentage.toDouble(),                       cornerStyle:
CornerStyle.bothCurve,
                  width: 0.15,
          sizeUnit: GaugeSizeUnit.factor,
          enableAnimation: true,
                  animationDuration: 1200,
            animationType: AnimationType.ease,
            gradient: SweepGradient(
                    colors: hasWarning
                        ? [
                      Colors.red[400]!,
                      Colors.orange[600]!,
                      Colors.red[700]!,
                    ]
: [
color.withOpacity(0.6),
color,
color.withOpacity(0.8),
                    ],                          stops:
const <double>[0.0, 0.5, 1.0],
                  ),
```

```
                    ),
],
                  annotations: <GaugeAnnotation>[
GaugeAnnotation(                          angle: 90,
positionFactor: 0.1,                          widget: Column(
mainAxisSize: MainAxisSize.min,
children: [                               Icon(
icon,                                  size: 28,
color: hasWarning ? Colors.red[700]
: color,
                            ),
                            const SizedBox(height: 4),
                            Text(
                              '$percentage%',
style: TextStyle(                                    fontSize:
18,                                  fontWeight:
FontWeight.bold,                                  color:
hasWarning ?
Colors.red[700] : Colors.black87,
                                ),
                            ),
                          ],
                        ),
                      ),
                    ],
                  ),
                ],
              ), ), if (hasWarning)              Container(
width: 24,                  height: 24,
decoration: BoxDecoration(                color: Colors.red,
shape: BoxShape.circle,                  border:
Border.all(color: Colors.white, width:
2),
                ),
                child: const Icon(
Icons.warning,                  size:
14,                  color:
Colors.white,
                ),
              ),
          ],
),
        const SizedBox(height: 8),
Text(           label,           style:
const TextStyle(             fontSize:
13,             fontWeight:
FontWeight.w600,             color:
Colors.black87,
          ),
```

```
        ),
Container(
          margin: const EdgeInsets.only(top: 4),
padding: const EdgeInsets.symmetric(horizontal: 8, vertical:
2),
          decoration: BoxDecoration(             color: hasWarning
? Colors.red[50] : Colors.green[50],             borderRadius:
BorderRadius.circular(12),             border: Border.all(
color: hasWarning ? Colors.red[200]! : Colors.green[200]!,
width: 1,
            ), ),
          child:
          Text(
            status, style:
            TextStyle(
            fontSize: 10,
              fontWeight: FontWeight.w600,
            color: hasWarning ? Colors.red[700] :
Colors.green[700],
            ),
          ),
        ),
      ],
    );
  }


  Future<void> _toggleValve() async {
bool? confirm = await showDialog<bool>(
context: context,       builder:
(BuildContext context) {         return
AlertDialog(           title: Row(
children: [
              Icon(
                Icons.warning_rounded,
color: Colors.orange[700],
size: 28,              ),
const SizedBox(width: 12),
const Text('Konfirmasi'),
            ],
),           content:
Text(
isValveOpen
                ? 'Apakah Anda yakin ingin mematikan pompa
dosing?'
                : 'Apakah Anda yakin ingin menyalakan pompa dosing
secara manual?',           ),           actions: [
TextButton(
              onPressed: () => Navigator.of(context).pop(false),
child: const Text('Batal'),
            ),
```

```
            ElevatedButton(     onPressed: () =>
          Navigator.of(context).pop(true),     style:
          ElevatedButton.styleFrom(       backgroundColor:
          isValveOpen
                    ? Colors.orange[700]
                    : Colors.red[700],
              ),    child: const Text('Ya,
            Lanjutkan'),
            ),
          ],
        );
      },
    );      if (confirm != true)
return;
     setState(() {
isLoading = true;
    });      try {       await Future.delayed(const
Duration(seconds: 2));
       if (mounted) {
setState(() {           isValveOpen =
!isValveOpen;           isLoading =
false;
        });

        ScaffoldMessenger.of(context).showSnackBar(
SnackBar(             content: Text(
isValveOpen
                  ? '✓ Pompa berhasil dinyalakan'
                  : '✓ Pompa berhasil dimatikan',
            ),
            backgroundColor: Colors.green[700],
behavior: SnackBarBehavior.floating,
duration: const Duration(seconds: 3),
          ),
        );
      }     } catch (e) {
if (mounted) {
setState(() {
isLoading = false;
        });

        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar( content: Text('❌ Gagal mengontrol
            pompa: $e'), backgroundColor: Colors.red[700],
            behavior: SnackBarBehavior.floating, duration:
            const Duration(seconds: 3),
          ),
        );
      }
```

```
    }
  }
   @override
void dispose() {
    // Cancel semua Firebase listeners
    _tempListener?.cancel();
    _turbidityListener?.cancel();
    _phListener?.cancel();
    _stokAsamListener?.cancel();
    _stokBasaListener?.cancel();
    _phControlLogListener?.cancel();
    _acidPumpListener?.cancel();
_basePumpListener?.cancel();     super.dispose();
  }
}
```