LAMPIRAN

```
/*
PROGRAM KOMPLET – PZEM 3 PHASE (JALUR TERPISAH)
Sudah diperbaiki agar:
✓ PZEM R (SoftwareSerial)
✓ PZEM S (Serial1)
✓ PZEM T (Serial2)
terbaca semua dengan stabil.
*/

#include <Wire.h>
#include <MPU6050.h>
#include <TFT_22_ILI9225.h>
#include <max6675.h>
#include <PZEM004Tv30.h>
#include <SoftwareSerial.h>
#include <SPI.h>

#define LCD_RST   A4
#define LCD_RS    A3
#define LCD_CS    A5
#define LCD_SDI   A2
```

```
#define LCD_CLK   A1
#define LCD_LED   0
#define LCD_BRIGHTNESS 200

#define MAX6675_SCK 5
#define MAX6675_CS  6
#define MAX6675_SO  7

// ----- PZEM -----
#define PZEM1_RX 10
#define PZEM1_TX 11

// ----- KONSTANTA -----
const float HIGH_TEMP_THRESHOLD = 60.0;
const float VERY_HIGH_TEMP_THRESHOLD = 80.0;
const float MEDIUM_VIBRATION_THRESHOLD = 1.0;
const float HIGH_VIBRATION_THRESHOLD = 1.5;
const float LOW_CURRENT_THRESHOLD = 15.0;
const float NORMAL_CURRENT_THRESHOLD = 25.0;
const float HIGH_CURRENT_THRESHOLD = 30.0;

// ===== OBJEK SENSOR =====
```

```
MPU6050 mpu;

MAX6675 thermocouple(MAX6675_SCK, MAX6675_CS, MAX6675_SO);

TFT_22_ILI9225 tft(LCD_RST, LCD_RS, LCD_CS, LCD_SDI, LCD_CLK,
LCD_LED);


// PZEM

SoftwareSerial pzem1Serial(PZEM1_RX, PZEM1_TX);

PZEM004Tv30 pzem1(pzem1Serial);


PZEM004Tv30 pzem2(&Serial1);

PZEM004Tv30 pzem3(&Serial2);


float voltageR = 0, currentR = 0;

float voltageS = 0, currentS = 0;

float voltageT = 0, currentT = 0;


float temperature = 0;

float vibrationX = 0, vibrationY = 0, vibrationZ = 0, totalVibration = 0;


// ===== FUZZY RESULT =====

struct FuzzyResult {

String motorStatus;
```

```
float faultLevel;
};
FuzzyResult fuzzyResult;

// ===== ESP CONFIG =====
String wifiStatus = "WiFi: ?";
String dataSendStatus = "Data: ?";
const String WIFI_SSID = "POCO F4";
const String WIFI_PASS = "tegar123";
const String SERVER_HOST = "10.161.83.137";
const String SERVER_PORT = "443";

unsigned long lastPZEMRead = 0;
unsigned long pzemInterval = 400;  // delay baca PZEM (penting)

/*
================================================================
SETUP
================================================================ */
void setup() {
```

```
Serial.begin(115200);
Serial3.begin(115200);

tft.begin();
tft.setOrientation(3);
tft.clear();
tft.setFont(Terminal12x16);
tft.drawText(10, 10, "Inisialisasi...", COLOR_WHITE);

Wire.begin();
mpu.initialize();
if (!mpu.testConnection()) {
tft.drawText(10, 30, "MPU ERROR!", COLOR_RED);
while (1);
}

konfigurasiESP();
delay(500);
}

/*
================================================================
===
```

```
LOOP UTAMA
============================================================
=== */
void loop() {

readAllSensorData();
processFuzzyLogic();
checkESPStatus();
displayData();
sendDataToESP();

delay(1000);
}

/*
============================================================
===
PZEM FIX
============================================================
=== */
void readPZEMData() {

if (millis() - lastPZEMRead < pzemInterval) return;
lastPZEMRead = millis();
```

```
// ===== R =====
float vR = pzem1.voltage();
float iR = pzem1.current();
if (!isnan(vR)) voltageR = vR;
if (!isnan(iR)) currentR = iR;

// small delay to avoid bus overlap
delay(30);

// ===== S =====
float vS = pzem2.voltage();
float iS = pzem2.current();
if (!isnan(vS)) voltageS = vS;
if (!isnan(iS)) currentS = iS;

delay(30);

// ===== T =====
float vT = pzem3.voltage();
float iT = pzem3.current();
if (!isnan(vT)) voltageT = vT;
```

```
if (!isnan(iT)) currentT = iT;


}


/*
==========================================================
=== */


void konfigurasiESP() {

String configString = "CONFIG:" + WIFI_SSID + "," + WIFI_PASS + "," +
SERVER_HOST + "," + SERVER_PORT;

Serial3.println(configString);

}


void readAllSensorData() {

temperature = thermocouple.readCelsius();

readVibrationData();

readPZEMData();

}


void readVibrationData() {

int16_t ax, ay, az;

mpu.getAcceleration(&ax, &ay, &az);
```

```
vibrationX = ax / 16384.0;

vibrationY = ay / 16384.0;

vibrationZ = az / 16384.0;

totalVibration = sqrt(vibrationX*vibrationX + vibrationY*vibrationY + vibrationZ*vibrationZ);

}


/*
================================================================
===

FUZZY

================================================================
=== */

float triangularMF(float x, float a, float b, float c) {

if (x <= a || x >= c) return 0.0;

if (x == b) return 1.0;

if (x < b) return (x - a) / (b - a);

return (c - x) / (c - b);

}


float trapezoidalMF(float x, float a, float b, float c, float d) {

if (x <= a || x >= d) return 0.0;

if (x >= b && x <= c) return 1.0;
```

```
if (x < b) return (x - a) / (b - a);
return (d - x) / (d - c);
}

void processFuzzyLogic() {

float maxCurrent = max(currentR, max(currentS, currentT));

 // membership functions (rendah, normal, tinggi) untuk Suhu
  float tempLow = trapezoidalMF(temperature, 0, 0, 20, 50);
  float tempNormal = triangularMF(temperature, 40, 60, 75);
  float tempHigh = trapezoidalMF(temperature, 70, 100, 100, 120);


float vibrationNormal = trapezoidalMF(totalVibration, 0, 0, 0.8, 1.0);
float vibrationHigh   = trapezoidalMF(totalVibration, 0.8, 1.0, 3.0, 3.0);

float currentLow    = trapezoidalMF(maxCurrent, 0, 0, 15, 20);
float currentNormal = triangularMF(maxCurrent, 15, 25, 30);
float currentHigh   = trapezoidalMF(maxCurrent, 25, 30, 35, 35);
```

```
  // Initialize output membership
  float motorUnload = 0;
  float motorBaik = 0;
  float motorCek = 0;
  float motorFault = 0;

 // Rule 1: Rendah, Normal, Rendah → Motor Unload
  motorUnload = max(motorUnload, min(tempLow, min(vibrationNormal, currentLow)));

  // Rule 2: Rendah, Normal, Normal → Motor Baik
  motorBaik = max(motorBaik, min(tempLow, min(vibrationNormal, currentNormal)));

  // Rule 3: Rendah, Normal, Tinggi → Motor Cek
  motorCek = max(motorCek, min(tempLow, min(vibrationNormal, currentHigh)));

  // Rule 4: Rendah, Getar, Rendah → Motor Cek
  motorCek = max(motorCek, min(tempLow, min(vibrationHigh, currentLow)));

  // Rule 5: Rendah, Getar, Normal → Motor Cek
  motorCek = max(motorCek, min(tempLow, min(vibrationHigh, currentNormal)));
```

```
// Rule 6: Rendah, Getar, Tinggi → Motor Fault
motorFault = max(motorFault, min(tempLow, min(vibrationHigh, currentHigh)));

// Rule 7: Normal, Normal, Rendah → Motor Unload
motorUnload = max(motorUnload, min(tempNormal, min(vibrationNormal, currentLow)));

// Rule 8: Normal, Normal, Normal → Motor Baik
motorBaik = max(motorBaik, min(tempNormal, min(vibrationNormal, currentNormal)));

// Rule 9: Normal, Normal, Tinggi → Motor Fault
motorFault = max(motorFault, min(tempNormal, min(vibrationNormal, currentHigh)));

// Rule 10: Normal, Getar, Rendah → Motor Cek
motorCek = max(motorCek, min(tempNormal, min(vibrationHigh, currentLow)));

// Rule 11: Normal, Getar, Normal → Motor Cek
motorCek = max(motorCek, min(tempNormal, min(vibrationHigh, currentNormal)));
```

```
// Rule 12: Normal, Getar, Tinggi → Motor Fault
motorFault = max(motorFault, min(tempNormal, min(vibrationHigh, currentHigh)));

// Rule 13: Tinggi, Normal, Rendah → Motor Unload
motorUnload = max(motorUnload, min(tempHigh, min(vibrationNormal, currentLow)));

// Rule 14: Tinggi, Normal, Normal → Motor Cek
motorCek = max(motorCek, min(tempHigh, min(vibrationNormal, currentNormal)));

// Rule 15: Tinggi, Normal, Tinggi → Motor Fault
motorFault = max(motorFault, min(tempHigh, min(vibrationNormal, currentHigh)));

// Rule 16: Tinggi, Getar, Rendah → Motor Cek
motorCek = max(motorCek, min(tempHigh, min(vibrationHigh, currentLow)));

// Rule 17: Tinggi, Getar, Normal → Motor Cek
motorCek = max(motorCek, min(tempHigh, min(vibrationHigh, currentNormal)));
```

```
  // Rule 18: Tinggi, Getar, Tinggi → Motor Fault
  motorFault = max(motorFault, min(tempHigh, min(vibrationHigh, currentHigh)));


float maxOutput = max(motorUnload, max(motorBaik, max(motorCek, motorFault)));


if (maxOutput == motorUnload) {
fuzzyResult.motorStatus = "Unload";
fuzzyResult.faultLevel = 0.2;
}
else if (maxOutput == motorBaik) {
fuzzyResult.motorStatus = "Baik";
fuzzyResult.faultLevel = 0.0;
}
else if (maxOutput == motorCek) {
fuzzyResult.motorStatus = "Cek";
fuzzyResult.faultLevel = 0.5;
}
else {
fuzzyResult.motorStatus = "Fault";
fuzzyResult.faultLevel = 1.0;
}
```

```
}

/*
================================================================
===
LCD
================================================================
=== */
void displayData() {
tft.drawText(10, 10, "Suhu: " + String(temperature, 1) + " C",
COLOR_YELLOW);
tft.drawText(10, 30, "Getaran: " + String(totalVibration, 2) + " g",
COLOR_CYAN);

float maxCurrent = max(currentR, max(currentS, currentT));
tft.drawText(10, 50, "Arus Max: " + String(maxCurrent, 2) + " A",
COLOR_GREEN);

tft.drawText(10, 70, "Status: " + fuzzyResult.motorStatus, COLOR_RED);
tft.drawText(10, 90, "Level: " + String(fuzzyResult.faultLevel * 100, 0) + "%",
COLOR_RED);

tft.drawText(10, 110, wifiStatus, COLOR_WHITE);
tft.drawText(10, 130, dataSendStatus, COLOR_WHITE);
}
```

```
/*
================================================================
===

ESP SEND

================================================================
=== */

void checkESPStatus() {

while (Serial3.available()) {

String msg = Serial3.readStringUntil('\n');

msg.trim();

if (msg == " ") continue;


if (msg.startsWith("WIFI_STATUS:CONNECTED")) wifiStatus = "WiFi:
Connected";

else if (msg.startsWith("WIFI_STATUS:DISCONNECTED")) wifiStatus =
"WiFi: Disconnected";

else if (msg.indexOf("Berhasil Dikirim") != -1) dataSendStatus = "Data: Sent
OK";

else if (msg.indexOf("Error on HTTP") != -1) dataSendStatus = "Data: Send Fail";




}

}
```

```
void sendDataToESP() {

  // Mencegah data NaN dikirim
  if (isnan(temperature)) return;

  String dataToSend =
    "suhu=" + String(temperature,1) +
    "&vib=" + String(totalVibration,3) +
    "&iR=" + String(currentR,3) +
    "&iS=" + String(currentS,3) +
    "&iT=" + String(currentT,3) +
    "&status=" + fuzzyResult.motorStatus +
    "&level=" + String(fuzzyResult.faultLevel,2);

  Serial.println("SEND: " + dataToSend);   // sangat penting untuk diagnosa
  Serial3.println(dataToSend);
}
```