LAMPIRAN – LAMPIRAN

Lampiran 1 Source Code Node 1

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>

// ====== KONFIG ======
const char* WIFI_SSID = "NYOMAN SUKARTINI";
const char* WIFI_PASS = "33355555";
const char* MQTT_HOST = "192.168.0.106";
const uint16_t MQTT_PORT = 1883;
const char* MQTT_USER = "admin";
const char* MQTT_PASSWD = "admin123";

// ====== MQTT TOPIC ======
#define TOPIC_TEMP "env/temp"
#define TOPIC_HUM  "env/hum"
#define TOPIC_MQ3  "env/mq3"

// ====== PIN ======
#define PIN_DHT   D4   // GPIO2
#define DHTTYPE   DHT22
#define PIN_MQ3A0 A0

WiFiClient espClient;
PubSubClient mqtt(espClient);
DHT dht(PIN_DHT, DHTTYPE);

unsigned long lastPub = 0;

// ====== WIFI ======
void connectWiFi() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASS);

  Serial.print("[WiFi] Connecting");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\n[WiFi] Connected");
  Serial.print("[WiFi] IP: ");
  Serial.println(WiFi.localIP());
}
```

```
// ====== MQTT ======
void connectMQTT() {
  mqtt.setServer(MQTT_HOST, MQTT_PORT);
  while (!mqtt.connected()) {
    String cid = "node1-" + String(ESP.getChipId(), HEX);
    Serial.print("[MQTT] Connecting... ");
    if (mqtt.connect(cid.c_str(), MQTT_USER, MQTT_PASSWD,
                     "stat/node1", 1, true, "offline")) {
      mqtt.publish("stat/node1", "online", true);
      Serial.println("OK");
    } else {
      Serial.println("FAILED");
      delay(1000);
    }
  }
}

// ====== SETUP ======
void setup() {
  Serial.begin(115200);
  Serial.println();
  Serial.println("=== NODE-1 START ===");

  connectWiFi();
  connectMQTT();
  dht.begin();

  Serial.println("Sensor DHT22 & MQ3 siap.");
  Serial.println("====================");
}

// ====== LOOP ======
void loop() {
  if (WiFi.status() != WL_CONNECTED) connectWiFi();
  if (!mqtt.connected()) connectMQTT();
  mqtt.loop();

  unsigned long now = millis();
  if (now - lastPub > 2000) { // update tiap 2 detik
    lastPub = now;

    float t = dht.readTemperature();
    float h = dht.readHumidity();
    int mq3 = analogRead(PIN_MQ3A0);

    // ====== SERIAL MONITOR ======
```

```
    Serial.println("----- SENSOR UPDATE -----");

    if (!isnan(t)) {
      Serial.print("Suhu      : ");
      Serial.print(t, 2);
      Serial.println(" °C");
    } else {
      Serial.println("Suhu      : ERROR");
    }

    if (!isnan(h)) {
      Serial.print("Kelembaban: ");
      Serial.print(h, 1);
      Serial.println(" %");
    } else {
      Serial.println("Kelembaban: ERROR");
    }

    Serial.print("MQ3 (ADC) : ");
    Serial.println(mq3);
    Serial.println("-------------------------");

    // ====== MQTT PUBLISH ======
    if (!isnan(t)) {
      char buf[16];
      dtostrf(t, 0, 2, buf);
      mqtt.publish(TOPIC_TEMP, buf, true);
    }

    if (!isnan(h)) {
      char buf[16];
      dtostrf(h, 0, 2, buf);
      mqtt.publish(TOPIC_HUM, buf, true);
    }

    {
      char buf[16];
      snprintf(buf, sizeof(buf), "%d", mq3);
      mqtt.publish(TOPIC_MQ3, buf, true);
    }
  }
}
```

Lampiran 2 Source Code Node 2

```
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <UniversalTelegramBot.h>

// ====== KONFIG WIFI & MQTT ======
const char* WIFI_SSID = "NYOMAN SUKARTINI";
const char* WIFI_PASS = "33355555";
const char* MQTT_HOST = "192.168.0.106";
const uint16_t MQTT_PORT = 1883;
const char* MQTT_USER = "admin";
const char* MQTT_PASSWD = "admin123";

// ====== KONFIG TELEGRAM ======
const char* BOT_TOKEN =
"7622194785:AAHgIkbwqAhC23S9kX8KWXUBb2zCJrupPUU";
const char* CHAT_ID = "8152895238";

// ====== TOPIK MQTT ======
#define TOPIC_TEMP "env/temp"
#define TOPIC_HUM "env/hum"
#define TOPIC_MQ3 "env/mq3"      // raw ADC from Node-1
#define TOPIC_MQ135 "env/mq135"  // raw ADC from Node-3
#define TOPIC_STAT_FAN "stat/fan"
#define TOPIC_STAT_LAMP "stat/lamp"
#define TOPIC_CMD_FAN "cmd/fan"
#define TOPIC_CMD_LAMP "cmd/lamp"
#define TOPIC_AVAIL_NODE3 "avail/node3"
#define TOPIC_SYNC_REQ "cmd/node3/sync"

// ====== OLED ======
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// ====== PIN ======
#define PIN_BUZZER D5  // aktif-LOW (via transistor)
#define PIN_SDA D2
#define PIN_SCL D1
#define PIN_MQ135A0 A0  // (unused now, kept)
```

```
// Default thresholds - initialized in setup from old ADC default
350
float TH_TEMP = 30.0;  // °C (unchanged)
int TH_MQ3_ADC;
int TH_MQ135_ADC;
int REL_MQ3_ADC;
int REL_MQ135_ADC;

// ====== HISTERESIS (persen dari ambang untuk rilis OFF)
const float HYS_RATIO = 0.80;  // 80% dari ambang untuk mematikan
otomatis


// ====== MODE AUTO/MANUAL (default: AUTO saat start) ======
bool autoFanEnabled = true;
bool autoLampEnabled = true;

// ====== STATE CACHE ======
float g_temp = NAN, g_hum = NAN;
int g_mq3_adc = -1, g_mq135_adc = -1;  // raw ADC values from
topics
String g_fan = "OFF", g_lamp = "OFF";
String g_node3_avail = "unknown";

// previous states to detect changes (for buzzer trigger)
String prev_g_fan = "UNKNOWN";
String prev_g_lamp = "UNKNOWN";

// ====== ANTI-SPAM ALERT ======
unsigned long lastAlertTemp = 0;
unsigned long lastAlertMQ3 = 0;
unsigned long lastAlertMQ135 = 0;
const unsigned long ALERT_COOLDOWN_MS = 60UL * 1000UL;  // 60s

// ====== TELEGRAM POLLING ======
unsigned long lastPoll = 0;
const unsigned long POLL_MS = 2000;  // responsif

// ====== DISPLAY ======
unsigned long lastReadMQ135 = 0;
unsigned long lastDisp = 0;

// ====== WAIT STATUS ======
const unsigned long STATUS_WAIT_TIMEOUT_MS = 4000;
const unsigned long STATUS_WAIT_STEP_MS = 10;
```

```
volatile bool mq135Updated = false;

// ====== NET/CLIENT ======
WiFiClientSecure clientTCP;  // Telegram
WiFiClient espClient;        // MQTT
PubSubClient mqtt(espClient);
UniversalTelegramBot bot(BOT_TOKEN, clientTCP);

// ====== UTIL ======
void buzzerOff() {
  digitalWrite(PIN_BUZZER, LOW);  // aktif-LOW -> HIGH = OFF
}

// Play buzzer pulses for mode (blocking short pulses)
// mode: 1 => lamp only (1 pulse), 2 => fan only (2 pulses), 3 =>
both (3 pulses)
void playBuzzerForMode(int mode) {
  if (mode <= 0) return;
  int pulses = 0;
  if (mode == 1) pulses = 1;
  else if (mode == 2) pulses = 2;
  else if (mode == 3) pulses = 3;

  for (int i = 0; i < pulses; ++i) {
    digitalWrite(PIN_BUZZER, HIGH);  // ON (aktif-LOW)
    delay(80);
    digitalWrite(PIN_BUZZER, LOW);  // OFF
    delay(80);
  }
}

// Determine buzzer mode from current fan/lamp states
// return 0..3 (0: off, 1: lamp only, 2: fan only, 3: both)
int buzzerModeFromState() {
  bool lampOn = g_lamp.equalsIgnoreCase("ON");
  bool fanOn = g_fan.equalsIgnoreCase("ON");
  if (!lampOn && !fanOn) return 0;
  if (lampOn && !fanOn) return 1;
  if (!lampOn && fanOn) return 2;
  return 3;
}

// --------- DRAW SCREEN (OLED) ---------
void drawScreen() {
  display.clearDisplay();
```

```
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);

  // Baris 1: T, H, B
  display.setCursor(0, 0);
  display.print("T:");
  if (!isnan(g_temp)) display.print(g_temp, 1);
  else display.print("--");
  display.print("C  ");

  display.print("H:");
  if (!isnan(g_hum)) display.print(g_hum, 0);
  else display.print("--");
  display.print("%  ");

  display.print("B:");
  int bm = buzzerModeFromState();
  if (bm == 0) display.print("-");
  else {
    display.print("M");
    display.print(bm);
  }

  // Baris 2: MQ3
  display.setCursor(0, 12);
  display.print("MQ3:");
  if (g_mq3_adc >= 0) {
    display.print(g_mq3_adc);  // integer display
  } else display.print("--");

  // Baris 3: MQ135
  display.setCursor(0, 24);
  display.print("MQ135:");
  if (g_mq135_adc >= 0) {
    display.print(g_mq135_adc);
  } else display.print("--");

  // Baris 4: Fan (A/M)
  display.setCursor(0, 36);
  display.print("Fan:");
  display.print(g_fan);
  display.print(autoFanEnabled ? "(A)" : "(M)");

  // Baris 5: Lamp (A/M)
  display.setCursor(0, 48);
  display.print("Lamp:");
```

```
  display.print(g_lamp);
  display.print(autoLampEnabled ? "(A)" : "(M)");

  display.display();
}

// --------- Alerts with wording (no buzzer here) ---------
void sendAlertOnce(const String& text, unsigned long& lastTs) {
  unsigned long now = millis();
  if (now - lastTs >= ALERT_COOLDOWN_MS) {
    bot.sendMessage(CHAT_ID, text, "");
    lastTs = now;
  }
}

// Kirim CMD + minta Node-3 publish state langsung
void sendCmdAndSync(const char* cmdTopic, const char* payload) {
  mqtt.publish(cmdTopic, payload);
  mqtt.publish(TOPIC_SYNC_REQ, "1");
}

bool waitForStatus(String& ref, const char* expected, unsigned
long timeout_ms) {
  unsigned long t0 = millis();
  String exp = String(expected);
  exp.toUpperCase();
  while (millis() - t0 < timeout_ms) {
    mqtt.loop();
    if (ref.equalsIgnoreCase(exp)) return true;
    delay(STATUS_WAIT_STEP_MS);
    yield();
  }
  return false;
}

void publishCmd(const char* topic, const char* payload) {
  mqtt.publish(topic, payload);
}

// ====== OTOMASI: kipas/lampu berbasis ambang + histeresis ======
void automationControl() {
  if (!mq135Updated) return;
  mq135Updated = false;

  Serial.println("=== DEBUG: automationControl() ===");
  Serial.print("AutoLampEnabled: ");
```

```
  Serial.println(autoLampEnabled);
  Serial.print("AutoFanEnabled : ");
  Serial.println(autoFanEnabled);
  Serial.print("Suhu : ");
  Serial.println(g_temp);
  Serial.print("MQ3 ADC : ");
  Serial.println(g_mq3_adc);
  Serial.print("MQ135 ADC : ");
  Serial.println(g_mq135_adc);
  Serial.println("==============================");

  // --- SUHU TINGGI -> KIPAS ON ---
  if (autoFanEnabled && !isnan(g_temp)) {
    if (g_temp >= TH_TEMP && !g_fan.equalsIgnoreCase("ON")) {
      sendCmdAndSync(TOPIC_CMD_FAN, "ON");
      waitForStatus(g_fan, "ON", STATUS_WAIT_TIMEOUT_MS);
      sendAlertOnce("📎 Otomasi: Suhu tinggi (" + String(g_temp,
1) + "°C), Kipas ON.", lastAlertTemp);
    } else if (g_temp < TH_TEMP && g_fan.equalsIgnoreCase("ON")) {
      sendCmdAndSync(TOPIC_CMD_FAN, "OFF");
      waitForStatus(g_fan, "OFF", STATUS_WAIT_TIMEOUT_MS);
      sendAlertOnce("📎 Otomasi: Suhu normal (" + String(g_temp,
1) + "°C), Kipas OFF.", lastAlertTemp);
    }
  }

  // --- MQ3 (ALKOHOL) -> LAMPU ON ---
  if (autoLampEnabled && g_mq3_adc >= 0) {
    if (g_mq3_adc >= TH_MQ3_ADC && !g_lamp.equalsIgnoreCase("ON"))
{
      sendCmdAndSync(TOPIC_CMD_LAMP, "ON");
      // waitForStatus(g_lamp, "ON", STATUS_WAIT_TIMEOUT_MS);
      sendAlertOnce("🍺 Otomasi: Alkohol terdeteksi (ADC=" +
String(g_mq3_adc) + "), Lampu ON.", lastAlertMQ3);
    }
  }

  // --- MQ135 (ASAP/GAS) -> LAMPU ON ---
  if (autoLampEnabled && g_mq135_adc >= 0) {
    if (g_mq135_adc >= TH_MQ135_ADC &&
!g_lamp.equalsIgnoreCase("ON")) {
      sendCmdAndSync(TOPIC_CMD_LAMP, "ON");
      // waitForStatus(g_lamp, "ON", STATUS_WAIT_TIMEOUT_MS);
      sendAlertOnce("💨 Otomasi: Asap/gas terdeteksi (ADC=" +
String(g_mq135_adc) + "), Lampu ON.", lastAlertMQ135);
    }
```

```
  }

  // --- MATIKAN LAMPU jika semua kondisi aman (pakai nilai
rilis/hysteresis) ---
  if (autoLampEnabled) {
    if (g_mq3_adc < REL_MQ3_ADC && g_mq135_adc < REL_MQ135_ADC &&
!g_lamp.equalsIgnoreCase("OFF")) {
      sendCmdAndSync(TOPIC_CMD_LAMP, "OFF");
      waitForStatus(g_lamp, "OFF", STATUS_WAIT_TIMEOUT_MS);
    }
  }
}

// Terapkan otomasi & laporkan bila ada perubahan (menghormati
flag AUTO)
void applyAutomationAndReport(const char* reason) {
  bool fanWasOn = g_fan.equalsIgnoreCase("ON");
  bool lampWasOn = g_lamp.equalsIgnoreCase("ON");

  automationControl();

  bool fanNowOn = g_fan.equalsIgnoreCase("ON");
  bool lampNowOn = g_lamp.equalsIgnoreCase("ON");

  if (fanNowOn != fanWasOn) {
    String msg = String("🔧 ") + reason + (fanNowOn ? " → Kipas
dihidupkan (ON)" : " → Kipas dimatikan (OFF)");
    msg += " — mode=" + String(autoFanEnabled ? "AUTO" :
"MANUAL");
    msg += ", MQ135 ADC=" + String(g_mq135_adc);
    msg += ", TH=" + String(TH_MQ135_ADC);
    msg += ", rilis<" + String(REL_MQ135_ADC) + ".";
    bot.sendMessage(CHAT_ID, msg, "");
  }

  if (lampNowOn != lampWasOn) {
    String msg = String("🔧 ") + reason + (lampNowOn ? " → Lampu
dinyalakan (ON)" : " → Lampu dimatikan (OFF)");
    msg += " — mode=" + String(autoLampEnabled ? "AUTO" :
"MANUAL");
    msg += ", MQ3 ADC=" + String(g_mq3_adc);
    msg += ", TH=" + String(TH_MQ3_ADC);
    msg += ", rilis<" + String(REL_MQ3_ADC) + ".";
    bot.sendMessage(CHAT_ID, msg, "");
  }
}
```

```
// ====== STATUS /status ======
void sendStatus() {
  String msg = "📊 Status Sensor:\n";
  msg += "• Suhu: " + String(isnan(g_temp) ? -1 : g_temp, 1) + "
°C\n";
  msg += "• Kelembaban: " + String(isnan(g_hum) ? -1 : g_hum, 0) +
" %\n";
  msg += "• MQ3 (alkohol): " + String(g_mq3_adc < 0 ? -1 :
g_mq3_adc) + "\n";
  msg += "• MQ135 (gas): " + String(g_mq135_adc < 0 ? -1 :
g_mq135_adc) + "\n";
  msg += "• Kipas: " + g_fan + " (" + (autoFanEnabled ? "AUTO" :
"MANUAL") + ")\n";
  msg += "• Lampu: " + g_lamp + " (" + (autoLampEnabled ? "AUTO" :
"MANUAL") + ")\n";
  msg += "• Node3: " + g_node3_avail + "\n\n";

  msg += "⚙ Ambang (ADC):\n";
  msg += "• Suhu ≥ " + String(TH_TEMP, 1) + " °C\n";
  msg += "• MQ3 ≥ " + String(TH_MQ3_ADC) + " (rilis < " +
String(REL_MQ3_ADC) + ")\n";
  msg += "• MQ135 ≥ " + String(TH_MQ135_ADC) + " (rilis < " +
String(REL_MQ135_ADC) + ")\n";

  bot.sendMessage(CHAT_ID, msg, "");
}


// ---------------- TELEGRAM ----------------
void handleTelegramMessages(int numNewMessages) {
  for (int i = 0; i < numNewMessages; i++) {
    String chat_id = bot.messages[i].chat_id;
    String text = bot.messages[i].text;

    if (chat_id != CHAT_ID) {
      bot.sendMessage(chat_id, "Maaf, tidak diizinkan.", "");
      continue;
    }

    if (text == "/start" || text == "/help") {
      String help = "Perintah:\n";
      help += "/status - Lihat semua sensor & status + ambang\n";
      help += "/fan_on, /fan_off - Kendalikan kipas (MASUK mode
MANUAL)\n";
```

```
      help += "/lamp_on, /lamp_off - Kendalikan lampu (MASUK mode
MANUAL)\n";
      help += "/fan_auto - Aktifkan mode otomatis kipas\n";
      help += "/lamp_auto - Aktifkan mode otomatis lampu\n";
      help += "/set_th_temp <nilai> - Ambang suhu (°C)\n";
      help += "/set_th_mq3 <ADC>\n";
      help += "/set_th_mq135 <ADC>\n";
      bot.sendMessage(CHAT_ID, help, "");
    } else if (text == "/status") {
      mqtt.publish(TOPIC_SYNC_REQ, "1");
      unsigned long t0 = millis();
      while (millis() - t0 < 600) {
        mqtt.loop();
        delay(10);
      }
      sendStatus();
    }

    // ---------- FAN MANUAL ----------
    else if (text == "/fan_on") {
      autoFanEnabled = false;  // masuk MANUAL
      sendCmdAndSync(TOPIC_CMD_FAN, "ON");
      bool ok = waitForStatus(g_fan, "ON",
STATUS_WAIT_TIMEOUT_MS);
      if (ok) bot.sendMessage(CHAT_ID, "☑ Kipas ON (mode MANUAL).
Gunakan /fan_auto untuk kembali otomatis.", "");
      sendStatus();
      if (!ok) bot.sendMessage(CHAT_ID, "⚠ Timeout menunggu
konfirmasi KIPAS.", "");
    } else if (text == "/fan_off") {
      autoFanEnabled = false;  // masuk MANUAL
      sendCmdAndSync(TOPIC_CMD_FAN, "OFF");
      bool ok = waitForStatus(g_fan, "OFF",
STATUS_WAIT_TIMEOUT_MS);
      if (ok) bot.sendMessage(CHAT_ID, "🛑 Kipas OFF (mode
MANUAL). Gunakan /fan_auto untuk kembali otomatis.", "");
      sendStatus();
      if (!ok) bot.sendMessage(CHAT_ID, "⚠ Timeout menunggu
konfirmasi KIPAS.", "");
    }

    // ---------- LAMP MANUAL ----------
    else if (text == "/lamp_on") {
      autoLampEnabled = false;  // masuk MANUAL
      sendCmdAndSync(TOPIC_CMD_LAMP, "ON");
```

```
      bool ok = waitForStatus(g_lamp, "ON",
STATUS_WAIT_TIMEOUT_MS);
      if (ok) bot.sendMessage(CHAT_ID, "☑ Lampu ON (mode MANUAL).
Gunakan /lamp_auto untuk kembali otomatis.", "");
      sendStatus();
      if (!ok) bot.sendMessage(CHAT_ID, "⚠ Timeout menunggu
konfirmasi LAMPU.", "");
    } else if (text == "/lamp_off") {
      autoLampEnabled = false;  // masuk MANUAL
      sendCmdAndSync(TOPIC_CMD_LAMP, "OFF");
      bool ok = waitForStatus(g_lamp, "OFF",
STATUS_WAIT_TIMEOUT_MS);
      if (ok) bot.sendMessage(CHAT_ID, "🛑 Lampu OFF (mode
MANUAL). Gunakan /lamp_auto untuk kembali otomatis.", "");
      sendStatus();
      if (!ok) bot.sendMessage(CHAT_ID, "⚠ Timeout menunggu
konfirmasi LAMPU.", "");
    }

    // ---------- ENABLE AUTO MODES ----------
    else if (text == "/fan_auto") {
      autoFanEnabled = true;
      bot.sendMessage(CHAT_ID, "🤖 Mode OTOMATIS KIPAS diaktifkan.
Menyesuaikan dengan ambang & nilai terbaru…", "");
      applyAutomationAndReport("Mode otomatis KIPAS diaktifkan");
      sendStatus();
    } else if (text == "/lamp_auto") {
      autoLampEnabled = true;
      bot.sendMessage(CHAT_ID, "🤖 Mode OTOMATIS LAMPU diaktifkan.
Menyesuaikan dengan ambang & nilai terbaru…", "");
      applyAutomationAndReport("Mode otomatis LAMPU diaktifkan");
      sendStatus();
    }

    // ---------- SET THRESHOLDS ----------
    else if (text.startsWith("/set_th_temp ")) {
      float v = text.substring(13).toFloat();
      if (v > 0 && v < 100) {
        TH_TEMP = v;
        bot.sendMessage(CHAT_ID, "Ambang suhu => " +
String(TH_TEMP, 1) + " °C", "");
        applyAutomationAndReport("Threshold DHT22 diubah");
      } else {
        bot.sendMessage(CHAT_ID, "Nilai tidak valid.", "");
      }
    }
```

```
    else if (text.startsWith("/set_th_mq3 ")) {
      int v = text.substring(12).toInt();  // ADC
      if (v >= 0 && v <= 1023) {
        TH_MQ3_ADC = v;
        REL_MQ3_ADC = TH_MQ3_ADC * HYS_RATIO;

        bot.sendMessage(CHAT_ID, "Ambang MQ3 => " +
String(TH_MQ3_ADC) + " (rilis < " + String(REL_MQ3_ADC) + ")",
"");

        applyAutomationAndReport("Threshold MQ3 diubah");
      } else {
        bot.sendMessage(CHAT_ID, "Nilai tidak valid (0–1023).",
"");
      }
    }

    else if (text.startsWith("/set_th_mq135 ")) {
      int v = text.substring(14).toInt();  // ADC
      if (v >= 0 && v <= 1023) {
        TH_MQ135_ADC = v;
        REL_MQ135_ADC = TH_MQ135_ADC * HYS_RATIO;

        bot.sendMessage(CHAT_ID, "Ambang MQ135 => " +
String(TH_MQ135_ADC) + " (rilis < " + String(REL_MQ135_ADC) + ")",
"");

        applyAutomationAndReport("Threshold MQ135 diubah");
      } else {
        bot.sendMessage(CHAT_ID, "Nilai tidak valid (0–1023).",
"");
      }
    }

    else {
      bot.sendMessage(CHAT_ID, "Perintah tidak dikenali. Gunakan
/help", "");
    }
  }
}

// ---------------- MQTT ----------------
void mqttCallback(char* topic, byte* payload, unsigned int len) {
  String msg;
  msg.reserve(len);
```

```
  for (unsigned int i = 0; i < len; i++) msg += (char)payload[i];
  msg.trim();

  String t = String(topic);
  String up = msg;
  up.toUpperCase();

  if (t == TOPIC_TEMP) {
    g_temp = msg.toFloat();
  } else if (t == TOPIC_HUM) {
    g_hum = msg.toFloat();
  } else if (t == TOPIC_MQ3) {
    g_mq3_adc = msg.toInt();
  } else if (t == TOPIC_MQ135) {
    g_mq135_adc = msg.toInt();
    Serial.printf("[NODE2] MQ135 update realtime: %d\n",
g_mq135_adc);
    mq135Updated = true;
  } else if (t == TOPIC_STAT_FAN) {
    // update state and trigger buzzer on change
    String newFan = up;
    Serial.printf("[MQTT] stat/fan  -> %s\n", newFan.c_str());
    if (!newFan.equalsIgnoreCase(g_fan)) {
      g_fan = newFan;
      // play buzzer according to new combined state
      int mode = buzzerModeFromState();
      playBuzzerForMode(mode);
    } else {
      g_fan = newFan;
    }
  } else if (t == TOPIC_STAT_LAMP) {
    // update state and trigger buzzer on change
    String newLamp = up;
    Serial.printf("[MQTT] stat/lamp -> %s\n", newLamp.c_str());
    if (!newLamp.equalsIgnoreCase(g_lamp)) {
      g_lamp = newLamp;
      int mode = buzzerModeFromState();
      playBuzzerForMode(mode);
    } else {
      g_lamp = newLamp;
    }
  } else if (t == TOPIC_AVAIL_NODE3) {
    g_node3_avail = msg;
  }
}
```

```
void connectWiFi() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(300);
    yield();
  }
  Serial.print("[WiFi] Connected. IP: ");
  Serial.println(WiFi.localIP());
}

void connectMQTT() {
  mqtt.setServer(MQTT_HOST, MQTT_PORT);
  mqtt.setCallback(mqttCallback);
  while (!mqtt.connected()) {
    String cid = "node2-tg-" + String(ESP.getChipId(), HEX);
    if (mqtt.connect(cid.c_str(), MQTT_USER, MQTT_PASSWD,
"stat/node2", 1, true, "offline")) {
      mqtt.publish("stat/node2", "online", true);
      mqtt.subscribe(TOPIC_TEMP);
      mqtt.subscribe(TOPIC_HUM);
      mqtt.subscribe(TOPIC_MQ3);
      mqtt.subscribe(TOPIC_MQ135);
      mqtt.subscribe(TOPIC_STAT_FAN);
      mqtt.subscribe(TOPIC_STAT_LAMP);
      mqtt.subscribe(TOPIC_AVAIL_NODE3);
      mqtt.subscribe("stat/#");
      mqtt.subscribe("avail/#");
      mqtt.publish(TOPIC_SYNC_REQ, "1");
      Serial.println("[MQTT] Connected & subscribed.");
    } else {
      delay(1000);
      yield();
    }
  }
}

// ---------------- SETUP / LOOP ----------------
void setup() {
  Serial.begin(115200);

  pinMode(PIN_BUZZER, OUTPUT);
  buzzerOff();  // ensure buzzer off at start

  Wire.begin(PIN_SDA, PIN_SCL);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
```

```
  display.clearDisplay();
  display.display();

  // Initialize thresholds (convert old ADC default 350)
  TH_MQ3_ADC = 350;
  TH_MQ135_ADC = 350;
  REL_MQ3_ADC = TH_MQ3_ADC * HYS_RATIO;
  REL_MQ135_ADC = TH_MQ135_ADC * HYS_RATIO;

  connectWiFi();

  clientTCP.setInsecure();  // gunakan root CA untuk produksi
  clientTCP.setTimeout(1500);
  bot.longPoll = 0;

  connectMQTT();

  // set prev states to current known states
  prev_g_fan = g_fan;
  prev_g_lamp = g_lamp;
}

void handleThresholdAlertsOnly() {
  bool alarm = false;
  if (!isnan(g_temp) && g_temp >= TH_TEMP) {
    sendAlertOnce("⚠ Suhu tinggi: " + String(g_temp, 1) + " °C",
lastAlertTemp);
    alarm = true;
  }
  if (g_mq3_adc >= 0 && g_mq3_adc >= TH_MQ3_ADC) {
    sendAlertOnce("🍺 Terdeteksi alkohol (MQ3): " +
String(g_mq3_adc), lastAlertMQ3);
    alarm = true;
  }
  if (g_mq135_adc >= 0 && g_mq135_adc >= TH_MQ135_ADC) {
    sendAlertOnce("💨 Terdeteksi gas (MQ135): " +
String(g_mq135_adc), lastAlertMQ135);
    alarm = true;
  }
  // NOTE: buzzer will NOT be triggered here — buzzer only reacts
to fan/lamp state changes
}

void loop() {
  if (WiFi.status() != WL_CONNECTED) connectWiFi();
  if (!mqtt.connected()) connectMQTT();
```

```
  mqtt.loop();
  mqtt.loop();

  unsigned long now = millis();

  // Otomasi kontrol kipas/lampu (berbasis ambang + histeresis)
  static unsigned long lastAuto = 0;
  if (millis() - lastAuto > 1000) {  // tiap 1 detik
    lastAuto = millis();
    automationControl();
  }

  // Notifikasi alert (tanpa mempengaruhi otomasi)
  static unsigned long lastAlertCheck = 0;
  if (millis() - lastAlertCheck > 1000) {  // 1 detik
    lastAlertCheck = millis();
    handleThresholdAlertsOnly();
  }

  // Polling Telegram
  if (now - lastPoll > POLL_MS) {
    lastPoll = now;
    int n = bot.getUpdates(bot.last_message_received + 1);
    while (n) {
      handleTelegramMessages(n);
      mqtt.loop();
      delay(10);
      n = bot.getUpdates(bot.last_message_received + 1);
    }
  }

  // Refresh OLED
  if (now - lastDisp > 500) {
    lastDisp = now;
    drawScreen();
  }
}
```

Lampiran 3 Source Code Node 3

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

// ====== KONFIG ======
const char* WIFI_SSID = "NYOMAN SUKARTINI";
const char* WIFI_PASS = "33355555";
const char* MQTT_HOST = "192.168.0.106";
const uint16_t MQTT_PORT = 1883;
const char* MQTT_USER   = "admin";
const char* MQTT_PASSWD = "admin123";

// ==== PIN Relay (NodeMCU/WeMos D1 mini) ====
// D5 = GPIO14, D6 = GPIO12
#define PIN_FAN   D5
#define PIN_LAMP  D6

// Analog MQ135
#define PIN_MQ135A0 A0

// UBAH ke false jika modul relay kamu aktif-HIGH
bool relayActiveLow = true;

// ====== TOPIK MQTT ======
#define TOPIC_CMD_FAN     "cmd/fan"
#define TOPIC_CMD_LAMP    "cmd/lamp"
#define TOPIC_STAT_FAN    "stat/fan"
#define TOPIC_STAT_LAMP   "stat/lamp"
#define TOPIC_AVAIL       "avail/node3"     // LWT
#define TOPIC_SYNC_REQ    "cmd/node3/sync"  // pancing publish
status
#define TOPIC_MQ135       "env/mq135"       // <-- publish MQ135
dari Node-3

WiFiClient espClient;
PubSubClient mqtt(espClient);

// ====== STATE ======
bool fanOn  = false;
bool lampOn = false;
int  mq135Val = -1;

// ====== UTIL RELAY ======
void setRelayPin(uint8_t pin, bool on) {
```

```
  if (relayActiveLow) digitalWrite(pin, on ? LOW : HIGH);
  else                digitalWrite(pin, on ? HIGH : LOW);
}

bool getRelayPin(uint8_t pin) {
  int lv = digitalRead(pin);
  return relayActiveLow ? (lv == LOW) : (lv == HIGH);
}

// ====== PUBLISH STATUS (retained) ======
void publishState() {
  mqtt.publish(TOPIC_STAT_FAN,  fanOn  ? "ON" : "OFF", true);
  mqtt.publish(TOPIC_STAT_LAMP, lampOn ? "ON" : "OFF", true);
}

// ====== PUBLISH MQ135 (retained) ======
void publishMQ135(int value) {
  char buf[16];
  snprintf(buf, sizeof(buf), "%d", value);
  mqtt.publish(TOPIC_MQ135, buf, true);
}

// ====== HANDLE PERINTAH ======
void applyCmd(const String& topic, const String& raw){
  String msg = raw; msg.trim();
  String up = msg; up.toUpperCase();
  bool on = (up=="ON" || up=="1" || up=="TRUE");

  if (topic == TOPIC_CMD_FAN) {
    setRelayPin(PIN_FAN, on);
    delay(15);
    fanOn = getRelayPin(PIN_FAN);
    publishState();
    Serial.printf("[CMD] FAN  <- %s  => stat=%s\n", up.c_str(),
fanOn ? "ON" : "OFF");
  } else if (topic == TOPIC_CMD_LAMP) {
    setRelayPin(PIN_LAMP, on);
    delay(15);
    lampOn = getRelayPin(PIN_LAMP);
    publishState();
    Serial.printf("[CMD] LAMP <- %s  => stat=%s\n", up.c_str(),
lampOn ? "ON" : "OFF");
  }
}

// ====== MQTT CALLBACK ======
```

```
void callback(char* topic, byte* payload, unsigned int len){
  String t = String(topic);
  String msg; msg.reserve(len);
  for (unsigned int i=0;i<len;i++) msg += (char)payload[i];
  msg.trim();

  if (t == TOPIC_SYNC_REQ) {   // pancing publish status sekarang
    publishState();
    // juga publish nilai MQ135 saat diminta (opsional)
    if (mq135Val >= 0) publishMQ135(mq135Val);
    return;
  }
  applyCmd(t, msg);
  Serial.printf("[MQTT] Diterima topic=%s payload=%s\n", topic,
msg.c_str());
}

// ====== WIFI/MQTT CONNECT ======
void connectWiFi(){
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(300);
    yield();
  }
  Serial.print("[WiFi] Connected. IP: ");
Serial.println(WiFi.localIP());
}

void connectMQTT(){
  mqtt.setServer(MQTT_HOST, MQTT_PORT);
  mqtt.setCallback(callback);

  while (!mqtt.connected()) {
    String cid = "node3-" + String(ESP.getChipId(), HEX);
    if (mqtt.connect(
          cid.c_str(),
          MQTT_USER, MQTT_PASSWD,
          TOPIC_AVAIL, 1, true, "offline"
        )) {
      mqtt.publish(TOPIC_AVAIL, "online", true);

      mqtt.subscribe(TOPIC_CMD_FAN);
      mqtt.subscribe(TOPIC_CMD_LAMP);
      mqtt.subscribe(TOPIC_SYNC_REQ);
```

```
      // Publish status awal (berdasarkan kondisi pin riil)
      fanOn  = getRelayPin(PIN_FAN);
      lampOn = getRelayPin(PIN_LAMP);
      publishState();

      // Jika ada nilai MQ135 sebelumnya, publish juga (saat
connect)
      if (mq135Val >= 0) publishMQ135(mq135Val);

      Serial.println("[MQTT] Connected & subscribed.");
    } else {
      delay(1000);
      yield();
    }
  }
}

// ====== SETUP ======
void setup(){
  Serial.begin(115200);

  pinMode(PIN_FAN,  OUTPUT);
  pinMode(PIN_LAMP, OUTPUT);
  // A0 tidak perlu pinMode, baca langsung

  // Default aman: OFF
  setRelayPin(PIN_FAN,  false);
  setRelayPin(PIN_LAMP, false);

  fanOn  = getRelayPin(PIN_FAN);
  lampOn = getRelayPin(PIN_LAMP);

  // Inisialisasi nilai MQ135
  mq135Val = analogRead(PIN_MQ135A0);

  connectWiFi();
  connectMQTT();
}

// ====== LOOP ======
void loop(){
  if (WiFi.status() != WL_CONNECTED) connectWiFi();
  if (!mqtt.connected()) connectMQTT();
  mqtt.loop();
```

```
  // Heartbeat: publish status berkala agar subscriber baru cepat
sinkron
  static unsigned long lastBeat = 0;
  static unsigned long lastReadMQ135 = 0;
  unsigned long now = millis();

  if (now - lastBeat > 10000UL) {
    lastBeat = now;
    bool curFan  = getRelayPin(PIN_FAN);
    bool curLamp = getRelayPin(PIN_LAMP);
    if (curFan != fanOn || curLamp != lampOn) {
      fanOn  = curFan;
      lampOn = curLamp;
    }
    publishState();
  }

  // Baca & publish MQ135 tiap 2s (retained)
  if (now - lastReadMQ135 > 500UL) {
    lastReadMQ135 = now;
    mq135Val = analogRead(PIN_MQ135A0);
    publishMQ135(mq135Val);
    Serial.printf("[SENSOR] MQ135 -> %d\n", mq135Val);
  }
}
```

UNDIKSHA