LAMPIRAN

1. Sketch Arduino UNO

```
#include <DHT.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>

// ====== KONFIGURASI ======
#define DHTPIN 7
#define DHTTYPE DHT22
#define RELAYPIN 3
#define BT_RX 10
#define BT_TX 11

DHT dht(DHTPIN, DHTTYPE);
SoftwareSerial bluetooth(BT_RX, BT_TX); // HC-05 di pin 10 RX, 11 TX
LiquidCrystal_I2C lcd(0x27, 16, 2); // Address 0x27, 16 kolom, 2 baris

// ====== VARIABEL ======
float temperature = 0;
String inputString = "";
String serialInputString = ""; // Buffer terpisah untuk Serial
bool fanStatus = false;
bool lcdWorking = false; // Flag untuk LCD
String lastCommand = "NONE";
unsigned long lastBTActivity = 0; // Waktu terakhir ada data BT
unsigned long lastSerialActivity = 0; // Waktu terakhir ada data Serial
const unsigned long BT_TIMEOUT = 1000; // 1 detik timeout
const unsigned long SERIAL_TIMEOUT = 500; // 0.5 detik timeout untuk serial

// ====== VARIABEL UNTUK MODE AUTO/MANUAL ======
bool autoMode = false; // false = manual, true = auto
float tempThreshold = 28.0; // Suhu threshold untuk mode auto
bool lastAutoState = false; // State terakhir kipas di mode auto
unsigned long lastTempCheck = 0;
const unsigned long TEMP_CHECK_INTERVAL = 2000; // Check suhu setiap 2 detik

// ====== COMMUNICATION CONTROL ======
bool serialBusy = false;
unsigned long lastSerialSend = 0;
const unsigned long SERIAL_SEND_DELAY = 100; // Minimum delay antar pengiriman

void setup() {
  Serial.begin(9600);
  bluetooth.begin(9600);
  dht.begin();

  pinMode(RELAYPIN, OUTPUT);
  digitalWrite(RELAYPIN, LOW);
  fanStatus = false;

  delay(1000); // Give time for serial to stabilize
  Serial.println("Starting system...");
  Serial.flush(); // Ensure message is sent

  // Inisialisasi LCD I2C
  lcd.init();
```

```
  // Coba address 0x27
  Wire.beginTransmission(0x27);
  if (Wire.endTransmission() == 0) {
    lcdWorking = true;
    Serial.println("LCD OK - Address 0x27");
    Serial.flush();

    lcd.backlight();
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("KIPAS PINTAR");
    lcd.setCursor(0, 1);
    lcd.print("AUTO/MANUAL");
    delay(2000);

  } else {
    // Coba address 0x3F
    lcd = LiquidCrystal_I2C(0x3F, 16, 2);
    lcd.init();
    Wire.beginTransmission(0x3F);
    if (Wire.endTransmission() == 0) {
      lcdWorking = true;
      Serial.println("LCD OK - Address 0x3F");
      Serial.flush();

      lcd.backlight();
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("KIPAS PINTAR");
      lcd.setCursor(0, 1);
      lcd.print("AUTO/MANUAL");
      delay(2000);

    } else {
      Serial.println("LCD GAGAL - Address 0x27 dan 0x3F");
      Serial.flush();
      lcdWorking = false;
    }
  }

  Serial.println("SYSTEM READY!");
  Serial.println("Commands: AUTO, MANUAL, ON, OFF, STATUS, SETTEMP:xx.x, TEST, HEARTBEAT");
  Serial.flush();
  bluetooth.println("SYSTEM READY!");

  // Set default ke mode MANUAL
  autoMode = false;
  lastTempCheck = millis();
  lastSerialSend = millis();
}

void loop() {
  readTemp();
  handleBluetooth();
  handleSerialFromNodeMCU();

  // Handle automatic mode
  if (autoMode) {
    handleAutoMode();
  }
```

```
  if (lcdWorking) {
    tampilkanLCD();
  }

  delay(100); // Reduced delay for better responsiveness
}

void readTemp() {
  float t = dht.readTemperature();
  if (!isnan(t)) {
    temperature = t;
  } else {
    Serial.println("DHT22 Error");
    Serial.flush();
  }
}

void handleAutoMode() {
  // Check suhu setiap TEMP_CHECK_INTERVAL
  if (millis() - lastTempCheck < TEMP_CHECK_INTERVAL) {
    return;
  }
  lastTempCheck = millis();

  bool shouldTurnOn = temperature >= tempThreshold;

  // Jika status berubah, update kipas
  if (shouldTurnOn != lastAutoState) {
    if (shouldTurnOn) {
      // Nyalakan kipas
      digitalWrite(RELAYPIN, HIGH);
      fanStatus = true;
      lastAutoState = true;

      String response = "KIPAS AUTO ON - SUHU: " + String(temperature, 1) + "C (>=" +
String(tempThreshold, 1) + "C)";
      safeSerialPrint(response);
      bluetooth.println(response);

    } else {
      // Matikan kipas
      digitalWrite(RELAYPIN, LOW);
      fanStatus = false;
      lastAutoState = false;

      String response = "KIPAS AUTO OFF - SUHU: " + String(temperature, 1) + "C (<" +
String(tempThreshold, 1) + "C)";
      safeSerialPrint(response);
      bluetooth.println(response);
    }
  }
}

void handleBluetooth() {
  // Cek timeout - jika ada data tapi tidak ada newline
  if (inputString.length() > 0 && (millis() - lastBTActivity) > BT_TIMEOUT) {
    Serial.print("BT TIMEOUT - Processing: [");
    Serial.print(inputString);
    Serial.println("]");
    Serial.flush();
```

```
    lastCommand = inputString;
    processCommand(inputString);
    inputString = "";
  }

  while (bluetooth.available()) {
    char c = bluetooth.read();
    lastBTActivity = millis(); // Update waktu aktivitas

    if (c == '\n' || c == '\r') {
      if (inputString.length() > 0) {
        inputString.trim();
        Serial.print("BT COMMAND: [");
        Serial.print(inputString);
        Serial.println("]");
        Serial.flush();

        lastCommand = inputString;
        processCommand(inputString);
        inputString = "";
      }
    } else if (c >= 32 && c <= 126) { // Karakter printable
      inputString += c;

      // Auto-process jika command sudah jelas
      if (inputString == "ON" || inputString == "OFF" ||
          inputString == "STATUS" || inputString == "TEST" ||
          inputString == "AUTO" || inputString == "MANUAL" ||
          inputString == "HEARTBEAT") {

        lastCommand = inputString;
        processCommand(inputString);
        inputString = "";
      }

      // Check untuk SETTEMP command
      if (inputString.startsWith("SETTEMP:") && inputString.length() >= 10) {
        lastCommand = inputString;
        processCommand(inputString);
        inputString = "";
      }

      // Reset jika buffer terlalu panjang
      if (inputString.length() > 30) {
        Serial.println("BT Buffer overflow - reset");
        Serial.flush();
        inputString = "";
      }
    }
  }
}

void handleSerialFromNodeMCU() {
  // Handle timeout untuk serial input
  if (serialInputString.length() > 0 && (millis() - lastSerialActivity) > SERIAL_TIMEOUT) {
    if (serialInputString.length() > 0) {
      serialInputString.trim();
      Serial.print("SERIAL TIMEOUT - Processing: [");
      Serial.print(serialInputString);
      Serial.println("]");
      Serial.flush();
```

```
      lastCommand = serialInputString;
      processCommand(serialInputString);
      serialInputString = "";
    }
  }

  while (Serial.available() && !serialBusy) {
    serialBusy = true;
    char c = Serial.read();
    lastSerialActivity = millis();

    if (c == '\n' || c == '\r') {
      if (serialInputString.length() > 0) {
        serialInputString.trim();
        Serial.print("SERIAL COMMAND: [");
        Serial.print(serialInputString);
        Serial.println("]");
        Serial.flush();

        lastCommand = serialInputString;
        processCommand(serialInputString);
        serialInputString = "";
      }
    } else if (c >= 32 && c <= 126) { // Karakter printable
      serialInputString += c;

      // Auto-process untuk command yang sudah jelas
      if (serialInputString == "ON" || serialInputString == "OFF" ||
          serialInputString == "STATUS" || serialInputString == "TEST" ||
          serialInputString == "AUTO" || serialInputString == "MANUAL" ||
          serialInputString == "HEARTBEAT") {

        lastCommand = serialInputString;
        processCommand(serialInputString);
        serialInputString = "";
      }

      // Check untuk SETTEMP command
      if (serialInputString.startsWith("SETTEMP:") && serialInputString.length() >= 10) {
        lastCommand = serialInputString;
        processCommand(serialInputString);
        serialInputString = "";
      }

      // Reset jika buffer terlalu panjang
      if (serialInputString.length() > 30) {
        Serial.println("SERIAL Buffer overflow - reset");
        Serial.flush();
        serialInputString = "";
      }
    }

    serialBusy = false;
    delay(10); // Small delay to prevent buffer issues
  }
}

void processCommand(String cmd) {
  cmd.trim();
  cmd.toUpperCase();
```

```
  if (cmd == "AUTO") {
    autoMode = true;
    lastAutoState = fanStatus; // Simpan state saat ini

    String response = "AUTO MODE ON - THRESHOLD: " + String(tempThreshold, 1) + "C | TEMP: " +
String(temperature, 1) + "C";
    safeSerialPrint(response);
    bluetooth.println(response);

    if (lcdWorking) {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("MODE: AUTO");
      lcd.setCursor(0, 1);
      lcd.print("TEMP: ");
      lcd.print(temperature, 1);
      lcd.print("C");
    }

  } else if (cmd == "MANUAL") {
    autoMode = false;

    String response = "MANUAL MODE ON - KIPAS: " + String(fanStatus ? "ON" : "OFF") + " | TEMP:
" + String(temperature, 1) + "C";
    safeSerialPrint(response);
    bluetooth.println(response);

    if (lcdWorking) {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("MODE: MANUAL");
      lcd.setCursor(0, 1);
      lcd.print("KIPAS: ");
      lcd.print(fanStatus ? "ON" : "OFF");
    }

  } else if (cmd.startsWith("SETTEMP:")) {
    // Parse temperature value
    int colonIndex = cmd.indexOf(':');
    if (colonIndex > 0) {
      String tempStr = cmd.substring(colonIndex + 1);
      float newTemp = tempStr.toFloat();

      if (newTemp >= 20.0 && newTemp <= 40.0) {
        tempThreshold = newTemp;

        String response = "THRESHOLD SET: " + String(tempThreshold, 1) + "C | MODE: " +
(autoMode ? "AUTO" : "MANUAL");
        safeSerialPrint(response);
        bluetooth.println(response);

        if (lcdWorking) {
          lcd.clear();
          lcd.setCursor(0, 0);
          lcd.print("THRESHOLD SET");
          lcd.setCursor(0, 1);
          lcd.print("TEMP: ");
          lcd.print(tempThreshold, 1);
          lcd.print("C");
          delay(1000);
```

```
        }
      } else {
        safeSerialPrint("ERROR: Temperature must be 20-40C");
        bluetooth.println("ERROR: Temperature must be 20-40C");
      }
    }

  } else if (cmd == "ON") {
    if (autoMode) {
      safeSerialPrint("WARNING: System in AUTO mode. Use MANUAL first.");
      bluetooth.println("WARNING: System in AUTO mode. Use MANUAL first.");
    } else {
      digitalWrite(RELAYPIN, HIGH);
      fanStatus = true;
      safeSerialPrint("KIPAS: ON (MANUAL)");
      bluetooth.println("KIPAS: ON (MANUAL)");

      if (lcdWorking) {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("KIPAS: ON");
        lcd.setCursor(0, 1);
        lcd.print("MODE: MANUAL");
      }
    }

  } else if (cmd == "OFF") {
    if (autoMode) {
      safeSerialPrint("WARNING: System in AUTO mode. Use MANUAL first.");
      bluetooth.println("WARNING: System in AUTO mode. Use MANUAL first.");
    } else {
      digitalWrite(RELAYPIN, LOW);
      fanStatus = false;
      safeSerialPrint("KIPAS: OFF (MANUAL)");
      bluetooth.println("KIPAS: OFF (MANUAL)");

      if (lcdWorking) {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("KIPAS: OFF");
        lcd.setCursor(0, 1);
        lcd.print("MODE: MANUAL");
      }
    }

  } else if (cmd == "STATUS") {
    String mode = autoMode ? "AUTO" : "MANUAL";
    String fanState = fanStatus ? "ON" : "OFF";
    String status = "MODE: " + mode + " | KIPAS: " + fanState + " | TEMP: " +
String(temperature, 1) + "C";

    if (autoMode) {
      status += " | THRESHOLD: " + String(tempThreshold, 1) + "C";
    }

    safeSerialPrint(status);
    bluetooth.println(status);

  } else if (cmd == "HEARTBEAT") {
    // Silent heartbeat response - just acknowledge
    safeSerialPrint("HEARTBEAT_OK");
```

```
  } else if (cmd == "TEST") {
    safeSerialPrint("TEST OK - Auto/Manual System Working!");
    bluetooth.println("TEST OK - Auto/Manual System Working!");

    if (lcdWorking) {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("TEST OK!");
      lcd.setCursor(0, 1);
      lcd.print("AUTO/MANUAL");
      delay(2000);
    }

  } else if (cmd.length() > 0) {
    Serial.print(">>> UNKNOWN COMMAND: [");
    Serial.print(cmd);
    Serial.println("] <<<");
    Serial.flush();
    bluetooth.print("UNKNOWN: [");
    bluetooth.print(cmd);
    bluetooth.println("]");
  }
}

// Safe serial print function dengan flow control
void safeSerialPrint(String message) {
  // Tunggu jika masih ada pengiriman sebelumnya
  while (millis() - lastSerialSend < SERIAL_SEND_DELAY) {
    delay(10);
  }

  Serial.println(message);
  Serial.flush(); // Tunggu sampai data benar-benar terkirim
  lastSerialSend = millis();
}

void tampilkanLCD() {
  if (!lcdWorking) return;

  static unsigned long lastUpdate = 0;
  if (millis() - lastUpdate < 2000) return; // Update setiap 2 detik
  lastUpdate = millis();

  lcd.clear();

  // Baris 1: Mode dan Status Kipas
  lcd.setCursor(0, 0);
  if (autoMode) {
    lcd.print("AUTO ");
    lcd.print(fanStatus ? "ON" : "OFF");
    lcd.print(" ");
    lcd.print(String(tempThreshold, 0));
    lcd.print("C");
  } else {
    lcd.print("MANUAL ");
    lcd.print(fanStatus ? "ON" : "OFF");
  }

  // Baris 2: Suhu saat ini
  lcd.setCursor(0, 1);
```

```
  lcd.print("SUHU: ");
  lcd.print(temperature, 1);
  lcd.print(" C");
}

// Fungsi untuk scan I2C address LCD
void scanI2C() {
  Serial.println("Scanning I2C...");
  Serial.flush();
  for (byte i = 8; i < 120; i++) {
    Wire.beginTransmission(i);
    if (Wire.endTransmission() == 0) {
      Serial.print("I2C device found at 0x");
      if (i < 16) Serial.print("0");
      Serial.println(i, HEX);
      Serial.flush();
    }
  }
  Serial.println("Scan complete");
  Serial.flush();
}
```

2. Sketch ESP8266

```
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>
#include <SoftwareSerial.h>

// ====== KONFIGURASI WIFI ======
const char* ssid = "SMART_FAN";
const char* password = "12345678";

// ====== KONFIGURASI TELEGRAM ======
#define BOT_TOKEN "8115348340:AAFNQ4-LjWgb0pH52ACBtk4_jXOzOYfK0hI"
#define CHAT_ID "7319836465"

// ====== PIN KONFIGURASI ======
#define LED_BUILTIN 2
#define ARDUINO_RX D5  // GPIO5 - ke TX Arduino
#define ARDUINO_TX D6  // GPIO4 - ke RX Arduino
// ====== OBJEK ======
WiFiClientSecure client;
UniversalTelegramBot bot(BOT_TOKEN, client);
SoftwareSerial arduino(ARDUINO_RX, ARDUINO_TX);
// ====== VARIABEL ======
unsigned long lastTimeBotRan = 0;
const unsigned long botRequestDelay = 1000; // Check pesan setiap 1 detik
String lastArduinoResponse = "";
String currentArduinoResponse = ""; // Tambah variabel untuk respon real-time
bool systemReady = false;
unsigned long lastHeartbeat = 0;
const unsigned long heartbeatInterval = 30000; // 30 detik
unsigned long lastLEDBlink = 0;
bool ledState = false;
bool waitingForResponse = false; // Flag untuk menunggu respon
unsigned long responseWaitStart = 0;
const unsigned long RESPONSE_WAIT_TIMEOUT = 3000; // 3 detik timeout untuk respon command
```

```
// Buffer untuk komunikasi Arduino
String arduinoBuffer = "";
unsigned long lastArduinoActivity = 0;
const unsigned long ARDUINO_TIMEOUT = 2000; // 2 detik timeout

// ====== FUNGSI DECLARATIONS ======
void connectToWiFi();
bool testTelegramConnection();
void sendTelegramMessage(String message);
void handleTelegramMessages();
void handleTelegramCommand(String command, String chat_id);
void sendToArduino(String command);
void handleArduinoResponse();
void sendHeartbeat();
void updateLEDIndicator();

// ====== SETUP ======
void setup() {
  Serial.begin(115200);
  arduino.begin(9600);

  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, HIGH); // LED off (aktif low)

  Serial.println();
  Serial.println("=== ESP8266 KIPAS PINTAR AUTO/MANUAL ===");
  Serial.println("Starting...");

  // Connect to WiFi
  connectToWiFi();

  // Setup Telegram SSL
  client.setInsecure(); // Untuk koneksi Telegram

  // Test koneksi ke Telegram
  if (testTelegramConnection()) {
    sendTelegramMessage("🟢 ESP8266 ONLINE!\n\nSistem Kipas Pintar dengan Mode Auto/Manual siap
digunakan.\n\nKirim /help untuk melihat perintah yang tersedia.");
    systemReady = true;
  }

  Serial.println("ESP8266 READY!");
  Serial.println("Commands: /auto, /manual, /on, /off, /status, /help, /settemp");

  // Blink LED untuk indikasi ready
  for (int i = 0; i < 5; i++) {
    digitalWrite(LED_BUILTIN, LOW);
    delay(200);
    digitalWrite(LED_BUILTIN, HIGH);
    delay(200);
  }

  lastHeartbeat = millis();
  lastLEDBlink = millis();
}

// ====== MAIN LOOP ======
void loop() {
  // Check WiFi connection
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("WiFi disconnected. Reconnecting...");
```

```
    connectToWiFi();
  }

  // Handle Telegram messages
  if (millis() - lastTimeBotRan > botRequestDelay) {
    handleTelegramMessages();
    lastTimeBotRan = millis();
  }

  // Handle Arduino responses
  handleArduinoResponse();

  // Send heartbeat every 30 seconds
  if (millis() - lastHeartbeat > heartbeatInterval) {
    sendHeartbeat();
    lastHeartbeat = millis();
  }

  // Update LED indicator
  updateLEDIndicator();

  delay(100);
}

// ====== WIFI CONNECTION ======
void connectToWiFi() {
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi");

  int attempts = 0;
  while (WiFi.status() != WL_CONNECTED && attempts < 30) {
    delay(500);
    Serial.print(".");
    attempts++;

    // Blink LED during connection
    digitalWrite(LED_BUILTIN, attempts % 2);
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println();
    Serial.println("WiFi Connected!");
    Serial.print("IP Address: ");
    Serial.println(WiFi.localIP());
    digitalWrite(LED_BUILTIN, LOW); // LED on
  } else {
    Serial.println();
    Serial.println("WiFi Connection Failed!");
    digitalWrite(LED_BUILTIN, HIGH); // LED off
  }
}

// ====== TELEGRAM FUNCTIONS ======
bool testTelegramConnection() {
  Serial.print("Testing Telegram connection...");

  // Add retry mechanism
  for (int i = 0; i < 3; i++) {
    bool result = bot.sendMessage(CHAT_ID, "🔄 Test connection", "");
    if (result) {
      Serial.println(" OK");
```

```
      return true;
    }
    Serial.print(".");
    delay(1000);
  }

  Serial.println(" FAILED");
  return false;
}

void handleTelegramMessages() {
  int numNewMessages = bot.getUpdates(bot.last_message_received + 1);

  for (int i = 0; i < numNewMessages; i++) {
    String chat_id = String(bot.messages[i].chat_id);
    String text = bot.messages[i].text;
    String from_name = bot.messages[i].from_name;

    Serial.println("Received: " + text + " from " + from_name + " (ID: " + chat_id + ")");

    // Security check - hanya terima dari chat_id yang diizinkan
    if (chat_id != CHAT_ID) {
      bot.sendMessage(chat_id, "❌ Akses ditolak! Anda tidak memiliki izin untuk menggunakan
bot ini.");
      Serial.println("Unauthorized access attempt from: " + chat_id);
      continue;
    }

    handleTelegramCommand(text, chat_id);
  }
}

void handleTelegramCommand(String command, String chat_id) {
  String originalCommand = command;
  command.toLowerCase();
  command.trim(); // Remove extra spaces

  Serial.println("Processing command: " + command);

  if (command == "/start") {
    String welcome = "🏠 SELAMAT DATANG DI KIPAS PINTAR AUTO/MANUAL!\n\n";
    welcome += "Bot ini dapat mengontrol kipas dalam 2 mode:\n\n";
    welcome += "🤖 MODE AUTO: Kipas otomatis berdasarkan suhu\n";
    welcome += "👤 MODE MANUAL: Kontrol kipas secara manual\n\n";
    welcome += "📋 PERINTAH YANG TERSEDIA:\n";
    welcome += "/auto - Aktifkan mode otomatis\n";
    welcome += "/manual - Aktifkan mode manual\n";
    welcome += "/on - Nyalakan kipas (mode manual)\n";
    welcome += "/off - Matikan kipas (mode manual)\n";
    welcome += "/settemp 28 - Set suhu threshold (mode auto)\n";
    welcome += "/status - Cek status & suhu\n";
    welcome += "/help - Tampilkan bantuan\n";
    welcome += "\n🔄 Status akan diupdate otomatis!\n\n";
    welcome += "Dibuat dengan ❤ untuk rumah pintar Anda!";

    bot.sendMessage(chat_id, welcome, "");

  } else if (command == "/help") {
    String help = "📋 BANTUAN PENGGUNAAN:\n\n";
    help += "🤖 MODE OTOMATIS:\n";
    help += "/auto - Aktifkan mode otomatis\n";
```

```
  help += "/settemp [suhu] - Set threshold (contoh: /settemp 28)\n\n";
  help += "👨‍💼 MODE MANUAL:\n";
  help += "/manual - Aktifkan mode manual\n";
  help += "🟢 /on - Nyalakan kipas\n";
  help += "🔴 /off - Matikan kipas\n\n";
  help += "📊 MONITORING:\n";
  help += "/status - Lihat status sistem lengkap\n\n";
  help += "💡 TIPS:\n";
  help += "- Mode auto: kipas menyala otomatis saat suhu tinggi\n";
  help += "- Mode manual: kontrol penuh di tangan Anda\n";
  help += "- Gunakan /status untuk monitoring rutin\n\n";
  help += "🔧 TROUBLESHOOTING:\n";
  help += "Jika tidak ada respon, coba /status untuk cek koneksi Arduino.";

  bot.sendMessage(chat_id, help, "");

} else if (command == "/auto") {
  currentArduinoResponse = ""; // Reset response
  waitingForResponse = true;
  responseWaitStart = millis();
  sendToArduino("AUTO");

  // Wait for response with timeout
  while (waitingForResponse && (millis() - responseWaitStart) < RESPONSE_WAIT_TIMEOUT) {
    handleArduinoResponse();
    delay(50);
  }

  String response = "🤖 MODE OTOMATIS DIAKTIFKAN\n\n";
  response += "Kipas akan menyala otomatis berdasarkan suhu.\n";
  response += "Gunakan /settemp untuk mengatur threshold.\n";
  response += "Cek /status untuk melihat pengaturan saat ini.\n\n";

  if (currentArduinoResponse != "" && currentArduinoResponse != "HEARTBEAT_OK") {
    response += "📡 Respon Arduino: " + currentArduinoResponse;
  } else if (lastArduinoResponse != "" && lastArduinoResponse != "HEARTBEAT_OK") {
    response += "📡 Respon Arduino: " + lastArduinoResponse;
  } else {
    response += "⚠️ Tidak ada respon dari Arduino dalam 3 detik.";
  }

  bot.sendMessage(chat_id, response, "");
  waitingForResponse = false;

} else if (command == "/manual") {
  currentArduinoResponse = ""; // Reset response
  waitingForResponse = true;
  responseWaitStart = millis();
  sendToArduino("MANUAL");

  // Wait for response with timeout
  while (waitingForResponse && (millis() - responseWaitStart) < RESPONSE_WAIT_TIMEOUT) {
    handleArduinoResponse();
    delay(50);
  }

  String response = "👨‍💼 MODE MANUAL DIAKTIFKAN\n\n";
  response += "Sekarang Anda dapat mengontrol kipas secara manual.\n";
  response += "Gunakan /on untuk menyalakan kipas.\n";
  response += "Gunakan /off untuk mematikan kipas.\n\n";
```

```
    if (currentArduinoResponse != "" && currentArduinoResponse != "HEARTBEAT_OK") {
      response += "📡 Respon Arduino: " + currentArduinoResponse;
    } else if (lastArduinoResponse != "" && lastArduinoResponse != "HEARTBEAT_OK") {
      response += "📡 Respon Arduino: " + lastArduinoResponse;
    } else {
      response += "⚠️ Tidak ada respon dari Arduino dalam 3 detik.";
    }

    bot.sendMessage(chat_id, response, "");
    waitingForResponse = false;

  } else if (command.startsWith("/settemp")) {
    // Parse temperature from command - improved parsing
    int spaceIndex = originalCommand.indexOf(' ');
    if (spaceIndex > 0 && spaceIndex < originalCommand.length() - 1) {
      String tempStr = originalCommand.substring(spaceIndex + 1);
      tempStr.trim();

      // Validate input is numeric
      bool isValidNumber = true;
      for (int i = 0; i < tempStr.length(); i++) {
        if (!isDigit(tempStr.charAt(i)) && tempStr.charAt(i) != '.') {
          isValidNumber = false;
          break;
        }
      }

      if (isValidNumber) {
        float tempValue = tempStr.toFloat();
        if (tempValue >= 20 && tempValue <= 40) {
          currentArduinoResponse = ""; // Reset response
          waitingForResponse = true;
          responseWaitStart = millis();
          sendToArduino("SETTEMP:" + String(tempValue, 1));

          // Wait for response with timeout
          while (waitingForResponse && (millis() - responseWaitStart) < RESPONSE_WAIT_TIMEOUT) {
            handleArduinoResponse();
            delay(50);
          }

          String response = "🌡️ THRESHOLD SUHU DIATUR\n\n";
          response += "Suhu threshold: " + String(tempValue, 1) + "°C\n";
          response += "Kipas akan menyala otomatis jika suhu ≥ " + String(tempValue, 1) +
"°C\n";
          response += "(Hanya berlaku di mode AUTO)\n\n";

          if (currentArduinoResponse != "" && currentArduinoResponse != "HEARTBEAT_OK") {
            response += "📡 Respon Arduino: " + currentArduinoResponse;
          } else if (lastArduinoResponse != "" && lastArduinoResponse != "HEARTBEAT_OK") {
            response += "📡 Respon Arduino: " + lastArduinoResponse;
          } else {
            response += "⚠️ Tidak ada respon dari Arduino dalam 3 detik.";
          }

          bot.sendMessage(chat_id, response, "");
          waitingForResponse = false;
        } else {
          bot.sendMessage(chat_id, "❌ Suhu harus antara 20-40°C!\nContoh: /settemp 28", "");
        }
      } else {
```

```
      bot.sendMessage(chat_id, "❌ Format suhu tidak valid!\nContoh: /settemp 28", "");
    }
  } else {
    bot.sendMessage(chat_id, "❌ Format salah!\nContoh: /settemp 28", "");
  }

} else if (command == "/on") {
  currentArduinoResponse = ""; // Reset response
  waitingForResponse = true;
  responseWaitStart = millis();
  sendToArduino("ON");

  // Wait for response with timeout
  while (waitingForResponse && (millis() - responseWaitStart) < RESPONSE_WAIT_TIMEOUT) {
    handleArduinoResponse();
    delay(50);
  }

  String response = "🟢 PERINTAH NYALAKAN KIPAS\n\n";
  response += "Perintah manual telah dikirim ke sistem.\n\n";

  if (currentArduinoResponse != "" && currentArduinoResponse != "HEARTBEAT_OK") {
    response += "📡 Respon Arduino: " + currentArduinoResponse;
  } else if (lastArduinoResponse != "" && lastArduinoResponse != "HEARTBEAT_OK") {
    response += "📡 Respon Arduino: " + lastArduinoResponse;
  } else {
    response += "⚠️ Tidak ada respon dari Arduino dalam 3 detik.";
  }

  bot.sendMessage(chat_id, response, "");
  waitingForResponse = false;

} else if (command == "/off") {
  currentArduinoResponse = ""; // Reset response
  waitingForResponse = true;
  responseWaitStart = millis();
  sendToArduino("OFF");

  // Wait for response with timeout
  while (waitingForResponse && (millis() - responseWaitStart) < RESPONSE_WAIT_TIMEOUT) {
    handleArduinoResponse();
    delay(50);
  }

  String response = "🔴 PERINTAH MATIKAN KIPAS\n\n";
  response += "Perintah manual telah dikirim ke sistem.\n\n";

  if (currentArduinoResponse != "" && currentArduinoResponse != "HEARTBEAT_OK") {
    response += "📡 Respon Arduino: " + currentArduinoResponse;
  } else if (lastArduinoResponse != "" && lastArduinoResponse != "HEARTBEAT_OK") {
    response += "📡 Respon Arduino: " + lastArduinoResponse;
  } else {
    response += "⚠️ Tidak ada respon dari Arduino dalam 3 detik.";
  }

  bot.sendMessage(chat_id, response, "");
  waitingForResponse = false;

} else if (command == "/status") {
  currentArduinoResponse = ""; // Reset response
  waitingForResponse = true;
```

```
    responseWaitStart = millis();
    sendToArduino("STATUS");

    // Wait longer for status response
    while (waitingForResponse && (millis() - responseWaitStart) < RESPONSE_WAIT_TIMEOUT) {
      handleArduinoResponse();
      delay(50);
    }

    String response = "📊 STATUS SISTEM KIPAS PINTAR:\n\n";
    response += "📡 WiFi: " + String(WiFi.status() == WL_CONNECTED ? "Terhubung" : "Terputus")
+ "\n";
    response += "📶 Signal: " + String(WiFi.RSSI()) + " dBm\n";
    response += "🌐 IP: " + WiFi.localIP().toString() + "\n\n";

    if (currentArduinoResponse != "" && currentArduinoResponse != "HEARTBEAT_OK") {
      response += "🔔 Arduino: " + currentArduinoResponse + "\n\n";
    } else if (lastArduinoResponse != "" && lastArduinoResponse != "HEARTBEAT_OK") {
      response += "🔔 Arduino: " + lastArduinoResponse + "\n\n";
    } else {
      response += "⚠ Arduino: Tidak ada respon dalam 3 detik\n\n";
    }

    bot.sendMessage(chat_id, response, "");
    waitingForResponse = false;

  } else if (command.startsWith("/")) {
    String response = "❌ PERINTAH TIDAK DIKENAL\n\n";
    response += "Perintah '" + command + "' tidak tersedia.\n\n";
    response += "📋 PERINTAH YANG TERSEDIA:\n";
    response += "/auto - Mode otomatis\n";
    response += "/manual - Mode manual\n";
    response += "/on - Nyalakan kipas\n";
    response += "/off - Matikan kipas\n";
    response += "/settemp [suhu] - Set threshold\n";
    response += "/status - Cek status\n";
    response += "/help - Bantuan lengkap";

    bot.sendMessage(chat_id, response, "");

  } else {
    // Handle non-command messages
    String response = "💬 Halo! Saya bot pengontrol kipas pintar dengan mode Auto/Manual.\n\n";
    response += "🤖 Mode AUTO: Kipas otomatis berdasarkan suhu\n";
    response += "👤 Mode MANUAL: Kontrol kipas secara manual\n\n";
    response += "Silakan gunakan perintah yang dimulai dengan '/'\n";
    response += "Ketik /help untuk melihat daftar lengkap perintah.";

    bot.sendMessage(chat_id, response, "");
  }
}

// ====== ARDUINO COMMUNICATION ======
void sendToArduino(String command) {
  Serial.println("Sending to Arduino: " + command);
  arduino.println(command);
  arduino.flush(); // Ensure data is sent

  // Clear previous response
  lastArduinoResponse = "";
  currentArduinoResponse = "";
```

```
  lastArduinoActivity = millis();

  // Blink LED untuk indikasi komunikasi
  digitalWrite(LED_BUILTIN, LOW);
  delay(100);
  digitalWrite(LED_BUILTIN, HIGH);
}

void handleArduinoResponse() {
  // Handle timeout
  if (arduinoBuffer.length() > 0 && (millis() - lastArduinoActivity) > ARDUINO_TIMEOUT) {
    if (arduinoBuffer.length() > 0) {
      arduinoBuffer.trim();
      Serial.println("Arduino Timeout Response: " + arduinoBuffer);

      // Save response dan set flag
      if (arduinoBuffer != "HEARTBEAT_OK") {
        lastArduinoResponse = arduinoBuffer;
        if (waitingForResponse) {
          currentArduinoResponse = arduinoBuffer;
        }
      }

      arduinoBuffer = "";
    }
  }

  while (arduino.available()) {
    char c = arduino.read();
    lastArduinoActivity = millis();

    if (c == '\n' || c == '\r') {
      if (arduinoBuffer.length() > 0) {
        arduinoBuffer.trim();
        Serial.println("Arduino Response: " + arduinoBuffer);

        // Save response dan set flag
        if (arduinoBuffer != "HEARTBEAT_OK") {
          lastArduinoResponse = arduinoBuffer;
          if (waitingForResponse) {
            currentArduinoResponse = arduinoBuffer;
            waitingForResponse = false; // Stop waiting setelah dapat respon
          }

          // Send critical errors to Telegram immediately
          if (arduinoBuffer.indexOf("ERROR") >= 0 ||
              arduinoBuffer.indexOf("GAGAL") >= 0 ||
              arduinoBuffer.indexOf("FAILED") >= 0) {
            sendTelegramMessage("⚠ SISTEM ALERT:\n" + arduinoBuffer);
          }
        } else {
          // Untuk HEARTBEAT_OK, tetap stop waiting jika sedang menunggu
          if (waitingForResponse) {
            waitingForResponse = false;
          }
        }

        arduinoBuffer = "";
      }
    } else if (c >= 32 && c <= 126) { // Printable characters only
      arduinoBuffer += c;
```

```
      // Prevent buffer overflow
      if (arduinoBuffer.length() > 200) {
        Serial.println("Arduino buffer overflow - reset");
        arduinoBuffer = "";
      }
    }
  }
}

// ====== UTILITY FUNCTIONS ======
void sendTelegramMessage(String message) {
  if (WiFi.status() == WL_CONNECTED && systemReady) {
    bool sent = bot.sendMessage(CHAT_ID, message, "");
    if (sent) {
      Serial.println("Sent to Telegram: " + message);
    } else {
      Serial.println("Failed to send to Telegram: " + message);
    }
  } else {
    Serial.println("Cannot send to Telegram - WiFi: " + String(WiFi.status()) + ", Ready: " +
String(systemReady));
  }
}

void sendHeartbeat() {
  if (systemReady) {
    sendToArduino("HEARTBEAT");
    Serial.println("Heartbeat sent to Arduino (silent)");
  }
}

void updateLEDIndicator() {
  // LED blinking pattern based on system status
  if (WiFi.status() != WL_CONNECTED) {
    // Fast blink if WiFi disconnected
    if (millis() - lastLEDBlink > 250) {
      ledState = !ledState;
      digitalWrite(LED_BUILTIN, ledState ? LOW : HIGH);
      lastLEDBlink = millis();
    }
  } else if (!systemReady) {
    // Slow blink if Telegram not ready
    if (millis() - lastLEDBlink > 1000) {
      ledState = !ledState;
      digitalWrite(LED_BUILTIN, ledState ? LOW : HIGH);
      lastLEDBlink = millis();
    }
  } else {
    // Solid on if everything OK
    digitalWrite(LED_BUILTIN, LOW);
  }
}
```