LAMPIRAN

Listing Code

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
#include <Servo.h>
#include <WiFiUdp.h>
#include <NTPClient.h>

// WiFi & ThingSpeak - DATA AKUN BARU
const char* ssid = "SOLIN_PROJECT";
const char* password = "17AGUSTUS2025";

// DATA DARI CHANNEL BARU:
const char* CHANNEL_ID = "3031335";
const char* READ_API_KEY = "6P02LA1I36QPJ9T9";
const char* WRITE_API_KEY = "PADNE6UX97YWS242";

// Build URL otomatis
String readServer = "http://api.thingspeak.com/channels/" +
String(CHANNEL_ID) + "/fields/1/last.txt?api_key=" +
String(READ_API_KEY);
const char* writeServer = "http://api.thingspeak.com/update";

// Servo dan Relay
Servo myservo;
int servoPin = D1;
int relayPin = D2;  // Pin untuk relay (aktif-low)

// Jadwal otomatis (jam, menit, detik)
const int autoTimes[2][3] = {
  {1, 57, 0},   // 01:57:00 WIB
  {17, 30, 0}   // 17:30:00 WIB
};
bool autoDoneToday[2] = {false, false};

// NTP Client
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", 7 * 3600); // GMT+7 WIB

// Manual control - improved logic
String lastPayload = "0";
unsigned long lastPayloadTime = 0;
unsigned long lastExecutionTime = 0;
const unsigned long MIN_EXECUTION_INTERVAL = 30000; // 30 detik
minimal antar eksekusi
```

```
void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);

  // Inisialisasi servo dan relay
  myservo.attach(servoPin);
  myservo.write(0);

  pinMode(relayPin, OUTPUT);
  digitalWrite(relayPin, HIGH); // Relay OFF saat startup (aktif-low)

  Serial.println("Inisialisasi:");
  Serial.println("- Servo di pin D1 (posisi 0°)");
  Serial.println("- Relay di pin D2 (OFF)");

  Serial.print("Menghubungkan ke WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi Terhubung!");

  timeClient.begin();

  // Reset field1 saat startup untuk clean state
  resetThingSpeakField();
  delay(2000);
}

void loop() {
  timeClient.update();

  int currentHour = timeClient.getHours();
  int currentMinute = timeClient.getMinutes();
  int currentSecond = timeClient.getSeconds();

  // Tampilkan jam sekarang setiap 10 detik
  static unsigned long lastTimeDisplay = 0;
  if (millis() - lastTimeDisplay > 10000) {
    Serial.printf("Waktu sekarang: %02d:%02d:%02d\n", currentHour,
currentMinute, currentSecond);
    lastTimeDisplay = millis();
  }

  // === Otomatis berdasarkan jadwal ===
  for (int i = 0; i < 2; i++) {
    if (currentHour == autoTimes[i][0] &&
```

```
      currentMinute == autoTimes[i][1] &&
      currentSecond >= 0 && currentSecond <= 5 &&
      !autoDoneToday[i] &&
      (millis() - lastExecutionTime > MIN_EXECUTION_INTERVAL)) {

    Serial.println("=== Eksekusi otomatis ===");
    jalankanServoRelay();
    autoDoneToday[i] = true;
    lastExecutionTime = millis();
   }

   // Reset flag jika sudah lewat 1 menit dari jadwal
   if (currentHour != autoTimes[i][0] || currentMinute != autoTimes[i][1]) {
    autoDoneToday[i] = false;
   }
  }

  // === Kontrol manual dari ThingSpeak ===
  checkManualCommand();

  delay(3000);
}

void checkManualCommand() {
  if (WiFi.status() == WL_CONNECTED) {
   WiFiClient client;
   HTTPClient http;

   http.begin(client, readServer.c_str());
   int httpCode = http.GET();

   if (httpCode == HTTP_CODE_OK) {
    String payload = http.getString();
    payload.trim();

    unsigned long currentTime = millis();

    if (payload != lastPayload || (currentTime - lastPayloadTime > 30000)) {
     Serial.println("Data dari ThingSpeak: " + payload);
     lastPayloadTime = currentTime;
    }

    if (payload == "1" &&
       lastPayload != "1" &&
       (currentTime - lastExecutionTime > MIN_EXECUTION_INTERVAL)) {

     Serial.println("=== Eksekusi manual (payload berubah) ===");
     lastPayload = "1";
```

```
        jalankanServoRelay();
        lastExecutionTime = currentTime;

        Serial.println("Waiting 10 seconds before reset...");
        delay(10000);
        resetThingSpeakField();

        lastPayload = "0";
       }
       else {
        lastPayload = payload;
       }

    } else {
     Serial.println("Gagal ambil data! HTTP Code: " + String(httpCode));
    }

    http.end();
  }
}

void jalankanServoRelay() {
  Serial.println("=== STARTING SERVO + RELAY SEQUENCE ===");

  // Aktifkan relay (aktif-low → LOW)
  digitalWrite(relayPin, LOW);
  Serial.println("□ RELAY: ON");

  myservo.write(180);
  Serial.println("□ SERVO: Moving to 180°");

  delay(5000);

  myservo.write(0);
  Serial.println("□ SERVO: Moving back to 0°");

  delay(1000);

  // Matikan relay (aktif-low → HIGH)
  digitalWrite(relayPin, HIGH);
  Serial.println("□ RELAY: OFF");

  Serial.println("=== SEQUENCE COMPLETED ===");
}

void resetThingSpeakField() {
  if (WiFi.status() == WL_CONNECTED) {
```

```
  WiFiClient client;
  HTTPClient httpWrite;
  String url = String(writeServer) + "?api_key=" + String(WRITE_API_KEY) +
"&field1=0";

  Serial.println("=== DEBUG RESET INFO ===");
  Serial.println("Write Server: " + String(writeServer));
  Serial.println("Write API Key: " + String(WRITE_API_KEY));
  Serial.println("Channel ID: " + String(CHANNEL_ID));
  Serial.println("Full URL: " + url);
  Serial.println("========================");

  Serial.println("Mencoba reset field1...");
  httpWrite.begin(client, url);
  httpWrite.addHeader("User-Agent", "ESP8266");
  int httpCodeWrite = httpWrite.GET();

  Serial.print("Reset HTTP Response Code: ");
  Serial.println(httpCodeWrite);

  if (httpCodeWrite == 200) {
    String response = httpWrite.getString();
    Serial.println("□ Field1 berhasil direset ke 0");
    Serial.println("Response: " + response);
    delay(3000);
    verifyReset();
  } else {
    String response = httpWrite.getString();
    Serial.println("□ Gagal reset field1, HTTP Code: " + String(httpCodeWrite));
    Serial.println("Error Response: " + response);

    delay(2000);
    Serial.println("Trying alternative URL format...");

    String altUrl = "https://api.thingspeak.com/update?api_key=" +
String(WRITE_API_KEY) + "&field1=0";
    Serial.println("Alt URL: " + altUrl);

    httpWrite.begin(client, altUrl);
    httpWrite.addHeader("User-Agent", "ESP8266");
    int altHttpCode = httpWrite.GET();
    Serial.println("Alternative HTTP Code: " + String(altHttpCode));

    if (altHttpCode == 200) {
      String altResponse = httpWrite.getString();
      Serial.println("□ Alternative method SUCCESS!");
      Serial.println("Response: " + altResponse);
    } else {
```

```
      String altResponse = httpWrite.getString();
      Serial.println("□ Alternative method also failed");
      Serial.println("Alt Response: " + altResponse);
     }
    }
    httpWrite.end();
  } else {
    Serial.println("WiFi disconnected, cannot reset field1");
  }
}

void verifyReset() {
  Serial.println("Verifying field reset...");
  WiFiClient client;
  HTTPClient http;

  http.begin(client, readServer.c_str());
  int httpCode = http.GET();

  if (httpCode == 200) {
    String payload = http.getString();
    payload.trim();
    Serial.println("Verification: Field1 = " + payload);

    if (payload == "0") {
      Serial.println("□ Reset verification SUCCESS");
    } else {
      Serial.println("□  Reset verification FAILED - field still = " + payload);
    }
  }
  http.end();
}
```