

PENGEMBANGAN PROYEK HIDROPONIK DENGAN METODE WICK MENGGUNAKAN NODEMCU ESP8266 DAN SENSOR DHT11



Disusun oleh:

(12) Aziz Hadi Wicaksono (11 SIJA A)

(16) Dhea Peni Nadine Shafira (11 SIJA A)

(20) Fadhirahman Adzdzaky (11 SIJA A)

(25) Gunawan Wibisana (11 SIJA A)

(30) Jonas Adrian Pradana (11 SIJA A)

(35) Lutfiansyah Aufa Windra (11 SIJA A)

**SMK NEGERI 2 DEPOK SLEMAN**

Alamat : Jl. STM Pembangunan, Mrican, Caturtunggal, Kec. Depok, Kebupaten

Sleman, Daerah Istimewa Yogyakarta 55281

*Telp: 0274 - 513515*

1. Lingkup Proyek
2. Tujuan : Menciptakan inovasi monitoring hidroponik wick dengan iot
3. Hasil : Dapat melihat kelembaban udara, banyaknya air, suhu disekitar, dan pengisian air secara otomatis
4. Alokasi Waktu : 1865 Menit
5. Alokasi Anggaran : Rp 463.000,00
6. Desain Produk/Jasa :



*Gambar 1. Desain Produk*

1. Alat dan Bahan
2. Alat

| **No** | **Alat** | **Spesifikasi** | **Kebutuhan** |
| --- | --- | --- | --- |
| 1. | Water Sensor | FC-28 atau YL-69 | Mengukur tingkat kelembaban air atau volume air |
| 2. | Sensor DHT | DHT11 | Mengukur suhu dan kelembapan udara |
| 3. | Pompa Mini | Pompa Mini 3V | Mengalirkan larutan nutrisi ke tanaman |
| 4. | ESP8266 | NodeMCU | Mikrokontroler untuk kontrol otomatis dan IoT |
| 5. | Relay | 1 channel 5V relay | Menghidupkan/mematikan pompa secara otomatis |
| 6. | Jumper | Female to Female  Male to Female  Male to Male | Menghubungkan komponen pada breadboard atau langsung |
| 7 | Smartphone | Xiomi 14 | Untuk mengontrol dan memantau sistem via WiFi |

1. Bahan

| **No** | **Bahan** | **Spesifikasi** | **Kebutuhan** |
| --- | --- | --- | --- |
| 1. | Lem Tembak | - | Merekatkan komponen atau menyegel wadah |
| 2. | Sterofom | - | Penyangga netpot   |  | | --- | |
| 3. | AB Mix | - | Memberi nutrisi pada tanaman |
| 4. | Rockwool | - | Media tanam tempat bibit tumbuh |
| 5. | Kabel USB | - | Menyalakan ESP8266 atau upload program |
| 6. | Netpot | - | Menampung  Rockwool |
| 7. | Kain Flanel | - | Sebagai Penyalur Air ke Rockwool |

1. Tolok Ukur Keberhasilan:
2. Sensor dapat berfungsi seperti yang diinginkan
3. Pengisian dan pemantauan level air berfungsi seperti yang diinginkan
4. Sensor dapat dimonitoring dengan NodeRed maupun MIT
5. Daftar Kegiatan

| No | Kegiatan | Deskripsi | Alokasi  Waktu |
| --- | --- | --- | --- |
| 1. | Membuat rancangan produk | Membuat desain produk | 120 Menit |
| 2. | Membuat desain wiring | Membuat desain wiring di fritzing | 60 Menit |
| 3 | Membeli alat dan bahan | Membeli alat dan bahan yang dibutuhkan | 30 Menit |
| 4. | Menyiapkan alat dan bahan | Menyiapkan alat dan bahan yang akan digunakan | 60 Menit |
| 5. | Mengimplementasikan desain wiring | Merangkai wiring sesuai desain yang telah dibuat | 30 Menit |
| 6. | Upload kode ke ESP8266 dengan menghubungkan micro USB ke ESP8266 | Upload code ke ESP8266 menggunakan Arduino IDE | 5 Menit |
| 7. | Membuat kontrol aplikasi | Membuat aplikasi kontrol dean MIT App Inventor untuk monitoring suhu | 180 Menit |
| 8. | Mengimplementasikan komponen IOT ke hidroponik | Memasukan komponen beserta wiringnya ke wadah hidroponik dan menerapkan IOT Hidroponik | 120 Menit |
| 9. | Uji coba produk dan evaluasi | Uji coba produk hidroponik | 60 Menit |
| 10. | Finishing | Menyelesaikan Produk Hidroponik | 30 Menit |
| Total | | | 1865 Menit |

1. Jadwal Kegiatan

| No | Kegiatan | Hari | Tanggal | Jam |
| --- | --- | --- | --- | --- |
| 1. | Membuat rancangan produk | Senin | 28 April 2025 | 08.00 - 10.30 |
| 2. | Membuat desain wiring | Rabu | 30 April 2025 | 08.30 - 09.30 |
| 3. | Membeli alat dan bahan | Sabtu | 10 Mei 2025 | 15.00 - 15.30 |
| 4. | Menyiapkan alat dan bahan | Rabu | 14 Mei 2025 | 08.00 - 09.00 |
| 5. | Mengimplementasikan desain wiring | Rabu | 14 Mei 2025 | 09.30 - 10.00 |
| 6. | Upload kode ke ESP8266 dengan menghubungkan micro USB ke ESP8266 | Rabu | 14 Mei 2025 | 10.20 - 10.25 |
| 7. | Memasukan wiring ke wadah hidroponik | Rabu | 14 Mei 2025 | 10.25 - 10.40 |
| 8. | Membuat aplikasi kontrol | Jumat | 16 Mei 2025 | 19.00 - 22.00 |
| 10. | Uji coba produk dan evaluasi | Jumat | 23 Mei 2025 | 11.45 - 12.45 |
| 11. | Finishing | Jumat | 23 Mei 2025 | 13.00 - 13.20 |

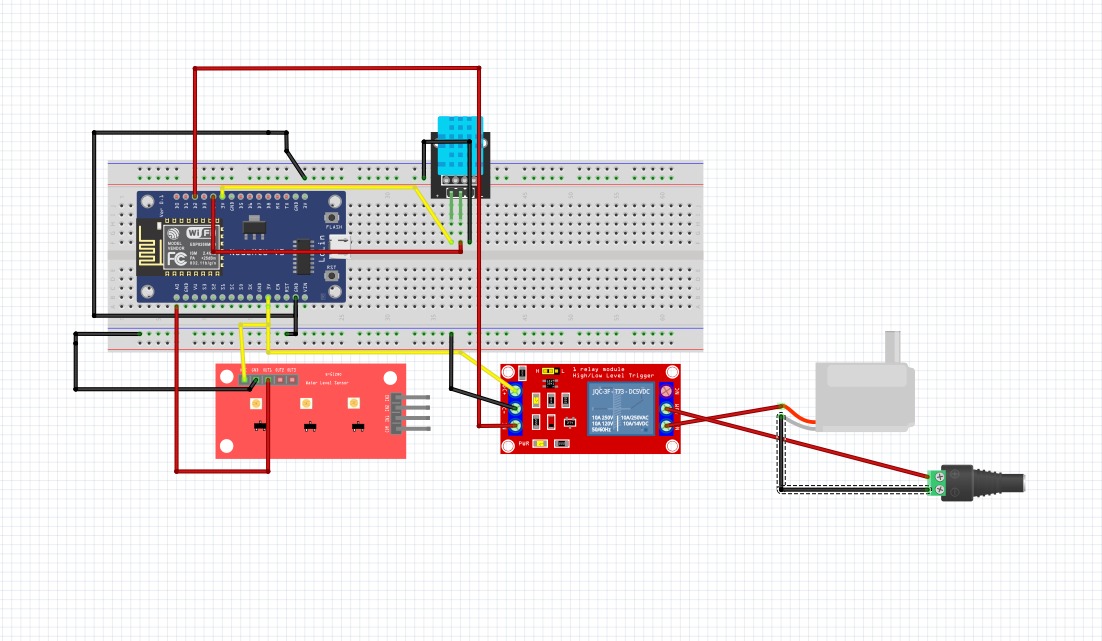
1. Sumber daya

Kompetensi yang dibutuhkan:

1. Sarana
2. Pembibit
3. Programer
4. Anggaran

| No | Komponen | Satuan | Harga Satuan  (Rp) | Jumlah | Harga |
| --- | --- | --- | --- | --- | --- |
| 1. | Pompa Mini | Item | Rp. 10.000 | 1 | Rp. 10.000 |
| 2. | ESP8266 | Item | Rp. 30.000 | 1 | Rp. 30.000 |
| 3. | Relay Single Channel | Item | Rp. 5.000 | 2 | Rp. 10.000 |
| 4. | Kabel Jumper | Item | Rp. 200 | 50 | Rp. 10.000 |
| 5. | Half Breadboard | Item | Rp. 10.000 | 1 | Rp. 10.000 |
| 6. | Ember Segi Empat | Item | Rp. 10.000 | 2 | Rp. 20.000 |
| 7. | AB Mix(50Ml) | Item | Rp. 5.000 | 2 | Rp. 10.000 |
| 8. | DHT11 Sensor | Item | Rp. 15.000 | 1 | Rp. 15.000 |
| 9. | Water Sensor | Item | Rp. 15.000 | 1 | Rp. 15.000 |
| 10. | Kabel Colokan | Item | Rp. 10.000 | 1 | Rp. 10.000 |
| 11. | Micro USB | Item | Rp. 18.000 | 1 | Rp. 18.000 |
| 12. | Kotak Listrik | Item | Rp. 10.000 | 1 | Rp. 10.000 |
| 13. | Converter Plug to Cable | Item | Rp. 2.000 | 1 | Rp 2.000 |
| 14. | Kepala Charger | Item | Gratis | 1 | Gratis |
| 15. | Kepala 12 Volt | Item | Rp. 10.000 | 1 | Rp. 10.000 |
|  | | | | | Rp. 180.000 |

1. Wiring
2. Desain Wiring



1. ESP8266 ke sensor DHT11

| **ESP8266** | **DHT11** |
| --- | --- |
| 3V | VCC |
| GND | GND |
| D4 | DATA |

1. ESP8266 ke sensor water level

| **ESP8266** | **Water Level** |
| --- | --- |
| 3V | VCC |
| GND | GND |
| A0 | S (Signal) |

1. ESP8266 ke relay

| **ESP8266** | **Relay** |
| --- | --- |
| 3V | VCC |
| GND | GND |
| D2 | IN |

1. Relay ke pompa

| **Relay** | **Pompa** |
| --- | --- |
| NO | + |
| power | - |

1. Sumber 5V to Relay

| **5V** | **Relay** |
| --- | --- |
| + | COM |

1. Sumber 5V to Pompa

| **5V** | **POMPA** |
| --- | --- |
| GND(-) | GND(-) |

1. Code program

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#include <DHT.h>

// ====== Konfigurasi WiFi ======

const char\* ssid = "SIJA-PNP";

const char\* wifiPass = "12345678";

// ====== Konfigurasi MQTT ======

const char\* mqtt\_server = "10.20.11.11";

const int mqtt\_port = 1883;

const char\* mqtt\_user = "onlykelompok4canusethisuser";

const char\* mqtt\_pass = "#123@#4";

// ====== Topik MQTT ======

const char\* topic\_water\_level = "kel4/water\_level";

const char\* topic\_temperature = "kel4/suhu";

const char\* topic\_humidity = "kel4/kelembapan";

const char\* topic\_pump\_control = "kel4/pompa/control"; // ON/OFF manual

const char\* topic\_pump\_status = "kel4/pompa/status";

const char\* topic\_mode\_control = "kel4/mode/control"; // AUTO/MANUAL

const char\* topic\_mode\_status = "kel4/mode/status";

const char\* topic\_threshold\_set = "kel4/threshold/set"; // format: "25,75"

const char\* topic\_threshold\_get = "kel4/threshold/status";

const char\* topic\_system\_status = "kel4/system/status"; // JSON status lengkap

// ====== Pin Konfigurasi ======

#define WATER\_LEVEL\_PIN A0 // Sensor level air ke A0

#define DHT\_PIN D4 // DHT11 data ke D4

#define RELAY\_PIN D2 // Relay pompa ke D2

// ====== Inisialisasi DHT ======

#define DHTTYPE DHT11

DHT dht(DHT\_PIN, DHTTYPE);

// ====== Global vars ======

WiFiClient espClient;

PubSubClient client(espClient);

// ====== Timing ======

unsigned long lastMsg = 0;

unsigned long pumpStartTime = 0;

unsigned long pumpStopTime = 0;

const long interval = 5000; // Interval publish sensor (5 detik)

const long maxPumpRunTime = 480000; // 8 menit (480 detik) 480000

const long minPumpRestTime = 10; // 5 menit (300 detik) 300000

// ====== System State ======

enum PumpMode { AUTO, MANUAL };

PumpMode currentMode = AUTO;

bool pumpState = false;

bool manualPumpCommand = false;

// ====== Threshold Settings ======

int thresholdLow = 25; // Pompa ON jika water level < 25%

int thresholdHigh = 75; // Pompa OFF jika water level > 75%

// ====== Safety Flags ======

bool sensorError = false;

bool pumpOverTime = false;

bool pumpInRestPeriod = false;

void setup\_wifi() {

delay(10);

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, wifiPass);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println();

Serial.print("WiFi connected, IP: ");

Serial.println(WiFi.localIP());

}

void callback(char\* topic, byte\* payload, unsigned int length) {

String msg;

for (unsigned int i = 0; i < length; i++) {

msg += char(payload[i]);

}

msg.toLowerCase();

msg.trim();

Serial.println("MQTT Received - Topic: " + String(topic) + " | Message: " + msg);

// ====== Kontrol Mode ======

if (String(topic) == topic\_mode\_control) {

if (msg == "auto") {

currentMode = AUTO;

Serial.println("Mode changed to: AUTO");

} else if (msg == "manual") {

currentMode = MANUAL;

Serial.println("Mode changed to: MANUAL");

}

publishModeStatus();

}

// ====== Kontrol Pompa Manual ======

else if (String(topic) == topic\_pump\_control) {

if (currentMode == MANUAL) {

if (msg == "on" || msg == "1" || msg == "true" || msg == "nyala") {

manualPumpCommand = true;

Serial.println("Manual Pump Command: ON");

} else if (msg == "off" || msg == "0" || msg == "false" || msg == "mati") {

manualPumpCommand = false;

Serial.println("Manual Pump Command: OFF");

}

} else {

Serial.println("Pump control ignored - System in AUTO mode");

}

}

// ====== Set Threshold ======

else if (String(topic) == topic\_threshold\_set) {

int commaIndex = msg.indexOf(',');

if (commaIndex > 0) {

int newLow = msg.substring(0, commaIndex).toInt();

int newHigh = msg.substring(commaIndex + 1).toInt();

if (newLow > 0 && newHigh > newLow && newHigh <= 100) {

thresholdLow = newLow;

thresholdHigh = newHigh;

Serial.println("Threshold updated - Low: " + String(thresholdLow) + "% | High: " + String(thresholdHigh) + "%");

publishThresholdStatus();

} else {

Serial.println("Invalid threshold format. Use: low,high (e.g., 25,75)");

}

}

}

}

void controlPump(bool state) {

if (state && !pumpState) {

// Cek apakah pompa masih dalam periode istirahat

if (pumpInRestPeriod && (millis() - pumpStopTime < minPumpRestTime)) {

Serial.println("Pump in rest period. Cannot start yet.");

return;

}

digitalWrite(RELAY\_PIN, LOW); // LOW = ON

pumpState = true;

pumpStartTime = millis();

pumpInRestPeriod = false;

pumpOverTime = false;

Serial.println("Pompa: ON");

} else if (!state && pumpState) {

digitalWrite(RELAY\_PIN, HIGH); // HIGH = OFF

pumpState = false;

pumpStopTime = millis();

pumpInRestPeriod = true;

Serial.println("Pompa: OFF");

}

}

void checkPumpSafety() {

// Cek apakah pompa sudah menyala terlalu lama

if (pumpState && (millis() - pumpStartTime > maxPumpRunTime)) {

controlPump(false);

pumpOverTime = true;

Serial.println("SAFETY: Pump stopped - Maximum runtime exceeded!");

}

}

void autoModeLogic(int waterLevel) {

if (sensorError) {

Serial.println("AUTO Mode disabled - Sensor error");

return;

}

if (pumpOverTime && waterLevel < thresholdHigh) {

Serial.println("AUTO Mode suspended - Pump overtime, waiting for rest period");

return;

}

// Hysteresis control untuk mencegah rapid cycling

if (!pumpState && waterLevel < thresholdLow) {

controlPump(true);

Serial.println("AUTO: Water level low (" + String(waterLevel) + "%) - Starting pump");

}

else if (pumpState && waterLevel > thresholdHigh) {

controlPump(false);

Serial.println("AUTO: Water level sufficient (" + String(waterLevel) + "%) - Stopping pump");

}

}

void publishModeStatus() {

client.publish(topic\_mode\_status, currentMode == AUTO ? "auto" : "manual");

}

void publishThresholdStatus() {

String thresholdMsg = String(thresholdLow) + "," + String(thresholdHigh);

client.publish(topic\_threshold\_get, thresholdMsg.c\_str());

}

void publishSystemStatus(float temp, float humidity, int waterLevel) {

// Publish JSON status lengkap

String systemStatus = "{";

systemStatus += "\"mode\":\"" + String(currentMode == AUTO ? "auto" : "manual") + "\",";

systemStatus += "\"pump\":\"" + String(pumpState ? "on" : "off") + "\",";

systemStatus += "\"water\_level\":" + String(waterLevel) + ",";

systemStatus += "\"temperature\":" + String(temp, 1) + ",";

systemStatus += "\"humidity\":" + String(humidity, 1) + ",";

systemStatus += "\"threshold\_low\":" + String(thresholdLow) + ",";

systemStatus += "\"threshold\_high\":" + String(thresholdHigh) + ",";

systemStatus += "\"sensor\_error\":" + String(sensorError ? "true" : "false") + ",";

systemStatus += "\"pump\_overtime\":" + String(pumpOverTime ? "true" : "false") + ",";

systemStatus += "\"rest\_period\":" + String(pumpInRestPeriod ? "true" : "false");

systemStatus += "}";

client.publish(topic\_system\_status, systemStatus.c\_str());

}

void reconnect() {

while (!client.connected()) {

Serial.print("Connecting to MQTT...");

if (String(mqtt\_user).length() == 0) {

if (client.connect("ESP8266Client")) {

Serial.println("connected");

}

} else {

if (client.connect("ESP8266Client", mqtt\_user, mqtt\_pass)) {

Serial.println("connected with auth");

}

}

if (client.connected()) {

// Subscribe ke semua topic control

client.subscribe(topic\_pump\_control);

client.subscribe(topic\_mode\_control);

client.subscribe(topic\_threshold\_set);

// Publish status awal

publishModeStatus();

publishThresholdStatus();

} else {

Serial.print("failed rc=");

Serial.print(client.state());

Serial.println(" retry in 5s");

delay(5000);

}

}

}

void setup() {

Serial.begin(115200);

pinMode(RELAY\_PIN, OUTPUT);

digitalWrite(RELAY\_PIN, HIGH); // Relay OFF saat start

dht.begin();

setup\_wifi();

client.setServer(mqtt\_server, mqtt\_port);

client.setCallback(callback);

Serial.println("=== Smart Water Pump Controller ===");

Serial.println("Mode: AUTO");

Serial.println("Threshold Low: " + String(thresholdLow) + "%");

Serial.println("Threshold High: " + String(thresholdHigh) + "%");

Serial.println("Setup complete");

}

void loop() {

if (!client.connected()) {

reconnect();

}

client.loop();

// Cek safety pompa

checkPumpSafety();

unsigned long now = millis();

if (now - lastMsg > interval) {

lastMsg = now;

// ====== Baca Sensor Water Level ======

int rawLevel = analogRead(WATER\_LEVEL\_PIN);

int levelPct = map(constrain(rawLevel, 0, 800), 0, 800, 0, 100);

// Deteksi sensor error (nilai tidak masuk akal)

sensorError = (rawLevel < 5 || rawLevel > 1020);

// ====== Baca DHT11 ======

float h = dht.readHumidity();

float t = dht.readTemperature();

if (isnan(h) || isnan(t)) {

Serial.println("Failed reading DHT!");

h = 0; t = 0;

}

// ====== Logic Control ======

if (currentMode == AUTO && !sensorError) {

autoModeLogic(levelPct);

} else if (currentMode == MANUAL) {

controlPump(manualPumpCommand);

}

// ====== Publish Data ======

if (!sensorError) {

Serial.printf("Mode: %s | Humidity: %.1f%% | Temp: %.1f°C | Water: %d%% | Pump: %s\n",

currentMode == AUTO ? "AUTO" : "MANUAL", h, t, levelPct, pumpState ? "ON" : "OFF");

// Konversi ke string

char bufT[8], bufH[8], bufL[8];

dtostrf(t, 1, 2, bufT);

dtostrf(h, 1, 2, bufH);

dtostrf(levelPct, 1, 2, bufL);

// Publish sensor data

client.publish(topic\_temperature, bufT);

client.publish(topic\_humidity, bufH);

client.publish(topic\_water\_level, bufL);

} else {

Serial.println("SENSOR ERROR - Water level sensor malfunction!");

}

// Publish status

client.publish(topic\_pump\_status, pumpState ? "on" : "off");

publishSystemStatus(t, h, sensorError ? -1 : levelPct);

}

}

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#include <DHT.h>

// ====== Konfigurasi WiFi ======

const char\* ssid = "SIJA-PNP";

const char\* wifiPass = "12345678";

// ====== Konfigurasi MQTT ======

const char\* mqtt\_server = "10.20.11.11";

const int mqtt\_port = 1883;

const char\* mqtt\_user = "onlykelompok4canusethisuser";

const char\* mqtt\_pass = "#123@#4";

// ====== Topik MQTT ======

const char\* topic\_water\_level = "kel4/water\_level";

const char\* topic\_temperature = "kel4/suhu";

const char\* topic\_humidity = "kel4/kelembapan";

const char\* topic\_pump\_control = "kel4/pompa/control"; // ON/OFF manual

const char\* topic\_pump\_status = "kel4/pompa/status";

const char\* topic\_mode\_control = "kel4/mode/control"; // AUTO/MANUAL

const char\* topic\_mode\_status = "kel4/mode/status";

const char\* topic\_threshold\_set = "kel4/threshold/set"; // format: "25,75"

const char\* topic\_threshold\_get = "kel4/threshold/status";

const char\* topic\_system\_status = "kel4/system/status"; // JSON status lengkap

// ====== Pin Konfigurasi ======

#define WATER\_LEVEL\_PIN A0 // Sensor level air ke A0

#define DHT\_PIN D4 // DHT11 data ke D4

#define RELAY\_PIN D2 // Relay pompa ke D2

// ====== Inisialisasi DHT ======

#define DHTTYPE DHT11

DHT dht(DHT\_PIN, DHTTYPE);

// ====== Global vars ======

WiFiClient espClient;

PubSubClient client(espClient);

// ====== Timing ======

unsigned long lastMsg = 0;

unsigned long pumpStartTime = 0;

unsigned long pumpStopTime = 0;

const long interval = 5000; // Interval publish sensor (5 detik)

const long maxPumpRunTime = 480000; // 8 menit (480 detik) 480000

const long minPumpRestTime = 10; // 5 menit (300 detik) 300000

// ====== System State ======

enum PumpMode { AUTO, MANUAL };

PumpMode currentMode = AUTO;

bool pumpState = false;

bool manualPumpCommand = false;

// ====== Threshold Settings ======

int thresholdLow = 25; // Pompa ON jika water level < 25%

int thresholdHigh = 75; // Pompa OFF jika water level > 75%

// ====== Safety Flags ======

bool sensorError = false;

bool pumpOverTime = false;

bool pumpInRestPeriod = false;

void setup\_wifi() {

delay(10);

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, wifiPass);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println();

Serial.print("WiFi connected, IP: ");

Serial.println(WiFi.localIP());

}

void callback(char\* topic, byte\* payload, unsigned int length) {

String msg;

for (unsigned int i = 0; i < length; i++) {

msg += char(payload[i]);

}

msg.toLowerCase();

msg.trim();

Serial.println("MQTT Received - Topic: " + String(topic) + " | Message: " + msg);

// ====== Kontrol Mode ======

if (String(topic) == topic\_mode\_control) {

if (msg == "auto") {

currentMode = AUTO;

Serial.println("Mode changed to: AUTO");

} else if (msg == "manual") {

currentMode = MANUAL;

Serial.println("Mode changed to: MANUAL");

}

publishModeStatus();

}

// ====== Kontrol Pompa Manual ======

else if (String(topic) == topic\_pump\_control) {

if (currentMode == MANUAL) {

if (msg == "on" || msg == "1" || msg == "true" || msg == "nyala") {

manualPumpCommand = true;

Serial.println("Manual Pump Command: ON");

} else if (msg == "off" || msg == "0" || msg == "false" || msg == "mati") {

manualPumpCommand = false;

Serial.println("Manual Pump Command: OFF");

}

} else {

Serial.println("Pump control ignored - System in AUTO mode");

}

}

// ====== Set Threshold ======

else if (String(topic) == topic\_threshold\_set) {

int commaIndex = msg.indexOf(',');

if (commaIndex > 0) {

int newLow = msg.substring(0, commaIndex).toInt();

int newHigh = msg.substring(commaIndex + 1).toInt();

if (newLow > 0 && newHigh > newLow && newHigh <= 100) {

thresholdLow = newLow;

thresholdHigh = newHigh;

Serial.println("Threshold updated - Low: " + String(thresholdLow) + "% | High: " + String(thresholdHigh) + "%");

publishThresholdStatus();

} else {

Serial.println("Invalid threshold format. Use: low,high (e.g., 25,75)");

}

}

}

}

void controlPump(bool state) {

if (state && !pumpState) {

// Cek apakah pompa masih dalam periode istirahat

if (pumpInRestPeriod && (millis() - pumpStopTime < minPumpRestTime)) {

Serial.println("Pump in rest period. Cannot start yet.");

return;

}

digitalWrite(RELAY\_PIN, LOW); // LOW = ON

pumpState = true;

pumpStartTime = millis();

pumpInRestPeriod = false;

pumpOverTime = false;

Serial.println("Pompa: ON");

} else if (!state && pumpState) {

digitalWrite(RELAY\_PIN, HIGH); // HIGH = OFF

pumpState = false;

pumpStopTime = millis();

pumpInRestPeriod = true;

Serial.println("Pompa: OFF");

}

}

void checkPumpSafety() {

// Cek apakah pompa sudah menyala terlalu lama

if (pumpState && (millis() - pumpStartTime > maxPumpRunTime)) {

controlPump(false);

pumpOverTime = true;

Serial.println("SAFETY: Pump stopped - Maximum runtime exceeded!");

}

}

void autoModeLogic(int waterLevel) {

if (sensorError) {

Serial.println("AUTO Mode disabled - Sensor error");

return;

}

if (pumpOverTime && waterLevel < thresholdHigh) {

Serial.println("AUTO Mode suspended - Pump overtime, waiting for rest period");

return;

}

// Hysteresis control untuk mencegah rapid cycling

if (!pumpState && waterLevel < thresholdLow) {

controlPump(true);

Serial.println("AUTO: Water level low (" + String(waterLevel) + "%) - Starting pump");

}

else if (pumpState && waterLevel > thresholdHigh) {

controlPump(false);

Serial.println("AUTO: Water level sufficient (" + String(waterLevel) + "%) - Stopping pump");

}

}

void publishModeStatus() {

client.publish(topic\_mode\_status, currentMode == AUTO ? "auto" : "manual");

}

void publishThresholdStatus() {

String thresholdMsg = String(thresholdLow) + "," + String(thresholdHigh);

client.publish(topic\_threshold\_get, thresholdMsg.c\_str());

}

void publishSystemStatus(float temp, float humidity, int waterLevel) {

// Publish JSON status lengkap

String systemStatus = "{";

systemStatus += "\"mode\":\"" + String(currentMode == AUTO ? "auto" : "manual") + "\",";

systemStatus += "\"pump\":\"" + String(pumpState ? "on" : "off") + "\",";

systemStatus += "\"water\_level\":" + String(waterLevel) + ",";

systemStatus += "\"temperature\":" + String(temp, 1) + ",";

systemStatus += "\"humidity\":" + String(humidity, 1) + ",";

systemStatus += "\"threshold\_low\":" + String(thresholdLow) + ",";

systemStatus += "\"threshold\_high\":" + String(thresholdHigh) + ",";

systemStatus += "\"sensor\_error\":" + String(sensorError ? "true" : "false") + ",";

systemStatus += "\"pump\_overtime\":" + String(pumpOverTime ? "true" : "false") + ",";

systemStatus += "\"rest\_period\":" + String(pumpInRestPeriod ? "true" : "false");

systemStatus += "}";

client.publish(topic\_system\_status, systemStatus.c\_str());

}

void reconnect() {

while (!client.connected()) {

Serial.print("Connecting to MQTT...");

if (String(mqtt\_user).length() == 0) {

if (client.connect("ESP8266Client")) {

Serial.println("connected");

}

} else {

if (client.connect("ESP8266Client", mqtt\_user, mqtt\_pass)) {

Serial.println("connected with auth");

}

}

if (client.connected()) {

// Subscribe ke semua topic control

client.subscribe(topic\_pump\_control);

client.subscribe(topic\_mode\_control);

client.subscribe(topic\_threshold\_set);

// Publish status awal

publishModeStatus();

publishThresholdStatus();

} else {

Serial.print("failed rc=");

Serial.print(client.state());

Serial.println(" retry in 5s");

delay(5000);

}

}

}

void setup() {

Serial.begin(115200);

pinMode(RELAY\_PIN, OUTPUT);

digitalWrite(RELAY\_PIN, HIGH); // Relay OFF saat start

dht.begin();

setup\_wifi();

client.setServer(mqtt\_server, mqtt\_port);

client.setCallback(callback);

Serial.println("=== Smart Water Pump Controller ===");

Serial.println("Mode: AUTO");

Serial.println("Threshold Low: " + String(thresholdLow) + "%");

Serial.println("Threshold High: " + String(thresholdHigh) + "%");

Serial.println("Setup complete");

}

void loop() {

if (!client.connected()) {

reconnect();

}

client.loop();

// Cek safety pompa

checkPumpSafety();

unsigned long now = millis();

if (now - lastMsg > interval) {

lastMsg = now;

// ====== Baca Sensor Water Level ======

int rawLevel = analogRead(WATER\_LEVEL\_PIN);

int levelPct = map(constrain(rawLevel, 0, 800), 0, 800, 0, 100);

// Deteksi sensor error (nilai tidak masuk akal)

sensorError = (rawLevel < 5 || rawLevel > 1020);

// ====== Baca DHT11 ======

float h = dht.readHumidity();

float t = dht.readTemperature();

if (isnan(h) || isnan(t)) {

Serial.println("Failed reading DHT!");

h = 0; t = 0;

}

// ====== Logic Control ======

if (currentMode == AUTO && !sensorError) {

autoModeLogic(levelPct);

} else if (currentMode == MANUAL) {

controlPump(manualPumpCommand);

}

// ====== Publish Data ======

if (!sensorError) {

Serial.printf("Mode: %s | Humidity: %.1f%% | Temp: %.1f°C | Water: %d%% | Pump: %s\n",

currentMode == AUTO ? "AUTO" : "MANUAL", h, t, levelPct, pumpState ? "ON" : "OFF");

// Konversi ke string

char bufT[8], bufH[8], bufL[8];

dtostrf(t, 1, 2, bufT);

dtostrf(h, 1, 2, bufH);

dtostrf(levelPct, 1, 2, bufL);

// Publish sensor data

client.publish(topic\_temperature, bufT);

client.publish(topic\_humidity, bufH);

client.publish(topic\_water\_level, bufL);

} else {

Serial.println("SENSOR ERROR - Water level sensor malfunction!");

}

// Publish status

client.publish(topic\_pump\_status, pumpState ? "on" : "off");

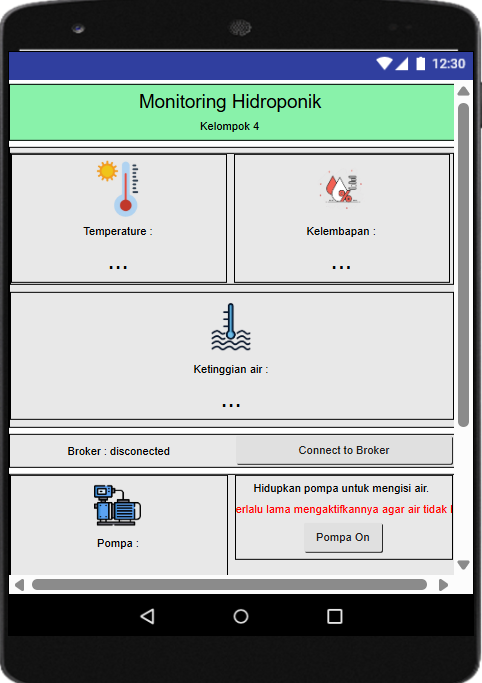
publishSystemStatus(t, h, sensorError ? -1 : levelPct);

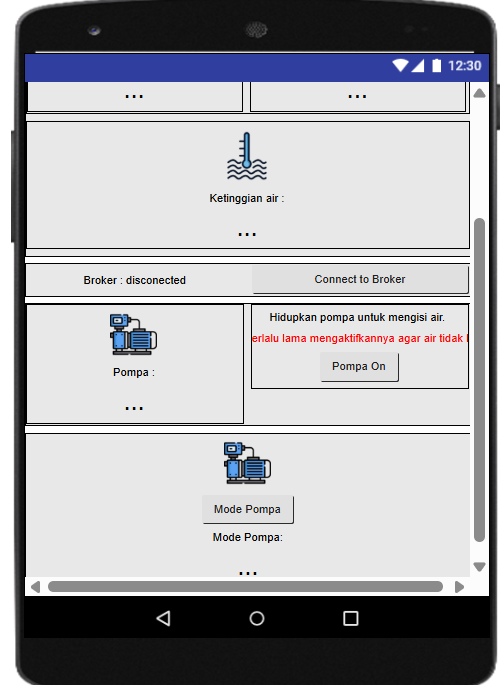
}

}

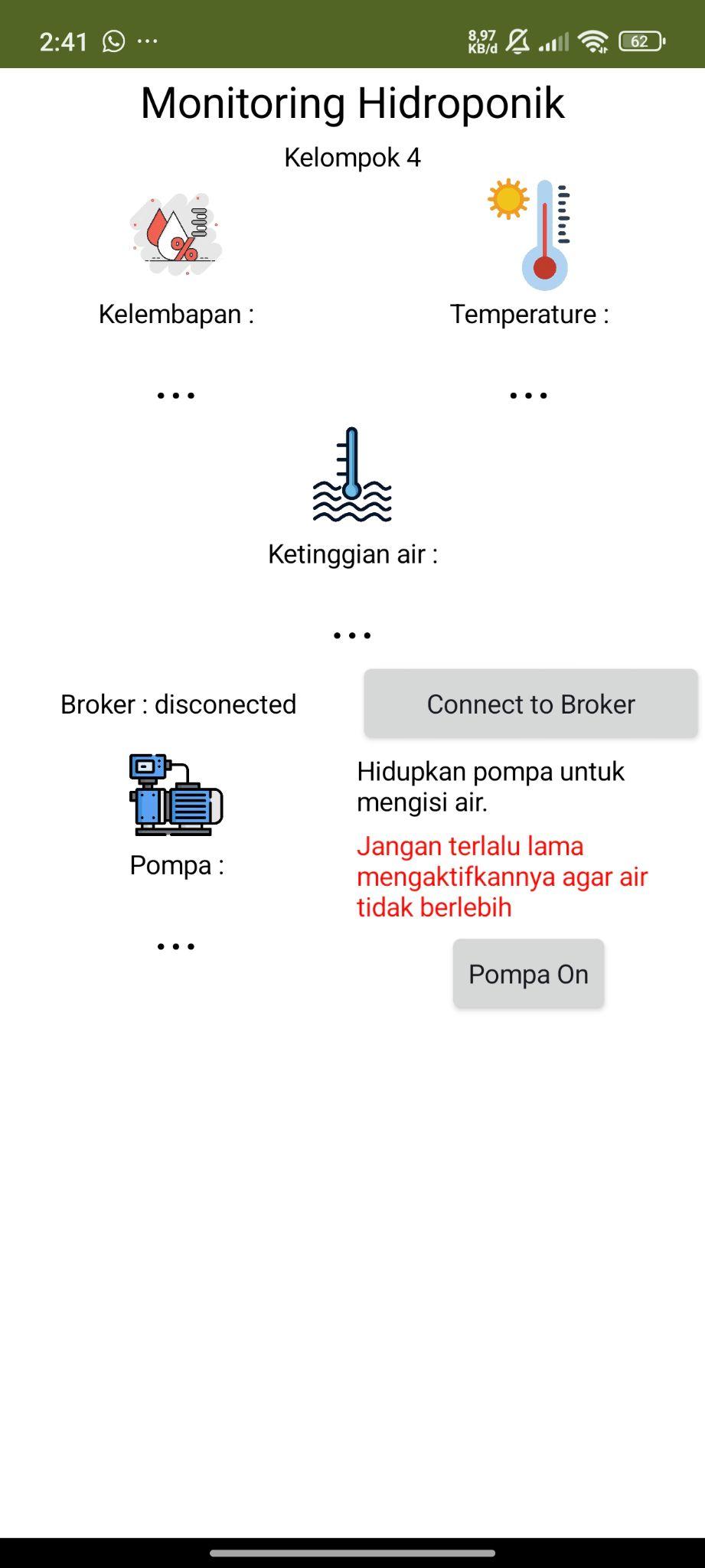
1. Desain App Monitoring (MIT App)

* Desain Di MIT App Inventor

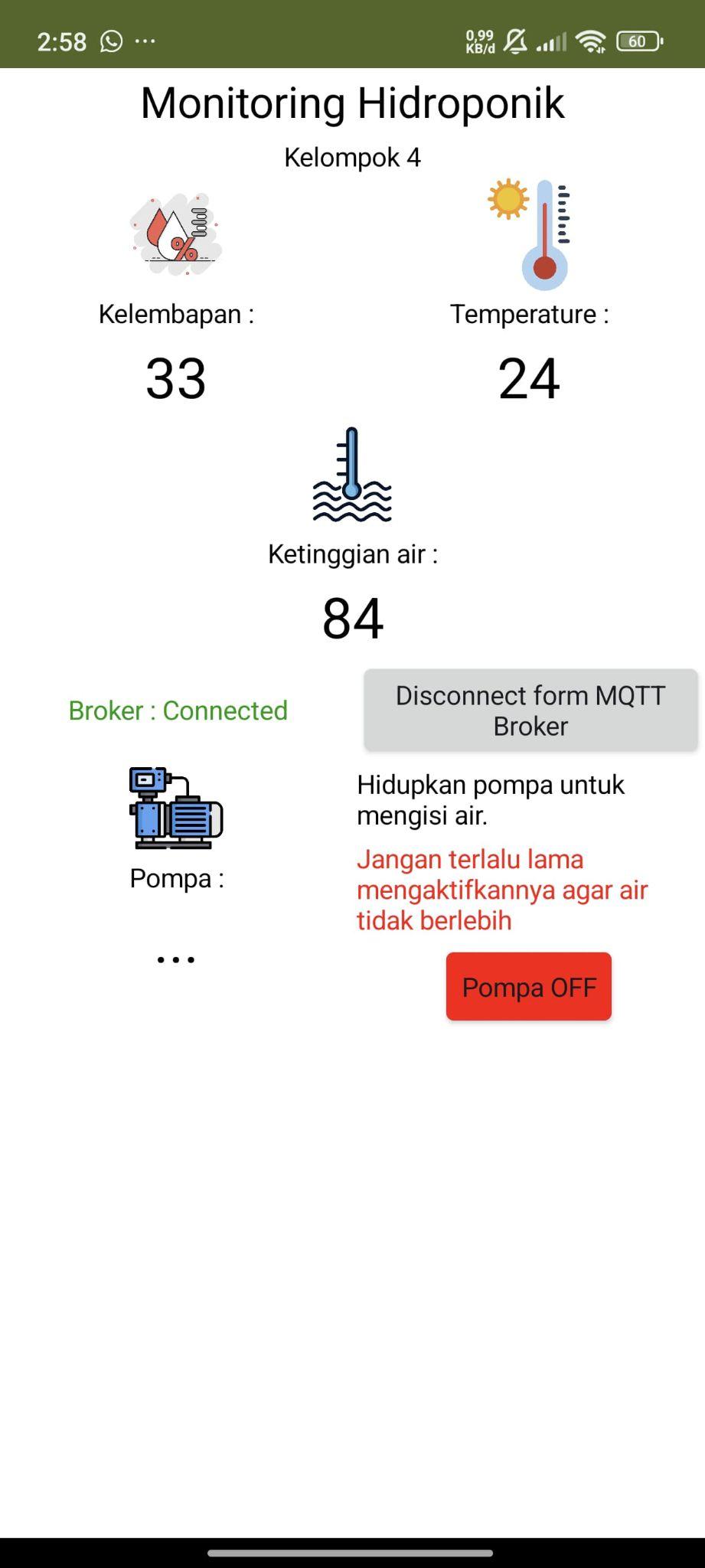




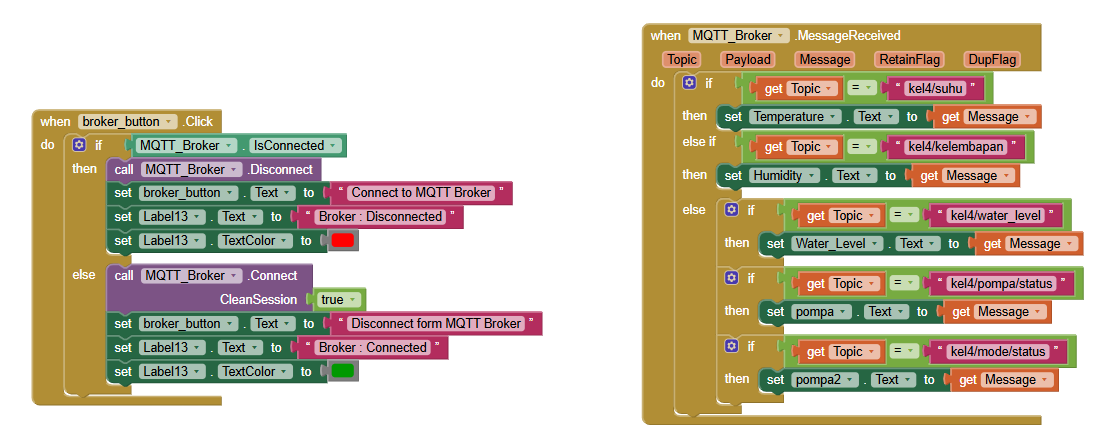
* Tampilan menggunakan handphone

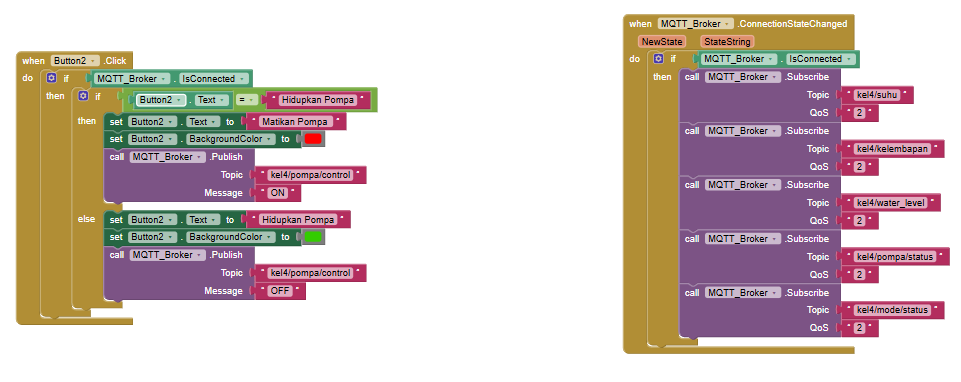


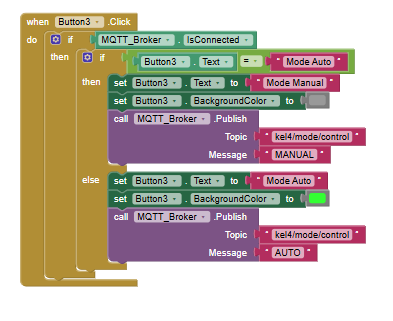
* Saat menerima data di handphone



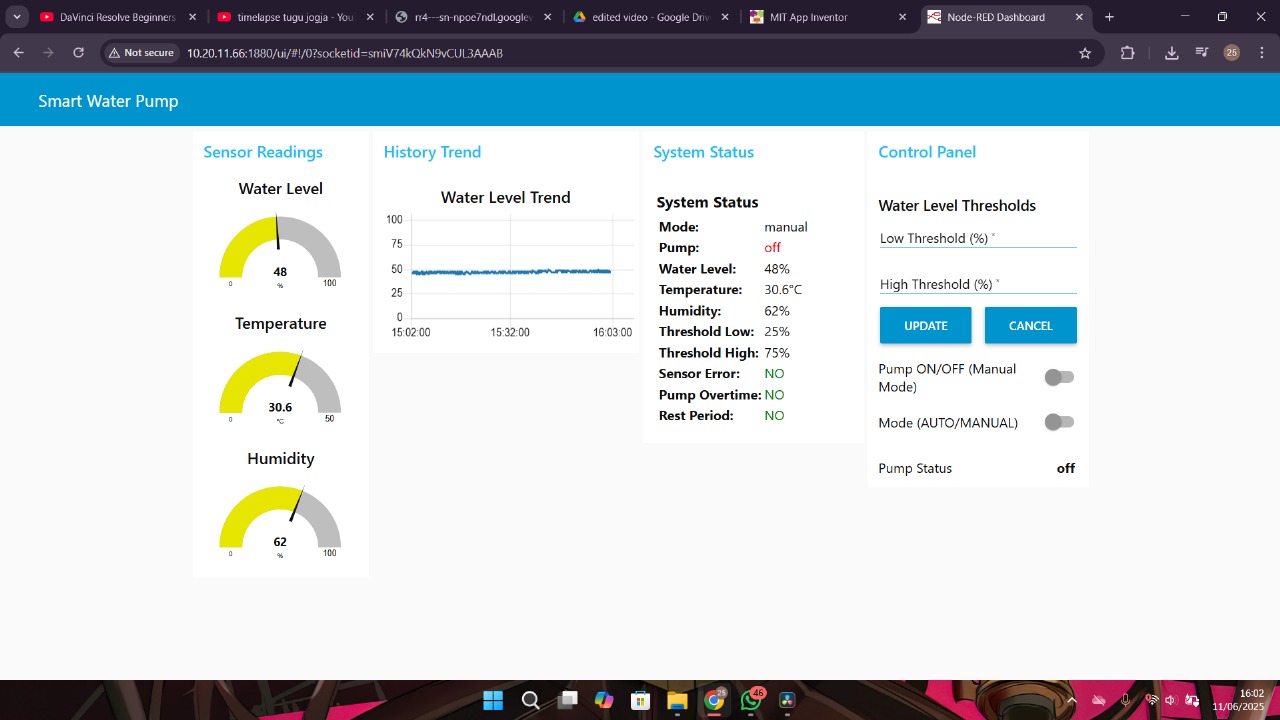
1. Blok App



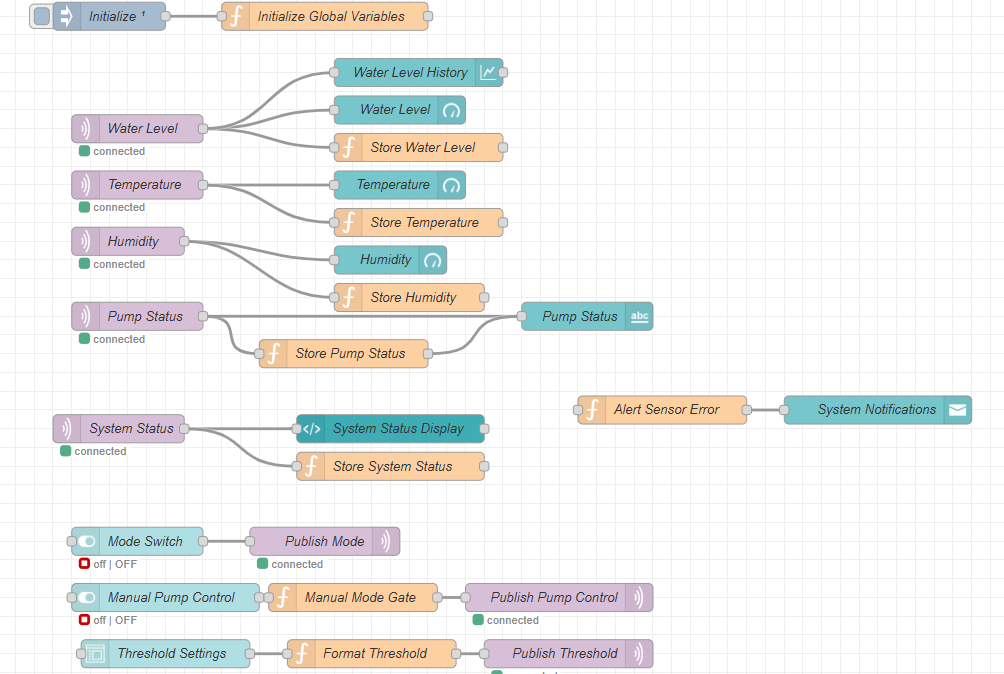




1. Desain Nodered



1. Kode block nodered



1. Protoype / Produk

