LAPORAN MANAJEMEN PROYEK UNTUK TUGAS AKHIR

MATA KULIAH INTERNET OF THINGS (IoT)

Sistem Jemuran Otomatis Berbasis ESP32 dengan Monitoring Real-Time via Web

Dosen Pengampu :

Reza Bagus Acintyasakti, S.Pd., M.T.

Disusun Oleh :

Muhammad Sulthan Alfahrezi (233140707111073)
Jayyid Jidan (2331407077111098)

Hikmal Rajendra (233140707111088)
Gabrielle Christian Sianipar (233140707111096)

PROGRAM STUDI D3 TEKNOLOGI INFORMASI

FAKULTAS VOKASI

UNIVERSITAS BRAWIJAYA

MALANG

2024

LAPORAN MANAJEMEN PROYEK UNTUK TUGAS AKHIR

MATA KULIAH INTERNET OF THINGS (IoT)

Sistem Jemuran Otomatis Berbasis ESP32 dengan Monitoring Real-Time via Web

Program Studi Teknologi Informasi, Fakultas Vokasi, Universitas Brawijaya

Abstrak : Proyek ini mengembangkan sistem jemuran otomatis berbasis ESP32 yang mampu mendeteksi curah hujan dan tingkat cahaya untuk secara otomatis membuka atau menutup jemuran. Sistem ini terintegrasi dengan web melalui backend Express.js dan frontend React.js yang menampilkan status real-time. Selain mode otomatis, sistem juga memiliki mode manual yang memungkinkan pengguna membuka dan menutup jemuran dari dashboard web. Tujuan proyek ini adalah menerapkan teknologi IoT untuk menyelesaikan masalah sehari-hari secara efisien dan dapat dipantau dari jarak jauh.

Abstract : *This project develops an ESP32-based automatic clothesline system that can detect rainfall and light levels to automatically open or close the clothesline. This system is integrated with the web through an Express.js backend and a React.js frontend that displays real-time status. In addition to the automatic mode, the system also has a manual mode that allows users to open and close the clothesline from the web dashboard. The goal of this project is to apply IoT technology to solve everyday problems efficiently and can be monitored remotely.*

BAB I

PENDAHULUAN

1.1 Latar belakang

Aktivitas menjemur pakaian merupakan kegiatan domestik yang sangat umum di Indonesia. Namun, perubahan cuaca yang tidak menentu, terutama saat musim hujan, seringkali menjadi kendala utama dalam proses ini. Tidak jarang, pakaian yang sudah dijemur harus kembali dicuci karena kehujanan secara tiba-tiba. Masalah ini menuntut solusi yang mampu memberikan respons cepat terhadap kondisi lingkungan, tanpa kehadiran atau intervensi langsung dari pengguna.

Seiring perkembangan teknologi, Internet of Things (IoT) menjadi salah satu pendekatan inovatif dalam menyelesaikan permasalahan sehari-hari. Dengan kemampuan menghubungkan perangkat fisik ke internet dan memungkinkan komunikasi dua arah secara otomatis, IoT memiliki potensi besar untuk diterapkan dalam sistem rumah tangga pintar (smart home). Salah satunya adalah dalam pengembangan sistem jemuran otomatis.

Sistem jemuran otomatis berbasis IoT ini tidak hanya bekerja berdasarkan deteksi sensor cuaca, melainkan juga menyediakan fitur kontrol manual melalui antarmuka web. Dengan adanya mode ganda (otomatis dan manual) ini, pengguna memiliki fleksibilitas lebih dalam mengontrol jemuran kapan saja dan di mana saja. Proyek ini menggabungkan beberapa komponen teknologi penting, seperti mikrokontroler ESP32, sensor hujan dan cahaya (LDR), servo motor, serta platform web menggunakan Express.js dan React.js.

1.2 Rumusan Masalah

1. Bagaimana merancang sistem jemuran otomatis yang mampu mendeteksi kondisi cuaca dan merespons secara otomatis?
2. Bagaimana cara menghubungkan ESP32 dengan backend dan frontend web agar data sensor dan kontrol jemuran dapat diakses secara real-time?
3. Bagaimana merancang sistem kontrol manual berbasis web sebagai alternatif ketika sistem otomatis tidak dapat dijalankan?

1.3 Tujuan

- Mengembangkan sistem jemuran otomatis berbasis ESP32 yang dapat membaca sensor hujan dan cahaya.
- Mengintegrasikan sistem dengan backend Express.js dan frontend React.js untuk menampilkan data sensor secara real-time.
- Menyediakan fitur kontrol manual berbasis web yang memungkinkan pengguna membuka atau menutup jemuran dari jarak jauh.
- Menerapkan prinsip dasar manajemen proyek dalam perencanaan dan pelaksanaan sistem IoT.

1.4 Manfaat Proyek

- Meminimalkan risiko pakaian basah akibat cuaca mendadak.
- Memberikan kemudahan dan kenyamanan bagi pengguna dalam mengelola aktivitas menjemur pakaian secara efisien.
- Menunjukkan penerapan nyata teknologi IoT dalam kehidupan sehari-hari.
- Meningkatkan keterampilan teknis dalam pemrograman mikrokontroler, integrasi API, serta pengembangan aplikasi web modern.

BAB II

PERENCANAAN PROYEK

2.1 Jadwal dan Timeline Proyek

TASK	ASSIGNED TO	PROGRESS	START	END
Inisiasi				
Menentukan tujuan utama proyek berdasarkan pengalaman pengguna.	Jayyid Jidan	100%	6/1/25	6/4/25
Membantu dalam diskusi teknis terkait kebutuhan tampilan awal.	Hikmal Rajendra	100%	6/2/25	6/5/25
Menyusun konsep awal arsitektur sistem dan kebutuhan teknis awal backend serta IoT.	Muhammad Sulthan Alfahrezi	100%	6/5/25	6/8/25
Perancanaan				
Membuat jadwal pengembangan backend dan integrasi IoT.	Muhammad Sulthan Alfahrezi	100%	6/8/25	6/9/25
Menentukan kebutuhan server, sensor, serta tools teknis lainnya.	Muhammad Sulthan Alfahrezi	100%	6/8/25	6/9/25
Mendesain wireframe dan prototipe awal UI/UX.	Jayyid Jidan	100%	6/9/25	6/12/25
Menentukan alur interaksi pengguna.	Gabrielle Christian S	100%	6/9/25	6/11/25
Merancang struktur halaman dan navigasi.	Hikmal Rajendra	100%	6/11/25	6/14/25
Menyusun rencana pengembangan frontend berdasarkan desain dari UI/UX.	Hikmal Rajendra	100%	6/12/25	6/15/25
Eksekusi				
Mengembangkan backend API dan logika komunikasi IoT.	Muhammad Sulthan Alfahrezi	100%	6/12/25	6/16/25
Menerapkan koneksi database, sensor, dan logika sistem backend.	Muhammad Sulthan Alfahrezi	100%	6/12/25	6/16/25
Mengembangkan frontend menggunakan framework web/mobile.	Jayyid Jidan	100%	6/13/25	6/18/25
Menghubungkan frontend ke backend dan menyesuaikan dengan UI.	Jayyid Jidan	100%	6/12/25	6/16/25
Memastikan kesesuaian antar UI, alur, dan implementasi.	Gabrielle Christian S	100%	6/13/25	6/15/25
Menyediakan aset UI dan mendampingi integrasi frontend.	Hikmal Rajendra	100%	6/14/25	6/18/25
Evaluasi				
Melakukan pengujian fungsional, integrasi, dan user testing.	Team	100%	6/14/25	6/17/25

Memperbaiki bug yang ditemukan pada frontend, backend, dan sistem IoT.	Team	100%	6/17/25	6/18/25
Menyusun laporan hasil pengujian dan dokumentasi teknis.	Team	75%	6/18/25	6/20/25

2.2 Sumber Daya

1. Hardware:
    a. ESP32 DevKit V4
    b. Sensor Hujan
    c. Sensor LDR
    d. Servo Motor
    e. Breadboard & Kabel Jumper
2. Software
    a. Arduino IDE
    b. Express.js (Node.js)
    c. React.js (Frontend)
    d. Figma (Desain UI)
    e. Postman & GitHub

2.3 Pembagian Tugas

Nama Anggota	**Tugas**
Muhammad Sulthan Al Fahrezi	Backend (Express.js) dan pemrograman ESP32
Jayyid Jiddan	Frontend React.js
Hikmal Rajendra Zulfa	Desain antarmuka di Figma

BAB III

IMPLEMENTASI DAN PROGRESS PROYEK

3.1 Deskripsi Teknis Proyek

Proyek ini menggabungkan perangkat keras (hardware) dan perangkat lunak (software) untuk membangun sistem jemuran otomatis berbasis Internet of Things (IoT). Sistem dirancang agar dapat bekerja secara **otomatis** berdasarkan pembacaan sensor dan **manual** melalui kontrol dari web dashboard.

Komponen Utama:

- **ESP32 DevKit V4** digunakan sebagai mikrokontroler utama yang menangani pembacaan sensor dan pengendalian servo motor.
- **Sensor Hujan** dan **Sensor LDR** digunakan untuk mendeteksi kondisi cuaca (apakah sedang hujan atau gelap).
- **Servo Motor** berfungsi untuk membuka dan menutup jemuran.
- **Backend Express.js** digunakan untuk menerima data dari ESP32 dan menyediakan API untuk kontrol manual.
- **Frontend React.js** digunakan sebagai dashboard pengguna untuk menampilkan status dan memberikan perintah secara manual.
- **Web Config** pada ESP32 memungkinkan pengguna mengatur SSID WiFi dan URL server tanpa harus memprogram ulang perangkat.

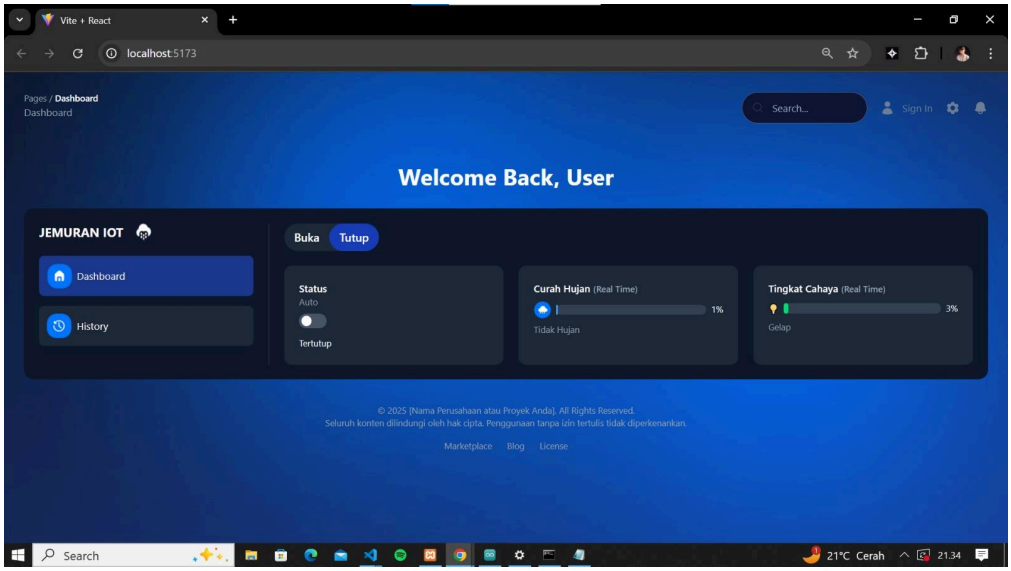
Alur Kerja Sistem:

1. ESP32 membaca nilai dari sensor LDR dan hujan setiap 5 detik.
2. Nilai sensor dikirim ke backend Express.js melalui HTTP POST dalam format JSON.
3. Backend menyimpan dan meneruskan data ke frontend React.js.
4. Di frontend, pengguna dapat melihat status jemuran dan beralih antara mode **otomatis** atau **manual**.
5. Jika mode manual diaktifkan, backend menerima perintah dari dashboard dan mengirimkan status kontrol ke ESP32 melalui endpoint /api/control.
6. ESP32 membaca status kontrol dan menggerakkan servo motor sesuai perintah.

3.2 Dokumentasi dan Pengerjaan

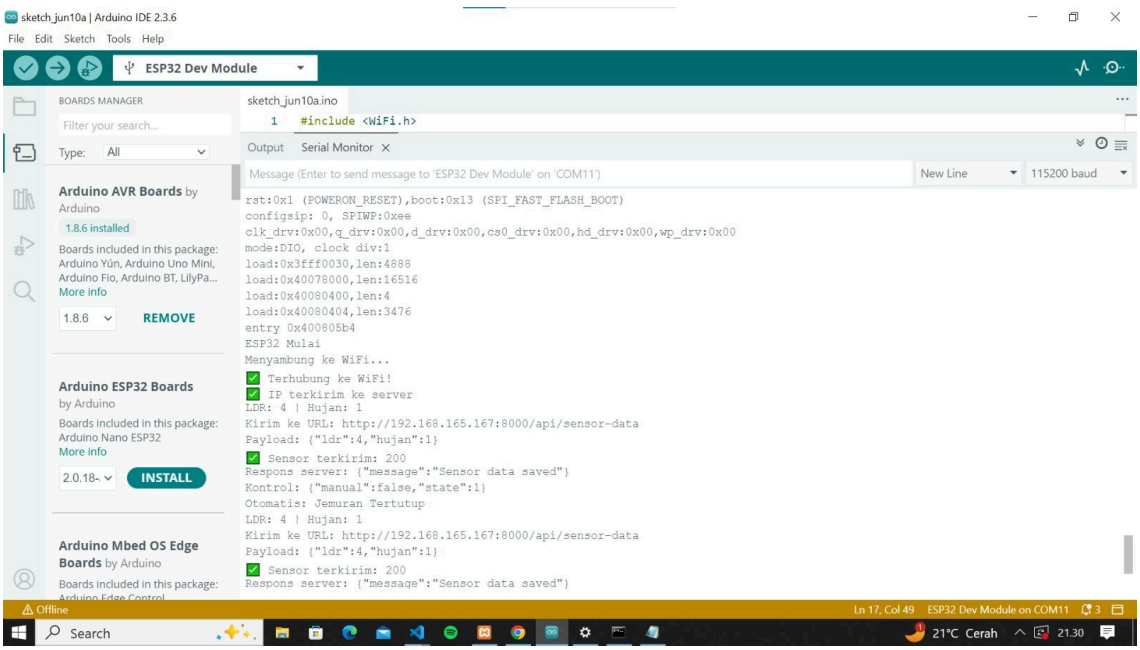
Screenshot Implementasi:

- **Dashboard Web (React)** menampilkan status sensor, tombol kontrol manual (Buka/Tutup), dan toggle switch mode.

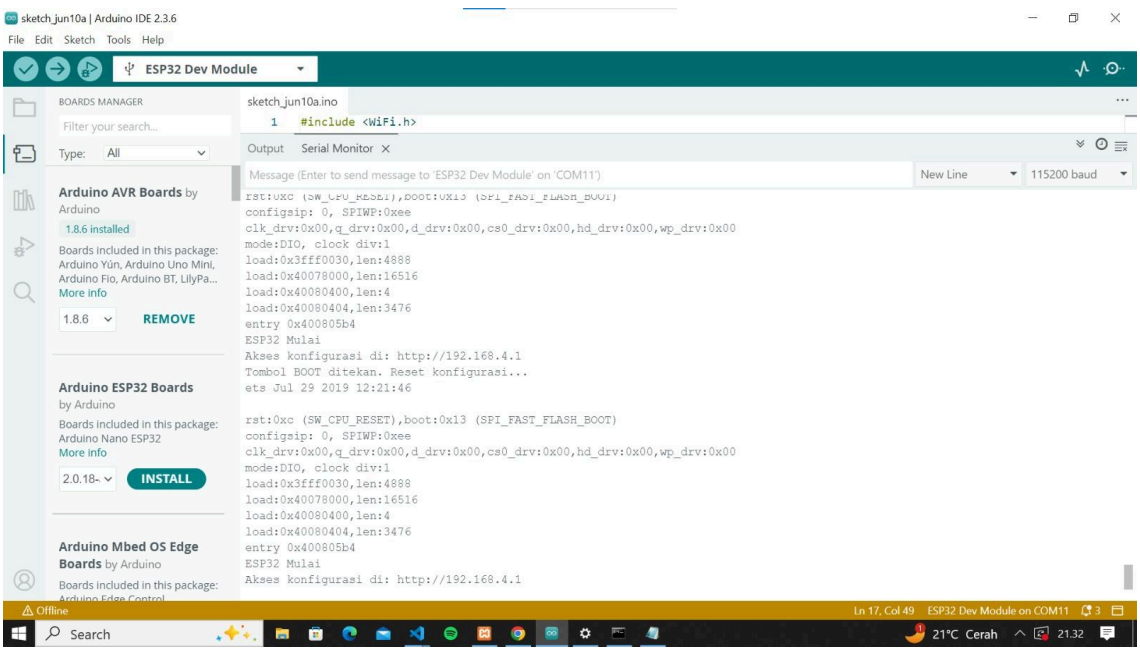


- **Serial Monitor ESP32** menunjukkan:

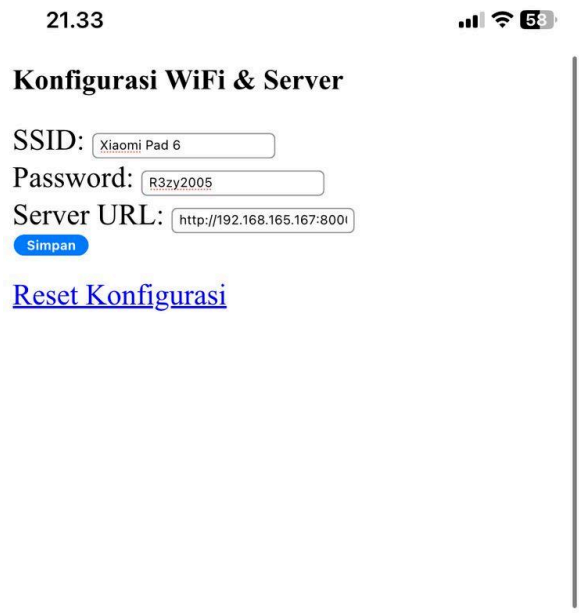
- Pengiriman data sensor ke server dan Respons server saat menerima kontrol manual



- Log koneksi WiFi dan server URL



- **Halaman Web Config** pada ESP32: formulir input SSID, password, dan URL server yang muncul saat ESP32 tidak bisa terhubung WiFi.



Kode Program ESP32:

```
#include <WiFi.h>
```

```
#include <WebServer.h>
#include <EEPROM.h>
#include <ESP32Servo.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>

#define EEPROM_SIZE 200
#define PIN_LDR 35
#define PIN_HUJAN 34
#define SERVO_PIN 25
#define RESET_PIN 0 // Tombol BOOT

WebServer server(80);
Servo jemuran;

String ssid = "", password = "", serverUrl = "";
bool manualMode = false;
int manualState = 0;
bool ipSent = false;
```

```
// Debounce tombol reset
bool lastResetButtonState = HIGH;
unsigned long lastDebounceTime = 0;
const unsigned long debounceDelay = 50;

void handleRoot() {
  String html = R"rawliteral(
    <form action="/save" method="post">
      <h2>Konfigurasi WiFi & Server</h2>
      SSID: <input name="ssid"><br>
      Password: <input name="pass"><br>
      Server URL: <input name="url"><br>
      <button type="submit">Simpan</button>
    </form>
    <p><a href="/reset">Reset Konfigurasi</a></p>
  )rawliteral";
  server.send(200, "text/html", html);
}

void handleSave() {
```

```
  ssid = server.arg("ssid");
  password = server.arg("pass");
  serverUrl = server.arg("url");

  if (!serverUrl.startsWith("http")) {
    server.send(400, "text/html", "<p>URL server tidak valid. Harus diawali dengan http:// atau https://</p>");
    return;
  }

  EEPROM.writeString(0, ssid);
  EEPROM.writeString(50, password);
  EEPROM.writeString(100, serverUrl);
  EEPROM.commit();

  server.send(200, "text/html", "<p>Berhasil disimpan! Rebooting...</p>");
  delay(2000);
  ESP.restart();
}
```

```
void handleReset() {
  for (int i = 0; i < EEPROM_SIZE; i++) EEPROM.write(i, 0);
  EEPROM.commit();
  server.send(200, "text/html", "<p>Konfigurasi di-reset! Rebooting...</p>");
  delay(2000);
  ESP.restart();
}

void handleNotFound() {
  server.send(404, "text/plain", "Halaman tidak ditemukan");
}

void setupWebConfig() {
  WiFi.softAP("ESP32_Config");
  IPAddress IP = WiFi.softAPIP();
  Serial.print("Akses konfigurasi di: http://");
  Serial.println(IP);

  server.on("/", handleRoot);
  server.on("/save", HTTP_POST, handleSave);
```

```
  server.on("/reset", handleReset);
  server.onNotFound(handleNotFound);
  server.begin();
}

void kirimDataSensor(int ldr, int hujan) {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    String url = serverUrl + "/api/sensor-data";

    Serial.println("Kirim ke URL: " + url);

    http.begin(url);
    http.addHeader("Content-Type", "application/json");

    String body = "{\"ldr\":" + String(ldr) + ",\"hujan\":" + String(hujan) + "}";
    Serial.println("Payload: " + body);

    int code = http.POST(body);
    String response = http.getString();
```

```
    if (code > 0) {
      Serial.println("✅ Sensor terkirim: " + String(code));
      Serial.println("Respons server: " + response);
    } else {
      Serial.println("❌ Gagal kirim sensor");
    }

    http.end();
  }
}


void ambilKontrol() {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(serverUrl + "/api/control");
    int code = http.GET();
    if (code == 200) {
      String res = http.getString();
```

```
      Serial.println("Kontrol: " + res);


      DynamicJsonDocument doc(256);
      DeserializationError error = deserializeJson(doc, res);
      if (!error) {
        manualMode = doc["manual"];
        manualState = doc["state"];
        if (manualMode) {
          jemuran.write(manualState == 1 ? 180 : 0);
        }
      } else {
        Serial.println("Gagal parsing JSON kontrol");
      }
    }
    http.end();
  }
}


void connectWiFi() {
  WiFi.begin(ssid.c_str(), password.c_str());
```

```
  Serial.print("Menyambung ke WiFi");
  int i = 0;
  while (WiFi.status() != WL_CONNECTED && i < 20) {
    delay(500);
    Serial.print(".");
    i++;
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\n✅ Terhubung ke WiFi!");
  } else {
    Serial.println("\n❌ Gagal terhubung ke WiFi. Reset konfigurasi dalam 5 detik...");
    delay(5000);
    for (int i = 0; i < EEPROM_SIZE; i++) EEPROM.write(i, 0);
    EEPROM.commit();
    ESP.restart();
  }
}

void kirimIPKeServer() {
```

```
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(serverUrl + "/api/update-ip");
    http.addHeader("Content-Type", "application/json");

    String ipStr = WiFi.localIP().toString();
    String body = "{\"ip\": \"" + ipStr + "\"}";

    int httpResponseCode = http.POST(body);
     Serial.println(httpResponseCode > 0 ? "✅ IP terkirim ke server" : "❌ Gagal kirim IP");

    http.end();
  }
}

void checkResetButton() {
  bool reading = digitalRead(RESET_PIN);
  if (reading != lastResetButtonState) {
    lastDebounceTime = millis();
```

```
    lastResetButtonState = reading;
  }

  if (reading == LOW && (millis() - lastDebounceTime) > debounceDelay) {
    Serial.println("Tombol BOOT ditekan. Reset konfigurasi...");
    for (int i = 0; i < EEPROM_SIZE; i++) EEPROM.write(i, 0);
    EEPROM.commit();
    delay(1000);
    ESP.restart();
  }
}

void setup() {
  Serial.begin(115200);
  delay(1000);
  Serial.println("ESP32 Mulai");

  EEPROM.begin(EEPROM_SIZE);
  pinMode(RESET_PIN, INPUT_PULLUP);
```

```
  ssid = EEPROM.readString(0);
  password = EEPROM.readString(50);
  serverUrl = EEPROM.readString(100);

  if (ssid.length() < 1 || password.length() < 1 || serverUrl.length() < 5) {
    setupWebConfig();
  } else {
    connectWiFi();
  }

  jemuran.attach(SERVO_PIN);
}

void loop() {
  checkResetButton();

  if (WiFi.getMode() == WIFI_AP) {
    server.handleClient();
    return;
  }
```

```
if (ssid.length() < 1 || password.length() < 1 || serverUrl.length() < 5) {
  setupWebConfig();
  return;
}

if (WiFi.status() != WL_CONNECTED) {
  connectWiFi();
  ipSent = false;
  return;
}

if (!ipSent) {
  kirimIPKeServer();
  ipSent = true;
}

int ldr = map(analogRead(PIN_LDR), 0, 4096, 100, 0);
int hujan = map(analogRead(PIN_HUJAN), 0, 4096, 100, 0);
```

```
  Serial.print("LDR: "); Serial.print(ldr);
  Serial.print(" | Hujan: "); Serial.println(hujan);

  kirimDataSensor(ldr, hujan);
  ambilKontrol();

  if (!manualMode) {
    if (ldr < 30 || hujan > 40) {
      jemuran.write(180);
      Serial.println("Otomatis: Jemuran Tertutup");
    } else {
      jemuran.write(0);
      Serial.println("Otomatis: Jemuran Terbuka");
    }
  }

  delay(5000);
}
```

- Menggunakan pustaka WiFi.h, ESP32Servo.h, EEPROM.h, HTTPClient.h, dan ArduinoJson.h.
- Menangani:
    - Mode Access Point (untuk konfigurasi awal)
    - Penyimpanan konfigurasi WiFi dan URL server ke EEPROM
    - Pengiriman data sensor ke backend Express.js
    - Pengambilan status kontrol dari API
    - Pengendalian mode otomatis/manual
    - Reset konfigurasi via tombol BOOT

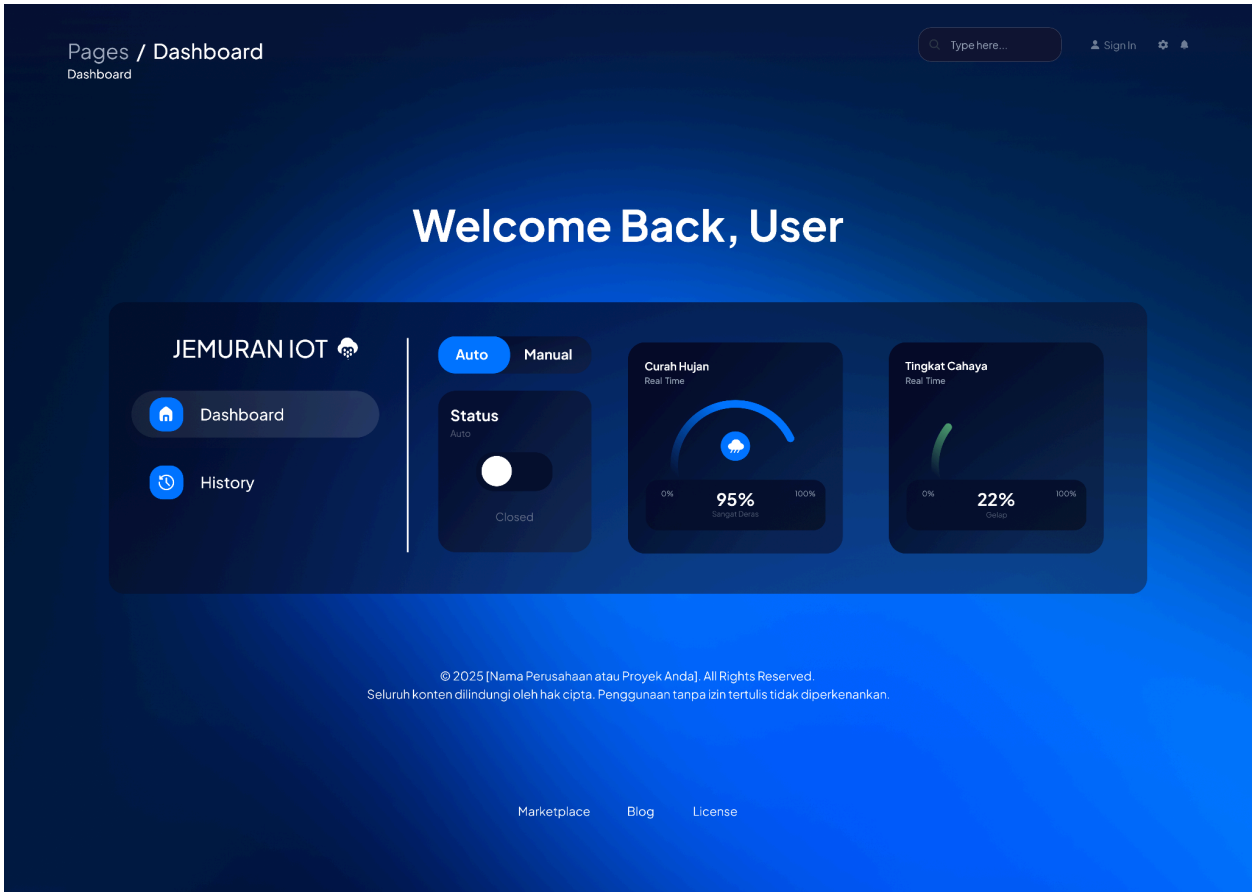
Repositori GitHub:

- Backend Express.js: github.com/Alfahreziii/express-jemuran
- Frontend React.js: github.com/Alfahreziii/react-jemuran

3.3 Kendala dan Solusi

Kendala	**Solusi**
ESP32 gagal terhubung ke WiFi jika konfigurasi salah	Ditambahkan mode Access Point untuk konfigurasi ulang SSID dan server
Delay pengiriman data sensor	Diatur delay optimal (5 detik) dan retry jika gagal koneksi
Kesalahan parsing JSON dari API kontrol manual	Diperbaiki dengan validasi tipe data dan penggunaan ArduinoJson
Sinkronisasi antara kontrol manual dan mode otomatis	Ditambahkan kondisi if (manualMode) agar kontrol servo tidak bentrok
UI awal kurang intuitif	Dibantu tim desain untuk membuat mockup di Figma yang lebih user-friendly

3.4 Tampilan Mockup (Figma)



Marketplace Blog License

BAB IV

EVALUASI DAN HASIL

4.1 Perbandingan Hasil Dengan Tujuan Awal

Secara umum, seluruh tujuan proyek yang telah dirumuskan di awal berhasil dicapai. Sistem jemuran otomatis mampu membaca kondisi lingkungan menggunakan sensor LDR (cahaya) dan sensor hujan, kemudian mengambil keputusan untuk membuka atau menutup jemuran secara otomatis berdasarkan data tersebut. Selain itu, sistem juga berhasil mengirimkan data sensor ke backend Express.js dan menampilkannya secara real-time melalui dashboard React.js.

Fitur mode manual yang menjadi pengembangan tambahan juga berhasil diimplementasikan dengan baik. Pengguna dapat berpindah dari mode otomatis ke mode manual secara langsung melalui toggle switch pada dashboard. Ketika mode manual diaktifkan, pengguna dapat mengklik tombol "Buka" atau "Tutup" untuk menggerakkan servo motor sesuai keinginan, tanpa mengandalkan data dari sensor.

Integrasi antara ESP32, backend, dan frontend berjalan secara sinkron. Hal ini menunjukkan bahwa protokol komunikasi (HTTP GET dan POST), pengolahan data JSON, dan parsing response berhasil dirancang dan diimplementasikan dengan benar. Sistem juga menunjukkan kemampuan untuk melakukan fallback ke mode konfigurasi apabila terjadi kegagalan dalam menyambungkan WiFi, yang merupakan nilai tambah dari sisi keandalan sistem.

4.2 Uji Coba Sistem

Pengujian dilakukan secara bertahap dan mencakup skenario berikut:

1. **Pengujian Sensor Hujan dan LDR**
    - Ketika sensor hujan mendeteksi nilai di atas ambang batas (misalnya di atas 40%), maka sistem otomatis menggerakkan servo untuk menutup jemuran.
    - Ketika nilai LDR menunjukkan kondisi gelap (misalnya <30%), sistem juga akan menutup jemuran.
    - Ketika kondisi terang dan tidak hujan, sistem membuka jemuran kembali.
2. **Pengujian Mode Manual**
    - User dapat menonaktifkan mode otomatis melalui toggle switch di dashboard.

- Setelah dalam mode manual, tombol "Buka" dan "Tutup" dapat digunakan untuk langsung mengontrol posisi servo motor.
- Perubahan status manual/otomatis dan posisi servo tampil secara real-time di antarmuka web.

3. **Pengujian Koneksi dan Komunikasi**
   - ESP32 berhasil terhubung ke jaringan WiFi dan mengirim data ke server lokal menggunakan IP statis.
   - Data sensor berhasil dikirim setiap interval 5 detik dan tersimpan di backend.
   - IP ESP32 juga berhasil dikirim ke endpoint khusus untuk pemantauan.
4. **Pengujian Reset Konfigurasi**
   - Ketika tombol BOOT ditekan, sistem akan menghapus konfigurasi WiFi dan server URL dari EEPROM.
   - Setelah itu, ESP32 akan masuk ke mode Access Point (AP) dan membuka halaman konfigurasi di 192.168.4.1.

4.3 Feedback dan Saran Perbaikan

Berdasarkan hasil evaluasi dan uji coba, berikut beberapa catatan dan saran yang dapat dijadikan bahan pengembangan lebih lanjut:

- **Penyimpanan histori**: Sistem saat ini belum menyimpan data histori. Penambahan database seperti Firebase, MongoDB, atau SQLite akan sangat membantu untuk pelaporan.
- **Notifikasi Otomatis**: Sistem dapat dikembangkan dengan fitur notifikasi real-time ke WhatsApp/Telegram saat hujan terdeteksi.
- **Daya Mandiri**: Penambahan sumber daya dari panel surya bisa menjadi solusi agar sistem tetap berjalan saat listrik padam.

BAB V

KESIMPULAN DAN REKOMENDASI

5.1 Kesimpulan Akhir Proyek

Proyek Sistem Jemuran Otomatis Berbasis ESP32 dengan Monitoring Real-Time via Web ini berhasil diimplementasikan dengan baik dan sesuai dengan tujuan yang telah dirancang di awal. Sistem mampu membaca kondisi lingkungan menggunakan sensor hujan dan sensor LDR untuk menentukan status jemuran secara otomatis. Ketika kondisi hujan atau cahaya rendah terdeteksi, sistem akan menutup jemuran secara otomatis menggunakan servo motor. Sebaliknya, saat kondisi terang dan tidak hujan, sistem akan membuka jemuran kembali.

Fitur tambahan berupa mode manual juga berhasil ditambahkan. Melalui dashboard berbasis React.js, pengguna dapat mengontrol posisi jemuran secara langsung melalui tombol interaktif. Pengguna juga dapat beralih antara mode otomatis dan manual dengan mudah, memberikan fleksibilitas penuh terhadap penggunaan sistem ini dalam berbagai situasi.

Dari sisi teknis, proyek ini memanfaatkan kombinasi berbagai teknologi: mikrokontroler ESP32 untuk pengolahan data sensor dan kontrol aktuator, Express.js sebagai backend API untuk komunikasi data, serta React.js untuk antarmuka pengguna yang responsif dan mudah diakses melalui browser. Koneksi antara semua komponen berjalan lancar, dan semua data berhasil dikirimkan dan diproses dengan baik.

Secara keseluruhan, proyek ini membuktikan bahwa teknologi Internet of Things (IoT) dapat diterapkan secara langsung dalam kehidupan sehari-hari untuk menyelesaikan masalah sederhana namun penting, seperti perlindungan pakaian dari hujan. Proyek ini juga menjadi media pembelajaran yang baik untuk memahami proses perencanaan, pelaksanaan, dan evaluasi dalam konteks manajemen proyek teknologi.

5.2 Rekomendasi

Berdasarkan hasil evaluasi dan observasi selama pengembangan proyek, berikut beberapa rekomendasi untuk pengembangan sistem ke depannya:

1. **Integrasi dengan Database Cloud**

   Untuk menyimpan data historis sensor dan status jemuran, sistem dapat dikembangkan lebih lanjut dengan integrasi ke database seperti **Firebase Realtime Database**,

MongoDB Atlas, atau **MySQL Cloud**. Hal ini akan memungkinkan pengguna melihat riwayat aktivitas jemuran dan kondisi cuaca.

2. **Notifikasi Otomatis ke Pengguna**
   Sistem dapat ditambahkan fitur notifikasi otomatis ke WhatsApp, Telegram, atau email saat kondisi hujan terdeteksi atau ketika jemuran berubah status. Ini meningkatkan pengalaman pengguna terutama saat mereka tidak berada di rumah.
3. **Peningkatan Keamanan Sistem**
   Backend server perlu dilengkapi dengan fitur keamanan seperti **autentikasi token JWT**, **CORS**, dan proteksi terhadap serangan umum (misalnya brute-force, request flooding). Hal ini penting terutama jika sistem dipublikasikan secara online.
4. **Optimasi Konsumsi Daya**
   ESP32 dapat diatur ke mode **deep sleep** untuk menghemat daya ketika tidak digunakan. Penambahan sumber energi alternatif seperti **panel surya** juga dapat menjadikan sistem lebih ramah lingkungan dan mandiri.
5. **Menggunakan MQTT/WebSocket untuk Komunikasi**
   Untuk efisiensi dan komunikasi real-time dua arah, protokol **MQTT** atau **WebSocket** dapat digunakan menggantikan metode HTTP polling yang cenderung boros bandwidth.
6. **Desain Fisik dan Pelindung Komponen**
   Dalam penggunaan outdoor, semua komponen elektronik perlu dilindungi dari air dan panas matahari. Kotak pelindung tahan air (IP65+) dan pengaturan kabel yang lebih rapi akan meningkatkan durabilitas alat.
7. **Pengembangan Aplikasi Mobile**
   Selain dashboard web, akan lebih baik jika sistem dikembangkan dalam bentuk **mobile app** berbasis Android/iOS, agar kontrol jemuran bisa dilakukan lebih cepat dan mudah.

Dengan mengimplementasikan rekomendasi-rekomendasi ini, sistem jemuran otomatis berbasis IoT yang telah dibangun dapat menjadi lebih canggih, lebih andal, dan lebih siap untuk digunakan dalam kehidupan nyata oleh masyarakat luas.

BAB VI

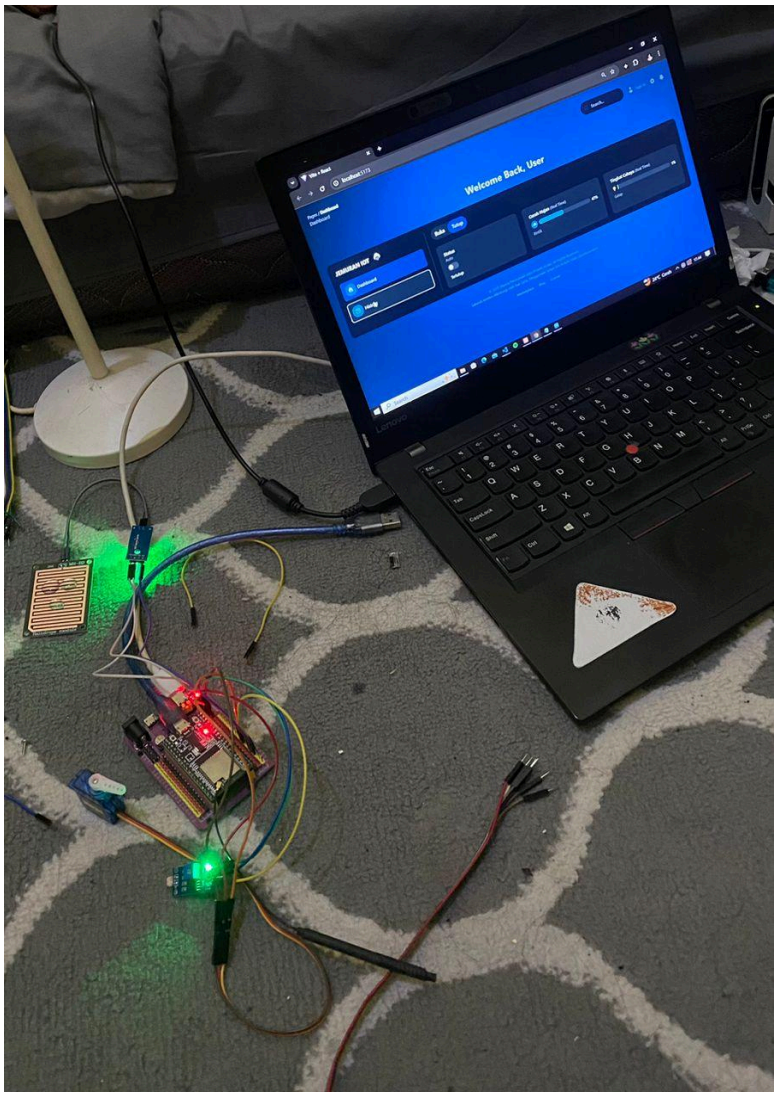
DAFTAR PUSTAKA

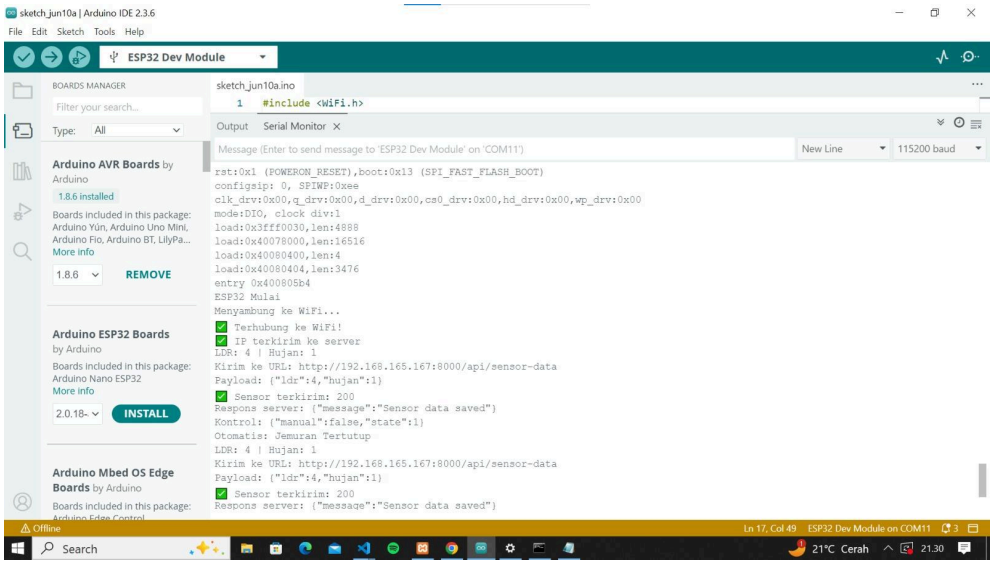
1. Datasheet ESP32 – Espressif
2. Dokumentasi Express.js – https://expressjs.com
3. Dokumentasi React.js – https://reactjs.org
4. Dokumentasi ArduinoJson – https://arduinojson.org
5. Dokumentasi Servo ESP32 – https://github.com/jkb-git/ESP32Servo

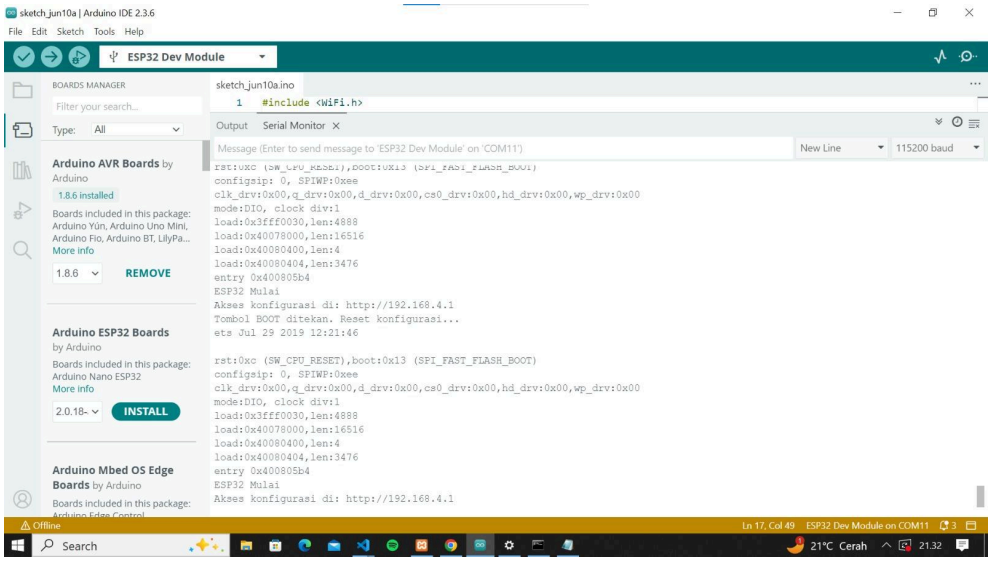
BAB VII

LAMPIRAN

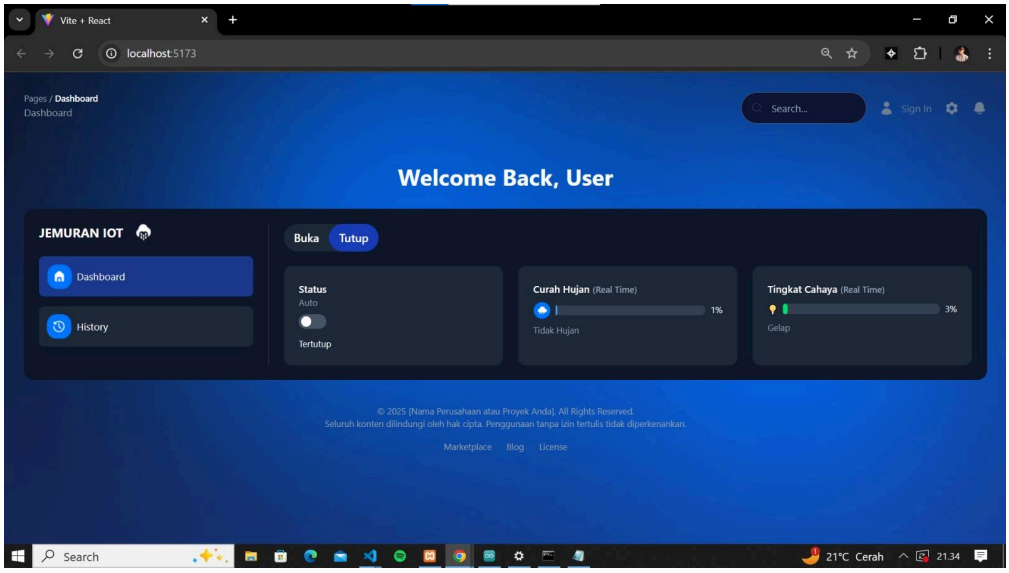
1. Kode Program ESP32
2. Link Prototype Desain UI/UX: https://www.figma.com/proto/lqmlB6iZW3bLBVksgYNYaE?node-id=529-44&t=vUoqA9TuEPwZ3bVG-6
3. Link GitHub:
    a. Backend: https://github.com/Alfahreziii/express-jemuran
    b. Frontend: https://github.com/Alfahreziii/react-jemuran
4. Screenshot Dashboard, Serial Monitor, Dokumentasi Pembuatan dan WiFi Config Page




sketch_jun10a | Arduino IDE 2.3.6
File Edit Sketch Tools Help
ESP32 Dev Module
BOARDS MANAGER
Filter your search...
Type: All
Arduino AVR Boards by Arduino
1.8.6 installed
Boards included in this package: Arduino Yún, Arduino Uno Mini, Arduino Fio, Arduino BT, LilyPa...
More info
1.8.6 REMOVE
Arduino ESP32 Boards
by Arduino
Boards included in this package: Arduino Nano ESP32
More info
2.0.18- INSTALL
Arduino Mbed OS Edge Boards by Arduino
Boards included in this package:
sketch_jun10a.ino
1 #include <WiFi.h>
Output Serial Monitor
Message (Enter to send message to 'ESP32 Dev Module' on 'COM11')
New Line
115200 baud
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:4888
load:0x40078000,len:16516
load:0x40080400,len:4
load:0x40080404,len:3476
entry 0x400805b4
ESP32 Mulai
Menyambung ke WiFi...
Terhubung ke WiFi!
IP terkirim ke server
LDR: 4 | Hujan: 1
Kirim ke URL: http://192.168.165.167:8000/api/sensor-data
Payload: {"ldr":4,"hujan":1}
Sensor terkirim: 200
Respons server: {"message":"Sensor data saved"}
Kontrol: {"manual":false,"state":1}
Otomatis: Jemuran Tertutup
LDR: 4 | Hujan: 1
Kirim ke URL: http://192.168.165.167:8000/api/sensor-data
Payload: {"ldr":4,"hujan":1}
Sensor terkirim: 200
Respons server: {"message":"Sensor data saved"}
Offline
Ln 17, Col 49 ESP32 Dev Module on COM11
Search
21°C Cerah
21.30


sketch_jun10a | Arduino IDE 2.3.6
File Edit Sketch Tools Help
ESP32 Dev Module
BOARDS MANAGER
Filter your search...
Type: All
Arduino AVR Boards by Arduino
1.8.6 installed
Boards included in this package: Arduino Yún, Arduino Uno Mini, Arduino Fio, Arduino BT, LilyPa...
More info
1.8.6 REMOVE
Arduino ESP32 Boards
by Arduino
Boards included in this package: Arduino Nano ESP32
More info
2.0.18- INSTALL
Arduino Mbed OS Edge Boards by Arduino
Boards included in this package:
sketch_jun10a.ino
1 #include <WiFi.h>
Output Serial Monitor
Message (Enter to send message to 'ESP32 Dev Module' on 'COM11')
New Line
115200 baud
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:4888
load:0x40078000,len:16516
load:0x40080400,len:4
load:0x40080404,len:3476
entry 0x400805b4
ESP32 Mulai
Akses konfigurasi di: http://192.168.4.1
Tombol BOOT ditekan. Reset konfigurasi...
ets Jul 29 2019 12:21:46
rst:0xc (SW_CPU_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:4888
load:0x40078000,len:16516
load:0x40080400,len:4
load:0x40080404,len:3476
entry 0x400805b4
ESP32 Mulai
Akses konfigurasi di: http://192.168.4.1
Offline
Ln 17, Col 49 ESP32 Dev Module on COM11
Search
21°C Cerah
21.32


21.33
58
Konfigurasi WiFi & Server
SSID: Xiaomi Pad 6
Password: R3zy2005
Server URL: http://192.168.165.167:800
Simpan
Reset Konfigurasi


Vite + React
localhost:5173
Pages / Dashboard
Dashboard
Search...
Sign In
Welcome Back, User
JEMURAN IOT
Dashboard
History
Buka
Tutup
Status
Auto
Tertutup
Curah Hujan (Real Time)
1%
Tidak Hujan
Tingkat Cahaya (Real Time)
3%
Gelap
© 2025 [Nama Perusahaan atau Proyek Anda]. All Rights Reserved.
Seluruh konten dilindungi oleh hak cipta. Penggunaan tanpa izin tertulis tidak diperkenankan.
Marketplace
Blog
License
Search
21°C Cerah
21:34