Laporan Sistem Perancangan dan Implementasi Sistem Monitoring dan Pengendalian Lampu Otomatis Berbasis IoT Menggunakan ESP32 dan Sensor BH1750

Disusun oleh :

Syandy Arda Syahnuari	(20230140148)
Muhammad Irfan Fauzi	(20230140136)
Muhammad Ridho Rizqullah	(20230140131)
Erindhito Nur Fauzan	(20230140115)
Muhammad Aqil Firdaus	(20230140145)
Muhammad Zaki Mahogra	(20230140144)

PROGRAM STUDI TEKNOLOGI INFORMASI FAKULTAS TEKNIK

UNIVERSITAS MUHAMMADIYAH YOGYAKARTA

2025/2026

Daftar Isi

BAB 1
Pendahuluan

1.1 Latar Belakang

Penerangan merupakan kebutuhan penting dalam kehidupan sehari-hari, baik di lingkungan rumah tangga, perkantoran, maupun fasilitas umum. Pada umumnya, pengendalian lampu masih dilakukan secara manual atau menggunakan sensor analog seperti LDR (Light Dependent Resistor). Namun, penggunaan LDR memiliki keterbatasan, antara lain tingkat akurasi yang rendah, sensitivitas terhadap noise, serta ketidakstabilan dalam mendeteksi perubahan intensitas cahaya.

Perkembangan teknologi Internet of Things (IoT) memungkinkan sistem penerangan dikembangkan menjadi lebih cerdas, otomatis, dan terintegrasi dengan jaringan internet. Salah satu solusi yang dapat digunakan adalah sensor cahaya digital BH1750, yang mampu mengukur intensitas cahaya secara akurat dalam satuan lux melalui komunikasi I2C. Sensor ini lebih stabil dan presisi dibandingkan sensor analog.

Untuk mendukung sistem tersebut, digunakan ESP32 sebagai mikrokontroler karena memiliki performa tinggi, konsumsi daya rendah, serta konektivitas Wi-Fi bawaan. Dengan integrasi sistem IoT ke dalam aplikasi berbasis web menggunakan Node.js sebagai backend dan React sebagai frontend, pengguna tidak hanya dapat mengontrol lampu secara otomatis (twilight switch), tetapi juga memantau data intensitas cahaya secara real-time, mengontrol lampu secara manual, serta melihat riwayat data melalui dashboard interaktif.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, rumusan masalah dalam pengembangan sistem ini adalah sebagai berikut:

1. Bagaimana merancang sistem lampu otomatis (twilight switch) berbasis ESP32 dan sensor BH1750 dengan tingkat akurasi yang baik?
2. Bagaimana mengintegrasikan perangkat IoT dengan sistem backend berbasis Node.js untuk menyimpan data sensor ke dalam database?
3. Bagaimana membangun dashboard web berbasis React untuk monitoring data secara real-time dan pengendalian lampu secara manual?

1.3 Batasan Masalah

Agar pembahasan lebih terfokus, maka batasan masalah dalam proyek ini adalah:

1. Mikrokontroler yang digunakan adalah ESP32 Dev Module.

2. Sensor cahaya yang digunakan adalah BH1750FVI.
3. Beban yang dikendalikan berupa lampu AC 220V melalui modul relay 1-channel.
4. Sistem monitoring menggunakan Node.js (Express) sebagai backend dan React.js sebagai frontend.
5. Komunikasi data antara ESP32 dan server menggunakan HTTP dengan metode REST API.

1.4 Tujuan

Tujuan dari pengembangan sistem ini adalah:

1. Mengimplementasikan sensor BH1750 sebagai sensor cahaya digital untuk meningkatkan akurasi sistem lampu otomatis.
2. Mengembangkan sistem monitoring berbasis web yang mampu menampilkan data intensitas cahaya dan status relay secara real-time.
3. Menyediakan fitur kontrol jarak jauh yang memungkinkan pengguna mengatur mode otomatis dan manual melalui dashboard web.

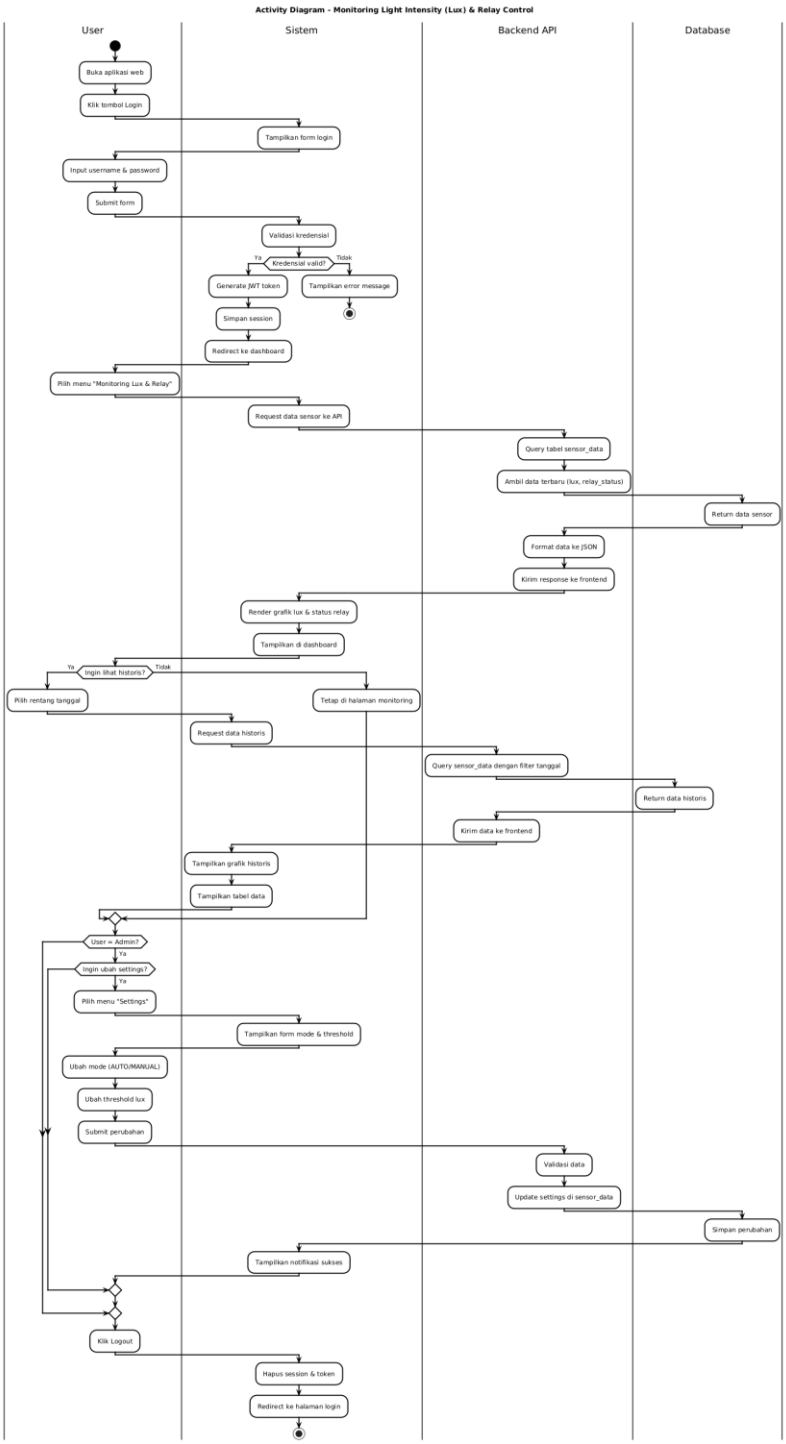
1.5 Manfaat

Manfaat yang diperoleh dari pengembangan sistem ini antara lain:

1. Efisiensi energi, karena lampu hanya menyala sesuai kondisi intensitas cahaya lingkungan.
2. Kemudahan monitoring, pengguna dapat memantau kondisi penerangan dari jarak jauh melalui web.
3. Automasi yang andal, mengurangi kesalahan manusia dalam pengoperasian lampu serta meningkatkan umur pakai perangkat.

BAB 2
Unified Modeling Language (UML)

2. 1 Activity Diagram

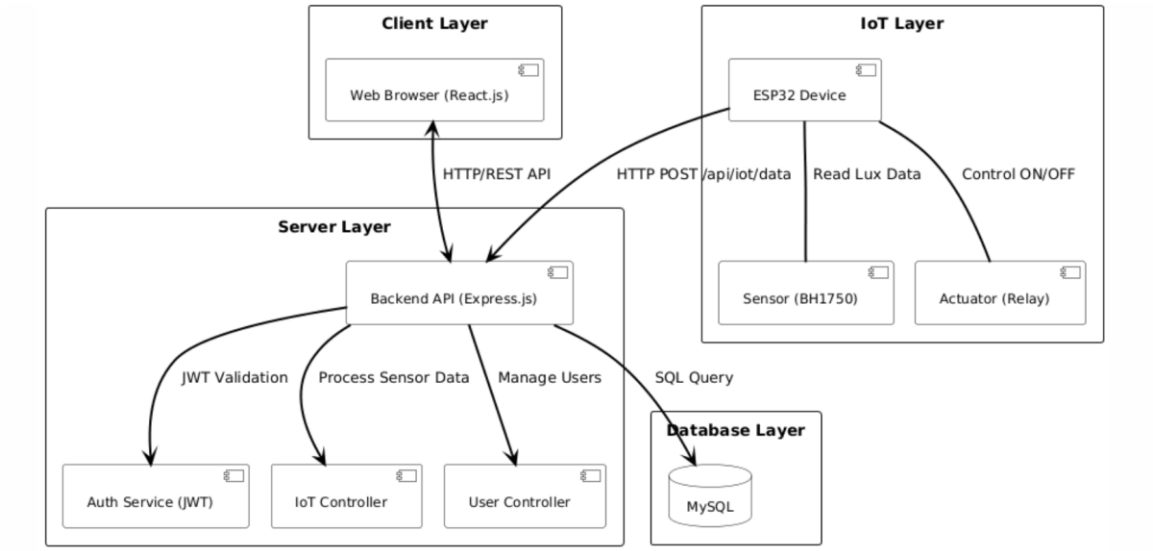


Activity Diagram (activity.puml)

Diagram ini menggambarkan alur kerja (workflow) pengguna dari awal hingga akhir saat menggunakan sistem. Diagram ini dibagi menjadi beberapa jalur (*swimlanes*) yaitu: User, Sistem (Frontend), Backend API, dan Database.

- Alur Login: Proses dimulai dengan user membuka web dan login. Sistem memvalidasi kredensial; jika valid, sistem membuat token JWT dan mengarahkan user ke dashboard.
- Monitoring: Di dashboard, sistem meminta data sensor terbaru (lux dan status relay) dari Backend API yang mengambilnya dari Database, lalu menampilkannya dalam bentuk grafik.
- Data Historis: User dapat memilih untuk melihat data historis berdasarkan rentang tanggal tertentu.
- Pengaturan (Admin): Jika user adalah Admin, terdapat fitur khusus untuk mengubah pengaturan mode (AUTO/MANUAL) dan nilai ambang batas (*threshold*) cahaya. Perubahan ini disimpan ke database.
- Logout: Proses diakhiri dengan menghapus sesi dan mengembalikan user ke halaman login.

2.2 Arsitektu Sistem



Architecture Diagram (arsitektur.puml)

Diagram ini menjelaskan teknologi dan struktur komponen yang digunakan dalam sistem, yang dibagi menjadi 4 lapisan utama:

- Client Layer: Sisi pengguna yang menggunakan Web Browser dengan teknologi React.js.
- Server Layer: Pusat logika aplikasi yang menggunakan Express.js (Backend API), dilengkapi dengan layanan Autentikasi (JWT), serta pengontrol untuk IoT dan User.
- Database Layer: Menggunakan MySQL untuk penyimpanan data.
- IoT Layer: Terdiri dari mikrokontroler ESP32 yang terhubung dengan sensor cahaya (BH1750) dan aktuator (Relay).

Alur Komunikasi:

- ESP32 mengirim data ke API melalui HTTP POST.
- Client (React) mengambil data dari API melalui HTTP/REST.
- API melakukan query SQL ke Database.

2.3 Class Diagram

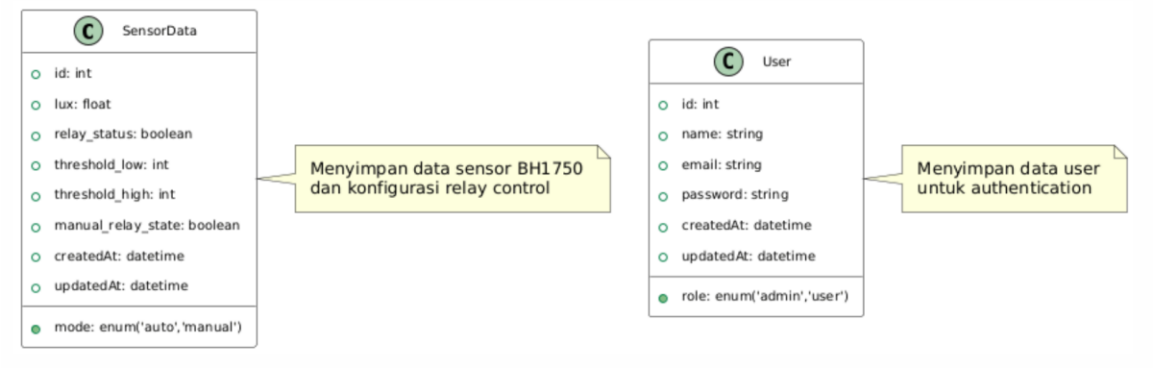
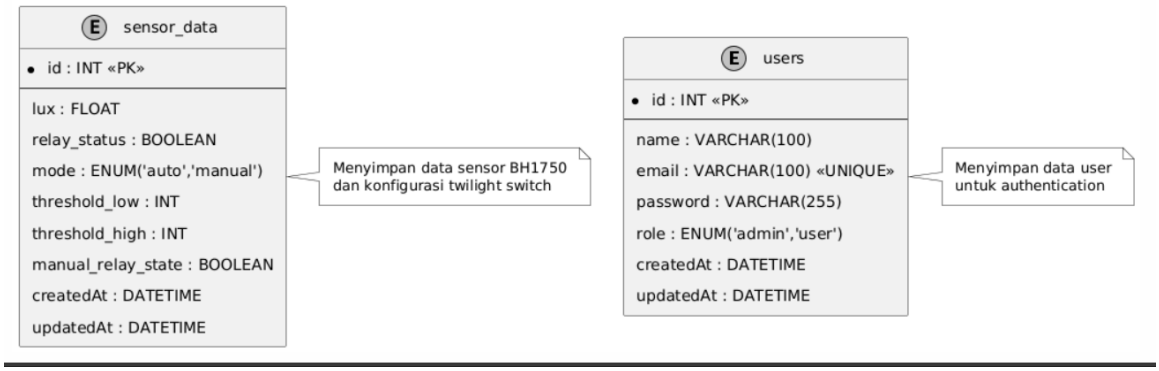


Diagram ini menggambarkan struktur database atau model data yang digunakan dalam sistem.

- Class SensorData:
    - Menyimpan data pembacaan sensor (lux).
    - Menyimpan status alat (relay_status, manual_relay_state).
    - Menyimpan konfigurasi otomatisasi (mode, threshold_low, threshold_high).
    - Dilengkapi *timestamp* (createdAt, updatedAt).
- Class User:
    - Menyimpan data identitas pengguna (name, email, password).
    - Menyimpan hak akses atau peran (role: 'admin' atau 'user') yang menentukan apakah user bisa mengubah pengaturan atau tidak.

2.4 Diagram ERD



Entity Relationship Diagram (erd.puml)

Diagram ini mendefinisikan skema database yang digunakan untuk menyimpan informasi sistem. Terdapat dua entitas (tabel) utama:

- Tabel users:
    - Berfungsi untuk manajemen otentikasi dan otorisasi.
    - Menyimpan data login seperti email (unik), password, dan name.
    - Memiliki kolom role dengan tipe ENUM ('admin', 'user') yang membedakan hak akses pengguna biasa dan administrator.
- Tabel sensor_data:
    - Tabel ini memiliki fungsi ganda: menyimpan data telemetri (hasil bacaan sensor) dan menyimpan konfigurasi sistem.
    - Data Telemetri: Kolom lux (intensitas cahaya) dan relay_status (status nyala/mati lampu).
    - Data Konfigurasi: Kolom mode (auto/manual), serta ambang batas threshold_low dan threshold_high untuk pemicu otomatisasi relay.

2.5 Flowchart Sistem

Flowchart (flowchart.puml)

Diagram ini menggambarkan algoritma atau alur logika aplikasi dari sisi pengguna saat menavigasi menu.

- Logika Login:

  Saat aplikasi dibuka, sistem mengecek sesi. Jika belum login, user diminta input kredensial. Jika valid, sesi dibuat; jika tidak, muncul pesan error.

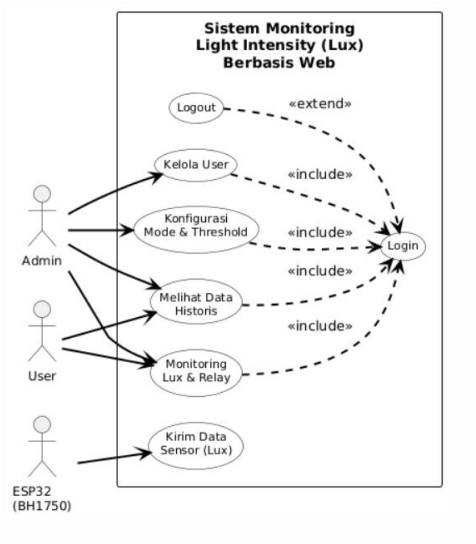
- Percabangan Menu (Decision Logic):

  1. Monitoring Real-time: Mengambil data terbaru dan memperbarui grafik.
  2. Lihat Data Historis: User memilih rentang waktu -> Query Database -> Tampilkan tabel/grafik.
  3. Settings (Pengaturan): Sistem melakukan pengecekan hak akses if (User = Admin?).
     - Jika Ya: Admin bisa mengubah mode (AUTO/MANUAL) dan nilai *threshold* lux, lalu database diperbarui.
     - Jika Tidak: Sistem menolak akses.

4. Kelola User: Sama seperti menu Settings, ini dilindungi oleh pengecekan Admin. Admin bisa menambah, mengedit, atau menghapus user.

5. Logout: Menghapus token sesi dan mengembalikan user ke halaman login.

2.6 Use Case Diagram



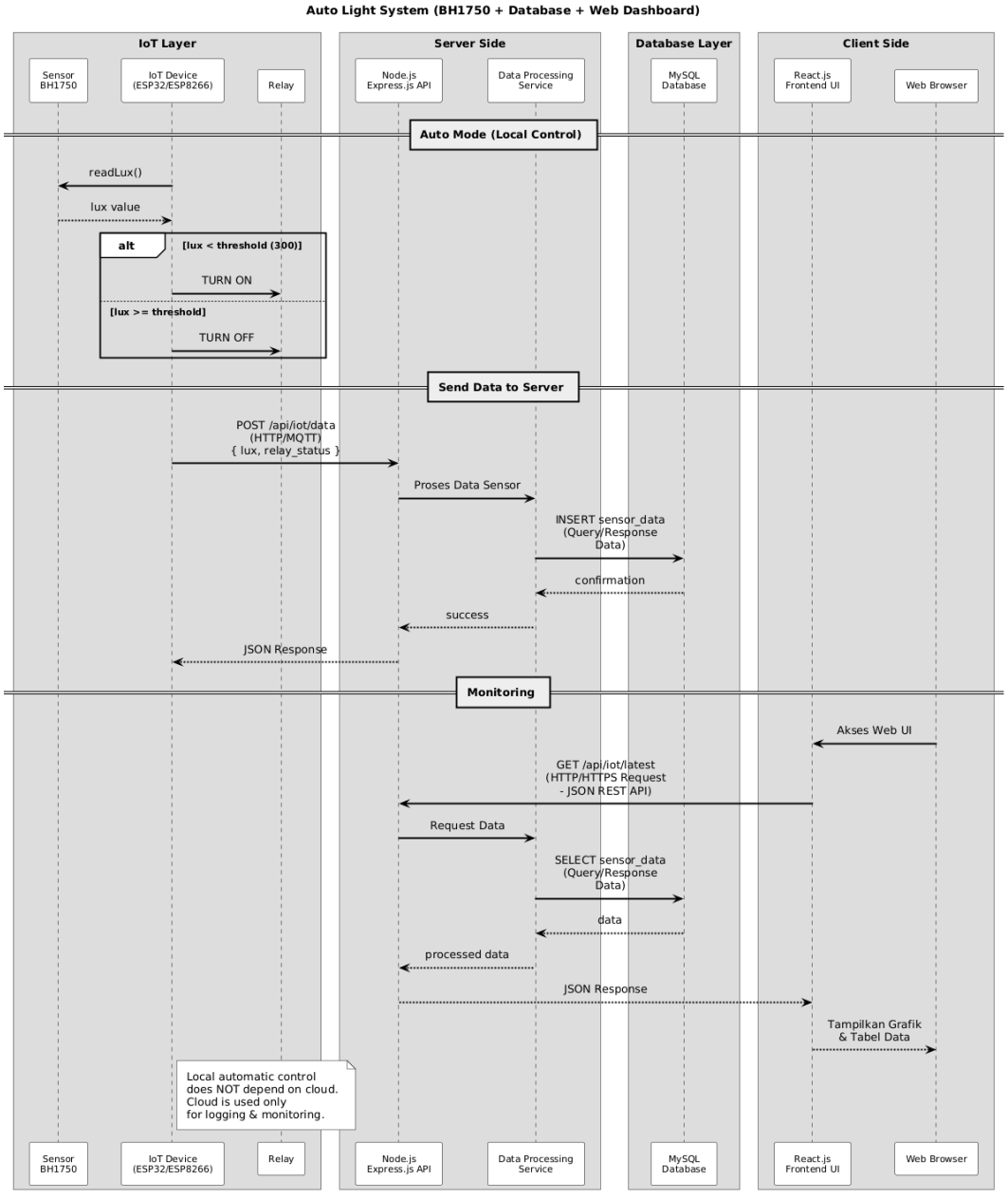
Use Case Diagram (usecase.puml)

Diagram ini memetakan siapa (aktor) yang berinteraksi dengan sistem dan apa (fungsionalitas) yang bisa mereka lakukan.

- Aktor:
    1. Admin: Memiliki akses penuh ke seluruh sistem.
    2. User: Memiliki akses terbatas (hanya monitoring).
    3. ESP32 (Device): Aktor mesin yang bekerja secara otonom mengirim data.
- Fungsionalitas (Use Cases):
    - Monitoring & Historis: Baik Admin maupun User dapat melihat data *real-time* dan data masa lalu.
    - Konfigurasi & Manajemen: Hanya Admin yang memiliki hak untuk:
        - "Kelola User" (Menambah/Mengedit pengguna).
        - "Konfigurasi Mode & Threshold" (Mengatur kapan relay harus nyala otomatis).

- Otomasi: ESP32 hanya memiliki satu tugas spesifik, yaitu "Kirim Data Sensor" ke sistem.
- Keamanan: Semua aktivitas pengguna (kecuali aksi ESP32) mewajibkan proses "Login" terlebih dahulu.

2.7 Sequence Diagram 1



sequence_1.puml: Auto Mode & Monitoring Flow

Diagram ini menggambarkan skenario utama sistem saat berjalan dalam Mode Otomatis dan bagaimana data dipantau oleh pengguna.

A. Kontrol Lokal (Edge Computing) Bagian ini menunjukkan bahwa keputusan menyalakan/mematikan lampu terjadi di dalam ESP32 itu sendiri, bukan di server.

- Sensor Reading: ESP32 membaca nilai cahaya dari sensor BH1750.
- Logika If-Else:
    - Jika lux < 300: Relay dinyalakan (ON).
    - Jika lux >= 300: Relay dimatikan (OFF).
- Penting: Ada catatan *"Local automatic control does NOT depend on cloud"*. Artinya, jika internet mati, fitur otomatisasi lampu tetap berfungsi normal karena logikanya ada di alat.

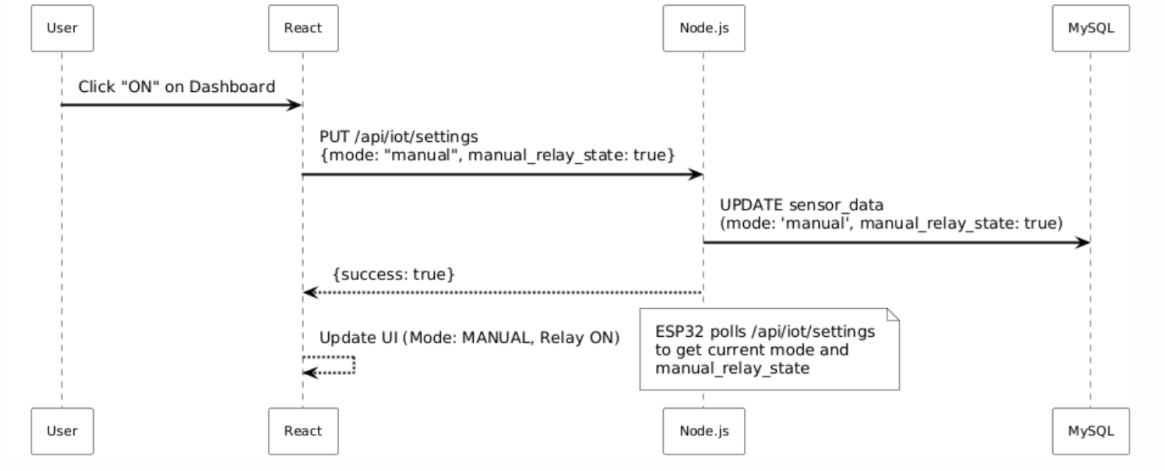
B. Pengiriman Data (Logging) Setelah melakukan aksi lokal, ESP32 melapor ke server.

- ESP32 mengirim data (lux & relay_status) via HTTP POST ke endpoint /api/iot/data.
- API meneruskan data ke *Data Processing Service* (DPS) untuk disimpan ke MySQL (INSERT sensor_data).

C. Monitoring (User View)

- User membuka Web Dashboard (React.js).
- Web meminta data terbaru via HTTP GET /api/iot/latest.
- Server mengambil data dari MySQL dan mengembalikannya ke Web untuk ditampilkan dalam bentuk grafik/tabel.

2.8 Sequence Diagram 2



sequence_2.puml: Manual Control Flow

Diagram ini menjelaskan skenario saat User ingin mengambil alih kendali lampu secara manual melalui Web Dashboard.

Alur Perubahan Mode:

1. Aksi User: User mengklik tombol "ON" di dashboard.
2. Request API: React mengirim HTTP PUT ke /api/iot/settings dengan *payload*: {mode: "manual", manual_relay_state: true}.
3. Update Database: Node.js mengupdate tabel sensor_data di MySQL, mengubah mode menjadi 'manual' dan status relay menjadi 'true'.
4. Feedback UI: Web menerima konfirmasi sukses dan memperbarui tampilan UI menjadi mode MANUAL dengan posisi relay ON.

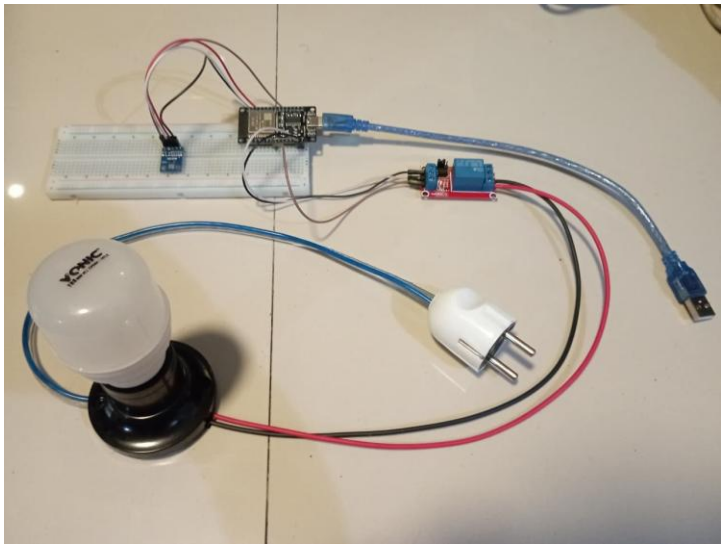
Mekanisme Sinkronisasi ke Alat (Polling): Diagram ini memiliki catatan krusial di bagian bawah:

"ESP32 polls /api/iot/settings to get current mode and manual_relay_state".

Ini berarti server tidak secara langsung "mendorong" perintah ke ESP32 saat user klik tombol. Sebaliknya, ESP32 secara berkala "bertanya" (polling) ke server: *"Apakah ada perubahan setting?"*. Jika server menjawab mode="manual" dan state="true", barulah ESP32 menyalakan relay.

BAB 3
Aplikasi

3.1 Foto Alat



3.1.1 Lampu LED

Lampu LED pada sistem ini digunakan sebagai beban/output yang menunjukkan hasil dari proses pengendalian lampu otomatis. Lampu LED akan menyala atau mati berdasarkan perintah yang diberikan oleh ESP32 melalui modul relay, baik secara otomatis berdasarkan intensitas cahaya dari sensor BH1750 maupun secara manual melalui dashboard web. Penggunaan lampu LED dipilih karena konsumsi dayanya rendah, respons cepat, dan efisien, sehingga sesuai untuk penerapan sistem monitoring dan pengendalian lampu berbasis IoT.

3.1.2 Modul Relay 5V (SRD-05VDC-SL-C)

Modul Relay 5V (SRD-05VDC-SL-C) berfungsi sebagai aktuator yang digunakan untuk mengendalikan lampu melalui ESP32. Relay bekerja sebagai saklar elektronik yang memungkinkan sinyal logika bertegangan rendah dari ESP32 mengontrol beban listrik secara aman. Pada sistem ini, relay diaktifkan atau dinonaktifkan berdasarkan logika otomatis dari sensor BH1750 maupun perintah manual yang dikirim melalui dashboard web.

3.1.3 Sensor Cahaya BH1750

Sensor BH1750 merupakan sensor cahaya digital yang digunakan untuk mengukur intensitas cahaya lingkungan dalam satuan lux. Sensor ini berkomunikasi dengan ESP32 melalui protokol I2C dan menghasilkan data yang akurat serta stabil. Nilai lux yang dibaca digunakan sebagai dasar pengambilan keputusan untuk menyalakan atau mematikan lampu secara otomatis serta ditampilkan pada sistem monitoring berbasis web.

3.1.4 ESP32

ESP32 adalah mikrokontroler yang berfungsi sebagai pusat kendali sistem. ESP32 membaca data intensitas cahaya dari sensor BH1750, memproses logika otomatis (AUTO/MANUAL), serta mengendalikan modul relay untuk menyalakan atau mematikan lampu. Selain itu, ESP32 memiliki Wi-Fi bawaan yang digunakan untuk mengirim data sensor dan status relay ke server melalui REST API serta menerima pengaturan dari dashboard web.

3.1.5 Jumper Cable

a. Jumper Male–Male

Kabel jumper male–male digunakan untuk menghubungkan antar pin pada modul atau board yang sama-sama memiliki header female, misalnya koneksi antara ESP32 dan breadboard.

b. Jumper Male–Female

Kabel jumper male–female digunakan untuk menghubungkan pin ESP32 (female header) ke modul eksternal seperti sensor BH1750 atau modul relay yang memiliki pin male.

Kabel jumper berfungsi sebagai media penghantar sinyal dan daya antar komponen elektronik dalam rangkaian.

3.1.6 Board ESP (Breadboard)

Breadboard digunakan sebagai media perakitan rangkaian sementara tanpa perlu penyolderan. Board ini memudahkan proses pemasangan, pengujian, dan modifikasi rangkaian ESP32, sensor BH1750, relay, dan komponen pendukung lainnya. Dengan breadboard, rangkaian dapat disusun lebih rapi dan fleksibel.

3.1.7 Kabel Positif dan Netral Lampu 5V

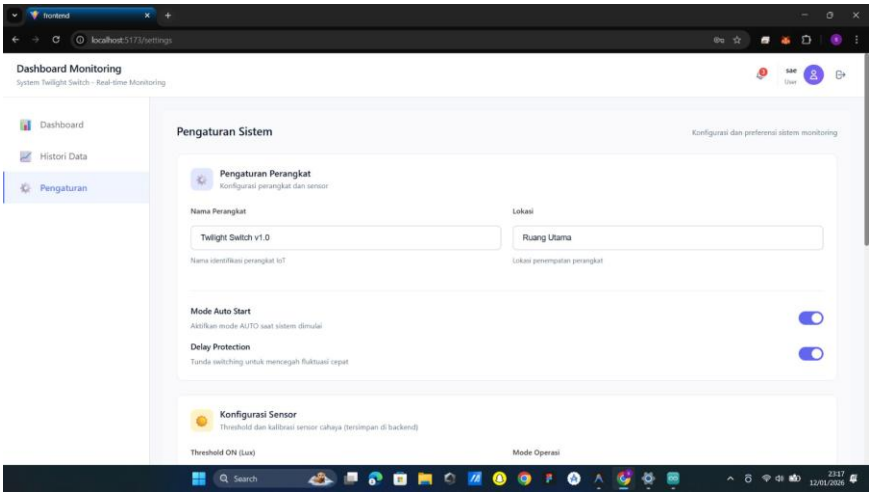
Kabel positif (+) dan netral (−) digunakan untuk menyalurkan sumber tegangan ke lampu 5V. Salah satu jalur kabel dilewatkan melalui modul relay sehingga:

- Saat relay aktif, arus mengalir dan lampu menyala.
- Saat relay nonaktif, arus terputus dan lampu mati.

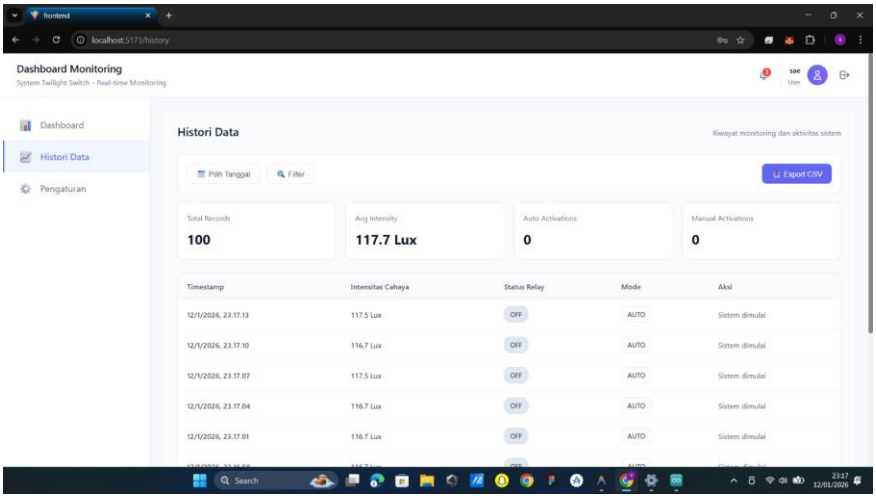
Kabel ini berfungsi sebagai jalur daya utama bagi lampu dan memungkinkan pengendalian lampu secara otomatis maupun manual melalui sistem IoT.

3.2 Tampilan Web per Homepage

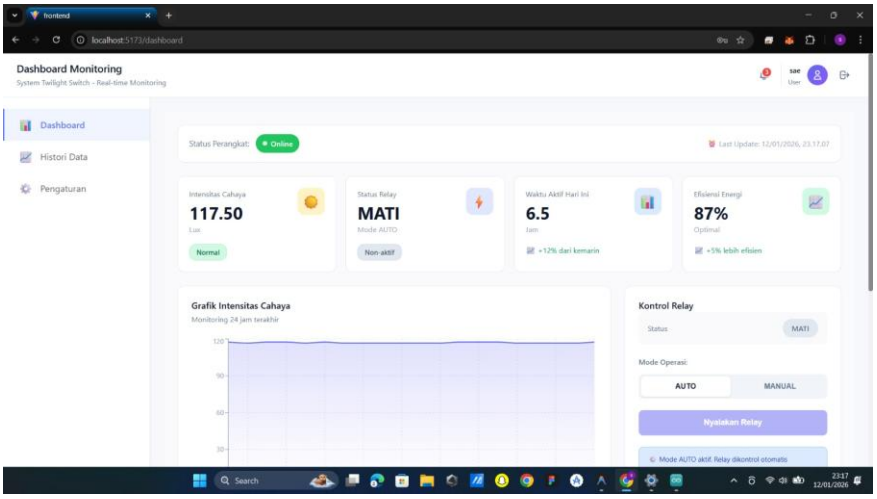
3.2.1 Tampilan Pengaturan User



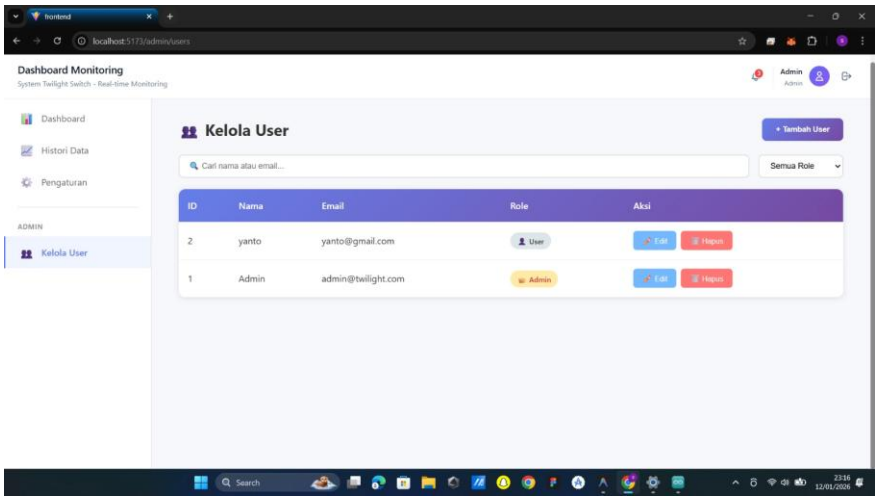
3.2.2 Tampilan History Data User



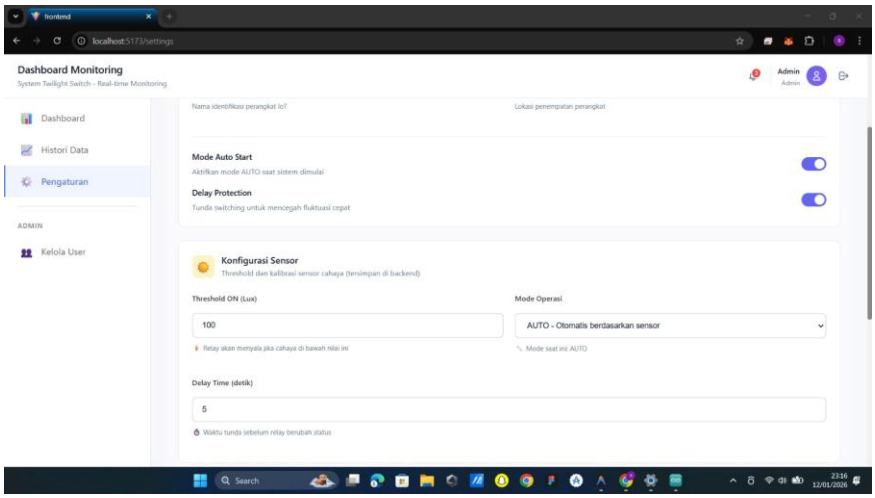
3.2.3 Tampilan Dashboard User



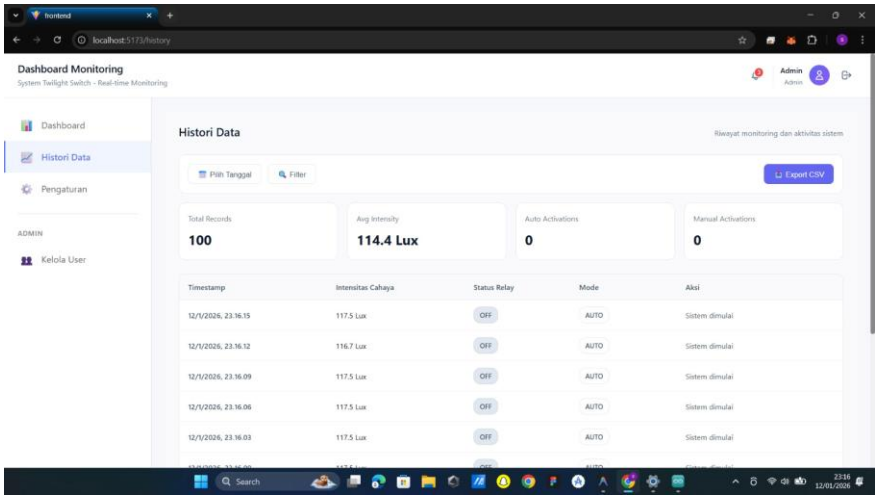
3.2.4 Tampilan Kelola User (Admin)



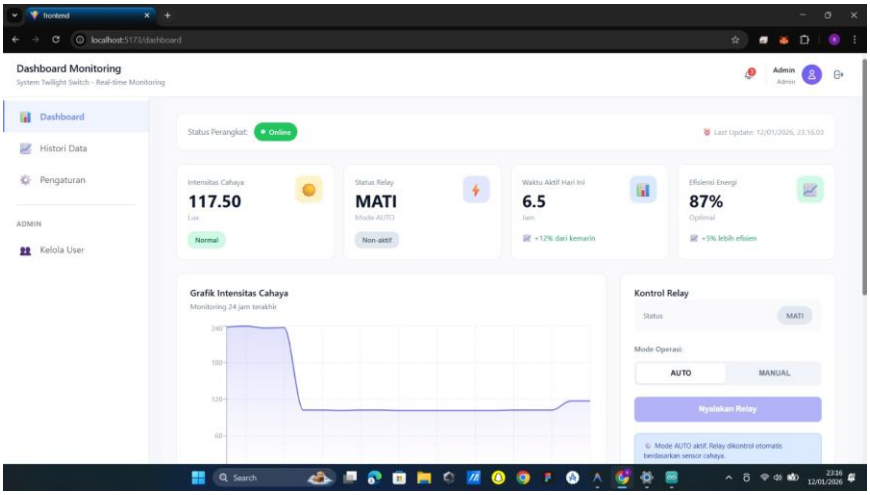
3.2.5 Tampilan Pengaturan Admin



3.2.6 Tampilan History Data Admin



3.2.7 Tampilan DashBoard Admin

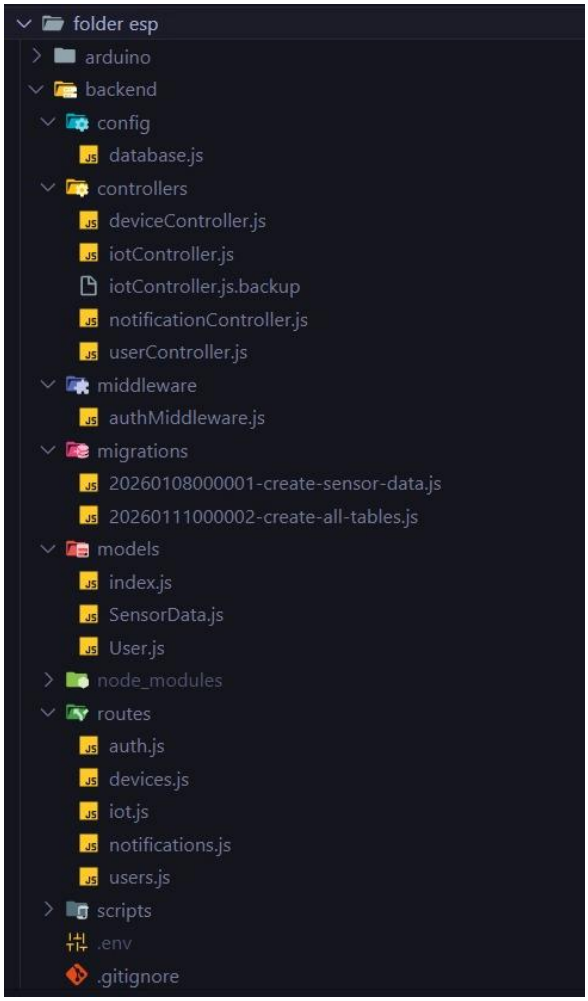


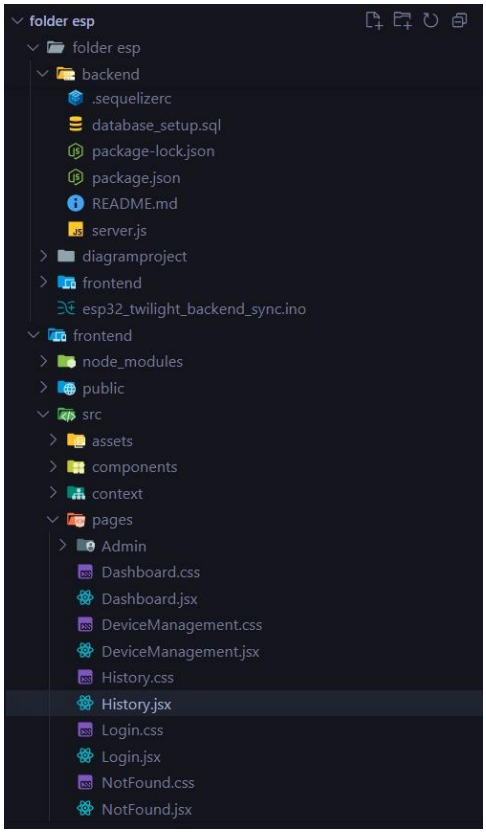
3.3 Penjelasan Susunan Folder dan Source Code

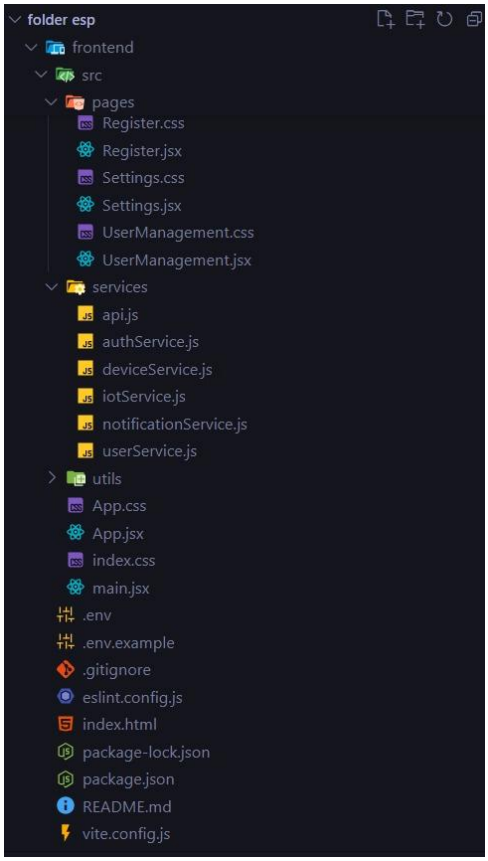
3.3.1 Susunan Folder

Penjelasan Struktur Folder

Proyek IoT Twilight Switch ini terdiri dari tiga komponen utama: frontend (React + Vite) untuk antarmuka pengguna berbasis web yang menampilkan dashboard monitoring lux sensor dan kontrol relay, backend (Node.js + Express) yang mengelola API endpoints untuk menerima data sensor dari ESP32, menyimpan ke database MySQL, serta mengatur mode operasi (auto/manual) dan threshold lux, dan arduino yang berisi kode ESP32 untuk membaca sensor BH1750 dan mengirim data ke backend secara berkala. Folder diagramproject berisi dokumentasi diagram sistem seperti arsitektur, use case, dan ERD untuk keperluan dokumentasi proyek.


folder esp
arduino
backend
config
database.js
controllers
deviceController.js
iotController.js
iotController.js.backup
notificationController.js
userController.js
middleware
authMiddleware.js
migrations
20260108000001-create-sensor-data.js
20260111000002-create-all-tables.js
models
index.js
SensorData.js
User.js
node_modules
routes
auth.js
devices.js
iot.js
notifications.js
users.js
scripts
.env
.gitignore


folder esp
folder esp
backend
.sequelizerc
database_setup.sql
package-lock.json
package.json
README.md
server.js
diagramproject
frontend
esp32_twilight_backend_sync.ino
frontend
node_modules
public
src
assets
components
context
pages
Admin
Dashboard.css
Dashboard.jsx
DeviceManagement.css
DeviceManagement.jsx
History.css
History.jsx
Login.css
Login.jsx
NotFound.css
NotFound.jsx


folder esp
frontend
src
pages
Register.css
Register.jsx
Settings.css
Settings.jsx
UserManagement.css
UserManagement.jsx
services
api.js
authService.js
deviceService.js
iotService.js
notificationService.js
userService.js
utils
App.css
App.jsx
index.css
main.jsx
.env
.env.example
.gitignore
eslint.config.js
index.html
package-lock.json
package.json
README.md
vite.config.js

3.3.2 Source Code

Source Code lengkap bisa diakses melalui link github berikut :

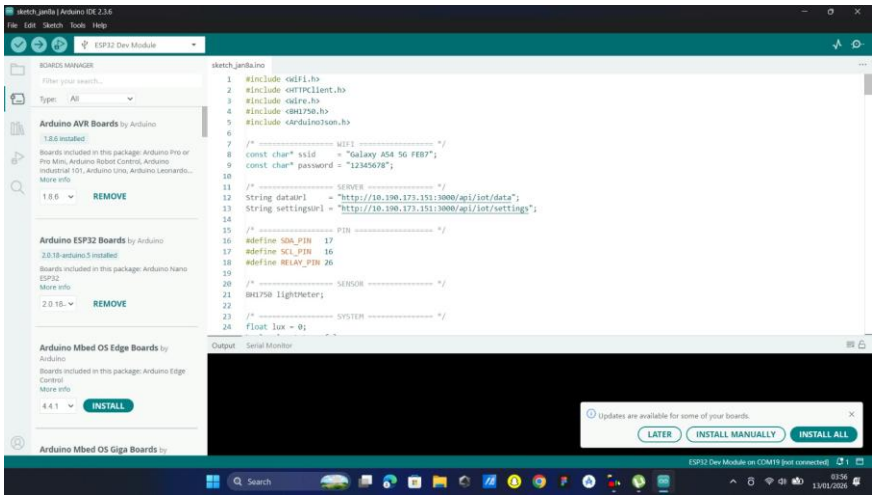
https://github.com/C-PPAW-TI503P-2025/Project-Kelompok6-Cahaya.git

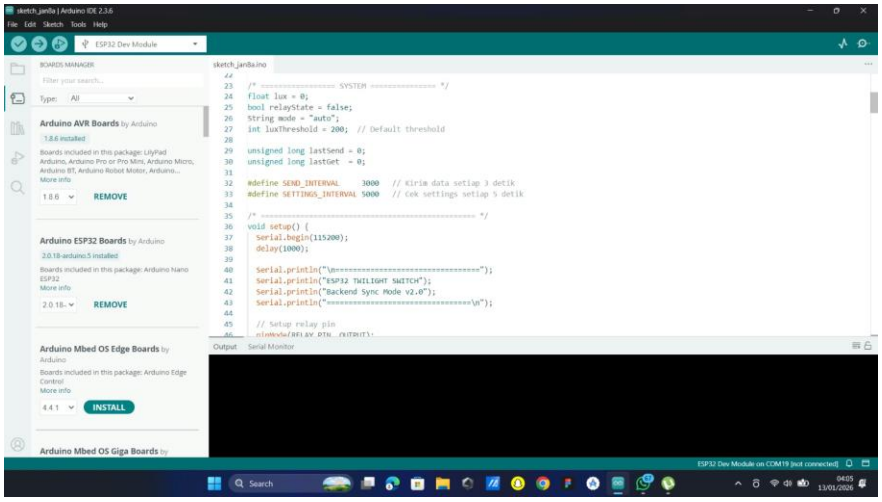
Penjelasan Arduino (ESP32)

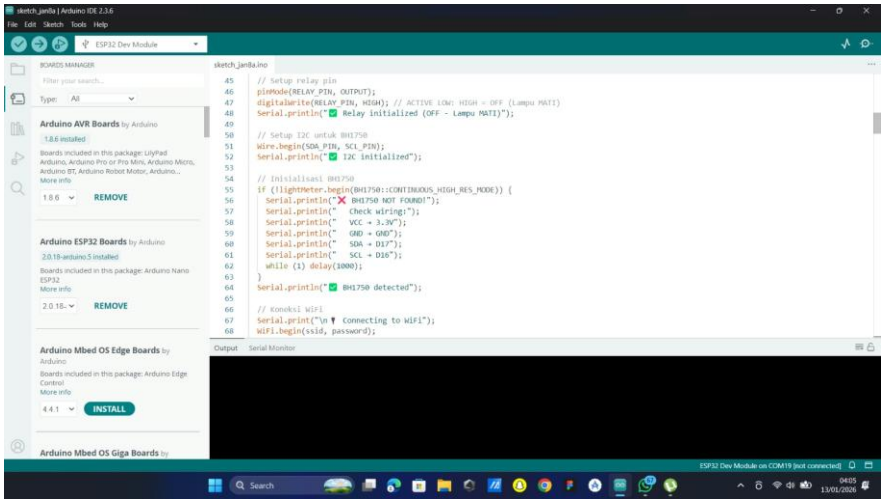
Kode Arduino

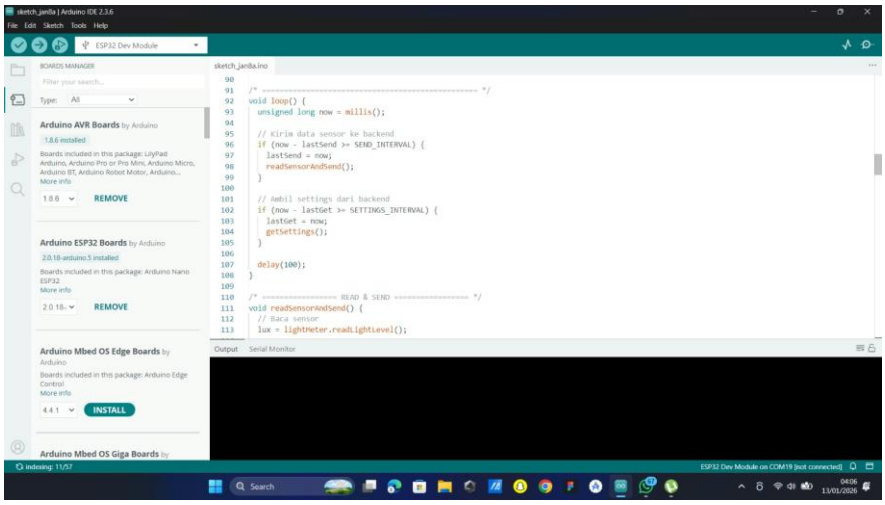
esp32_twilight_backend_sync.ino berjalan pada mikrokontroler ESP32 yang terhubung dengan sensor cahaya BH1750 (via I2C pada pin D17/SDA dan D16/SCL) dan modul relay (pin D26 dengan logika ACTIVE LOW). ESP32 secara otomatis membaca intensitas cahaya (lux) setiap 3 detik dan mengirimkannya ke backend melalui HTTP POST, kemudian menerima respons berupa perintah relay (ON/OFF) berdasarkan logika yang diproses di backend. Setiap 5 detik, ESP32 juga melakukan HTTP GET untuk mengambil pengaturan terbaru (mode auto/manual, threshold lux) dari backend, sehingga kontrol relay dapat dilakukan secara sinkron antara perangkat fisik dan aplikasi web tanpa perlu hard-coding logika di sisi ESP32.

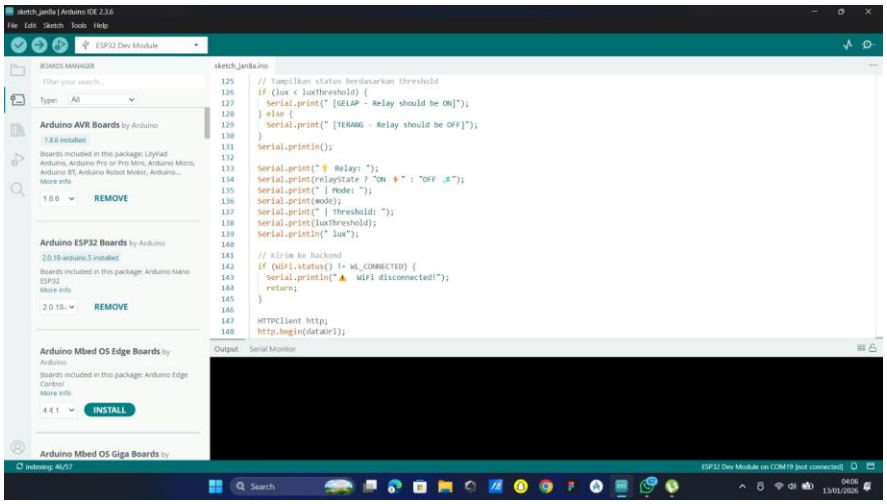
Tangkapan layer Arduino

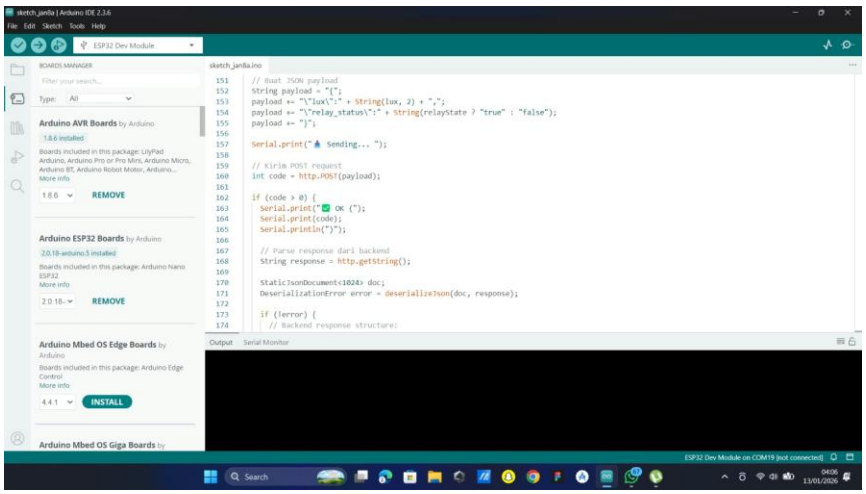


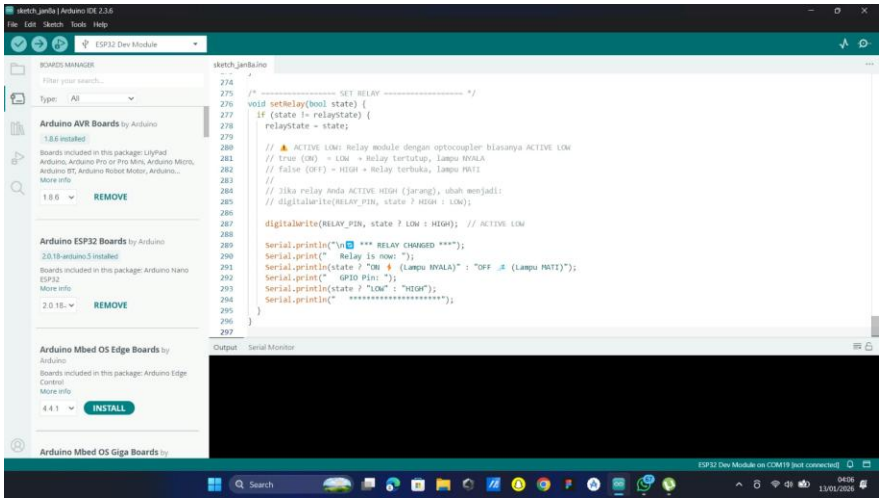

sketch_jan8a | Arduino IDE 2.3.6
File Edit Sketch Tools Help
ESP32 Dev Module
BOARDS MANAGER
Arduino AVR Boards by Arduino
1.8.6 installed
REMOVE
Arduino ESP32 Boards by Arduino
2.0.18-arduino.5 installed
REMOVE
Arduino Mbed OS Edge Boards by Arduino
INSTALL
Arduino Mbed OS Giga Boards by
sketch_jan8a.ino
/* ================ SYSTEM ================ */
float lux = 0;
bool relayState = false;
String mode = "auto";
int luxThreshold = 200; // Default threshold
unsigned long lastSend = 0;
unsigned long lastGet = 0;
#define SEND_INTERVAL 3000 // Kirim data setiap 3 detik
#define SETTINGS_INTERVAL 5000 // Cek settings setiap 5 detik
/* == */
void setup() {
Serial.begin(115200);
delay(1000);
Serial.println("\n==============================");
Serial.println("ESP32 TWILIGHT SWITCH");
Serial.println("Backend Sync Mode v2.0");
Serial.println("==============================\n");
// Setup relay pin
Output Serial Monitor
ESP32 Dev Module on COM19 [not connected]
Search
0405
13/01/2026


sketch_jan8a | Arduino IDE 2.3.6
File Edit Sketch Tools Help
ESP32 Dev Module
BOARDS MANAGER
Arduino AVR Boards by Arduino
1.8.6 installed
REMOVE
Arduino ESP32 Boards by Arduino
2.0.18-arduino.5 installed
REMOVE
Arduino Mbed OS Edge Boards by Arduino
INSTALL
Arduino Mbed OS Giga Boards by
sketch_jan8a.ino
// Setup relay pin
pinMode(RELAY_PIN, OUTPUT);
digitalWrite(RELAY_PIN, HIGH); // ACTIVE LOW: HIGH = OFF (Lampu MATI)
Serial.println("✅ Relay initialized (OFF - Lampu MATI)");
// Setup I2C untuk BH1750
Wire.begin(SDA_PIN, SCL_PIN);
Serial.println("✅ I2C initialized");
// Inisialisasi BH1750
if (!lightMeter.begin(BH1750::CONTINUOUS_HIGH_RES_MODE)) {
Serial.println("❌ BH1750 NOT FOUND!");
Serial.println(" Check wiring:");
Serial.println(" VCC → 3.3V");
Serial.println(" GND → GND");
Serial.println(" SDA → D17");
Serial.println(" SCL → D16");
while (1) delay(1000);
}
Serial.println("✅ BH1750 detected");
// Konoksi WiFi
Serial.print("\n📡 Connecting to WiFi");
WiFi.begin(ssid, password);
Output Serial Monitor
ESP32 Dev Module on COM19 [not connected]
Search
0405
13/01/2026


sketch_jan8a | Arduino IDE 2.3.6
File Edit Sketch Tools Help
ESP32 Dev Module
BOARDS MANAGER
Arduino AVR Boards by Arduino
1.8.6 installed
REMOVE
Arduino ESP32 Boards by Arduino
2.0.18-arduino.5 installed
REMOVE
Arduino Mbed OS Edge Boards by Arduino
INSTALL
Arduino Mbed OS Giga Boards by
sketch_jan8a.ino
/* == */
void loop() {
unsigned long now = millis();
// Kirim data sensor ke backend
if (now - lastSend >= SEND_INTERVAL) {
lastSend = now;
readSensorAndSend();
}
// Ambil settings dari backend
if (now - lastGet >= SETTINGS_INTERVAL) {
lastGet = now;
getSettings();
}
delay(100);
}
/* ================ READ & SEND ================ */
void readSensorAndSend() {
// Baca sensor
lux = lightMeter.readLightLevel();
Output Serial Monitor
indexing: 11/57
ESP32 Dev Module on COM19 [not connected]
Search
0406
13/01/2026


sketch_jan8a | Arduino IDE 2.3.6
File Edit Sketch Tools Help
ESP32 Dev Module
BOARDS MANAGER
Arduino AVR Boards by Arduino
1.8.6 installed
REMOVE
Arduino ESP32 Boards by Arduino
2.0.18-arduino.5 installed
REMOVE
Arduino Mbed OS Edge Boards by Arduino
INSTALL
Arduino Mbed OS Giga Boards by
sketch_jan8a.ino
// Tampilkan status berdasarkan threshold
if (lux < luxThreshold) {
Serial.print(" [GELAP - Relay should be ON]");
} else {
Serial.print(" [TERANG - Relay should be OFF]");
}
Serial.println();
Serial.print(" Relay: ");
Serial.print(relayState ? "ON " : "OFF ");
Serial.print(" | Mode: ");
Serial.print(mode);
Serial.print(" | Threshold: ");
Serial.print(luxThreshold);
Serial.println(" lux");
// Kirim ke backend
if (WiFi.status() != WL_CONNECTED) {
Serial.println(" WiFi disconnected!");
return;
}
HTTPClient http;
http.begin(dataUrl);
Output Serial Monitor
ESP32 Dev Module on COM19 [not connected]
Search
04:06
13/01/2026


sketch_jan8a | Arduino IDE 2.3.6
File Edit Sketch Tools Help
ESP32 Dev Module
BOARDS MANAGER
Arduino AVR Boards by Arduino
1.8.6 installed
REMOVE
Arduino ESP32 Boards by Arduino
2.0.18-arduino.5 installed
REMOVE
Arduino Mbed OS Edge Boards by Arduino
INSTALL
Arduino Mbed OS Giga Boards by
sketch_jan8a.ino
// Buat JSON payload
String payload = "{";
payload += "\"lux\":" + String(lux, 2) + ",";
payload += "\"relay_status\":" + String(relayState ? "true" : "false");
payload += "}";
Serial.print(" Sending... ");
// Kirim POST request
int code = http.POST(payload);
if (code > 0) {
Serial.print(" OK (");
Serial.print(code);
Serial.println(")");
// Parse response dari backend
String response = http.getString();
StaticJsonDocument<1024> doc;
DeserializationError error = deserializeJson(doc, response);
if (!error) {
// Backend response structure:
Output Serial Monitor
ESP32 Dev Module on COM19 [not connected]
Search
04:06
13/01/2026


sketch_jan8a | Arduino IDE 2.3.6
File Edit Sketch Tools Help
ESP32 Dev Module
BOARDS MANAGER
Arduino AVR Boards by Arduino
1.8.6 installed
REMOVE
Arduino ESP32 Boards by Arduino
2.0.18-arduino.5 installed
REMOVE
Arduino Mbed OS Edge Boards by Arduino
INSTALL
Arduino Mbed OS Giga Boards by
sketch_jan8a.ino
/* ---------- SET RELAY ---------- */
void setRelay(bool state) {
if (state != relayState) {
relayState = state;
// ACTIVE LOW: Relay module dengan optocoupler biasanya ACTIVE LOW
// true (ON) = LOW = Relay tertutup, lampu NYALA
// false (OFF) = HIGH = Relay terbuka, lampu MATI
//
// Jika relay Anda ACTIVE HIGH (jarang), ubah menjadi:
// digitalWrite(RELAY_PIN, state ? HIGH : LOW);
digitalWrite(RELAY_PIN, state ? LOW : HIGH); // ACTIVE LOW
Serial.println("\n *** RELAY CHANGED ***");
Serial.print(" Relay is now: ");
Serial.println(state ? "ON (Lampu NYALA)" : "OFF (Lampu MATI)");
Serial.print(" GPIO Pin: ");
Serial.println(state ? "LOW" : "HIGH");
Serial.println(" **********");
}
}
Output Serial Monitor
ESP32 Dev Module on COM19 [not connected]
Search
04:06
13/01/2026

BAB 4
Kesimpulan

Berdasarkan hasil perancangan dan implementasi yang telah dilakukan, sistem monitoring dan pengendalian lampu otomatis berbasis IoT menggunakan ESP32 dan sensor BH1750 berhasil direalisasikan dengan baik. Sensor BH1750 mampu mengukur intensitas cahaya lingkungan secara akurat dan stabil dalam satuan lux, sehingga sistem dapat menentukan kondisi terang atau gelap dengan lebih presisi dibandingkan sensor analog. ESP32 berperan sebagai pusat kendali yang mengolah data sensor serta mengontrol modul relay untuk menyalakan atau mematikan lampu secara otomatis.

Integrasi perangkat IoT dengan sistem backend berbasis Node.js dan database MySQL memungkinkan data sensor dan status relay disimpan serta dikelola secara terpusat. Melalui dashboard web berbasis React, pengguna dapat melakukan monitoring data intensitas cahaya secara real-time maupun historis, serta melakukan pengendalian lampu secara manual sesuai dengan hak akses yang dimiliki. Mekanisme autentikasi dan otorisasi juga memastikan keamanan akses sistem.

Secara keseluruhan, sistem ini memberikan manfaat berupa peningkatan efisiensi energi, kemudahan monitoring jarak jauh, serta automasi pengendalian lampu yang andal. Dengan adanya logika kontrol lokal pada ESP32, sistem tetap dapat berfungsi meskipun koneksi internet terputus. Sistem yang dikembangkan ini diharapkan dapat menjadi solusi penerangan cerdas yang dapat dikembangkan lebih lanjut untuk skala yang lebih besar.

Daftar Pustaka

[1] Sethi, P., & Sarangi, S. R. (2017). Internet of Things: Architectures, Protocols, and Applications. *Journal of Electrical and Computer Engineering*, 2017, 1-25. https://doi.org/10.1155/2017/9324035

[2] Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. *Future Generation Computer Systems*, 29(7), 1645-1660. https://doi.org/10.1016/j.future.2013.01.010

[3] Singh, D., Tripathi, G., & Jara, A. J. (2014). A Survey of Internet-of-Things: Future Vision, Architecture, Challenges and Services. *IEEE World Forum on Internet of Things (WF-IoT)*, 287-292. https://doi.org/10.1109/WF-IoT.2014.6803174

[4] Patel, K. K., & Patel, S. M. (2016). Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges. *International Journal of Engineering Science and Computing*, 6(5), 6122-6131.

[5] Kodali, R. K., Swamy, G., & Lakshmi, B. (2015). An Implementation of IoT for Healthcare. *IEEE Recent Advances in Intelligent Computational Systems (RAICS)*, 411-416. https://doi.org/10.1109/RAICS.2015.7488451

[6] Minerva, R., Biru, A., & Rotondi, D. (2015). Towards a Definition of the Internet of Things (IoT). *IEEE Internet Initiative*, 1(1), 1-86.

[7] Saini, H., Thakur, A., Ahuja, S., Sabharwal, N., & Kumar, N. (2016). Arduino Based Automatic Wireless Weather Station with Remote Graphical Application and Alerts. *3rd International Conference on Signal Processing and Integrated Networks (SPIN)*, 605-609. https://doi.org/10.1109/SPIN.2016.7566773

[8] Leccese, F., Cagnetti, M., & Trinca, D. (2014). A Smart City Application: A Fully Controlled Street Lighting Isle Based on Raspberry-Pi Card, a ZigBee Sensor Network and WiMAX. *Sensors*, 14(12), 24408-24424. https://doi.org/10.3390/s141224408

[9] Fielding, R. T., & Taylor, R. N. (2002). Principled Design of the Modern Web Architecture. *ACM Transactions on Internet Technology (TOIT)*, 2(2), 115-150. https://doi.org/10.1145/514183.514185

[10] Jones, M., Bradley, J., & Sakimura, N. (2015). JSON Web Token (JWT). *RFC 7519, Internet Engineering Task Force*. https://doi.org/10.17487/RFC7519

Lampiran

- Activity Diagram
  https://github.com/C-PPAW-TI503P-2025/Project-Kelompok6-Cahaya/blob/main/diagramproject/activity.puml
- Arsitektur Diagram
  https://github.com/C-PPAW-TI503P-2025/Project-Kelompok6-Cahaya/blob/main/diagramproject/arsitektur.puml
- Class Diagram
  https://github.com/C-PPAW-TI503P-2025/Project-Kelompok6-Cahaya/blob/main/diagramproject/class.puml
- ERD (Entity Relationship Diagram)
  https://github.com/C-PPAW-TI503P-2025/Project-Kelompok6-Cahaya/blob/main/diagramproject/erd.puml
- Flowchart Diagram
  https://github.com/C-PPAW-TI503P-2025/Project-Kelompok6-Cahaya/blob/main/diagramproject/flowchart.puml
- Sequence Diagram 1
  https://github.com/C-PPAW-TI503P-2025/Project-Kelompok6-Cahaya/blob/main/diagramproject/sequence_1.puml
- Sequence Diagram 2
  https://github.com/C-PPAW-TI503P-2025/Project-Kelompok6-Cahaya/blob/main/diagramproject/sequence_2.puml
- Use Case Diagram
  https://github.com/C-PPAW-TI503P-2025/Project-Kelompok6-Cahaya/blob/main/diagramproject/usecase.puml