

**LAPORAN TEMPAT SAMPAH DENGAN  
PENDETEKSI GERAKAN  
PENGEMBANGAN APLIKASI WEB**



**Nama Anggota:**

1. Pradipa Yogananda 20230140103
2. Imron Satrio Purnomo 20230140122
3. M. Alfiansyah Azad 20230140126
4. Harits Zhafran Nayotama 20230140141
5. Muhammad Razzin Ayyuri 20230140121
6. Rafli Ramadhani Oktavianto Khufi 20230140127

**FAKULTAS TEKNIK  
PROGRAM STUDI TEKNOLOGI INFORMASI  
UNIVERSITAS MUHAMMADIYAH YOGYAKARTA  
2025/2026**

# DAFTAR ISI

DAFTAR ISI .....	2
BAB 1 PENDAHULUAN.....	2
1.1 Latar Belakang .....	2
1.2 Rumusan Masalah .....	2
1.3 Tujuan Penelitian .....	2
1.4 Manfaat .....	2
BAB 2 UML .....	3
2.1 Arsitektur.....	3
2.2 Class Diagram .....	3
2.3 Entity Relationship Diagram .....	4
2.4 Use Case Diagram.....	5
2.5 Sequence Diagram.....	5
2.6 Flowchart Sistem .....	6
2.7 Activity Diagram .....	7
BAB 3 APLIKASI.....	9
3.1 Deskripsi Aplikasi.....	9
3.2 Teknologi yang Digunakan.....	9
3.3 Fitur Aplikasi .....	9
3.4 Implementasi Sistem .....	9
3.5 Pengujian Sistem .....	9
3.6 Tampilan Halaman Web/Aplikasi.....	9
BAB IV KESIMPULAN .....	11
DAFTAR PUSTAKA .....	11
LAMPIRAN .....	11

# BAB 1 PENDAHULUAN

## 1.1 Latar Belakang

Perkembangan teknologi informasi dan Internet of Things (IoT) mendorong terciptanya solusi cerdas untuk permasalahan sehari-hari, salah satunya pengelolaan sampah. Tempat sampah konvensional masih memiliki keterbatasan, seperti kontak fisik langsung dan kurangnya efisiensi pemantauan. Oleh karena itu, dikembangkan sistem **Tempat Sampah dengan Pendekripsi Gerakan** yang terintegrasi dengan **aplikasi web** untuk memantau dan mengelola data secara digital.

## 1.2 Rumusan Masalah

1. Bagaimana merancang sistem tempat sampah dengan pendekripsi gerakan?
2. Bagaimana memodelkan sistem menggunakan UML?
3. Bagaimana mengimplementasikan aplikasi web sebagai media monitoring dan pengelolaan data?

## 1.3 Tujuan Penelitian

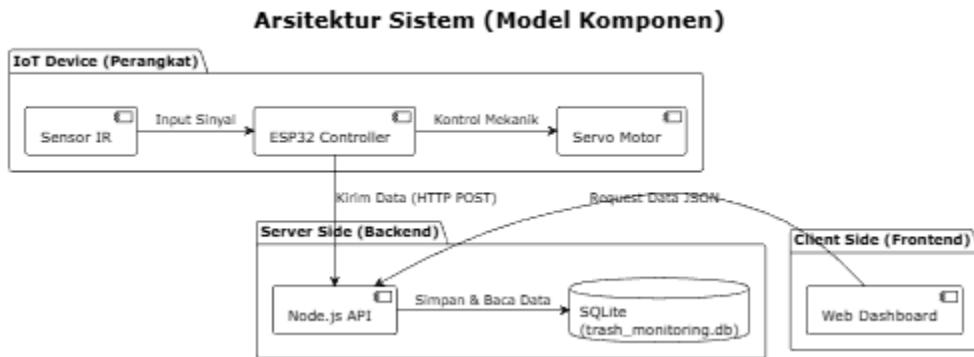
1. Merancang sistem tempat sampah otomatis berbasis pendekripsi gerakan.
2. Membuat pemodelan sistem menggunakan UML.
3. Mengembangkan aplikasi web untuk monitoring dan pengelolaan data.

## 1.4 Manfaat

- Meningkatkan kebersihan dan kenyamanan pengguna.
- Memudahkan pengelolaan dan pemantauan tempat sampah.
- Sebagai media pembelajaran pengembangan aplikasi web.

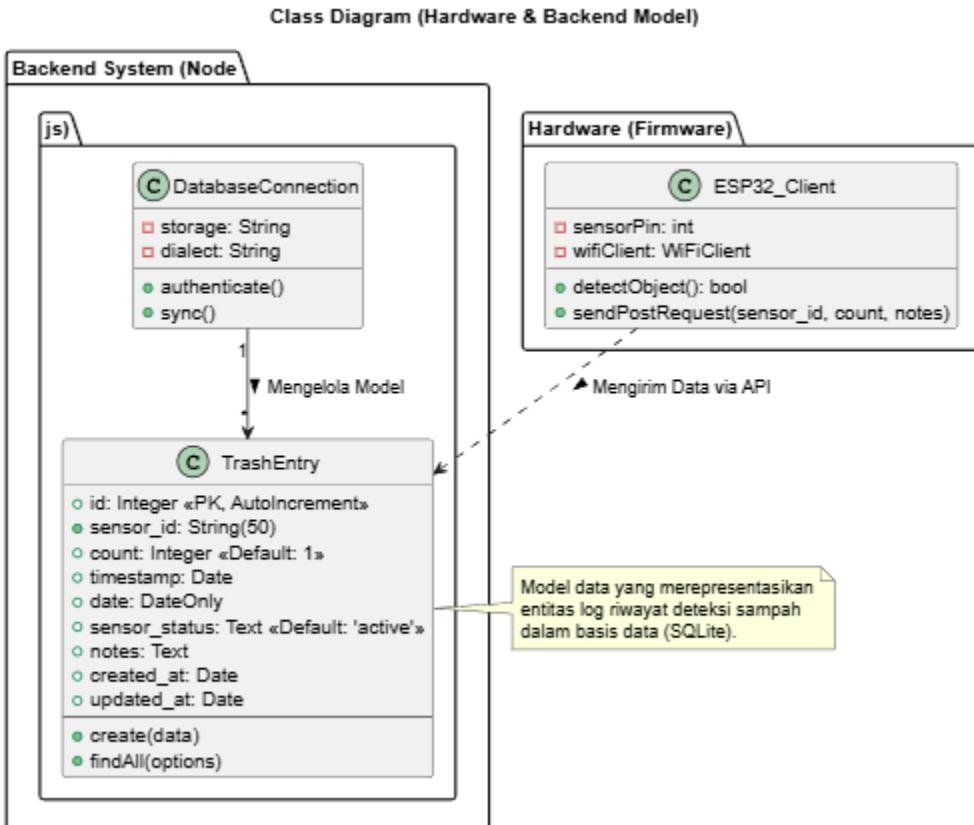
# BAB 2 UML

## 2.1 Arsitektur



Deskripsi: Diagram arsitektur ini dirancang menggunakan model komponen untuk menggambarkan topologi sistem secara sederhana. Sensor IR dan Motor Servo terhubung ke ESP32 sebagai pengendali utama. ESP32 mengirimkan data deteksi melalui protokol HTTP ke backend berbasis Node.js. Data tersebut kemudian disimpan ke dalam database lokal SQLite (trash\_monitoring.db) dan ditampilkan ke pengguna melalui Web Dashboard.

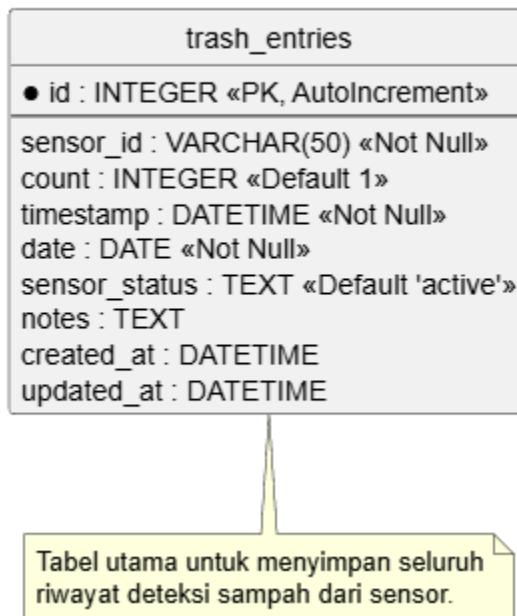
## 2.2 Class Diagram



Penjelasan: Class diagram menggambarkan struktur kelas pada sisi firmware dan backend. Kelas ESP32\_Client menangani pembacaan sensor dan komunikasi jaringan. Di sisi server, kelas TrashEntry merepresentasikan model data yang dibangun menggunakan Sequelize ORM. Atribut kelas ini (seperti sensor\_id, count, notes) disesuaikan dengan kebutuhan pencatatan log pada database SQLite.

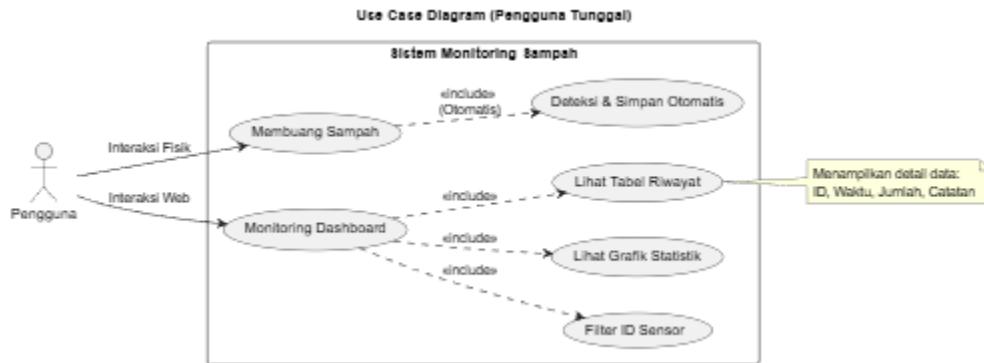
## 2.3 Entity Relationship Diagram

**ERD - Skema Database SQLite**



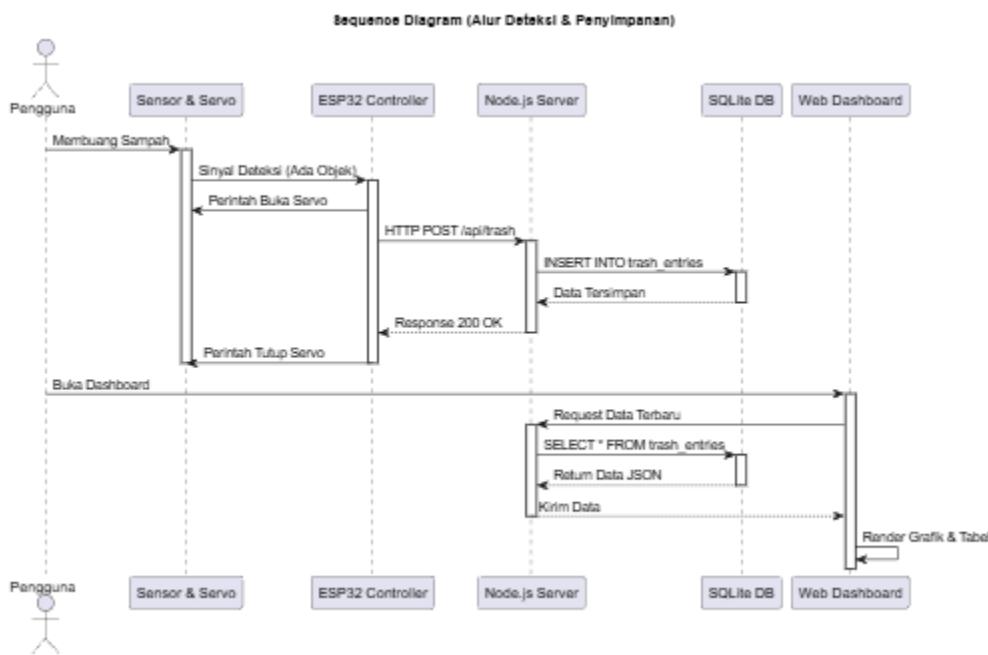
Penjelasan: ERD ini menjelaskan desain basis data SQLite yang digunakan. Sistem hanya memiliki satu tabel utama yaitu `trash_entries`. Kolom-kolom kunci meliputi `sensor_id` (identitas alat), `timestamp` (waktu kejadian), dan `count` (jumlah deteksi). Struktur tabel ini dirancang untuk mendukung fitur visualisasi grafik dan tabel riwayat pada dashboard web.

## 2.4 Use Case Diagram



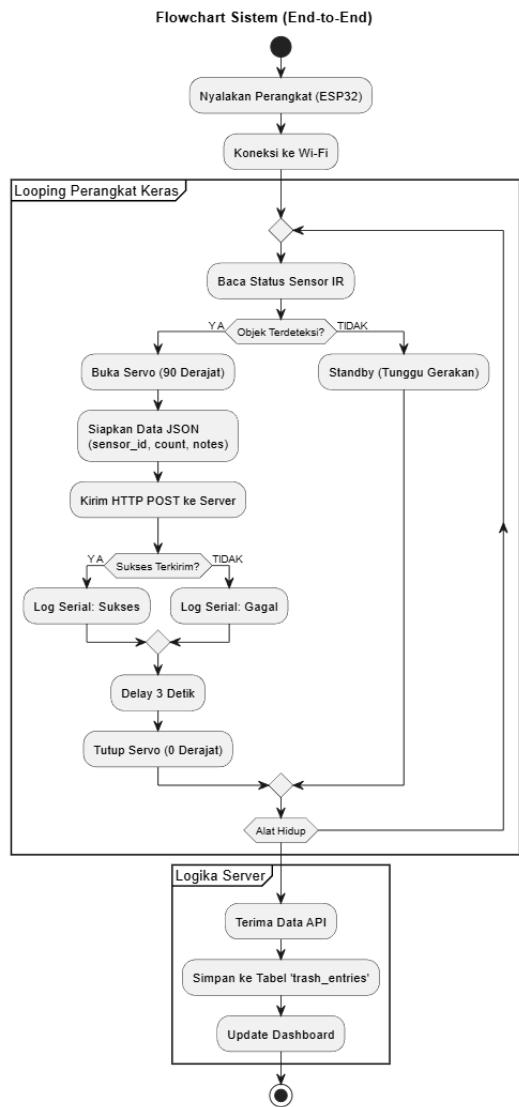
Penjelasan: Use Case Diagram menunjukkan interaksi antara Pengguna dengan sistem. Pengguna memiliki peran ganda: melakukan interaksi fisik (membuang sampah) yang memicu pencatatan otomatis, dan melakukan interaksi digital (monitoring) melalui web dashboard. Fitur web mencakup melihat grafik statistik mingguan, memfilter data sensor, dan memeriksa tabel riwayat deteksi.

## 2.5 Sequence Diagram



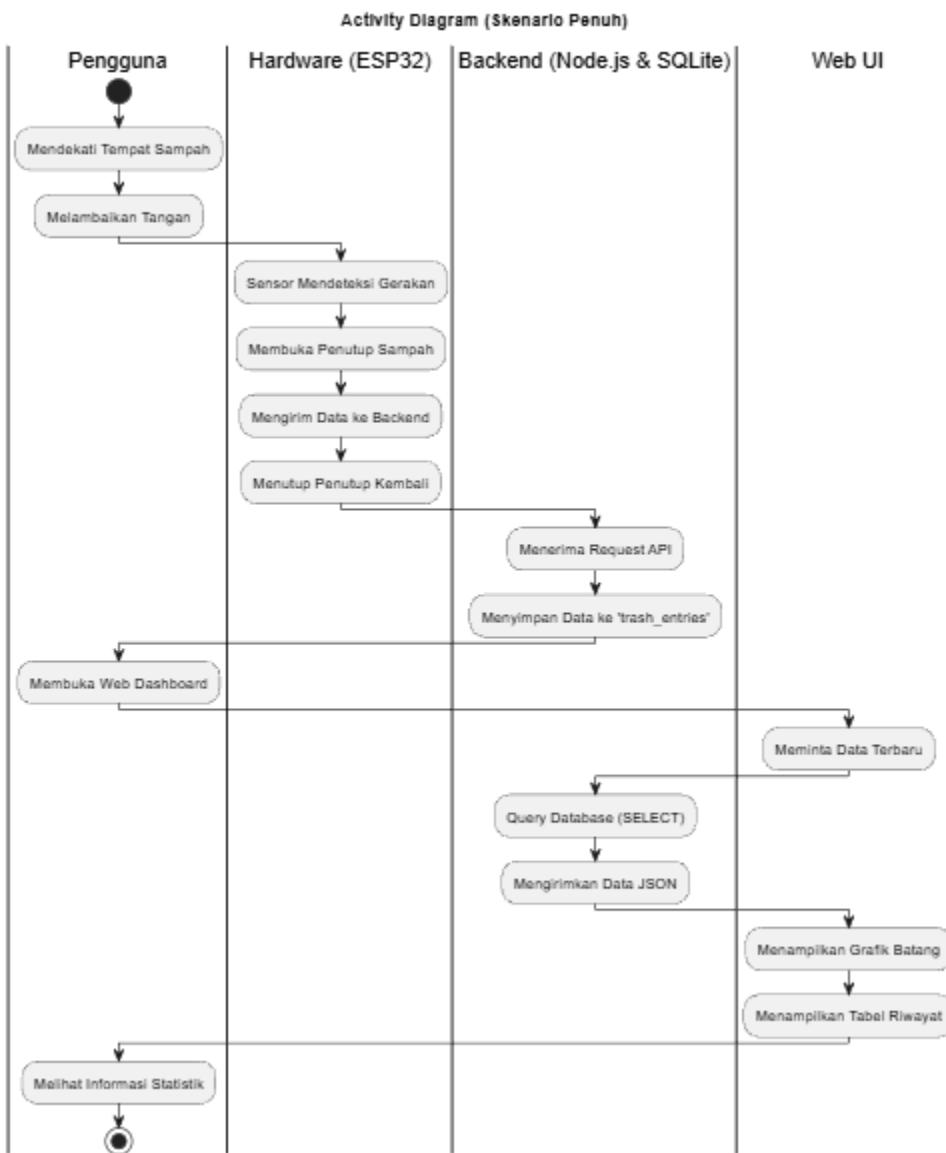
Penjelasan: Diagram ini merinci urutan proses berdasarkan waktu. Ketika sensor mendeteksi objek, ESP32 menggerakkan motor servo sekaligus mengirim data ke server Node.js. Server kemudian menyimpan data tersebut ke database SQLite. Diagram juga memvisualisasikan proses saat pengguna membuka dashboard, di mana sistem mengambil data terbaru dari database untuk ditampilkan.

## 2.6 Flowchart Sistem



Penjelasan: Flowchart menggambarkan algoritma yang berjalan pada mikrokontroler. Setelah terhubung ke Wi-Fi, sistem masuk ke perulangan utama (loop) untuk memantau sensor. Jika gerakan terdeteksi, sistem melakukan aksi mekanik (buka tutup) dan aksi jaringan (kirim data ke server). Logika ini memastikan data hanya dikirim saat ada aktivitas nyata, untuk efisiensi penyimpanan database.

## 2.7 Activity Diagram



Penjelasan: Activity Diagram memetakan alur kerja sistem secara menyeluruh menggunakan swimlanes untuk memisahkan tanggung jawab. Diagram ini memperjelas batasan tugas antara aktivitas fisik pengguna, respon otomatis perangkat keras, pemrosesan data di server backend, hingga visualisasi informasi pada antarmuka web.

## BAB 3 APLIKASI

### 3.1 Deskripsi Aplikasi

Aplikasi web tempat sampah dengan pendekripsi gerakan berfungsi sebagai media monitoring dan pengelolaan data. Aplikasi ini menampilkan informasi status tempat sampah secara real-time.

### 3.2 Teknologi yang Digunakan

- Bahasa Pemrograman: HTML, CSS, JavaScript, dan PHP
- Basis Data: MySQL
- Perangkat Pendukung: Sensor pendekripsi gerakan

### 3.3 Fitur Aplikasi

1. Monitoring status tempat sampah
2. Pengelolaan data pengguna
3. Tampilan data berbasis web

### 3.4 Implementasi Sistem

Implementasi sistem dilakukan dengan mengintegrasikan perangkat keras (sensor) dengan aplikasi web melalui basis data.

### 3.5 Pengujian Sistem

Pengujian dilakukan untuk memastikan setiap fitur berjalan sesuai dengan fungsinya.

### 3.6 Tampilan Halaman Web/Aplikasi

## IR Trash Monitoring System

Filter Sensor:  Refresh Last update: 05.05.01

Hari Ini  
**3**

Total  
**3**

Entry Hari Ini  
**3**

Total Entry  
**3**

**Grafik 7 Hari Terakhir**  


A bar chart titled 'Grafik 7 Hari Terakhir' showing the number of detections over the last 7 days. The y-axis represents the count from 0 to 3. A single blue bar reaches the value of 3. The x-axis is labeled 'Rab, 14 Jan'. A legend indicates the blue bar represents 'Jumlah Deteksi'.

Riwayat DeteksiPreviousPage 1 of 1Next

ID	Sensor ID	Tanggal	Waktu	Count	Notes
3	BIN-01	14/1/2026	16.57.43	1	Sample data 3
2	BIN-02	14/1/2026	16.57.40	1	Sample data 2
1	BIN-01	14/1/2026	16.57.37	1	Sample data 1

API Status: Connected

### Implementasi Antarmuka Dashboard

1. Halaman dashboard IR Trash Monitoring System berfungsi sebagai pusat pemantauan data sensor secara real-time. Tampilan antarmuka terdiri dari empat komponen utama:
2. Panel Filter & Statistik: Bagian atas memuat fitur Filter Sensor (opsi: Semua, BIN-01, BIN-02, BIN-03) untuk memilih data berdasarkan perangkat spesifik. Disandingkan dengan kartu ringkasan yang menampilkan jumlah deteksi hari ini dan total keseluruhan.

3. Grafik Mingguan: Bagian tengah menampilkan Grafik 7 Hari Terakhir dalam bentuk diagram batang, memberikan visualisasi tren aktivitas pembuangan sampah selama satu minggu ke belakang.
4. Tabel Riwayat Deteksi: Bagian bawah menyajikan log data detail dalam format tabel. Kolom yang ditampilkan meliputi ID, Sensor ID, Tanggal, Waktu, Count, dan Notes, sesuai dengan data yang dikirimkan oleh mikrokontroler.
5. Status Koneksi: Pada bagian bawah halaman (footer), terdapat indikator API Status berwarna hijau ("Connected") yang menandakan sistem terhubung stabil dengan server backend.

## BAB IV KESIMPULAN

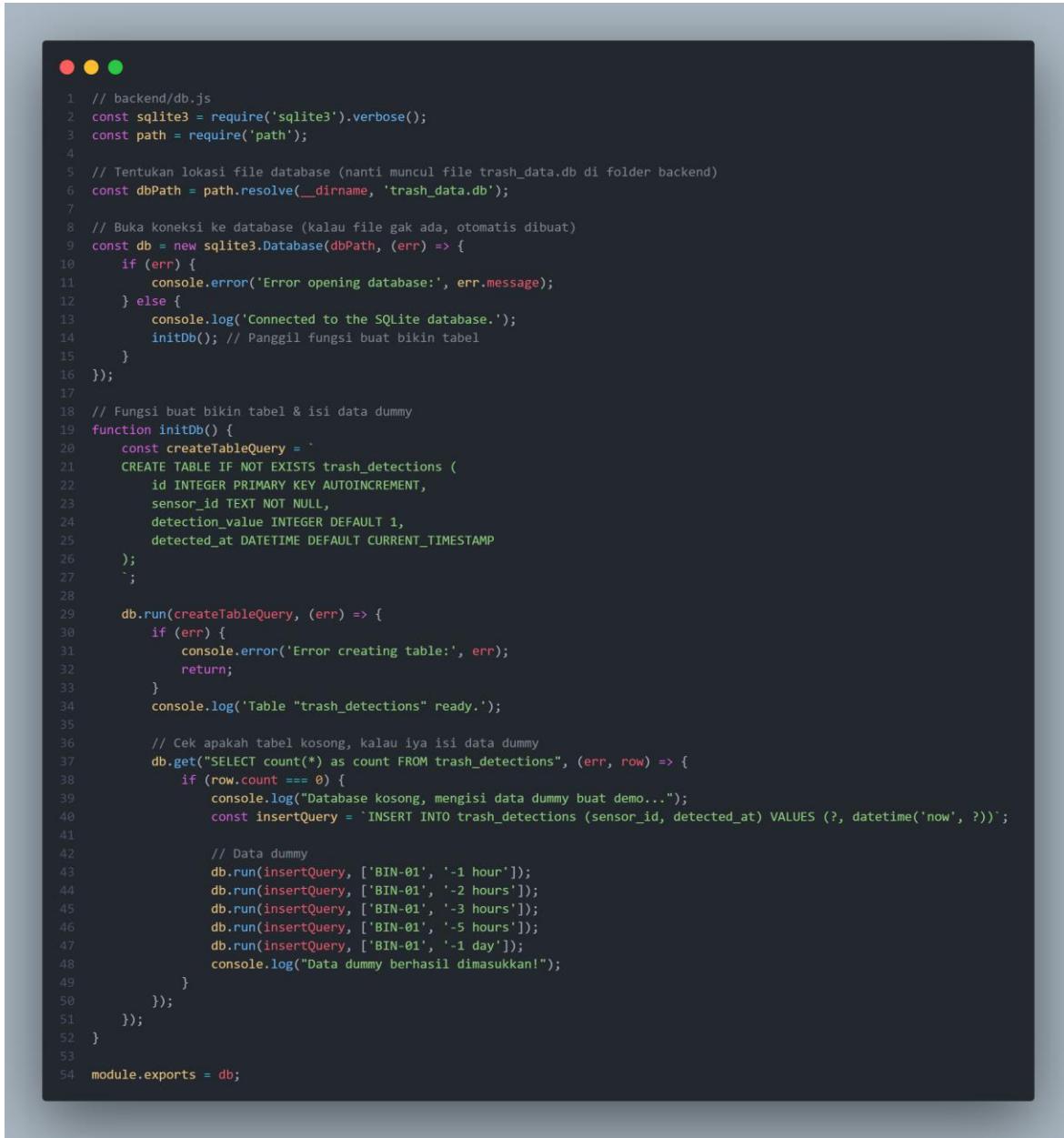
Berdasarkan hasil perancangan dan implementasi, dapat disimpulkan bahwa aplikasi web tempat sampah dengan pendekripsi gerakan berhasil dikembangkan dan dapat membantu pengelolaan sampah secara lebih efisien. Sistem ini juga menjadi sarana pembelajaran yang baik dalam pengembangan aplikasi web.

## DAFTAR PUSTAKA

1. Pressman, R. S. (2015). *Software Engineering*. McGraw-Hill.
2. Sommerville, I. (2016). *Software Engineering*. Pearson Education.

# LAMPIRAN

## File db.js



```
1 // backend/db.js
2 const sqlite3 = require('sqlite3').verbose();
3 const path = require('path');
4
5 // Tentukan lokasi file database (nanti muncul file trash_data.db di folder backend)
6 const dbPath = path.resolve(__dirname, 'trash_data.db');
7
8 // Buka koneksi ke database (kalau file gak ada, otomatis dibuat)
9 const db = new sqlite3.Database(dbPath, (err) => {
10     if (err) {
11         console.error('Error opening database:', err.message);
12     } else {
13         console.log('Connected to the SQLite database.');
14         initDb(); // Panggil fungsi buat bikin tabel
15     }
16 });
17
18 // Fungsi buat bikin tabel & isi data dummy
19 function initDb() {
20     const createTableQuery = `
21     CREATE TABLE IF NOT EXISTS trash_detections (
22         id INTEGER PRIMARY KEY AUTOINCREMENT,
23         sensor_id TEXT NOT NULL,
24         detection_value INTEGER DEFAULT 1,
25         detected_at DATETIME DEFAULT CURRENT_TIMESTAMP
26     );
27 `;
28
29     db.run(createTableQuery, (err) => {
30         if (err) {
31             console.error('Error creating table:', err);
32             return;
33         }
34         console.log('Table "trash_detections" ready.');
35
36         // Cek apakah tabel kosong, kalau iya isi data dummy
37         db.get("SELECT count(*) as count FROM trash_detections", (err, row) => {
38             if (row.count === 0) {
39                 console.log("Database kosong, mengisi data dummy buat demo...");
40                 const insertQuery = `INSERT INTO trash_detections (sensor_id, detected_at) VALUES (?, datetime('now', ?))`;
41
42                 // Data dummy
43                 db.run(insertQuery, ['BIN-01', '-1 hour']);
44                 db.run(insertQuery, ['BIN-01', '-2 hours']);
45                 db.run(insertQuery, ['BIN-01', '-3 hours']);
46                 db.run(insertQuery, ['BIN-01', '-5 hours']);
47                 db.run(insertQuery, ['BIN-01', '-1 day']);
48                 console.log("Data dummy berhasil dimasukkan!");
49             }
50         });
51     });
52 }
53
54 module.exports = db;
```

## File TrashSensorMain.ino



```
#define SENSOR_PIN_25 // Pin reading atten sensor 2B (GPIO 16 - ADC)
#define DEBOUNCE_TIME 200 // Debounce time
#define MAX_DEBOUNCE 5000 // Maximum digital interval output
// Variables
unsigned long lastInstitutionTime = 0;
int inPinState = 0;
int attempts = 0;
const unsigned long SEND_INTERVAL = 5000; // Circle data setup & serial
// WiFi & API Configuration
const char* ssid = "WIFI_SSID";
const char* password = "WIFI_PASSWORD";
const char* servername = "http://192.168.1.200:5000/api/trash/entries";
const char* sensorid = "TRASH_SENSOR";
// Libraries and global variables
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266HTTPUpdateServer.h>
#include <ESP8266HTTPServer.h>
#include <ESP8266HTTPClient.h>
#include <ESP8266mDNS.h>
#include <ArduinoJson.h>
#include <Arduino.h>
void setup() {
    Serial.begin(115200);
    delay(1000);
    Serial.println("Trash Sensor IR Sensor Detector Initialized ---");
    Serial.print("Connecting to WiFi....");
    // Initialize pins sensor
    pinMode(SENSOR_PIN, INPUT);
    // Connect to WiFi
    WiFi.begin(ssid, password);
    // WiFi loop
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        attempts++;
        if (attempts > 10) {
            Serial.print("Error connecting to WiFi!");
            attempts = 0;
        }
    }
    if (WiFi.status() == WL_CONNECTED) {
        Serial.println(" WiFi Connected");
        Serial.print("IP: ");
        Serial.println(WiFi.localIP());
    } else {
        Serial.println(" WiFi Connection Failed");
    }
    Serial.println("System Ready");
}
void loop() {
// Read sensor
int sensorValue = analogRead(SENSOR_PIN);
// If sensor value is lower than threshold
if (sensorValue < 1000) {
    static unsigned long lastInstitutionTime = 0;
    if (millis() - lastInstitutionTime > SEND_INTERVAL) {
        lastInstitutionTime = millis();
        Serial.println(sensorValue);
        inPinState = 1;
    }
}
// Detach sensor (this is sensor 2B message by myself)
if (inPinState == 1) {
    if (millis() - lastInstitutionTime > DEBOUNCE_TIME) {
        lastInstitutionTime = millis();
        Serial.println("Sample Value: " + String(inPinState));
        Serial.println("Sample Value Total: " + String(attempts));
        // Optional: Blue or LED indicator
        // digitalWrite(led_PIR, HIGH);
        // delay(1000);
        // digitalWrite(led_PIR, LOW);
        // delay(1000);
    }
}
// Write data to WiFi setup (END_INTERVAL)
if (millis() - lastInstitutionTime > END_INTERVAL || WiFi.status() == WL_CONNECTED) {
    WiFiClient client;
    semphover(transcount);
    transcount = 0; // Reset counter
    lastInstitutionTime = millis();
}
// Connect WiFi (End loop)
if (WiFi.status() == WL_CONNECTED) {
    Serial.println("WiFi Disconnected. Reconnecting...");
    WiFi.disconnect();
}
delay(100); // Small delay until stability
}
void handleServer(int count) {
    WiFiClient http;
    // Post JSON payload
    String doc = "{'sensor_id': sensorID,";
    doc += "'sensor_val': " + String(inPinState);
    doc += "'time': " + String(millis());
    doc += "}";
    http.println(doc);
    http.println("Content-Type: application/json");
    http.println("Content-Length: " + String(doc.length()));
    for (int i = 0; i < count; i++) {
        int httpResponseCode = http.post(doc);
        if (httpResponseCode > 0) {
            Serial.println("Entry " + String(i));
            Serial.println("Status: " + String(httpResponseCode));
            Serial.println("Reason: " + String(httpReason));
        } else {
            Serial.println("Error sending entry");
            Serial.println("Status: " + String(httpResponseCode));
            Serial.println("Reason: " + String(httpReason));
        }
    }
    delay(1000); // Delay before request
}
http.end();
}
```

## File route.js

```
1 import { NextResponse } from 'next/server';
2 import { sequelize, ensureDbReady } from '@/lib/db';
3
4 export const runtime = 'nodejs';
5
6 function addDays(dateStr, delta) {
7   const d = new Date(dateStr + 'T00:00:00.000Z');
8   d.setUTCDate(d.getUTCDate() + delta);
9   return d.toISOString().slice(0, 10);
10 }
11
12 export async function GET(req) {
13   await ensureDbReady();
14
15   const { searchParams } = new URL(req.url);
16   const days = Math.min(90, Math.max(1, parseInt(searchParams.get('days') || '7', 10)));
17   const sensor_id = searchParams.get('sensor_id');
18
19   const end = new Date().toISOString().slice(0, 10);
20   const start = addDays(end, -(days - 1));
21
22   const whereSql = sensor_id ? 'AND sensor_id = :sensor_id' : '';
23   const replacements = sensor_id ? { start, end, sensor_id } : { start, end };
24
25   const [rows] = await sequelize.query(
26     `SELECT date, COALESCE(SUM(count), 0) as total
27     FROM trash_entries
28     WHERE date BETWEEN :start AND :end ${whereSql}
29     GROUP BY date
30     ORDER BY date ASC`,
31     { replacements }
32   );
33
34   // biar tanggal yang gak ada data tetap muncul (0)
35   const map = new Map(rows.map(r => [r.date, Number(r.total || 0)]));
36   const series = [];
37   for (let i = 0; i < days; i++) {
38     const d = addDays(start, i);
39     series.push({ date: d, total: map.get(d) ?? 0 });
40   }
41
42   return NextResponse.json({
43     success: true,
44     data: { start, end, days, series },
45   });
46 }
```