# **LAPORAN PRAKTIKUM**

# **Membuat Tampilan Interface Web Dashboard IoT**

**Fakultas Vokasi, Universitas Brawijaya**

## *Author(s):*

*Wildan Aridh Takhfif  
 Fakultas Vokasi, Universitas Brawijaya  
 Email: apipkph@studen.ub.ac.id*

## **Abstract**

## Pada praktikum ini, kami mengimplementasikan tampilan antarmuka (interface) untuk dashboard monitoring sistem Internet of Things (IoT). Menggunakan Laravel sebagai framework backend dan Chart.js untuk visualisasi data, dashboard ini menampilkan grafik perbandingan sensor dan memungkinkan pengguna untuk mengunduh data transaksi sensor dalam format Excel. Dengan menggunakan Laravel untuk backend dan paket maatwebsite/excel untuk ekspor data, serta Chart.js untuk grafik, sistem ini memungkinkan pengguna untuk memonitor data sensor secara real-time. Hasil praktikum menunjukkan bahwa dashboard berhasil menampilkan grafik interaktif dan mendukung ekspor data dalam format Excel.

## *Keywords—Internet of Things, Laravel, Dashboard, Chart.js, Excel Export*

## **1. Introduction**

### **1.1 Latar belakang praktikum IoT yang dilakukan**

Pada perkembangan teknologi terkini, sistem Internet of Things (IoT) memainkan peran penting dalam berbagai sektor, termasuk pemantauan data secara real-time. Untuk mempermudah pemantauan data sensor dari perangkat IoT seperti ESP32, pembuatan sebuah dashboard berbasis web sangatlah berguna. Dashboard ini memungkinkan pengguna untuk memvisualisasikan data dalam bentuk grafik dan mengelola data secara efisien dengan ekspor ke dalam format Excel.

### **1.2 Tujuan eksperimen**

Tujuan dari praktikum ini adalah untuk membuat sebuah dashboard berbasis web yang dapat menampilkan grafik sensor IoT yang dikirimkan melalui ESP32, serta memungkinkan pengguna untuk mengekspor data transaksi sensor ke dalam format Excel.

## **2. Methodology**

### **2.1 Tools & Materials**

* **Mikrokontroler**: ESP32
* **Software**: Laravel, Chart.js, Maatwebsite Excel, PHP
* **Paket Laravel**: Maatwebsite/Excel untuk ekspor data, dan Chart.js untuk grafik
* **Database**: MySQL (untuk menyimpan data transaksi sensor)

### **2.2 Implementation Steps**

**Instalasi dan Konfigurasi Laravel**:

* Menginstal paket maatwebsite/excel untuk ekspor data ke Excel.
* Membuat controller GraphController untuk menangani pengambilan data transaksi sensor dan menampilkan grafik.

**Buat Export untuk Data Sensor**:

* Menggunakan Laravel Artisan untuk membuat TransaksiSensorExport yang memungkinkan ekspor data transaksi sensor ke dalam file Excel.

**Desain dan Implementasi Tampilan (View)**:

* Membuat tampilan menggunakan Blade (Laravel), dengan Chart.js untuk visualisasi grafik dan tombol ekspor untuk mengunduh data.

**Pengujian**:

* Menjalankan server Laravel menggunakan perintah php artisan serve, lalu mengakses dashboard untuk memverifikasi apakah grafik data sensor tampil dengan benar dan ekspor data ke Excel berfungsi.

## **3. Results and Discussion**

### **3.1 Experimental Results**

Setelah mengikuti langkah-langkah implementasi, berikut adalah hasil yang didapatkan:

* **Grafik Sensor**:  
   Dashboard berhasil menampilkan grafik perbandingan antara dua sensor menggunakan Chart.js. Grafik ini diperbarui dengan data terbaru dari database yang didapatkan dari sensor yang terhubung ke ESP32.
* **Fitur Ekspor Excel**:  
   Pengguna dapat mengekspor data transaksi sensor dalam format Excel menggunakan tombol "Export". Fitur ini memungkinkan data sensor untuk diunduh dan dianalisis lebih lanjut menggunakan perangkat lunak spreadsheet.
* **Tampilan Dashboard**:  
   Antarmuka dashboard dirancang dengan layout yang bersih dan responsif, menggunakan CSS untuk desain dan animasi dari animate.css. Pengguna dapat memilih periode data yang ditampilkan (24 Jam, 7 Hari, 30 Hari, atau Custom).

**Tabel 1: Data Transaksi Sensor yang Tersimpan di Database**

| **Nama Sensor** | **Nilai1** | **Nilai2** |
| --- | --- | --- |
| Sensor 1 | 22.4 | 45.6 |
| Sensor 2 | 20.1 | 44.0 |

### **3.2 Discussion**

Dashboard yang dibangun dapat menampilkan data sensor secara visual dengan menggunakan grafik interaktif. Ekspor data dalam format Excel memungkinkan pengguna untuk menyimpan data untuk keperluan analisis lebih lanjut. Fitur-fitur ini sangat berguna dalam pengembangan aplikasi IoT untuk pemantauan real-time. Dengan menggunakan Laravel dan Chart.js, kita dapat membuat sistem yang efisien dan mudah digunakan. Pengguna dapat melihat perbandingan data sensor dalam waktu nyata dan mendapatkan data historis dengan ekspor ke Excel.

## **4. Appendix**

### **4.1 Kode Program untuk GraphController**

php

SalinEdit

<?php

namespace App\Http\Controllers;

use App\Exports\TransaksiSensorExport;

use Maatwebsite\Excel\Facades\Excel;

use App\Models\TransaksiSensor;

class GraphController extends Controller

{

public function index()

{

$transaksiSensors = TransaksiSensor::latest()->take(10)->get();

$labels = $transaksiSensors->pluck('nama\_sensor');

$dataNilai1 = $transaksiSensors->pluck('nilai1');

$dataNilai2 = $transaksiSensors->pluck('nilai2');

return view('graph', compact('labels', 'dataNilai1', 'dataNilai2'));

}

public function exportToExcel()

{

return Excel::download(new TransaksiSensorExport, 'transaksi\_sensor.xlsx');

}

}

### **4.2 Kode Program untuk TransaksiSensorExport**

php

SalinEdit

<?php

namespace App\Exports;

use App\Models\TransaksiSensor;

use Maatwebsite\Excel\Concerns\FromCollection;

class TransaksiSensorExport implements FromCollection

{

public function collection()

{

return TransaksiSensor::all();

}

}

### **4.3 Kode Program untuk Tampilan (Graph Blade)**

html

SalinEdit

<!DOCTYPE html>

<html lang="id">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Dashboard Monitoring Sensor | Sistem IoT</title>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">

<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&display=swap" rel="stylesheet">

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css">

<style>/\* CSS styles \*/</style>

</head>

<body>

<div class="dashboard-container">

<div class="header animate\_\_animated animate\_\_fadeIn">

<h1 class="header-title">

<i class="fas fa-chart-network"></i>

Dashboard Monitoring Sensor

</h1>

<div class="time-selector">

<button class="time-btn active">24 Jam</button>

<button class="time-btn">7 Hari</button>

<button class="time-btn">30 Hari</button>

<button class="time-btn">Custom</button>

</div>

</div>

<div class="card animate\_\_animated animate\_\_fadeIn animate\_\_delay-1s">

<div class="card-header">

<h2 class="card-title">

<i class="fas fa-wave-square"></i>

Grafik Perbandingan Sensor

</h2>

<div class="card-actions">

<button class="btn btn-outline" onclick="window.location.href='{{ route('graph.export') }}'">

<i class="fas fa-download"></i> Export

</button>

</div>

</div>

<div class="chart-container">

<canvas id="sensorChart"></canvas>

</div>

<div class="data-summary">

<!-- Data summary cards -->

</div>

</div>

</div>

<script>

const labels = @json($labels);

const dataNilai1 = @json($dataNilai1);

const dataNilai2 = @json($dataNilai2);

const ctx = document.getElementById('sensorChart').getContext('2d');

new Chart(ctx, {

type: 'line',

data: {

labels: labels,

datasets: [

{

label: 'Sensor 1',

data: dataNilai1,

borderColor: '#4361ee',

backgroundColor: 'rgba(67, 97, 238, 0.1)',

borderWidth: 2,

fill: true

},

{

label: 'Sensor 2',

data: dataNilai2,

borderColor: '#4cc9f0',

backgroundColor: 'rgba(76, 201, 240, 0.1)',

borderWidth: 2,

fill: true

}

]

}

});

</script>

</body>

</html>