Pengembangan Dashboard Web Terintegrasi untuk Kontrol dan Monitoring OEE pada CIM Miniature Smart Factory

Tugas Akhir

Oleh:

Andhika Pratama (4212201096)

Program Studi Teknik Mekatronika
Jurusan Teknik Elektro
Politeknik Negeri Batam
2026

Pernyataan Keaslian Tugas Akhir

Saya yang bertandatangan dibawah ini menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya yang berjudul : "Pengembangan Dashboard Web Terintegrasi untuk Kontrol dan Monitoring OEE pada CIM Miniature Smart Factory" adalah **hasil karya sendiri**, **diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan**, dan **bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.** Semua referensi yang dikutip atau dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan saya ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Batam, 09 Januari 2026

Andhika Pratama
NIM: 4212201096

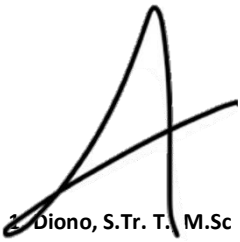
Lembar Pengesahan

Tugas Akhir disusun untuk memenuhi salah satu syarat memperoleh gelar Sarjana Terapan Teknik (S.Tr.T)
di
Politeknik Negeri Batam

Oleh:
Andhika Pratama (4212201096)

Tanggal Sidang: 09 Januari 2026

Disetujui oleh :



2. Diono, S.Tr. T., M.Sc
NIK: 120243

1. Fitriyanti Nakul, S.Pd., M.Si
NIK: 118197

2. Feralia Fitri, S.T., M.T
NIK: 125364

Pengembangan Dashboard Web Terintegrasi untuk Kontrol dan Monitoring OEE pada CIM Miniature Smart Factory

Abstrak

Pemanfaatan media ajar dalam bentuk *trainer kit* sangat krusial dalam pendidikan teknik untuk menjembatani teori dan aplikasi praktis sistem otomasi industri, khususnya dalam mendukung pemahaman konsep CIM (Computer Integrated Manufacturing) di lingkungan pendidikan. Namun, media ajar saat ini umumnya masih berfokus pada pengoperasian perangkat keras secara parsial, tanpa sistem pengawasan dan visualisasi data performa yang representatif terhadap standar industri. Kondisi ini menyulitkan pemahaman konsep pengawasan produksi yang komprehensif pada ekosistem pabrik. Untuk mengatasinya, Penelitian ini mengembangkan *dashboard* web terintegrasi untuk kontrol dan monitoring *Overall Equipment Effectiveness* (OEE) yang diimplementasikan pada miniatur *Smart Factory*. Metode pengerjaan dilakukan melalui pengembangan perangkat lunak berbasis JavaScript dengan protokol WebSocket untuk komunikasi data secara *real-time*. Sistem ini dilengkapi dengan fitur-fitur kontrol operasional seperti start, stop, dan reset, serta fitur pendukung berupa pengecekan status koneksi, histori proses, dan sinkronisasi waktu. Seluruh fitur tersebut diimplementasikan pada miniatur sistem konveyor, ARM robot, dan SCARA robot untuk simulasi penyortiran material logam dan non-logam secara otomatis. Melalui *dashboard* web, dilakukan kalkulasi parameter OEE yang mencakup *availability rate*, *performance rate*, dan *quality rate* sebagai indikator efektivitas produksi berdasarkan data sensor. Hasil penelitian menunjukkan bahwa integrasi fitur kontrol operasional, monitoring status koneksi, dan tabel histori *downtime* berfungsi secara akurat sesuai skenario industri. Keberadaan sistem ini dapat memberikan kontribusi lebih sebagai modul praktikum yang efektif dalam mensimulasikan pengawasan produksi skala industri melalui media miniatur yang terintegrasi.

Kata kunci: CIM, Dashboard Web, Kontrol dan Monitoring, Media Ajar, OEE, Smart Factory

Development of Integrated Web Dashboard for OEE Control and Monitoring at CIM Miniature Smart Factory

Abstract

The use of teaching media in the form of trainer kits is crucial in engineering education to bridge the gap between theory and practical application of industrial automation systems especially in supporting the understanding of the concept of Computer Integrated Manufacturing (CIM) in an educational environment. However, current teaching media generally still focus on partial hardware operation, without a monitoring system and performance data visualization that is representative of industry standards. This condition makes it difficult to understand the comprehensive concept of production monitoring in a factory ecosystem. To overcome this, this study developed an integrated web dashboard for Overall Equipment Effectiveness (OEE) control and monitoring, which was implemented in a miniature Smart Factory. The method was implemented through the development of JavaScript-based software with the WebSocket protocol for real-time data communication. This system is equipped with operational control features such as start, stop, and reset, as well as supporting features such as connection status checking, process history, and time synchronization. All of these features are implemented in a miniature conveyor system, ARM robot, and SCARA robot for the simulation of automatic sorting of metal and non-metal materials. Through the web dashboard, OEE parameters are calculated, including availability rate, performance rate, and quality rate as indicators of production effectiveness based on sensor data. The results of the study show that the integration of operational control features, connection status monitoring, and downtime history tables function accurately according to industrial scenarios. The existence of this system can contribute more as an effective practicum module in simulating industrial-scale production monitoring through integrated miniature media.

Keywords: CIM, Web Dashboard, Control and Monitoring, Teaching Media, OEE, Smart Factory

Kata Pengantar

Puji syukur saya ucapkan ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga penelitian mengenai perancangan dan realisasi CIM Miniature Smart Factory ini dapat diselesaikan. Laporan ini disusun sebagai bentuk pertanggungjawaban ilmiah atas pengembangan Dashboard Kontrol dan Monitoring OEE berbasis web yang dirancang untuk mengintegrasikan sistem kendali serta pengawasan performa produksi secara real-time.

Penelitian ini dikembangkan dengan tujuan untuk menghasilkan platform pemantauan yang mampu menyajikan data efektivitas produksi melalui indikator Overall Equipment Effectiveness (OEE). Dengan integrasi kontrol sistem dan pemanfaatan data dari sensor proximity serta induktif, dashboard ini diharapkan dapat memberikan kemudahan dalam pengawasan operasional, analisis downtime, dan pengambilan keputusan yang cepat dalam ekosistem smart factory. Penulis berharap sistem monitoring ini dapat menjadi referensi yang bermanfaat bagi pengembangan trainer kit otomasi industri serta membantu mahasiswa dalam memahami manajemen data produksi secara digital.

Penulis menyampaikan apresiasi dan terima kasih sebesar-besarnya kepada Ibu Fitriyanti Nakul atas bimbingan, arahan, serta dedikasi yang telah diberikan selama proses penyelesaian Tugas Akhir ini. Ucapan terima kasih juga disampaikan kepada seluruh pihak yang telah berkontribusi dalam proses pengembangan perangkat lunak hingga penyusunan laporan ini selesai. Kami berharap Tugas Akhir ini dapat memberikan manfaat yang luas serta menjadi referensi yang dapat diandalkan dalam pemahaman sistem monitoring pada lingkungan smart factory.

Batam, 06 Januari 2026

Andhika Pratama

DESKRIPSI UMUM

Pengembangan Dashboard Web Terintegrasi untuk Kontrol dan Monitoring OEE pada CIM Miniature Smart Factory merupakan program aplikasi berbasis web yang dirancang sebagai media pendukung modul ajar untuk mensimulasikan kontrol dan pemantauan sistem pabrik pintar secara *real-time*. Pengembangan dashboard ini bertujuan untuk memberikan pemahaman praktis mengenai peningkatan efisiensi dan fleksibilitas dalam proses manufaktur melalui representasi miniatur. Fokus utama program terletak pada pemantauan parameter OEE (*Overall Equipment Effectiveness*) yang mencakup *availability rate* (AR), *performance rate* (PR), dan *quality rate* (QR) sebagai indikator utama efektivitas produksi yang dipelajari mahasiswa. Fitur utama yang diintegrasikan meliputi kendali operasional melalui fungsi *start*, *stop*, dan *reset*, serta pengolahan data sensor *proximity* dan induktif untuk klasifikasi objek material logam (metal) dan non-logam. Selain itu, sebagai perangkat instruksional, dashboard ini dilengkapi dengan fitur monitoring status koneksi IP, penghitungan jumlah produk, serta penyajian histori *downtime* dalam bentuk tabel data untuk kebutuhan analisis performa mesin. Melalui visualisasi data yang interaktif, program ini berfungsi sebagai instrumen pembelajaran yang mendukung pengawasan akurat serta melatih kemampuan pengambilan keputusan dalam manajemen produksi pada ekosistem *smart factory*.

Kode Program Dashboard Kontrol dan Monitoring OEE pada CIM Miniature Smart Factory

Dashboard dari web kontrol dan monitoring sistem akan menampilkan manajemen informasi dari basis data hasil pembacaan data sensor yang diolah dari perangkat modul ESP32. Beberapa bagian kode program utama pada sistem kontrol dan monitoring OEE pada CIM Miniature Smart Factory, terdiri dari: (a) program yang ditanamkan pada mikrokontroler ESP32 untuk komunikasi I2C dalam menerima data sensor dari sistem CIM Miniature Smart Factory, dan sebagai server dalam komunikasi Websocket dengan web; (b) program HTML untuk dashboard web monitoring data OEE; (c) program javascript dashboard web monitoring data OEE.

a) Program ESP32 sebagai Master I2C untuk menerima data sensor dari sistem CIM Miniature Smart Factory dan sebagai Server dalam komunikasi menggunakan protokol WebSocket dengan web.

```
#include <Wire.h>
#include <WiFi.h>
#include <WebSocketsServer.h>
#include <Arduino.h>

//inilialisasi alamat slave i2C
#define I2C_SLAVE_ADDRESS_1 0x08
// #define I2C_SLAVE_ADDRESS_2 0x09
// #define I2C_SLAVE_ADDRESS_3 0x10

//set variabel access point wifi
const char* ssid = "ESP32_AP";
const char* password = "12345678";

WebSocketsServer webSocket(81); //inilialisasi websocket
dan port 81

//inilialisasi variabel
int    status_progress,    set_conveyor,    set_armrobot,
set_scara,     object,      set_object,      count_metal,
count_nonmetal;
unsigned long previous_millis, previous_millis1;
String send_todo_slave;


//fungsi penanganan websocket dan penerimaan payload dari
web ke esp32
void webSocketEvent(uint8_t num, WStype_t type, uint8_t
*payload, size_t length) {
    if (type == WStype_TEXT) {
        String message = (char*)payload;

        if(message.indexOf('?') != -1) {
```

```
                findNumber(message.indexOf('?'),        message,
        set_object);
                status_progress = 1;
                set_conveyor = 1;
              }
              else if(message == "stop") {
                status_progress = 0;
                set_conveyor = 0;
              }
              else if(message == "pause") {
                status_progress = 2;
                set_conveyor = 0;
              }
              else if(message == "reset") {
                status_progress = 3;
                wireWrite(I2C_SLAVE_ADDRESS_1,
        send_todo_slave);
                //                wireWrite(I2C_SLAVE_ADDRESS_2,
        send_todo_slave);
                //                wireWrite(I2C_SLAVE_ADDRESS_3,
        send_todo_slave);
                ESP.restart();
              }

              send_todo_slave = "!" + String(status_progress) +
                                "@" + String(set_conveyor) +
                                "#" + String(set_armrobot) +
                                "$" + String(set_scara) +
                                "%" + String(object) +
                                "&" + String(set_object) +
                                "m" + String(count_metal) +
                                "n" + String(count_nonmetal);

              wireWrite(I2C_SLAVE_ADDRESS_1, send_todo_slave);

            }
        }

        void setup() {
          Serial.begin(115200); //inilialisasi serial 115200
          Wire.begin(); //inilalisasi I2C
          Wire.setClock(100000);

          //inilalisasi access point wifi
          WiFi.softAP(ssid, password);
          Serial.println("Access Point Berjalan!");
          Serial.print("IP Address: ");
          Serial.println(WiFi.softAPIP());

          //inilialisasi websocket
          webSocket.begin();
          webSocket.onEvent(webSocketEvent);

          send_todo_slave = "!" + String(status_progress) +
                            "@" + String(set_conveyor) +
                            "#" + String(set_armrobot) +
                            "$" + String(set_scara) +
```

```
                          "%" + String(object) +
                          "&" + String(set_object) +
                          "m" + String(count_metal) +
                          "n" + String(count_nonmetal);

  Serial.println("ESP32 Master siap.");
}

void loop() {
  unsigned long current_millis = millis();

  //debugging lewat serial
  if(Serial.available() > 0) {
    String get_todo = Serial.readStringUntil('\n');

    if(get_todo == "start") {
      status_progress = 1;
      set_conveyor = 1;
      Serial.println("Program akan dijalankan");
    }
    else if(get_todo == "stop") {
      status_progress = 0;
      set_conveyor = 0;
      Serial.println("Program akan dihentikan");
    }
    else if(get_todo == "pause") {
      status_progress = 2;
      set_conveyor = 0;
      Serial.println("Program akan dijeda");
    }
    else if(get_todo == "up_metal") {
      count_metal += 1;
    }
    else if(get_todo == "up_nonmetal") {
      count_nonmetal += 1;
    }

    send_todo_slave = "!" + String(status_progress) +
                      "@" + String(set_conveyor) +
                      "#" + String(set_armrobot) +
                      "$" + String(set_scara) +
                      "%" + String(object) +
                      "&" + String(set_object) +
                      "m" + String(count_metal) +
                      "n" + String(count_nonmetal);

    wireWrite(I2C_SLAVE_ADDRESS_1, send_todo_slave);

  }

  //program mengirim text/data dari esp32 ke web, dan
menerima data dari esp32 slave
  if(current_millis - previous_millis >= 200) {
    previous_millis = current_millis;

    webSocket.sendTXT(0, "#" + String(status_progress) +
                         "m" + String(count_metal) +
```

```
                                    "n" + String(count_nonmetal));

    wireGet(I2C_SLAVE_ADDRESS_1,          status_progress,
set_conveyor, set_armrobot, set_scara, object, set_object,
count_metal, count_nonmetal);
    //    wireGet(I2C_SLAVE_ADDRESS_2,    status_progress,
set_conveyor, set_armrobot, set_scara, object, set_object,
count_metal, count_nonmetal);
    //    wireGet(I2C_SLAVE_ADDRESS_3,    status_progress,
set_conveyor, set_armrobot, set_scara, object, set_object,
count_metal, count_nonmetal);
    //     Serial.println("Status     Progress:     "     +
String(status_progress));

  }

  webSocket.loop(); //websocket berjalan secara loop

}

void wireWrite(uint8_t slave_address, String datasend) {
  Wire.beginTransmission(slave_address);
  Wire.write((uint8_t*)datasend.c_str(),
datasend.length());
  Serial.println("Master Mengirim: " + datasend);
  Wire.endTransmission();
}

/*fungsi untuk mengakses alamat slave dan menerima data
menggunakan
  fungsi parsing data */
void wireGet(uint8_t slave_address, int &value_data1, int
&value_data2,
             int  &value_data3,  int  &value_data4,  int
&value_data5, int &value_data6,
             int &value_data7, int &value_data8) {

  Wire.requestFrom(slave_address, 18);
  String response = "";
  while (Wire.available()) {
    char c = Wire.read();
    if (c == '\0') break;
    response += c;
  }

  findNumber(response.indexOf('!'),              response,
value_data1);
  findNumber(response.indexOf('@'),              response,
value_data2);
  findNumber(response.indexOf('#'),              response,
value_data3);
  findNumber(response.indexOf('$'),              response,
value_data4);
  findNumber(response.indexOf('%'),              response,
value_data5);
  findNumber(response.indexOf('&'),              response,
value_data6);
```

```
  findNumber(response.indexOf('m'),                response,
value_data7);
  findNumber(response.indexOf('n'),                response,
value_data8);

  if (response.length() > 0) {
    Serial.println("Balasan dari Slave : " + response);
  }

}

//fungsi mencari nilai dengan metode parsing data
void findNumber(int idx, String raw_data, int &store_data)
{
  store_data = 0;
  if(idx != -1) {
    int i = idx + 1;

    while      (i      <      raw_data.length()      &&
isDigit(raw_data.charAt(i))) {
      store_data = store_data * 10 + (raw_data.charAt(i) -
'0');
      i++;
    }

  }
}
```

Gambar 1. Program ESP32

Pada uraian program (Gambar 1), mengimplementasikan sistem berbasis ESP32 yang berfungsi sebagai master dalam komunikasi I2C dengan beberapa slave serta sebagai server WebSocket untuk berkomunikasi dengan antarmuka web. ESP32 beroperasi dalam mode Access Point WiFi dan menerima perintah dari klien melalui WebSocket, seperti "start," "stop," "pause," dan "reset," untuk mengontrol conveyor, robot lengan (arm robot), dan robot SCARA. Data status operasi, jumlah objek metal dan nonmetal yang terdeteksi, serta perintah lainnya dikirim ke perangkat slave melalui I2C. Selain itu, ESP32 menerima data dari slave dan meneruskannya ke klien WebSocket. Program ini juga mendukung debugging melalui komunikasi serial dan melakukan pengolahan data dengan metode parsing untuk membaca serta mengirimkan status operasional.

b) Program HTML Dashboard Web Monitoring Data OEE yang terdiri dari 2 halaman : (1) Halaman Utama dan (2) Halaman Histori.

(1) Halaman Utama

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
     <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
    <title>PBLMK232-12 Smart Factory</title>
    <link rel="stylesheet" href="../css/menu_webpage.css" />
  </head>
  <body>
    <!-- Judul Halaman -->
    <header>
      <h1>SMART FACTORY CONTROL SYSTEM</h1>
    </header>

    <div class="main-container">
      <!-- Kontainer Kiri -->
      <div class="container">
        <div class="atas">
          <div class="samping-3">
            <div class="time-card">
              <div class="time-label">Time</div>
              <div id="waktu" class="time-value"></div>
            </div>

            <div class="date-card">
              <div class="date-label">Date</div>
              <div id="tanggal" class="date-value"></div>
            </div>
          </div>

          <div class="availability-card">
           <div class="availability-label">Availability (AR)</div>
            <div id="availability" class="availability-bar">
                 <div class="availability-progress" style="width:
0%"></div>
              <span class="availability-progress-text">0%</span>
            </div>
          </div>

          <div class="performance-card">
            <div class="performance-label">Performance (PR)</div>
            <div id="performance" class="performance-bar">
                  <div class="performance-progress" style="width:
0%"></div>
              <span class="performance-progress-text">0%</span>
            </div>
          </div>
```

```
          <div class="quality-card">
            <div class="quality-label">Quality (QR)</div>
            <div id="quality" class="quality-bar">
             <div class="quality-progress" style="width: 0%"></div>
              <span class="quality-progress-text">0%</span>
            </div>
          </div>

          <div class="set-target-card">
            <div class="set-target-label">Set Target</div>
            <div id="set-target" class="set-target-bar">
                  <div  class="set-target-progress"  style="width:
0%"></div>
              <span class="set-target-progress-text">0</span>
            </div>
          </div>

          <div class="ip-card">
            <div class="ip-label">IP CONNECTION</div>
            <input
              type="text"
              id="ipInput"
              placeholder="Enter IP address"
              value="192.168.4.1"
            />
            <button
              class="button-connect"
              id="actionconnectbutton"
              onclick="toggleConnection()"
            >
              CONNECT
            </button>
            <div id="ip-text"></div>
          </div>
        </div>
      </div>

      <!-- Kontainer Tengah -->
      <div class="container">
        <div class="atas-1">
          <div class="total-card">
            <div class="samping">
              <div class="total-chart">
                              <canvas   id="chart"   width="150"
height="150"></canvas>
                <div class="chart-text">
                  <p class="total">TOTAL</p>
                  <p class="value" id="total-count">0</p>
                </div>
              </div>

              <div class="legend">
                <div>
                  <span class="legend-box metal"></span
                  ><span class="metal-label">METAL : </span>
                  <span id="metal-count">0</span>
                </div>
```

```
                <div>
                  <span class="legend-box non-metal"></span
                  ><span class="nonmetal-label">NONMETAL: </span>
                  <span id="nonmetal-count">0</span>
                </div>
              </div>
            </div>
          </div>

          <div class="process-card">
            <div class="samping">
              <div class="process-chart">
                          <canvas  id="chart-process"  width="150"
height="150"></canvas>
                <div class="chart-text">
                  <p class="process">PROCESS</p>
                  <p class="value" id="percen-count">0</p>
                </div>
              </div>

              <div class="legend">
                <div>
                  <span class="legend-box not-good"></span
                  ><span class="notgood-label">NOT GOOD: </span>
                  <span id="notgood-count">0</span>
                </div>
                <div>
                  <span class="legend-box good"></span
                  ><span class="good-label">GOOD: </span>
                  <span id="good-count">0</span>
                </div>
              </div>
            </div>
          </div>

          <div class="samping-2">
            <div class="status-card">
              <div class="status-label">Status</div>
              <div class="status-indicator">
                <div id="statusCircle" class="off">OFF</div>
              </div>
            </div>

            <div class="request-card">
              <div class="request-label">Request</div>
              <div class="request-controls">
                <select class="request-select" id="object-select">
                  <option value="object1">Metal</option>
                  <option value="object2">Nonmetal</option>
                </select>
                <div class="request-buttons">
                  <button
                    id="tombol-start"
                    onclick="sendStart(); pindahKePause()"
                    class="btn-start"
                  >
                    Start
```

```
                </button>
                <button
                  id="tombol-pause"
                  onclick="sendPause(); pindahKeStart()"
                  class="btn-pause"
                >
                  Pause
                </button>
                <button
                  id="tombol-stop"
                  onclick="sendStop(); stopButton()"
                  class="btn-stop"
                >
                  Stop
                </button>
                <button
                  id="tombol-reset"
                  class="btn-reset"
                  onclick="sendReset();"
                >
                  Reset
                </button>
              </div>
            </div>
          </div>
        </div>
      </div>

      <!-- Kontainer Kanan -->
      <div class="container">
        <div class="atas-1">
          <div class="oee-card">
            <svg class="chart-oee">
                <circle class="oee-background" cx="100" cy="100"
r="75"></circle>
              <circle
                class="oee-progress"
                cx="100"
                cy="100"
                r="75"
                stroke-dasharray="0 100"
              ></circle>
            </svg>
            <div class="oee-text">
              <p class="oee">OEE</p>
              <p id="persentase-oee">0%</p>
            </div>
          </div>

          <div class="loadtime-card">
            <div class="loadtime-label">Loading Time</div>
            <div id="loadtime-value">00 : 00 : 00</div>
          </div>

          <div class="downtime-card">
            <div class="downtime-label">Total Downtime</div>
```

```
            <div id="downtime-value">00 : 00 : 00</div>
          </div>

          <div class="operatetime-card">
            <div class="operatetime-label">Operation Time</div>
            <div id="operatetime-value">00 : 00 : 00</div>
          </div>

          <div class="history-card">
               <button onclick="redirectToHistory()" class="btn-
history">
              History
            </button>
            <div class="samping">
              <button
                id="tombol-testbar"
                onclick="random_gcount(); random_metalnonmetal()"
              >
                Test OEE
              </button>
              <button id="tombol-input">Input Object</button>

              <div id="popup" class="popup">
                <div class="popup-content">
                  <input
                    type="text"
                    id="numberInput"
                    placeholder="Masukkan jumlah object"
                  />
                  <button id="submitBtn">Submit</button>
                  <button id="cancelBtn">Cancel</button>
                  <p id="errorMessage" class="error-message"></p>
                </div>
              </div>
            </div>
            <div id="set-target-value" class="set-target-value">
              Set Target: 0
            </div>
          </div>
        </div>
      </div>
    </div>

    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script src="../js/timeDate.js"></script>
    <script src="../js/navigate.js"></script>
    <script src="../js/totalObject.js"></script>
    <script src="../js/processObject.js"></script>
    <script src="../js/status.js"></script>
    <script src="../js/timer.js"></script>
    <script src="../js/progressOEE.js"></script>
    <script src="../js/esp32.js"></script>
    <script src="../js/connectionIP.js"></script>
    <script src="../js/inputObject.js"></script>

  </body>
</html>
```

Gambar 3. Program Dashboard Kontrol dan Monitoring OEE (Halaman Utama)

Halaman utama Smart Factory Control System (gambar 2) menampilkan informasi penting untuk pemantauan dan kontrol produksi. Tersedia judul web, indikator waktu dan tanggal, serta indikator kinerja OEE untuk analisis efisiensi. Pengguna dapat melihat hasil produksi, jumlah dan kualitas objek, serta status mesin secara real- time. Terdapat menu konfigurasi IP, kontrol sistem, dan persentase OEE untuk optimasi kinerja. Selain itu, indikator waktu operasi serta menu histori dan input target objek membantu pencatatan dan perencanaan produksi.

(2) Halaman Histori

```
          </div>

          <div class="operatetime-card">
            <div class="operatetime-label">Operation Time</div>
            <div id="operatetime-value">00 : 00 : 00</div>
          </div>
        </div>
      </div>

      <!-- Kontainer Kanan -->
      <div class="container-2">
        <div class="table-container">
          <table id="dataTable" border="1">
            <thead>
              <tr>
                <th>No</th>
                <th>Date</th>
                <th>Time</th>
                <th>Down Time (minutes)</th>
                <th>Operation Time (minutes)</th>
                <th>Loading Time (minutes)</th>
                <th>Total Count</th>
                <th>Total Process</th>
              </tr>
            </thead>
            <tbody>
              <!-- Rows will be dynamically added here -->
            </tbody>
          </table>
        </div>
        <div class="buttons">
         <button onclick="redirectToControl()" id="controlButton">
            CONTROL
          </button>
          <button id="clearButton">CLEAR</button>
          <button id="saveButton">SAVE</button>
          <button id="addButton">ADD</button>
        </div>
      </div>
    </div>
    <script src="../js/table.js"></script>
    <script src="../js/navigate.js"></script>
    <script src="../js/timeDate.js"></script>
    <script src="../js/timer2.js"></script>
  </body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
     <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
    <title>PBLMK232-12 Smart Factory</title>
    <link rel="stylesheet" href="../css/history_webpage.css" />
  </head>
  <body>
    <!-- Judul Halaman -->
    <header>
      <h1>SMART FACTORY CONTROL SYSTEM</h1>
    </header>

    <div class="main-container">
      <!-- Kontainer Kiri -->
      <div class="container">
        <div class="atas">
          <div class="id-card">
            <div class="id-label">ID Machine</div>
            <div class="id-value">1</div>
          </div>

          <div class="machine-name-card">
            <div class="machine-name-label">Machine Name</div>
            <div class="machine-name-value">PBLMK232-12</div>
          </div>

          <div class="machine-no-card">
            <div class="machine-no-label">Machine No.</div>
            <div class="machine-no-value">123ABC</div>
          </div>

          <div class="samping-3">
            <div class="time-card">
              <div class="time-label">Time</div>
              <div id="waktu" class="time-value"></div>
            </div>

            <div class="date-card">
              <div class="date-label">Date</div>
              <div id="tanggal" class="date-value"></div>
            </div>
          </div>

          <div class="loadtime-card">
            <div class="loadtime-label">Loading Time</div>
            <div id="loadtime-value">00 : 00 : 00</div>
          </div>

          <div class="downtime-card">
            <div class="downtime-label">Total Downtime</div>
            <div id="downtime-value">00 : 00 : 00</div>
```

Gambar 3 Program Dashboard Kontrol dan Monitoring OEE (Halaman Histori)

Halaman histori (gambar 3) terdiri dari judul web dan identitas mesin, termasuk nama, ID, dan nomor mesin. Menampilkan indikator waktu dan tanggal real-time serta waktu operasi mesin untuk pemantauan efisiensi. Tersedia menu tabel data historis untuk pencatatan riwayat produksi dan menu kontrol data untuk menyimpan atau menghapus data sesuai kebutuhan.

c) Program Javascript Dashboard Web Monitoring Data OEE

```
 /*program tersebut dikumpulkan jadi satu, tetapi bisa dipisahkan
sesuai
  nama file yang sudah di comment jika ingin dipisahkan */

 ///// connectionIP.js for menu_webpage.html /////

 //inilialisasi variabel
let isConnected = false;
const show_websocket_status = document.getElementById("ip-text");
const                          connectbutton                          =
document.getElementById("actionconnectbutton");

//membuat fungsi menginput ip address dan melakukan koneksi via
websocket
function toggleConnection() {
  const ipInput = document.getElementById("ipInput");

  if (!isConnected) {
    const ipAddress = ipInput.value;

    if (!ipAddress) {
      alert("Please enter a valid IP address.");
      return;
    }

    initializeWebSocket(ipAddress);
    console.log(`Connecting to ${ipAddress}...`);
    isConnected = true;

  } else {

    closingwebsocket();
    console.log("Disconnecting...");
    isConnected = false;

  }
}

//fungsi memberi label status websocket terkoneksi atau belum
function webscoketstatus() {
  if(isConnected) {
    show_websocket_status.textContent = "WEBSOCKET IS CONNECTED";
  }
  else {
      show_websocket_status.textContent   =   "WEBSOCKET   IS   NOT
CONNECTED";
  }
}
```

```
///// esp32.js for menu_webpage.html /////

//inilialisasi variabel
let socket;
let valuestatus;
let send_esp32;
const selectobject = document.getElementById("object-select");
let socket_value;

// Inisialisasi WebSocket
function initializeWebSocket(IP_ADDRESS) {
  // const ESP32_IP = '192.168.4.1';
  // socket = new WebSocket(`ws://${ESP32_IP}:81/`);
  socket = new WebSocket(`ws://${IP_ADDRESS}:81/`);
  socket_value = socket.readyState;

  //fungsi ketika websocket sudah terbuka
  socket.onopen = function() {
    console.log('WebSocket connected!');

    connectbutton.textContent = "DISCONNECT";
    connectbutton.classList.remove("button-connect");
    connectbutton.classList.add("button-disconnect");
    webscoketstatus();
  };

  //fungsi menangani ketika websocket menerima pesan dari ESP32
  socket.onmessage = function(event) {
    console.log('Message from ESP32:', event.data);

    if(String(event.data).length > 0) {
      let temp_str = String(event.data).match(/\d+/g);
      valuestatus = temp_str[0];
      metalCount = parseInt(temp_str[1]);
      nonmetalCount = parseInt(temp_str[2]);
      // console.log('Message from after Parse status progress: ',
valuestatus);
       // console.log('Message from after Parse count metal: ',
metalCount);
      // console.log('Message from after Parse count non metal: ',
nonmetalCount);
      updateChart();
    }

  };

  //fungsi ketika websocket sudah ditutup
  socket.onclose = function() {
    console.log('WebSocket connection closed');
    connectbutton.textContent = "CONNECT";
    connectbutton.classList.remove("button-disconnect");
    connectbutton.classList.add("button-connect");
    webscoketstatus();
  };

  //fungsi ketika websocket mengalami error
  socket.onerror = function(error) {
```

```
    console.error('WebSocket error:', error);
  };
}

//fungsi menutup websocket
function closingwebsocket() {
  if(socket.readyState == WebSocket.OPEN) {
    socket.close();
  }
}

//fungsi mengubah nilai string select request ke nilai bulatan
function request_todo() {
  if(selectobject.value === 'object1') {
    return 1;
  }
  else if(selectobject.value === 'object2') {
    return 0;
  }
}

// Kirim data 'start' ke ESP32
function sendStart() {
  if (socket.readyState === WebSocket.OPEN) {
    send_esp32 = 'start?' + request_todo().toString();
    socket.send(send_esp32);
    console.log('Sending start command to ESP32...');
    console.log(send_esp32);
  }
}

// Kirim data pause ke ESP32
function sendPause() {
  if (socket.readyState === WebSocket.OPEN) {
    send_esp32 = 'pause';
    socket.send(send_esp32);
    console.log('Sending pause command to ESP32...');
  }
}

// Kirim data stop ke ESP32
function sendStop() {
  if (socket.readyState === WebSocket.OPEN) {
    send_esp32 = 'stop';
    socket.send(send_esp32);
    console.log('Sending stop command to ESP32...');
  }
}

// Kirim data reset ke ESP32
function sendReset() {
  if (socket.readyState === WebSocket.OPEN) {
    send_esp32 = 'reset';
    socket.send(send_esp32);
    console.log('Sending reset command to ESP32...');
  }
}
```

```
//pemanggilan websocketstatus ketika memuat halaman web
window.onload = function() {
  webscoketstatus();
};

///// processObject.js for menu_webpage.html /////

//inilialisasi variabel yang dikaitkan dengan element html
const resetProcess = document.getElementById("tombol-reset");
const chartCanvas2 = document.getElementById("chart-process");
const goodCountSpan = document.getElementById("good-count");
const notgoodCountSpan = document.getElementById("notgood-count");
const percenCountSpan = document.getElementById("percen-count");

//inilialisasi variabel
let goodCount = 0;
let notgoodCount = 0;

//membuat objek process circle chart
const chart2 = new Chart(chartCanvas2, {
  type: "doughnut",
  data: {
    datasets: [
      {
        data: [goodCount, notgoodCount],
        backgroundColor: ["green", "red"],
        borderColor: ["green", "red"],
        borderWidth: 1,
      },
    ],
  },
});

//fungsi memperbarui process chart
function updateChart_process() {
  chart2.data.datasets[0].data[0] = goodCount;
  chart2.data.datasets[0].data[1] = notgoodCount;
  chart2.update();

  goodCountSpan.textContent = goodCount;
  notgoodCountSpan.textContent = notgoodCount;
  percenCountSpan.textContent = goodCount + notgoodCount;

                      sessionStorage.setItem('totalcountprocess',
percenCountSpan.textContent);

  const totalObjects = goodCount + notgoodCount;
    const   progressBar   =   document.querySelector(".set-target-
progress");
    const   progressText   =   document.querySelector(".set-target-
progress-text");

  //barisan kode pada bar set target
  let percentage = (totalObjects / maxSetTarget) * 100;
  if (maxSetTarget === 0) {
```

```
    percentage = 0; // Jika maxSetTarget 0, progress bar direset
ke 0%
  }

  progressBar.style.width = percentage + "%";
  progressText.textContent = totalObjects;

}

//fungsi debugging process chart
function random_gcount() {
  goodCount += 2;
  notgoodCount += 1;
  updateChart_process();
}

//fungsi event buat reset process
resetProcess.addEventListener("click", function () {
  if (goodCount > 0) {
    goodCount -= goodCount;
    updateChart_process();
  }

  if (notgoodCount > 0) {
    notgoodCount -= notgoodCount;
    updateChart_process();
  }

  //barisan kode pada set target
   const setTargetValueDisplay = document.getElementById("set-
target-value");
  setTargetValueDisplay.textContent = "Set Target: 0";
  maxSetTarget = 0; // Reset maxSetTarget ke 0

});
```

///// inputObject.js for menu_webpage.html /////

```
let maxSetTarget = 0; // Variabel untuk menyimpan nilai maksimum
set target

document.getElementById("tombol-input").addEventListener("click",
function () {
  const numberInput = document.getElementById("numberInput");
  numberInput.value = ""; // Reset nilai input
  numberInput.placeholder = "Masukkan jumlah objek";
   document.getElementById("popup").style.display = "flex"; //
Tampilkan popup
});

document.getElementById("cancelBtn").addEventListener("click",
function () {
  document.getElementById("popup").style.display = "none";
  document.getElementById("errorMessage").textContent = "";
});
```

```
document.getElementById("submitBtn").addEventListener("click",
function () {
  const numberInput = document.getElementById("numberInput");
  const errorMessage = document.getElementById("errorMessage");
   const  setTargetValueDisplay  =  document.getElementById("set-
target-value"); // Elemen untuk menampilkan nilai set-target

  let value = parseInt(numberInput.value);

  if (isNaN(value)) {
      errorMessage.textContent  =  "Silakan  masukkan  angka  yang
valid!";
    return;
  }

  if (value < 0) {
    errorMessage.textContent = "Angka tidak boleh negatif!";
    return;
  }

  errorMessage.textContent = "";

  // Set nilai maksimum progress bar sesuai input
  maxSetTarget = value;

  // Tampilkan nilai set-target di elemen yang sudah disediakan
  setTargetValueDisplay.textContent = `Set Target: ${value}`;

  alert("Input valid: " + value);
   document.getElementById("popup").style.display  =  "none";  //
Tutup popup

  // Update progress bar setelah set target diubah
  updateChart_process();
});

///// progressOEE.js for menu_webpage.html /////

//inilialisasi variabel yang dikaitkan dengan element html
const tombolTestbar = document.getElementById("tombol-testbar");
const resetOEE = document.getElementById("tombol-reset");
const progressText = document.getElementById("persentase-oee");
const oeeProgress = document.querySelector(".oee-progress");

const                   progressBar_availability                    =
document.getElementById("availability");
const                    progressBar_performance                    =
document.getElementById("performance");
const progressBar_quality = document.getElementById("quality");

const                     progress_availability                     =
progressBar_availability.querySelector(
  ".availability-progress"
);
const                      progress_performance                     =
progressBar_performance.querySelector(
  ".performance-progress"
```

```
);
const                          progress_quality                          =
progressBar_quality.querySelector(".quality-progress");

const                     persentase_availability                        =
progressBar_availability.querySelector(
  ".availability-progress-text"
);
const                     persentase_performance                         =
progressBar_performance.querySelector(
  ".performance-progress-text"
);
const persentase_quality = progressBar_quality.querySelector(
  ".quality-progress-text"
);

//inilialisasi variabel
let progressValue_availability = 0;
let progressValue_performance = 0;
let progressValue_quality = 0;

//fungsi kalkulasi OEE
function calculateOEE() {
  oeeprogress = Math.floor(
    (progressValue_availability / 100) *
      (progressValue_performance / 100) *
      (progressValue_quality / 100) *
      100
  );
  updateProgress();
}

//fungsi perbarui progress OEE
function updateProgress() {
  const dashArray = (oeeprogress / 100) * 475; // 2 * π * r = 2 *
π * 90
  oeeProgress.style.strokeDasharray = `${dashArray} 475`;
  progressText.textContent = `${oeeprogress}%`;
}

//debugging progress OEE
tombolTestbar.addEventListener("click", function () {
  const getRandomValue = (currentValue) => {
    let change = (Math.floor(Math.random() * 9) - 4) * 5; // Nilai
acak kelipatan 5, antara -20 hingga +20
    let newValue = currentValue + change;
    return newValue < 0 ? 0 : newValue > 100 ? 100 : newValue; //
Pastikan nilai berada di antara 0 dan 100
  };

                    progressValue_availability                         =
getRandomValue(progressValue_availability);
                     progressValue_performance                         =
getRandomValue(progressValue_performance);
  progressValue_quality = getRandomValue(progressValue_quality);

  // Update lebar progress bar
```

```
                progress_availability.style.width                =
`${progressValue_availability}%`;
                 progress_performance.style.width                =
`${progressValue_performance}%`;
   progress_quality.style.width = `${progressValue_quality}%`;

  // Update teks persentase
               persentase_availability.textContent               =
`${progressValue_availability}%`;
                persentase_performance.textContent               =
`${progressValue_performance}%`;
  persentase_quality.textContent = `${progressValue_quality}%`;

  // Perhitungan warna gradasi merah-hijau
  const updateColor = (element, value) => {
    const r = Math.floor(255 * (1 - value / 100));
    const g = Math.floor((255 * value) / 100);
    element.style.backgroundColor = `rgb(${r}, ${g}, 0)`;
  };

  updateColor(progress_availability, progressValue_availability);
  updateColor(progress_performance, progressValue_performance);
  updateColor(progress_quality, progressValue_quality);

  calculateOEE();
});

//fungsi reset OEE
resetOEE.addEventListener("click", function () {
  progressValue_availability = 0;
  progressValue_performance = 0;
  progressValue_quality = 0;

                 progress_availability.style.width               =
`${progressValue_availability}%`;
                  progress_performance.style.width               =
`${progressValue_performance}%`;
   progress_quality.style.width = `${progressValue_quality}%`;

               persentase_availability.textContent               =
`${progressValue_availability}%`;
                persentase_performance.textContent               =
`${progressValue_performance}%`;
  persentase_quality.textContent = `${progressValue_quality}%`;

  const resetColor = (element) => {
     element.style.backgroundColor = `rgb(255, 0, 0)`; // Merah
sebagai nilai awal
  };

  resetColor(progress_availability);
  resetColor(progress_performance);
  resetColor(progress_quality);

  calculateOEE();
});
```

```
calculateOEE();

///// totalObject.js for menu_webpage.html /////

//inilialisasi variabel yang dikaitkan dengan element html
const resetTotal = document.getElementById("tombol-reset");
const chartCanvas = document.getElementById("chart");
const metalCountSpan = document.getElementById("metal-count");
const   nonmetalCountSpan   =   document.getElementById("nonmetal-
count");
const totalCountSpan = document.getElementById("total-count");

//inilialisasi variabel
let metalCount = 0;
let nonmetalCount = 0;

//membuat objek counting object chart
const chart = new Chart(chartCanvas, {
  type: "doughnut",
  data: {
    datasets: [
      {
        data: [metalCount, nonmetalCount],
        backgroundColor: ["#00cc99", "orange"],
        borderColor: ["#00cc99", "orange"],
        borderWidth: 1,
      },
    ],
  },
});

//fungsi pembaruan counting object chart
function updateChart() {
  chart.data.datasets[0].data[0] = metalCount;
  chart.data.datasets[0].data[1] = nonmetalCount;
  chart.update();

  metalCountSpan.textContent = metalCount;
  nonmetalCountSpan.textContent = nonmetalCount;
  totalCountSpan.textContent = metalCount + nonmetalCount;

   sessionStorage.setItem('totalcountobject',totalCountSpan.textCo
ntent);

}

//fungsi debugging counting object chart
function random_metalnonmetal() {
  metalCount += 1;
  nonmetalCount += 2;
  updateChart();
}

//fungsi reset nilai counting object
resetTotal.addEventListener("click", function () {
  if (metalCount > 0) {
    metalCount -= metalCount;
```

```
    updateChart();
  }

  if (nonmetalCount > 0) {
    nonmetalCount -= nonmetalCount;
    updateChart();
  }
});

///// status.js for menu_webpage.html /////

//inilialisasi variabel yang dikaitkan dengan element html
const tombolStartStatus = document.getElementById("tombol-start");
const tombolPauseStatus = document.getElementById("tombol-pause");
const tombolStopStatus = document.getElementById("tombol-stop");
const statusIndicator = document.getElementById("statusCircle");

function pindahKePause() {
  // Disable and hide Button start
  tombolStartStatus.disabled = true;
  tombolStartStatus.style.display = "none";
  // Show Button pause
  tombolPauseStatus.style.display = "inline-block";
}

function pindahKeStart() {
  tombolStartStatus.disabled = false;
  tombolStartStatus.style.display = "inline-block";
  // Hide Button Pause
  tombolPauseStatus.style.display = "none";
}

function stopButton() {
  // Show Button START and set it to ON
  tombolStartStatus.disabled = false;
  tombolStartStatus.style.display = "inline-block";
  // Hide Button PAUSE
  tombolPauseStatus.style.display = "none";
}

tombolStartStatus.addEventListener("click", function () {
  // console.log('Nilai di area tombol: ', valuestatus );
  if (statusIndicator.textContent === "OFF") {
    statusIndicator.textContent = "ON";
    statusIndicator.style.backgroundColor = "limegreen";
  } else if (statusIndicator.textContent === "PAUSE") {
    statusIndicator.textContent = "ON";
    statusIndicator.style.backgroundColor = "limegreen";
  }
});

tombolPauseStatus.addEventListener("click", function () {
  if (statusIndicator.textContent === "ON") {
    statusIndicator.textContent = "PAUSE";
    statusIndicator.style.backgroundColor = "dimgray";
  }
});
```

```
// Fungsi untuk memformat waktu
function formatTime(totalSeconds) {
  const hours = String(Math.floor(totalSeconds / 3600)).padStart(2,
"0");
   const  minutes  =  String(Math.floor((totalSeconds  %  3600)  /
60)).padStart(
    2,
    "0"
  );
  const seconds = String(totalSeconds % 60).padStart(2, "0");
  return `${hours} : ${minutes} : ${seconds}`;
}

// Fungsi untuk memperbarui downtime
function updateDowntime() {
  secondsdowntime++;
  downtime.textContent = `${formatTime(secondsdowntime)}`;
  updateLoadingTime();
}

// Fungsi untuk memperbarui operatetime
function updateOperatetime() {
  secondsoperatetime++;
  operatetime.textContent = `${formatTime(secondsoperatetime)}`;
  updateLoadingTime();
}

// Fungsi untuk memperbarui loadingtime
function updateLoadingTime() {
  const totalSeconds = secondsdowntime + secondsoperatetime;
  loadingtime.textContent = `${formatTime(totalSeconds)}`;
}

//     Event     listener     untuk     tombol     start
tombolStartTimer.addEventListener("click", function () {
  if (!isoperatetimeRunning) {
    // Start the operation timer
    isoperatetimeRunning = true;
    isdowntimeRunning = false;
    operatetimeInterval = setInterval(updateOperatetime, 1000);
    clearInterval(downtimeInterval);
    downtime.textContent = `${formatTime(secondsdowntime)}`;
  }
});

// Event listener untuk tombol pause
tombolPauseTimer.addEventListener("click", function () {
  if (!isdowntimeRunning) {
    // Start the downtime timer
    isoperatetimeRunning = false;
    isdowntimeRunning = true;
    downtimeInterval = setInterval(updateDowntime, 1000);
    clearInterval(operatetimeInterval);
    operatetime.textContent = `${formatTime(secondsoperatetime)}`;
  }
});
```

```
// Event listener untuk tombol stop
tombolStopTimer.addEventListener("click", function () {
  // Stop all timers
  clearInterval(downtimeInterval);
  clearInterval(operatetimeInterval);
  isoperatetimeRunning = false;
  isdowntimeRunning = false;
});

// Event listener untuk tombol reset
resetTimer.addEventListener("click", function () {
  // Reset semua timer ke nilai awal
  secondsdowntime = 0;
  secondsoperatetime = 0;

  // Perbarui tampilan waktu ke nilai awal
  downtime.textContent = `${formatTime(secondsdowntime)}`;
  operatetime.textContent = `${formatTime(secondsoperatetime)}`;
  loadingtime.textContent = `${formatTime(
    secondsdowntime + secondsoperatetime
  )}`;

  // Periksa kondisi tombol yang terakhir ditekan
  if (isoperatetimeRunning) {
    // Jika operasi timer aktif, tetap jalankan operatetime interval
    clearInterval(downtimeInterval); // Pastikan interval downtime
dihentikan
    clearInterval(operatetimeInterval); // Hentikan interval untuk
memulai ulang
    operatetimeInterval = setInterval(updateOperatetime, 1000);
  } else if (isdowntimeRunning) {
    // Jika downtime timer aktif, tetap jalankan downtime interval
      clearInterval(operatetimeInterval);  //  Pastikan  interval
operatetime dihentikan
     clearInterval(downtimeInterval); // Hentikan interval untuk
memulai ulang
    downtimeInterval = setInterval(updateDowntime, 1000);
  } else {
     // Jika tidak ada timer aktif (stop ditekan sebelumnya),
hentikan semua interval
    clearInterval(downtimeInterval);
    clearInterval(operatetimeInterval);
  }
});

// Simpan data ke localStorage
function saveTimerState() {
  const state = {
    secondsdowntime,
    secondsoperatetime,
    isdowntimeRunning,
    isoperatetimeRunning,
    lastUpdated: Date.now(),
  };
  localStorage.setItem("timerState", JSON.stringify(state));
}
```

```
// Ambil data dari localStorage
function loadTimerState() {
  const state = JSON.parse(localStorage.getItem("timerState"));
  if (state) {
    const elapsedTime = Math.floor((Date.now() - state.lastUpdated)
/ 1000);
    secondsdowntime =
          state.secondsdowntime   +   (state.isdowntimeRunning   ?
elapsedTime : 0);
    secondsoperatetime =
        state.secondsoperatetime + (state.isoperatetimeRunning ?
elapsedTime : 0);
    isdowntimeRunning = state.isdowntimeRunning;
    isoperatetimeRunning = state.isoperatetimeRunning;

    downtime.textContent = formatTime(secondsdowntime);
    operatetime.textContent = formatTime(secondsoperatetime);
       loadingtime.textContent   =   formatTime(secondsdowntime   +
secondsoperatetime);

    if (isoperatetimeRunning) {
      operatetimeInterval = setInterval(updateOperatetime, 1000);
    }
    if (isdowntimeRunning) {
      downtimeInterval = setInterval(updateDowntime, 1000);
    }
  }
}

// Simpan status timer setiap detik
setInterval(() => {
  saveTimerState();
  console.log(
    "Timer state saved:",
    JSON.parse(localStorage.getItem("timerState"))
  );
}, 1000);

// Panggil loadTimerState saat halaman dimuat
loadTimerState();

///// timer2.js for history_webpage.html /////

const loadingtime_ = document.getElementById("loadtime-value");
const downtime_ = document.getElementById("downtime-value");
const operatetime_ = document.getElementById("operatetime-value");

// Fungsi untuk mengambil data dari localStorage
function loadTimerState() {
  const state = JSON.parse(localStorage.getItem("timerState"));
  if (state) {
    const elapsedTime = Math.floor((Date.now() - state.lastUpdated)
/ 1000);
    const secondsdowntime =
          state.secondsdowntime   +   (state.isdowntimeRunning   ?
elapsedTime : 0);
    const secondsoperatetime =
```

```
        state.secondsoperatetime + (state.isoperatetimeRunning ?
elapsedTime : 0);

    downtime_.textContent = formatTime(secondsdowntime);
    operatetime_.textContent = formatTime(secondsoperatetime);
      loadingtime_.textContent = formatTime(secondsdowntime +
secondsoperatetime);
  }
}

// Fungsi untuk memformat waktu
function formatTime(totalSeconds) {
  const hours = String(Math.floor(totalSeconds / 3600)).padStart(2,
"0");
   const minutes = String(Math.floor((totalSeconds % 3600) /
60)).padStart(
    2,
    "0"
  );
  const seconds = String(totalSeconds % 60).padStart(2, "0");
  return `${hours} : ${minutes} : ${seconds}`;
}

// Perbarui timer setiap detik
setInterval(() => {
  const state = JSON.parse(localStorage.getItem("timerState"));
  console.log("Timer state loaded on Page B:", state);
  loadTimerState();
}, 1000);

// Panggil loadTimerState untuk pertama kali saat halaman dimuat
loadTimerState();

window.addEventListener("storage", (event) => {
  if (event.key === "timerState") {
      console.log("Timer state updated from another page:",
event.newValue);
    loadTimerState();
  }
});


///// table.js for history_webpage.html /////

const tableBody = document.querySelector("#dataTable tbody");

// Sample data
let sampleData = JSON.parse(localStorage.getItem("tableData")) ||
[]; // Load data from localStorage
let currentIndex = sampleData.length; // Track the current index
for new rows

function populateTable(data) {
  tableBody.innerHTML = ""; // Clear the table body
  const rowCount = Math.max(10, data.length); // Ensure at least
10 rows
```

```
  for (let i = 0; i < rowCount; i++) {
    const row = data[i] || {}; // Get data for the row, or an empty
object if no data
    const tr = document.createElement("tr");
    tr.innerHTML = `
            <td>${row.no || ""}</td>
            <td>${row.tanggal || ""}</td>
            <td>${row.waktu || ""}</td>
            <td>${row.downTime || ""}</td>
            <td>${row.operationTime || ""}</td>
            <td>${row.loadingTime || ""}</td>
            <td>${row.totalcount || ""}</td>
            <td>${row.totalprocess || ""}</td>
        `;
    tableBody.appendChild(tr);
  }
}

// Fungsi untuk mengambil nilai menit dari timer
function getMinutesFromFormattedTime(formattedTime) {
  const [hours, minutes] = formattedTime.split(":").map(Number);
  return hours * 60 + minutes; // Konversi ke menit
}

// Fungsi untuk mengambil data timer dari elemen
function getTimerData() {
  const downtimeValue =
    document.getElementById("downtime-value")?.textContent || "00
: 00 : 00";
  const operatetimeValue =
     document.getElementById("operatetime-value")?.textContent ||
"00 : 00 : 00";
  const loadingtimeValue =
    document.getElementById("loadtime-value")?.textContent || "00
: 00 : 00";

  console.log("Downtime Value:", downtimeValue);
  console.log("Operatetime Value:", operatetimeValue);
  console.log("Loadingtime Value:", loadingtimeValue);

  return {
    downTime: getMinutesFromFormattedTime(downtimeValue),
    operationTime: getMinutesFromFormattedTime(operatetimeValue),
    loadingTime: getMinutesFromFormattedTime(loadingtimeValue),
  };
}

function generateData() {
  // Ambil nilai timer
  const timerData = getTimerData();

  // Get current date and time
  const now = new Date();
  const currentDate = now.toLocaleDateString("en-GB"); // Format:
DD-MM-YYYY
  const currentTime = now.toLocaleTimeString("en-GB"); // Format:
HH:MM:SS
```

```
  return {
    no: ++currentIndex,
    tanggal: currentDate, // Use current date
    waktu: currentTime, // Use current time
     downTime: timerData.downTime, // Menggunakan downtime dari
timer
       operationTime:   timerData.operationTime,   //   Menggunakan
operatetime dari timer
    loadingTime: timerData.loadingTime, // Menggunakan loadingtime
dari timer
    totalcount: sessionStorage.getItem('totalcountobject'),
    totalprocess: sessionStorage.getItem('totalcountprocess'),
  };
}

// Add new random data
function addRandomData() {
  const newData = generateData();
  sampleData.push(newData);
  saveDataToLocalStorage(); // Save updated data to localStorage
  populateTable(sampleData);
}

// Save data to localStorage
function saveDataToLocalStorage() {
  localStorage.setItem("tableData", JSON.stringify(sampleData));
}

// Clear button functionality
document.getElementById("clearButton").addEventListener("click",
() => {
  sampleData = []; // Clear the sampleData array
  currentIndex = 0; // Reset the currentIndex
  saveDataToLocalStorage(); // Clear data in localStorage
  populateTable(sampleData); // Reinitialize with empty rows
});

// Save button functionality
document.getElementById("saveButton").addEventListener("click",
() => {
  const rows = [...document.querySelectorAll("#dataTable tr")];
  const data = rows.map((row) =>
    [...row.children].map((cell) => cell.innerText)
  );

  // Convert data to XLSX using SheetJS
  const ws = XLSX.utils.aoa_to_sheet(data); // Array of arrays
  const wb = XLSX.utils.book_new();
  XLSX.utils.book_append_sheet(wb, ws, "DataTable");

  // Save as Excel file
  XLSX.writeFile(wb, "table_data.xlsx");
});

// Add button functionality
```

```
document.getElementById("addButton").addEventListener("click", ()
=> {
  addRandomData();
});

// Load data from localStorage on page load
window.addEventListener("DOMContentLoaded", () => {
  populateTable(sampleData);
});

// Include SheetJS library
const script = document.createElement("script");
script.src =
   "https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.18.5/xlsx.full.m
in.js";
document.body.appendChild(script);

///// navigate.js for menu_webpage.html and history_webpage.html
/////

function redirectToHistory() {
  // Ubah URL di sini ke nama file tujuan dalam folder yang sama
  window.location.href = "history_webpage.html";
}

function redirectToControl() {
  // Ubah URL di sini ke nama file tujuan dalam folder yang sama
  window.location.href = "menu_webpage.html";
}
```

Gambar 4. Javascript Dashboard Web Monitoring Data OEE

Uraian gambar program javascript Dashboard Web Monitoring Data OEE, terdapat berbagai file dengan fungsi spesifik untuk mendukung komunikasi perangkat, pengelolaan data, dan tampilan antarmuka secara real-time. Setiap file bekerja secara terintegrasi, memastikan kelancaran monitoring dan kontrol proses industri.

connectionIP.js menangani input alamat IP pengguna serta pengaturan koneksi WebSocket ke perangkat. Sistem akan memeriksa status koneksi sebelum menghubungkan atau memutuskan perangkat dan memperbarui tampilan secara otomatis. esp32.js bertanggung jawab atas komunikasi WebSocket, menerima dan memproses pesan dari perangkat, serta memungkinkan pengguna mengirim perintah seperti mulai, jeda, berhenti, dan mereset semua baik hasil pengolahan data dan visualisasi data serta pengoperasian sistem. File ini juga memastikan status koneksi tetap diperbarui saat halaman dimuat ulang.

processObject.js berfungsi membuat dan memperbarui grafik berdasarkan jumlah objek yang memenuhi dan tidak memenuhi standar, serta mengelola tampilan progress bar serta memiliki fitur reset. inputObject.js memastikan validitas angka.

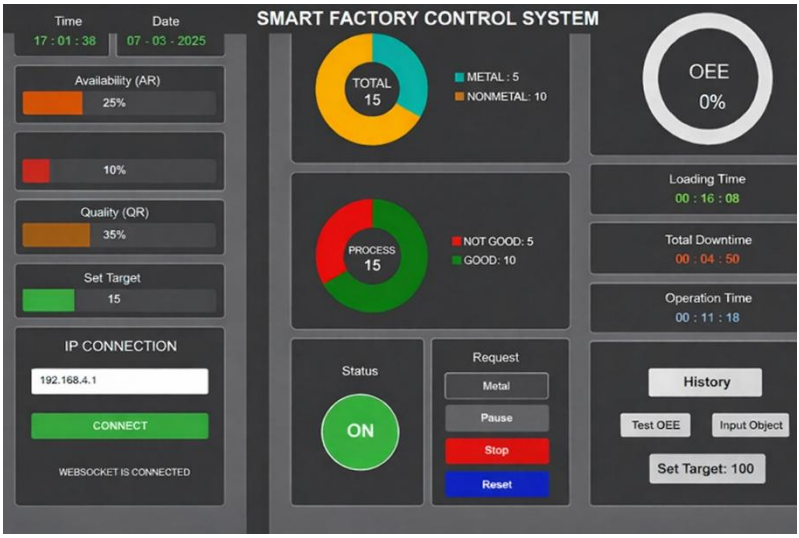
yang dimasukkan pengguna dalam target pemrosesan dan langsung memperbarui tampilan antarmuka. progressOEE.js menghitung efektivitas peralatan (OEE) berdasarkan faktor availability, performance, dan quality, lalu menampilkan data dengan pembaruan nilai real-time. totalObject.js mengelola data objek logam dan non-logam, menyajikan statistik dalam grafik lingkaran, serta menyediakan fitur reset perhitungan.

timeDate.js menampilkan waktu dan tanggal real-time di halaman web untuk memastikan akurasi pencatatan data. timer.js mencatat dan mengelola durasi operasional serta waktu jeda, dengan fitur penyimpanan data agar tetap tersedia meskipun halaman dimuat ulang. timer2.js digunakan di halaman lain untuk menampilkan data waktu yang telah dicatat, sehingga informasi tetap sinkron di seluruh sistem.

table.js bertanggung jawab untuk menampilkan dan memperbarui tabel riwayat berdasarkan data dari localStorage atau input pengguna. Sistem memastikan tabel selalu memiliki jumlah baris minimum, menampilkan informasi downtime, operation time, dan total proses, serta memungkinkan pengguna menambahkan data baru, menyimpan tabel sebagai file Excel, atau menghapus riwayat jika diperlukan. navigate.js mengatur navigasi antar halaman, memastikan pengguna dapat berpindah ke halaman riwayat atau kontrol dengan cepat dan efisien.

Dengan struktur file yang terorganisir dan fungsionalitas yang saling mendukung, sistem ini memberikan solusi pemantauan dan pengelolaan data OEE secara efektif. Integrasi antar file memastikan keakuratan data, kelancaran interaksi pengguna, serta optimasi kontrol perangkat dalam proses industri.

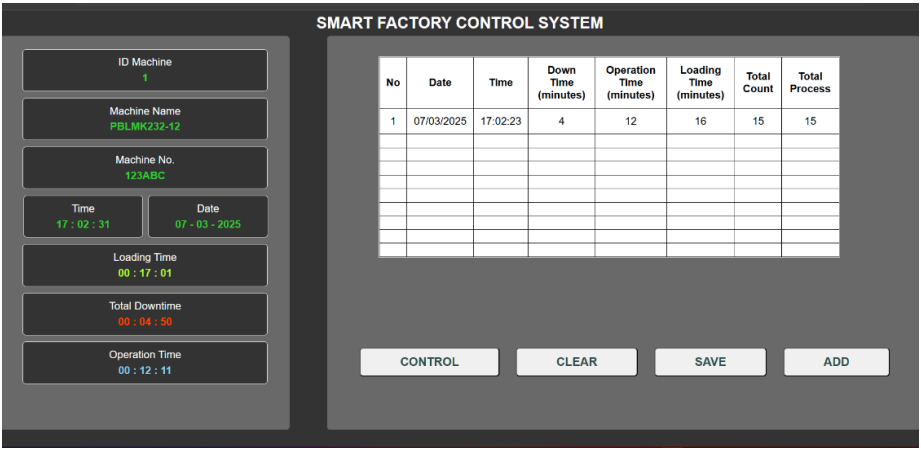
Tampilan Dashboard Web OEE dan Desain CIM Miniatur Smart Factory



Gambar 5. Tampilan Dashboard Monitoring Halaman Utama

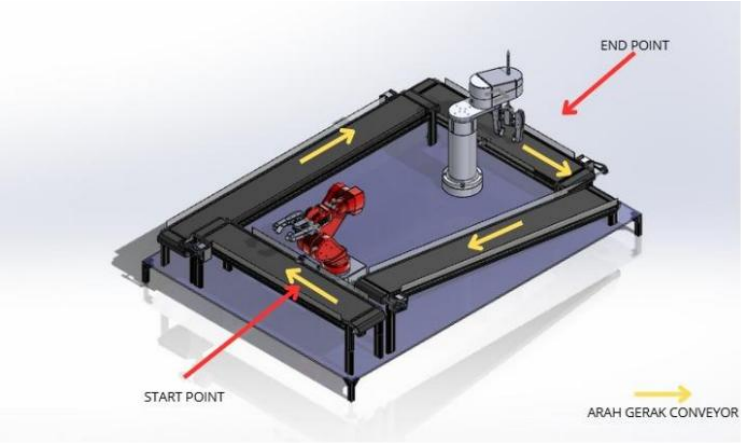
Tampilan Dashboard Smart Factory System ditunjukkan pada gambar 5 dan 6. Pada gambar 5 tampilan dashboard monitoring halaman utama disajikan dalam tiga panel utama: panel kiri menampilkan informasi waktu, tanggal, dan parameter OEE seperti Availability (A), Performance (P), dan Quality (Q) dengan indikator persentase, serta pengaturan target produksi; panel tengah berisi grafik lingkaran yang menampilkan jumlah total objek yang diproses, sistem ini menerapkan *selective counting* di mana angka hanya akan bertambah jika sensor mendeteksi objek yang sesuai dengan *request* awal dan berhasil disimpan oleh robot SCARA. Jika objek tidak sesuai, sistem tetap menjalankan konveyor tanpa melakukan penghitungan, sehingga data yang tampil murni menunjukkan hasil pemrosesan yang valid. Hal ini berlaku baik untuk objek logam maupun non-logam, serta ditampilkan juga status kualitas objek (baik atau tidak baik), dashboard ini dilengkapi dengan indikator status koneksi perangkat dan tombol kontrol operasi (Mulai, Jeda, Henti, Reset); kemudian panel kanan menampilkan nilai OEE dalam persen, informasi loading time, downtime, dan operation time, serta tombol navigasi untuk mengakses riwayat data atau input target

pemrosesan. Sistem ini juga mendukung koneksi ke perangkat melalui WebSocket, ditunjukkan oleh tampilan IP connection dan status koneksi.



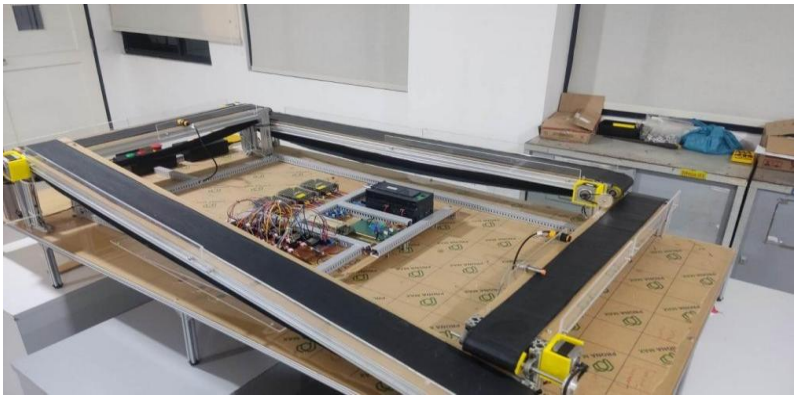
Gambar 6. Tampilan Dashboard Monitoring Halaman Histori

Pada Gambar 6 tampilan dashboard monitoring halaman histori yang digunakan untuk memantau dan mencatat data operasional mesin dalam sebuah sistem manufaktur. Pada panel kiri, terdapat informasi mengenai ID mesin, nama mesin, dan nomor mesin, serta data real-time seperti waktu, tanggal, loading time, total downtime, dan operation time yang diperbarui secara real time. Panel utama di tengah menampilkan tabel riwayat proses produksi yang mencakup nomor, tanggal, waktu, downtime, operation time, loading time, total count, dan total proses. Data pada kolom total count dan total proses merupakan hasil penghitungan objek yang telah divalidasi sesuai dengan permintaan (*request*) awal; sistem hanya mencatat objek yang berhasil diproses oleh robot SCARA, sementara objek yang tidak sesuai permintaan akan diabaikan oleh sistem dan tidak masuk dalam pencatatan histori. Di bawah tabel, terdapat beberapa tombol kontrol, termasuk CONTROL untuk mengelola operasi mesin, CLEAR untuk menghapus data, SAVE untuk menyimpan data, dan ADD untuk menambahkan entri baru. Sistem ini memungkinkan pemantauan dan pengelolaan proses produksi secara lebih efisien dengan pencatatan data operasional yang terstruktur.



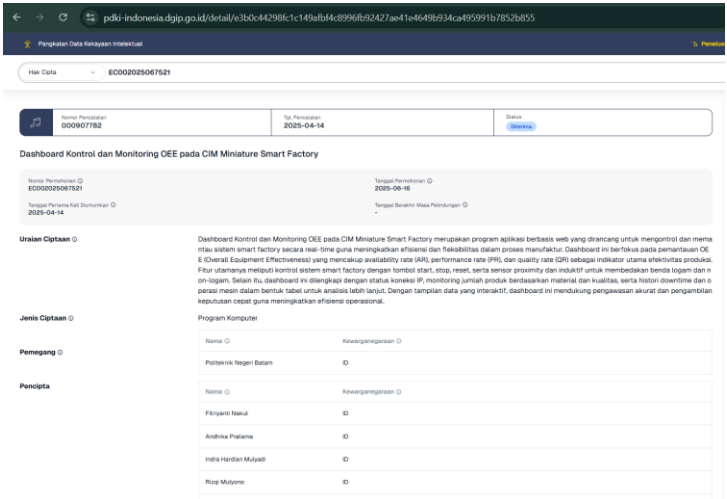
Gambar 7. Tampilan Desain CIM Miniatur Smart Factory

Gambar 7 dan 8 merupakan tampilan desain miniatur *smart factory* yang berbasis *closed-loop conveyor* (lintasan tertutup) dengan 4 buah konveyor, ARM robot, dan SCARA robot. Tanda panah menunjukkan arah konveyor bergerak. Mekanisme sistem dimulai dari permintaan objek metal atau nonmetal, kemudian dilanjutkan menuju *Start Point*, dimana material masuk ke sistem konveyor melalui ARM robot. SCARA robot yang terpasang akan melakukan tugas otomatisasi yang terkontrol yaitu mengambil objek sesuai dengan permintaan. Karena lintasannya berbentuk loop, sistem ini sangat efisien untuk proses yang memerlukan pengerjaan berulang.

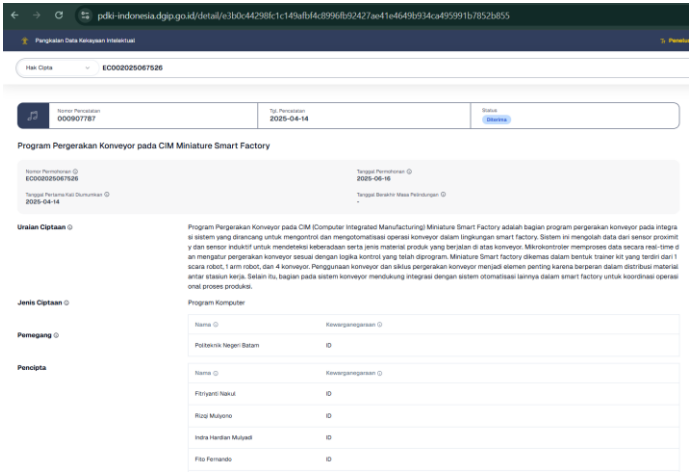


Gambar 8. Tampilan Set-up Pengujian sistem CIM Miniature Smart Factory dikontrol dan monitoring dari dashboard web

Halaman DJKI



Gambar 9. Halaman DJKI Dashboard Kontrol dan Monitoring OEE Pada CIM Miniature Smart Factory



Gambar 10. Halaman DJKI Program Pergerakan Konveyor pada CIM Miniature Smart Factory