DAFTAR PUSTAKA

Abdallah, F.S.M. *et al.* (2023) 'Intelligent solar panel monitoring system and shading detection using artificial neural networks', *Energy Reports*, 9, pp. 324–334. Available at: https://doi.org/https://doi.org/10.1016/j.egyr.2023.

05.163.

Akhtar, I. *et al.* (2021) 'Feasibility Analysis of Solar Technology Implementation in Restructured Power Sector with Reduced Carbon Footprints', *IEEE Access*, 9, pp. 30306–30320. Available at: https://doi.org/10.1109/ACCESS.

2021.3059297.

Alvarez, D.L., Al-Sumaiti, A.S. and Rivera, S.R. (2020) 'Estimation of an Optimal PV Panel Cleaning Strategy Based on Both Annual Radiation Profile and Module Degradation',*IEEE Access*, 8, pp. 63832–63839. Available at: https://doi.org/10.1109/ACCESS.2020.2983322.

Babu, P.R. *et al.* (2023) 'Smart Solar Energy Monitor Using ESP 32 Controller', in *2023 7th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pp. 423–428. Available at: https://doi.org/10.1109/ICECA58529.2023.10395644.

Boonnam, N. and Lanteng, O. (2024) 'Energy yield database management system based on solar photovoltaic cell using internet of things technology', *e-Prime - Advances in Electrical Engineering, Electronics and Energy*, 8, p. 100563. Available at: https://doi.org/https://doi.org/10.1016/j.prime.2024.100563.

Bosman, L.B. *et al.* (2020) 'PV system predictive maintenance: Challenges, current approaches, and opportunities', *Energies*. MDPI AG. Available at: https://doi.org/10.3390/en13061398.

Darwish, Z.A., Sopian, K. and Fudholi, A. (2021) 'Reduced output of photovoltaic modules due to different types of dust particles', *Journal of Cleaner Production*, 280. Available at: https://doi.org/10.1016/j.jclepro.2020.124317.

Hueros-Barrios, P.J. *et al.* (2023) 'A low-cost digital twin for real-time monitoring of photovoltaic panels', in *2023 IEEE 32nd International Symposium on Industrial Electronics (ISIE)*, pp. 1–6. Available at: https://doi.org/10.1109/ISIE51358.2023.10228172.

Kadiyan, N., Pandey, K. and akash, Dr.A.P. (2018) 'Issues and Maintenance of Photo-Voltaic cell based system', *2018 International Conference on Power Energy, Environment and Intelligent Control (PEEIC)*, pp. 77–81. Available at: https://api.semanticscholar.org/CorpusID:77387552.

Kazem, H.A. *et al.* (2020a) 'A review of dust accumulation and cleaning methods solar photovoltaic systems', *Journal of Cleaner Production*. Elsevier Ltd. ıilable at: https://doi.org/10.1016/j.jclepro.2020.123187.

Kazem, H.A. *et al.* (2020b) 'A review of dust accumulation and cleaning methods for solar photovoltaic systems', *Journal of Cleaner Production*. Elsevier Ltd. Available at: https://doi.org/10.1016/j.jclepro.2020.123187.

LÓpez-Vargas, A., Fuentes, M. and Vivar, M. (2018) 'On the application of IoT for real-time monitoring of small stand-alone PV systems: Results from a new smart datalogger', in *2018 IEEE 7th World Conference on Photovoltaic Energy Conversion (WCPEC) (A Joint Conference of 45th IEEE PVSC, 28th PVSEC & 34th EU PVSEC)*, pp. 605–607. Available at: https://doi.org/10.1109/PVSC.2018.8547612.

Martati (2023) *Sistem Deteksi Debu dan Kotoran pada Permukaan Panel Surya Menggunakan Algoritme Reduced Random Forest*. Tesis. Universitas Hasanuddin.

Nezamisavojbolaghi, M. *et al.* (2023) 'The Impact of Dust Deposition on PV Panels' Efficiency and Mitigation Solutions: Review Article', *Energies*. Multidisciplinary Digital Publishing Institute (MDPI). Available at:

https://doi.org/10.3390/en16248022.

Oh, S. (2019) 'Analytic and Monte-Carlo studies of the effect of dust accumulation on photovoltaics', *Solar Energy*, 188, pp. 1243–1247. Available at: https://doi.org/10.1016/j.solener.2019.07.011.

Phoolwani, U.K. *et al.* (2020) 'IoT Based Solar Panel Analysis using Thermal Imaging', in *2020 IEEE International Students' Conference on Electrical, Electronics and Computer Science, SCEECS 2020*. Institute of Electrical and Electronics Engineers Inc. Available at: https://doi.org/10.1109/SCEECS48394.2020.114.

Rouibah, N. *et al.* (2019) 'A low-cost monitoring system for maximum power point of a photovoltaic system using IoT technique', in *2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)*, pp. 1–5. Available at: https://doi.org/10.1109/WITS.2019.8723724.

Rusănescu, C.O. *et al.* (2023) 'The Effect of Dust Deposition on the Performance of Photovoltaic Panels', *Energies*, 16(19). Available at: https://doi.org/10.3390/

en16196794.

Samaulah, H. *et al.* (2018) 'Efficiency Analysis of Tracking and Stationary Solar Panel Modes Against Solar Radiation', *Journal of Engineering Sciences*, 5(1). Available at: https://doi.org/10.21272/jes.2018.5(1).h4.

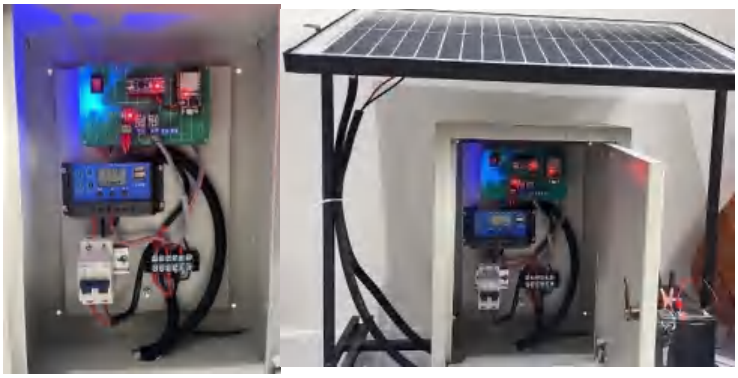
Sinaga, W.D. and Prabowo, Y. (2018) 'Monitoring Tegangan Dan Arus Yang Dihasilkan Oleh Sel Surya Berbasis Web Secara Online', in. Available at: ɔs://api.semanticscholar.org/CorpusID:209932762.

, M., Kulaksiz, A.A. and Unluturk, A. (2019) 'Image Processing-based ɜessment of Dust Accumulation on Photovoltaic Modules', in *Proceedings 019 IEEE 1st Global Power, Energy and Communication Conference,*

GPECOM 2019. Institute of Electrical and Electronics Engineers Inc., pp. 308–311. Available at: https://doi.org/10.1109/GPECOM.2019.8778578.

LAMPIRAN

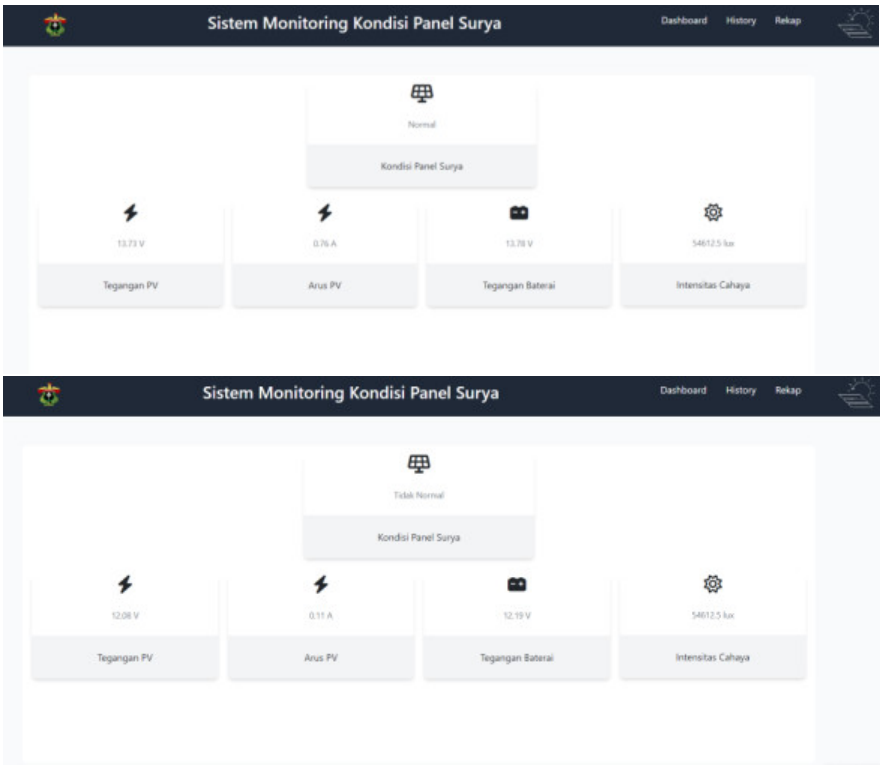
Lampiran 1 Dokumentasi Proses pembuatan rangka dan perakitan komponen



Lampiran 2 Dokumentasi proses pengambilan data

1. Pengambilan Dataset:

2. Pengujian pengaruh debu dengan beban motor DC 12 Volt



3. Pengujian sistem:

Lampiran 3 Kode program *Mikrokontroller*

1. Kode Arduino nano

```
#include <Wire.h>
#include "ACS712.h"
#include <BH1750.h>
#include <SoftwareSerial.h>
BH1750 lightMeter;
ACS712 currentSensor(ACS712_05B, A0);
const int analogPinPV = A1;
const int analogPinBattery = A2;
const int relay = 4;
float Vmodul = 0.0;
float Vbattery = 0.0;
float hasilPV = 0.0;
float hasilBattery = 0.0;
float R1 = 30000.0; // 30k ohm resistor
float R2 = 7500.0;  // 7500 ohm resistor
float R1B = 30000.0; // 29.8k ohm resistor
float R2B = 7500.0;  // 7500 ohm resistor
```

```
int valuePV = 0;
int valueBattery = 0;
SoftwareSerial espSerial(2,3); // RX (2), TX (3)
void setup() {
    Serial.begin(9600);
    Wire.begin();
    lightMeter.begin();
      pinMode(relay, OUTPUT);
      pinMode(analogPinPV, INPUT);
      pinMode(analogPinBattery, INPUT);
      digitalWrite(relay, HIGH);
      delay(2000);
      digitalWrite(relay, LOW);
    espSerial.begin(9600); // Inisialisasi komunikasi dengan ESP32
    int zero = currentSensor.calibrate();
    Serial.println("Zero point = " + String(zero));
    delay(1000);
}

void loop() {
    digitalWrite(relay, LOW);
    float currentPV = currentSensor.getCurrentDC();
    if (currentPV < 0) {
        currentPV = 0;
    }
    float lux = lightMeter.readLightLevel();
    // Mengukur tegangan PV
    valuePV = analogRead(analogPinPV);
    Vmodul = (valuePV * 5.0) / 1023.0;
    hasilPV = Vmodul / (R2 / (R1 + R2));
    // Mengukur tegangan baterai
    valueBattery = analogRead(analogPinBattery);
    Vbattery = (valueBattery * 5.0) / 1023.0;
    hasilBattery = Vbattery / (R2B / (R1B + R2B));
    // Menampilkan hasil pengukuran di Serial Monitor
    Serial.print("V_PV: ");
    Serial.print(hasilPV, 2);
    Serial.print(" volt, V_BAT: ");
    Serial.print(hasilBattery, 2);
    Serial.print(" volt, I_PV = ");
    Serial.print(currentPV, 2);
    Serial.print(" A, Cahaya: ");
        al.print(lux);
        al.println(" LUX");
        engirim data ke ESP32 melalui Software Serial
        erial.print("PV=");
        erial.print(hasilPV, 2);
```

```
    espSerial.print(" BAT=");
    espSerial.print(hasilBattery, 2);
    espSerial.print(" CURRENT=");
    espSerial.print(currentPV, 2);
    espSerial.print(" LUX=");
    espSerial.println(lux);
    delay(30000);
}
```

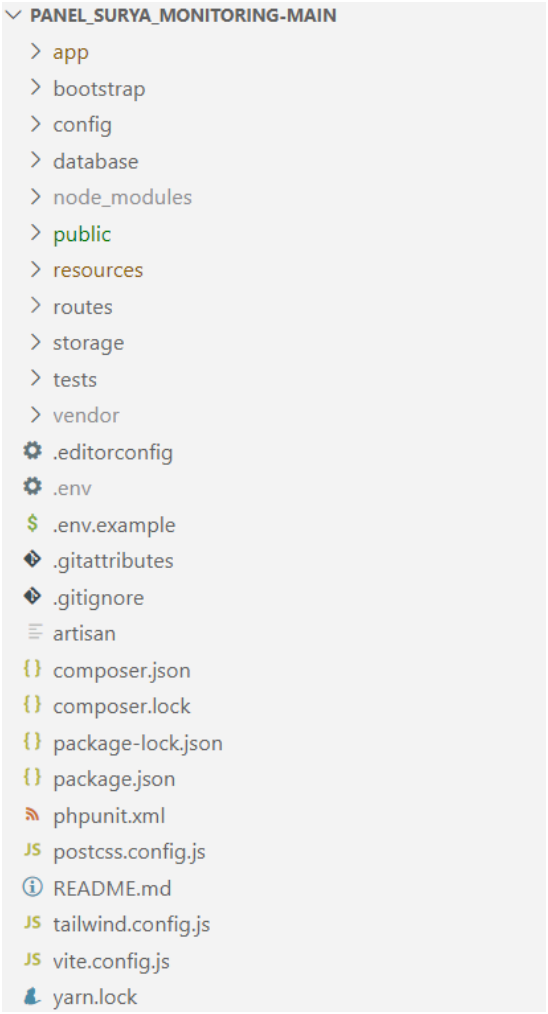
2. Kode Esp32

```
#include <WiFi.h>
#include <HTTPClient.h>
HardwareSerial arduinoserial(2); // RX2 = 16 dan TX2 = 17
const char *ssid = "BSB";
const char *password = "Robot2024";
const char *serverUrl = "http://192.168.1.9:80/api/input"; //koneksi
ke server

void setup() {
    Serial.begin(115200);
    arduinoserial.begin(9600);
    // Connect to WiFi
    WiFi.begin(ssid, password);
    Serial.print("Connecting to WiFi");
    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
        Serial.print(".");
    }
    Serial.println("Connected to WiFi");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
    Serial.println();
    delay(1000);
}

void loop() {
    if (arduinoserial.available() > 0) {
        float teganganPV, teganganBat, arusPV, intensitasCahaya;
        // Parsing data dari Arduino
        sscanf(arduinoserial.readStringUntil('\n').c_str(), "PV=%f
BAT=%f CURRENT=%f LUX=%f",
               &teganganPV, &teganganBat, &arusPV,
        tasCahaya);

        // Menampilkan data yang diparsing dari Arduino
        Serial.print("Tegangan PV: ");
```

```
        Serial.print(teganganPV);
        Serial.print(" V, Tegangan Bat: ");
        Serial.print(teganganBat);
        Serial.print(" V, Arus PV: ");
        Serial.print(arusPV);
        Serial.print(" A, Intensitas Cahaya: ");
        Serial.print(intensitasCahaya);
        Serial.println(" LUX");
        // Mengirimkan data ke server melalui HTTP POST request
        HTTPClient http;
        String url = String(serverUrl);
        String postData = "tegangan_pv=" + String(teganganPV) +
                          "&tegangan_bat=" + String(teganganBat) +
                          "&arus_pv=" + String(arusPV) +
                          "&intensitas_cahaya=" +
String(intensitasCahaya);

        // Mulai koneksi HTTP
        http.begin(url);
        http.addHeader("Content-Type", "application/x-www-form-
urlencoded");
        int httpResponseCode = http.POST(postData);
        if (httpResponseCode > 0) {
            Serial.print("HTTP Response code: ");
            Serial.println(httpResponseCode);
            String response = http.getString();
            Serial.println(response);
        } else {
            Serial.print("HTTP Error code: ");
            Serial.println(httpResponseCode);
        }
        http.end();
    }
    delay(20000);
}
```

Lampiran 4 Kode Program *Website*

Kode berikut merupakan sebagian kode yang merupakan inti kode dari keseluruhan kode untuk *website* yang di masukkan pada lampiran. Berikut keseluruhan direktori yang dibuat menggunakan *framework laravel*:



1. Koneksi endpoint dengan ESP32 (API.php)

```php
<?php
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\DataInputController;
use App\Http\Controllers\DashboardController;
Route::middleware('auth:sanctum')->get('/user', function (Request
$request) {
    return $request->user();
});
Route::post('/input', [DataInputController::class, 'input'])-
    nput');
    et('/getsensor', [DashboardController::class,
    sor']);
    et('/getsensorcondition', [DashboardController::class,
    ondition']);
```

2. RealtimeSensor.php

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class realtimesensor extends Model
{
    use HasFactory;
    protected $guarded = ['id'];
}
```

3. Untuk backend (ReadingCondition.php)

```
<?php
namespace App\Models;
use Carbon\Carbon;
use DateTime;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class ReadingCondition extends Model
{
    protected $fillable = ['voltage', 'lux', 'time'];

    public static function evaluateCondition($voltage, $lux, $time)
    {
$thresholds = [
["range_time" => "08:00-10:00", "threshold_voltage" => 12.10,
"threshold_lux" => 25000],
            ["range_time" => "10:00-12:00", "threshold_voltage" =>
12.10, "threshold_lux" => 50000],
            ["range_time" => "12:00-14:00", "threshold_voltage" =>
12.10, "threshold_lux" => 50000],
            ["range_time" => "14:00-16:00", "threshold_voltage" =>
12.10, "threshold_lux" => 50000],
            ["range_time" => "16:00-17:00", "threshold_voltage" =>
12.10, "threshold_lux" => 35000],
            ["range_time" => "17:00-17:30", "threshold_voltage" =>
12.00, "threshold_lux" => 2392],
            ["range_time" => "17:30-17:40", "threshold_voltage" =>
12.00, "threshold_lux" => 1000],
            ["range_time" => "17:41-18:00", "threshold_voltage" =>
 threshold_lux" => 500]
 ];
 $threshold = null;
 $currentTime = DateTime::createFromFormat('H:i', $time);
```

```
        foreach ($thresholds as $item) {
            $range = explode('-', $item['range_time']);
            $start = DateTime::createFromFormat('H:i',
trim($range[0]));
            $end = DateTime::createFromFormat('H:i',
trim($range[1]));

            if ($currentTime >= $start && $currentTime <= $end) {
                $threshold = $item;
                break;
            }
        }

        if (!$threshold) {
            return "Malam Hari";
        }

        if ($voltage > $threshold['threshold_voltage'] && $lux <
$threshold['threshold_lux']) {
            return "Normal";
        } elseif ($voltage < $threshold['threshold_voltage'] && $lux
> $threshold['threshold_lux']) {
            return "Tidak Normal";
        } elseif ($voltage < $threshold['threshold_voltage'] && $lux
< $threshold['threshold_lux']) {
            return "Normal";
        }
        return "Tidak Normal";
    }}
```

4. Untuk Frontend (Dashboard.blade.php)

views > Dashboard.blade.php

```
@extends('template.Main')
@section('head')
<script
src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.4/Chart.js">
</script>
{{-- <style>
  #dashboard-main{
    background-image: url('/assets/img/Energy_panel_power_solar_light-512.webp');
  background-repeat: no-repeat;
  background-origin: content-box;
  background-size: 10% auto;
  background-position-x: right;
  background-position-y: bottom;
  } --}}
</style>
@endsection
@section('content')
<div id="dashboard-main" class='bg-white  container-sm my-5 rounded-md h-full p-3  space-y-7'  >

    <div class='w-full flex justify-center text-center'>

        @include('component.Card', [
            'title' => 'Kondisi Panel Surya',
            'icon' => '<i class="fa-solid fa-solar-panel text-3xl"></i>',
            'value' => '<span id="panel_value"></span>'
            ])
    </div>
    <div class='w-full text-center d-flex justify-content-center gap-3 '>
        <div class='w-full'>
            @include('component.Card', [
                'title' => 'Tegangan PV',
                'icon' => '<i class="fa-solid fa-bolt text-3xl"></i>',
                'value' => '<span id="tegangan_value"></span>'
                ])
        </div>
        <div class='w-full'>
            @include('component.Card', [
                'title' => 'Arus PV',
                'icon' => '<i class="fa-solid fa-bolt text-3xl"></i>',
                'value' => '<span id="arus_value"></span>'
                ])
        </div>
        <div class='w-full'>
            @include('component.Card', [
                'title' => 'Tegangan Baterai',
                'icon' => '<i class="fa-solid fa-car-battery text-3xl"></i>',
                'value' => '<span id="batt_value"></span>'
                ])
        </div>
        <div class='w-full'>
            @include('component.Card', [
                'title' => 'Intensitas Cahaya',
                'icon' => '<i class="fa-regular fa-sun text-3xl "></i>',
                'value' => '<span id="lux_value"></span>'
                ])
        </div>
    </div>



    <div class='w-full flex justify-center'>
      {{-- <canvas id="myChart" style="width:100%;max-width:700px"></canvas> --}}
    </div>
</div>
@stop
```

```
@push('script')
<script>
  document.addEventListener("DOMContentLoaded", function(event) {

    const xValues = [50,60,70,80,90,100,110,120,130,140,150];
  const yValues = [7,8,8,9,9,9,10,11,14,14,15];

  new Chart("myChart", {
    type: "line",
    data: {
      labels: xValues,
      datasets: [{
        fill: false,
        lineTension: 0,
        backgroundColor: "rgba(0,0,255,1.0)",
        borderColor: "rgba(0,0,255,0.1)",
        data: yValues
      }]
    },
    options: {
      legend: {display: false},
      scales: {
        yAxes: [{ticks: {min: 6, max:16}}],
      }
    }
  });
  })
  </script>
```

```
<script type='module'>
    const interval = 30*1000
    function convertTimeToMinutes(time) {
  let [hours, minutes] = time.split(":").map(Number);
  return hours * 60 + minutes;
}
    function evaluateCondition(voltage, lux, time) {
  let thresholds = [
    { range_time: "08:00-10:00", threshold_voltage: 12.10, threshold_lux: 25000 },
    { range_time: "10:00-12:00", threshold_voltage: 12.10, threshold_lux: 50000 },
    { range_time: "12:00-14:00", threshold_voltage: 12.10, threshold_lux: 50000 },
    { range_time: "14:00-16:00", threshold_voltage: 12.10, threshold_lux: 50000 },
    { range_time: "16:00-17:00", threshold_voltage: 12.10, threshold_lux: 35000 },
    { range_time: "17:00-17:30", threshold_voltage: 12.00, threshold_lux: 2392 },
    { range_time: "17:30-17:40", threshold_voltage: 12.00, threshold_lux: 1000 },
    { range_time: "17:41-18:00", threshold_voltage: 12.00, threshold_lux: 500 }
  ];

  let threshold = null;
  let currentTime = convertTimeToMinutes(time);
  for (let i = 0; i < thresholds.length; i++) {
    let item = thresholds[i];
    let range = item.range_time.split('-');
    let start = convertTimeToMinutes(range[0]);
    let end = convertTimeToMinutes(range[1]);

    if (currentTime >= start && currentTime <= end) {
      threshold = item;
      break;
    }
  }

  if (!threshold) {
    return "Malam Hari";
  }
```

```
  if (voltage > threshold.threshold_voltage && lux < threshold.threshold_lux) {
    return "Normal";
  } else if (voltage < threshold.threshold_voltage && lux > threshold.threshold_lux) {
    return "Tidak Normal";
  } else if (voltage < threshold.threshold_voltage && lux < threshold.threshold_lux) {
    return "Normal";
  }

  return "Tidak Normal";
}
```

```
  document.addEventListener("DOMContentLoaded", function(event) {
      const getSensorData = () => {
      return axios.get(`${window.location.href}api/getsensor`)
      .then(function (response) {
      const data = response?.data
      const dataPrev = localStorage.hasOwnProperty("prev") ? JSON.parse(localStorage.getItem("prev")) : null
      const isValPrev = localStorage.hasOwnProperty("prev") ? (JSON.stringify(data) === JSON.stringify(dataPrev)) : false
      localStorage.setItem('prev', JSON.stringify(data))

      const timeNow = new Date().getTime();
      const timeApi = new Date(data?.created_at).getTime();

      const difference = Math.round((timeNow - timeApi) / 1000 / 60);
        console.log({difference}, difference > 1 , isValPrev)
        console.log({data}, {dataPrev})
      const time = new Date(data.created_at)
      const clock = time.getHours() + ":" + time.getMinutes()
      const condition = evaluateCondition(data?.arus_pv, data?.intensitas_cahaya, clock)
                panel_value.innerText = condition
                lux_value.innerText = `${data?.intensitas_cahaya} lux`
                batt_value.innerText = `${data?.tegangan_bat} V`
                arus_value.innerText = `${data?.arus_pv} A`
                tegangan_value.innerText = `${data?.tegangan_pv} V`

                if(difference > 1 && isValPrev){
                  console.log('prev')
                  panel_value.innerText = "Sensor Tidak Terhubung"
                lux_value.innerText = 0
                batt_value.innerText = 0
                arus_value.innerText = 0
                tegangan_value.innerText = 0
                }

      })}
      getSensorData()
      setInterval(() => {
          getSensorData()
          }, interval);
  });
</script>
```

Lampiran 5 Tampilan website kondisi malam hari