LAMPIRAN

1. main..cpp (Arduino Uno)

```
#include <Arduino.h>
#include <SoftwareSerial.h>

SoftwareSerial toEsp(2, 3); // RX, TX

//  green light on relay
const int relayPin_green = 5;
const int checkPLN_green = 11;
const int analogIn_green = A2;

//  yellow light on relay
const int relayPin_yellow = 6;
const int checkPLN_yellow = 12;
const int analogIn_yellow = A1;

//  red light on relay
const int relayPin_red = 7;
const int checkPLN_red = 13;
const int analogIn_red = A0;

const int mVperAmp = 66;    // 66 mV per Amp for ACS712-30A
const int ACSoffset = 2500; // Offset for zero current (mV)

float amplitude_current = 0.0; // Amplitude current
float effective_value = 0.0;   // Effective current (RMS)

int getMaxValue(int duration, int analogPin)
{
  int sensorValue; // Value read from the sensor
  int sensorMax = 0;
  uint32_t start_time = millis();
  while ((millis() - start_time) < duration)
  {
    sensorValue = analogRead(analogPin);
    if (sensorValue > sensorMax)
    {
      sensorMax = sensorValue; // Record the maximum sensor value
    }
  }
  return sensorMax;
}

void controlRelay(int relayPin, int checkPLN, int analogIn , int duration)
{
  digitalWrite(relayPin, LOW);
  String relayStat;
  if (relayPin == relayPin_green)
  {
    relayStat = "Green";
    Serial.println("Green ON");
    digitalWrite(relayPin_yellow, HIGH);
    digitalWrite(relayPin_red, HIGH);
  }
  else if (relayPin == relayPin_yellow)
  {
    relayStat = "Yellow";
    Serial.println("Yellow ON");
    digitalWrite(relayPin_green, HIGH);
    digitalWrite(relayPin_red, HIGH);
  }
  else if (relayPin == relayPin_red)
  {
    relayStat = "Red";
    Serial.println("Red ON");
    digitalWrite(relayPin_green, HIGH);
    digitalWrite(relayPin_yellow, HIGH);
  }
```

```
  for (int i = 0; i < duration; i++)
  {
    int ii = duration - i;
    String pln;
    byte x = digitalRead(checkPLN);
    Serial.println(relayStat);
    if (x == 0)
    {
      Serial.println("PLN ON");
      pln = "PLN ON";
    }
    else
    {
      Serial.println("PLN OFF");
      pln = "PLN OFF";
    }

    int sensorMax = getMaxValue(1000, analogIn); // Get max value over 1 second

    float voltage = (sensorMax / 1024.0) * 5000.0; // Convert to millivolts
    amplitude_current = (voltage - ACSoffset) / mVperAmp;
    effective_value = amplitude_current / 1.414; // RMS calculation

    Serial.print("Current Max (A) : ");
    Serial.println(amplitude_current, 3);
    Serial.print("Current RMS (A) : ");
    Serial.println(effective_value, 3);
    Serial.println();

    toEsp.print(relayStat + "," + pln + "," + String(amplitude_current, 3) + "," + String(effective_value, 3) + "," + String(voltage, 3) +"," +ii+"\n");
    // delay(1000); // Delay between readings
  }
}

void setup()
{
  pinMode(relayPin_green, OUTPUT);
  pinMode(analogIn_green, INPUT);
  pinMode(checkPLN_green, INPUT);

  pinMode(relayPin_yellow, OUTPUT);
  pinMode(analogIn_yellow, INPUT);
  pinMode(checkPLN_yellow, INPUT);

  pinMode(relayPin_red, OUTPUT);
  pinMode(analogIn_red, INPUT);
  pinMode(checkPLN_red, INPUT);
  Serial.begin(115200);
  toEsp.begin(9600);
  digitalWrite(relayPin_green, HIGH);
// Ensure relay starts OFF
  digitalWrite(relayPin_yellow, HIGH);
// Ensure relay starts OFF
  digitalWrite(relayPin_red, HIGH);
// Ensure relay starts OFF
}

void loop()
{
  controlRelay(relayPin_green, checkPLN_green, analogIn_green , 30);
//60 - 90 detik
  controlRelay(relayPin_yellow, checkPLN_yellow, analogIn_yellow ,  5);
// 3- 5 detik
  controlRelay(relayPin_red, checkPLN_red, analogIn_red, 30);
//30 - 60 detik
}
```

2. Main.cpp (ESP8266)

```
#include <Arduino.h>
#include <SoftwareSerial.h>

SoftwareSerial toEsp(2, 3); // RX, TX

//  green light on relay
const int relayPin_green = 5;
const int checkPLN_green = 11;
const int analogIn_green = A2;

//  yellow light on relay
const int relayPin_yellow = 6;
const int checkPLN_yellow = 12;
const int analogIn_yellow = A1;

//  red light on relay
const int relayPin_red = 7;
const int checkPLN_red = 13;
const int analogIn_red = A0;

const int mVperAmp = 66;    // 66 mV per Amp for ACS712-30A
const int ACSoffset = 2500; // Offset for zero current (mV)

float amplitude_current = 0.0; // Amplitude current
float effective_value = 0.0;   // Effective current (RMS)

int getMaxValue(int duration, int analogPin)
{
  int sensorValue; // Value read from the sensor
  int sensorMax = 0;
  uint32_t start_time = millis();

  while ((millis() - start_time) < duration)
  {
    sensorValue = analogRead(analogPin);
    if (sensorValue > sensorMax)
    {
      sensorMax = sensorValue; // Record the maximum sensor value
    }
  }
  return sensorMax;
}

void controlRelay(int relayPin, int checkPLN, int analogIn , int duration)
{
  digitalWrite(relayPin, LOW);
  String relayStat;
  if (relayPin == relayPin_green)
  {
    relayStat = "Green";
    Serial.println("Green ON");
    digitalWrite(relayPin_yellow, HIGH);
    digitalWrite(relayPin_red, HIGH);
  }
  else if (relayPin == relayPin_yellow)
  {
    relayStat = "Yellow";
    Serial.println("Yellow ON");
    digitalWrite(relayPin_green, HIGH);
    digitalWrite(relayPin_red, HIGH);
  }
  else if (relayPin == relayPin_red)
  {
    relayStat = "Red";
    Serial.println("Red ON");
    digitalWrite(relayPin_green, HIGH);
```

```
    digitalWrite(relayPin_yellow, HIGH);
  }
  for (int i = 0; i < duration; i++)
  {
    int ii = duration - i;
    String pln;
    byte x = digitalRead(checkPLN);
    Serial.println(relayStat);
    if (x == 0)
    {
      Serial.println("PLN ON");
      pln = "PLN ON";
    }
    else
    {
      Serial.println("PLN OFF");
      pln = "PLN OFF";
    }

    int sensorMax = getMaxValue(1000, analogIn); // Get max value over 1 second

    float voltage = (sensorMax / 1024.0) * 5000.0; // Convert to millivolts
    amplitude_current = (voltage - ACSoffset) / mVperAmp;
    effective_value = amplitude_current / 1.414; // RMS calculation

    Serial.print("Current Max (A) : ");
    Serial.println(amplitude_current, 3);
    Serial.print("Current RMS (A) : ");
    Serial.println(effective_value, 3);
    Serial.println();

    toEsp.print(relayStat + "," + pln + "," + String(amplitude_current, 3) + "," + String(effective_value, 3) + "," + String(voltage, 3) +"," +ii+"\n");
    // delay(1000); // Delay between readings
  }
}

void setup()
{
  pinMode(relayPin_green, OUTPUT);
  pinMode(analogIn_green, INPUT);
  pinMode(checkPLN_green, INPUT);

  pinMode(relayPin_yellow, OUTPUT);
  pinMode(analogIn_yellow, INPUT);
  pinMode(checkPLN_yellow, INPUT);

  pinMode(relayPin_red, OUTPUT);
  pinMode(analogIn_red, INPUT);
  pinMode(checkPLN_red, INPUT);
  Serial.begin(115200);
  toEsp.begin(9600);
  digitalWrite(relayPin_green, HIGH);
// Ensure relay starts OFF
  digitalWrite(relayPin_yellow, HIGH);
// Ensure relay starts OFF
  digitalWrite(relayPin_red, HIGH);
// Ensure relay starts OFF
}

void loop()
{
  controlRelay(relayPin_green, checkPLN_green, analogIn_green , 30);
//60 - 90 detik
  controlRelay(relayPin_yellow, checkPLN_yellow, analogIn_yellow ,  5);
// 3 - 5 detik
```

```
  controlRelay(relayPin_red,
checkPLN_red, analogIn_red, 30);
//30 - 60 detik
}
```

3. index.js (Server)

```
const express = require('express');
const http = require('http');
const cors = require('cors');
const dotenv = require('dotenv');
const socket = require('./socket');
const app = express();
const server = http.createServer(app);
const io = socket.init(server);
const iosend = socket.getIO();


dotenv.config();

app.use(express.json());
app.use(express.urlencoded({ extended:
true }));
// app.use(fileUpload());
app.options('*', cors());
app.use(cors());

function isValidDate(dateString) {
  // Check if the string matches the
format YYYY-MM-DD using a regular
expression
  const regex = /^\d{4}-\d{2}-\d{2}$/;
  if (!regex.test(dateString)) {
    return false;
  }

  // Parse the string to a Date object
  const date = new Date(dateString);

  // Check if the date is valid by
comparing it with the components of
the original string
  const timestamp = date.getTime();
  if (isNaN(timestamp)) {
    return false;
  }

  // Further check to ensure that the
date components match the input
string
  const [year, month, day] =
dateString.split('-').map(Number);
  if (date.getUTCFullYear() !== year ||
date.getUTCMonth() + 1 !== month ||
date.getUTCDate() !== day) {
    return false;
  }

  return true;
}

function timeStringToDate(timeStr) {
  const [hours, minutes, seconds] =
timeStr.split(':').map(Number);
  const date = new Date();
  date.setHours(hours, minutes, seconds,
0);
  return date;
}

// Handle POST requests to '/post'
app.post('/', async (req, res) => {
  // Access the POST data sent in the
request body
  try {
    var postData = req.body;
    // console.log(postData);
    const {no ,light ,value2 ,pln } =
postData;
```

```
    console.log(no ,light ,value2 ,pln);
    // rms = value2
    iosend.emit('datanya',  {
      no : no,
      light : light,
      pln : pln,
      rms : value2});

    return
res.status(200).send({ message: 'OK
data' });

  } catch (error) {
    console.error(error);
    return
res.status(500).send({ message:
'Internal server error' });
  }

});

app.get('/', async (req, res) => {
  return res.status(200).send({ message:
'This is a GET request' });

})

// app error handler
app.use((err, req, res, next) => {
  console.log(err);
  res.status(500).send('Something
broke!');
});

io.on('connection', (socket) => {
  let userID = socket.id;
  console.log('A user connected: ' +
userID);

  socket.on('scan_dia', (data) => {
    console.log('Received scan_dia event:
' + data);
  });

  socket.on('disconnect', () => {
    console.log('User disconnected: ' +
userID);
  });
});

module.exports = {
  app,
  server,
  io
};

const port = process.env.PORT || 3004;

// Start the server
server.listen(port, () => {
  console.log(`Server is running on
http://localhost:${port}`);
});
```