Program Smart Power Consumption Monitoring System Using ESP 32 and PZEM004T Sensor

```
//Pengambilan Library Sensor RFID
#include <MFRC522.h>
#include <SPI.h>
//Pengambilan Library LCD
#include <LiquidCrystal_I2C.h>

//Pengambilan Library Untuk Web Server
#include "WiFi.h"
#include "ESPAsyncWebServer.h"
#include "SPIFFS.h"

//Pengambilan Library Sensor PZEM004T Versi 3.0
#include <PZEM004Tv30.h>
PZEM004Tv30 pzem(&Serial2);

//GPIO LED
const int ledMerah = 27; //GPIO 27
const int ledKuning = 25; //GPIO 25
const int ledHijau = 26; //GPIO 26

//setting PWM properties
const int frekuensi = 5000;
const int ledChannelKuning = 0;
const int resolution = 8;

//multitask
TaskHandle_t Task1;
TaskHandle_t Task2;

//Inisialisasi Aktuator
const int relay = 13;
const int PIN_RST = 15;
const int PIN_SS  = 4;
const int BUZZER  = 2;

// Pembuatan objek MFRC522
String uidTag;
MFRC522 mfrc(PIN_SS, PIN_RST);

float daya, energi, tegangan, arus;
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Replace with your network credentials
const char* ssid = "Cendrawasih";
const char* password = "cendrawasih";

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

String getTegangan(){
    return String(tegangan);
}

```

```
String getArus(){
    return String(arus);
}

String getDaya(){
    return String(daya);
}

String getEnergi(){
    return String(energi);
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  SPI.begin();
  mfrc.PCD_Init();
  pinMode(ledHijau, OUTPUT);
  pinMode(ledMerah, OUTPUT);
  pinMode(relay, OUTPUT);
  pinMode(BUZZER, OUTPUT);
  lcd.init();
  lcd.backlight();
  lcd.blink_on();

  //konfigurasi fungsi LED PWM
  ledcSetup(ledChannelKuning, frekuensi, resolution);

  //menghubungkan channel ke GPIO untuk bisa dikontrol
  ledcAttachPin(ledKuning,ledChannelKuning);

  //SPIFFS
  if (! SPIFFS.begin (true)) {
    Serial.println ("An Error has occurred while mounting SPIFFS");
    return;
  }

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
  }

  // Print ESP32 Local IP Address
    Serial.println(WiFi.localIP());

  // Route for web page
  server.on ("/", HTTP_GET, [] (AsyncWebServerRequest * request) {
  request-> send (SPIFFS, "/Webserver_spowcom.html");
  });

  server.on ("/style.css", HTTP_GET, [] (AsyncWebServerRequest * request) {
  request-> send (SPIFFS, "/style.css", "text/css");
  });

  server.on("/image.png", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send(SPIFFS, "/image.png", "image/png");
  });

  server.on("/timer.png", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send(SPIFFS, "/timer.png", "timer/png");
  });

  server.on ("/chart.js", HTTP_GET, [] (AsyncWebServerRequest * request) {
  request-> send (SPIFFS, "/chart.js", "text/js");
  });

  server.on ("/app.js", HTTP_GET, [] (AsyncWebServerRequest * request) {
  request-> send (SPIFFS, "/app.js", "app/js");
  });

  server.on ("/tegangan", HTTP_GET, [] (AsyncWebServerRequest * request) {
  request-> send_P (200, "text / plain", getTegangan().c_str ());
  });

  server.on ("/arus", HTTP_GET, [] (AsyncWebServerRequest * request) {
  request-> send_P (200, "text / plain", getArus().c_str ());
  });

  server.on ("/daya", HTTP_GET, [] (AsyncWebServerRequest * request) {
  request-> send_P (200, "text / plain", getDaya().c_str ());
  });

  server.on ("/energi", HTTP_GET, [] (AsyncWebServerRequest * request) {
  request-> send_P (200, "text / plain", getEnergi().c_str ());
  });

  // start server
  server.begin ();

  xTaskCreatePinnedToCore(
                  Task1code,   /* Task function. */
                  "Task1",     /* name of task. */
                  10000,       /* Stack size of task */
                  NULL,        /* parameter of the task */
                  1,           /* priority of the task */
                  &Task1,     /* Task handle to keep track of created task */
                  0);          /* pin task to core 0 */

  xTaskCreatePinnedToCore(
                  Task2code,   /* Task function. */
                  "Task2",     /* name of task. */
                  10000,       /* Stack size of task */
                  NULL,        /* parameter of the task */
                  2,           /* priority of the task */
                  &Task2,     /* Task handle to keep track of created task */
                  1);          /* pin task to core 1 */
```

```
}

void Task1code( void * pvParameters ){
  Serial.print("Task1 running on core ");
  Serial.println(xPortGetCoreID());

  for(;;){
    //menghitung besaran listrik
    tegangan = pzem.voltage(); //volt
    arus = pzem.current(); //ampere
    daya = pzem.power(); //watt
    energi = pzem.energy(); //kWh
    vTaskDelay(50 / portTICK_PERIOD_MS);

    if(energi < 0.5){
      digitalWrite(ledHijau, HIGH);
      digitalWrite(ledMerah, LOW);
    } else {
      digitalWrite(ledHijau, LOW);
      digitalWrite(ledMerah, HIGH);
    }

    if(energi >= 0.5 || daya > 1000){
      digitalWrite(ledMerah, HIGH);
      for(int dutyCycle = 0; dutyCycle <= 255; dutyCycle++){
        // changing the LED brightness with PWM
        ledcWrite(ledChannelKuning, dutyCycle);
        delay(15);
      }
      // menurunkan LED brightness
      for(int dutyCycle = 255; dutyCycle >= 0; dutyCycle--){
        // ubah LED brightness dengan PWM
        ledcWrite(ledChannelKuning, dutyCycle);
        delay(10);
      }
    }
  }
}

void Task2code( void * pvParameters ){
  Serial.print("Task2 running on core ");
  Serial.println(xPortGetCoreID());

  for(;;){
    Serial.print("Daya : ");
    Serial.print(daya);
    Serial.println(" W");
    Serial.print("Energi : ");
    Serial.print(energi);
    Serial.println(" kWh");
    Serial.print("Tegangan : ");
    Serial.print(tegangan);
    Serial.println(" V");
    Serial.print("Arus : ");
    Serial.print(arus);
    Serial.println(" A");
    Serial.println(WiFi.localIP());
    Serial.println("");
    if(energi >= 0.5){
      digitalWrite(BUZZER, HIGH);
      delay(500);
      digitalWrite(BUZZER, LOW);
    } else {
      digitalWrite(BUZZER, LOW);
    }
    delay(1000);
    vTaskDelay(60 / portTICK_PERIOD_MS);
  }
}

void displayParameter_Listrik(){
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Vrms :");
  lcd.setCursor(7, 0);
  lcd.print(tegangan);
  lcd.print(" V          ");
  lcd.setCursor(0, 1);
  lcd.print("Irms :");
  lcd.setCursor(7, 1);
  lcd.print(arus);
  lcd.print(" A          ");
  delay(3500);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Daya  :");
  lcd.setCursor(8, 0);
  lcd.print(daya);
  lcd.print(" W          ");
  lcd.setCursor(0, 1);
  lcd.print("Energi:");
  lcd.setCursor(8, 1);
  lcd.print(energi);
  lcd.print(" kWh          ");
  delay(3500);
}

void loop() {
  // put your main code here, to run repeatedly:
  lcd.setCursor (0, 0);
  lcd.print(" Param. Listrik?");
  lcd.setCursor (0, 1);
  lcd.print(" Tempelkan Tag! ");

  // Cek untuk kartu yang baru disisipkan
  if (!mfrc.PICC_IsNewCardPresent())
    return;

  // Jika nomor tag tidak diperoleh
    if (!mfrc.PICC_ReadCardSerial())
    return;

  // Peroleh UID pada tag
  uidTag = "";
  for (byte j = 0; j < mfrc.uid.size; j++) {
    char teks[3];
    sprintf(teks, "%02X", mfrc.uid.uidByte[j]);
    uidTag += teks;
  }

  Serial.print(" UID : ");
  Serial.println(uidTag);

  if(uidTag.substring(0) == "A5200C46") {
    if (isnan(daya) && isnan(energi) && isnan(tegangan) && isnan(arus)) {
      Serial.print("Gagal Membaca Parameter Listrik");
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Gagal Membaca");
      lcd.setCursor(0, 1);
      lcd.print("Param. Listrik!");
    } else {
      displayParameter_Listrik();
      displayParameter_Listrik();
      displayParameter_Listrik();
      displayParameter_Listrik();
      displayParameter_Listrik();
    }
    Serial.println();
    delay(3500);
    lcd.clear();
  } else {
    lcd.clear();
    lcd.setCursor (0, 1);
    lcd.print("Akses ditolak !!");
    delay (1500);
  }
  // Ubah status kartu ACTIVE ke status HALT
  mfrc.PICC_HaltA();
}
```