

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

```
1 //Pengambilan Library Sensor RFID
2 #include <MFRC522.h>
3 #include <SPI.h>
4 //Pengambilan Library LCD
5 #include <LiquidCrystal_I2C.h>
6
7 //Pengambilan Library Untuk Web Server
8 #include "WiFi.h"
9 #include "ESPAsyncWebServer.h"
10 #include "SPIFFS.h"
11
12 //Pengambilan Library Sensor PZEM004T Versi 3.0
13 #include <PZEM004Tv30.h>
14 PZEM004Tv30 pzem(&Serial2);
15
16 //GPIO LED
17 const int ledMerah = 27; //GPIO 27
18 const int ledKuning = 25; //GPIO 25
19 const int ledHijau = 26; //GPIO 26
20
21 //setting PWM properties
22 const int frekuensi = 5000;
23 const int ledChannelKuning = 0;
24 const int resolution = 8;
25
26 //multitask
27 TaskHandle_t Task1;
28 TaskHandle_t Task2;
29
30 //Inisialisasi Aktuator
31 const int relay = 13;
32 const int PIN_RST = 15;
33 const int PIN_SS = 4;
34 const int BUZZER = 2;
35
36 // Pembuatan objek MFRC522
37 String uidTag;
38 MFRC522 mfrc(PIN_SS, PIN_RST);
39
40 float daya, energi, tegangan, arus;
41 LiquidCrystal_I2C lcd(0x27, 16, 2);
42
43 // Replace with your network credentials
44 const char* ssid = "Cendrawasih";
45 const char* password = "cendrawasih";
46
47 // Create AsyncWebServer object on port 80
48 AsyncWebServer server(80);
49
50 String getTegangan() {
51     return String(tegangan);
52 }
53
```

```

54 String getArus(){
55     return String(arus);
56 }
57
58 String getDaya(){
59     return String(daya);
60 }
61
62 String getEnergi(){
63     return String(energi);
64 }
65
66 void setup() {
67     // put your setup code here, to run once:
68     Serial.begin(115200);
69     SPI.begin();
70     mfrc.PCD_Init();
71     pinMode(ledHijau, OUTPUT);
72     pinMode(ledMerah, OUTPUT);
73     pinMode(relay, OUTPUT);
74     pinMode(BUZZER, OUTPUT);
75     lcd.init();
76     lcd.backlight();
77     lcd.blink_on();
78
79     //konfigurasi fungsi LED PWM
80     ledcSetup(ledChannelKuning, frekuensi, resolution);
81
82     //menghubungkan channel ke GPIO untuk bisa dikontrol
83     ledcAttachPin(ledKuning, ledChannelKuning);
84
85     //SPIFFS
86     if (! SPIFFS.begin (true)) {
87         Serial.println ("An Error has occurred while mounting SPIFFS");
88         return;
89     }
90
91     // Connect to Wi-Fi
92     WiFi.begin(ssid, password);
93     while (WiFi.status() != WL_CONNECTED) {
94         delay(1000);
95         Serial.println("Connecting to WiFi..");
96     }
97
98     // Print ESP32 Local IP Address
99     Serial.println(WiFi.localIP());
100
101     // Route for web page
102     server.on ("/", HTTP_GET, [] (AsyncWebServerRequest * request) {
103         request-> send (SPIFFS, "/Webserver_spowcom.html");
104     });
105
106     server.on ("/style.css", HTTP_GET, [] (AsyncWebServerRequest * request) {
107         request-> send (SPIFFS, "/style.css", "text/css");

```

```

108     });
109
110     server.on("/image.png", HTTP_GET, [] (AsyncWebServerRequest *request){
111     request->send(SPIFFS, "/image.png", "image/png");
112     });
113
114     server.on("/timer.png", HTTP_GET, [] (AsyncWebServerRequest *request){
115     request->send(SPIFFS, "/timer.png", "timer/png");
116     });
117
118     server.on ("/chart.js", HTTP_GET, [] (AsyncWebServerRequest * request) {
119     request-> send (SPIFFS, "/chart.js", "text/js");
120     });
121
122     server.on ("/app.js", HTTP_GET, [] (AsyncWebServerRequest * request) {
123     request-> send (SPIFFS, "/app.js", "app/js");
124     });
125
126     server.on ("/tegangan", HTTP_GET, [] (AsyncWebServerRequest * request) {
127     request-> send_P (200, "text / plain", getTegangan().c_str ());
128     });
129
130     server.on ("/arus", HTTP_GET, [] (AsyncWebServerRequest * request) {
131     request-> send_P (200, "text / plain", getArus().c_str ());
132     });
133
134     server.on ("/daya", HTTP_GET, [] (AsyncWebServerRequest * request) {
135     request-> send_P (200, "text / plain", getDaya().c_str ());
136     });
137
138     server.on ("/energi", HTTP_GET, [] (AsyncWebServerRequest * request) {
139     request-> send_P (200, "text / plain", getEnergi().c_str ());
140     });
141
142     // start server
143     server.begin ();
144
145     xTaskCreatePinnedToCore(
146         Task1code,      /* Task function. */
147         "Task1",        /* name of task. */
148         10000,          /* Stack size of task */
149         NULL,           /* parameter of the task */
150         1,              /* priority of the task */
151         &Task1,         /* Task handle to keep track of created task */
152         0);             /* pin task to core 0 */
153
154     xTaskCreatePinnedToCore(
155         Task2code,      /* Task function. */
156         "Task2",        /* name of task. */
157         10000,          /* Stack size of task */
158         NULL,           /* parameter of the task */
159         2,              /* priority of the task */
160         &Task2,         /* Task handle to keep track of created task */
161         1);             /* pin task to core 1 */

```

```

162 }
163
164 void Task1code( void * pvParameters ){
165     Serial.print("Task1 running on core ");
166     Serial.println(xPortGetCoreID());
167
168     for(;;){
169         //menghitung besaran listrik
170         tegangan = pzem.voltage(); //volt
171         arus = pzem.current(); //ampere
172         daya = pzem.power(); //watt
173         energi = pzem.energy(); //kWh
174         vTaskDelay(50 / portTICK_PERIOD_MS);
175
176         if(energi < 0.5){
177             digitalWrite(ledHijau, HIGH);
178             digitalWrite(ledMerah, LOW);
179         } else {
180             digitalWrite(ledHijau, LOW);
181             digitalWrite(ledMerah, HIGH);
182         }
183
184         if(energi >= 0.5 || daya > 1000){
185             digitalWrite(ledMerah, HIGH);
186             for(int dutyCycle = 0; dutyCycle <= 255; dutyCycle++){
187                 // changing the LED brightness with PWM
188                 ledcWrite(ledChannelKuning, dutyCycle);
189                 delay(15);
190             }
191             // menurunkan LED brightness
192             for(int dutyCycle = 255; dutyCycle >= 0; dutyCycle--){
193                 // ubah LED brightness dengan PWM
194                 ledcWrite(ledChannelKuning, dutyCycle);
195                 delay(10);
196             }
197         }
198     }
199 }
200
201 void Task2code( void * pvParameters ){
202     Serial.print("Task2 running on core ");
203     Serial.println(xPortGetCoreID());
204
205     for(;;){
206         Serial.print("Daya : ");
207         Serial.print(daya);
208         Serial.println(" W");
209         Serial.print("Energi : ");
210         Serial.print(energi);
211         Serial.println(" kWh");
212         Serial.print("Tegangan : ");
213         Serial.print(tegangan);
214         Serial.println(" V");
215         Serial.print("Arus : ");

```

```

216     Serial.print(arus);
217     Serial.println(" A");
218     Serial.println(WiFi.localIP());
219     Serial.println("");
220     if(energi >= 0.5){
221         digitalWrite(BUZZER, HIGH);
222         delay(500);
223         digitalWrite(BUZZER, LOW);
224     } else {
225         digitalWrite(BUZZER, LOW);
226     }
227     delay(1000);
228     vTaskDelay(60 / portTICK_PERIOD_MS);
229 }
230 }
231
232 void displayParameter_Listrik(){
233     lcd.clear();
234     lcd.setCursor(0, 0);
235     lcd.print("Vrms :");
236     lcd.setCursor(7, 0);
237     lcd.print(tegangan);
238     lcd.print(" V      ");
239     lcd.setCursor(0, 1);
240     lcd.print("Irms :");
241     lcd.setCursor(7, 1);
242     lcd.print(arus);
243     lcd.print(" A      ");
244     delay(3500);
245     lcd.clear();
246     lcd.setCursor(0, 0);
247     lcd.print("Daya :");
248     lcd.setCursor(8, 0);
249     lcd.print(daya);
250     lcd.print(" W      ");
251     lcd.setCursor(0, 1);
252     lcd.print("Energi:");
253     lcd.setCursor(8, 1);
254     lcd.print(energi);
255     lcd.print(" kWh      ");
256     delay(3500);
257 }
258
259 void loop() {
260     // put your main code here, to run repeatedly:
261     lcd.setCursor (0, 0);
262     lcd.print(" Param. Listrik?");
263     lcd.setCursor (0, 1);
264     lcd.print(" Tempelkan Tag! ");
265
266     // Cek untuk kartu yang baru disisipkan
267     if (!mfrc.PICC_IsNewCardPresent())
268         return;
269

```

```

270 // Jika nomor tag tidak diperoleh
271 if (!mfrc.PICC_ReadCardSerial())
272     return;
273
274 // Peroleh UID pada tag
275 uidTag = "";
276 for (byte j = 0; j < mfrc.uid.size; j++) {
277     char teks[3];
278     sprintf(teks, "%02X", mfrc.uid.uidByte[j]);
279     uidTag += teks;
280 }
281
282 Serial.print(" UID : ");
283 Serial.println(uidTag);
284
285 if(uidTag.substring(0) == "A5200C46") {
286     if (isnan(daya) && isnan(energi) && isnan(tegangan) && isnan(arus)) {
287         Serial.print("Gagal Membaca Parameter Listrik");
288         lcd.clear();
289         lcd.setCursor(0, 0);
290         lcd.print("Gagal Membaca");
291         lcd.setCursor(0, 1);
292         lcd.print("Param. Listrik!");
293     } else {
294         displayParameter_Listrik();
295         displayParameter_Listrik();
296         displayParameter_Listrik();
297         displayParameter_Listrik();
298         displayParameter_Listrik();
299     }
300     Serial.println();
301     delay(3500);
302     lcd.clear();
303 } else {
304     lcd.clear();
305     lcd.setCursor (0, 1);
306     lcd.print("Akses ditolak !!");
307     delay (1500);
308 }
309 // Ubah status kartu ACTIVE ke status HALT
310 mfrc.PICC_HaltA();
311 }

```