|  |  |
| --- | --- |
| Program Smart Power Consumption Monitoring System Using ESP 32 and PZEM004T Sensor | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302  303  304  305  306  307  308  309  310  311 | *//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();  } |