ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Харківський національний університет радіоелектроніки
Кафедра ЕОМ

Квалівікаційна робота
Перший(бакалаврський) рівень

ПРОГРАМНО-АПАРАТНЕ ІОТ-РІШЕННЯ ДЛЯ РОЗУМНОГО КЕРУВАННЯ ЖИВЛЕННЯМ ПРИСТРОЇВ ВІД АКУМУЛЯТОРУ

Здобувач:
Олексій БАТУРІН
КІУКІ-21-5

Керівник:
Яна НІ
ст. викл. кафедри ЕОМ

Мета та завдання роботи

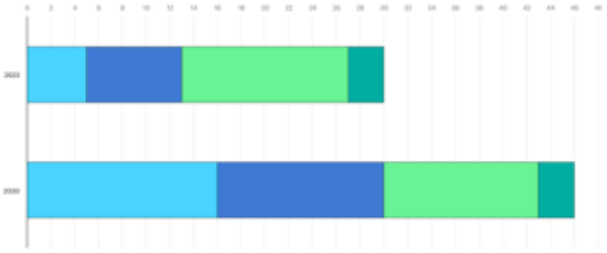
Метою роботи є розробка програмно-апаратного IoT-рішення для ефективного керування енергоспоживанням побутових приладів, підключених до акумулятора сонячної електростанції.

Завдання:

- Зчитувати значення напруги з акумулятору;
- Зчитувати показники потужності домашньої системи;
- Керувати енергоспоживанням пристроїв, що приєднуються;
- Зручний інтерфейс для задання меж;

Актуальність роботи

- Зростання альтернативних джерел енергії
- Нестабільність енергопостачання в Україні
- Оптимізація енергоспоживання
- Енергетична безпека



Графік зростання попиту на відновлювану енергію

Огляд існуючих рішень

- Комерційні інтегровані системи;
- Ринки DIY та open-source;
- Універсальні IoT-платформи;
- Нішеві українські рішення.

OpenEnergyMonitor

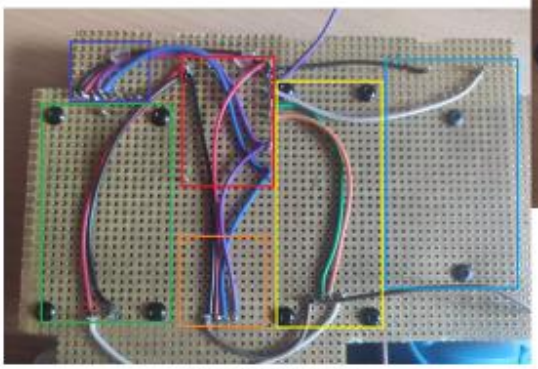
Використані компоненти

- мікроконтролер ESP32;
- датчик напруги INA226;
- перетворючав напруги DC-DC LM 2596;
- вимірювач споживаної енергії PZEM-004T;
- твердотільне реле;
- OLED-дисплей.

Схема пристрою

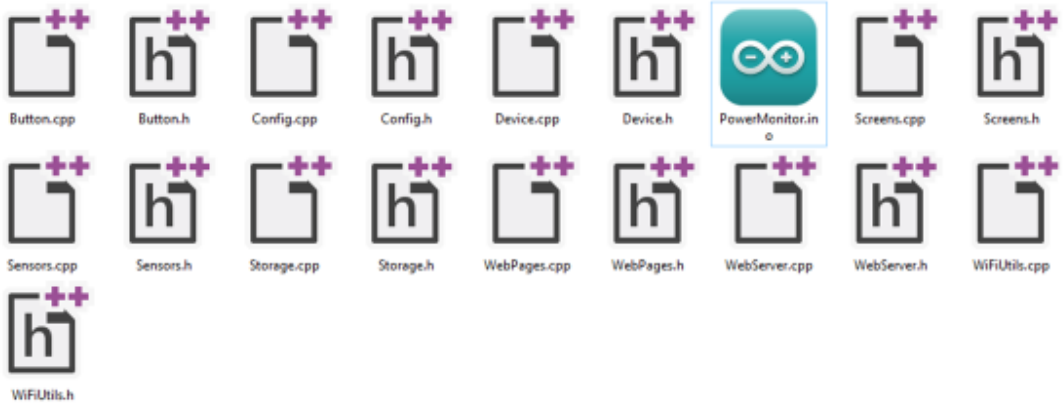
Апаратна частина

Передня сторона



Зворотня сторона

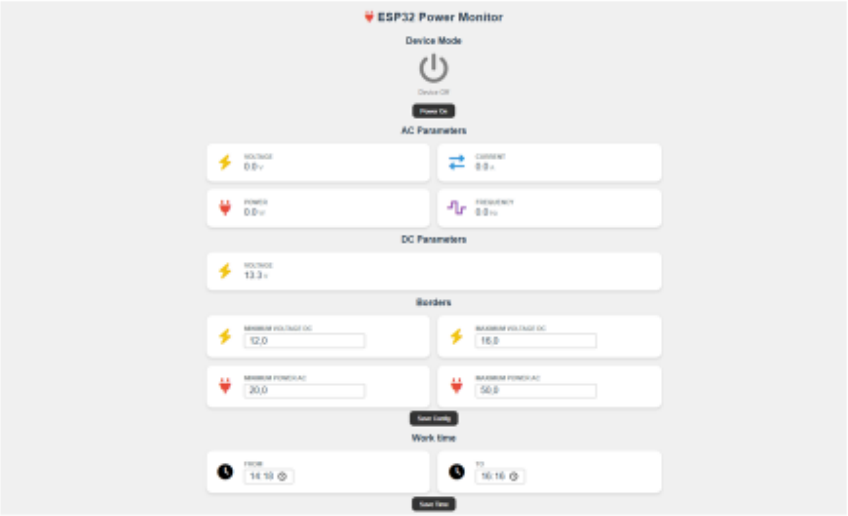
Програмна частина



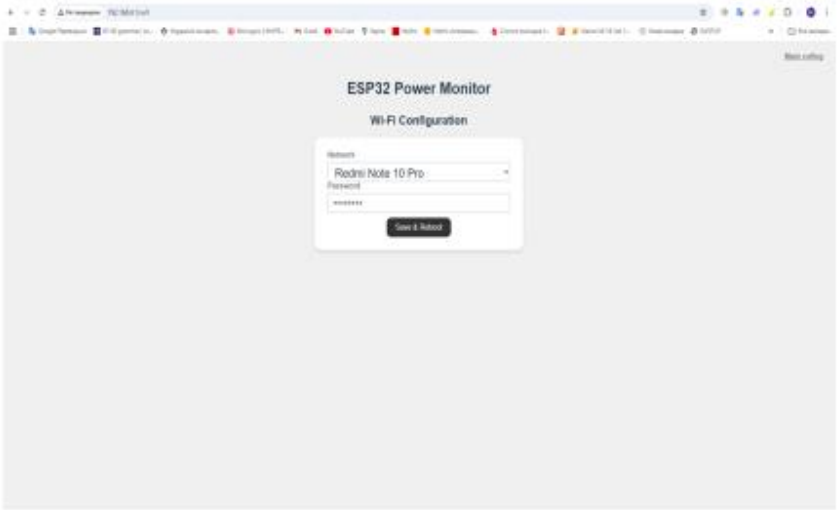
Алгоритм керування навантаженням

- Керування базується на порогових значеннях напруги, потужності та часового інтервалу.
- Після кожного опитування датчиків система обчислює поточні параметри та порівнює їх з встановленими межами. Якщо умови задоволені, реле вмикається; якщо ні — вимикається.
- У разі ручного ввімкнення реле поза межами інтервалу, система активує режим "override", який залишається активним до кінця поточного циклу.

Інтерфейс головної сторінки пристрою



Інтерфейс сторінки для підключення Wi-Fi



Тестування пристрою

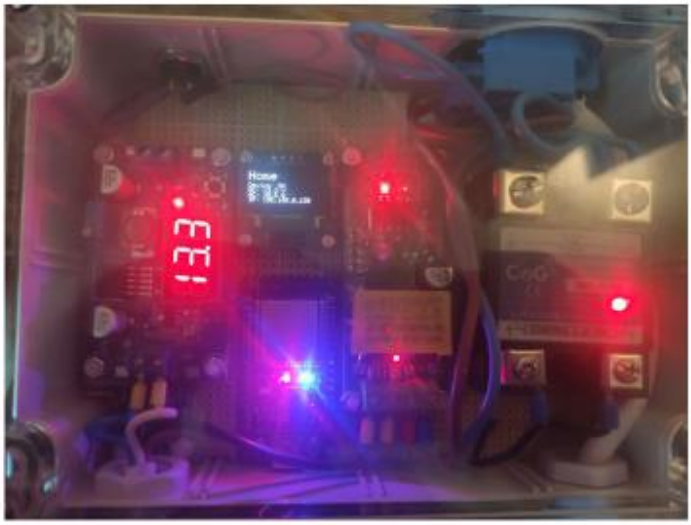
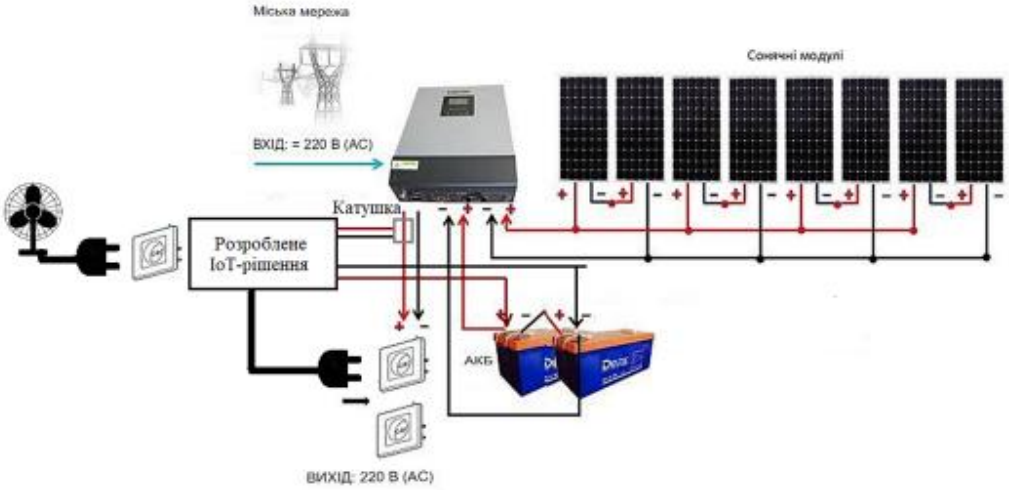


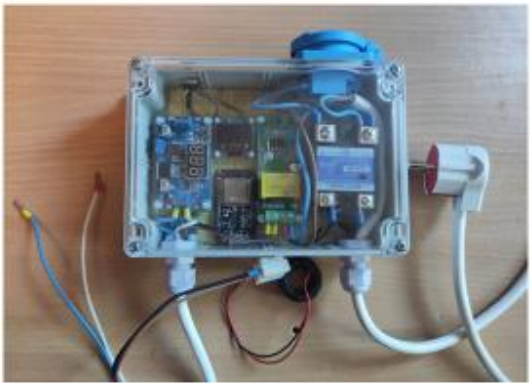
Схема підключення



Демонстрація роботи

Висновки

Розроблене IoT-рішення для керування енергоспоживанням на основі ESP32 ефективно оптимізує використання енергії сонячних електростанцій, забезпечуючи точне вимірювання параметрів, автоматичне включення/вимикання навантажень за пороговими значеннями та можливість віддаленого моніторингу.



ДОДАТОК Б
ПРОГРАМНИЙ КОД ЗАСТОСУНКУ

Б.1 Файл PowerMonitor.ino

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include "Config.h"
#include "Device.h"
#include "Button.h"
#include "Screens.h"
#include "Sensors.h"
#include "Storage.h"
#include "WiFiUtils.h"
#include "WebServer.h"

static Button button(BUTTON_PIN);
static unsigned long lastActivity = 0;
static constexpr unsigned long IDLE_TIMEOUT = 60000UL;

bool inTimeWindow() {
  struct tm tm; if(!getLocalTime(&tm)) return true;
  uint16_t now = tm.tm_hour*60+tm.tm_min,
           from= window.fromH*60+window.fromM,
           to  = window.toH  *60+window.toM;
  return (from<=to) ? (now>=from && now<to) : (now>=from ||
now<to);
}

time_t nextWindowEnd() {
  time_t now=time(nullptr);
  struct tm tmEnd;
  localtime_r(&now,&tmEnd);
  tmEnd.tm_hour=window.toH;
  tmEnd.tm_min=window.toM;
  tmEnd.tm_sec=0;
  time_t end=mktime(&tmEnd);
  if(end<=now) {
    end+=24*3600;
  }
  return end;
}


void updateDeviceState() {
  time_t now=time(nullptr);
  bool inOverride = overrideActive && now<overrideUntil;
```

```
  bool timeOK = inTimeWindow() || inOverride;
  if(device.isOn()) {
    if(!timeOK || powerSystem>maxPowerSystem ||
voltageDC<minVoltageDC){
      device.turnOff(); overrideActive=false;
    }
  }
else {
    if(timeOK && voltageDC>maxVoltageDC &&
powerSystem<minPowerSystem)
      device.turnOn();
  }
}


void setup() {
  Serial.begin(115200);
  Wire.begin();
  if(!display.begin(SSD1306_SWITCHCAPVCC, OLED_ADDR)){
    Serial.println("SSD1306 not found");
    while(1) delay(1);
  }
  display.setTextColor(SSD1306_WHITE);

  device.begin();
  button.begin();

  loadThresholds();
  loadWindow();
  initSensors();
  startWiFi();
  initTime();
  setupWeb();
  WiFi.onEvent(WiFiStationDisconnected,
WiFiEvent_t::ARDUINO_EVENT_WIFI_STA_DISCONNECTED);

  screenManager.add(&_main); screenManager.add(&_sys);
  screenManager.add(&_bor);  screenManager.add(&_wifi);
  screenManager.show(); lastActivity = millis();
}

void loop() {
  readSensors();
  if(button.wasPressed()) { screenManager.next();
lastActivity=millis(); }
  if(millis()-lastActivity > IDLE_TIMEOUT) {
screenManager.reset(); lastActivity=millis(); }
  updateDeviceState();
  screenManager.show();
  delay(50);
}
```

Б.2 Файли Config.h та Config.cpp

```
#pragma once
#include <Arduino.h>
#include <Preferences.h>

#define PZEM_RX_PIN 16
#define PZEM_TX_PIN 17
#define INA226_ADDR 0x44
#define OLED_ADDR   0x3C
#define SCREEN_WIDTH  128
#define SCREEN_HEIGHT 64
#define BUTTON_PIN 15
#define DEVICE_PIN 2

extern const char* ntpServer;
extern const long  gmtOffset_sec;
extern const long  daylightOffset_sec;

extern float powerSystem, voltageSystem, currentSystem,
frequencySystem, voltageDC;
extern String ssid, pass;
extern float minVoltageDC, maxVoltageDC, minPowerSystem,
maxPowerSystem;

struct TimeWindow { uint8_t fromH, fromM, toH, toM; };
extern TimeWindow window;

extern bool  overrideActive;
extern time_t overrideUntil;

#include "Config.h"

const char* ntpServer         = "pool.ntp.org";
const long  gmtOffset_sec     = 7200;
const long  daylightOffset_sec= 3600;

String ssid, pass;

float powerSystem   = 0, voltageSystem = 0, currentSystem = 0,
      energySystem  = 0, frequencySystem = 0, pfSystem    = 0,
      voltageDC     = 0;

float minVoltageDC  = 23.0, maxVoltageDC  = 28.0,
      minPowerSystem= 500.0, maxPowerSystem= 1000.0;

TimeWindow window   = {6,0, 23,0};

bool  overrideActive = false;
time_t overrideUntil = 0;
```

Б.3 Файли Device.h та Device.cpp

```
#pragma once
#include <Arduino.h>

class Device {
  int  pin;
  bool state = false;
  unsigned long lastToggle = 0;
  static constexpr unsigned long LOCK_MS = 5*60*1000UL;
public:
  explicit Device(int p): pin(p) {}
  void begin();
  void turnOn();
  void turnOff();
  void toggle();
  bool isOn() const { return state; }
};
extern Device device;

#include "Device.h"
#include "Config.h"

void Device::begin() { pinMode(pin, OUTPUT); digitalWrite(pin,
LOW); }

void Device::turnOn() {
  if (!state) { digitalWrite(pin, HIGH); state = true; }
}

void Device::turnOff() {
  if (state) { digitalWrite(pin, LOW);  state = false;
lastToggle = millis(); }
}

void Device::toggle() { state ? turnOff() : turnOn(); }

Device device(DEVICE_PIN);
```

Б.4 Файли Button.h та Button.cpp

```
#pragma once
#include <Arduino.h>

class Button {
  int pin;
  bool stable, last;
  unsigned long lastChange;
  const unsigned long debounce;
```

```
public:
  explicit Button(int p, unsigned long d=50): pin(p),
debounce(d) {}
  void begin();
  bool wasPressed();
};
#include "Button.h"
void Button::begin() {
  pinMode(pin, INPUT_PULLUP);
  last = stable = digitalRead(pin);
  lastChange = millis();
}
bool Button::wasPressed() {
  bool r = digitalRead(pin);
  if (r != last) { last = r; lastChange = millis(); }
  if (millis() - lastChange > debounce && r != stable) {
    stable = r; return stable == LOW;
  }
  return false;
}
```

Б.5 Файли Screens.h та Screens.cpp

```
#pragma once
#include <Adafruit_SSD1306.h>
#include "Config.h"
#include "Device.h"

extern Adafruit_SSD1306 display;

class Screen { public: virtual void show() = 0; };

class MainScreen   : public Screen { void show() override; };
class SystemScreen : public Screen { void show() override; };
class BorderScreen : public Screen { void show() override; };
class WifiScreen   : public Screen { void show() override; };

class ScreenManager {
  Screen* list[4]; uint8_t count=0, idx=0;
public:
  void add(Screen* s){ if(count<4) list[count++]=s; }
  void next(){ if(count) { idx=(idx+1)%count; list[idx]->show();
} }
  void reset(){ idx=0; show(); }
  void show(){ if(count) list[idx]->show(); }
};
extern ScreenManager screenManager;
extern MainScreen   _main;
extern SystemScreen _sys;
extern BorderScreen _bor;
extern WifiScreen   _wifi;
```

```
#include "Screens.h"
#include <WiFi.h>

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -
1);
ScreenManager     screenManager;
MainScreen   _main;
SystemScreen _sys;
BorderScreen _bor;
WifiScreen   _wifi;


static void header(const __FlashStringHelper* txt){
  display.clearDisplay();
  display.setTextSize(2);
  display.setCursor(0,0);
  display.print(txt);
  display.setTextSize(1);
}

void MainScreen::show() {
  header(F("Home"));
  display.setCursor(0,20);
  display.printf("Device: %s\n", device.isOn()?"ON":"OFF");
  display.printf("DC: %.1f V\n", voltageDC);
  display.printf("AC: %.1f W\n", powerSystem);
  display.printf("IP: %s\n", WiFi.localIP().toString().c_str());
  display.display();
}

void SystemScreen::show() {
  header(F("AC "));
  display.setCursor(0,20);
  display.printf("U: %.1f V\nI: %.2f A\nP: %.1f W\nF: %.1f
Hz\n",

voltageSystem,currentSystem,powerSystem,frequencySystem);
  display.display();
}

void BorderScreen::show() {
  header(F("Borders"));
  display.setCursor(0,18);
  display.printf("V DC min: %.1f \nV DC max: %.1f \nP AC min:
%.1f \nP AC max: %.1f\n",

minVoltageDC,maxVoltageDC,minPowerSystem,maxPowerSystem);
  char buf[12];
  sprintf(buf,"%02u:%02u -
%02u:%02u",window.fromH,window.fromM,window.toH,window.toM);
  display.printf("Time %s",buf);
  display.display();
}
```

```
void WifiScreen::show() {
  header(F("WIFI"));
  display.setCursor(0,20);
  display.printf("SSID: %s\nIP:   %s",
                 WiFi.SSID().c_str(),
WiFi.localIP().toString().c_str());
  display.display();
}
```

Б.6 Файли Sensors.h та Sensors.cpp

```
#pragma once
#include <PZEM004Tv30.h>
#include <INA226_WE.h>
#include "Config.h"

void   initSensors();
void   readSensors();
extern PZEM004Tv30 pzem;
extern INA226_WE   ina226;
#include "Sensors.h"

PZEM004Tv30 pzem(Serial2, PZEM_RX_PIN, PZEM_TX_PIN);
INA226_WE   ina226(&Wire, INA226_ADDR);

void initSensors() {
  if (!ina226.init()) { Serial.println("INA226 error"); while
(1) delay(1); }
  ina226.setMeasureMode(CONTINUOUS);
}

void readSensors() {
  voltageSystem   = pzem.voltage();
  currentSystem   = pzem.current();
  powerSystem     = pzem.power();
  frequencySystem = pzem.frequency();
  voltageDC       = ina226.getBusVoltage_V() * 0.947;
}
```

Б.7 Файли Storage.h та Storage.cpp

```
#pragma once
void loadThresholds();
void saveThresholds();
void loadWindow();
void saveWindow();
#include "Storage.h"
```

```
#include "Config.h"
#include <Preferences.h>

static Preferences cfg;
void loadThresholds() {
  cfg.begin("limits", true);
  minVoltageDC   = cfg.getFloat("minVDC", minVoltageDC);
  maxVoltageDC   = cfg.getFloat("maxVDC", maxVoltageDC);
  minPowerSystem = cfg.getFloat("minPAC", minPowerSystem);
  maxPowerSystem = cfg.getFloat("maxPAC", maxPowerSystem);
  cfg.end();
}
void saveThresholds() {
  cfg.begin("limits", false);
  cfg.putFloat("minVDC", minVoltageDC);
  cfg.putFloat("maxVDC", maxVoltageDC);
  cfg.putFloat("minPAC", minPowerSystem);
  cfg.putFloat("maxPAC", maxPowerSystem);
  cfg.end();
}
void loadWindow() {
  cfg.begin("limits", true);
  window.fromH = cfg.getUChar("fH", window.fromH);
  window.fromM = cfg.getUChar("fM", window.fromM);
  window.toH   = cfg.getUChar("tH", window.toH);
  window.toM   = cfg.getUChar("tM", window.toM);
  cfg.end();
}
void saveWindow() {
  cfg.begin("limits", false);
  cfg.putUChar("fH", window.fromH); cfg.putUChar("fM",
window.fromM);
  cfg.putUChar("tH", window.toH);   cfg.putUChar("tM",
window.toM);
  cfg.end();
}
```

Б.8 Файли WiFiUtils.h та WiFiUtils.cpp

```
#pragma once
#include <WiFi.h>

bool startWiFi();
void initTime();
void WiFiStationDisconnected(WiFiEvent_t event, WiFiEventInfo_t
info);
#include "WiFiUtils.h"
#include "Config.h"
#include <Preferences.h>

static Preferences wifiPrefs;
```

```
static const char* AP_SSID = "Esp32_PowerMonitor";
static const char* AP_PASS = "12345678";

bool startWiFi() {
  wifiPrefs.begin("wifi", true);
  ssid = wifiPrefs.getString("ssid");
  pass = wifiPrefs.getString("pass");
  wifiPrefs.end();

  if (ssid.length()) {
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid.c_str(), pass.c_str());
    unsigned long t0 = millis();
    while (WiFi.status()!=WL_CONNECTED && millis()-t0<10000)
delay(100);
  }
  if (WiFi.status()!=WL_CONNECTED) {
    WiFi.mode(WIFI_AP);
    WiFi.softAP(AP_SSID, AP_PASS);
  }
  return WiFi.getMode()==WIFI_STA;
}

void initTime() {
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
  while (!time(nullptr)) { Serial.print('.'); delay(500); }
  Serial.println();
}

void WiFiStationDisconnected(WiFiEvent_t event, WiFiEventInfo_t
info){
  WiFi.begin(ssid, pass);
}
```

Б.9 Файли WebPages.h та WebPages.cpp

```
#pragma once
#include <Arduino.h>
extern const char index_html[] PROGMEM;
extern const char wifi_html[]  PROGMEM;
String processor(const String& var);
#include "WebPages.h"
#include "Config.h"
#include "Device.h"
#include <WiFi.h>

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML>
<html lang="en">
<head>
  <title>ESP32 Power Monitor</title>
```

```
  <meta name="viewport" content="width=device-width, initial-
scale=1">
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0/css/all.min.css">
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 20px;
      background-color: #f0f0f0;
    }
    .grid {
      display: grid;
      grid-template-columns: repeat(auto-fit, minmax(400px,
1fr));
      gap: 20px;
      max-width: 1200px;
      margin: 0 auto;
    }
    .card {
      background: white;
      border-radius: 15px;
      padding: 25px;
      box-shadow: 0 4px 8px rgba(0,0,0,0.1);
      display: flex;
      align-items: center;
      text-align: left;
    }
    .icon {
      font-size: 40px;
      margin-right: 25px;
      min-width: 50px;
      text-align: center;
    }
    .content {
      display: flex;
      flex-direction: column;
    }
    .label {
      font-size: 16px;
      color: #7f8c8d;
      margin-bottom: 5px;
      font-weight: bold;
    }

    .device-status {
    display: flex;
    flex-direction: column;
    align-items: center;
    text-align: center;
    font-family: 'Segoe UI', sans-serif;
    transition: color 0.3s ease;
```

```
margin-bottom: 10px;
}

.power-icon {
font-size: 5rem;
transition: transform 0.3s ease;
}

.status-label {
margin-top: 8px;
font-size: 1.1rem;
}

.device-status.on {
color: limegreen;
}
.device-status.off {
color: gray;
}

.button-container {
display: flex;
justify-content: center;
}

.Button {
background-color: #333;
color: white;
border: none;
align-content: center;
padding: 10px 20px;
font-size: 1rem;
margin-top: 10px;
border-radius: 10px;
cursor: pointer;
transition: background 0.3s;
}

.Button:hover {
background-color: #555;
}

#toggleButton:hover {
background-color: #555;
}

.text-field__input {
  display: block;
  width: 100%;
  height: calc(2.25rem + 2px);
  padding: 0.375rem 0.75rem;
  font-family: inherit;
  font-size: 24px;
```

```
      color: #2c3e50;
      font-weight: 400;
      line-height: 1.5;
      background-color: #fff;
      background-clip: padding-box;
      border: 1px solid #bdbdbd;
      box-sizing: border-box;
      border-radius: 0.25rem;
      transition: border-color 0.15s ease-in-out, box-shadow
0.15s ease-in-out;
    }

    .value {
      font-size: 24px;
      color: #2c3e50;
      display: flex;
      align-items: baseline;
    }
    .unit {
      font-size: 16px;
      color: #95a5a6;
      margin-left: 5px;
    }

    .fa-bolt { color: #f1c40f; }
    .fa-exchange-alt { color: #3498db; }
    .fa-plug { color: #e74c3c; }
    .fa-chart-line { color: #2ecc71; }
    .fa-wave-square { color: #9b59b6; }
    .fa-percent { color: #e67e22; }

    h1 {
      text-align: center;
      margin: 30px 0;
      color: #2c3e50;
    }
    h2 {
      text-align: center;
      margin: 20px 0;
      color: #2c3e50;
    }
  </style>
  <script>
    let minVoltageDC = 0.0;
    let maxVoltageDC = 0.0;
    let minPowerSystem = 0.0;
    let maxPowerSystem = 0.0;

    function updateUI(status) {
      const statusDiv = document.getElementById('deviceStatus');
      const toggleBtn = document.getElementById('toggleButton');

      statusDiv.classList.toggle('on', status);
```

```
      statusDiv.classList.toggle('off', !status);
      statusDiv.querySelector('.status-label').textContent =
status ? 'Device On' : 'Device Off';
      toggleBtn.textContent = status ? 'Power Off' : 'Power On';
    }

    function updateData() {
      var xhttp = new XMLHttpRequest();
      xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
          var data = JSON.parse(this.responseText);
          document.getElementById('voltageSystem').innerHTML =
data.voltageSystem + '<span class="unit">V</span>';
          document.getElementById('currentSystem').innerHTML =
data.currentSystem + '<span class="unit">A</span>';
          document.getElementById('powerSystem').innerHTML =
data.powerSystem + '<span class="unit">W</span>';
          document.getElementById('frequencySystem').innerHTML =
data.frequencySystem + '<span class="unit">Hz</span>';
          document.getElementById('voltageDC').innerHTML =
data.voltageDC + '<span class="unit">V</span>';
          updateUI(data.statusDevice);
        }
      };
      xhttp.open("GET", "/data", true);
      xhttp.send();
    }

    function updateBorder() {
      var xhttp = new XMLHttpRequest();
      xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
          var data = JSON.parse(this.responseText);
          ({
            minVoltageDC,
            maxVoltageDC,
            minPowerSystem,
            maxPowerSystem
          } = data);

          document.getElementById('minV').value = minVoltageDC;
          document.getElementById('maxV').value = maxVoltageDC;
          document.getElementById('minP').value =
minPowerSystem;
          document.getElementById('maxP').value =
maxPowerSystem;
        }
      };
      xhttp.open("GET", "/borders", true);
      xhttp.send();
    }

    function handleToggleClick () {
```

```
      fetch('/toggle', { method: 'POST' })
        .then(r => r.json())
        .then(data => {
          updateUI(data.statusDevice);
        })
        .catch(console.error);
    }

    function num (id) {
      return
parseFloat(document.getElementById(id).value.replace(',', '.'));
    }

    function saveBorders() {
      /* собираем значения и кодируем как form-urlencoded */
      const body = new URLSearchParams({
        minV: num('minV'),
        maxV: num('maxV'),
        minP: num('minP'),
        maxP: num('maxP')
      });

      fetch('/borders', {
        method: 'POST',
        headers: {'Content-Type':'application/x-www-form-
urlencoded'},
        body
      })
      .then(r => r.json())
      .then(data => {
        ({minVoltageDC, maxVoltageDC, minPowerSystem,
maxPowerSystem} = data);
        updateBorder();
      })
      .catch(console.error);
    }

    function updateWindow() {
      fetch('/window')
        .then(r => r.json())
        .then(w => {
          document.getElementById('from').value = w.from;
          document.getElementById('to').value   = w.to;
        });
    }

    function saveWindow() {
      const body = new URLSearchParams({
        from: document.getElementById('from').value,
        to:   document.getElementById('to').value
      });
      fetch('/window', {method:'POST',
                        headers:{'Content-Type':'application/x-
```

```
www-form-urlencoded'},
                        body})
        .then(()=>updateWindow());
    }

    setInterval(updateData, 1000);

    window.addEventListener('load', () => {
      updateData();
      updateBorder();
      updateWindow();
    });

    document.addEventListener('DOMContentLoaded', () => {

document.getElementById('toggleButton').addEventListener('click'
, handleToggleClick);

document.getElementById('saveButton').addEventListener('click',
saveBorders);

document.getElementById('saveTime').addEventListener('click',
saveWindow);
  });
  </script>
</head>
<body>
  <header style="display:flex;justify-content:flex-end;padding:0
10px;">
    <a href="/wifi" class="label">Wi-Fi setup</a>
  </header>

  <h1><i class="fas fa-plug"></i> ESP32 Power Monitor</h1>

  <h2> Device Mode</h2>
  <div id="deviceStatus" class="device-status off">
    <i class="fas fa-power-off power-icon"></i>
    <div class="status-label">Device Off</div>
  </div>

  <div class="button-container">
    <button id="toggleButton" class="Button">Power On</button>
  </div>

  <h2> AC Parameters</h2>
  <div class="grid">
    <div class="card">
      <i class="fas fa-bolt icon"></i>
      <div class="content">
        <div class="label">VOLTAGE</div>
        <div class="value" id="voltageSystem">%VOLTAGE AC%<span
class="unit">V</span></div>
      </div>
```

```
    </div>
    <div class="card">
      <i class="fas fa-exchange-alt icon"></i>
      <div class="content">
        <div class="label">CURRENT</div>
        <div class="value" id="currentSystem">%CURRENT AC%<span
class="unit">A</span></div>
      </div>
    </div>
    <div class="card">
      <i class="fas fa-plug icon"></i>
      <div class="content">
        <div class="label">POWER</div>
        <div class="value" id="powerSystem">%POWER AC%<span
class="unit">W</span></div>
      </div>
    </div>
    <div class="card">
      <i class="fas fa-wave-square icon"></i>
      <div class="content">
        <div class="label">FREQUENCY</div>
        <div class="value" id="frequencySystem">%FREQUENCY
AC%<span class="unit">Hz</span></div>
      </div>
    </div>
  </div>

  <h2> DC Parameters</h2>
  <div class="grid">
    <div class="card">
      <i class="fas fa-bolt icon"></i>
      <div class="content">
        <div class="label">VOLTAGE</div>
        <div class="value" id="voltageDC">%VOLTAGE DC%<span
class="unit">V</span></div>
        </div>
    </div>
  </div>

  <h2> Borders </h2>
  <div class="grid">
    <div class="card">
    <i class="fas fa-bolt icon"></i>
      <div class="content">
        <label class="label">MINIMUM VOLTAGE DC</label>
        <input class="text-field__input" id="minV"
type="number" step="0.1" value="%minVoltageDC%">
      </div>
    </div>

    <div class="card">
    <i class="fas fa-bolt icon"></i>
      <div class="content">
```

```
        <label class="label">MAXIMUM VOLTAGE DC</label>
        <input class="text-field__input" id="maxV"
type="number" step="0.1" value="%maxVoltageDC%">
      </div>
    </div>

    <div class="card">
    <i class="fas fa-plug icon"></i>
      <div class="content">
        <label class="label">MINIMUM POWER AC</label>
        <input class="text-field__input" id="minP"
type="number" step="0.5" class="value" value="%minPowerSystem%">
      </div>
    </div>

    <div class="card">
    <i class="fas fa-plug icon"></i>
      <div class="content">
        <label class="label">MAXIMUM POWER AC</label>
        <input class="text-field__input" id="maxP"
type="number" step="0.5" class="value" value="%maxPowerSystem%">
      </div>
    </div>
  </div>

  <div class="button-container">
    <button id="saveButton" class="Button">Save Config</button>
  </div>

  <h2> Work time </h2>
  <div class="grid">
    <div class="card">
    <i class="fas fa-clock icon"></i>
      <div class="content">
        <label class="label">FROM</label>
        <input id="from" type="time" class="text-field__input"
value="%from%">
      </div>
    </div>
    <div class="card">
    <i class="fas fa-clock icon"></i>
      <div class="content">
        <label class="label">TO</label>
        <input id="to"   type="time" class="text-field__input"
value="%to%">
      </div>
    </div>
  </div>

  <div class="button-container">
    <button id="saveTime" class="Button">Save Time</button>
  </div>
```

```
</body>
</html>
)rawliteral";

const char wifi_html[] PROGMEM = R"rawliteral(
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>ESP32 Wi-Fi Setup</title>
  <meta name="viewport" content="width=device-width, initial-
scale=1" />
  <link
    rel="stylesheet"
    href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0/css/all.min.css"
  />
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 20px;
      background-color: #f0f0f0;
    }
    .card {
      background: #fff;
      border-radius: 15px;
      padding: 25px 30px;
      margin: 0 auto;
      max-width: 420px;
      box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    }

    .label {
      font-size: 16px;
      color: #7f8c8d;
      margin-bottom: 5px;
      font-weight: bold;
    }
    .text-field__select,
    .text-field__input {
      display: block;
      width: 100%;
      height: calc(2.25rem + 2px);
      padding: 0.375rem 0.75rem;
      font-family: inherit;
      font-size: 24px;
      color: #2c3e50;
      font-weight: 400;
      line-height: 1.5;
      background-color: #fff;
      background-clip: padding-box;
```

```
      border: 1px solid #bdbdbd;
      border-radius: 0.25rem;
      box-sizing: border-box;
      transition: border-color 0.15s ease-in-out, box-shadow
0.15s ease-in-out;
    }
    .content {
      display: flex;
      flex-direction: column;
    }

    .button-container {
    display: flex;
    justify-content: center;
    }
    .Button {
    background-color: #333;
    color: white;
    border: none;
    align-content: center;
    padding: 10px 20px;
    font-size: 1rem;
    margin-top: 10px;
    border-radius: 10px;
    cursor: pointer;
    transition: background 0.3s;
    }

    .Button:hover {
    background-color: #555;
    }
    h1 {
      text-align: center;
      margin: 30px 0;
      color: #2c3e50;
    }
    h2 {
      text-align: center;
      margin: 20px 0;
      color: #2c3e50;
    }

    .fa-plug { color: #e74c3c; }

  </style>
</head>
<script>
let ssidSel;

window.addEventListener('DOMContentLoaded', () => {
  ssidSel = document.getElementById('ssid');
  scan();
});
```

```
async function scan() {
  try {
    const res  = await fetch('/scan');
    const list = await res.json();
    ssidSel.innerHTML = list.map(s =>
`<option>${s}</option>`).join('');
  } catch (e) {
    console.error(e);
  }
}

document.getElementById('wifiForm').addEventListener('submit',
async e => {
  e.preventDefault();
  const formData = new FormData(e.target);
  await fetch('/save', { method: 'POST', body: formData });
});
</script>
<body>

  <header style="display:flex;justify-content:flex-end;padding:0
10px;">
    <a href="/" class="label">Main cofing</a>
  </header>

  <h1><i class="fas fa-plug"></i> ESP32 Power Monitor</h1>
  <h2> Wi-Fi Configuration</h2>

  <form class="card" id="wifiForm" action="/save" method="post">
    <div class="content">
      <label class="label" for="ssid"
        > Network</label
      >
      <select name="ssid" id="ssid" class="text-field__select"
required></select>
    </div>

    <div class="content">
      <label class="label" for="pass"> Password</label>
      <input type="password" name="pass" id="pass" class="text-
field__input" placeholder="••••••••"
       />
    </div>

    <div class="button-container">
      <button type="submit" class="Button"> Save & Reboot
</button>
    </div>

  </form>
</body>
</html>
```

```
)rawliteral";

String processor(const String& var) {
  if(var=="VOLTAGE AC")      return isnan(voltageSystem) ?
"Error":String(voltageSystem,1);
  else if(var=="CURRENT AC") return isnan(currentSystem) ?
"Error":String(currentSystem,1);
  else if(var=="POWER AC")   return isnan(powerSystem)   ?
"Error":String(powerSystem,1);
  else if(var=="FREQUENCY AC") return
isnan(frequencySystem)?"Error":String(frequencySystem,1);
  else if(var=="VOLTAGE DC") return isnan(voltageDC) ?
"Error":String(voltageDC,1);
  else if(var=="from" || var=="to") {
    char buf[6]; uint8_t h =
(var=="from")?window.fromH:window.toH;
    uint8_t m = (var=="from")?window.fromM:window.toM;
    sprintf(buf,"%02u:%02u",h,m); return String(buf);
  }
  return String();
}
```

Б.10 Файли WebServer.h та WebServer.cpp

```
#pragma once
#include <ESPAsyncWebServer.h>
void setupWeb();
#include "WebServer.h"
#include "WebPages.h"
#include "Device.h"
#include "Config.h"
#include "Storage.h"
#include <WiFi.h>

AsyncWebServer server(80);

extern bool  inTimeWindow();
extern time_t nextWindowEnd();

void setupWeb() {

  server.on("/", HTTP_GET, [](auto *req){
    req->send_P(200,"text/html",index_html,processor);
  });

  server.on("/data", HTTP_GET, [](auto *req){
    String j =
String("{\"statusDevice\":")+(device.isOn()?"true":"false")+
      ",\"voltageSystem\":\""+String(voltageSystem,1)+
      "\",\"currentSystem\":\""+String(currentSystem,1)+
      "\",\"powerSystem\":\""+String(powerSystem,1)+
```

```
      "\",\"frequencySystem\":\""+String(frequencySystem,1)+
      "\",\"voltageDC\":\""+String(voltageDC,1)+"\"}";
    req->send(200,"application/json",j);
  });

  server.on("/toggle", HTTP_POST, [](auto *req){
    if(device.isOn()){ device.turnOff(); overrideActive=false; }
    else {
      device.turnOn();
      if(!inTimeWindow()){ overrideActive=true;
overrideUntil=nextWindowEnd(); }
    }
    req->send(200,"application/json",

String("{\"statusDevice\":")+(device.isOn()?"true":"false")+"}")
;
  });

  server.on("/borders", HTTP_GET, [](auto *r){
    r->send(200,"application/json",
      String("{\"minVoltageDC\":\"")+String(minVoltageDC,1)+
      "\",\"maxVoltageDC\":\""+String(maxVoltageDC,1)+
      "\",\"minPowerSystem\":\""+String(minPowerSystem,1)+

"\",\"maxPowerSystem\":\""+String(maxPowerSystem,1)+"\"}");
  });

  server.on("/borders", HTTP_POST, [](auto *r){
    auto setF=[&](const char* n,float &v){
      if(r->hasParam(n,true)) v = r->getParam(n,true)-
>value().toFloat();
    };
    setF("minV",minVoltageDC); setF("maxV",maxVoltageDC);
    setF("minP",minPowerSystem); setF("maxP",maxPowerSystem);
    saveThresholds();
    r->send(200,"application/json",
      String("{\"minVoltageDC\":\"")+String(minVoltageDC,1)+
      "\",\"maxVoltageDC\":\""+String(maxVoltageDC,1)+
      "\",\"minPowerSystem\":\""+String(minPowerSystem,1)+

"\",\"maxPowerSystem\":\""+String(maxPowerSystem,1)+"\"}");
  });

  server.on("/window", HTTP_GET, [](AsyncWebServerRequest *req){
    char buf[6];
    sprintf(buf,"%02u:%02u",window.fromH,window.fromM);
    String json = "{\"from\":\""; json += buf; json +=
"\",\"to\":\"";
    sprintf(buf,"%02u:%02u",window.toH,window.toM);
    json += buf; json += "\"}";
    req->send(200,"application/json",json);
  });
```

```
  server.on("/window", HTTP_POST, [](auto *r){
    auto getHM=[&](const char* p,uint8_t &h,uint8_t &m){
      if(r->hasParam(p,true)){ String v=r->getParam(p,true)-
>value();
       h=v.substring(0,2).toInt(); m=v.substring(3,5).toInt();
}
    };
    getHM("from",window.fromH,window.fromM);
    getHM("to",  window.toH,  window.toM);
    saveWindow(); r->send(200,"text/plain","OK");
  });

  server.on("/wifi", HTTP_GET, [](AsyncWebServerRequest *req){
    req->send_P(200, "text/html", wifi_html);
  });

  server.on("/scan", HTTP_GET, [](AsyncWebServerRequest *req){
    int16_t n = WiFi.scanNetworks(false, false);
    String  json = "[";
    for (int i = 0; i < n; ++i) {
      if (i) json += ',';
      json += '"' + WiFi.SSID(i) + '"';
    }
    json += "]";
    WiFi.scanDelete();
    req->send(200, "application/json", json);
  });

  server.on("/save", HTTP_POST, [](auto *r){
    if(r->hasParam("ssid",true)&&r->hasParam("pass",true)){
      Preferences p; p.begin("wifi",false);
      p.putString("ssid",r->getParam("ssid",true)->value());
      p.putString("pass",r->getParam("pass",true)->value());
      p.end(); r->send(200,"text/plain","OK"); delay(300);
ESP.restart();
    } else r->send(400,"text/plain","Bad request");
  });

  server.begin();
}
```