УДК 004.72:004.891:621.31
DOI https://doi.org/10.26661/2786-6254-2024-2-06

РОЗРОБКА АРХІТЕКТУРИ IOT СИСТЕМИ ВІДДАЛЕНОГО УПРАВЛІННЯ ПРИСТРОЯМИ З ВИКОРИСТАННЯМ ПЛАТФОРМИ ESP32

Драєвський Д. С.
викладач
Відокремлений структурний підрозділ «Економіко-правничий фаховий коледж Запорізького національного університету»
вул. Університетська, 66, Запоріжжя, Україна
orcid.org/0009-0008-3810-6595
draevskdmitry@gmail.com

Мухін В. В.
кандидат технічних наук, доцент,
доцент кафедри програмної інженерії
Запорізький національний університет
вул. Університетська, 66, Запоріжжя, Україна
orcid.org/0009-0007-0270-8239
comentssno@gmail.com

Ключові слова: *інтернет речей, архітектура системи, мікроконтролер, проєктування систем.*

Один з напрямів розвитку сучасного виробництва полягає у створенні так званих «розумних» пристроїв. Такі пристрої є основою новітніх концепцій Індустрії 4.0 та 5.0. Ці нові підходи з'явились в результаті розвитку ідеї інтернету речей. Інтернет речей представляє революційну технологію, яка з'єднує фізичні об'єкти, датчиками, мікроконтролери та програмне забезпечення для збору даних та керування пристроями. Така інтеграція «розумних» пристроїв в життя та виробництво дозволяє автоматизувати рутинні завдання, що призводить до значного скорочення витрат і споживання енергії, тим самим сприяючи сталому розвитку та економічному зростанню. Основні сфери застосування IoT охоплюють різні сектори, включаючи промисловість, розумні міста, агропромисловість і логістику. Важливим напрямком розробки IoT є розробка оптимальної системної архітектури, яка підвищує безпеку, масштабованість і стійкість. Ефективні архітектури повинні включати шифрування та автентифікацію для захисту конфіденційних даних, забезпечення безперебійної інтеграції нових пристроїв і підтримки надійності в критично важливих умовах, зокрема, при раптовому зникненні електроживлення. Дослідження спрямоване на розробку оптимальної архітектури IoT системи і програмної реалізації для керування пристроями за допомогою платформи ESP32, вирішення таких завдань, як розробка схеми для автоматичної системи керування освітленням і впровадження програмного забезпечення для шаблонів керування освітленням. Проєкт включає такі IoT пристрої, як мікроконтролер ESP-WROOM-32, електромеханічні реле, світлодіодні лампи, детектор напруги з оптопарою, підвищуючий імпульсний перетворювач напруги; діоди Шотткі, конденсатор. Запропоновано алгоритм визначення стану системи із використанням електромеханічних реле та детектора напруги. Продемонстровано стійкість розробленої системи при раптовому зникненні енергозабезпечення.

DEVELOPMENT OF IOT ARCHITECTURE OF REMOTE CONTROL SYSTEM OF DEVICES USING THE ESP32 PLATFORM

Draevskii D. S.
Lecturer
Separate Structural Subdivision "Economic and Legal Vocational College of Zaporizhzhia National University"
Universytetska str., 66, Zaporizhzhia, Ukraine
orcid.org/0009-0008-3810-6595
draevskdmitry@gmail.com

Mukhin V.V.
Candidate of Technical Sciences, Associate Professor,
Professor at the Department of Software Engineering
Zaporizhzhia National University,
Universytetska str., 66, Zaporizhzhia, Ukraine
orcid.org/0009-0007-0270-8239
comentssno@gmail.com

Key words: *Internet of Things, system architecture, microcontroller, system design.*

One of the key directions in modern manufacturing development is the creation of so-called "smart" devices, which form the basis of cutting-edge Industry 4.0 and 5.0 concepts. These new approaches have emerged as a result of the evolving Internet of Things (IoT) technology, which connects physical objects equipped with sensors, microcontrollers, and software to collect data and control devices. Integrating smart devices into everyday life and production enables the automation of routine tasks, leading to significant cost and energy reductions, thus promoting sustainable development and economic growth. The primary IoT application areas span diverse sectors, including industry, smart cities, agriculture, and logistics. A critical direction in IoT development is designing an optimal system architecture that enhances security, scalability, and resilience. Effective architectures should incorporate encryption and authentication to protect sensitive data, ensure seamless integration of new devices, and maintain reliability in critical conditions, particularly during unexpected power outages. This study focuses on developing an optimal IoT system architecture and software implementation for device management using the ESP32 platform, addressing tasks such as designing an automatic lighting control system and implementing control software. The project involves IoT devices like the ESP-WROOM-32 microcontroller, electromechanical relays, LED lamps, a voltage detector with an optocoupler, a boost converter, Schottky diodes, and a capacitor. An algorithm for system state determination using electromechanical relays and a voltage detector is proposed. The system's resilience in case of sudden power loss is demonstrated.

Вступ. Інтернет речей (Internet of Things, IoT) – це поширена технологія, що передбачає з'єднання фізичних об'єктів, які мають вбудовані сенсори, програмне забезпечення та інші технології, з мережею для обміну даними між собою та іншими системами через Інтернет [1]. Автоматизація рутинних завдань з використанням інтелектуальних пристроїв дозволяє значно знизити витрати та енергоспоживання в промисловості та побуті, що сприяє сталому розвитку та економічному зростанню. Окремі напрями застосування IoT пов'язані із промисловістю, розвитком розумних міст, агропромисловістю, транспортом та логістикою тощо. Одним з важливих напрямів розвитку IoT систем є проєктування та імплементація оптимальних архітектур, що могли б враховувати специфіку застосування у різних галузях.

Ефективна архітектура пристроїв інтернету речей та відповідного програмного забезпечення безпосередньо впливає на забезпечення безпеки, масштабованості та стійкості систем IoT. Зокрема, така архітектура повинна включати

засоби шифрування та автентифікації для захисту конфіденційних даних, зібраних з кількох підключених пристроїв. Масштабованість визначає здатність системи автоматично обробляти дані з нових пристроїв без необхідності змінювати програмний код. Здатність швидко відновлюватися після потенційних збоїв, забезпечуючи надійність важливих для життя та безпеки даних і додатків визначає стійкість систем IoT. Така оптимальна архітектура сприяє взаємодії різних пристроїв, зниженому енергоспоживанню, ефективній та безпечній обробці даних, тобто ключовим аспектам впровадження додатків Інтернету речей у різних галузях.

Об'єктом дослідження є процес розробки систем інтернету речей.

Метою даної роботи є розробка оптимальної архітектури системи та застосунку інтернету речей для керування пристроями із використанням платформи ESP32 [1].

Для досягнення поставленої мети необхідно розв'язати такі задачі: спроєктувати принципову схему пристроїв IoT на прикладі системи автоматичного керування світлом; розробити програмну реалізацію заданих шаблонів керування світлом.

Огляд літератури. Оглядова стаття [2] містить вичерпний огляд архітектур, технологій, протоколів і форматів даних, які найчастіше використовуються на сучасних платформах IoT. На цій основі було проведено порівняльний аналіз найпопулярніших відкритих платформ IoT.

У [3] були розглянуті типові архітектури систем Інтернету речей: трирівнева, сервіс-орієнтована архітектура та архітектура проміжного програмного забезпечення. Розглянуто основні елементи та особливості реалізації цих підходів.

У роботі [4] розглянуто сучасний стан розвитку різних архітектур IoT. У статті наведено детальну класифікацію систем IoT, зокрема: сумісність, масштабованість, безпека та енергоефективність. Крім того, було розглянуто важливість технологій блокчейну та великих даних та їх аналіз по відношенню до IoT.

Робота [5] містить огляд 167 публікацій. Розглянуто останні тенденції, поточний стан, останні розробки, виклики, безпеку, конфіденційність, застосування IoT та майбутні напрямки досліджень.

Систематичне дослідження сучасної архітектури IoT та основних напрямків досліджень представлено в [6]. Крім того, представлено класифікацію технічних проблем та специфіку впровадження систем IoT відповідно до включених досліджень. Ця модель класифікації може слугувати еталоном у майбутніх роботах.

Стаття [7] присвячена дослідженню проблеми експоненціального зростання обсягу даних, що генеруються системами IoT. Враховуються такі особливості систем IoT, як обмеження ресурсів і енергії, неоднорідність пристроїв і даних, а також масштабованість. У таких інфраструктурах пропонується впроваджувати інтелектуальні засоби аналізу даних. Розглядаються особливості IoT систем на основі програмно визначеної мережі. Проведено детальне дослідження та критичний аналіз попередніх робіт, пов'язаних з аналізом даних IoT у програмно визначених мережах. Виконано критичних аналіз попередніх досліджень та запропоновано майбутні напрями розробок у цій галузі.

В [8] розглянуто методику створення проєктів для навчальних курсів на основі платформи ESP-32. Детально описано проєкт системи моніторингу вуликів та пристрою моніторингу ферментації вина.

У роботі [9] запропоновано систему освітлення розумної кімнати із використанням методів нечіткої логіки на основі ESP32. Завдяки дизайну системи, вона може контролювати яскравість світло із різними заздалегідь визначеними рівнями. Продемонстровано, що розроблене IoT рішення може зменшити споживання електроенергії до 70%.

Дослідження спрямоване на комплексну оцінку та характеристику повнофункціональної сенсорної мережі, побудованої на основі мікроконтролера ESP32 виконано у [10]. Оцінка охоплює три ключові аспекти: продуктивність мережі, вбудований периферійний пристрій на основі ESP32 і систему збору даних.

У [11] аналізується можливість використання мікроконтролеру ESP32 із вбудованою камерою для завдань класифікації зображень за допомогою згорткової нейронної мережі. Виконано обчислювальні експерименти з різною тривалістю фотозйомки та подальшої обробки зображенть. Розгорнуто згортову нейронну мережу, попередньо навчену на іншому пристрої, із застосуванням архітектури MobileNet на мікроконтролері. Продемонстровано, що потужності ESP32 достатньо для одночасної роботи як камери, так і згорткової нейронної мережі.

Отже, з аналізу літератури можна зробити висновок, про перспективність розробки узагальнених підходів до проєктування та імплементації архітектур IoT систем. Актуальним напрямом є врахування в таких архітектурах особливостей умов експлуатації та специфічних сценаріїв роботи IoT систем.

Методи. Особливості побудови архітектури IoT пристроїв та відповідного програмного забезпечення розглянуто на прикладі системи автоматичного керування світлом.

Для розробки тестового проєкту використано пристрої, які є загальновживаними для IoT, а отже, не вимагають специфічних налаштувань.

Передбачено можливість задавати різні сценарії керування світлом. Система повинна бути стійкою до можливих відключень електроенергії.

Для реалізації цього функціоналу застосовано такі пристрої: плата розробника ESP-WROOM-32; електромеханічне реле SRD-5VDC-SL-C управління 5В навантаження 250В 10А; світлодіодна лампа 220 В; детектор напруги з оптопарою, пристрій має оптогальванічну розв'язку, це дає можливість підключати модуль безпосередньо до виводу мікроконтролера. За наявності змінної напруги на вході, вихід OUT матиме логічний нуль; блок живлення AC 220V to DC 5V 700mA; модуль зарядки TP4056, який підтримує вхідні роз'єми Type-C та MicroUSB для заряджання Li-ion акумуляторів; MT3608 підвищуючий імпульсний перетворювач напруги; два діоди Шотткі SS14; діод 1N4007S; акумулятор Videx LI-ION 2200 mAh; конденсатор 22000мкФ 16В.

Одна з особливостей модуля ESP32 полягає в тому, що він може не тільки підключатися до існуючої Wi-Fi мережі та працювати як веб-сервер, але також може встановлювати власну мережу, дозволяючи іншим пристроям підключатися безпосередньо до нього та отримувати доступ. Отже, ESP32 може працювати в трьох різних режимах: режим станції, режим точки доступу та обидва перші режими одночасно.

Рисунок 1 відображає електричну схему пристроїв, які використовуються для віддаленого управління світлом.

До платформи підключено 4 реле до пінів gpio 25, gpio 26, gpio 27, gpio 14. Відповідно gpio 34, gpio 35, gpio 32, gpio 33 піни до яких підключено детектор напруги. Пін gpio 39 за допомогою якого перевіряється наявність електроенергії (напруга 220).

Забезпечення безперебійної роботи систем IoT під час відключень електроенергії надзвичайно важливо, оскільки ці системи часто виконують завдання, від яких залежить безпека та стабільність процесів прийняття рішень. У промислових умовах, де системи інтернету речей контролюють виробниче обладнання, втрата зв'язку або даних під час відключення електроенергії може призвести до фінансових втрат і навіть небезпечних ситуацій. Тому важливо забезпечити резервне живлення або інші засоби підтримки функціональності IoT, щоб система могла продовжувати зберігати дані та функціонувати за будь-яких умов. Отже, зменшуючи ризик і запобігаючи можливим наслідкам аварій.

Для збереження параметрів при відключенні електроенергії доступна можливість перемикання між зовнішнім живленням та акумулятором для живлення плати ESP-WROOM-32. На рисунку 2 наведена схема, яка дозволяє здійснювати перехід

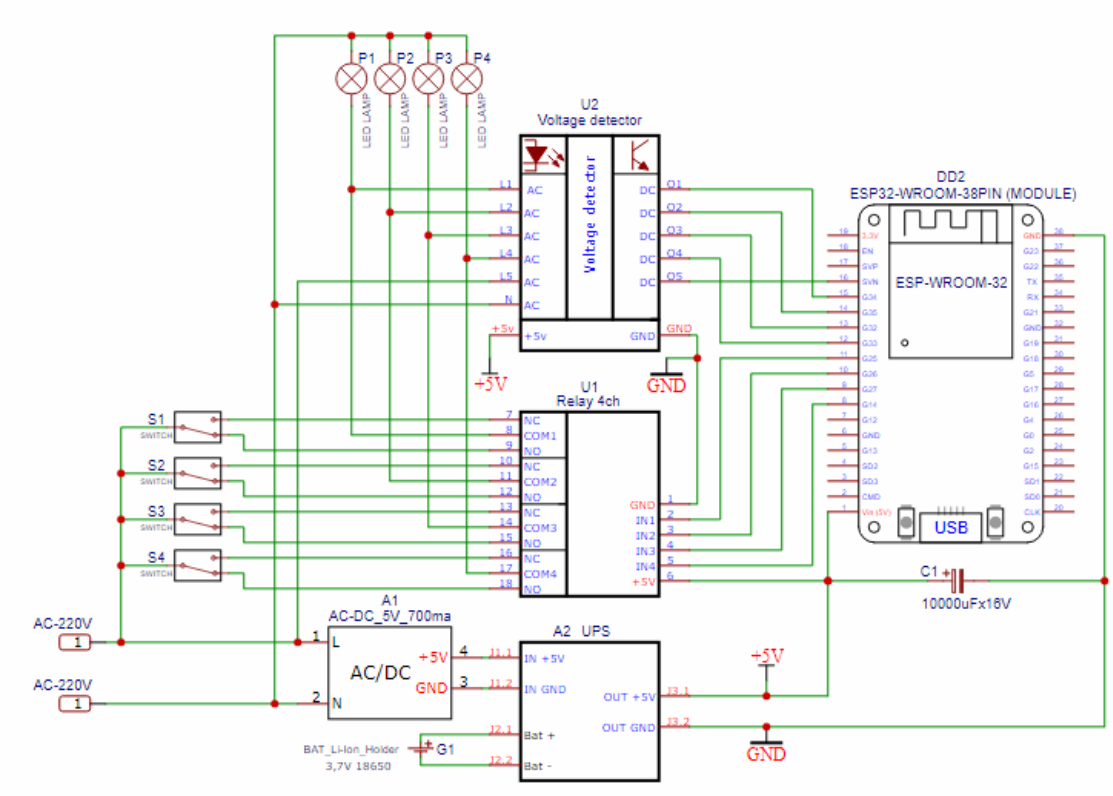

Рис. 1. Електрична схема пристрою керування

від зовнішнього живлення до живлення від акумулятора, а також контролювати стан акумулятора.

Наведені на рисунках 1, 2 принципові схеми функціонування IoT системи керування світлом повинні доповнюватись відповідним програмним забезпеченням.

Результати. Розглянемо деякі особливості сценаріїв керування обладнанням тестового проєкту.

Більшість сценаріїв використовують бібліотеку WiFi.h. Ця бібліотека надає спеціальні методи ESP для роботи з Wi-Fi, які викликаються для підключення до мережі. Після цього підключається бібліотека WebServer.h, в якій є кілька методів, які використовуються для налаштування серверу і обробки вхідних HTTP запитів, без необхідності реалізовувати низькорівневі деталі підключення та передачі даних.

Для збереження параметрів при відключенні електроенергії використовується EEPROM – область пам'яті на мікроконтролері ESP32. Використання змінної EEPROM_SIZE визначає обсяг доступної пам'яті для зберігання даних в EEPROM.

```
#include "WiFi.h"
#include <WebServer.h>
#include <EEPROM.h>

#define EEPROM_SIZE 512 // розмір EEPROM для ESP32
```

В даному випадку мікроконтролер ESP32 налаштовується в режимі станції та точки доступу, отже безпека з'єднання забезпечується SSID мережі та паролем.

```
String _ssid = ""; // для зберігання SSID
String _password = ""; // для зберігання пароля мережі
String _ssidAP = ""; // SSID AP точки доступу
String _passwordAP = ""; // пароль точки доступу

IPAddress apIP(192, 168, 4, 1);
```

Після цього оголошується об'єкт бібліотеки WebServer. Конструктор цього об'єкта як параметр приймає номер порту (який сервер прослуховуватиме).

```
WebServer HTTP(80);
```

Далі оголошуються виводи GPIO плати NodeMCU, до яких підключені реле, детектори напруги та їх початковий стан.

```
const int relayPins[] = {25, 26, 27, 14}; // піни, до яких підключено реле
const int numRelays = sizeof(relayPins) / sizeof(relayPins[0]);
int relayStates[]={ HIGH, HIGH, HIGH, HIGH}; // стан реле вимкнено
intVoltage_Sensor[]={LOW,LOW,LOW,LOW};// детектор напруги
int Voltage_Sensor_Pin[]={34,35,32,33}; // піни, до яких підключено детектор напруги
```

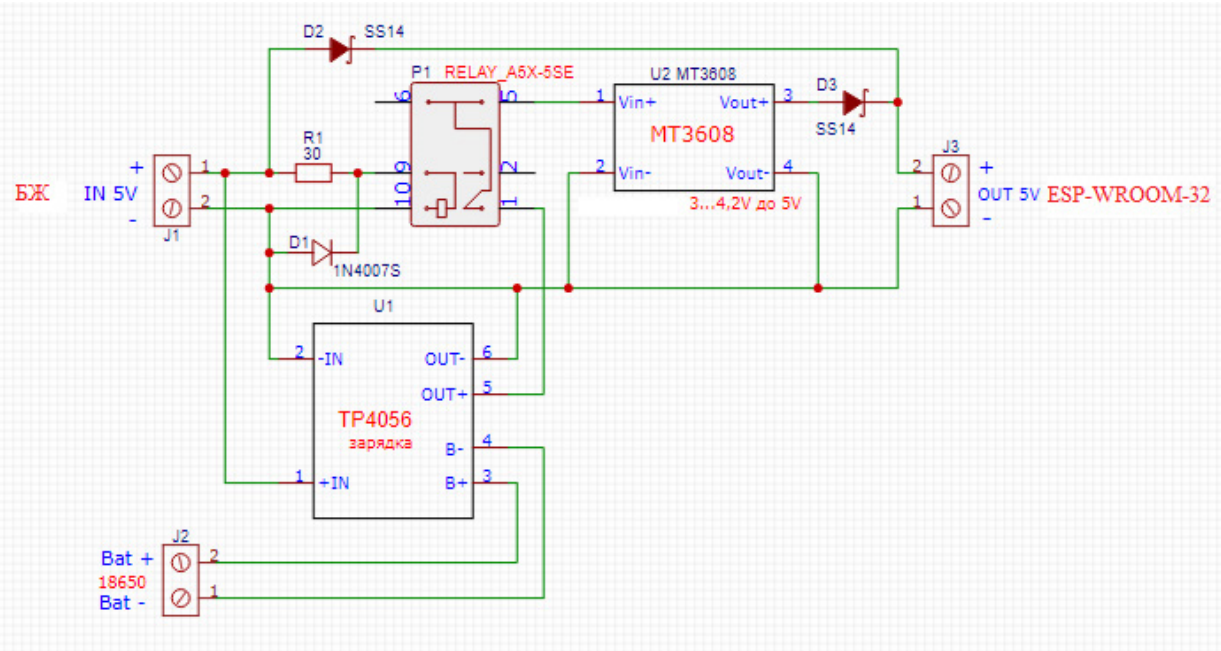

Рис. 2. Схема джерела безперебійного живлення

```
int Light_Pin=39; // пін за допомогою якого перевіряємо наявність світла (напруга) 05 - 2
void loadArraysFromEEPROM(); // завантаження параметрів з EEPROM
String webString=""; // текст для надсилання
```

Перед запуском функції setup() налаштовується сервер HTTP. Насамперед, для налагодження відкривається послідовне з'єднання та встановлюються порти GPIO.

```
Serial.begin(115200);
pinMode(Light_Pin, INPUT);
loadArraysFromEEPROM(); // завантаження параметрів з EEPROM
// готуємо GPI для реле
for (int i = 0; i < numRelays; i++) {
  pinMode(Voltage_Sensor_Pin[i],INPUT); // налаштовує цей контакт на вхід
  digitalWrite(Voltage_Sensor_Pin[i],Voltage_Sensor[i]); }
for (int i = 0; i < numRelays; i++) {
  pinMode(relayPins[i], OUTPUT);
  digitalWrite(relayPins[i], relayStates[i]); }
```

Щоб обробити реальні вхідні запити HTTP в функції loop(), викликається метод handleClient() об'єкта HTTP. Також згідно з запитом змінюється стан реле та перевіряється за допомогою датчика наявність напруги.

```
void loop() {
  HTTP.handleClient();
  if (digitalRead(Light_Pin) == 0) // якщо світло є
  {
  for (int i = 0; i < numRelays; i++) {
    relayStates[i]=digitalRead(relayPins[i]);
    digitalWrite(relayPins[i], relayStates[i]); }
  for (int i = 0; i < numRelays; i++) {
  Voltage_Sensor[i] = digitalRead(Voltage_Sensor_Pin[i]);
  //перевіряємо за допомогою датчика наявність напруги
    digitalWrite(Voltage_Sensor_Pin[i],Voltage_Sensor[i]);}
  }
  else
  {
  // якщо світла немає
  saveArraysToEEPROM();
  }
}
```

Щоб відповісти на запит HTTP, використовується метод send(). Хоча цей метод можна викликати з іншим набором аргументів, його найпростіша форма складається з коду HTTP відповіді, типу контенту і самого контенту.

Далі створюються вісім функцій для обробки запитів на включення (виключення) реле.

```
void rel1_on0() {
digitalWrite(relayPins[0], LOW);
HTTP.send(200, "text/html","gpio"+String((int) relayPins[0])+" on");
 // увімкнути освітлення 1 житлової кімнати
}
void rel1_off0() {
digitalWrite(relayPins[0],HIGH );
HTTP.send(200, "text/html","gpio"+String((int) relayPins[0])+" off"); // вимкнути освітлення 1 житлової кімнати
}
```

Функція baza() просто поєднує у великий рядок стан реле та наявність напруги і повертає її у функцію HTTP.send().

При відсутності електроенергії розроблено функції які надають можливість зберігати та відновлювати параметри освітлення в EEPROM - область пам'яті на мікроконтролері ESP32.

```
// Функція збереження параметрів освітлення при вимкненні світла EEPROM
void saveArraysToEEPROM() {
EEPROM.begin(EEPROM_SIZE);
// Зберігаємо значення стану реле
for (int i = 0; i < sizeof(relayStates) / sizeof(relayStates[0]); i++) {
EEPROM.put(i * sizeof(relayStates[0]), relayStates[i]); }
// Зберігаємо значення детектора напруги
int startIndex = sizeof(relayStates);
for (int i = 0; i < sizeof(Voltage_Sensor) / sizeof(Voltage_Sensor[0]); i++) {
EEPROM.put(startIndex + i * sizeof(Voltage_Sensor[0]), Voltage_Sensor[i]); }
EEPROM.commit(); // Зберігаємо зміни
EEPROM.end(); }

// Функція для завантаження параметрів з EEPROM
void loadArraysFromEEPROM() {
EEPROM.begin(EEPROM_SIZE);
// Завантажуємо значення стану реле
for (int i = 0; i < sizeof(relayStates) / sizeof(relayStates[0]); i++) {
EEPROM.get(i * sizeof(relayStates[0]), relayStates[i]);}
// Завантажуємо значення детектора напруги
int startIndex = sizeof(relayStates);
for (int i = 0; i < sizeof(Voltage_Sensor) / sizeof(Voltage_Sensor[0]); i++) {
EEPROM.get(startIndex + i * sizeof(Voltage_Sensor[0]), Voltage_Sensor[i]); }
EEPROM.end();}
```

Дослідимо алгоритм виявлення стану освітлення на прикладі однієї світлодіодної лампи.

Коли світло вимкнене:

– Реле знаходиться у вимкненому стані (логічна 1), а детектор напруги сигналізує логічну 1, що вказує на відсутність змінної напруги на лампі. Для ввімкнення світла необхідно надіслати GET-запит із командою /pinon для активації реле. У результаті реле переходить у ввімкнений стан (логічний 0), і детектор напруги також фіксує логічний 0.

– Реле перебуває у ввімкненому стані (логічний 0), тоді як детектор напруги показує логічну 1 (лампа вимкнена через прохідний вимикач). Щоб увімкнути світло зі смартфона, потрібно надіслати GET-запит із командою /pinoff для вимкнення реле. У результаті реле переходить у стан вимкнення (логічна 1), а детектор напруги фіксує логічний 0 (світло ввімкнено).

Коли світло ввімкнене:

– Реле перебуває у ввімкненому стані (логічний 0), а детектор напруги фіксує логічний 0, що свідчить про наявність змінної напруги на лампі. Щоб вимкнути світло, необхідно надіслати GET-запит із командою /pinoff для відключення реле. У результаті реле переходить у вимкнений стан (логічна 1), а детектор напруги починає повертати логічну 1.

– Реле перебуває у стані вимкнення (логічна 1), тоді як детектор напруги сигналізує логічний 0, що вказує на ввімкнене світло через прохідний вимикач. Для вимкнення світла за допомогою смартфона необхідно надіслати GET-запит із командою /pinon для активації реле. У результаті реле переходить у ввімкнений стан (логічний 0), а детектор напруги починає повертати логічну 1, що свідчить про вимкнення світла.

Висновки. Розглянуто процес проєктування апаратної частини та програмної реалізації системи IoT на прикладі пристрою автоматичного керування світлом. Запропоновано ефективну архітектуру з точки зору стійкості системи при відключеннях електроенергії.

ЛІТЕРАТУРА

1. Espressif Systems. ESP32. Available online: https://www.espressif.com/en/products/socs/esp32 (accessed on 09 November 2024).
2. Domínguez-Bolaño T., Campos O., Barral V., Escudero C.J., García-Naya J.A. An overview of IoT architectures, technologies, and existing open-source projects. Internet of Things Volume 20, November 2022, 100626. URL: https://doi.org/10.1016/j.iot.2022.100626.
3. Lombardi M., Pascale F., Santaniello D. Internet of Things: A General Overview between Architectures, Protocols and Applications. Information 2021, 12, 87. URL: https://doi.org/10.3390/info12020087.
4. Kumar A., Sharma S., Singh A., Alwadain A., Choi B.-J., Manual-Brenosa J., Ortega-Mansilla A., Goyal N. Revolutionary Strategies Analysis and Proposed System for Future Infrastructure in Internet of Things. Sustainability. 2022; 14(1):71. URL: https://doi.org/10.3390/su14010071.
5. Goyal P., Sahoo A.K., Sharma T.K. Internet of things: Architecture and enabling technologies. Materials Today: Proceedings, 34, 3, 2021, pp. 719–735. URL: https://doi.org/10.1016/j.matpr.2020.04.678.
6. Nikoui T.S., Rahmani A.M., Balador A. Javadi H.H.S. Internet of things architecture challenges: A systematic review. International Journal of Communication Systems, vol. 34, no. 4, pp. E4678, 2021. URL: https://doi.org/10.1002/dac.4678.
7. Marshoodulla S.Z., Saha G. A survey of data mining methodologies in the environment of IoT and its variants, Journal of Network and Computer Applications, 228, (103907), (2024). URL: https://doi.org/10.1016/j.jnca.2024.103907.
8. Hercog D, Lerher T, Truntič M, Težak O. Design and Implementation of ESP32-Based IoT Devices. Sensors. 2023, 23(15):6739. URL: https://doi.org/10.3390/s23156739.
9. Ramadhani R.R., Yuliana M., Pratiarso A. Smart Room Lighting System for Energy Efficiency in Indoor Environment. International Journal of Artificial Intelligence & Robotics (IJAIR). 2022, 4(2), 48–58. URL: https://doi.org/10.25139/ijair.v4i2.5266.
10. Espinosa-Gavira M.J., Agüera-Pérez A., Palomares-Salas J.C., Sierra-Fernandez J.M., Remigio-Carmona P., González de-la-Rosa J.J. Characterization and Performance Evaluation of ESP32 for Real-time Synchronized Sensor Networks, Procedia Computer Science, Volume 237, 2024, Pages 261-268, ISSN 1877-0509. URL: https://doi.org/10.1016/j.procs.2024.05.104.
11. Sineglazov V., Khotsyanovsky V. Camera Image Processing on ESP32 Microcontroller with Help of Convolutional Neural Network. Automation and computer-integrated technologies, 2022, Vol. 2, No. 72. URL: https://doi.org/10.18372/1990-5548.72.16939.

REFERENCES

1. Espressif Systems. ESP32. Available online: https://www.espressif.com/en/products/socs/esp32 (accessed on 09 November 2024).

2. Domínguez-Bolaño T., Campos O., Barral V., Escudero C.J., García-Naya J.A. (2022). An overview of IoT architectures, technologies, and existing open-source projects. Internet of Things Volume 20, 100626. https://doi.org/10.1016/j.iot.2022.100626
3. Lombardi M., Pascale F., Santaniello D. (2021). Internet of Things: A General Overview between Architectures, Protocols and Applications. Information. 12, 87. https://doi.org/10.3390/info12020087
4. Kumar A., Sharma S., Singh A., Alwadain A., Choi B.-J., Manual-Brenosa J., Ortega-Mansilla A., Goyal N. (2022). Revolutionary Strategies Analysis and Proposed System for Future Infrastructure in Internet of Things. Sustainability. 14(1):71. https://doi.org/10.3390/su14010071
5. Goyal P., Sahoo A.K., Sharma T.K. (2021). Internet of things: Architecture and enabling technologies. Materials Today: Proceedings, 34, 3, pp. 719-735. https://doi.org/10.1016/j.matpr.2020.04.678
6. Nikoui T.S., Rahmani A.M., Balador A. Javadi H.H.S. (2021). Internet of things architecture challenges: A systematic review. International Journal of Communication Systems, vol. 34, no. 4, pp. E4678. https://doi.org/10.1002/dac.4678
7. Marshoodulla S.Z., Saha G. (2024). A survey of data mining methodologies in the environment of IoT and its variants, Journal of Network and Computer Applications, 228, (103907). https://doi.org/10.1016/j.jnca.2024.103907
8. Hercog D, Lerher T, Truntič M, Težak O. (2023). Design and Implementation of ESP32-Based IoT Devices. Sensors. 23(15):6739. https://doi.org/10.3390/s23156739
9. Ramadhani R.R., Yuliana M., Pratiarso A. (2022). Smart Room Lighting System for Energy Efficiency in Indoor Environment. International Journal of Artificial Intelligence & Robotics (IJAIR). 4(2), 48-58. https://doi.org/10.25139/ijair.v4i2.5266
10. Espinosa-Gavira M.J., Agüera-Pérez A., Palomares-Salas J.C., Sierra-Fernandez J.M., Remigio-Carmona P., González de-la-Rosa J.J. (2024). Characterization and Performance Evaluation of ESP32 for Real-time Synchronized Sensor Networks, Procedia Computer Science, Volume 237, Pages 261-268, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2024.05.104.
11. Sineglazov V., Khotsyanovsky V. (2022). Camera Image Processing on ESP32 Microcontroller with Help of Convolutional Neural Network. Automation and computer-integrated technologies. Vol. 2, No. 72. https://doi.org/10.18372/1990-5548.72.16939