УДК 004.35:007.52

DOI 10.31651/2076-5886-2023-1-27-40

PACS 07.05.Tp, 07.07.Df

ДОМІНІЧЕНКО Віталій Станіславович
студент спеціальності «Інформаційні система та технології» Черкаського національного університету імені Богдана Хмельницького
e-mail: dominichenko_v_s@vu.cdu.edu.ua

ПІСКУН Олександр Варфоломійович
кандидат технічних наук, доцент, завідувач кафедри прикладної математики та інформатики, Черкаський національний університет ім. Б. Хмельницького
e-mail: piskun@ukr.net
ORCID 0000-0001-5334-6337

ГЛАДКА Людмила Іванівна
к.ф.-м.н., доцент кафедри автоматизації та комп'ютерно-інтегрованих технологій, Черкаський національний університет імені Б. Хмельницького
e-mail: l_i_gladka@vu.cdu.edu.ua
ORCID 0000-0002-7030-9666

РОЗРОБКА АПАРАТНО-ПРОГРАМНОЇ СИСТЕМИ ДЛЯ АВТОМАТИЧНОЇ ГРИ НА «СПІВАЮЧІЙ» ЧАШІ З ДИСТАНЦІЙНИМ КЕРУВАННЯМ ЧЕРЕЗ WI-FI

У статті описано розробку апаратно-програмної системи автоматизованої гри на «співаючій» чаші з дистанційним керуванням через Wi-Fi. Сучасний темп життя призводить до зростання рівня стресу серед людей, що обумовлює актуальність засобів для релаксації, зокрема «співаючих» чаш. Проте традиційна гра на такій чаші потребує постійної присутності людини. Розроблена система усуває цей недолік, автоматизуючи процес гри з можливістю дистанційного керування через веб-інтерфейс. Апаратну основу становить плата розробки NodeMCU ESP8266 з вбудованим Wi-Fi-модулем та електромагніт, керований через MOSFET-транзистор. Реалізовано три режими роботи: «Manual» (поодинокий удар), «UnpredictaBell» (псевдовипадкові удари у заданих діапазонах) та «DreamDive» (поступове затухання з подальшим наростанням для медитативного пробудження). Веб-інтерфейс розроблено як односторінковий застосунок на базі бібліотеки Preact.js та забезпечує зручне керування з будь-якого пристрою, оснащеного браузером. Пристрій підтримує два режими Wi-Fi: точка доступу та клієнт локальної мережі. Взаємодія між інтерфейсом і пристроєм здійснюється через REST API на базі асинхронного HTTP-сервера ESPAsyncWebServer. У статті описано архітектуру системи, обґрунтування вибору апаратних компонентів, алгоритми трьох режимів керування та процес розробки веб-інтерфейсу. Результати тестування підтверджують коректність роботи системи та відповідність усім поставленим вимогам.

Ключові слова: *«співаюча» чаша, Інтернет речей, NodeMCU ESP8266, веб-інтерфейс, вбудовані системи, RTOS, релаксація.*

Вступ

Сучасний ритм життя, насичений постійними стресовими навантаженнями, спонукає мільйони людей до пошуку ефективних засобів психологічного

розвантаження. Згідно зі статистикою Statista [1], хронічний стрес є одним із провідних чинників, що негативно впливають на продуктивність праці, якість міжособистісних стосунків та стан здоров'я загалом. Як наслідок, зростає популярність практик, заснованих на медитації та звукотерапії.

Серед інструментів звукотерапії особливе місце посідає «співаюча» чаша – стародавній музичний інструмент, поширений у різноманітних культурах Сходу. Як показано в дослідженні Goldsby et al. [2], регулярне прослуховування звуків «співаючої» чаші позитивно впливає на настрій, знижує рівень напруженості та покращує суб'єктивне відчуття добробуту. Коливання чаші породжують звук із багатим спектром обертонів у широкому діапазоні частот, що відрізняє її від більшості сучасних електронних засобів релаксації. Проте практичне застосування чаші потребує постійної уваги та фізичної присутності людини, що суттєво обмежує сценарії її використання – зокрема, унеможливлює застосування чаші під час засинання чи медитації без партнера.

Завдання автоматизації процесу гри на чаші з наданням можливості дистанційного керування через бездротову мережу є актуальним і вирішується у цій роботі. Запропонована апаратно-програмна система поєднує можливості платформи Arduino, Wi-Fi-модуля ESP8266, електромагнітного приводу та динамічного веб-інтерфейсу, реалізуючи тим самим повноцінний IoT-пристрій побутового призначення.

Метою роботи є розробка і практична реалізація апаратно-програмної системи автоматизованої гри на «співаючій» чаші, що забезпечує дистанційне керування режимами гри через Wi-Fi-мережу з використанням веб-інтерфейсу, а також дослідження ефективності обраних апаратних та програмних рішень і підтвердження їх достатності для задоволення визначених вимог.

Виклад основного матеріалу

1. Огляд предметної галузі та обґрунтування підходу

1.1. «Співаюча» чаша як акустичний об'єкт

«Співаюча» чаша є різновидом стоячого дзвону і в тих чи інших формах існує в багатьох культурах світу (рис. 1). Найпоширеніші чаші виготовляються зі сплавів металів методом кування або лиття. Звучання чаші утворюється або шляхом удару по її стінці медіатором, або рівномірними обертальними рухами медіатора по зовнішньому краю – метод, що викликає явище резонансу («заводить» чашу). У цій роботі реалізовано метод удару, оскільки він технічно придатний для автоматизації за допомогою електромагнітного привода з прийнятною складністю механізму.

Акустичний сигнал чаші характеризується широким спектром обертонів у діапазоні приблизно від 100 Гц до 8 кГц (рис. 2). Після удару звук поступово затухає впродовж десятків секунд. Важливою особливістю є те, що сила удару безпосередньо впливає на амплітуду коливань і, відповідно, на гучність та тривалість звучання, проте не змінює суттєво спектральний склад сигналу. Ця лінійна залежність між механічним впливом і акустичним результатом є фундаментальною для алгоритмів керування, реалізованих у роботі: управляючи лише силою удару та інтервалом між ударами, можна створювати різноманітні акустичні ефекти.

1.2. Підхід на основі Інтернету речей

Системи дистанційного керування фізичними пристроями є усталеним напрямком у галузі Інтернету речей (IoT). Концепція IoT, детально описана в роботах Gubbi et al. [3] та Atzori et al. [9], передбачає інтеграцію фізичних пристроїв до мережі з метою обміну даними та дистанційного управління ними. Atzori et al. визначають ключові компоненти IoT-системи: сенсори та виконавчі механізми, мережевий рівень передачі

даних і рівень застосунків. У розробленій системі ці компоненти представлені відповідно: електромагнітним приводом (виконавчий механізм), Wi-Fi-модулем ESP8266 (мережевий рівень) і веб-інтерфейсом на базі REST API (рівень застосунків).

Рис. 1. Різноманіття форм «співаючих» чаш

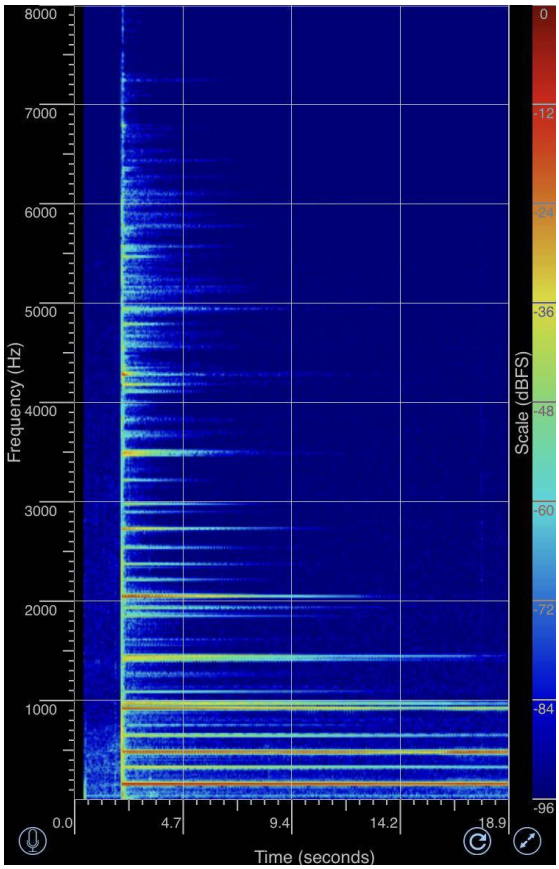

Рис. 2. Спектрограма звуку «співаючої» чаші, що демонструє широкий спектр обертонів

Для організації взаємодії між клієнтом і вбудованим сервером обрано архітектурний стиль REST (Representational State Transfer), описаний у докторській дисертації Філдінга [4]. Основними принципами REST є: клієнт-серверна архітектура, відсутність стану сесії на сервері (stateless), підтримка кешування, уніформний інтерфейс та багатошарова система. У контексті мікроконтролерної системи з обмеженими ресурсами ці принципи є особливо цінними: відсутність сесійного стану суттєво знижує споживання оперативної пам'яті, а уніформний інтерфейс спрощує реалізацію клієнта.

1.3. Вибір апаратної платформи

Платформа Arduino обрана як базова завдяки розвиненій екосистемі бібліотек, широкій документованості та великій спільноті розробників. Порівняно з іншими мікроконтролерними платформами, Arduino-сумісні плати мають нижчий поріг входження для швидкого прототипування без жертвування функціональністю. Для проектів класу IoT, де потрібна інтеграція Wi-Fi, природним вибором є сімейство ESP8266/ESP32 завдяки вбудованому бездротовому інтерфейсу та підтримці Arduino IDE через сторонній пакет плат.

Протягом розробки системи змінилося два апаратних покоління: від конфігурації Arduino Mega 2560 + ESP8266 (де обидва модулі є фізично окремими і взаємодіють через UART) до плати NodeMCU ESP8266, де мікроконтролер і Wi-Fi-чип є єдиним пристроєм із нативною підтримкою мережевого стека та файлової системи. Детальне порівняння платформ наведено в наступному розділі.

2. Постановка задачі

Необхідно розробити апаратно-програмну систему, яка відповідає таким функціональним вимогам:

1. Автоматизоване керування електромагнітним ударним механізмом для гри на «співаючій» чаші у щонайменше трьох режимах із різними характеристиками сили та частоти ударів.
2. Дистанційне керування пристроєм через динамічний веб-інтерфейс за протоколом HTTP без перезавантаження сторінки при зміні параметрів.
3. Підтримка двох режимів Wi-Fi: точки доступу (AP) для первинного налаштування та клієнта зовнішньої мережі (STA) для інтеграції в домашню мережу.
4. Зберігання параметрів Wi-Fi-підключення в енергонезалежній пам'яті з автоматичним відновленням з'єднання після перезапуску.
5. Підтримка одночасного підключення кількох HTTP-клієнтів без порушення стабільності роботи виконавчого механізму.
6. Коректне адаптивне відображення веб-інтерфейсу на мобільних пристроях та настільних комп'ютерах.

Нефункціональними обмеженнями системи є апаратні ресурси плати NodeMCU ESP8266: 128 КБ оперативної пам'яті (RAM) та 4 МБ флеш-пам'яті [5], з яких частина відводиться під файлову систему SPIFFS для розміщення файлів веб-інтерфейсу. Крім того, відсутність на платі апаратного годинника реального часу (RTC) і неможливість синхронізації часу без підключення до Інтернет вносять додаткові обмеження на реалізацію будильникових функцій.

Вхідними параметрами системи є: режим роботи, значення сили та інтервалу ударів (цілочисельні в діапазоні 1–5), час завершення (мітка часу, передана клієнтом у вигляді дельти від millis()), дані підключення до Wi-Fi (SSID та пароль). Вихідним є

механічний удар по чаші через електромагнітний привод, тривалість активації якого визначається обраною силою у відповідності до наперед встановлених констант.

3. Апаратна складова системи

3.1. Порівняльний аналіз апаратних платформ і вибір NodeMCU ESP8266

На початковому етапі проекту використовувався модифікований контролер Arduino Mega 2560 з вбудованим Wi-Fi-модулем ESP8266 на одній платі. Обидва компоненти – процесор ATmega2560 і чіп ESP8266 – на такій платі можуть працювати як спільно (через UART-з'єднання), так і незалежно. Для спільного режиму ESP8266 прошивається прошивкою NonOS SDK, після чого ATmega2560 управляє ним за допомогою AT-команд.

Проте ця конфігурація виявила ряд принципових обмежень. По-перше, відсутність підтримки асинхронного HTTP-сервера призводила до взаємного блокування: під час обробки HTTP-запиту ATmega2560 не міг одночасно керувати GPIO для електромагніту. По-друге, весь HTML/CSS/JavaScript код доводилося зберігати безпосередньо у Flash-рядках у тексті програми через функцію F(""), що унеможливлювало використання сучасних JavaScript-фреймворків. По-третє, програмування плати вимагало перемикання DIP-перемикачів для зміни режиму роботи між ATmega та ESP8266.

Перехід на плату NodeMCU ESP8266 version 1.0 (рис. 3) усунув усі ці проблеми. Плата побудована на модулі ESP-12E, що містить мікросхему ESP8266 з 32-бітним RISC-мікропроцесором Tensilica Xtensa LX106. Процесор підтримує операційну систему реального часу (RTOS) і працює на тактовій частоті від 80 до 160 МГц [5]. Планувальник RTOS гарантує детерміновану обробку конкурентних задач: Wi-Fi-стека, HTTP-сервера та керування GPIO одночасно і без взаємного блокування. NodeMCU також має 128 КБ оперативної пам'яті та 4 МБ флеш-пам'яті, що є достатнім для зберігання файлів SPA-застосунку у файловій системі SPIFFS.

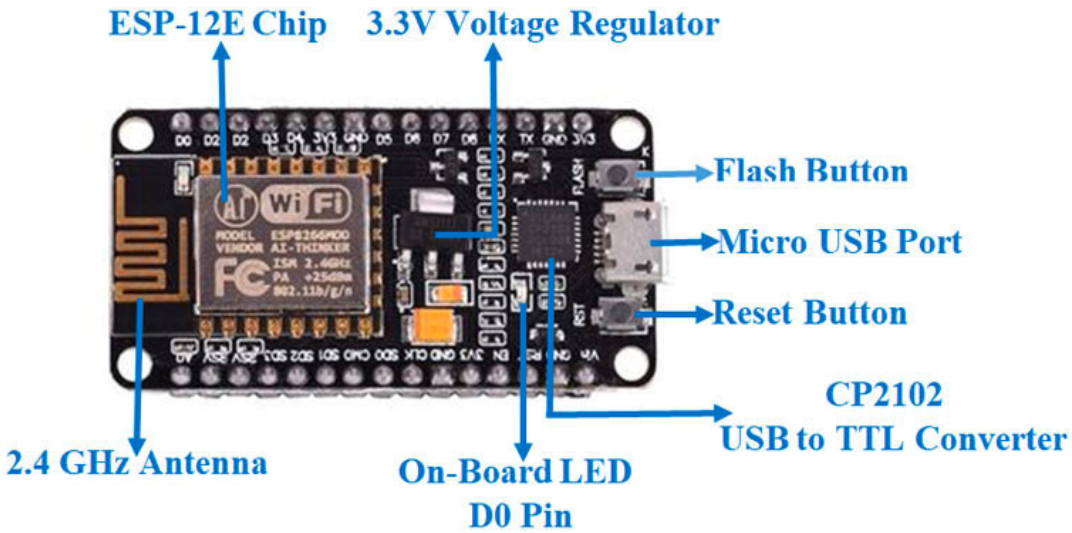

Рис. 3. Плата розробки NodeMCU version 1.0

Ключові переваги NodeMCU ESP8266 перед конфігурацією Arduino Mega 2560 + ESP8266 зведено в таблиці 1.

Плата повністю сумісна з Arduino IDE завдяки підтримці пакету ESP8266 Board Package. Для завантаження файлів веб-інтерфейсу у SPIFFS використовується стороннє розширення ESP8266 Filesystem Uploader для Arduino IDE.

Таблиця 1

Порівняння апаратних платформ

Характеристика	***Arduino Mega 2560 + ESP8266***	***NodeMCU ESP8266***
Файлова система	Відсутня (тільки Flash-рядки F(""))	SPIFFS (до 3 МБ для файлів застосунку)
Асинхронний HTTP-сервер	Не підтримується	Підтримується (ESPAsyncWebServer)
Кількість фізичних плат	2 + перемикач режимів	1
Тактова частота	16 МГц (ATmega2560)	80–160 МГц
Програмування	DIP-перемикачі + 2 окремих завантаження	Один кабель micro-USB
Wi-Fi-взаємодія	Через AT-команди по UART	Нативна підтримка в SDK
Оперативна пам'ять	8 КБ (ATmega2560)	128 КБ
Підтримка SPA-застосунків	Неможлива	Повноцінна (хостинг JS/HTML/CSS)
Конкурентна обробка задач	Відсутня (послідовне виконання)	RTOS-планувальник

3.2. Виконавчий механізм та схема керування

Механічним елементом, що здійснює удари по стінці чаші, є електромагніт із номінальною вантажопідйомністю 55 кг, робочою напругою 12 В та струмом споживання до 340 мА. Безпосереднє керування електромагнітом від GPIO-виходу NodeMCU неможливе з двох причин: максимальна вихідна напруга GPIO становить 3,3 В при максимальному струмі ~12 мА, тоді як електромагніт потребує 12 В / 340 мА; крім того, необхідна гальванічна ізоляція індуктивного навантаження від чутливої мікроконтролерної схеми для захисту від зворотних ЕРС.

Обидві проблеми вирішено застосуванням модуля комутації на базі польового MOSFET-транзистора F5305S (рис. 4) з вбудованою оптичною розв'язкою [6]. Допустима напруга керуючого сигналу: 3–20 В (сумісно з 3,3 В NodeMCU). Вихідні параметри: 5–36 В при тривалому струмі до 5 А (максимальний – 20 А). Сигнал рівня логічної «1» (3,3 В) від GPIO-піна D7 перемикає MOSFET, підключаючи обмотку електромагніту до джерела живлення 12 В.

Рис. 4. Модуль MOSFET-транзистора F5305S

Вся система живиться від одного джерела постійного струму напругою 12 В, підключеного до плати розширення NodeMCU. Плата розширення містить

понижуючий конвертер 12 В → 5 В / 800 мА, з якого через вбудований стабілізатор NodeMCU формується напруга 3,3 В для мікроконтролера. Таким чином, обидва компоненти – плата розробки та електромагніт – живляться від одного джерела, що виключає необхідність окремих блоків живлення у кінцевому пристрої.

3.3. Метод кодування сили удару

Оскільки плата NodeMCU ESP8266 не забезпечує аналогового керування силою вихідного струму GPIO (лише напругою), а типова реалізація ШІМ-регулювання струму в індуктивному навантаженні потребувала б додаткового LC-фільтра, у роботі застосовано альтернативний метод – часове кодування сили удару: сила удару визначається тривалістю утримання електромагніту у ввімкненому стані.

При короткому імпульсі (мала тривалість) сердечник електромагніта не встигає суттєво розігнатися до моменту механічного контакту з медіатором, забезпечуючи слабкий удар. При тривалому імпульсі (максимальне значення – до 350 мс) сердечник набирає значну швидкість і наносить сильний удар по чаші. Після завершення імпульсу пружний елемент конструкції повертає медіатор у вихідне положення, готуючи систему до наступного удару. Межі діапазону тривалостей встановлено експериментально: MIN_STRENGTH = 150 мс, MAX_STRENGTH = 350 мс. Такий підхід дозволяє обійти апаратне обмеження платформи без додаткових схемотехнічних рішень.

4. Алгоритм керування

4.1. Загальна структура програми

Загальна структура програми відповідає стандартній схемі Arduino-застосунку з функціями setup() та loop(), розширеною засобами асинхронного програмування RTOS. Функція setup() виконує одноразову ініціалізацію: налаштування GPIO-піна D7 як виходу, монтування файлової системи SPIFFS, зчитування конфігурації з JSON-файлу, спробу підключення до збереженої Wi-Fi-мережі та запуск асинхронного HTTP-сервера з реєстрацією обробників запитів. Функція loop() реалізує основний цикл: перевіряє поточний активний режим і виконує відповідні дії для керування електромагнітом – формує імпульси потрібної тривалості та витримує задані інтервали між ними.

Логіка ініціалізації Wi-Fi реалізована за принципом «спроба – відкат». Якщо у JSON-конфігурації збережено дані зовнішньої мережі (SSID та пароль), система намагається підключитися в режимі STA протягом фіксованого таймауту. У разі невдачі (мережа недоступна або змінила пароль) конфігурація очищується, і система переходить у режим точки доступу (AP) із фіксованим SSID «MAUMVI». Такий підхід забезпечує роботу пристрою в будь-яких умовах без необхідності ручного скидання налаштувань.

4.2. Режим «Manual»

Режим «Manual» призначений для поодиноких ударів на розсуд користувача та реалізує найпростішу форму взаємодії. Після отримання HTTP GET-запиту /api/hit?holdTime=<мс> сервер зчитує параметр holdTime – тривалість активації електромагніту в мілісекундах – і передає його в чергу задачі керування приводом. Задача встановлює GPIO D7 у стан HIGH на задану тривалість, після чого повертає його в LOW.

З боку веб-інтерфейсу сила удару задається тривалістю утримання кнопки «Hit the Bowl»: JavaScript-обробник фіксує момент натискання (mousedown / touchstart) і момент відпускання (mouseup / touchend), обчислює різницю в мілісекундах (обмежену значенням MAX_STRENGTH), після чого надсилає її як параметр holdTime. Таким

чином, між зусиллям утримання кнопки і силою механічного удару існує прямий, інтуїтивно зрозумілий зв'язок. Навколо кнопки у веб-інтерфейсі анімується коло наростаючої яскравості, що дає користувачу візуальний зворотний зв'язок про накопичену силу.

4.3. Режим «UnpredictaBell»

Режим «UnpredictaBell» реалізує автономну гру з псевдовипадковим характером ударів, що наближає автоматичну гру до живої. Монотонна регулярна послідовність ударів дратує слухача, тому введення контрольованої варіативності є принциповою вимогою до алгоритму.

Користувач задає два цілочисельних параметри в діапазоні від 1 до 5: центральне значення сили удару та центральне значення інтервалу між ударами. Ці параметри лінійно масштабуються до фізичних діапазонів: тривалість удару – від MIN_STRENGTH = 150 мс до MAX_STRENGTH = 350 мс; інтервал між ударами – від MIN_INTERVAL = 3000 мс до MAX_INTERVAL = 15000 мс. Після масштабування фактичні значення тривалості удару та інтервалу для кожного наступного удару генеруються за допомогою функції random() в певному симетричному околі центрального значення. Це запобігає монотонності звучання і водночас зберігає загальний характер гри, заданий користувачем.

4.4. Режим «DreamDive»

Режим «DreamDive» є найскладнішим алгоритмічно і орієнтований на сценарій поступового засинання під звуки чаші з наступним плавним пробудженням у заданий час. Алгоритм складається з трьох послідовних фаз:

Фаза 1 – затухання (Fade Out). Починаючи від поточних параметрів сили та інтервалу (успадкованих від режиму «UnpredictaBell» або заданих за замовчуванням), система поступово зменшує силу ударів і збільшує паузи між ними. Тривалість фази задається користувачем у хвилинах (параметр transitionTime). Зміна параметрів відбувається лінійно з кожним наступним ударом – від початкових значень до мінімально допустимих.

Фаза 2 – очікування (Awaiting). Система не виконує ударів. Тривалість цієї фази автоматично розраховується сервером як різниця між заданим часом завершення режиму та сумою тривалостей фаз 1 і 3, що дозволяє точно дотримати час завершення незалежно від тривалості переходів.

Фаза 3 – наростання (Fade In). Система виконує дзеркальний до Фази 1 процес: від мінімальних значень параметрів до початкових. Тривалість також дорівнює transitionTime.

Оскільки NodeMCU не має доступу до реального астрономічного часу без підключення до Інтернет, час завершення режиму передається клієнтом у вигляді дельт відносно поточної мілісекундної мітки пристрою (millis()). Клієнтський браузер конвертує бажаний час будильника в мілісекунди відносно Epoch, обчислює дельту від поточного millis() плати та передає окремі тривалості для кожної з трьох фаз. Таким чином, плата не потребує синхронізації часу, а браузер бере на себе роль перекладача між астрономічним часом і відносними мілісекундами пристрою.

Після завершення режиму передбачено опціональний автоматичний перехід до режиму «UnpredictaBell» (параметр toSwitch), що дозволяє продовжити акустичний супровід після пробудження без додаткових дій з боку користувача.

4.5. Збереження конфігурації у файловій системі

Зберігання параметрів Wi-Fi-підключення здійснюється через JSON-файл у

файловій системі SPIFFS флеш-пам'яті. Такий підхід обрано замість EEPROM з кількох міркувань: ресурс перезаписів флеш-пам'яті (понад 30 мільйонів циклів [8]) значно перевищує ресурс EEPROM (100 тисяч циклів); JSON-формат забезпечує зручну роботу зі структурованими даними без ручного управління адресами; файлова система дозволяє зберігати будь-яку кількість файлів конфігурації, не обмежуючись фіксованим обсягом EEPROM. Збереження відбувається лише за фактом зміни параметрів Wi-Fi-підключення, що мінімізує кількість операцій запису і подовжує ресурс пам'яті.

5. Веб-інтерфейс

5.1. Архітектура та технологічний стек

Веб-інтерфейс реалізовано як односторінковий застосунок (SPA – Single Page Application) на основі бібліотеки Preact.js [7]. Preact.js є легковаговим аналогом React.js з ідентичним API, проте значно меншим розміром бандлу (близько 3 КБ у стисненому вигляді порівняно з ~40 КБ у React), що є критично важливим в умовах обмеженої флеш-пам'яті NodeMCU. Компонентна модель Preact дозволяє декларативно описувати UI і автоматично оновлювати лише ті частини DOM, стан яких змінився, – без ручного маніпулювання HTML і без перезавантаження сторінки.

Скомпільовані статичні файли (HTML, CSS, JavaScript) зберігаються у файловій системі SPIFFS і роздаються клієнту асинхронним HTTP-сервером ESPAsyncWebServer як статичний контент. Завдяки кешуванню браузером статичних ресурсів після першого завантаження, подальша взаємодія з пристроєм генерує лише невеликі JSON-запити, що мінімізує навантаження на вбудований сервер.

Відмова від шаблонного серверного рендерингу (SSR) на користь SPA зумовлена двома принциповими чинниками. По-перше, SSR потребує генерації нової HTML-сторінки для кожного запиту, що навантажує процесор NodeMCU. По-друге, при SSR будь-яке оновлення стану (наприклад, зміна відсотка завершеності прогрес-бару DreamDive) вимагає повного перезавантаження сторінки, тоді як у SPA оновлюється лише відповідний компонент.

5.2. REST API системи

Комунікація між браузерним клієнтом і сервером здійснюється виключно через HTTP GET-запити до задокументованих ендпоінтів. Вибір методу GET (замість POST/PUT) обумовлений простотою реалізації на стороні ESPAsyncWebServer та мінімізацією обсягу коду парсингу на мікроконтролері. Параметри запитів передаються як рядок запиту (query string).

При отриманні запиту обробник зчитує параметри рядка запиту, вносить зміни до глобального стану системи та повертає HTTP 200 без тіла відповіді – за винятком /api/status, що повертає JSON-об'єкт із поточним режимом, значеннями параметрів та відсотком завершеності поточної фази. Клієнт опитує цей ендпоінт з фіксованим інтервалом (polling) для оновлення інтерфейсу.

5.3. Функціональні можливості та особливості UX

Фінальна версія веб-інтерфейсу (рис. 6) містить такі елементи управління:

- панель вкладок для перемикання між трьома режимами роботи;
- режим «Manual»: велика кругла кнопка «Hit the Bowl» із анімованим світловим індикатором наростання сили удару при утриманні;
- режим «UnpredictaBell»: два сегментних перемикачі (1–5) для сили та інтервалу, кнопки START/STOP;

– режим «DreamDive»: годинний вибірник часу завершення, параметр тривалості переходів у хвилинах, перемикач опції «Перейти до UnpredictaBell після завершення»; після запуску – три-сегментний прогрес-бар із назвою поточної фази та відсотком її виконання, кнопка STOP;
– модальне вікно Wi-Fi: поля для SSID та пароля, кнопка Connect.

Рис. 5. Приклади GET-запитів, що надсилаються веб-інтерфейсом

Рис. 6. Остаточний UI/UX дизайн веб-інтерфейсу

Усі зміни стану відображаються у реальному часі без перезавантаження сторінки. Інтерфейс спроектовано з урахуванням принципів адаптивного дизайну: усі елементи оптимізовані для взаємодії пальцем на сенсорному екрані мобільного пристрою, а ширина блоку обмежена з центруванням для коректного відображення на широких моніторах.

6. Результати тестування та аналіз

6.1. Функціональне тестування

У ході функціонального тестування перевірялися такі аспекти роботи системи:

Коректність режимів гри. Усі три режими перевірено на відповідність заданим параметрам. Режим «Manual» коректно передає тривалість утримання кнопки у вигляді параметра holdTime і відтворює удар відповідної сили. Режим «UnpredictaBell» формує псевдовипадкові послідовності в заданих діапазонах. Режим «DreamDive» коректно переходить між фазами та завершується в заданий час.

Перемикання режимів. Зміна режиму в процесі активної роботи зупиняє поточний цикл ударів і запускає новий без зависань і збоїв.

Wi-Fi-підключення. Система коректно запускається в режимі AP при першому включенні. Після введення даних зовнішньої мережі та перезапуску система підключається в режимі STA і обслуговує HTTP-запити за статичною IP-адресою.

Збереження конфігурації. Параметри Wi-Fi-підключення успішно зчитуються з JSON-файлу SPIFFS після кожного перезапуску плати.

Часова точність режиму «DreamDive». Перевірено відповідність часу завершення режиму заданому значенню будильника з урахуванням дельта-передачі від браузера. Відхилення склало ±1–2 секунди за 15-хвилинний цикл, що є прийнятним для даного застосування.

6.2. Аналіз проблеми асинхронності та її вирішення

Ключовою технічною проблемою прототипної версії системи (на Arduino Mega 2560 із синхронним сервером) була взаємна блокуванність HTTP-сервера та виконавчого механізму. Синхронна обробка HTTP-запиту займала весь процесорний час, під час якого PWM/GPIO-сигнал для електромагніта не оновлювався. Як вимушений захід у прототипі запроваджувалася примусова 2-секундна затримка між запитами («лоадер»), що компенсувала нестабільність за рахунок суттєвого зниження швидкодії інтерфейсу.

Перехід до NodeMCU ESP8266 з асинхронним сервером ESPAsyncWebServer усунув цю проблему на архітектурному рівні: планувальник RTOS перемикає контекст між задачами обслуговування Wi-Fi-стека, HTTP-сервера та керування GPIO, гарантуючи своєчасне виконання кожної без блокування інших. У фінальній версії затримка «лоадера» відсутня, а під час активної роботи виконавчого механізму веб-сервер продовжує відповідати на запити без затримки.

Зведені результати тестування наведено в таблиці 2.

Таблиця 2

Зведені результати функціонального тестування системи

Параметр тестування	**Очікуваний результат**	**Фактичний результат**
1	2	3
Запуск у режимі AP (без збереженої конфігурації)	SSID «MAUMVI» доступний	Виконано
Підключення до зовнішньої мережі (STA)	Статична IP доступна	Виконано
Відновлення підключення після перезапуску	Автоматичне відновлення	Виконано
Одночасні підключення ≥2 клієнтів	Без збоїв	Виконано

Продовження таблиці 2

1	2	3
Відповідь /api/hit під час активного удару	Без затримки	Виконано (асинхронний сервер)
Прогрес-бар «DreamDive» (точність часу)	±5 с за 15 хв	±1–2 с
Адаптивність: мобільний / ПК	Коректне відображення	Виконано
Збереження конфігурації SPIFFS	Дані між перезапусками	Виконано

6.3. Обговорення обмежень

Серед виявлених обмежень системи варто зазначити такі. По-перше, відсутність на платі NodeMCU апаратного аналогового управління виходом GPIO унеможливила реалізацію плавного ШІМ-регулювання сили удару, замість якого запроваджено метод часового кодування. Це спрощення виявилося практично достатнім, проте теоретично обмежує плавність регулювання. По-друге, NodeMCU не підтримує синхронізацію астрономічного часу без підключення до Інтернет, що потребує передавання часових параметрів від браузерного клієнта й унеможливлює роботу будильникового режиму без пристрою-клієнта в момент запуску. По-третє, розмір SPIFFS-розділу обмежує загальний обсяг файлів веб-інтерфейсу, що зумовлює вибір на користь легковагових бібліотек.

Висновки

У роботі розроблено апаратно-програмну систему для автоматизованої гри на «співаючій» чаші з дистанційним керуванням через Wi-Fi. Система включає:

- обґрунтований вибір апаратної бази (NodeMCU ESP8266) на основі порівняльного аналізу з конфігурацією Arduino Mega 2560 + ESP8266 за критеріями асинхронності, підтримки файлової системи, зручності програмування та обчислювальної потужності;
- реалізацію електромагнітного ударного механізму з керуванням через MOSFET-транзистор F5305S і методом часового кодування сили удару;
- три режими роботи – «Manual», «UnpredictaBell» та «DreamDive» – що охоплюють широкий спектр сценаріїв використання від поодинокого удару до медитативного будильника з нелінійним профілем наростання/затухання;
- REST API на базі асинхронного HTTP-сервера ESPAsyncWebServer для стабільної паралельної обробки HTTP-запитів і керування виконавчим механізмом;
- динамічний SPA-веб-інтерфейс на Preact.js із підтримкою адаптивного відображення та одночасного підключення кількох клієнтів.

Наукова новизна роботи полягає в такому: (1) запропоновано та обґрунтовано метод часового кодування сили електромагнітного удару як альтернативу ШІМ-регулюванню у разі апаратної неможливості аналогового управління виходом GPIO мікроконтролера; (2) розроблено алгоритм «DreamDive» з лінійним профілем наростання/затухання параметрів гри, прив'язаним до астрономічного часу через дельта-передачу від браузерного клієнта; (3) обґрунтовано і реалізовано архітектуру IoT-системи акустичного пристрою побутового призначення на базі RTOS-мікроконтролера з асинхронним HTTP-сервером, що розв'язує проблему конкуренції між задачами мережевого стека і керування виконавчим механізмом.

Перспективами подальшого розвитку є: інтеграція мікрофонного датчика для адаптивного регулювання сили удару за акустичним зворотним зв'язком; розширення алгоритмів генерації ритмічних патернів із застосуванням методів машинного навчання; реалізація WebSocket-з'єднання замість поточного polling-підходу для оновлення стану інтерфейсу в реальному часі; перехід на платформу ESP32 для збільшення обчислювальних ресурсів і розширення функціональності.

Список використаної літератури

1. Statista. Stress and burnout – Statistics & Facts [Електронний ресурс]. – 2023. – Режим доступу: https://www.statista.com/topics/2099/stress-and-burnout/
2. Goldsby T.L., Goldsby M.E., McWalters M., Mills P.J. Effects of Singing Bowl Sound Meditation on Mood, Tension, and Well-Being: An Observational Study // Journal of Evidence-Based Complementary & Alternative Medicine. – 2017. – Vol. 22(3). – P. 401-406.
3. Gubbi J., Buyya R., Marusic S., Palaniswami M. Internet of Things (IoT): A vision, architectural elements, and future directions // Future Generation Computer Systems. – 2013. – Vol. 29(7). – P. 1645-1660.
4. Fielding R.T. Architectural Styles and the Design of Network-based Software Architectures : doctoral dissertation. – University of California, Irvine, 2000. – 162 p.
5. NodeMCU ESP8266 [Електронний ресурс]. – components101.com. – 2020. – Режим доступу: https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet
6. IRF5305S Datasheet – International Rectifier [Електронний ресурс]. – 1999. – Режим доступу: https://html.alldatasheet.com/html-pdf/68168/IRF/IRF5305S/47/1/IRF5305S.html
7. Preact – Getting Started [Електронний ресурс]. – preactjs.com. – 2023. – Режим доступу: https://preactjs.com/guide/v10/getting-started/
8. ESP8266 Arduino Core documentation – Filesystem [Електронний ресурс]. – 2017. – Режим доступу: https://arduino-esp8266.readthedocs.io/en/latest/filesystem.html
9. Atzori L., Iera A., Morabito G. The Internet of Things: A survey // Computer Networks. – 2010. – Vol. 54(15). – P. 2787-2805.

References

1. Statista. (2023). Stress and burnout – Statistics & Facts. https://www.statista.com/topics/2099/stress-and-burnout/
2. Goldsby T.L., Goldsby M.E., McWalters M., Mills P.J. (2017). Effects of Singing Bowl Sound Meditation on Mood, Tension, and Well-Being: An Observational Study. Journal of Evidence-Based Complementary & Alternative Medicine, 22(3), 401-406.
3. Gubbi J., Buyya R., Marusic S., Palaniswami M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), 1645-1660.
4. Fielding R.T. (2000). Architectural Styles and the Design of Network-based Software Architectures [Doctoral dissertation]. University of California, Irvine.
5. NodeMCU ESP8266. (2020). components101.com. https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet
6. IRF5305S Datasheet. (1999). International Rectifier. https://html.alldatasheet.com/html-pdf/68168/IRF/IRF5305S/47/1/IRF5305S.html
7. Preact – Getting Started. (2023). preactjs.com. https://preactjs.com/guide/v10/getting-started/
8. ESP8266 Arduino Core documentation – Filesystem. (2017). https://arduino-esp8266.readthedocs.io/en/latest/filesystem.html
9. Atzori L., Iera A., Morabito G. (2010). The Internet of Things: A survey. Computer Networks, 54(15), 2787-2805.

DOMINICHENKO Vitaliy,
Student, Department of Informatics and Applied Mathematics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
PISKUN Oleksandr,
Candidate of Technical Sciences, Associate Professor, Head of Department of Applied Mathematics and Informatics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
HLADKA Liudmyla,
PhD in Physical and Mathematical Sciences, Associate Professor of the Department of Automation and Computer-Integrated Technologies, The Bohdan Khmelnytsky National

University of Cherkasy, Ukraine

DEVELOPMENT OF A HARDWARE-SOFTWARE SYSTEM FOR AUTOMATED PLAYING OF A SINGING BOWL WITH REMOTE WI-FI CONTROL

Summary. ***Introduction.*** *Modern living conditions lead to increased chronic stress levels among a significant portion of the population. As shown by Goldsby et al. [2], systematic exposure to singing bowl sounds positively affects mood, reduces tension and improves subjective well-being. However, traditional playing of a singing bowl requires constant human presence, which limits its use in unattended scenarios such as falling asleep or solo meditation. The development of an automated system capable of playing the bowl independently while allowing remote control is therefore a relevant engineering task at the intersection of IoT systems and wellness technology.*

Purpose. *The purpose of this work is to develop and practically implement a hardware-software system for automated playing of a singing bowl with remote Wi-Fi control, and to investigate the effectiveness of the selected hardware and software solutions.*

Results. *The system is built on the NodeMCU ESP8266 development board, selected through comparative analysis against the Arduino Mega 2560 + ESP8266 configuration. NodeMCU's native RTOS enables concurrent asynchronous HTTP server operation and GPIO actuator control; its built-in SPIFFS file system hosts SPA web interface assets; and its single-board architecture simplifies development and deployment. The electromagnetic actuator, switched through an F5305S MOSFET module with optical isolation, implements strike strength control through temporal encoding: activation pulse duration (150–350 ms) determines strike strength, compensating for the inability to perform analog current control via NodeMCU GPIO. Three operating modes are implemented: Manual (single strike on demand, strength proportional to button hold duration), UnpredictaBell (pseudo-random autonomous playing within user-defined strength and interval ranges), and DreamDive (gradual fade-out followed by fade-in timed to a user-specified alarm, suitable for sleep and wake scenarios). Since NodeMCU has no access to astronomical time without Internet connectivity, the DreamDive mode receives timing parameters as millisecond deltas calculated by the browser client from Unix Epoch. The web interface is a Single Page Application built with Preact.js (a 3 KB lightweight React alternative) and communicates with the device via a REST API on ESPAsyncWebServer. Configuration persistence uses JSON files in SPIFFS flash memory, providing over 30 million write cycles. Functional testing confirmed correct operation across all modes, stable Wi-Fi in both AP and STA modes, concurrent multi-client support without instability, and DreamDive timing accuracy of ±1–2 seconds over a 15-minute period.*

Conclusion. *A fully functional hardware-software system for automated singing bowl playing with remote Wi-Fi control has been developed and validated. The scientific contributions are: (1) a temporal encoding method for electromagnetic actuator force as an alternative to PWM under GPIO analog output constraints; (2) the DreamDive algorithm with a linear fade profile tied to astronomical time via browser-side delta computation; (3) an RTOS-based IoT architecture resolving the concurrency conflict between network stack and actuator control tasks. Future work includes acoustic feedback via a microphone sensor, machine-learning-based rhythm pattern generation, WebSocket-based real-time state synchronization, and migration to the ESP32 platform for greater resources.*

Keywords: *singing bowl, Internet of Things, NodeMCU ESP8266, web interface, electromagnetic actuator, MOSFET, RTOS, embedded systems, stress reduction.*

Одержано редакцією 17.11.2023 р.
Прийнято до публікації 06.12.2023 р.