DOI 10.31891/2307-5732-2021-301-5-109-113
УДК 62-50:004.5

ЛІЩИНА Н.М.
Луцький національний університет
ORCID ID: https://orcid.org/0000-0002-5200-536X
e-mail: lischyna@gmail.com
ЛІЩИНА В.О.
Луцький національний університет
ORCID ID: https://orcid.org/0000-0002-2371-3850
e-mail: lvaleriy@gmail.com
ЯЩУК А.А.
Луцький національний університет
ORCID ID: https://orcid.org/0000-0003-4872-7949
e-mail: xxxxandyxxxx@gmail.com
СІВАКОВСЬКА (СУРИНОВИЧ) О.М.
Луцький національний університет
ORCID ID: https://orcid.org/0000-0002-9300-0039
e-mail: sivom@ukr.net

СИСТЕМА УПРАВЛІННЯ РОЗУМНИМ БУДИНКОМ

Здійснено огляд алгоритмів взаємозв'язку терміналу та модулів для реалізації системи «Розумний дім» з використанням операційної системи Android та розробка термінальної частини системи управління розумним будинком. Система складається з планшета з операційною системою Android, де встановлено власну розробку, а саме програму – термінал, та з різних модулів та датчиків, які базуються на використанні мікроконтролера ESP8266-01. Використання модульної системи в терміналі значно спрощує взаємодію з системою та значно збільшує швидкість виконання певних функцій та додавання функціоналу для нового модуля.

Ключові слова: «Розумний будинок», термінал, мікроконтролер ESP8266, Android.

LISHCHYNA N., LISHCHYNA V., YASHCHUK A., SIVAKOVSKA (SURYNOVYCH) O.
Lutsk National Technical University

SMART HOME MANAGEMENT SYSTEM

The purpose of this work is to review the algorithms of the relationship between the terminal and the modules for the implementation of the system "Smart Home" on the Android operating system. And the development of its own system "Smart Home". The system consists of a tablet with the Android operating system, which has its own developed program - the terminal, and various modules and sensors based on the microcontroller ESP8266-01. The main object in the development was to write universal, open and well-thought-out code, which allows you to implement any universal platform for a smart home, and at the same time wrote the minimum code for your modules. The use of a modular system in the terminal greatly simplifies the interaction with the system and significantly increases the speed of performing certain functions and adding functionality for a new module. This means that in the future it will be quite easy to add any new independent module. All elements of this system are controlled, both from the head device - the terminal, and the client (any smartphone where the special software will be installed). The use of more universal objects for embedded modules, the use of one main window for most of the implemented functionality, new functions for forming commands, the use of new libraries for any functionality in this project, such as sending broadcast messages, searching for devices on the network, internal methods for microcontroller management ware considered. The results of this work can be used for deployment systems with control of climatic parameters and energy consumption, in ordinary house, and in specialized rooms with more rigid requirements for the microclimate of the environment. In the future, it is planned to add a power consumption sensor to the system and improve sharing.

Key words: "Smart home", terminal, ESP8266 microcontroller, Android.

Постановка проблеми

Система «Розумний будинок» забезпечує нашу безпеку, комфорт та ресурсозбереження для усіх користувачів. Даними системами є можливість керувати централізовано – із пульта-дисплею, автоматично, заданими алгоритмами, або із кишенькових ґаджетів. Завдяки регулярному зростанню тарифів на різні ресурси, що необхідні для комфортного проживання у оселі, проблемам електробезпеки побутових приладів, оптимізація енергоспоживання на даний момент − є однією з основних цілей «Смарт будинку».

Головним негативним фактором новітніх систем розумних будинків є відсутність універсального стандарту, який міг би відповідати усім розумним пристроям на міжнародному ринку. При розробці «розумного будинку», на теперішній час, ви зможете наповнити його «смарт речами» від різних виробників наприклад: «Smart Things» від Google, Apple, Xiaomi. Проте більшість компаній не вважають за потрібне зробити сумісність власних пристроїв із продуктами різних компаній, і при цьому кожна компанія випускає

власний додаток для контролю модулів в «розумному будинку». Деякі з них звісно роблять відкрите API для розробників, але більшість все ж робить їх не доступними, що ускладнює розвиток технологій і їхню сумісність.

Аналіз останніх досліджень і публікацій

Аналіз літературних джерел показав існування великої кількості протоколів бездротової передачі даних для «Розумного будинку». Найбільш поширеними являються наступні бездротові протоколи: Wi-Fi, ZigBeeта MiWi, X10, ZWave, BluetoothLowEnergy (BLE).

Зростання попиту на системи «Розумний будинок», на теперішній час, реалізує вихід із таких актуальних проблем: недостатньо високий рівень сумісності та стандартизації різних протоколів; надійність даних систем; захист систем від стороннього доступу; висока ціна та складність реалізації системи для недосвідченого користувача.

Метою дослідження є огляд алгоритмів взаємозв'язку терміналу та модулів, для реалізації системи «Розумний дім» з використанням операційної системи Android та розробка термінальної частини системи управління розумним будинком.

Виклад основного матеріалу

Людина має можливість керувати розумним будинком за допомогою засобів управління, котрі знаходяться у цьому самому житлі та в його оточенні, але вмонтовані в теж саме комунікаційне середовище, що й інші загальні компоненти системи. Але реалізовано багато випадків, коли буває потрібним або бажаним дистанційне керування системами розумного житла.

Концепція розумного житла надає можливості у створенні в рамках вмонтованої системи головного управління підсистеми віддаленого керування. Дана підсистема надає можливість отримувати інформацію про події та стан деяких показників та віддалено відправляти певні команди керування усією системою або її певних елементів [1].

Відкритими системами є EIB, MODbus, BACnet / LonWorks, KNX. Людина, котра зробила свій вибір на користь відкритої системи, має широкий аспект можливостей та обладнання від різних виробників, яке, має без проблем інтегруватися у загальні інженерні та програмні комплекси, котрі без жодних адаптованих модулів-шлюзів працюють бездоганно. Окрім того, ринок підприємств котрі займаються системними інтеграторами відкритих систем уже досить масивний, що дає замовнику змогу вибору реалізатора ще на етапах самого проєктування, відладки та монтажу систем автоматизації.

Так як наш термінал розумного будинку буде працювати на операційній системі Android, то написання під неї програм здійснюється з використанням трьох мов: Java, Kotlin, C++. Kotlin – пропонує велику кількість переваг над Java для розробки JVM та Android і чудово співпрацює із Java у тих же самих проєктах. Це загальна, безкоштовна, відкрита, типізована мова програмування, спочатку розроблена для JVM (Java Virtual Machine) та Android, що поєднує об'єктно-орієнтовані та функціональні парадигми програмування.

Виберемо модулі для отримання інформації та відправки команд. В наш час є лише два мікроконтролера, які є доступними і можуть забезпечити повноцінну роботу з Wi-Fi мережею – це ESP32 та ESP8266.

Мікроконтролер ESP8266 добре відомий широкому колу розробників різних рівнів. Набори для розробки цього чипу Wi-Fi були досить відомими в останні роки.

Система взаємодії між пристроями в системі «Розумного будинку» створена на основі архітектури з центральним пристроєм-контролером (сервером). В основі цієї архітектури є те, що всі запити, які надходять від клієнта, в нашому випадку це термінал та мобільний додаток, надходять в стек обробки на сервері і тільки після цього перетворюються в команди і розсилаються до відповідних мікроконтролерів.

Термінал – це елемент, який є мозком всього будинку, він приймає, аналізує, обробляє дані та спілкується зі всіма елементами «смарт будинку». В нашому випадку – це буде планшет з операційною системою Android, на якому буде встановлено розроблений нами додаток-сервер, що буде містити в собі графічний інтерфейс та фонові сервіси для взаємодії як з користувачем, так і з смарт речами розумного будинку.

Вся взаємодія та спілкування між елементами «смарт будинку» буде відбуватися через Wi-Fi, використовуючи технологію WebSocket. WebSocket – це протокол зв'язку, який заснований на TCP протоколі та призначений для спілкування між клієнтом та сервером в режимі реального часу.

В проєкті для Android була використана бібліотека для застосування даного протоколу під назвою «Java-WebSocket» від «TooTallNate» [2]. Дана бібліотека містить базовий сервер WebSocket і реалізацію клієнта, написану на Java. Основні класи використовують реалізацію класів java.nio, що дозволяє використовувати неблокувальну модель, керовану подіями (подібно до API WebSocket для веб-браузерів).

Для реалізації WebSocket сервера було використано абстрактний клас «org.java_websocket.server.WebSocketServer», що реалізує на стороні сервера протокол WebSocket. Використання Java-Websocket дуже схоже на використання веб-сокетів javascript. Ви просто берете клієнтський або серверний клас і замінюєте його абстрактні методи відповідно до логіки додатку.

Є такі типи подій: onOpen, onMessage, onClose, onError, onStart (тільки для сервера). OnStart – викликається, як тільки сервер запущено. OnOpen – коли новий клієнт успішно підключився. OnMessage – викликається коли прийшло якесь повідомлення від будь якого клієнта, або сервера. OnClose – даний метод

є подією для моменту коли один з клієнтів відключився у зв'язку з якоюсь причиною. OnError – коли виникла якась помилка на стороні сервера, або підключення до сервера.

Оскільки, щоб клієнти могли спілкуватися між собою необхідно знати точну IP адресу сервера та порт. Оскільки за замовчуванням в локальній мережі IP адреси задає Wi-Fi роутер, то необхідно зробити так, щоб дану адресу знали всі клієнти. Є два варіанти вирішення цієї проблеми. Перший – це задати статичну IP адресу серверу, що може ускладнити подальше виготовлення системи «Розумного будинку», другий зробити певний сервіс, який би завжди говорив на якій IP адресі знаходиться даний сервер. Ми виберемо другий варіант, так як він доволі простий в написані та ніяк не впливає на виготовлення кінцевого виробу.

Ми назвали даний сервіс BroadcastListenerService, так як в його основі використовуються так звані UDP Broadcast повідомлення. UDP − один із видів протоколів в стеці TCP/IP. Він є одним з найлегших протоколів транспортного рівня моделі OSI, який обмінюється повідомленнями без авторизації та гарантії доставки. Broadcast повідомлення – це такий тип повідомлень, які відправляються на широкомовну IP адресу(*.*.*.255), дані повідомлення приймаються всіма пристроями в даній мережі. Даний сервіс прослуховує широкомовну адресу і коли надходить повідомлення з наступним вмістом: «*SMART :: get IP*», відправляє у відповідь свою IP адресу у наступному вигляді: «SMART :: my IP = X.X.X.X». Після чого пристрій при отриманні даного повідомлення парсить його та підключається вже до WebSocket сервера де і відбувається подальше спілкування.

Після чого сервер перевіряє тип пристрою за допомогою структури deviceId, так як вона є унікальною і містить в собі всі необхідні дані для того, щоб зрозуміти який вид пристрою підключається та як з ним працювати. Наприклад, «HaT:1:20201011.150720» – це є датчик першої версії, який вимірює температуру та створений 11.10.2020.

Після проходження даної ідентифікації сервер додає даний пристрій в список підключених пристроїв, який буде використовувати для подальшого спілкування з ними. Для спілкування використовуються заготовлені нами команди. Є два види команд: getData – дана команда запитує в певного пристрою про його стан, setData – команда встановлює певний стан для конкретного датчика. В даному випадку змінна «query» відповідає за тип запиту, їх є два типи перший це «*identity*», а також «*command*» цей тип просто показує, що даний запит є командою. В свою чергу «command» вже відповідає за тип команди. «*deviceId*» просто вказує, до якого девайсу звертаємося.

Дана відповідь містить в собі тип відповіді, за це відповідає змінна «*responseType*». Наступна змінна – це «*data*», саме в цій змінній знаходяться всі необхідні дані. Змінна «*status*» показує – чи коректно виконався запит. «message» – показує саме повідомлення помилки, якщо вона є. «device» – об'єкт, який показує, з якого пристрою прийшло повідомлення, для подальшої роботи і виводу на екран терміналу даних з даного повідомлення.

Об'єкт «*value*» відповідає за стан, який необхідно встановити певному датчику, для кожного пристрою є свої значення. Наприклад, даний запит встановлює відповідні значення для кожного кольору, де *r* – червоний, *g* – зелений і *b* – синій.

Оскільки за основу терміналу було обрано Android планшет, то для розробки нам необхідно завантажити Android Studio останньої версії. Дане програмне забезпечення можна завантажити з офіційного сайту Android. Після його встановлення необхідно також вставити Android SDK, починаючи з 5-ї версії Android. Пристрої на базі Android надають платформу для розробки корисних додатків із повністю відкритим вихідним кодом [5].

Після вибору пункту меню відриється вікно нового проєкту, де потрібно ввести назву додатку (Application name), посилання на сайт компанії виробника (Company domain., дане поле можна залишити порожнім), і назву пакету (Package name). Ми введемо SmartHome, порожньо, com.piedream.smarthometerminal відповідно. Так як головне вікно буде містити різні нестандартні елементи управління, то ми вибираємо «Empty Activity».

Проєктна структура містить в собі два основних пункти: App, Gradle Scripts.

Почнемо з створення сервісу для пошуку сервера. В даній папці створимо Kotlin клас з назвою «BroadcastListenerService», який буде наслідуватися від «Service». В середині даного класу створимо ще один внутрішній клас під назвою – «ServerThread», який буде наслідуватися від «Runnable». В даному класі необхідно переписати метод «run», в якому відбувається виконання всього коду в фоновому процесі. В нашому випадку ми реалізуємо сервер для приймання datagram пакетів. Також для реалізації даного функціоналу нам необхідно написати додаткові функції, які ми будемо використовувати в подальшому, це такі методи: «getIpv4HostAddress», «getBroadcastIp», «sendUDP».

Почнемо з getBroadcastIp, даний метод буде вертати текстовий рядок з IP адресою для широкомовних повідомлень. Для того щоб сформувати дану IP адресу, ми беремо поточну IP адресу та перетворюємо останній склад в 255. Тобто, наприклад якщо є IP адреса 192.168.1.120, то широкомовною адресою в даній підмережі буде 192.168.1.255.

Наступною функцією буде «getIpv4HostAddress» – дана функція отримує поточну IP адресу пристрою. Далі напишемо функцію для відправки повідомлень на широкомовну адресу. Дана функція використовує методи класу DatagramSocket та DatagramPacket.

Далі перепишемо метод run. В даному методі буде перевірка чи сокет відкрито, якщо ні, то ми його відкриваємо на 6001 порті та заносимо в глобальну змінну для подальшого використання. Також в даному

методі буде перевірка чи прийшло якесь повідомлення, якщо так, то ми перевіряємо його вміст і якщо воно буде схожим на «SMART :: get IP», то ми викликаємо метод «sendUDP», куди передаємо рядок з наступним вмістом : «SMART :: my IP = ${getIpv4HostAddress()}», також необхідно передати ip адресу: куди саме відправити дане повідомлення, тобто пристрою, від якого прийшов запит, для цього нам необхідно з класу «DatagramPacket» викликати метод «getAddress()».

Після того як пристрій отримає ip адресу, він починає підключатися до сервера. Тому наступним ми створимо клас «MainWebSocketServer», який буде наслідуватися від «WebSocketServer», та буде обробляти відповідні колбеки з класу «WebSocketListener», який необхідно створити. Даний клас буде просто інтерфейсом і буде містити п’ять методів.

Наступним ми створимо клас для обробки всіх повідомлень та взаємодії між всіма пристроями в мережі. Даний клас назвемо «MainServerService», він буде наслідуватися від Service та WebSocketListener. В даному класі нам необхідно створити глобальні статичні змінні з списком підключених пристроїв та параметром, який буде показувати чи сервіс запущено, наприклад «isAlive», для того щоб мати змогу спілкуватися з під’єднаними клієнтами та в разі завершення сервісу запустити його заново. Також необхідно створити глобальні змінні, такі як mBinder, port, mainSocket. Зміна mBinder буде використовуватися при під’єднання до сервіса. «port» буде відповідати номеру порта на якому буде працювати вебсокет сервер, в нашому випадку – це 6080. А »mainSocket» в свою чергу буде відповідати ініціалізованому класу «MainWebSocketServer».

Також даний клас буде відправляти широкомовні повідомлення та події в середині додатку, за це буде відповідає метод «sendBroadcast», він приймає назву події, та якщо необхідно – дані, які потрібно передати з цією подією.

Так як кожна кімната може містити одинакові пристрої, то необхідно створити універсальний код. Найпростішим варіантом реалізації даного функціоналу буде створення своїх кастомних віджетів (рис. 1).

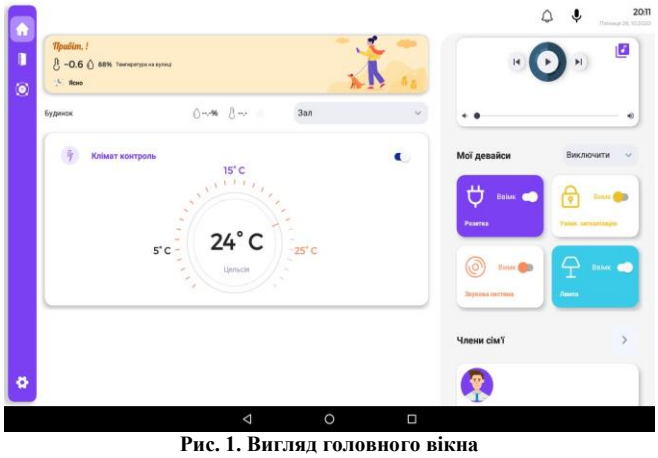

Рис. 1. Вигляд головного вікна

Тобто для прикладу створимо віджет для вимірювання температури і вологості. Для цього створимо папку «widgets», та клас «HumidityAndTemperatureWidget». Даний клас буде наслідуватися від LinearLayout, для того щоб можна було використовувати даний клас як віджет та MainBroadcastListener, для того щоб можна було отримувати повідомлення від MainServerService. Так як кожен девайс містить унікальний id, то можна фільтрувати повідомлення відповідно до віджету. Тобто необхідно створити функцію для встановлення id, для того щоб цей віджет відповідав за конкретний датчик, назвемо її «updateDeviceId». Найголовнішим методом в даному класі є «onDeviceMessageReceived», так як саме в цій функції будуть зчитуватися всі здані, які прийшли від певного датчика.

Так як пристроїв може бути багато, то ми розглянемо розробку лише одного пристрою – це буде датчик вимірювання температури та вологості. При розробці пристрою необхідно створити його принципову схему, та розробити відповідне програмне забезпечення. Почнемо з розробки принципіальної схеми, для цього потрібно вибрати електроні елементи, які задіяні. Для вимірювання температури та вологості ми вибрали DHT-11 так як він є дешевим, популярним та доволі простим в використанні.

Також використовується модуль esp8266, але не повноцінна, а урізана версія даного модуля – esp8266-01 [3]. Наступним кроком буде написання прошивки для даного модуля. Для цього нам знадобиться Arduino IDE. Далі необхідно встановити та імпортувати додаткові бібліотеки, такі як ESP8266WiFi.h; DHT.h; WebSocketClient.h; WiFiUdp.h; ArduinoOTA.h; ArduinoJson.h [4].

«Setup» функція викликається тільки раз при запуску пристрою, тому тут просто виконуються певні настройки роботи пристрою та ініціалізовуються зміні. А «loop» викликається, в свою чергу, кожен такт роботи мікроконтролера, саме тут буде відбуватися прослуховування оновлень, широкомовних повідомлень та команд від термінала та клієнта. Також далі нам необхідно написати, за яким принципом буде працювати wifi модуль в даному мікроконтролері, в нашому випадку це WIFI_STA. Після чого оголосимо DHT змінну, куди необхідно передати, до якого піну підключено датчик та версію датчика, в нашому випадку – це 2 та DHT11 відповідно. Для успішного обміну даними, також необхідно написати код для успішного підключення до wi-fi мережі, після успішного підключення можна продовжувати виконання коду, інакше перезапустити модуль і почати виконання коду з самого початку, тобто це необхідно для переініціалізації модуля та повторної спроби перепідключитися до мережі.

Після написання коду необхідно прошити модуль даною прошивкою. Для цього в Arduino IDE є всі необхідні компоненти. Мікроконтролер можна прошити двома способами, або через спеціальну схему для прошивання, або через Arduino (рис. 2) [6].

За аналогією розроблено такі модулі, як: модуль відкривання дверей, модуль виявлення руху, модуль керування розеткою, модуль управління світильником із можливістю задання кольору.

Модуль відкривання дверей реалізований завдяки датчику, який розроблений саме для цієї цілі. Цей датчик складається з двох частин: в першій знаходиться магніт, а в другій – геркон. Він спрацьовує при віддаленні датчику від магніту приблизну на 20-30 міліметрів, після чого в герконі розмикаються контакти та розривається електричне коло, це фіксує мікроконтролер та відправляє сигнал на центр керування.

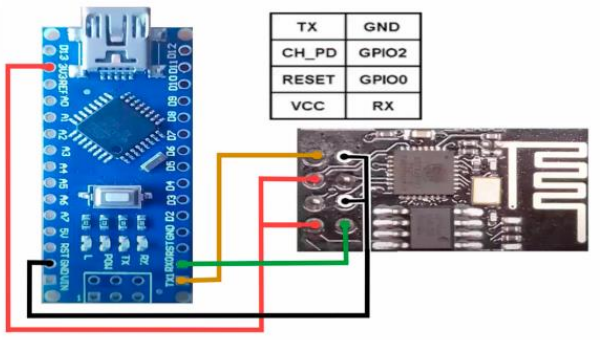

Рис. 2. Прошивка мікроконтролера

Модуль виявлення руху реалізований за допомогою спеціалізованого датчика. Він працює за принципом пасивного інфрачервоного сигналу. Коли у області, яка контролюється, рухається живий об'єкт, який виділяє тепло, то лінза Френеля фокусує дані зміни на різних областях чутливого напівпровідникового елемента, що також називають піроелектричним елементом.

Модуль для управління світлом складається з трьох компонентів: сам мікроконтролер, RGB світлодіод та фільтр живлення. Сам діод складається з 4 виводів, один з яких – це живлення, наступні ж відповідають за певний колір.

Висновки

Розглянуто використання більш універсальних об'єктів для вбудованих модулів, використання одного головного вікна для більшої частини реалізованого функціоналу, нові функції для формування команд, використання нових бібліотек для будь-якого функціоналу в даному проєкті, такі як надсилання широкомовних повідомлень, пошук пристроїв в мережі, внутрішні методи для керування мікроконтролером.

Універсальний, відкритий та продуманий код дає змогу реалізувати будь-кому універсальну платформу для розумного будинку, пишучи мінімальний код для своїх модулів. Також використання модульної системи в терміналі значно спрощує взаємодію з системою та значно збільшує швидкість виконання певних функцій та добавлення функціоналу для нового модуля. Сучасний підхід до графічного інтерфейсу дає змогу розібратися будь-кому з всіма можливостями розумного будинку. Нова форма для всіх команд, яка зменшує розмір самого повідомлення, що значно впливає на швидкість пересилання його.

Література

1. IoT-проект для умного дома [Електронний ресурс]. – 2016. – Режим доступу : https://habr.com/ru/company/intel/blog/396737

2. Маклин Д. Разработка приложений для Android : навч. посібник / Д. Маклин, С. Хашими, С. Коматинени. – Пітер, 2011. – 185 с.

3. ESP8266 WiFi library [Електронний ресурс]. – Режим доступу : https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html

4. Среда разработки Arduino [Електронний ресурс]. – Режим доступу : http://arduino.ru/Arduino_environment

5. Ліщина Н.М. Аналіз середовищ реалізації додатка для дистанційного керування системою «Smart House» / Н.М. Ліщина, Р.Б. Дунець, М.Б. Мальчевський // Тези доповідей VIII Міжнародної науково-практичної конференції «Інформаційні технології в освіті, науці і виробництві (ІТОНВ-2021)» (21-22 травня 2021 року). – Луцьк : Відділ іміджу та промоції Луцького НТУ, 2021. С.127-130.

6. Повстяна Ю.С. Система радіоелектронної боротьби на базі ARDUINO UNO R3 / Ю.С. Повстяна, А.А. Ящук, В.О. Ліщина, М.М. Поліщук, М.І. Потейчук // Комп'ютерно–інтегровані технології: освіта, наука, виробництво : науковий журнал. – Луцьк : Видавництво ЛНТУ, 2020. – Випуск № 38. – С. 10–14.

References

1. IoT-proekt dlia umnoho doma [Elektronnyi resurs]. – 2016. – Rezhym dostupu : https://habr.com/ru/company/intel/blog/396737

2. Maklyn D. Razrabotka prylozhenyi dlia Android : navch. posibnyk / D. Maklyn, S. Khashymy, S. Komatyneny. – Piter, 2011. – 185 c.

3. ESP8266 WiFi library [Elektronnyi resurs]. – Rezhym dostupu : https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html

4. Sreda razrabotky Arduino [Elektronnyi resurs]. – Rezhym dostupu : http://arduino.ru/Arduino_environment

5. Lishchyna N.M. Analiz seredovyshch realizatsii dodatka dlia dystantsiinoho keruvannia systemoiu «Smart House» / N.M. Lishchyna, R.B. Dunets, M.B. Malchevskyi // Tezy dopovidei VIII Mizhnarodnoi naukovo-praktychnoi konferentsii «Informatsiini tekhnolohii v osviti, nautsi i vyrobnytstvi (ITONV-2021)» (21-22 travnia 2021 roku). – Lutsk : Viddil imidzhu ta promotsii Lutskoho NTU, 2021. S.127-130.

6. Povstiana Yu.S. Systema radioelektronnoi borotby na bazi ARDUINO UNO R3 / Yu.S. Povstiana, A.A. Yashchuk, V.O. Lishchyna, M.M. Polishchuk, M.I. Poteichuk // Kompiuterno–intehrovani tekhnolohii: osvita, nauka, vyrobnytstvo : naukovyi zhurnal. – Lutsk : Vydavnytstvo LNTU, 2020. – Vypusk № 38. – S. 10–14.

Рецензія/Peer review : 26.09.2021 р. Надрукована/Printed :10.10.2021 р.