DOI: 10.31319/2519-8106.1(50)2024.304856
УДК 004.94

Peremitko Mykhailo, PhD student, Department of Mathematical Modeling and System Analysis
e-mail: mikhailperemitko@gmail.com
Перемітько М.В., здобувач третього (доктора філософії) рівня вищої освіти, кафедра математичного моделювання та системного аналізу

Nadryhailo Tetiana, Candidate of Technical Sciences, Associate Professor of the Department of Mathematical Modeling and System Analysis
Надригайло Т.Ж., кандидат технічних наук, доцент кафедри математичного моделювання та системного аналізу
ORCID: 0000-0003-1239-5946
e-mail: ntatiana62@gmail.com

Дніпровський державний технічний університет, м. Кам'янське
Dniprovsky State Technical University, Kamianske

COMMUNICATION BETWEEN REACT NATIVE MOBILE APPLICATIONS AND BLUETOOTH LOW ENERGY DEVICES

КОМУНІКАЦІЯ МОБІЛЬНИХ ДОДАТКІВ НА REACT NATIVE ТА BLUETOOTH LOW ENERGY ПРИСТРОЇВ

Today, smartphones are essential for communication, info access, and work flexibility. They integrate with various devices like fitness trackers, enabling health monitoring and home automation. This sync, often via Bluetooth Low Energy (BLE) tech, conserves energy and expands smartphone functions. Meanwhile, React Native is an optimal choice for cross-platform software development. These technologies facilitate efficient wireless interaction between smartphones and other smart devices. This research is focused on the overview of these technologies and dedicated React Native tools for BLE interaction handling.

***Keywords**: smartphone, mobile application, smart device, Bluetooth Low Energy, React Native.*

У сучасному світі смартфони відіграють важливу роль, ставши невід'ємним елементом повсякденного життя. Вони є основним засобом спілкування, доступу до інформації, розваг, а також забезпечують можливості для роботи та навчання в будь-якому місці і в будь-який час.

Окрім смартфонів, суспільство все частіше користується додатковими пристроями, такими як фітнес-трекери, смарт-годинники, ваги, медичні пристрої та інші гаджети Інтернету речей (IoT). Окрім незалежної роботи окремо ці пристрої взаємодіють і зі смартфонами у тому числі. Ця інтеграція дозволяє користувачам контролювати своє здоров'я та фізичну активність, отримувати важливі сповіщення, керувати побутовою технікою та автоматизувати різні аспекти повсякденного життя через взаємодію з різними пристроями. Наприклад, смартфон може синхронізуватись з фітнес-трекером для відстеження кроків та серцевого ритму, або зі домашнім пристроєм для керування освітленням, опаленням чи системою безпеки. Така взаємодія розширює можливості смартфонів і сприяє утворенню екосистеми синхронізованих пристроїв.

Подібна синхронізація смартфонів та наведених пристроїв в переважній більшості випадків досягається за допомогою технології BLE (Bluetooth Low Energy), бездротової технології зв'язку, що розроблена для енергоефективного обміну даними між пристроями на короткій відстані. Ця технологія є оптимальною з точки зору споживання енергії та дозволяє передавати дані на відстань до 10 метрів з невеликою швидкістю. Оптимізована робота з точки споживання енергії є ключовим фактором у подібних інтеграціях через обмежену ємність акумуляторів, вбудованих у розумні пристрої.

У свою чергу, React Native є раціональним вибором в якості технології для розробки програмного забезпечення для смартфонів з різними операційними системами. Кросплатфор-

мний підхід, що використовується у цьому інструменті дозволяє оптимізувати розробку за рахунок можливості використовувати одну реалізацію на кількох платформах.

За допомогою технологій, зазначених вище, можливо оптимальним шляхом створити програмне забезпечення для бездротової взаємодії між смартфоном та іншими розумними пристроями.

***Ключові слова**: смартфон, мобільний додаток, розумний пристрій, Bluetooth Low Energy, React Native.*

Problem's Formulation

Software development for mobile devices does not lose its relevance, but only gains momentum in its development. The toolkit is becoming more extensive, and an increasing number of technologies for creating such software are emerging. Among the available solutions, React Native occupies a significant position. It is an open platform for mobile application development that allows the creation of cross-platform applications, i.e., those that are compatible with multiple operating systems. Researching and using such tools is important because of the efficiency they provide. In particular, the primary advantage is that the same code can be utilized to develop applications for both the Android and iOS platforms, with the objective of incorporating as much of the common code as possible. Consequently, the support and maintenance of the software are also simplified.

As the prevalence of Internet of Things (IoT) devices in everyday life continues to grow, there is a pressing need to develop software that ensures seamless and efficient wireless interaction between these devices and mobile applications on smartphones. In this regard, it is crucial to conduct a comprehensive examination of the available solutions and tools in the context of React Native technology and Bluetooth Low Energy devices, their interaction, and the potential for communication between them.

Thus, it is important to study the available solutions and tools in terms of React Native technology and devices with Bluetooth Low Energy support, their interaction and the possibility of communication between them.

Analysis of recent research and publications

The problems of using Bluetooth Low Energy and the implementation of interaction between devices are considered in the materials of A. Pakula and E. Palamarchuk [1], the review of the technology is made from the point of view of the Android operating system and the available support of this system for this type of wireless communication.

In the article by A. Ozerchuk [2], the Bluetooth Low Energy technology, its main components, energy efficiency issues, and the method of communication between devices that support this technology are considered in more detail.

Nevertheless, the study of such integration from the perspective of different mobile platforms and React Native technology is not yet sufficiently covered. There is a need for a more in-depth analysis of how React Native interacts with Bluetooth functionality on iOS and Android platforms, particularly in the use of their Bluetooth Low Energy (BLE) Application Programming Interfaces (APIs). In addition, it is important to identify which BLE features can be effectively used with React Native, as well as the challenges that arise when developing cross-platform applications using this technology.

Formulation of the study purpose

The objective of this research is to examine the efficacy of software tools for establishing a wireless connection between React Native mobile applications and Bluetooth Low Energy-enabled smart devices.

Presenting main material

It is crucial to begin by defining the primary objective of utilizing React Native as a mobile development tool instead of traditional methods, such as those employed in native mobile application development.

Native mobile application development is a conventional approach to creating applications that employs the programming language and software architecture of the platform, or operating system, to develop a code base that is then compiled into native applications compatible with a single platform. Although this methodology has been employed since the advent of smartphones, the tools and technologies have evolved over time. For Android, the principal programming language was previously Java or C++, but since 2017, Android has also supported the comparatively more recent and more flexible Kotlin language. In turn, Objective-C was the sole language for iOS development until 2014, when Apple introduced its Swift programming language for developers [3].

Developing native apps not only forces developers to learn and use separate programming languages for each operating system, but also encourages them to use the recommended software architecture of each platform. For iOS, the recommended architecture is Model-View-Controller (MVC), while Android applications typically use the Model-View-ViewModel (MVVM) architecture. As a result, the codebases of native applications have very little in common with each other. Consequently, knowledge of one codebase provides only limited insight into the other.

The strongest argument for the power of native mobile app development is user experience and speed reaction. Because developers write native code, they can use the platform's APIs (application programming interfaces) and support libraries, and create user interface mockups using specific and optimized interface components provided by the platform. As a result, with the help of these technologies it is possible to achieve the best results in terms of optimization, speed and user experience.

The main problem in the development of native applications is, of course, separate code bases. Developing the same application for multiple platforms requires knowledge of platform-specific development methods and technologies. Developing and maintaining native mobile apps for multiple platforms is comparable to running multiple separate software projects simultaneously, often requiring more developers and other project personnel than a single app project. As the number of supported platforms increases, the cost and effort of development and subsequent support and maintenance also rise.

To solve this problem, cross-platform development tools were created. Cross-platform frameworks allow developers to write source code in a common programming language, which is then compiled into native binaries for each platform using a cross-compiler. The main job of the cross-compiler is to specify which native UI components and platform features to use. An app developed using a cross-platform framework can have a very close user experience to a full native mobile app, but there are exceptions.

The main advantages of cross-platform frameworks are performance and user experience. The user interface is displayed on every device using native components, so the app will look native if the developers take into account the design features of the specific platform. While no cross-platform framework can fully compete with the speed of full native apps, tests and research show that such frameworks are optimized enough to match the level of apps built with native tools in the vast majority of use cases.

One of these frameworks is React Native, which has proven itself as one of the most stable and optimal tools for creating cross-platform software for mobile devices.

React Native is an open source software created by Facebook that allows developers to create mobile applications using JavaScript and React. The main idea behind React Native is to allow developers to create mobile applications using the same principles as for web development with React, but with the ability to deploy on iOS and Android platforms [3].

One of the main advantages is also that React Native takes a declarative approach to user interface development. Developers describe what the interface should look like in different states, and React Native automatically manages to update the display when the app state changes. This simplifies software development and maintenance.

Additionally, React Native has a large developer community that is constantly growing. This means that developers can easily find solutions to their problems, consult and share experiences with other community members. In turn, the presence of such a large and active community contributes to the existence and further development of a wide range of third-party libraries and modules available for React Native. This allows developers to easily and quickly integrate various features and functionality into their applications without having to build everything from scratch.

In order to understand how the framework can interact with Bluetooth Low Energy technology and communicate with other devices wirelessly, it is necessary to have an idea about the general framework architecture.

The React Native architecture includes several core components, including a JavaScript virtual machine, a React Native bridge, and native modules. The application code runs on the JS virtual machine along with any third-party libraries used. Calls from native modules are routed through the React Native bridge to native APIs and third-party libraries, and results are passed back through the bridge as needed. It allows using JavaScript to develop applications while using native user interface components and platform capabilities [4].

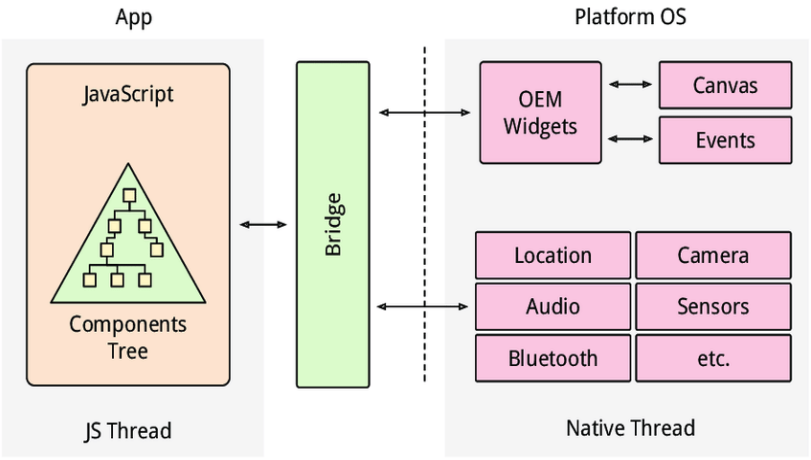

Fig. 1. A diagram of the main components of the React Native architecture

Next, more details about each of the components, their connection and interaction are shown in the diagram (Fig. 1).

The implementation of React Native applications is based on the use of a built-in JavaScript virtual machine (JS VM), which executes the application's JavaScript code. The JS VM is responsible for executing application logic, rendering components, and managing application state. React Native uses JavaScriptCore for iOS and JavaScriptCore or V8 for Android.

The bridge is one of the key components of the React Native architecture. It provides interaction between the JavaScript code of the application and the native components of the platform. The bridge is responsible for forwarding calls and events between JavaScript and native code.

In turn, native modules are parts of functionality implemented in the programming languages of a specific platform (Objective-C for iOS, Java or Kotlin for Android). React Native provides a way to call these native modules from JavaScript code, allowing you to use platform features such as camera, geolocation, Bluetooth, and more.

It is important to note that at the time of writing, the architecture is undergoing changes as the new modified architecture is gradually being implemented in new versions of the framework. However, it has not yet acquired the status of the main and stable architecture. Consequently, the vast majority of applications that have been created and are being created continue to work with the support of the original architecture.

Thus, to access the Bluetooth Low Energy functionality, appropriate native modules are used that allow making calls to system functions responsible for establishing a wireless connection and data transfer on each of the mobile operating systems.

Thanks to a large and active community of developers, several library integrations of native modules have been created to date, allowing you to use the Bluetooth Low Energy software interface of each of the Android and iOS mobile operating systems. Another great advantage is that these libraries are open source. Open source libraries are software components or modules that are distributed with open source code and are available for use, modification and further distribution under the terms of a license that allows free use and modification. The implementation of functionality in such libraries is completely transparent and understandable. In the event that the necessary functions are not available, it is always possible to add the necessary parts yourself if they are verified by the developers responsible for this library and its support.

Before moving on to specific examples of such modules for working with Bluetooth Low Energy, it is worth providing general information about the wireless technology itself and its features.

Bluetooth Low Energy (BLE) is a wireless technology that was developed by the Bluetooth Special Interest Group (SIG) for short-range communication over short distances. Unlike previous versions of Bluetooth, BLE was specifically designed as a low-power technology that can be used independently. Although it uses the Bluetooth brand and uses many elements of its parent technology, Bluetooth Low Energy should be seen as a distinct technology with different specifics and goals.

The Bluetooth Low Energy (BLE) architecture is based on the concept of a client-server model using the Generic Attribute Profile (GATT).

Among the main components of the technology, the following can be distinguished (Fig. 2).

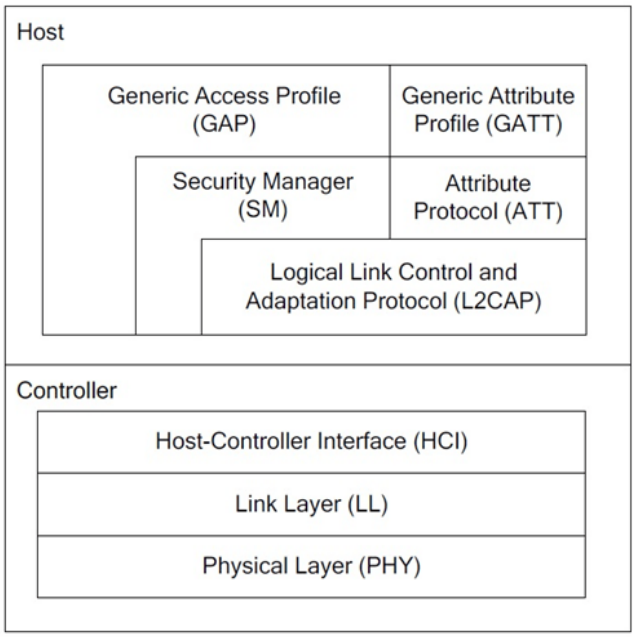

Fig. 2. Diagram of the main components of the Bluetooth Low Energy architecture

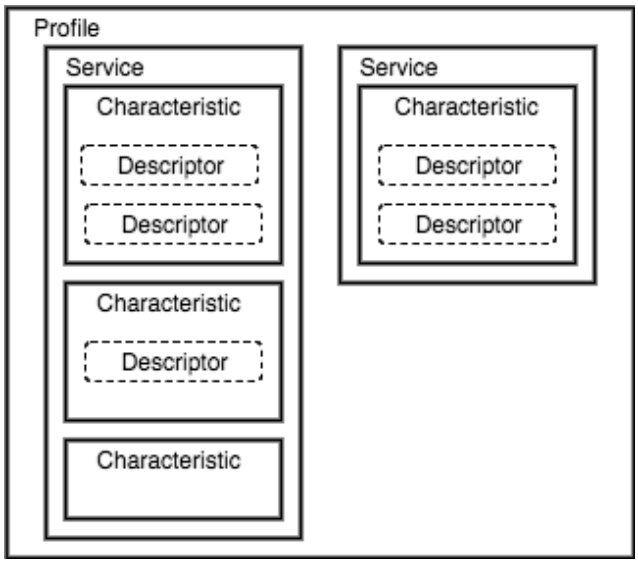

Fig. 3. Diagram of the main components of the GATT profile

*Physical Layer (PHY):*The BLE physical layer defines the characteristics of the radio frequency communication, such as frequency, modulation, and signal strength.

*Link Layer (LL):*The communication layer is responsible for connection management, data transfer and error control.

*Host Controller Interface (HCI):*An interface that provides communication between the Bluetooth controller and higher levels of the protocol stack.

*Logical Link Control and Adaptation Protocol (L2CAP):*A protocol that provides multiplexing and management of data flows between different applications [5].

Interaction between smartphones and other devices takes place according to the GATT protocol. GATT (Generic ATTribute Profile) is an abbreviation that stands for General Attribute Profile, and defines the way of data exchange between two Bluetooth Low Energy devices using concepts called services and characteristics (Fig. 3). It uses a common data transfer protocol called the Attribute Protocol (ATT), which is used to store services, characteristics and related data in a simple table using 16-bit identifiers for each entry in the table [5].

GATT is used after a persistent connection has been established between two devices, which means you've already gone through the GAP-driven advertisement process.

It is important to note that in the GAT protocol, connections are exclusive. This implies that a BLE peripheral can only be connected to a single central device (e.g., mobile phone) at a time. Once a peripheral is connected to a central device, it ceases to broadcast its presence and other devices are unable to detect it or establish a connection until the existing connection is terminated.

Establishing a connection is the only way to allow bidirectional communication, where a central device can send meaningful data to a peripheral device and vice versa.

The following procedure is usually used for interaction between a smartphone and a smart device via BLE.

Advertising: The smart device sends advertising packets to notify the smartphone of its presence.

Connection Establishment: The smartphone sends a connection request to the smart device.

*Interaction through GATT:*After a successful connection, the smartphone and the smart device communicate using GATT. A smartphone can read or write the characteristics (attributes) of a smart device, for example, receive data from sensors or control various functions.

Among the available libraries compatible with React Native there are 2 variants: react-native-ble-manager and react-native-ble-plx. Both libraries focus on providing access to Bluetooth Low Energy interoperability system functions through the integration of appropriate native modules. Each of these libraries has been maintained and improved for the last 8 years, but over the last year, the frequency of use of react-native-ble-manager has increased significantly compared to the competing module (Fig. 4). It is important to choose such a library, which is used by a larger number of people, accordingly, support and identification of potential problems in the work will also be more active. But in this case, it is worth noting that both solutions are proven years of use by many developers, regardless of the current popularity of this or that library.

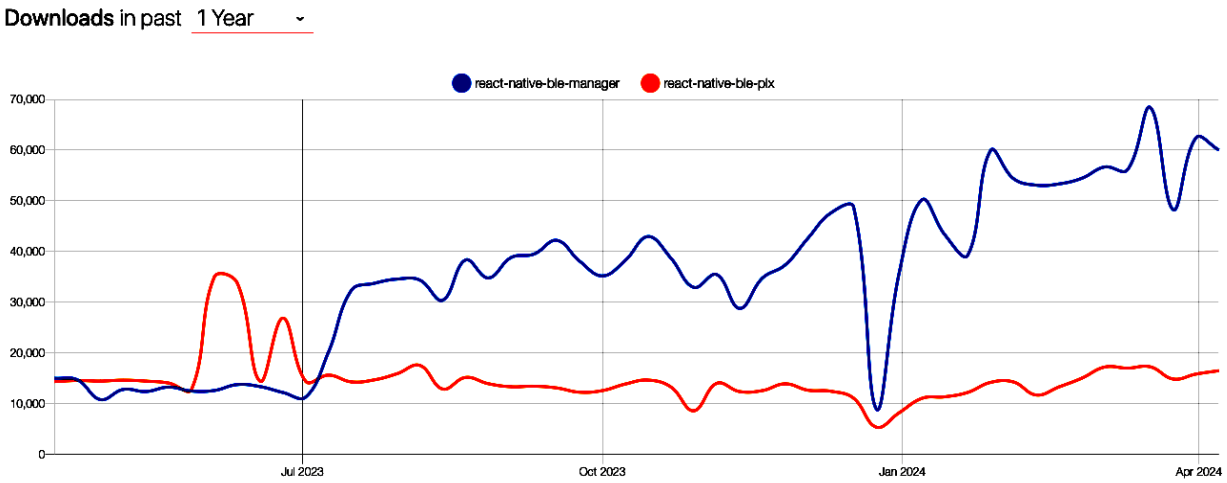

Fig. 4. Graph of the number of downloads of the react-native-ble-manager and react-native-ble-plx libraries over the last year

The react-native-ble-plx library supports everything you might expect: scanning, connecting, communication, running in the background, filtering devices by name and ID, checking package integrity for multi-device connections, discovering services and features. This solution provides a wide range of functions for a finer work with Bluetooth Low Energy. It is also worth noting the good documentation of all available functions and examples of their use.

The main elements of the react-native-ble-plx software interface include the following functions:

1. Track the current state of the Bluetooth connection using bleManager.state() and onStateChange().
2. Scan surrounding devices using bleManager.startDeviceScan() and bleManager.stopDeviceScan().
3. Establishing communication with surrounding devices using bleManager.connectToDevice() and device.cancelConnection().
4. Obtaining information from the device about available services and characteristics of its profile using bleManager.discoverAllServicesAndCharacteristicsForDevice() and bleManager.servicesForDevice().
5. Read and write data by modifying characteristics with device.readCharacteristicForService() and device.writeCharacteristicWithResponseForService().
6. Receiving notification of characteristic change using device.monitorCharacteristicForService().

In turn, the react-native-ble-manager library also provides all the necessary functionality for working with Bluetooth Low Energy devices. It is not targeted at more complex usage scenarios such as multi-device connections, guaranteed transactions, or data packet processing. But the functionality of this library is well suited to most typical application needs. There is also built-in support for working with beacons on both iOS and Android.

The main elements of the react-native-ble-manager software interface include the following functions:

1. Scan surrounding devices using BleManager.scan(), BleManager.stopScan(), and BleManager.compassionScan().
2. Establishing communication with surrounding devices using BleManager.connect() and BleManager.disconnect().

3. Track the current Bluetooth connection status with *BleManager.enableBluetooth()*and BleManager.checkState().
4. Read and write data by modifying characteristics with BleManager.read(), BleManager.write(), and BleManager.writeWithoutResponse().
5. Receive notifications about changes in characteristics using *BleManager.startNotification()* and BleManager.stopNotification().

Given the above basic functionality of both libraries, it can be assumed that their software interfaces are very similar, therefore, in the case of the need to replace one with another during development, the migration should be quite trivial. In summary, both tools are up-to-date, actively supported by the community, and provide all the necessary functionality to work with Bluetooth Low Energy in the React Native ecosystem. The choice of one or another library may be due to ease of use or the need for certain exclusive functionality.

Conclusions

Achieving communication between mobile applications built with React Native and smart devices with Bluetooth Low Energy support is possible using two different open source libraries: react-native-ble-plx and react-native-ble-manager. Together with the React Native framework, these libraries provide opportunities to optimize software development due to simultaneous compatibility with both iOS and Android mobile operating systems. This toolkit is worthy of attention and further research to implement more efficient approaches in mobile software development.

References

[1] Pakula A.A., Palamarchuk Ye.A. (2022) Vykorystannya tekhnolohiyi Bluetooth Low Energy dlya rozumnykh prystroyiv v mobil'niy rozrobtsi, *Materialy XV konferentsiyi «Informatsiyni tekhnolohiyi i avtomatyzatsiya - 2022», Odesa, 20-21 zhovtnya 2022 r.*, 166-168.
[2] Ozerchuk I.M. (2023) Bluetooth Low Energy, yak osnova peredachi danykh pry nadmalomu enerhosporozhivanni, *Naukovyy zhurnal "Kompyuterno-intehrovani tekhnolohiyi: osvita, nauka, vyrobnytstvo" vypusk № 51, Luts'k,* 174-180.
[3] Bezverkhiy.O, Kutsenko O. (2021) Rozrobka krosplatformennykh dodatkiv, *Materialy VII ISPC Transfer of Innovative Technologies*, *Kyiv*, 102-105.
[4] Domarats'kyy I.V., Bahnyuk N.V., Bortnyk K.Ya., Tyshchuk I.V. (2023) Zasoby rozrobky krosplatformennoho mobil'noho dodatku, *Naukovyy zhurnal "Kompyuterno-intehrovani tekhnolohiyi: osvita, nauka, vyrobnytstvo" vypusk № 53, Luts'k*, 111-116.
[5] Carles Gomez, Joaquim Oller, Josep Paradells (2012) Overview and evaluation of Bluetooth Low Energy, *An Emerging Low-Power Wireless Technology*, *Sensors*, *12*, 11734-11753.

Список використаної літератури

1. Пакула А.А., Паламарчук Є.А Використання технології Bluetooth Low Energy для розумних пристроїв в мобільній розробці // Матеріали XV конференції «Інформаційні технології і автоматизація - 2022» , Одеса, 20-21 жовтня 2022 р. – 2022, с.166-168.
2. Озерчук І.М. Bluetooth Low Energy, як основа передачі даних при надмалому енергоспоживанні // Науковий журнал "Комп’ютерно-інтегровані технології: освіта, наука, виробництво" випуск № 51, Луцьк, 2023, с.174-180.
3. Безверхий.О, Куценко О. Розробка кросплатформенних додатків // Матеріали VII ISPC Transfer of Innovative Technologies 2021, Київ, с.102-105.
4. Домарацький І.В., Багнюк Н.В., Бортник К.Я., Тищук І.В., Засоби розробки кросплатформенного мобільного додатку // Науковий журнал "Комп’ютерно-інтегровані технології: освіта, наука, виробництво" випуск № 53, Луцьк, 2023, с.111-116.
5. Carles Gomez, Joaquim Oller, Josep Paradells Overview and evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology // Sensors, 12, 2012, с.11734-11753.

Надійшла до редколегії 08.04.2024