Journal of Edge Computing, **2024**, 3(2), pp. 126-135 https://doi.org/10.55056/jec.745

Advancing IoT interoperability: dynamic data serialization using ThingsBoard

Dmytro I. Shvaika[1,2], *Andrii I.* Shvaika[1,2] and *Volodymyr O.* Artemchuk[1,3,4,5]

[1]*G.E. Pukhov Institute for Modelling in Energy Engineering of the National Academy of Sciences of Ukraine, 15 General Naumov Str., Kyiv, 03164, Ukraine*

[2]*ThingsBoard, Inc., 110 Duane Str., Suite 1C, New York, 10007, USA*

[3]*Center for Information-analytical and Technical Support of Nuclear Power Facilities Monitoring of the NAS of Ukraine, 34a Palladin Ave., Kyiv, 03142, Ukraine*

[4]*Kyiv National Economic University named after Vadym Hetman, 54/1 Peremohy Ave., Kyiv, 03057, Ukraine*

[5]*National Aviation University, 1 Liubomyra Huzara Ave., Kyiv, 03058, Ukraine*

Abstract. Benchmarking leading data serialization protocols such as schemaless JSON with binary serialization formats demonstrates the superior performance of the latter in Internet of Things (IoT) ecosystems. However, ease of integration and maintenance are equally important factors for real-world applications. IoT developers choose schemaless JSON formats for primary serialization because of their user-friendliness. However, interest in using Protocol Buffers directly at the device level in Internet of Things ecosystems is growing. Many IoT devices now transfer data exclusively via Protobuf, while others are switching to this format to improve efficiency and reduce network load. However, the static nature of Protobuf requires constant developer intervention, which undermines the scalability and versatility of the platforms, especially in cloud deployments. We explore the challenges of integrating devices that communicate exclusively through Protobuf into IoT platforms using the ThingsBoard as an example. Our study proposes a dynamic method for integrating new Protobuf-compatible devices by automating the compilation of the scheme into the platform's code base. This approach aims to simplify integration and maintenance, which, in addition to productivity, are key factors in operating efficiency in IoT environments.[1]

Keywords: IoT platform ThingsBoard, data serialization, protocol buffers

1. Introduction

The Internet of Things (IoT) is gaining more and more popularity [17]. The study by Debnath and Chettri [2] and Villamil, Hernandez and Tarazona [24] highlights various IoT applications, including in industry, business, and improving the quality of life. Uckelmann, Harrison and Michahelles [23] emphasize the potential of IoT to revolutionize business processes. Porkodi and Bhuvaneswari [16] provides a detailed overview of communication technology standards in the IoT, such as RFID tags and sensors. A study by Khang et al. [10] addresses the limitations

[1]This article is an extended version of a conference talk presented at the 4th Edge Computing Workshop (doors 2024) [18].

shvaikad@gmail.com (D. I. Shvaika); andrew.shvayka@gmail.com (A. I. Shvaika); ak24avo@gmail.com (V. O. Artemchuk)

https://www.researchgate.net/profile/Volodymyr-Artemchuk-2 (V. O. Artemchuk)

0009-0001-3088-3997 (D. I. Shvaika); 0009-0006-8461-3550 (A. I. Shvaika); 0000-0001-8819-4564 (V. O. Artemchuk)

of single-channel communication in hydroponic systems, emphasizing the need for reliable multi-channel communication in IoT-based monitoring systems. Alsayaydeh et al. [1] has developed a programmable home security system using GSM technology with a Raspberry Pi 3 as the main controller and a passive infrared sensor (PIR) to detect intruders.

Effective data management ensures seamless communication and operation in increasingly complex networks. As IoT ecosystems evolve, the demands for more sophisticated data serialization methods that improve performance and simplify integration and maintenance become critical.

While widely used serialization protocols such as JSON provide simplicity and user-friendliness, their limitations in dealing with large-scale IoT applications become apparent. Meanwhile, binary serialization formats such as Protocol Buffers (Protobuf) offer significant performance improvements through compact and efficient data representation. However, integrating Protobuf into existing IoT platforms presents substantial challenges due to its static nature, which requires frequent manual intervention by developers to update the system and integrate devices, especially in scalable cloud environments.

This study examines the new trend of using Protobuf directly at the device level. It solves the double problem of increasing the efficiency of data transmission and reducing the load on the network.

The IoT platform ThingsBoard was chosen as the research tool. ThingsBoard, Inc. was founded in 2016 by a team of programmers from Ukraine and specializes in developing software products for IoT [22]. Recent study by Di Felice and Paolone [3] have highlighted its vital role in the academic world. The authors describe ThingsBoard as an advanced open platform for IoT, noting its extensive functionality and documentation. They analyzed 55 articles in the Scopus database, finding various applications of ThingsBoard in different fields and confirming its versatility and effectiveness for IoT-related projects.

Focusing on ThingsBoard, we offer a dynamic Protobuf schema compilation solution. Our research aims to demonstrate a real-time static schema integration solution that should increase IoT systems' overall adaptability and scalability.

This paper has the following structure: Section 2 presents a brief comparative analysis of the leaders of schemaless and binary data serialization protocols; Chapter 3 explores the challenges of Protobuf device integration in IoT platforms; Chapter 4 describes the dynamic compilation of Protobuf in ThingsBoards. Finally, Chapter 5 is Discussion. Section 6 concludes the paper.

2. Comparative analysis of data serialization protocols for IoT

The advancement of IoT systems is accelerating rapidly. In 2019, there were approximately 8.6 billion connected devices, and the forecast for 2030 is 29.42 billion (figure 1) [19]. These numbers underscore IoT technology's unprecedented growth and potential impact in the coming years.

When integrated with an IoT platform, a standard device transforms into an IoT device, allowing it to exchange data with other IoT devices or cloud servers. This necessitates a standardized data exchange format at the application level. To address this challenge, libraries offering standardized data formats are readily accessible. However, the costs related to data (de)serialization and transmission with these libraries are largely undocumented in the realm of

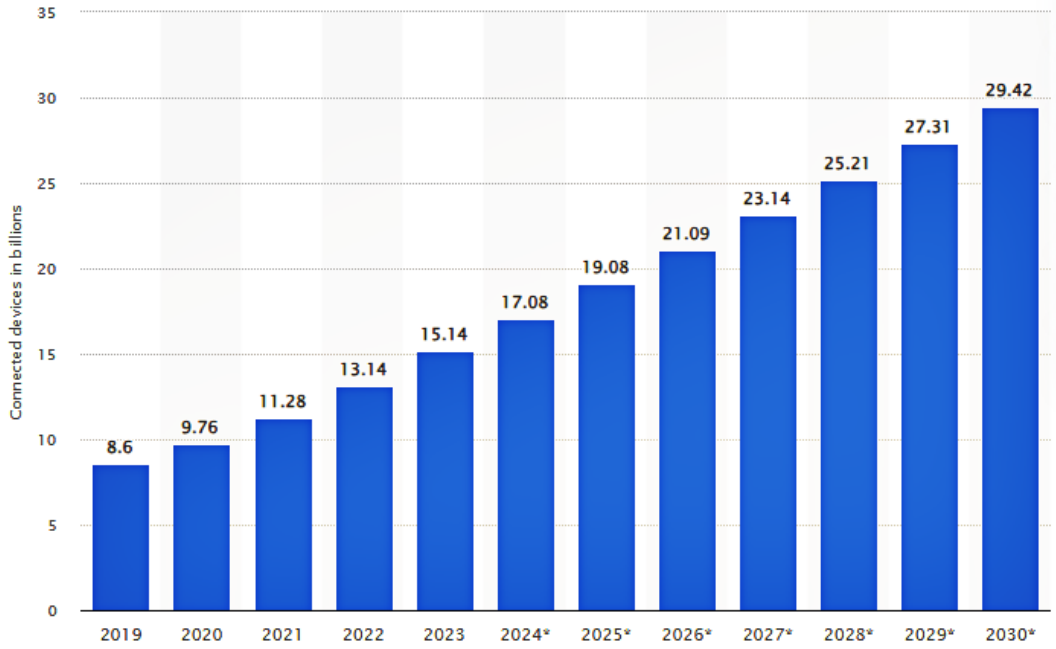

Figure 1: Number of IoT connected devices. Data marked with (*) are predicted [19].

IoT or documented in limited capacities for specific protocols [4, 6, 11].

JavaScript Object Notation (JSON) [9] is a data-interchange format easily readable by humans. The high rate of flexibility, combined with a lightweight structure, avoids unnecessary overhead associated with the benefits of standardization. It is based on a subset of the JavaScript programming language to be parsed correctly by JavaScript compiler [25].

The effectiveness of using Google's Protocol Buffer (Protobuf) [8] as a processing protocol and communication standard in the transportation domain of the Internet of Things is evaluated in the study Popić et al. [15]. Protobuf is a strong candidate for standardized communication methods in the Internet of Things industry, mainly due to its reduced network overhead.

The study by Friesel and Spinczyk [6] evaluated the efficiency of JSON encoding by comparing it with other serialization formats in the IoT framework. The results underscored the efficacy of Protobuf, emphasizing their suitability for energy-efficient data serialization in the context of contemporary, high-capacity IoT devices.

In the Luis et al. [11] study, the focus was on assessing the performance metrics of PSON, comparing it against a spectrum of formats, including Protobuf. This comprehensive analysis covered various dimensions such as serialization/deserialization velocities, binary file dimensions, and encoding sizes. The benchmarking reveals that Protobuf leads in data serialization for IoT due to its high efficiency in size and speed, broad language support, and extensibility.

Despite the rapid development and potential advantages of formats such as PSON, Protobuf's continued use in various IoT applications reaffirms its importance.

A comprehensive overview of the JSON-compliant binary serialization specification, analyzing serialization's history, evolution, and technical details from the 1960s to recent developments, is provided in Viotti and Kinderkhedia [25]. The article takes a detailed look at schema-driven and non-schema-driven binary serialization, with a particular focus on how JSON interacts with binary serialization.

Table 1 provides critical characteristics of leading data serialization protocols JSON and Protobuf, highlighting their respective advantages and constraints.

Table 1
Comparison of JSON and Protobuf (based on [6, 11, 25])

Feature	JSON	Protobuf
Format	Text-based	Binary
Schema	Schema-less	Schema-driven
Readability	Human-readable	Requires tools for reading
Efficiency	Less efficient, larger file sizes	Higher efficiency, smaller file sizes
Performance	Potentially slower due to text parsing	High serialization/ deserialization speed
Resource Savings	May require more bandwidth and storage	Better for bandwidth and storage constrained environments
Type Safety	Dynamic, with types defined in data	Static, with strict types defined in **.proto** files
Version Compatibility	Native support can be more complex	Easy version management without breaking code
Language Support	Broad support by most languages	Support through generated code for various languages
API Integration	Widely used in Web APIs	Popular for internal and system communications
Message Size	Larger messages due to text nature	More compact messages due to binary format

The decision to use JSON, Protobuf (or any binary serialization format) depends on the particular use case. For example, web APIs typically use JSON because of its ubiquity and ease of use with JavaScript. However, for internal data storage or communication between services, especially in resource-constrained environments, it is better to use the binary format [25].

3. Challenge of device integration over Protobuf in IoT platforms

The challenge of integrating devices over Protobuf in IoT platforms is a universal issue, not confined to a specific platform. Therefore, we have selected ThingsBoard as the research tool for our analysis. IoT platform ThingsBoard [22], with its open-source nature and comprehensive features, provides a robust foundation for exploring these challenges and potential solutions in a detailed and practical manner.

The ThingsBoard IoT developers opted for schemaless JSON formats for primary serialization in external communication. These formats facilitate data exchange with IoT devices due to their user-friendly nature.

The IoT platform ThingsBoard uses Protobuf to exchange inter-component data. This provides significant advantages in compactness, speed, and bidirectional interoperability, which are critical for increased efficiency and scalability. This use extends beyond the IoT platform to other products of the ThingsBoard company, including ThingsBoard Edge and the MQTT broker TBMQ.

There is growing interest in using Protobuf directly at the device level. Many IoT devices transfer data exclusively via Protobuf, while others are looking for ways to switch to this format to improve efficiency and reduce network load.

Integrating IoT devices that exclusively communicate using Protobuf into IoT platforms exemplifies a pressing challenge, particularly for open-source platforms like ThingsBoard. Protobuf's static nature necessitates additional developer intervention for each new device type, undermining the platform's universality and scalability, especially in cloud deployments. To integrate a new Protobuf-compatible device, developers must manually define and compile the device's schema into the platform's codebase. This process is both time-consuming and prone to errors.

A notable example is the integration of Efento devices into ThingsBoard using CoAP and Protobuf for seamless connectivity. Describing the interaction between Efento NB-IoT sensors and the IoT platform ThingsBoard is [5]. The current version of the IoT platform ThingsBoard supports integration with the following Efento NB-IoT sensors (figure 2):

- temperature,
- humidity,
- air pressure,
- differential pressure,
- open/close,
- leakage,
- I/O.

Requires Efento devices with FW version: 06.02+ [21].

Simultaneously, with device firmware versions constantly evolving, a scenario emerges wherein the platform must continually adapt to support new or updated devices. This interdependence raises questions about the platform's sustainability in the IoT environment.

This scenario underscores IoT platforms' need to develop more dynamic and versatile data serialization solutions. A mechanism that allows for the real-time, dynamic compilation and loading of Protobuf schemas would revolutionize device integration, enabling seamless adaptation to new devices and data formats without extensive developer intervention or system disruption.

4. Dynamic schema compilation in Protobuf by ThingsBoard

The preference of Protobuf in IoT applications lies in its binary format's efficiency and the reduced load it imposes on network transmission. However, its static nature presents a formidable

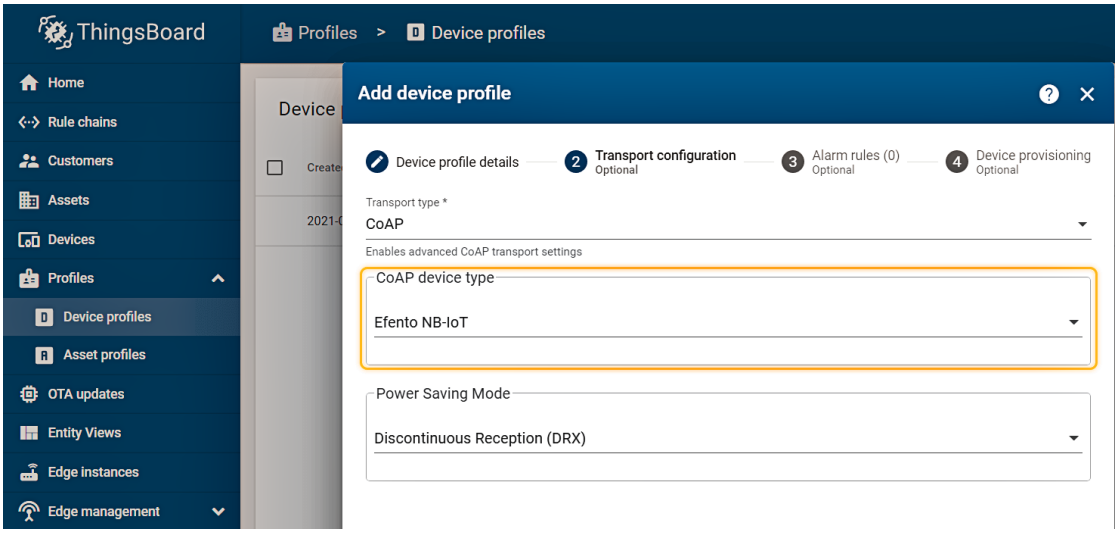

Figure 2: Configuring Efento sensors to send data to the IoT platform ThingsBoard [21].

challenge. Typically, **.proto** files must be pre-compiled using the Protobuf compiler (**protoc**), producing source code for the desired programming languages. Any alterations to the schema necessitate a tedious cycle of recompilation and redeployment, impeding the rapid adaptability required in the fluid IoT ecosystems.

Addressing this, ThingsBoard proposes a software tool enabling the real-time compilation of user-uploaded Protobuf schemas. This approach departs from traditional methods by allowing dynamic interpretation of Protobuf schema, thus permitting devices to communicate their data in Protobuf without necessitating system downtime or recompilation of the entire codebase. The solution is encapsulated within the IoT platform ThingsBoard through the concept of Device Profiles [21], which associate devices with their respective data transmission schemas. The current version of the IoT platform ThingsBoard supports customizable proto schemas for telemetry upload and attribute upload. It implements the ability to define a schema for downlink messages (RPC calls and attribute updates) (figure 3). ThingsBoard parses the Protobuf structures dynamically, which is why it does not yet support some Protobuf features like OneOf, extensions, and maps.

This dynamic process significantly reduces network traffic by transmitting data in Protobuf's compact form. It is only translated into a more verbose format like JSON when user interaction or specific system functions require it.

ThingsBoard's feature to dynamically interpret and compile Protobuf schemas for IoT devices without system downtime is quite innovative. This capability helps facilitate rapid updates and flexibility, which is crucial for IoT ecosystems where device configurations might change frequently.

As for other IoT platforms providing similar dynamic Protobuf schema compilation capabilities, it is less common. Most IoT platforms and systems that use Protobuf typically require the traditional compile-time schema definition and compilation approach. However, some platforms offer flexibility through other means or partially support dynamic schemas. Here are a few

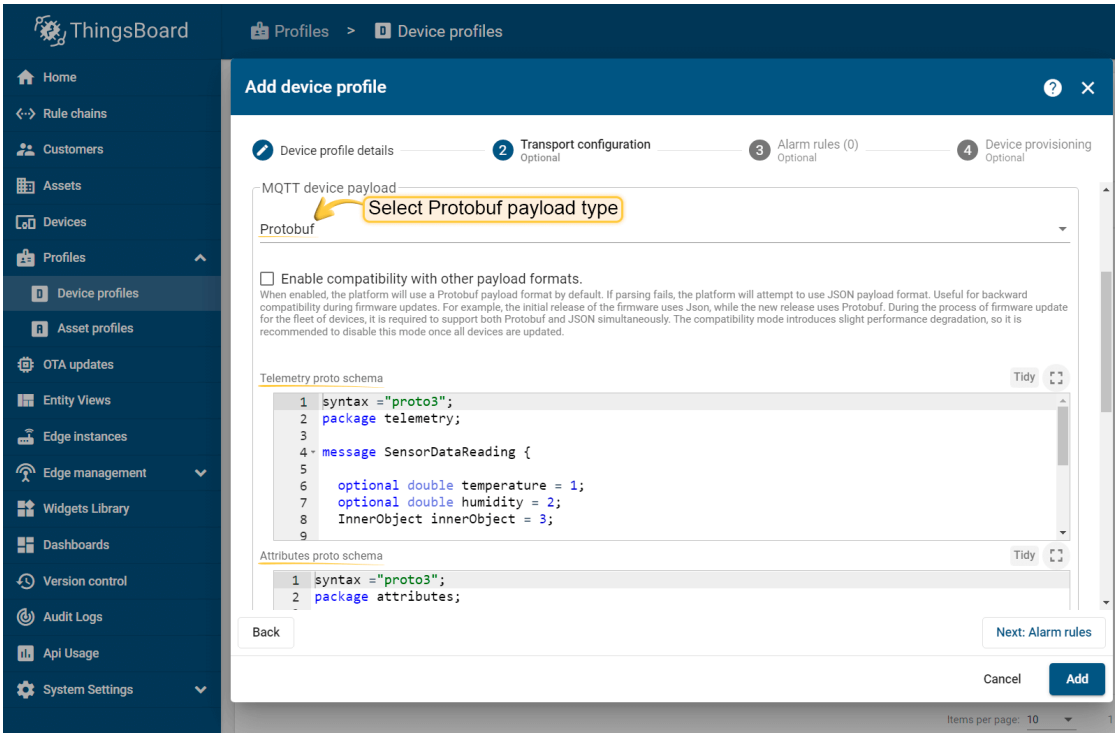

Figure 3: Uploading customizable proto schemas by IoT platform ThingsBoard [21].

notable mentions:

- *AWS IoT Core* [20] – While it does not support dynamic Protobuf schema compilation in the same way, AWS IoT Core is highly flexible and supports various communication and messaging protocols. It uses a rule engine that can transform and process input data from devices on the fly, providing some adaptability.
- *Google Cloud IoT* [7] – Similar to AWS, it primarily relies on static Protobuf schemas that need to be defined and compiled beforehand. However, Google Cloud's infrastructure, including its Pub/Sub and Dataflow services, allows for dynamic data handling and extensive data processing, which offers flexibility in handling data from IoT devices.
- *Azure IoT Hub* [12] – This platform also typically relies on predefined Protobuf schemas. Azure IoT Hub, however, integrates well with other Azure services like Stream Analytics, which can process data dynamically as it arrives from IoT devices. However, this is more about data processing than on-the-fly schema compilation.
- *Node-RED* [13] – An open-source programming tool for wiring together hardware devices, APIs, and online services in new and exciting ways. Node-RED can dynamically handle data structures using its flow-based programming model, which might provide some schema flexibility, although not specifically for Protobuf.
- *OpenHAB* [14] – An open-source IoT home automation platform. It allows dynamic handling of device data through its flexible configuration and scripting environment, but like others, it does not explicitly support dynamic Protobuf schemas.

That is a valuable point, especially considering the open-source nature of ThingsBoard. Most open-source platforms choose a more conservative approach to ensure stability and predictability of data processing. However, the IoT field is rapidly evolving, and in the future, more platforms may use similar capabilities to meet the need for more flexible data schema management. The approach provided by ThingsBoard ensures that as IoT devices evolve or new ones join the network, the system can swiftly accommodate them without extensive manual interventions or halts in operation. It represents a leap toward an adaptable IoT platform capable of keeping pace with the sector's rapid growth and diverse devices.

5. Discussion

In this study, we demonstrated how dynamic Protobuf schema compilation can improve the integration of Protobuf-compatible devices into the IoT platform ThingsBoard. Dynamic schema compilation effectively addresses integration challenges, reducing the need for constant developer intervention.

Despite the promising results, our solution has limitations. The current implementation does not support all Protobuf features. Further testing is required to evaluate performance in diverse IoT environments. Future research should optimize the dynamic compilation process and extend support for additional Protobuf features.

A subsequent paper will provide a more detailed description of the Protobuf schema compilation process and how the compiled version is used to process messages from devices sending data in Protobuf format. Additionally, future publications will discuss the results of testing the impact of this solution on the overall system performance and network load. The reason for deferring these aspects to a future paper is the extensive number of concepts that need to be introduced for proper explanation.

6. Conclusions

This article explored the evolving landscape of data serialization protocols in IoT, with a particular focus on ThingsBoard's dynamic schema compilation feature. We have demonstrated how Protobuf, despite its efficiency and reduced network load, faces challenges in static schema compilation, limiting IoT devices' adaptability. Our findings suggest that the innovative real-time, user-driven schema compilation solution can significantly enhance IoT platforms' flexibility, scalability, and overall performance.

For future research and development, it would be insightful to delve deeper into how such dynamic data serialization mechanisms can further benefit edge computing scenarios. Specifically, it investigates the impact on latency reduction, bandwidth optimization, and overall system responsiveness when deploying IoT devices in edge-centric networks. Additionally, exploring the integration of these serialization techniques with edge computing models could offer novel approaches to managing data flow and processing between edge devices and central systems, ultimately contributing to the scalability and robustness of IoT solutions.

References

[1] Alsayaydeh, J.A.J., Aziz, A., Rahman, A., Salim, S.N.S., Zainon, M., Baharudin, Z., Abbasi, M.I. and Khang, A.W.Y., 2021. Development of programmable home security using GSM system for early prevention. *Journal of Engineering and Applied Sciences*, 16(1), pp.1–10. Available from: https://www.arpnjournals.org/jeas/research_papers/rp_2021/jeas_0121_8470.pdf.

[2] Debnath, D. and Chettri, S.K., 2021. Internet of Things: Current Research, Challenges, Trends and Applications. In: X.Z. Gao, R. Kumar, S. Srivastava and B.P. Soni, eds. *Applications of Artificial Intelligence in Engineering*. Singapore: Springer, Algorithms for Intelligent Systems, pp.679–694. Available from: https://doi.org/978-981-33-4604-8_52.

[3] Di Felice, P. and Paolone, G., 2024. Papers Mentioning Things Board: A Systematic Mapping Study. *Journal of Computer Science*, 20(5), pp.574–584. Available from: https://doi.org/10.3844/jcssp.2024.574.584.

[4] Domínguez-Bolaño, T., Campos, O., Barral, V., Escudero, C.J. and García-Naya, J.A., 2022. An overview of IoT architectures, technologies, and existing open-source projects. *Internet of Things*, 20, p.100626. Available from: https://doi.org/10.1016/j.iot.2022.100626.

[5] Efento, 2024. Efento NB-IoT sensors and ThingsBoard. Available from: https://getefento.com/library/efento-nb-iot-sensors-and-things-board/.

[6] Friesel, D. and Spinczyk, O., 2021. Data Serialization Formats for the Internet of Things. *Electronic Communications of the EASST*, 80. Available from: https://doi.org/10.14279/tuj.eceasst.80.1134.

[7] Google, 2024. Connected device solutions. Available from: https://cloud.google.com/iot-core.

[8] Google, 2024. Google. Protocol Buffers. Available from: https://developers.google.com/protocol-buffers.

[9] JSON (javascript object notation), 2008. Available from: https://www.json.org/json-en.html.

[10] Khang, A.W.Y., Alsayaydeh, J.A.J., Gani, J.A.B.M., Pusppanathan, J.B., Teh, A.A., Ismail, A.F.M.F. and Geok, T.K., 2023. Reliable Multi-Path Communication for IoT Based Solar Automated Monitoring as Motivation Towards Multi-Farming Hydroponic. *International Journal of Interactive Mobile Technologies*, 17(21), pp.115–128. Available from: https://doi.org/10.3991/ijim.v17i21.43555.

[11] Luis, A., Casares, P., Cuadrado-Gallego, J.J. and Patricio, M.P., 2021. Pson: A serialization format for IoT sensor networks. *Sensors*, 21(13), 4559. Available from: https://doi.org/10.3390/s21134559.

[12] Microsoft, 2024. Azure IoT hub. Available from: https://azure.microsoft.com/en-us/products/iot-hub.

[13] OpenJSFoundation, 2024. Node-Red low-code programming for event-driven applications. Available from: https://nodered.org/.

[14] OpenHAB, 2024. OpenHAB empowering the smart home. Available from: https://www.openhab.org/.

[15] Popić, S., Pezer, D., Mrazovac, B. and Teslić, N., 2016. Performance evaluation of using Protocol Buffers in the Internet of Things communication. *2016 International Conference*

on Smart Systems and Technologies (SST). pp.261–265. Available from: https://doi.org/10.1109/SST.2016.7765670.

[16] Porkodi, R. and Bhuvaneswari, V., 2014. The Internet of Things (IoT) Applications and Communication Enabling Technology Standards: An Overview. *2014 International Conference on Intelligent Computing Applications*. IEEE, pp.324–329. Available from: https://doi.org/10.1109/ICICA.2014.73.

[17] Shapovalov, Y.B., Bilyk, Z.I., Usenko, S.A., Shapovalov, V.B., Postova, K.H., Zhadan, S.O. and Antonenko, P.D., 2023. Harnessing personal smart tools for enhanced STEM education: exploring IoT integration. *Educational Technology Quarterly*, 2023(2), Jun., p.210–232. Available from: https://doi.org/10.55056/etq.604.

[18] Shvaika, D.I., Shvaika, A.I. and Artemchuk, V.O., 2024. Data serialization protocols in IoT: problems and solutions using the ThingsBoard platform as an example. In: T.A. Vakaliuk and S.O. Semerikov, eds. *Proceedings of the 4th Edge Computing Workshop (doors 2024), Zhytomyr, Ukraine, April 5, 2024*. CEUR-WS.org, *CEUR Workshop Proceedings*, vol. 3666, pp.70–75. Available from: https://ceur-ws.org/Vol-3666/paper11.pdf.

[19] Statista, 2023. Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2030. Available from: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/.

[20] AWS, 2024. AWS IoT core. Available from: https://aws.amazon.com/iot-core/?nc1=h_ls.

[21] ThingsBoard, 2023. Device Profiles. Available from: https://thingsboard.io/docs/user-guide/device-profiles/.

[22] ThingsBoard, 2024. ThingsBoard IoT Platform. Available from: https://thingsboard.io.

[23] Uckelmann, D., Harrison, M. and Michahelles, F., eds, 2011. *Architecting the Internet of Things*. Springer Berlin Heidelberg. Available from: https://doi.org/10.1007/978-3-642-19157-2.

[24] Villamil, S., Hernandez, C. and Tarazona, G., 2020. An overview of Internet of Things. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 18(5), p.2320. Available from: https://doi.org/10.12928/telkomnika.v18i5.15911.

[25] Viotti, J.C. and Kinderkhedia, M., 2022. A Survey of JSON-compatible Binary Serialization Specifications. *CoRR*, abs/2201.02089. 2201.02089, Available from: https://doi.org/10.1109/ReCoSoC.2014.6861361.