INTELLIGENT SYSTEM OF TEMPERATURE MONITORING IN REFRIGERATED CONTAINERS

Halyna Vlakh-Vyhrynovska, PhD, as-prof.; Ulyana Dzelendziak, PhD, as-prof.; Oleh Ivanyuk, PhD, as-prof.; Mishel Vyhrynovskiy, student;
National University "Lviv Polytechnic," Ukraine,
e-mail: halyna.i.vlakh-vyhrynovska@lpnu.ua

https://doi.org/

Abstract. The work emphasizes the importance of maintaining a consistent temperature while transporting temperature-sensitive goods in the logistics cold chain. Refrigerated transportation plays a critical role in this chain, and the safe and efficient transport of perishable goods, such as food or pharmaceuticals, heavily relies on maintaining appropriate temperature levels. Real-time temperature monitoring and timely response to deviations are vital to prevent spoilage and financial losses. This paper presents an intelligent temperature monitoring system that employs a WiFi-enabled IoT platform NodeMCU-ESP32 to collect real-time temperature data, which is then transmitted to the AWS cloud platform. The system uses machine learning algorithms to predict temperature deviations and sends alerts to relevant stakeholders via the Telegram messaging service. The system architecture comprises AWS services, including API Gateway, IAM, Lambda, DynamoDB, and SNS, offering a scalable, secure, and cost-effective temperature monitoring solution.

Key words: Temperature Monitoring, Refrigerated Container, API Gateway, IAM, Lambda, DynamoDB, SNS, Telegram.

1. Introduction

The freight transportation market is transitioning to next-generation solutions, with the widespread use of Internet of Things (IoT) principles [1]. In particular, intelligent temperature monitoring systems for refrigerated containers are becoming increasingly important [2-3]. A refrigerated container is equipped with a refrigeration unit and is commonly used for transporting goods over long distances that require storage and transportation at specific temperatures, such as food products with a short shelf life and other goods that require particular conditions during a long journey while retaining their properties [4-5]. Developing a real-time system of this class is essential for avoiding cargo spoilage and significant losses for companies and enabling carriers to respond quickly to temperature deviations. The Amazon Web Services (AWS) platform provides cloud-based data storage and processing, and data is sent to the Telegram Bot through the Telegram social network program for relevant stakeholders [6-7]. Such a system can be part of a broader monitoring system for a fleet of refrigerators.

2. Disadvantages

Shipping may seem straightforward, but many products, pharmaceuticals, and industrial goods require specialized temperature-controlled shipping, including refrigerated containers, to ensure product quality. During container transportation, a specific temperature regime is needed, and deviations from the specified norms, such as interruptions in the power supply of refrigerators or container/equipment malfunctions, can lead to cargo spoilage and severe losses for companies [3-5]. It should be noted that there have been cases of unscrupulous drivers intentionally disconnecting refrigerators from the power supply to save fuel. One solution to ensure that the temperature inside the truck stays within the specified range is to use a temperature monitoring system. High-quality temperature monitoring requires finding a solution that meets the increasing requirements of maintaining the temperature regime and minimizing risks, reducing the probability of spoilage of the transported goods. Literature suggests that the cold chain has not received enough attention from the research community, especially when implementing cloud solutions for continuously collecting, storing, and transmitting data from IoT devices [8-9]. This article aims to implement cloud-based solutions in temperature monitoring systems, which are crucial for preserving and securing cargo transportation. It also provides suppliers, carriers, and cargo buyers with access to advanced analytics, opening up new opportunities.

3. Goal

This research aims to develop a temperature monitoring system for refrigerated containers with a scalable cloud architecture for collecting, processing, storing, and transmitting data.

4. The intelligent temperature monitoring system in refrigerated containers

The onboard terminal provides the data source for this intelligent system. The collected data are sent to a managed cloud database on the AWS platform to store temperature and humidity data from the container. The data are transmitted to the Telegram Bot for relevant stakeholders through the Telegram social media application. Such a system can be part of a broader monitoring system for a fleet of refrigerators

Refrigerated containers have become more common in recent years, and optimal control of the temperature inside these containers is crucial. The proposed intelligent temperature monitoring system aims to provide maximum temperature control throughout the journey to ensure the quality of the transported goods. The collected data will be sent online to the vehicle's driver and other interested parties, such as the supplier, carrier, or buyer.

The system uses an onboard terminal, a set of AWS cloud platform services, and the Telegram messaging application. AWS is a cloud computing platform that offers a wide range of services and tools for computing, storing, and managing data, including solutions for the Internet of Things (IoT) and machine learning. Essential AWS features make it a powerful tool for scientific research.

The proposed monitoring system consists of two parts: hardware, consisting of an on-board temperature measurement terminal implemented on the NodeMCU-ESP32 IoT platform and a DHT22 temperature and humidity measurement sensor, and a server, implemented on the AWS cloud platform for collecting, processing, storing, and displaying data received from the on-board terminal. Temperature readings are then transmitted to users via the Telegram Bot through the Telegram social network program. The server and client parts exchange data using the HTTP protocol. The structural diagram of the intelligent temperature monitoring system in refrigerated containers is shown in Fig. 1.

NodeMCU-ESP32 is an IoT platform with integrated Wi-Fi and Bluetooth controllers that can provide Wi-Fi 802.11n and BT4.2 functionality using SPI/SDIO or I^2C/UART interfaces. The DHT22 temperature and humidity measurement sensor has a temperature measurement range from -40 to +125°C and a humidity measurement range from 0 to 100%. The sensor data is read using a digital protocol and sent to the NodeMCU-ESP32 board. NodeMCU-ESP32 programming is implemented in the C++ programming language using the Arduino IDE [9,10].

The software utilized in this work includes API Gateway, IAM, Lambda, DynamoDB, and SNS of the AWS cloud platform. The API Gateway receives temperature data from the onboard terminal, and IAM is used to authorize access to the API Gateway. AWS Lambda processes data received from the API Gateway and DynamoDB to store temperature data. AWS SNS sends temperature data to the Telegram app.

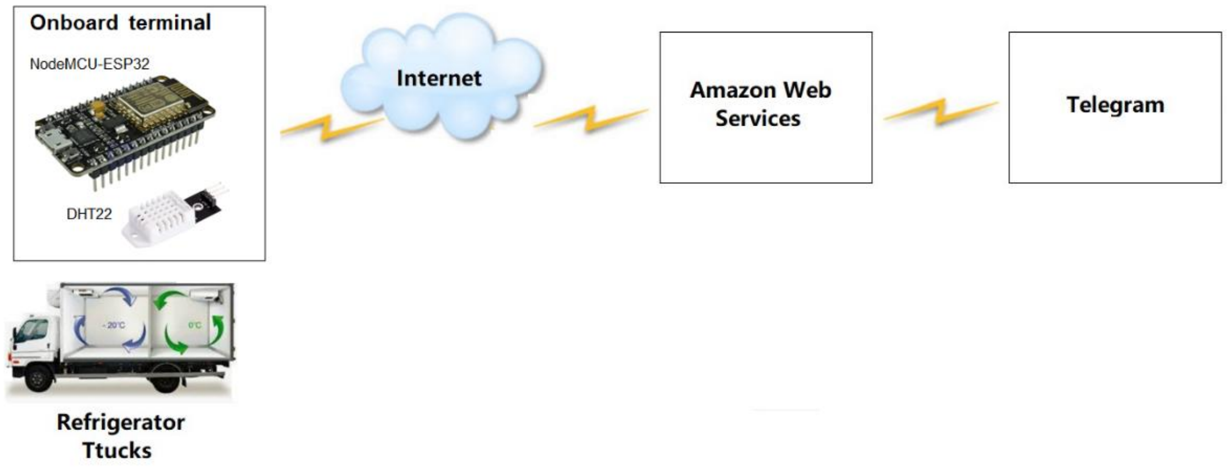


Fig. 1. Structural diagram of the intelligent temperature monitoring system in refrigerated containers

The deployment of infrastructure based on selected AWS services to collect, process, and store data received from the onboard terminal and with subsequent automatic notification in Telegram is presented in Fig. 2. AWS API Gateway is a managed service that allows developers to create, publish, and manage APIs that can securely access temperature data from various sources. API Gateway provides easy authentication and access control, including regulation and management of API keys. AWS Identity and Access Management (IAM) [11] is a service that enables us to manage access to AWS services and resources securely. With IAM, we can create users, groups, and roles and assign permissions to access AWS resources. Our temperature monitoring system can use IAM to create roles and policies that grant access to the API Gateway endpoint and other resources. AWS Lambda [12] is a serverless computing service that executes code in response to events. Lambda can process incoming temperature data and trigger alerts based on predefined thresholds in a temperature monitoring system. We can write a Lambda function that analyzes the input data, performs all necessary calculations, stores the data in DynamoDB, and sends an alert when the temperature exceeds a threshold. AWS DynamoDB [13] is a fully managed NoSQL database service that can store and retrieve real-time temperature data. DynamoDB provides low latency and high throughput for read and writes operations, making it an ideal choice for a temperature monitoring system. AWS SNS (Simple Notification Service) [14] is highly available and scalable messaging service that can send subscribers notifications via email, SMS, or push notifications. In the context of a temperature monitoring system, SNS can send messages to designated individuals or groups based on predefined thresholds. For example, SNS can send an email or social media alert when the temperature exceeds a specified threshold [15]. In summary, using AWS services such as API Gateway, IAM, Lambda, DynamoDB, and SNS, we can create a scalable and reliable refrigerated container temperature monitoring system that can process and store incoming temperature readings from sensors and notify relevant stakeholders in case of critical temperatures in real-time. Methodology. To create our temperature monitoring system, we will create three AWS Lambda functions: Post-to-DynamoDB: This function contains the logic to read the temperature data from the onboard terminal and store it in DynamoDB. DynamoDB-to-SNS: This function will receive temperature data from DynamoDB and send an alert via SNS if the temperature exceeds a given threshold. SNS-to-Telegram: This function will receive notifications from an SNS topic and send them to a Telegram channel using the Telegram API.

Implementation: The cloud receives data from the onboard IoT terminal through an API Gateway created using the AWS Management Console. A REST API "sensor" is made, and a POST method is added to retrieve the temperature and humidity data in the refrigerated container with its resource ID in Amazon Resource Names (ARNs), which is passed to the integration proxy of type Lambda proxy. The "sensor" resource sends data to the Lambda Function service in the post-to-dynamodb function. Before Lambda can access other services, an IAM role is created in the Identity and Access Management (IAM) service to manage user access to workloads and scaling securely. This role is bound to Amazon Lambda and has full access to AWS services: CloudWatch, DynamoDB, and an SNS topic created to send notifications if the temperature is out of range. We also make a bot_policy rule for the Telegram bot. Next, the Lambda function executes the written code, according to which the data received from the API Gateway will be recorded in the non-relational NoSQL database DynamoDB, where the temperature data sent from the onboard terminal are stored in the configured table. After the data is written to DynamoDB, another Lambda function, dynamodb-to-sns, is executed. It sends data to the SNS service when temperature readings go out of range to the topic pushes-from-dynamodb created in it. This topic then passes the lambda to the subscription, which calls another sns-to-telegram Lambda function with previously passed environment variables. These variables are the endpoints of the bot's telegram bot token and ID, called SensorUpdate, to which notifications are sent. We used JavaScript and Python programming languages to write Lambda functions post-to-dynamodb, dynamodb-to-sns, and sns-to-telegram. Fig. 3 shows the AWS Lambda interface with a list of functions created to implement the temperature monitoring system's operation in the Node.js runtime environment.

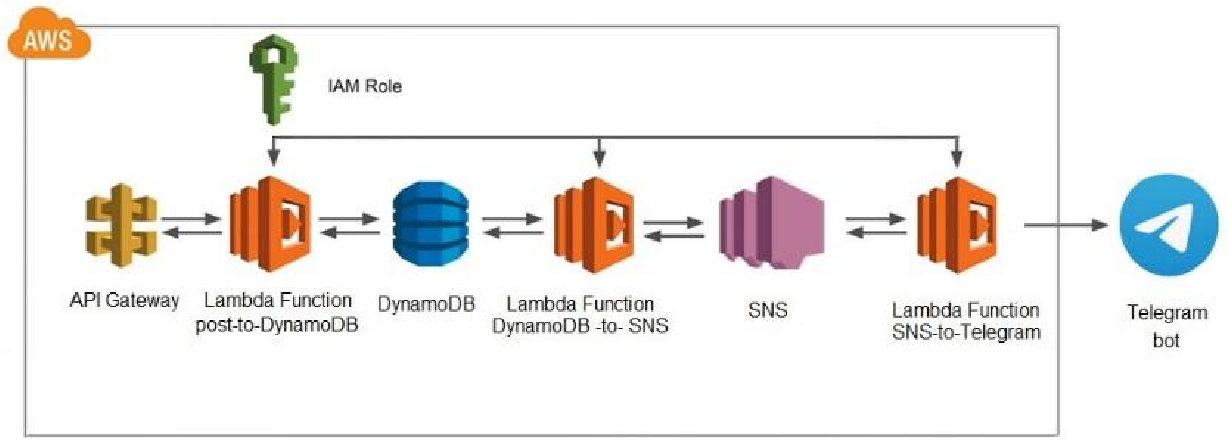
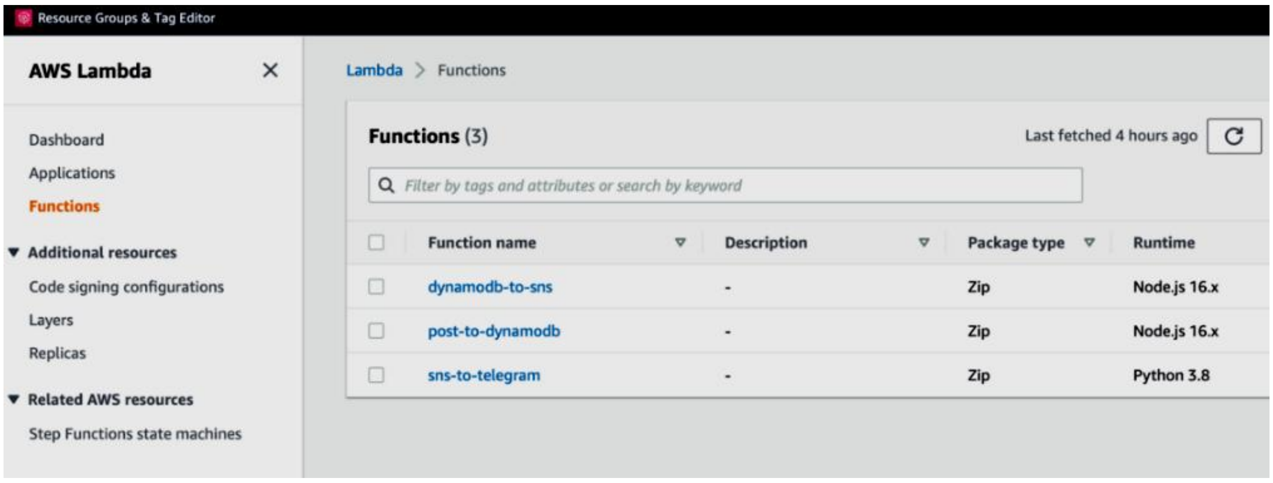
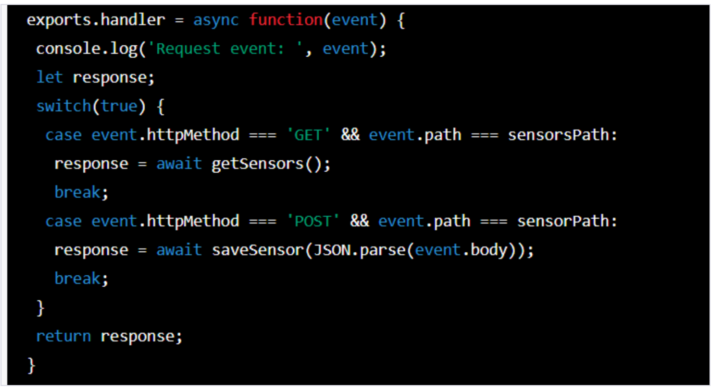

AWS
IAM Role
API Gateway
Lambda Function post-to-DynamoDB
DynamoDB
Lambda Function DynamoDB -to- SNS
SNS
Lambda Function SNS-to-Telegram
Telegram bot

Fig. 2. Infrastructure deployment in AWS



Function name	Description	Package type	Runtime
dynamodb-to-sns	-	Zip	Node.js 16.x
post-to-dynamodb	-	Zip	Node.js 16.x
sns-to-telegram	-	Zip	Python 3.8

Fig. 3. AWS Lambda interface with created functions post-to-dynamodb, dynamodb-to-sns, sns-to-telegram



```
exports.handler = async function(event) {
 console.log('Request event: ', event);
 let response;
 switch(true) {
  case event.httpMethod === 'GET' && event.path === sensorsPath:
   response = await getSensors();
   break;
  case event.httpMethod === 'POST' && event.path === sensorPath:
   response = await saveSensor(JSON.parse(event.body));
   break;
 }
 return response;
}
```

Fig. 4. Implementation of the Lambda exports.handler function

Consider a piece of code for writing a Post-to-DynamoDB function based on the primary function Lambda exports.handler Fig. 6. The function accepts an event parameter that contains information about the incoming HTTP request. The function uses a switch statement to determine the appropriate action based on the HTTP method and path in the request.

If the HTTP method is GET and the path is '/sensors,' it calls the getSensors() function to get all the temperature data from the DynamoDB table and return it as a response. If the HTTP method is POST and the path is '/sensor,' it calls the saveSensor() function to save the temperature data in the request body to a DynamoDB table and returns a success message as a response.

The function would return a null response if none of these conditions are met.

The results. In this study, we have developed a temperature monitoring system for reefer containers using AWS services and evaluated its performance under experimental conditions. The experiment results showed that the temperature monitoring system could accurately and reliably control the temperature of refrigerated containers. After setting up our lambda functions, we have a fully functional pipeline that processes the data and sends it to the Telegram channel. We can test our pipeline by sending data to the API Gateway endpoint, which will trigger the Post-to-DynamoDB function shown in Fig. 5. When the temperature exceeds the specified threshold, the DynamoDB-to-SNS function will receive data from DynamoDB and send it to the SNS topic, which will trigger the SNS-to-Telegram function. Finally, the SNS-to-Telegram function will send a message to the Telegram channel shown in Fig. 6.

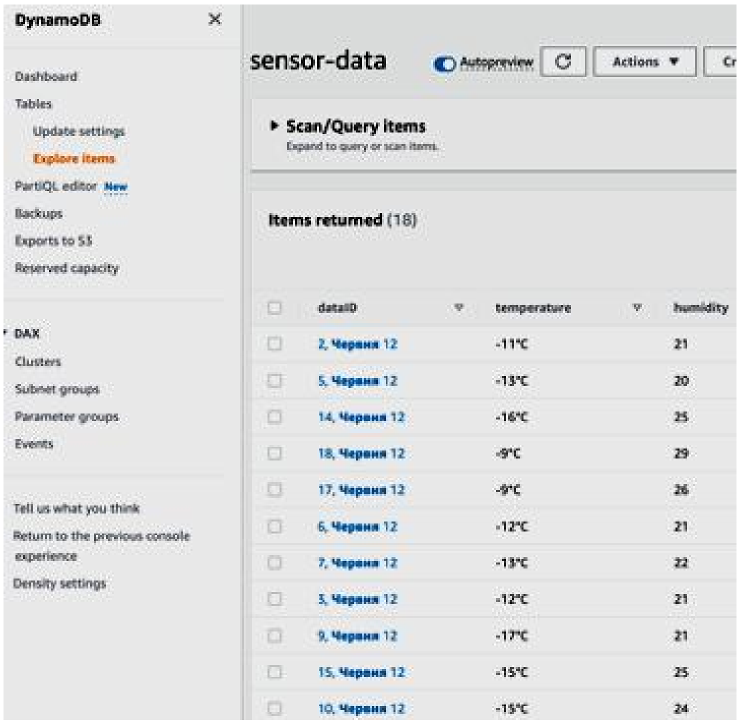
Advantages of the system:

1. Real-time monitoring: The system monitors in real-time the temperature inside the truck, allowing the logistics team to take immediate action in case of any temperature fluctuations.

2. Remote Access: Data is stored in the cloud, allowing the logistics team to access temperature data from anywhere in the world via a web dashboard or mobile app.

3. Automatic Alerts: The system is configured to send alerts in case of any temperature deviations, allowing the logistics team to take precautionary measures before damage occurs.

4. Cost-effectiveness: The system can be built on available hardware components, making it a cost-effective solution for small and medium-sized carrier companies.

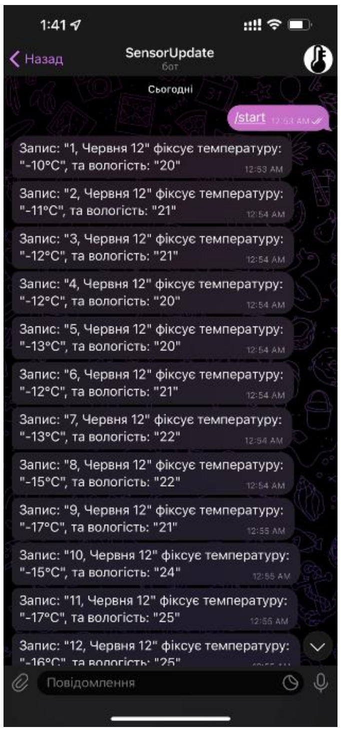


sensor-data

Items returned (18)

dataID	temperature	humidity
2, Червня 12	-11°C	21
5, Червня 12	-13°C	20
14, Червня 12	-16°C	25
18, Червня 12	-9°C	29
17, Червня 12	-9°C	26
6, Червня 12	-12°C	21
7, Червня 12	-13°C	22
3, Червня 12	-12°C	21
9, Червня 12	-17°C	21
15, Червня 12	-15°C	25
10, Червня 12	-15°C	24

Fig. 5. DynamoDB database table with temperature readings exceeding the specified threshold



/start 12:53 AM

Запис: "1, Червня 12" фіксує температуру: "-10°C", та вологість: "20" 12:53 AM

Запис: "2, Червня 12" фіксує температуру: "-11°C", та вологість: "21" 12:54 AM

Запис: "3, Червня 12" фіксує температуру: "-12°C", та вологість: "21" 12:54 AM

Запис: "4, Червня 12" фіксує температуру: "-12°C", та вологість: "20" 12:54 AM

Запис: "5, Червня 12" фіксує температуру: "-13°C", та вологість: "20" 12:54 AM

Запис: "6, Червня 12" фіксує температуру: "-12°C", та вологість: "21" 12:54 AM

Запис: "7, Червня 12" фіксує температуру: "-13°C", та вологість: "22" 12:54 AM

Запис: "8, Червня 12" фіксує температуру: "-15°C", та вологість: "22" 12:54 AM

Запис: "9, Червня 12" фіксує температуру: "-17°C", та вологість: "21" 12:55 AM

Запис: "10, Червня 12" фіксує температуру: "-15°C", та вологість: "24" 12:55 AM

Запис: "11, Червня 12" фіксує температуру: "-17°C", та вологість: "25" 12:55 AM

Запис: "12, Червня 12" фіксує температуру: "-16°C", та вологість: "25"

Fig. 6. Dialogue with the Telegram bot. Notification when the temperature exceeds the specified threshold

5. Conclusions

A temperature monitoring system for refrigerated trucks using the NodeMCU-ESP32 and AWS provides an efficient and cost-effective way to ensure that perishable goods are transported at the needed temperature. With real-time monitoring, remote access, and automatic alerts, logistics companies can ensure that their products are delivered safely and in optimal condition, reducing financial losses and health risks. Using the power of the AWS cloud, the system can scale to accommodate large fleets of trucks and provide valuable insight into the efficiency of logistics operations. Implementation is simple, and the AWS Management Console provides an intuitive interface for creating and managing AWS resources. The proposed system can be applied in various branches of industry, including healthcare, food storage, and manufacturing, to ensure product safety and quality.

6. Gratitude

The authors express their gratitude to the Staff members of the Departments of Computerized Automatic Systems and the Information-Measuring Technologies of Lviv Polytechnic National University.

7. Conflict of interest

The authors declare the absence of any financial or other potential conflict related to this work.

References

[1] Christoph Heinbach, Pascal Meier & Oliver Thomas. Designing a shared freight service intelligence platform for transport stakeholders using mobile telematics. Information Systems and e-Business Management, no. 20, 2022, pp. 847–888. https://doi.org/10.1007/s10257-022-00572-502

[2] Bob Castelein, Harry Geerlings, Ron Van Duin. The reefer container market and academic research: A review study. Journal of Cleaner Production, no. 256, 2020, 120654. https://doi.org/10.1016/j.jclepro

[3] Peiyao Tang; Octavian Adrian Postolache; Yangyang Hao; Meisu Zhong. 11th International Symposium on Advanced Topics in Electrical Engineering (ATEE), 2019. https://doi: 10.1109/ATEE.2019.8724950

[4] Jean-Paul Rodrigue. Fifth Edition. The Geography of Transport Systems, 2020, pp. 456. https://doi.org/10.4324/9780429346323

[5] SimonVolpert, Philipp Eichhammer, FlorianHeld, Thomas Huffe, Hans P.Reiser, JörgDomaschka. The view on systems monitoring and its requirements from future Cloud-to-Thing applications and infrastructures. Future Generation Computer Systems, no. 141, 2023, pp. 243-257. https://doi.org/10.1016/j.future.2022.11.024

[6] Sabrine Khriji, Yahia Benbelgacem, Rym Cheour, Dhouha El Houssaini, Olfa Kanoun. Design and implementation of a cloud-based event-driven architecture for real-time data processing in wireless sensor networks. The Journal of Supercomputing. No. 78, 2022, pp. 3374–3401. https://doi: 10.1007/s11227-021-03955-6

[7] C. Marinescu. Cloud Computing (Third edition). Theory and Practice, chapter 2 - The cloud ecosystem. 2023, pp. 13-40. https://doi.org/10.1016/B978-0-32-385277-7.00009-9

[8] [Hoa Tran-Dang, Nicolas Krommenacker, Patrick Charpentier &Dong-Seong Kim. The Internet of Things for Logistics: Perspectives, Application Review, and Challenges. IETE Technical Review. Volume 39(4), Issue 1 2020, pp. 93-121. https://doi.org/10.1080/02564602.2020.1827308

[9] Guie Li. Development of cold chain logistics transportation system based on 5G network and Internet of things system. Microprocessors and Microsystems. No. 80, 2021, 103565. https://doi.org/10.1016/j.micpro.2020.103565

[10] Oqaidi Mohammed, Ait Abdelouahid Rachida, Debauche Olivier, Marzak Abdelaziz. An open-source and low-cost Smart Auditorium. Procedia Computer Science, no. 191, 2021, pp. 518-523. [Online]. Available: https://doi.org/ 10.1016/ j.procs.2021.07.076

[11] Amazon Web Services: Architecture of API Gateway. Inc, 2023. https://docs.aws.amazon.com/apigateway/latest/developer-guide/welcome.html#api-gateway-overview-aws-backbone

[12] Amazon Web Services: AWS Identity and Access Management. Inc, 2023. https://docs.aws.amazon.com/IAM/latest/ UserGuide/introduction.html

[13] Amazon Web Services: AWS Lambda Documentation. Inc, 2023. https://docs.aws.amazon.com/lambda/?icmpid= docs_ homepage_compute

[14] Amazon Web Services: Programming with DynamoDB and the AWS SDKs. Inc, 2023. https://docs.aws.amazon.com/ amazondynamodb/latest/developerguide/Programming.html

[15] Amazon Web Services: Amazon SNS event sources and destinations. Inc, 2023. https://docs.aws.amazon.com/sns/ latest/dg/sns-event-sources-and-destinations.html