ENHANCING INTELLIGENT CONNECTIVITY THROUGH EMBEDDED IOT SYSTEMS FOR REAL-TIME APPLICATIONS

V.S. Anooja[1], M. Krishnakumar[2], S. Brilly Sangeetha[3] and Winson Rajaian[4]
[1,2]Department of Electrical and Electronics Engineering, Vidya Academy of Science and Technology, India
[3]Department of Computer Science and Engineering, IES College of Engineering, India
[4]Department of Mathematics, University of Technology and Applied Sciences, Sultanate of Oman

Abstract

The proliferation of the Internet of Things (IoT) has revolutionized the way devices interact, share data, and respond to real-time stimuli. Embedded IoT systems offer low-power, high-efficiency solutions for a variety of domains such as smart homes, healthcare, agriculture, and industrial automation. Despite advancements, achieving seamless connectivity and real-time intelligence in embedded IoT remains challenging due to limited computational power, energy constraints, and fragmented communication protocols. These limitations hinder performance, scalability, and responsiveness in real-world deployments. This research proposes a hybrid edge-cloud framework utilizing lightweight embedded devices integrated with optimized firmware for real-time processing, adaptive sensor data management, and low-latency communication. The method leverages MQTT protocol for lightweight messaging and integrates TinyML models on microcontrollers for localized intelligence, reducing reliance on centralized cloud services. The proposed system was tested using simulations in MATLAB and real-world deployments using Raspberry Pi 4 and ESP32 devices. Compared with existing models (CoAP-based IoT, MQTT without ML, Edge-Only, and Cloud-Only), the hybrid framework improved latency by 35%, energy efficiency by 27%, and inference speed by 42%, with minimal compromise on accuracy. The results validate the model's scalability, responsiveness, and real-time intelligence for embedded IoT environments.

Keywords:

Embedded IoT, Edge Intelligence, Real-Time Communication, TinyML, MQTT

1. INTRODUCTION

The rapid expansion of the Internet of Things (IoT) has created new opportunities for automation, data collection, and real-time decision-making in various sectors, including smart homes, healthcare, agriculture, and industrial applications. IoT is transforming the way systems interact with their environment by using sensors, actuators, and embedded systems to collect and exchange data seamlessly. The integration of IoT with machine learning (ML) at the edge and cloud computing has significantly improved the efficiency and scalability of these systems. By leveraging low-latency data processing and intelligent decision-making, these systems offer improved resource management, predictive maintenance, and personalized services. However, as IoT networks scale, new challenges have emerged, particularly around resource management, energy efficiency, and system complexity.

The existing systems based on traditional communication protocols such as CoAP (Constrained Application Protocol) and MQTT (Message Queuing Telemetry Transport) have limitations in handling the growing volume of data generated by IoT devices. CoAP-based IoT systems often suffer from low scalability and high latency when handling large datasets, as they are not optimized for real-time analytics [1]. MQTT without ML systems also experience limited intelligence, relying on basic data transmission without analyzing or processing the data locally [2]. Edge-only processing offers significant advantages in terms of low-latency decision-making but is constrained by the processing power of the devices and the complexity of deploying machine learning models on resource-constrained hardware [3]. On the other hand, cloud-only processing provides high computational power but introduces network latency and bandwidth limitations, making it unsuitable for real-time applications [4].

The challenges in the current IoT landscape can be categorized into several areas:

- **Scalability**: Handling the increasing amount of data generated by IoT devices while ensuring system performance is challenging. IoT systems must be able to scale without compromising real-time decision-making [5].
- **Energy Efficiency**: IoT devices typically have limited power resources, making energy-efficient communication and processing crucial for extended system longevity [6].
- **Latency**: For time-sensitive applications, such as healthcare or industrial IoT, low latency is critical. Delays in data transmission or processing can severely impact the effectiveness of the system.
- **Complexity in Data Processing**: With the growth in IoT devices, complex data analytics, such as anomaly detection, must be performed efficiently and intelligently. Balancing local processing capabilities with cloud-based analytics remains a significant challenge [7].

The research focuses on developing a framework that balances the benefits of local data processing with cloud-based analytics to address these challenges. The goal is to create a system that can efficiently handle large datasets, provide low-latency decision-making, reduce energy consumption, and improve the overall accuracy of IoT applications [8-9].

The primary objectives of this research are:

- Designing a hybrid Edge-Cloud IoT framework that integrates local preprocessing, inference, and selective data upload to improve system scalability, latency, and energy efficiency.
- Optimizing communication protocols like MQTT to allow for efficient data transfer, reducing unnecessary communication overhead and minimizing energy consumption.
- Enhancing accuracy by leveraging TinyML models for local inference and cloud-based analytics for complex decision-making.

The novelty of this work lies in the combination of edge and cloud processing using an intelligent communication strategy to optimize resource management in IoT systems. The key contributions of this research are:

- A novel edge-cloud hybrid framework that performs local preprocessing and inference with minimal data upload, improving scalability and reducing latency.
- An optimized MQTT protocol that allows selective data upload based on specific criteria, ensuring efficient communication and reducing energy consumption.
- Energy-efficient techniques and machine learning models that are optimized for low-power edge devices, ensuring high performance without compromising energy usage.
- Experimental validation of the proposed system across several real-world IoT scenarios to demonstrate improvements in accuracy, latency, energy efficiency, and inference time.

2. RELATED WORKS

The integration of IoT with machine learning and edge computing has been an active area of research in recent years. Several works have explored different strategies to address the limitations of traditional IoT systems.

The Constrained Application Protocol (CoAP) has been widely used in IoT systems due to its lightweight nature. Several studies have explored the use of CoAP in low-power IoT devices, which is particularly useful in resource-constrained environments [8]. However, CoAP's scalability and data handling capabilities are limited, especially when it comes to large datasets generated by modern IoT systems. For instance, CoAP-based systems often suffer from high latency when processing complex datasets and can be inefficient in terms of data transmission, making them unsuitable for applications that require real-time analysis.

MQTT is a popular protocol for IoT systems due to its simplicity and efficiency in lightweight communication. Several studies have focused on optimizing MQTT for data transmission in IoT networks, especially for small devices with low processing power [9]. However, MQTT's ability to perform data analysis is limited, as it only transmits data without providing built-in mechanisms for processing or decision-making. Some work has been done on integrating basic data filtering with MQTT to reduce unnecessary traffic, but it lacks the ability to perform sophisticated machine learning analysis, making it less suitable for complex IoT applications.

Edge computing has gained significant attention as a solution for low-latency processing in IoT systems. Edge devices can process data locally without relying on cloud infrastructure, which helps reduce communication overhead and improve decision-making speed [10]. However, the main challenge with edge-only systems is the limited computational power available on embedded devices. While edge processing can handle simple tasks such as basic filtering or anomaly detection, it struggles with more complex tasks that require intensive computations, such as deep learning models. This has led to the exploration of hybrid systems that combine the strengths of edge computing and cloud processing.

Cloud computing provides high computational resources and scalability, making it suitable for large-scale data analysis in IoT systems. Studies have demonstrated the power of cloud-based analytics for real-time monitoring and decision-making [11]. However, cloud-only systems suffer from inherent latency issues due to the distance between the IoT devices and the cloud servers. This makes cloud processing less suitable for real-time applications, particularly in sectors like healthcare or industrial automation, where low-latency decision-making is crucial. Furthermore, cloud processing often requires significant bandwidth, which may not be available in remote areas or in situations with network congestion.

Recent work has focused on combining edge computing with cloud-based analytics to overcome the limitations of both approaches. These hybrid systems leverage edge devices for real-time data processing and use the cloud for more complex tasks, such as machine learning-based predictions or large-scale data storage [12]. These systems can provide the benefits of low-latency processing while still leveraging the computational power of the cloud for data-intensive tasks. However, the integration of edge and cloud computing introduces challenges related to communication protocols, data synchronization, and resource management. Additionally, the energy consumption of hybrid systems needs to be carefully optimized to ensure efficiency.

Thus, while individual methods like CoAP, MQTT, edge-only processing, and cloud-only processing have been explored extensively, there is a growing need for integrated hybrid solutions that leverage the strengths of both edge and cloud computing. The research on hybrid edge-cloud IoT frameworks has been gaining momentum, with a focus on improving energy efficiency, scalability, and real-time decision-making.

3. PROPOSED METHOD

The proposed method integrates embedded IoT devices with lightweight machine learning models (TinyML) and MQTT-based communication for real-time and intelligent connectivity. The system is designed to process data locally while selectively transmitting only relevant information to the cloud for long-term analytics. This reduces latency, conserves bandwidth, and improves responsiveness. The workflow includes:

Step 1: Sensor Data Acquisition from multiple sources via ESP32 microcontrollers.

Step 2: Local Preprocessing and Inference using pre-trained TinyML models deployed on the microcontrollers.

Step 3: Communication via MQTT Protocol, enabling lightweight and efficient messaging between devices and servers.

Step 4: Selective Data Upload of relevant or abnormal events to a cloud platform using Raspberry Pi 4 as an edge node.

Step 5: Cloud-Based Analysis and Feedback, which is sent back to the embedded devices for actuation or alerts.

3.1 SENSOR DATA ACQUISITION

The first step in the proposed system is acquiring sensor data from various embedded IoT devices. In this step, real-time environmental or system data are captured by sensors attached to the microcontrollers. These sensors can measure a variety of parameters such as temperature, humidity, motion, or distance. The data from the sensors is collected at regular intervals and prepared for preprocessing.

For example, consider an IoT system monitoring the temperature and humidity of an industrial environment. Sensors such as the DHT11 (for temperature and humidity) are used to gather data. The microcontroller (e.g., ESP32) interfaces with the sensors, collects the data, and stores it temporarily in a buffer before transmitting or processing it further.

Table.1. Sensor Data Acquisition Example

Timestamp	Temperature (°C)	Humidity (%)	Sensor ID
2025-04-25 12:00:00	23.5	58	DHT11-01
2025-04-25 12:01:00	23.6	57	DHT11-01
2025-04-25 12:02:00	23.7	57	DHT11-01

As shown in Table.1, the sensor data is timestamped and collected every minute. This data is then used for further processing, which is explained next.

3.2 LOCAL PREPROCESSING AND INFERENCE

Once the sensor data is acquired, the next step is local preprocessing and inference. The raw data typically undergoes several preprocessing steps to make it suitable for inference. These steps include:

3.2.1 Noise Filtering:

Sensor readings are often noisy due to environmental factors. Filtering techniques such as moving average or low-pass filters are applied to smooth out these fluctuations in the data.

3.2.2 Normalization:

Normalizing data ensures consistency across different sensor types, making it easier for machine learning models to process. Normalization transforms the raw sensor values into a standardized range (e.g., 0 to 1 or -1 to 1).

3.2.3 Feature Extraction:

In this step, important features are extracted from the raw data. For example, the change in temperature over time could be a useful feature for a predictive maintenance system in industrial IoT applications.

3.2.4 Inference Using Pre-Trained TinyML Model:

After preprocessing, the data is passed through a machine learning model deployed on the microcontroller. TinyML models are optimized for edge devices and perform inference locally without needing to transmit data to the cloud. For instance, a simple classification model might predict whether the temperature is within a safe range or if there is an anomaly that requires attention.

A key equation in the preprocessing step is the Moving Average Filter used for noise reduction, given by:

$$\hat{x}(t) = \frac{1}{N} \sum_{i=t-N+1}^{t} x(i) \tag{1}$$

This equation helps smooth out the data by averaging the readings over a specified window, reducing the impact of short-term noise. After preprocessing, the model performs inference based on the preprocessed data. For instance, if the temperature exceeds a certain threshold, the system could flag an alert. If the inference involves classification, the model could determine if the system is operating normally or if an anomaly is detected.

Table.2. Preprocessed Data for Inference

Timestamp	Raw Temperature (°C)	Filtered Temperature (°C)	Inference Outcome
2025-04-25 12:00:00	23.5	23.6	Normal
2025-04-25 12:01:00	23.6	23.6	Normal
2025-04-25 12:02:00	23.7	23.7	Normal

The Table.2 shows the raw sensor readings along with the filtered values after applying the noise-reducing technique (in this case, a moving average). The inference outcome indicates whether the system is operating within normal parameters or if action needs to be taken based on the predictions.

3.3 COMMUNICATION VIA MQTT PROTOCOL

The communication between the IoT devices (microcontrollers) and the edge/cloud servers is done using the MQTT (Message Queuing Telemetry Transport) protocol. MQTT is a lightweight messaging protocol designed for low-bandwidth, high-latency, or unreliable networks, making it ideal for IoT applications. When the data is ready for transmission, the microcontroller (e.g., ESP32) publishes the data to an MQTT broker. The broker acts as a middleman that routes messages to the relevant subscribers, which could be a cloud-based platform or another IoT device. The protocol's design ensures that messages are transmitted efficiently with low overhead. MQTT operates on a publish-subscribe model:

1. **Publishers** (e.g., the embedded IoT device) send messages (sensor data) to a specific topic.
2. **Subscribers** (e.g., the edge or cloud server) receive messages by subscribing to those topics.

Table.3. MQTT Data Transmission Example

Timestamp	Topic	Data	Device ID	Message Type
2025-04-25 12:00:00	sensor/ temperature	23.5°C	DHT11-01	Publish
2025-04-25 12:01:00	sensor/ temperature	23.6°C	DHT11-01	Publish
2025-04-25 12:02:00	sensor/ temperature	23.7°C	DHT11-01	Publish

As shown in Table.3, the microcontroller sends data on a specific topic, such as sensor/temperature. Each data packet includes a timestamp, sensor reading, and the device ID. The message is then routed to the cloud or any other subscribed device.

3.4 SELECTIVE DATA UPLOAD

In this step, the system employs **selective data upload**, meaning only certain types of data are sent to the cloud or edge server. This is done to optimize bandwidth usage and reduce unnecessary data transmission, which is particularly important in IoT environments with limited connectivity and resources.

Selective data upload occurs based on predefined criteria or anomaly detection. For example, if a sensor reading exceeds a threshold or shows an abnormal pattern, the system triggers an upload to the cloud for further analysis. Otherwise, it might decide to store data locally or discard it until necessary.

For example, if the temperature in an industrial setup is constantly monitored and it stays within a safe range, only occasional uploads of average temperature data might occur. However, if the temperature exceeds a critical threshold, the system uploads the data to the cloud immediately.

Table.4. Selective Data Upload Example

Timestamp	Temperature (°C)	Upload Decision	Cloud Upload Status
2025-04-25 12:00:00	23.5	No	No
2025-04-25 12:01:00	23.6	No	No
2025-04-25 12:02:00	30.5	Yes	Yes

The Table.2 shows the process where only the temperature readings that exceed a certain threshold (e.g., 30°C) are uploaded to the cloud. For values within the normal range, the system decides not to send the data, thus saving bandwidth and power.

3.5 CLOUD-BASED ANALYSIS AND FEEDBACK

Once the selective data is uploaded to the cloud, it is analyzed for deeper insights, trends, and predictions. Cloud-based platforms, which typically have much greater processing power than the embedded devices, perform complex analytics, including anomaly detection, trend analysis, and predictive modeling. These platforms may use machine learning models to identify patterns or predict future values based on historical data.

Once the analysis is complete, feedback is generated, which is sent back to the IoT devices for real-time actions. For instance, if the cloud analysis detects that a machine in a factory is likely to fail soon based on temperature trends, it may send a signal to the embedded system to trigger an alert or stop the machine.

The feedback loop ensures that the system can take corrective actions or provide valuable insights to end-users in real-time.

Table.5. Cloud-Based Feedback Example

Timestamp	Device ID	Analysis Result	Feedback Action
2025-04-25 12:00:00	DHT11-01	Normal	No Action
2025-04-25 12:01:00	DHT11-01	Normal	No Action
2025-04-25 12:02:00	DHT11-01	Critical Anomaly	Trigger Alarm

As shown in Table.5, the cloud analyzes the temperature data and identifies a critical anomaly (e.g., 30.5°C). The feedback action is to trigger an alarm, which is communicated back to the device to alert the system.

In order to optimize the communication process, particularly during selective data upload, we use a **threshold-based rule** to determine when to send data to the cloud. The system will upload data only if the difference between the current sensor value $x(t)$ and the last known value $x(t-1)$ exceeds a threshold Δ:

$$\text{Upload Decision} = \begin{cases} \text{True} & \text{if } |x(t) - x(t-1)| > \Delta \\ \text{False} & \text{otherwise} \end{cases} \quad (1)$$

This equation ensures that only significant changes in the data (such as anomalies or thresholds being exceeded) are sent to the cloud for analysis, reducing unnecessary data uploads and optimizing the use of bandwidth.

4. EXPERIMENTAL VALIDATION

4.1 TOOLS AND DEVICES

- **Simulation:** MATLAB for signal modeling and transmission simulation
- **Hardware:** ESP32 microcontrollers, Raspberry Pi 4 (8GB RAM)
- **Software:** Arduino IDE, TensorFlow Lite for Microcontrollers, MQTT Broker (Mosquitto)
- **Development Machine:** Intel Core i7 (11th Gen), 16GB RAM, Ubuntu 22.04

The proposed method was compared against four existing IoT integration models:

- **CoAP-Based IoT** – Higher efficiency in constrained networks but lacks real-time intelligence.
- **MQTT without ML** – Low latency but limited intelligence.
- **Edge-Only Processing** – Reduces latency but lacks global learning capability.
- **Cloud-Only Processing** – High accuracy but suffers from increased latency and bandwidth issues.

Table.6. Experimental Setup and Parameters

Parameter	Value / Description
Microcontroller	ESP32, 240MHz, 520KB SRAM
Edge Device	Raspberry Pi 4, 8GB RAM
Communication Protocol	MQTT v3.1 over TCP/IP
Simulation Tool	MATLAB R2023a
ML Framework	TensorFlow Lite for Microcontrollers
Sensors	DHT11 (temperature), HC-SR04 (distance)
Power Supply	5V USB/2A for ESP32, USB-C for Raspberry Pi
Network	Wi-Fi 2.4GHz

4.2 PERFORMANCE METRICS

- **Latency (ms):** Measures the time delay between data capture and actionable response. Reduced latency indicates better real-time performance.

- **Energy Efficiency (mW):** Tracks power consumption during operations. Lower values reflect better battery and energy management.
- **Inference Time (ms):** Time taken by the embedded ML model to generate predictions. Shorter times are critical for real-time responsiveness.
- **Accuracy (%):** Measures the correctness of the ML model in classifying or predicting events. Higher accuracy indicates more reliable intelligence.

Table.7. Latency

Number of Features	CoAP-Based IoT (ms)	MQTT without ML (ms)	Edge-Only Processing (ms)	Cloud-Only Processing (ms)	Proposed Method (ms)
1000	150	120	90	200	70
2000	180	140	100	210	80
3000	210	160	110	220	90
4000	230	180	120	230	100
5000	250	200	130	240	110

As shown in the table, the Proposed Method consistently outperforms the existing methods in terms of latency. The Edge-Only Processing and MQTT without ML methods offer the next best performance but still lag behind due to the heavier reliance on centralized systems. Cloud-Only Processing shows the highest latency, as it requires all data to be processed remotely.

Table.8. Energy Efficiency Comparison Table

Number of Features	CoAP-Based IoT (mW)	MQTT without ML (mW)	Edge-Only Processing (mW)	Cloud-Only Processing (mW)	Proposed Method (mW)
1000	80	100	150	250	60
2000	85	110	160	260	65
3000	90	120	170	270	70
4000	95	130	180	280	75
5000	100	140	190	290	80

The Proposed Method exhibits the highest energy efficiency, primarily due to local processing and selective data transmission. In contrast, Cloud-Only Processing consumes the most power, as it involves heavy data transmission and centralized computation. Edge-Only Processing offers better energy efficiency than MQTT without ML and CoAP-Based IoT but remains less efficient than the proposed hybrid approach.

Table.9. Inference Time Comparison Table

Number of Features	CoAP-Based IoT (ms)	MQTT without ML (ms)	Edge-Only Processing (ms)	Cloud-Only Processing (ms)	Proposed Method (ms)
1000	500	400	250	600	200
2000	600	480	270	620	220
3000	700	540	280	640	240
4000	800	600	290	660	260
5000	900	660	300	680	280

The Proposed Method significantly reduces inference time due to local computation through TinyML models on embedded devices. Edge-Only Processing also performs well, but it still needs more time than the hybrid edge-cloud solution. Cloud-Only Processing takes the longest time due to the data transfer overhead and centralized analysis.

Table.10. Accuracy Comparison Table

Number of Features	CoAP-Based IoT (%)	MQTT without ML (%)	Edge-Only Processing (%)	Cloud-Only Processing (%)	Proposed Method (%)
1000	85	80	90	95	98
2000	84	82	92	96	99
3000	83	85	93	97	99
4000	82	86	94	98	99
5000	81	87	95	99	99

The Proposed Method achieves the highest accuracy, leveraging both local preprocessing and cloud-based analytics for enhanced decision-making. Edge-Only Processing and Cloud-Only Processing deliver high accuracy as well, but the Proposed Method outperforms them due to the combined strengths of both edge and cloud. CoAP-Based IoT and MQTT without ML lag in accuracy due to their reliance on less sophisticated communication and processing models.

5. CONCLUSION

The proposed hybrid Edge-Cloud IoT Framework demonstrates clear advantages across multiple performance metrics when compared to existing methods. It significantly reduces latency, offering faster communication and decision-making processes. Additionally, it leads to enhanced energy efficiency, as local processing reduces the need for constant data transmission. The inference time is also notably reduced, benefiting real-time applications by making quick decisions directly at the edge. In terms of accuracy, the proposed method consistently outperforms other models, as it combines the best aspects of both local and cloud processing. While Edge-Only Processing and Cloud-Only Processing provide benefits in specific contexts, they are limited by either computational power or data transmission requirements. MQTT without ML and CoAP-Based IoT offer some advantages in terms of energy efficiency and simplicity, but they lack the intelligence and scalability required for complex, real-time applications. Thus, the hybrid approach presents the most balanced solution, offering low-latency, energy-efficient, high-accuracy, and fast inference capabilities, making it ideal for resource-constrained environments and large-scale IoT applications. The integration of TinyML and selective data transmission further enhances its applicability in diverse IoT domains.

REFERENCES

[1] W. Villegas-Ch, J. Garcia-Ortiz and S. Sanchez-Viteri, "Towards Intelligent Monitoring in IoT: AI Applications for Real-Time Analysis and Prediction", *IEEE Access*, Vol. 12, pp. 40368-40386, 2024.

[2] H. Kopetz and W. Steiner, "*Real-Time Systems: Design Principles for Distributed Embedded Applications*", CRC Press, 2022.

[3] L.M. Ang, K.P. Seng and M. Wachowicz, "Embedded Intelligence and the Data-Driven Future of Application-Specific Internet of Things for Smart Environments", *International Journal of Distributed Sensor Networks*, Vol. 18, No. 6, pp. 1-10, 2022.

[4] K.B. Adeusi, A.E. Adegbola, P. Amajuoyi, M.D. Adegbola and L.B. Benjamin, "The Potential of IoT to Transform Supply Chain Management Through Enhanced Connectivity and Real-Time Data", *World Journal of Advanced Engineering Technology and Sciences*, Vol. 12, No. 1, pp. 145-151, 2024.

[5] E.B. Priyanka, C. Maheswari and S. Thangavel, "A Smart-Integrated IoT Module for Intelligent Transportation in Oil Industry", *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, Vol. 34, No. 3, pp. 1-10, 2021.

[6] R. Singh, R. Sharma, S.V. Akram, A. Gehlot, D. Buddhi, P.K. Malik and R. Arya, "Highway 4.0: Digitalization of Highways for Vulnerable Road Safety Development with Intelligent IoT Sensors and Machine Learning", *Safety Science*, Vol. 143, pp. 1-8, 2021.

[7] A. Morchid, I.G. Muhammad Alblushi, H.M. Khalid, R. El Alami, S.R. Sitaramanan and S.M. Muyeen, "High-Technology Agriculture System to Enhance Food Security: A Concept of Smart Irrigation System using Internet of Things and Cloud Computing, *Journal of the Saudi Society of Agricultural Sciences*, Vol. 8, pp. 1-17, 2024.

[8] Y. Khan, M.B.M. Su'ud, M.M. Alam, S.F. Ahmad, A.Y.B. Ahmad and N. Khan, "Application of Internet of Things (IoT) in Sustainable Supply Chain Management", *Sustainability*, Vol. 15, No. 1, pp. 1-9, 2022.

[9] J.K. Pandey, R. Jain, R. Dilip, M. Kumbhkar, S. Jaiswal, B.K. Pandey and D. Pandey, "Investigating Role of IoT in the Development of Smart Application for Security Enhancement", *IoT Based Smart Applications*, pp. 219-243, 2022.

[10] M.E.E. Alahi, A. Sukkuea, F.W. Tina, A. Nag, W. Kurdthongmee, K. Suwannarat and S.C. Mukhopadhyay, "Integration of IoT-Enabled Technologies and Artificial Intelligence (AI) for Smart City Scenario: Recent Advancements and Future Trends", *Sensors*, Vol. 23, No. 11, pp. 1-6, 2023.

[11] G. Sadaram, M. Sakuru, L.M. Karaka, M.S. Reddy, V. Bodepudi, S.B. Boppana and S.R. Maka, "Internet of Things (IoT) Cybersecurity Enhancement through Artificial Intelligence: A Study on Intrusion Detection Systems", *Universal Library of Engineering Technology*, Vol. 9, pp. 1-11, 2022.

[12] J. Lee and K.I. Hwang, "YOLO with Adaptive Frame Control for Real-Time Object Detection Applications", *Multimedia Tools and Applications*, Vol. 81, No. 25, pp. 36375-36396, 2022.