COMPUTERIZED AUTOMATIC SYSTEMS

PROTOCOLS COMPARISON FOR REAL-TIME DATA STREAMING FROM IOT DEVICES TO A CLOUD-BASED SOLUTION

Anton Shykhmat, Ph. D. Student,
Lviv Polytechnic National University, Ukraine; e-mail: anton.o.shykhmat@lpnu.ua
Zenoviy Veres, Ph. D., Assistant,
Lviv Polytechnic National University, Ukraine, e-mail: zenovii.y.veres@lpnu.ua

https://doi.org/10.23939/istcmtm2023.04.044

Abstract. The proper detection and prevention of malfunctions are crucial in mitigating maintenance costs and equipment replacements for agricultural vehicles, ultimately reducing the expenses associated with crop cultivation. Predictive analytics for agriculture vehicles leverage machine learning and sensor data to anticipate equipment faults, optimize maintenance schedules, and enhance operational efficiency in the farming industry. It heavily relies on real-time data transmission to continuously monitor equipment performance, enabling timely identification of potential issues and preemptive maintenance actions to prevent costly breakdowns and downtime. This paper employs a qualitative analysis approach utilizing the Architecture Tradeoff Analysis Method to evaluate and select an optimal data protocol from a set of candidates, including SOAP, HTTP, REST, CoAP, WebSocket, XMPP, MQTT, and AMQP. The analysis considers sensitivity points, tradeoff factors, risks, and quality attribute scenarios relevant to the usage scenarios. The findings indicate that MQTT is the preferred protocol for real-time data streaming in resource-constrained environments, contingent upon a reliable connection.

Key words: IoT; Data transmission protocols; HTTP; MQTT.

1. Introduction

Ukraine takes an important place in the food provision on world markets, and its agricultural sector plays a significant role in the formation of foreign exchange earnings. In particular, in January – September 2020, 22.2 billion dollars were secured due to the agriculture sector, which accounted for 45.1 % of the country's total merchandise exports. According to the State Statistics Service, more than 309,000 workers will be employed in agriculture in 2021. The large-scale aggression of the Russian Federation has significantly worsened working conditions and reduced the possibility of exporting agricultural products in many ways, which negatively affects the financial indicators of farm holdings. In addition, it is necessary to address the problem of the loss of agricultural machinery as a result of hostilities or its abduction by the Russian military. Thus, according to the press service of the Ministry of Agrarian Policy and Food of Ukraine, “As of June 8, 2022, farmers from seven war-affected regions of Ukraine lost 2,281 units of agricultural machinery. These are farmers from Kyiv Oblast, Sumy Oblast, Chernihiv Oblast, Luhansk Oblast, Donetsk Oblast, Kherson Oblast, and Mykolaiv Oblast”. Thus, agriculture in Ukraine, in addition to the existing major problems with an outdated technical and technological base and an inc- rease in the cost of growing crops [1], has also problems with a decrease in the amount of working capital, the possibility of attracting credit and the destruction of equipment. It should be noted that the wear and tear of technological equipment reaches 70–80 %, and this increases the risk of malfunctioning of the equipment as a whole or its parts. Many studies have found that repair and maintenance costs depend on differences in machinery performance, availability of spare parts, operator skills, crop and weather conditions, maintenance policies, and other factors. However, in all studies, there is a pattern that as the age of the equipment increases, the costs of repair and maintenance increase [2].

Frequent malfunctions require more capital investment and time to solve them. As a consequence, the cost of growing crops is increasing which affects the final prices for the consumer. According to a study by the Global Network Against Food Crises, rising prices are leading to hunger in many regions of the world. At the moment, almost 30 million people suffer from hunger precisely because of the increased cost of products [3].

Timely detection and prevention of malfunctions are the key approaches to reduce maintenance costs, and update and replace equipment, which will reduce the cost of growing crops.

2. Drawbacks

The most crucial aspect of agriculture vehicles is timely repair and maintenance. Agriculture vehicles are manufactured in such a way that they can withstand thousands of hours of service. However, these indicators can be achieved only under the condition of systematic maintenance of the equipment. Farmers can cut machinery repair costs by 25 percent by improving routine maintenance procedures, according to a study of production machinery maintenance, costs, and simulation. One of the easiest ways to prevent breakdowns in rolling

stock is to forecast repair and maintenance costs based on historical data. This approach is based on the average accumulated costs for repair and maintenance of equipment, the age of the equipment, and the number of hours per year spent on repair and maintenance [4]. Another solution is the prevention of malfunctions with the help of self-diagnosis of the engine based on the analysis of the following data: engine revolutions, engine load, airflow, coolant temperature, pressure in the intake manifold, intake camshaft position, exhaust camshaft position, and fuel supply angle. A neural network-based solution, trained using the emulation of engine operation and potential malfunctions is used for such purposes [5]. The main disadvantages of such an approach are the small number of predicted failures and dependency on artificially generated data for model training instead of real historical data due to the complexity of its collection. The Internet of Things (IoT) technology provides an ability to solve this problem by building a digital platform that allows farmers to:

1. Store telemetry from agricultural machinery concerning geographical location and natural conditions.
2. Analyze the received data, detect malfunctions that have already occurred, as well as predict potential malfunctions, and receive notifications about them.
3. Plan the costs of maintenance, renewal, and replacement of agricultural machinery, which will reduce the cost of growing crops.

The development of this platform heavily relies on the presence of real data ingested by agriculture vehicles that unlock the abilities:

1) to track the geographic position of agricultural machinery by the device;
2) to transmit processed data by the device in real-time to the cloud application for further analysis;
3) to perform the analysis of ingested data to detect malfunctions that have already occurred;
4) to create a model for predicting potential malfunctions and notifying farmers about them.

Properly selected data transmission protocol for telemetry data delivery from agricultural vehicles is critical as it addresses a set of Quality Attributes such as efficient data communication, optimal resource utilization, and compatibility among a diverse array of interconnected devices, ultimately impacting the system's reliability, security, and overall performance.

3. Goal

The goal of this paper is to offer a comprehensive survey of message exchange protocols utilized in the Internet Of Things domain and select an optimal protocol for real-time data transmission scenarios through a qualitative analysis that leverages the Architecture Tradeoff Analysis Method.

4. High-level vision of the digital platform architecture

The typical architectural view of the digital platform contains six main components, where the first one is located in the vehicle and others are deployed into the cloud platform:

1. An IoT device, connected to the CAN bus of the agriculture vehicle intercepts all signals, processes them, tracks the geographic position of the tractor at the current moment ties the signals to this position, and sends geospatial and telemetry data to the cloud application.
2. The data-receiving component in the cloud application receives geospatial and telemetry data and saves it.
3. Data analysis component in the cloud application that analyzes the stored data and identifies agriculture vehicle malfunctions that have already occurred.
4. Fault prediction component in the cloud application, that generates a fault utilizing machine learning models for different parts of equipment.
5. A database in a cloud application that stores the data received from the device and the results of their analysis.
6. A separate database in a cloud application that stores archived data older than 1 year, usually in raw data formats like parquet files or Data Lakes.

Utilizing this approach to system separation allows to address availability, reliability, data integrity, extensibility, and scalability non-functional requirements. The intended structure of the system is shown in Fig. 1.

In case real-time data streaming from machinery/equipment to an offsite location (e. g., cloud), is required to monitor important parameters 24×7 or predictively detects vehicle malfunctioning issues, the data transmission process becomes crucial: the choice of an appropriate data transmission protocol significantly influences the efficiency, reliability, and overall performance of the system.

The next section compares the data transmission protocols and provides the recommended protocol that satisfies the non-functional requirements and the high-level vision of the digital platform architecture.

5. Data-transfer protocols analysis and selection

SOAP (Simple Object Access Protocol) is a protocol to exchange structured messages in distributed computing systems. The development of the protocol specification began in 1998, and only in 2003 it received the status of recommended to use by the World Wide Web Consortium (W3C) [6]. SOAP uses XML to de- scribe messages and extends some application-layer pro- tocols such as HTTP, FTM, and SMTP. Due to this, the

protocol allows communication between several systems developed using different programming languages [7]. SOAP supports 2 styles of interaction between the client and the service:

1. RPC (Remote Procedure Call) – the body of the message contains an element with the name of the method or operation that the client wants to perform on the service. In turn, this element contains values for each parameter of the called method/operation. Only the request-response approach is supported.

2. Document – the message body contains one or more child elements called parts. Supports both synchronous ("request-response") approach and asynchronous (deferred response to a request)

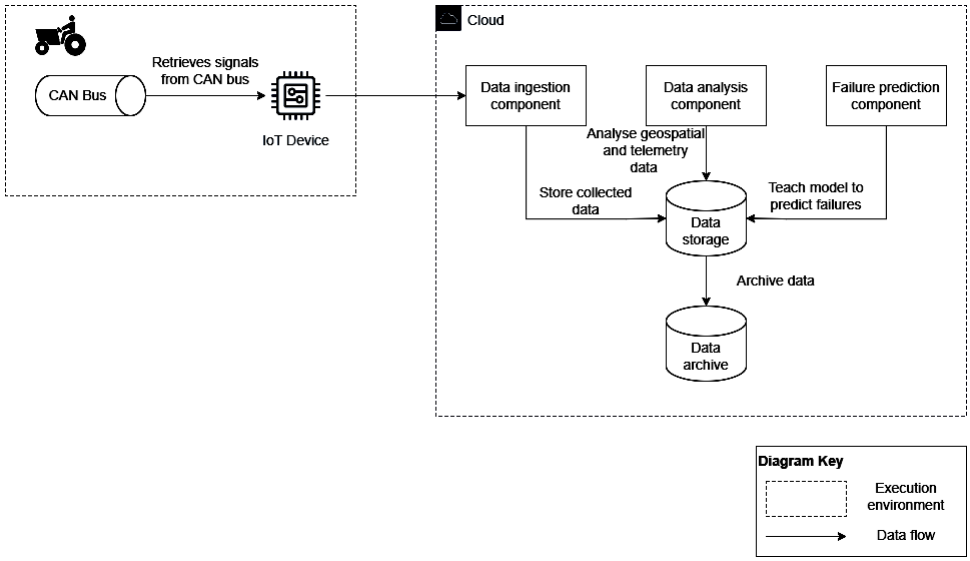

Fig. 1. High-level architecture view of the digital platform

SOAP is verbose (which increases the amount of traffic between the client and the service) and complex from a programming point of view, with slow processing speed, but it is versatile and has built-in error handling that makes it easier to resolve them and uses an XML specification that describes in detail a set of rules for encoding messages in a way that is readable by both humans and computer applications [8].

REST (English – Representational state transfer) is an architectural style for building web services based on the HTTP protocol [8]. Services supporting this style are called RESTful services. Such services do not store the state of the client, which makes their use fast, reli- able, and scalable. RESTful services, in response to re- quests made to a resource URI, most often respond in (but are not limited to) HTML, JSON, or XML formats. RESTful services most often use 4 HTTP methods [9]:

1. GET – for reading the resource.
2. POST – to create a new resource.
3. PUT – to change an existing resource.
4. DELETE – to delete an existing resource.

It should be noted that the GET, PUT, and DELETE methods must be idempotent, that is, return the same result regardless of how many times the request is executed. For example, if the resource exists, the first execution of the DELETE request will delete the resource and as a result, it will cease to exist, the second and subsequent requests will not delete anything, because the resource does not exist, but the result will be the same – the lack of existence of the resource. The POST method is not identical. The following 2 HTTP methods are sometimes used:

1. HEAD – get the headers returned by the server in response to a resource request.
2. OPTIONS – get a list of methods supported by the server for a specific resource.

The advantages of using REST are its simplicity, security, and scalability. However, the disadvantages include the impossibility of sending one message to many recipients with one request, the increased need for node resources, the increased time for sending and proc- essing requests (compared to application protocols built on top of the UDP transport layer protocol), and the lack of automatic resending of a message in the event of a processing error of this message on the server, which reduces reliability.

CoAP (Constrained Application Protocol) is a data transmission protocol designed for use in devices with limited resources and networks with limited bandwidth, as they usually have packet loss [10]. The CoAP proto- col uses a request-response model for communication in networks that suffer from packet loss. It is built on the UDP protocol but is not limited to it [10] and provides the exchange of messages between two or more

nodes [7]. The CoAP protocol is based on the REST architecture, but a node can be both a client and a server at the same time. In asynchronous messaging, the client requests an action on a resource using server-side method code. The server responds with a code that may contain an image of the resource. Request/response semantics include GET, POST, PUT, and DELETE methods [8]. In addition, the protocol adds a new method - Ob- serve. This method is used to implement the subscription concept built into the protocol. This is a regular GET request with the observe option, which tells the server that the client wants to receive information about each resource update [10]. Responses have HTTP-like status codes in the format 2. xx (successful requests), 4. xx (client errors), 5. xx (server errors). The CoAP protocol supports 4 types of messages [11]:

1. CON (Confirmable) – a message for reliable transmission. The sender is waiting for confirmation of receipt of the message by the recipient. If no confirmation is received, the message is resent after a certain period.
2. NON (Non-Confirmable) – a message for unreliable transmission when the sender does not want to receive confirmation of receiving the message.
3. ACK (Acknowledgement) – confirmation of receipt of the message by the recipient.
4. RST (Reset) – confirms receipt of the message, but indicates that the message does not contain information necessary for its successful processing.

Since the CoAP protocol is similar to the HTTP protocol (the main difference in the transport protocols on which they are built is UDP versus TCP), it can work with HTTP using proxy components [10]:

1. CoAP-HTTP proxy allows access to HTTP server resources for CoAP clients. Since CoAP methods are equivalent to HTTP methods, the construction of HTTP, TCP, and TLS (optional) can be done easily.
2. HTTP-CoAP proxy allows access to CoAP server resources for HTTP clients. To send an HTTP request to the proxy component, the client must specify the absolute path to the resource including the scheme (coap/coaps) when calling the method. Once the component proxy has received the message, it requests the specified CoAP resource. All HTTP methods from the RFC 2616 specification, except for OPTIONS, TRACE, and CONNECT, can be converted to CoAP.

Thus, the CoAP protocol has advantages in its reduced need for client resources, security using DTLS, support for synchronous and asynchronous communication, and fast message transfer. Disadvantages include lower reliability of data transmission (compared to application protocols built on the TCP transport layer protocol), the inability to send one message to many recipients (for this, you need to send a separate message to each recipient), and problems with communication with NAT devices.

WebSocket is a protocol that provides reliable full-duplex channels using a single TCP connection. WebSocket supports the HTTP protocol, which allows the use of HTTP proxies, which are usually present when organizing Internet access through firewalls [12]. Web servers and clients, such as browsers, can transfer data in real time between them. Once a connection is established, servers can send content to clients without requiring the clients to request it first. Messages are exchanged over the established connection, which remains open, in a standardized format. WebSocket support is present in almost all modern browsers, but the server must also have WebSocket support for messaging [7].

The WebSocket protocol supports 5 types of messages:

1. Upgrade – a message sent by the client to the server to open a communication channel.
2. Accept – a message sent by the server to the client upon successful opening of the communication channel.
3. Message – messages exchanged between the client and the server for data transfer.
4. Close – a message sent by the server to the client to close the communication channel.
5. Close Response – a message sent by the client to the server in response to closing the communication channel.

The advantages of the WebSocket protocol are the fast message transmission since it is not necessary to open a connection between the client and the server eve- ry time, the reduced volume of traffic, and the security using TLS. The main disadvantages of WebSocket in- clude the requirement for a stable TCP connection and the lack of automatic restoration of a lost connection, which reduces the reliability of this protocol, increased requirements for node resources, problems with scaling and load distribution, and is critical for use in agricul- tural machinery.

XMPP (eXtensible Messaging and Presence Protocol) is designed for instant messaging, contract support, and network presence communication. Allows the exchange of structured and extensible data in XML format between two network nodes in near real-time mode [7]. In the context of IoT, this protocol allows the exchange of messages between sensors, devices, and applications. XMPP assigns a unique address, also called a JabberID (JID), to each network element. The JID format is similar to the email format and contains a hostname, a domain name, and a resource [7]. In this case, the name of the node and the resource are not manda- tory. The protocol supports a client-server architecture, which imposes certain restrictions – clients cannot com-

municate with each other directly, but only through the XMPP server. When a client starts working with an XMPP server, it opens a long-lived TCP connection and starts an XML stream to the server. When the server accepts the client, it also starts an XML stream to the client. As a result, there are 2 XML streams – one from the client to the server, and the other from the server to the client. XMPP supports 3 types of messages:

1. PRESENCE – is used to exchange information about the presence between network nodes.
2. MESSAGE – used for communication between two network nodes.
3. IQ – is used to exchange information between the XMPP server and the client.

The main advantages of the XMPP protocol are its simple addressing, scalability, and security since the protocol uses TLS and SASL for connections, but it has its disadvantages, in the form of slower transmission and data processing due to increased traffic volume, due to their is a textual data transfer in XML format and the TCP transport protocol is used, and the reliability is re- duced because there is no confirmation of message proc- essing.

MQTT (Message Queue Telemetry Transport) is a simple and lightweight protocol of the "publisher-subscriber" model, designed specifically for exchanging messages between devices with limited resources and networks with limited bandwidth. The protocol provides message distribution using a one-to-many model and is independent of their content. MQTT works reliably and flawlessly in networks with high latency and limited bandwidth without requiring significant resources and device power [7]. The paradigm of the protocol is for many clients to connect to a server called a broker. Clients can have the role of a publisher (sending mes- sages to the broker) and subscriber (receiving messages from the broker). Subscribers and publishers do not know about each other, which makes it possible to achieve weak connectivity between applications interact- ing via the MQTT protocol. The advantage of this loose coupling is that the MQTT broker guarantees buffering and caching in case of network problems, which also means that publishers and subscribers do not have to be on the network at the same time to exchange information [8]. Using MQTT, clients can subscribe to all data or specific data from a publisher's information tree. MQTT supports 14 message types [8] and 3 message delivery guarantee types [7]:

1. A message can be delivered at most once.
2. The message will be delivered at least once.
3. The message will be delivered exactly once.

Each party to a message exchange determines the type of message delivery guarantee it wants to use. Usually, these types are the same for the publisher and the subscriber, but there are situations when clients use different types. The MQTT protocol has a special version optimized for embedded wireless devices that do not support the TCP/IP protocol, called MQTT-SN. The optimization principle of this version is to use a shortened header name, which is formed by converting the tape into a two-byte alias. In addition, MQTT-SN allows devices to enter sleep mode when their work is not needed and receive any information that is waiting for them when they wake up. Therefore, the MQTT protocol has advantages in its scalability, architecture that provides loose coupling between clients, reliability, because it guarantees different types of message delivery, and security using SSL/TLS. Disadvantages include slower sending of messages (compared to application protocols built on top of the UDP transport layer protocol).

AMQP (Advanced Message Queuing Protocol) is an open middleware standard for providing messaging at the application level [13]. The protocol is built on the TCP transport protocol and supports 2 types of interac- tion – "request-response" and "publisher-subscriber". As with MQTT, the AMQP server acts as a broker.

AMQP uses the following components to route messages:

1. Exchanges are components that publishers use to send messages.
2. Queues – components from which subscribers receive messages.
3. Bindings – components that send messages from the exchanger to a specific queue.

Exchangers and queues can be permanent or temporary. Long-lived exchangers and queues survive broker restarts, while temporary ones do not (they must be recreated after the broker is started). Exchanges do not store messages. Messages are stored in queues until they are picked up by the client. AMQP uses TLS and SASL for security. In addition, AMQP supports transactions (including distributed ones), but this in turn increases the volume of the message and the time for its transmission and processing. Like MQTT, the AMQP protocol has advantages in scalability and an architecture that provides loose coupling between clients.

Data transfer protocols SOAP, HTTP, REST, CoAP, WebSocket, XMPP, MQTT, and AMQP are evaluated based on qualitative analysis from the Architecture Tradeoff Analysis Method. Each protocol's sensitivity points, tradeoff points, risks, and applicability to the quality attribute scenarios were analyzed. The comparative characteristics of the considered data transfer protocols are described in Table 1, and their advantages and disadvantages are collected in Table 2.

Table 1

Protocol	Transport protocol	Communication model	Exchange template	Security	Correspondence
SOAP	TCP	sync and async	request-response, request	WS-Security, TLS	one-to-one
HTTP/REST	TCP	sync	request-response	TLS	one-to-one
CoAP	UDP	sync	request-response	DTLS	one-to-one
WebSocket	TCP	async	request	TLS	one-to-one
XMPP	TCP	async	request	SASL, TLS	one-to-one
MQTT	TCP	async	publish-subscribe	TLS	one-to-many
AMQP	TCP	sync and async	request-response, publish-subscribe	TLS	one-to-one, one-to-many

Table 2

Protocol	Pros	Cons
SOAP	– high level of security; – high reliability	– the development complexity; – high data transmission latency; – high data processing latency; – impossible to send one message to many recipients with one request; – high resource consumption; – not applicable or limited applicability for the networks with limited bandwidth
HTTP/REST	– ease of development; – possibility to cache answers; – the average latency for data transmission; – the average latency for data processing	– impossible to send one message to many recipients with one request; – low reliability due to the lack of automatic message resend in the event of an error processing this message on the server; – not applicable or limited applicability for the networks with limited bandwidth
CoAP	– ease of development; – low data transmission latency; – low data processing latency; – low resource consumption; – applicable in networks with limited bandwidth	– limited reliability resulting from UDP utilization; – communication issues between NAT components; – impossible to send one message to many recipients with one request
WebSocket	– duplex communication; – low data transmission latency; – low data processing latency; – applicable in networks with limited bandwidth	– the complexity of development; – not supported by all clients; – low reliability due to the lack of automatic connection restoration in case the connection is lost; – impossible to send one message to many recipients with one request; – high resource consumption
XMPP	– the ability to track information about the presence of a node in the network; – the average latency for data transmission; – the average latency for data processing	– low reliability due to lack of confirmation of message processing; – if the XMPP server does not work, the ability to exchange messages becomes impossible; – impossible to send one message to many recipients with one request; – high resource consumption; – not applicable for the networks with limited bandwidth
MQTT	– low data transmission latency; – low data processing latency; – low resource consumption; – possibility to send one message to many recipients with one request; – high reliability; – applicable in networks with limited bandwidth	– the complexity of development; – in the event of a failure of the broker, the message exchange is impossible
AMQP	– the average latency for data transmission; – the average latency for data processing; – low resource consumption; – possibility to send one message to many recipients with one request; – high reliability	– the complexity of development; – in the event of a failure of the broker, the message exchange is impossible; – not applicable for the networks with limited bandwidth

Table 3

Protocol	Typical message exchange scenarios
SOAP	Exchange of messages between business applications that are in the same network
HTTP/REST	Periodic sending of data from IoT devices to applications where the loss of messages is not critical. Message exchange between business applications
CoAP	Periodic data consumption from IoT devices by the application in networks with limited bandwidth
WebSocket	Sending data from IoT devices to applications in real-time with minimal delay in networks with limited bandwidth when message loss is not critical
XMPP	Exchange of messages between IoT devices. IoT devices' status monitoring
MQTT	Sending data from IoT devices to applications in real-time with minimal latency over bandwidth-constrained networks
AMQP	Bi-directional communication between IoT devices and applications with transactional support

Basic usage scenarios for each protocol are listed in Table 3.

According to the tests performed to deliver 1K messages over MQTTS and HTTPS – MQTTS was shown 20 times faster and required 50 times less traffic on the task of posting consistent time-valuable data and is more efficient from a power consumption point of view [14]

Considering the above scenarios and advantages such as low latency of data transmission and processing, applicability in networks with limited bandwidth, and high reliability and scalability that fulfill non-functional requirements of a digital platform, the MQTT protocol is proposed for real-time transmission of geospatial and telemetry data time from IoT devices to cloud application.

6. Conclusions

The real-time data streaming from machinery/equipment for monitoring or predictive detection of vehicle malfunctioning issues heavily depends on the selected data transmission protocol. It significantly influences the efficiency, reliability, and overall performance of the system that could not be addressed after the system is built and delivered to the customers without significant efforts to re-work and re-deliver the solution. Qualitative analysis from the Architecture Tradeoff Analysis Method was utilized to compare the set of transfer protocols SOAP, HTTP, REST, CoAP, WebSocket, XMPP, MQTT, AMQP based on their sensitivity points, tradeoff points, risks, and applicability to the quality attribute scenarios and not limit the selection by properties comparison only. Based on the performed analysis, the MQTT protocol is chosen for the scenario of real-time data transmission in the digital platform since it fulfills the high-level vision of the system architecture and addresses non-functional requi- rements like efficiency, reliability, and overall perfor- mance of the data delivery.

7. Gratitude

The authors thank the Editorial Board of the Scientific journal "Measuring Equipment and Metrology" for their support.

8. Mutual claims of authors

The authors have no claims against each other.

References

[1] Semernia K. V. Suchasni finansovo-ekonomichni problemy funktsionuvannia ta rozvytku ahrarnykh pidpryiemstv.Aktualni problemy sotsialno-ekonomichnykh system v umovakh transformatsiinoi ekonomiky: Zbirnyk naukovykh statei za materialamy IV Vseukrainskoi naukovopraktychnoi konferentsii (12–13 kvitnia 2018 r.) Chastyna 1. Dnipro: NMetAU, 2018. 367 p. Access mode: https://nmetau.edu.ua/file/sbornik_18_1.pdf

[2] S. A. Al-Suhaibani, M. F. Wahby. Farm tractors breakdown classification. Journal of the Saudi Society of Agricultural Sciences. Riyadh: King Saud University, 2017, 294–298. DOI: 10.1016/j.jssas.2015.09.005

[3] Global Network Against Food Crises. 2022 Global Report on Food Crises [Electronic resource]. FSIN, 2022, 5–10. Access mode: https://docs.wfp.org/api/documents/WFP-0000138913/download/

[4] R. Khodabakhshian. Prediction of repair and maintenance costs of farm tractors by using Preventive Maintenance. International Journal of Agriculture Sciences. Pune: Bioinfo Publications, 2011, 39–42. DOI: 10.9735/0975-3710.3.1.39-44

[5] M. Xiao, W. Wang, K. Wang, W. Zhang, H. Zhang. Fault Diagnosis of High-Power Tractor Engine Based on Competitive Multiswarm Cooperative Particle Swarm Optimizer Algorithm. London: Hindawi, 2020, 1–13. DOI: 10.1155/2020/8829257

[6] Y. Lafon, C. Bournez. SOAP 1.2 Pressrelease [Electronic resource]. W3C, 2003. Access mode: https://www.w3.org/2003/06/soap12-pressrelease

[7] S. Misra, A. Mukherjee, A. Roy. Introduction to IoT. – Cambridge: Cambridge University Press, 2021, 184–200. DOI: 10.1017/9781108913560

[8] IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things / D. Hanes, G. Salgueiro, P. Grossetete, R. Barton, J. Henry. Indianapo- lis, Indiana: Cisco Press, 2017, 177–204. ISBN: 978-1587144561. Access mode: https://www.amazon.com/ IoT-Fundamentals-Networking-Technologies-Protocols/dp/1587144565

[9] N. M. Shaikh, Y. Ingle. Application of Restful APIs in IOT: A Review. Haryana: iJRASET, 2021, p. 9. DOI: 10.22214/ijraset.2021.33013

[10] Alabbas Alhaj, A. Constraint Application Protocol (CoAP) for the IoT. Frankf. Univ. Appl. Sci., 2018, p. 1. DOI: 10.13140/RG.2.2.33265.17766

[11] Z. Shelby, K. Hartke, C. Bornamn. The Constrained Ap- plication Protocol (CoAP) [Electronic resource]. IETF, 2014. Access mode: https://datatracker.ietf.org/ doc/html/rfc7252

[12] I. Fette, A. Melnikov. The WebSocket Protocol [Elec- tronic resource]. IETF, 2011. Access mode: https://datatracker.ietf.org/doc/html/rfc6455

[13] S. Vinoski. IEEE Internet Computing, Vol. 10, Iss. 6. Cyprus: University of Cyprus, 2006, 87–89. DOI: 10.1109/MIC.2006.116

[14] J. Barnitskyi. HTTP vs MQTT performance tests [Elec- tronic resource]. Flespi, 2018. Access mode: https://flespi.com/blog/http-vs-mqtt-performance-tests