ISSN 2224-087X. Електроніка та інформаційні технології. 2022. Випуск 19. С. 75–85
Electronics and information technologies. 2022. Issue 19. P. 75–85

УДК 629.31 DOI: https://doi.org/10.30970/eli.19.7

# CUSTOMIZABLE IOT SOLUTION BASED ON ESP32 MCU

I. Berizka, R. Romanyshyn, O. Savitskyy

*Ivan Franko National University of Lviv,*
*107, Tarnavskoho St., 79017 Lviv, Ukraine*
*ihor.berizka@lnu.edu.ua*

Sensing and perception are one of the key aspects in robotics. It is necessary for robots to be able to measure some physical parameters in the world and making sense of such data for performing different tasks and act according to surrounding environment conditions. Right now, robotics is seeing a revolution in use of sensors. Traditionally robots have been designed to have maximum stiffness and applications have been designed to be predictable in their operation. As robots emerge from the fences areas and are designed for a wider range of applications: from collaborative robotics to autonomously driving cars. It is essential to have perception capabilities that allow estimation of the state of the robot but also the state of the surrounding environment. Due to these new requirements the importance of sensing and perception has increased significantly over the last decade and will without doubt to continue to grow in the future [18].

The objective of this paper is to develop customizable IoT solution for remote monitoring of selected physical parameters. The created prototype is based on ESP32 MCU and consists of two parts: hardware and software & cloud. Also, mobile application for wireless device configuration and online sensors data monitoring was developed. Several filtering methods were implemented for proper data preprocessing of near-surface water level, outside temperature, solar power and self-diagnostics sensors' data. Also, possibility of remote software upgrade is implemented. The prototype was setup with solar power supply and some experimental results are demonstrated in order to approve that this approach is fully functional, autonomous and low-cost.

*Keywords*: IoT, digital filters, wireless device configuration, MQTT, cloud, sensing and perception.

## Hardware and technologies

### *ESP32 Module*

In this project ESP32 module designed by Espressif company is used as main MCU. It is supposed to be the most effective solution because of the following reasons:

- Robust design: Module is capable of functioning reliably in industrial environments, with an operating temperature ranging from -40C to +125C.
- Low-power consumption: Module is engineered for mobile devices, wearable electronics and IoT applications.
- High level of integration: ESP32 is highly-integrated with built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules.

- Hybrid WIFI and Bluetooth chip: Module can perform as a standalone system or as a slave device to a host MCU, reducing communication stack overhead on the main application processor. ESP32 can interface with other systems to provide WIFI and Bluetooth functionality through its SPI / SDIO or I2C / UART interfaces.
- Low cost: Module costs approx. 10$.

Fig.1 ESP32 module.

For more information about module visit official website [1].

*A. Sensors*

- 2x INA260 current sensors: digital-output, current, power, and voltage monitor with an I2C interface with an integrated precision shunt resistor. Sensor features up to 16 programmable addresses on the I2C interface. The digital interface allows programmable alert thresholds, analog-to-digital converter (ADC) conversion times, and averaging. To facilitate ease of use, an internal multiplier enables direct readouts of current in amperes and power in watts [4].
- DS18B20 temperature sensor: digital thermometer which provides 9-bit Celsius temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. Device communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with MCU. Each sensor has its own unique serial code, which allows multiple sensors on the same bus [5].
- GUT800 water level sensor: ultrasonic sensor with analog output in range of 4-20mA.

*B. Communication protocol*

MQTT is one of the most commonly used protocols on IoT projects. In addition, it is designed as a lightweight messaging protocol that uses publish/subscribe operations to exchange data between clients and the server. Furthermore, its small size, low power usage, minimized data packets and ease of implementation make the protocol ideal for the “machine-to-machine” or “Internet of Things” world [2]. General structure and data flow of typical IoT solutions are shown on Figure 2.

Like any other internet protocol, MQTT is based on clients and a server. Likewise, the server is the guy who is responsible for handling the client`s requests of receiving or sending data between each other. MQTT server is called broker and the clients are simply connected devices.

So:

- When a device (a client) wants to send data to the broker, this operation is called `publish`.
- When a device (a client) wants to receive data from the broker, this is called `subscribe`.

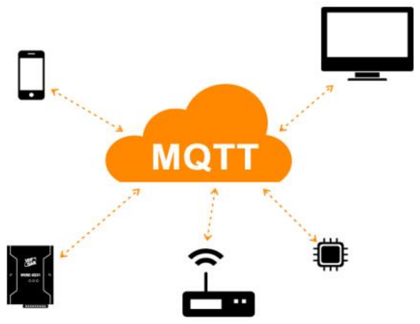


Fig.2 Data flow in systems, which uses MQTT protocol.

In addition, these clients are publishing and subscribing to topics. So, the broker here is the one that handles the publishing/subscribing actions to the target topics.

Typical MQTT components are:

- Broker, which is the server that handles the data transmission between the clients.
- A topic, which is the place a device want to put or retrieve a message to/from.
- The message, which is the data that a device receives, “when subscribing” from a topic or send “when publishing” to a topic.
- Publish, is the process a device does to send its message to the broker.
- Subscribe, where a device does to retrieve a message from the broker.

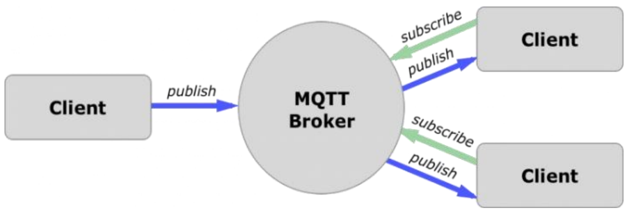


Fig. 3. Client-server architecture of MQTT protocol.

## *C. Cloud*

AWS Kinesis Stream [7] is used for fetching data from CloudMQTT broker [10].

Amazon Kinesis makes it easy to collect, process, and analyze real-time, streaming data so you can get timely insights and react quickly to new information. Amazon Kinesis offers key capabilities to cost-effectively process streaming data at any scale, along with the flexibility to choose the tools that best suit the requirements of your application.

Then, ƛ – function [17] runs every 15 min and adds timestamp to received data.

AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume - there is no charge when your code is not running.

Next step is AWS Delivery Stream, which is part of Amazon Kinesis Data Firehouse [13]. It is fully managed service for delivering real-time streaming data to destinations such as Amazon S3.

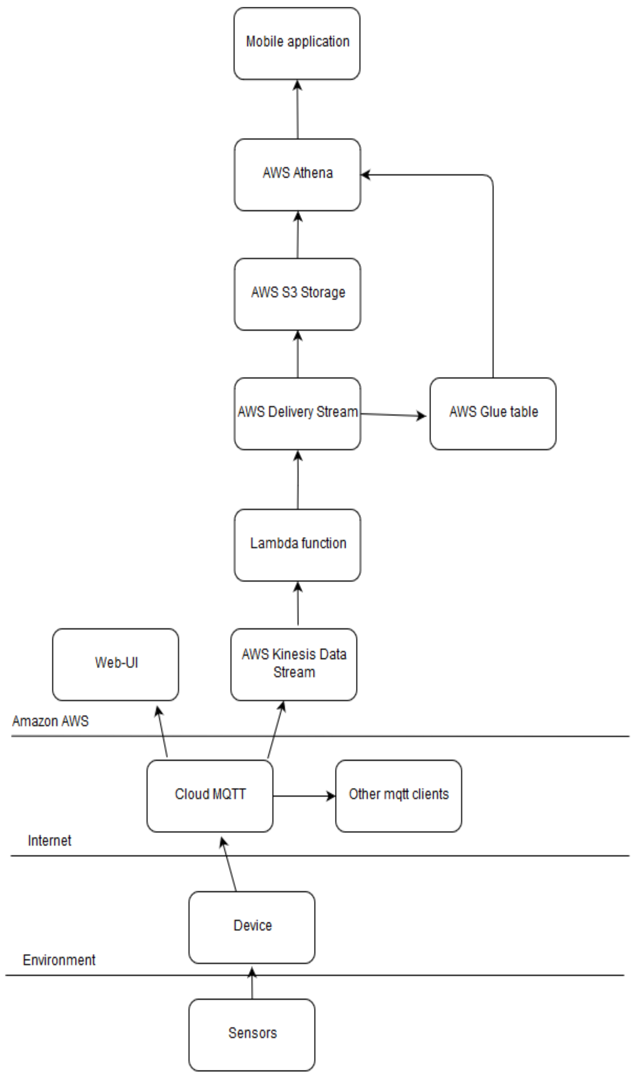


Fig. 4. Data flow in system.

Then data is processed using Amazon Glue [14]. It is used to organize, validate and format data. AWS Glue is a fully managed ETL (extract, transform, and load) service that makes it simple and cost-effective to categorize your data, clean it, enrich it, and move it reliably between various data stores.

Amazon S3 service is used as data storage. This is an object storage service that offers industry-leading scalability, data availability, security, and performance [3]. Data is stored in parquet format [8].

### *D. Wireless device configuration*

Communication between device and mobile phone is based on TCP/IP sockets. TCP/IP sockets are chosen because of their pros:

- Reliable byte-stream channel (in order, all arrive, no duplicates)
- Flow control
- Connection-oriented
- Bidirectional

If device (ESP32) is in configuration state – it turns on AP (Access Point) mode and runs TCP/IP server which listens for incoming connections. Mobile app fetches data required for connection (such as IP and port number) from QR-code. Data in QR-code is in JSON format. This gives opportunity to easily interpret and store nested and complex configs.

Then mobile app tries to connect to device. When connection is established, client (mobile app) sends serialized JSON file with configs via socket to server (device). On server side received raw data is deserialized back to JSON format. If configs are valid, device tries to connect to specified WIFI network and to MQTT broker. If previous steps were successful – device turns off AP mode and starts data acquisition process. Otherwise, AP is on and appropriate error code is returned back to app. One of the most pros of raw TCP/IP sockets is that server (device) can initiate data exchange, which is not possible in standard RESTFull API.

### *E. Prototype design*

3D model of the prototype is created in Altium designer. The combination of all hardware described in previous sections is presented here. Also, Device Configuration button and Device State LED are provided on the board.

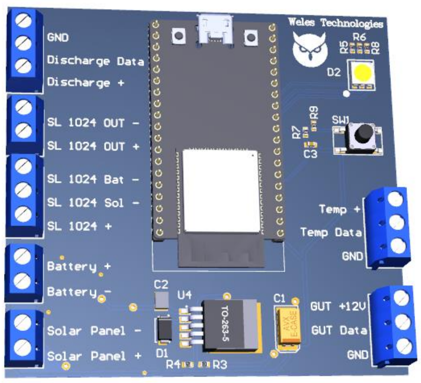


Fig.5. 3D model of prototype

*F. Data preprocessing*

The developed firmware contains following filtering methods for the acquired data preprocessing [15, 16]:

- median filter
- average filter [6]
- exponential smoothing [6]

Median filtering works perfectly for slowly varying data and is not sensitive to high level outliers. On the other hand - average filter and exponential smoothing works well with fast changing signals. As temperature and depth values change slowly (in normal conditions) - for preprocessing is used median filter with window size = 9. As battery charging\discharging can change really fast over the time - for preprocessing are used average and exponential smoothing filters.

## Solution architecture

Systems consists of two parts: hardware for data acquisition and sending to cloud and cloud for processing and plotting data (Figures 5, 6).

Device configuration is done wirelessly using mobile app. Each device has its own configs: access point name, password, device type (depends on set of connected sensors) and etc. Some of configs are saved as QR-code [11] printed on top of device. They are used for device identification and proper data transmitting. After device is recognized – you are connected to it and can configure additional parameters: set connected sensors, change default sensors configs and etc. When device is configured – it starts fetching data from sensors and sending to MQTT cloud in real-time.

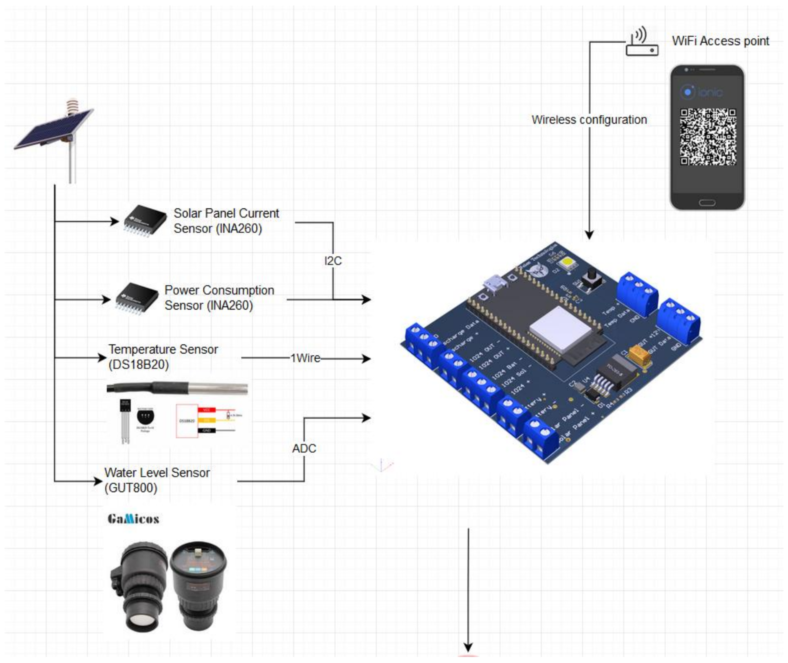

WiFi Access point
Wireless configuration
Solar Panel Current Sensor (INA260)
I2C
Power Consumption Sensor (INA260)
Temperature Sensor (DS18B20)
1Wire
ADC
Water Level Sensor (GUT800)
GaMicos

Fig. 6. Hardware part of solution

CloudMQTT service [10] is selected as MQTT cloud. Data is fetched from MQTT cloud as described in previous chapter. Now multiple ways for data monitoring and processing can be developed. We've managed to implement basic mobile application with Ionic Framework [9] that makes it possible to be used for both Android and IOS, and static Web SPA

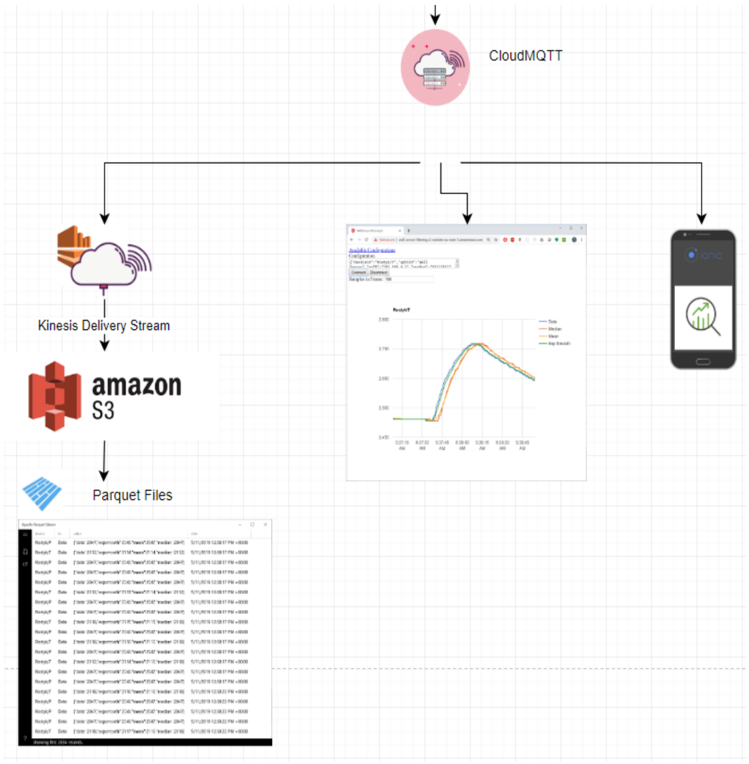


Fig. 7. Cloud part of solution.

## Prototype

Device manufactured according to model on figure 5 is shown below.

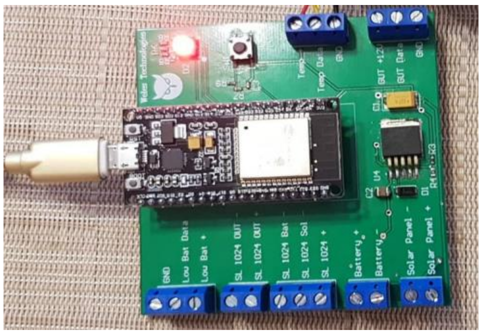


Fig. 8. Scheme printed according to 3D model on figure above.

Setup without solar panel is shown on figure 9. It consists of device described in previous chapters and power supply.

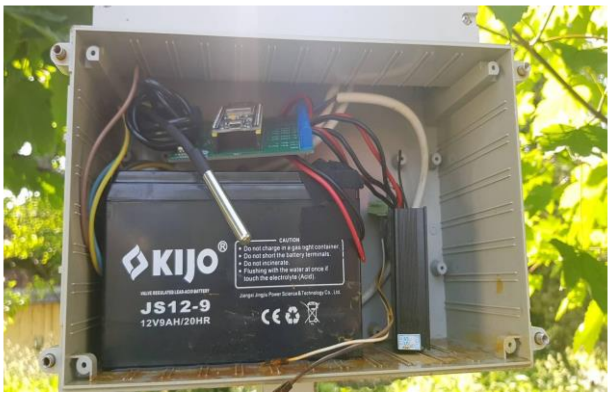


Fig. 9. Working prototype

Power supply consist of solar panel, charge controller and battery.

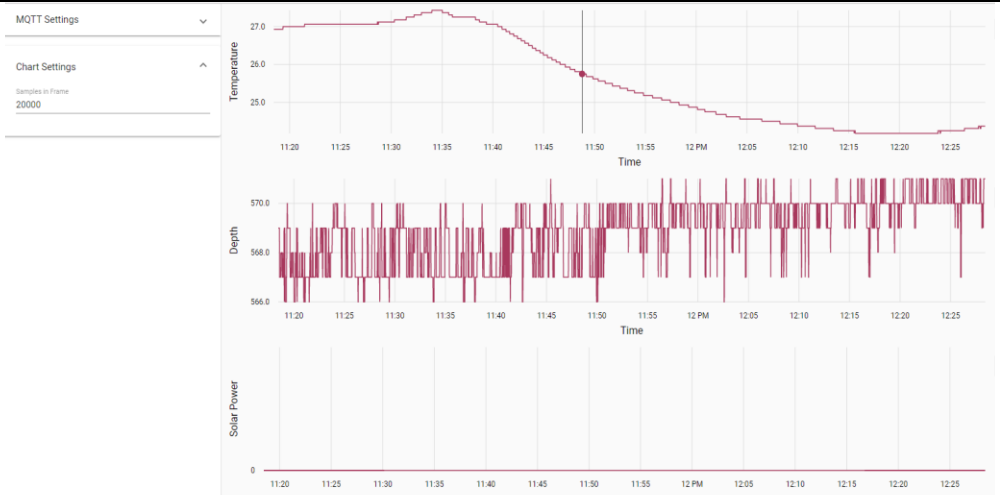
Device includes two INA260 sensors: one for monitoring power consumption and second – for monitoring battery charging. This is important data, because we need to know if selected solar panel provides enough power all over the year.

GUT800 sensor also requires additional hardware component and software configuration. As sensor generates analog current signal in range of 4-20mA - there is additional precise resistor in schematic. This resistor is used as current to voltage converter. Its value is selected to generate voltage in 0-1V range. As standard ADC configurations on ESP32 module provides reference voltage of 3.3V - additional software configs are applied to set reference voltage to 1.1V. This configs allow to measure analog signal more precise.

Also, there is led for monitoring current device state. Device can be in the following states:

- Red led is blinking: no WIFI configs. Launch AP mode. Wait for connections. If there is available client - read configs.
- Yellow led is blinking: not connected to WIFI.
- Green led is blinking: no MQTT configs.
- Blue led is blinking: provided wrong MQTT credentials.
- Violet led is blinking: no configs for sensors.
- All leds are blinking: device is working normally.

Web-site interface is shown on figures 10 and 11.



MQTT Settings

Chart Settings

Samples in Frame: 20000

Fig. 10. Data received from device in real-time

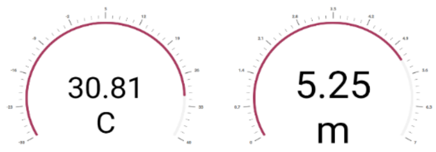


Fig. 11. Gauge chart on web-site

**Conclusion**

In this paper we introduce a dedicated customizable IoT solution for remote monitoring of selected physical parameters. The created prototype is based on ESP32 MCU and comprises hardware module and software part that enables cloud communication. Also, mobile application for wireless device configuration and online sensors data monitoring was developed. Several filtering methods were implemented for proper data pre-processing of near-surface water level, outside temperature, solar power and self-diagnostics data from sensors. Possibility of remote software upgrade is implemented. The prototype was setup with solar power supply in the field. Stability, functional and regression testing was performed to elucidate some experimental cases. This proves that developed prototype is fully functional, autonomous, low-cost, and might be used in different domains: from IoT to robotics.

## References

[1] Espressif Systems – Wi-Fi and Bluetooth chipsets and solutions.
Retrieved from https://www.espressif.com/

[2] MQTT protocol. Retrieved from https://1sheeld.com/mqtt-protocol/

[3] Amazon S3. Retrieved from https://aws.amazon.com/s3/?nc1=h_ls

[4] INA260 current sensor datasheet.
Retrieved from http://www.ti.com/lit/ds/symlink/ina260.pdf

[5] DS18B20 temperature sensor.
Retrieved from https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

[6] *I. Karbovnyk, Z. Lybun, V. Rabyk*. Data mining practise. Lviv, 2015 (in Ukrainian).

[7] Amazon Kinesis. Retrieved from https://aws.amazon.com/kinesis/?nc1=h_ls

[8] Parquet data format. Retrieved from https://parquet.apache.org/documentation/latest/

[9] Ionic framework. Retrieved from https://ionicframework.com/docs/v3/intro/tutorial/

[10] CloudMQTT documentation.
Retrieved from https://www.cloudmqtt.com/docs/index.html

[11] QR-code. Retrieved from https://www.the-qrcode-generator.com/whats-a-qr-code

[12] Amazon Athena. Retrieved from https://aws.amazon.com/ru/blogs/aws/amazon-athena-interactive-sql-queries-for-data-in-amazon-s3/?fbclid=IwAR2py-W0oAR21mQjDNbiS1GynlujRyopKrAJcwdbUmARmJ3CqncnSzzlAdU

[13] Amazon Kinesis Data Firehouse.
Retrieved from https://docs.aws.amazon.com/firehose/latest/dev/firehose-dg.pdf

[14] AWS Glue. Retrieved from https://docs.aws.amazon.com/glue/latest/dg/glue-dg.pdf

[15] *A. Lozynskyy, I. Romanyshyn, B. Rusyn et al*. Robust Approach to Estimation of the Intensity of Noisy Signal with Additive Uncorrelated Impulse Interference / IEEE Second Intern. Conf. on Data Stream Mining & Processing (DSMP). August 21-25, 2018, Lviv, Ukraine. / Conf. Paper / Lviv: 2018. – Pp.251-254. DOI: 10.1109/DSMP.2018.8478625.

[16] *A. Lozynskyy, I. Romanyshyn, B. Rusyn*. Intensity Estimation of Noise-Like Signal in Presence of Uncorrelated Pulse Interferences // Radioelectronics and Communications. – 2019, vol. 62, No 5. – Pp. 214-222. DOI: 10.3103/S0735272719050030.

[17] AWS Lambda. Retrieved from https://aws.amazon.com/lambda/

[18] Springer handbook of Robotics by Bruno Siciliano, Oussame Khatib, 2nd edition, Springer, Berlin 2016

## ПРОГРАМНО-КОНФІГУРОВАНЕ IOT-РІШЕННЯ НА ОСНОВІ МІКРОКОНТРОЛЕРА ESP32

**І. Берізка, З. Романишин, О. Савицький**

*Львівський національний університет імені Івана Франка,*
*вул. Ген. Тарнавського, 107, Львів, 79017, Україна*
*ihor.berizka@lnu.edu.ua*

Вимірювання та аналіз даних відіграє одну з ключових ролей у сучасних технологічних галузях. Наприклад, для робототехніки доволі важливо мати можливість виміряти різні параметри робочого середовища та адекватно їх обробляти. Ці дані надають можливість виконувати поставлені задачі та відповідним чином реагувати на зміни робочого середовища. Сьогодні можна спостерігати революцію у застосуванні сенсорів у робототехнічних системах. Раніше роботів розробляли максимально простими та спеціалізованими: тобто вони мали працювати у чітко визначеному середовищі та виконувати максимально передбачувано одну конкретну задачу. Проте, на сьогоднішньому етапі, розробники намагаються максимально розширити ці обмежуючі рамки. Таким чином, все частіше можна спостерігати за розробками таких складних систем як колаборативні роботи та повністю автономні машини. Саме тому необхідно мати легко розширюваний фреймворк, який надає змогу визначити стан та виміряти різні параметри не тільки робота, а й робочого середовища. Аналізуючи всі ці вимоги можна зробити висновок, що потреба у таких розробках вже зростає та зростатиме ще більше. Адже рівень складності систем поступово зростає, разом з потребою моніторингу все більшої кількості параметрів.

Метою цієї роботи є розробка IoT рішення та фреймворку для збору та моніторингу вибраних фізичних параметрів віддалено. Розроблений прототип базується на мікроконтролері ESP32 та складається з кількох частин: апаратної та програмної, яка орієнтована на роботу з хмарним сервісом. Також розроблено мобільний додаток, через який можна віддалено налаштовувати пристрій і спостерігати за даними в режимі реального часу. Протестовано кілька методів цифрової фільтрації для попередньої обробки даних. У роботі наведено приклад з вимірюванням рівня приповерхневих вод, температури довкілля, потужності, яку виробляє сонячна панель та фіксування деяких діагностичних даних системи. Також реалізовано віддалене оновлення програмного забезпечення на пристрої. Прототип тестувався у польових умовах протягом шести місяців та показав стабільну роботу. На основі тестів можна зробити висновок про функціональність запропонованого фреймворку, автономність та порівняно низьку вартість.

*Ключові слова:* інтернет речей, цифрові фільтри, конфігурація бездротових пристроїв, протокол MQTT, хмара, сенсори і сприйняття.

*Стаття надійшла до редакції 07.10.2022.*
*Прийнята до друку 25.10.2022.*