# PROTOTYP ZINTEGROWANEGO SYSTEMU DO MONITOROWANIA BEZPIECZEŃSTWA I PARAMETRÓW ŚRODOWISKOWYCH W POMIESZCZENIACH

european student magazine
Numer 1 / 2025

Wojciech **Błaszczyk**[1], Aleksandra **Sikora**[2*]

[1] Politechnika Świętokrzyska, Wydział Elektrotechniki, Informatyki i Automatyki, al. Tysiąclecia Państwa Polskiego 7, 25-314 Kielce, Polska, s092651@student.tu.kielce.pl
[2] Politechnika Świętokrzyska, Wydział Elektrotechniki, Informatyki i Automatyki, al. Tysiąclecia Państwa Polskiego 7, 25-314 Kielce, Polska, asikora@tu.kielce.pl
[*] Corresponding author

**Streszczenie –** W artykule przedstawiono zaprojektowanie, zbudowanie oraz implementację oprogramowania dla prototypu zintegrowanego systemu monitorowania bezpieczeństwa i warunków środowiskowych w pomieszczeniach. System umożliwia obserwację temperatury, wilgotności, obecności dymu oraz stanu otwarcia drzwi. W projekcie zastosowano technologie Python, Flask, MQTT, Socket.IO, C++, HTML, CSS i JavaScript oraz czujniki DHT11, MQ-2 i kontaktron CMD14. Do wykonania jednostki centralnej systemu wykorzystano Raspberry Pi 4, współpracujące z Arduino Uno R4 WiFi. Opracowany system zapewnia wizualizację monitorowanych parametrów w interfejsie webowym. Przeprowadzone testy wydajnościowe i funkcjonalne potwierdziły poprawność działania oraz możliwość dalszego rozwoju systemu.

**Słowa kluczowe** – Arduino, czujnik dymu, czujnik otwarcia drzwi, czujnik temperatury i wilgotności, Raspberry Pi

## WSTĘP

Internet Rzeczy (IoT) to technologia umożliwiająca połącznie urządzeń zdolnych do wzajemnej komunikacji, zbierania danych oraz przesyłania w czasie rzeczywistym. Dzięki temu IoT znajduje zastosowanie w wielu obszarach, takich jak inteligentne budynki, przemysł czy systemy monitorowania bezpieczeństwa. Rozwój tej technologii pozwala na automatyzację procesów oraz zwiększenie kontroli nad środowiskiem użytkownika.

W artykule przedstawiono prototyp zintegrowanego systemu monitorowania warunków środowiskowych i bezpieczeństwa w pomieszczeniach. System ten odczytuje temperaturę, wilgotność, wykrywa dym oraz monitoruje stan otwarcia drzwi. Jest to model testowy, przedstawiający potencjał technologii IoT w tego typu rozwiązaniach. Dzięki modułowej budowie możliwe jest przeprowadzanie testów działania i dalsza optymalizacja i rozbudowa systemu.

## I. ARCHITEKTURA SYSTEMU

Zaprojektowany system monitorowania bezpieczeństwa i warunków środowiskowych opiera się na trzech głównych warstwach: sprzętowej, przetwarzania danych oraz prezentacji. Każda z nich realizuje określane zadania, zapewniając prawidłowe działanie i możliwość dalszej rozbudowy systemu.

Warstwa sprzętowa obejmuje wszystkie fizyczne komponenty systemu:

- Czujniki DHT11 [4], MQ-2 [5] i kontaktron CMD14 [2], zbierające dane o panujących warunkach środowiskowych i zdarzeniach (obecność dymu, otwarcie drzwi).
- Raspberry Pi 4B [13] odbiera dane z czujników podłączonych bezpośrednio przez GPIO.
- Arduino Uno R4 WiFi [1] odbiera dane z czujnika MQ-2 i generuje sygnały wyjściowe.
- Buzzer aktywowany jest w przypadku wykrycia dymu.

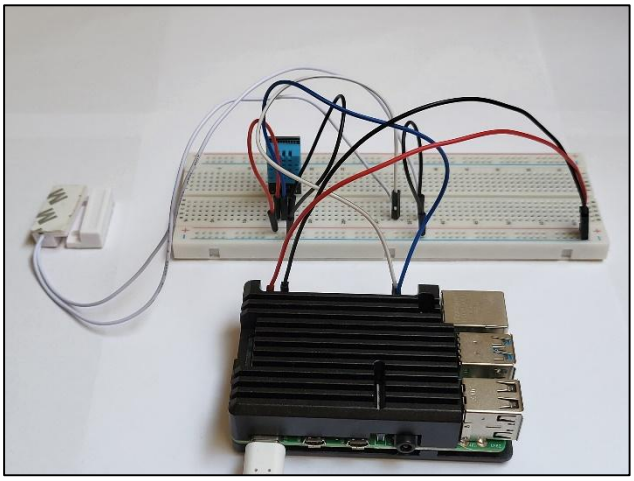
Warstwa przetwarzania danych:

- Raspberry Pi 4B pełni funkcję centralnej jednostki przetwarzającej dane. Obsługuje serwer MQTT (Mosquitto) [6], który umożliwia odbieranie danych z Arduino, a także przetwarza zebrane informacje w Pythonie. Na ich podstawie generowane są powiadomienia CallMeBot [3] oraz hostowany jest interfejs użytkownika.
- Arduino przetwarza dane lokalnie, porównując odczytane wartości z ustalonym progiem alarmowym.

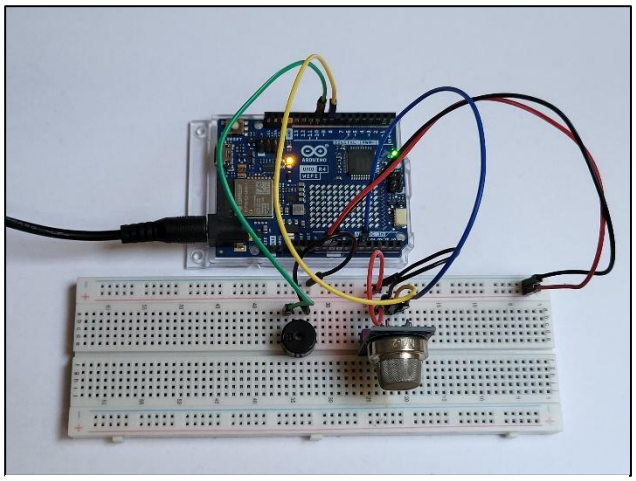
Warstwa prezentacji:

- Interfejs użytkownika opracowany przy użyciu HTML [8], CSS [7] i JavaScript [9], hostowany jest na serwerze Flask [11]. Odpowiada za wyświetlanie rzeczywistych danych monitorowanych parametrów.
- CallMeBot [3] do przesyłania powiadomień alarmowych za pośrednictwem WhatsApp.

Fizyczne połączenie komponentów systemu przedstawiono na poniższych zdjęciach. Rysunek 1 ukazuje sposób podłączenia czujników do jednostki centralnej, natomiast Rysunek 2 przedstawia połączenie elementów z Arduino.



**Rys. 1 Połączenie elmentów z Raspberry Pi**



**Rys. 2 Połączenie elementów z Arduino**

## II. MECHANIZMY PRZETWARZANIA I KOMUNIKACJI W SYSTEMIE

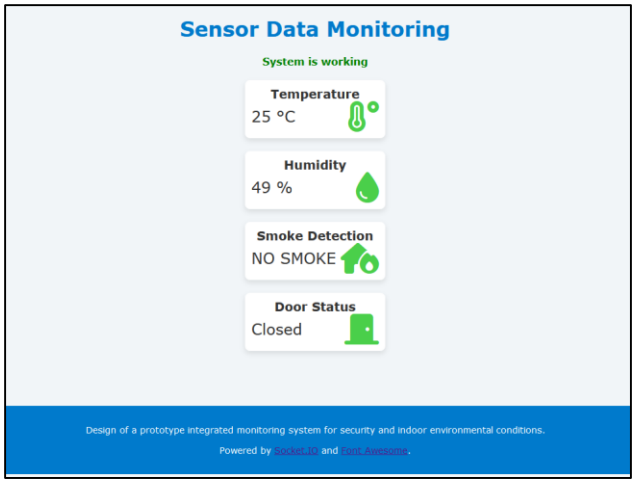
Dane pozyskiwane z czujników podlegają analizie, a w przypadku wykrycia przekroczenia wartości progowych generowane są alerty. Każdy z monitorowanych parametrów jest odczytywany w określonych odstępach czasu, co zapewnia stałą aktualizację informacji. W celu zapewnienia ciągłości działania systemu zastosowano mechanizmy eliminacji błędów, które umożliwiają kontynuację pracy w przypadku niedostępności danych lub awarii jednego z komponentów.

Pozyskane dane są interpretowane w warstwie przetwarzania, gdzie przeprowadzana jest ich analiza w czasie rzeczywistym. W celu zapewnienia stabilnej pracy systemu wdrożono mechanizmy zarządzania wątkami oraz synchronizacji operacji [12]. Dzięki temu możliwe jest równoczesne monitorowanie czujników, obsługa powiadomień oraz obsługa interfejsu użytkownika. Zaimplementowano również mechanizm monitorowania działania aplikacji, oparty na cyklicznym wysyłaniu sygnału kontrolnego (heartbeat). Brak odpowiedzi w czasie 10 sekund sygnalizuje potencjalną usterkę.

W systemie wdrożono mechanizm powiadomień informujących o wykryciu potencjalnych zagrożeń. Pozwala on na ustalenie odstępów czasowych pomiędzy wysłaniem alertu, zapobiegając ich nadmiarowości. Komunikaty generowane są automatycznie i przekazywane użytkownikowi w formie wiadomości tekstowych, zawierających informacje o wartościach monitorowanych parametrów.

Do realizacji komunikacji wykorzystano protokół MQTT (Message Queue Telemetry Transport) w celu przesyłania danych z Arduino do Raspberry Pi oraz Socket.IO [14] do komunikacji pomiędzy backendem, a interfejsem użytkownika. Interfejs użytkownika odbiera dane przesyłane za pośrednictwem Socket.IO i aktualizuje je w sposób automatyczny, bez konieczności odświeżania strony.

Interfejs użytkownika przedstawiony na Rysunku 1, zapewnia wizualizację danych w czasie rzeczywistym, zapewniając dostęp do informacji o stanie systemu oraz aktualnych wartości pomiarów. W przypadku wykrycia zagrożenia odpowiednie sekcje zmieniają kolorystykę. Zastosowane rozwiązanie pozwala na dostosowanie interfejsu również do urządzeń mobilnych [10].

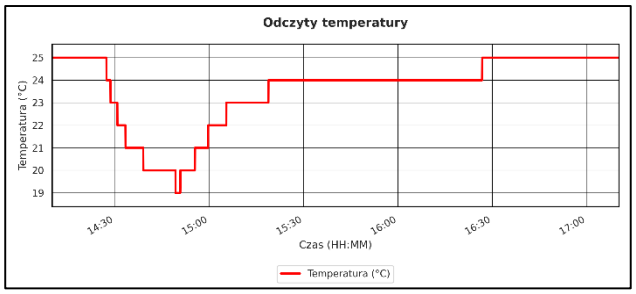



**Rys. 3 Interfejs użytkownika**

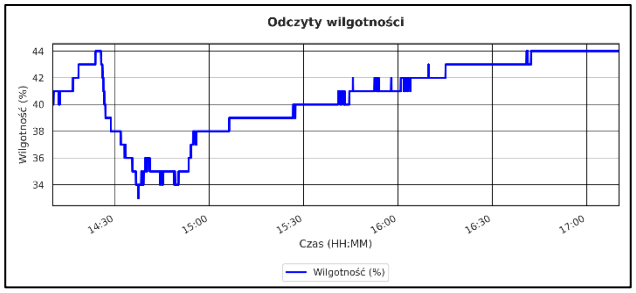
## III. TESTY FUNKCJONALNE

Testy funkcjonalne zostały przeprowadzone w celu weryfikacji poprawności działania podstawowych funkcji. Sprawdzono czy elementy takie jak: zbieranie danych z czujników, analizowanie ich w czasie rzeczywistym, generowanie powiadomień o zagrożeniach oraz prezentowanie monitorowanych parametrów w interfejsie użytkownika odbywa się poprawnie. Obejmowały one różnorodne scenariusze, aby ocenić reakcje systemu na poszczególne zdarzenia.

Prawidłowość odczytu danych z czujnika DHT11, monitorującego temperaturę i wilgotność, została zweryfikowana na podstawie analizy wartości pomiarowych zarejestrowanych w czasie testów. Odczyty były zapisywane przez trzy godziny, a uzyskane wyniki przedstawiono na wykresach. Na Rysunku 4 zaprezentowano zmiany temperatury w czasie, natomiast Rysunek 5 przedstawia odczyty wilgotności.




**Rys. 1 Odczyty temperatury**

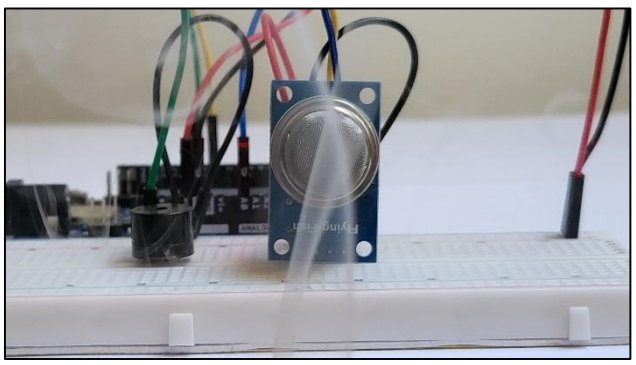



**Rys. 2 Odczyty wilgotności**

Czujnik rejestruje jedynie wartości całkowite z dokładnością dla temperatury ± 2˚C, a dla wilgotności ± 5% [4]. Otrzymane wyniki potwierdzają poprawność jego działania.

Działanie czujnika drzwi sprawdzono poprzez wielokrotne zamykanie i otwieranie drzwi. System poprawnie rejestrował każdą zmianę stanu, generując powiadomienia oraz komunikaty w konsoli.

Sprawność czujnika dymu oceniono poprzez monitorowanie wartości odczytywanych co 2 sekundy. Testy wykazały stabilność odczytów w warunkach bezdymnych oraz prawidłowe wykrywanie wzrostu stężenia w sytuacji kontrolowanej ekspozycji czujnika na dym.



**Rys. 6 Czujnik MQ-2 w obecności dymu**

Poprawność analizy wartości progowych sprawdzono przez celowe obniżenie dopuszczalnych wartości temperatury i wilgotności w pliku konfiguracyjnym systemu. W trakcie testów przeprowadzono trzy scenariusze:

1. Przekroczenie dopuszczalnych wartości zarówno dla temperatury, jak i wilgotności jednocześnie.
2. Przekroczenie dopuszczalnej wartości tylko dla temperatury.
3. Przekroczenie dopuszczalnej wartości tylko dla wilgotności.

W każdym przypadku system poprawnie rejestrował zdarzenie, wysyłając powiadomienia oraz zmieniając kolor odpowiednich ikon na czerwony.

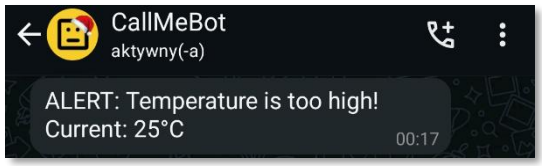
Podobne testy przeprowadzono dla czujnika drzwi, obejmując następujące scenariusze:

1. Otwarcie drzwi w trakcie działania systemu.
2. Zamknięcie drzwi w trakcie pracy systemu.
3. Drzwi otwarte w momencie uruchomienia systemu.

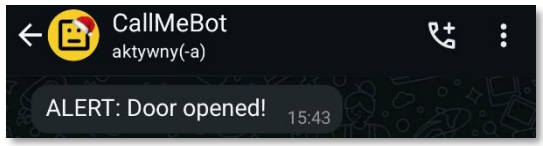
Wszystkie przypadki zakończyły się poprawnym rozpoznaniem stanu drzwi, co zostało potwierdzone odpowiednimi komunikatami oraz zmianą kolorystyki interfejsu użytkownika.

Działanie czujnika dymu przetestowano w kontrolowanych warunkach. W momencie przekroczenia progu 180 ppm system generował powiadomienia alarmowe, a ikona czujnika w interfejsie zmieniła kolor na czerwony. Powrót wartości poniżej progu skutkował automatycznym wyłączeniem alarmu.

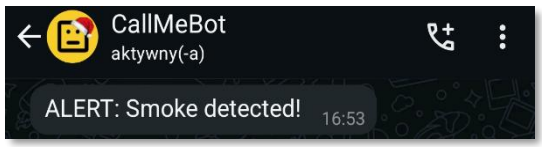
System umożliwia przesyłanie powiadomień WhatsApp za pośrednictwem CallMeBot. Testy wykazały, że wszystkie komunikaty docierały do obiorcy. Zweryfikowano również skuteczność ograniczenia częstotliwości wysyłania alertów co zapobiega generowaniu nadmiarowych powiadomień. Na poniższych zrzutach ekranu przedstawiono odebrane powiadomienia.




**Rys. 7 Powiadomienie dla zbyt wysokiej temperatury**




**Rys. 8 Powiadomienie dla otwartych drzwi**




**Rys. 9 Powiadomienie dla wykrycia dymu**

## IV. TESTY NIEFUNKCJONALNE

Testy niefunkcjonalne zostały przeprowadzone w celu sprawdzenia elementów systemu, wpływających na jakość jego działania. Skoncentrowano się na weryfikacji wydajności, reakcji na sytuacje awaryjne oraz możliwości korzystania z interfejsu na różnych urządzeniach.

Testy wydajnościowe miały na celu ocenienie szybkości działania systemu oraz jego zdolności do jednoczesnego przetwarzania danych z czujników. Dzięki dodaniu znaczników czasu do komunikatów wyświetlonych w konsoli, określono że różnice czasowe między odczytami z różnych czujników wynoszą od 5 milisekund do maksymalnie 1.3 sekundy.

Reakcje systemu na awarie przetestowano w scenariuszach, takich jak odłączanie czujników oraz wyłączenie Arduino. W każdym przypadku odłączenie pojedynczego komponentu nie wpływało na działanie pozostałych elementów systemu.

## V. WNIOSKI

W ramach projektu zrealizowano prototyp systemu monitorowania bezpieczeństwa i warunków środowiskowych w pomieszczeniach. Opracowany algorytm działania systemu, skutecznie integrując dane z czujników i przetwarzając je w sposób pozwalający na efektywne monitorowanie ich parametrów.

Dzięki wykorzystaniu wielowątkowości możliwe jest jednoczesne przetwarzanie danych z różnych źródeł, co gwarantuje stabilność działania. Mechanizm zapewniający synchronizację dostępu do zasobów, eliminuje ryzyko konfliktów między wątkami i zwiększa stabilność działania systemu.

Zaimplementowane programowo mechanizmy zabezpieczające przed błędami, takie jak obsługa wyjątków w przypadku nieprawidłowych odczytów z czujników lub braku danych, zwiększają odporność i zapewniają ciągłość pracy systemu.

Przeprowadzone testy potwierdziły poprawność działania systemu. System spełnił wszystkie wymagania, co potwierdza jego przydatność.

## PROTOTYPE OF AN INTEGRATED SYSTEM FOR MONITORING SECURITY AND INDOOR ENVIRONMENTAL CONDITIONS

The article presents design, construct and implement the software for a prototype of an integrated system for monitoring safety and environmental conditions in indoor spaces. The system enables observation of temperature, humidity, smoke presence and door status. The project utilized technologies such as Python, Flask, MQTT, Socket.IO, C++, HTML, CSS and JavaScript along with DHT11, MQ-2 sensors and CMD14 reed switch. The central unit of the system was built using a Raspberry Pi 4, operating in conjunction with an Arduino Uno R4 WiFi. The developed system provides visualization of monitored parameters through a web interface. Performance and functional tests have confirmed the system's correct operation and its potential for further development.

**Key words**: Arduino, door open sensor, Raspberry Pi, smoke sensor, temperature and humidity sensor

## BIBLIOGRAFIA

[1] Arduino, Dokumentacja Arduino UNO R4 WiFi, https://docs.arduino.cc/hardware/uno-r4-wifi/#features [data dostępu: 10.12.2024]
[2] Botland, Czujnik magnetyczny otwarcia drzwi/okien - kontaktron CMD14, https://botland.com.pl/czujniki-magnetyczne/3104-czujnik-magnetyczny-otwarcia-drzwi-okien-kontaktron-cmd14-srubki-5904422366247.html [data dostępu: 11.12.2024]
[3] CallMeBot, Free API to Send Signal Messaging, https://www.callmebot.com/ [data publikacji: 24.03.2021, data dostępu: 15.12.2024]
[4] D-Robotics, Dokumentacja techniczna czujnika wilgotności i temperatury DHT11, https://botland.com.pl/index.php?controller=attachment&id_attachment=251 [data publikacji: 30.07.2010, data dostępu: 10.12.2024]
[5] Dokumentacja techniczna czujnika półprzewodnikowego MQ-2, https://botland.com.pl/index.php?controller=attachment&id_attachment=94 [data dostępu: 11.12.2024]
[6] Eclipse, Mosquitto an open source MQTT broker, https://mosquitto.org/ [data dostępu: 12.12.2024]
[7] Mozilla Developer Network, CSS: Styling the content, https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics [data ostatniej aktualizacji: 24.12.2024, data dostępu: 30.12.2024]
[8] Mozilla Developer Network, HTML: Creating the content, https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics [data ostatniej aktualizacji: 19.12.2024, data dostępu: 30.12.2024]
[9] Mozilla Developer Network, JavaScript: Adding interactivity, https://developer.mozilla.org/en-US/docs/Learn_web_development/Getting_started/Your_first_website/Adding_interactivity [data ostatniej aktualizacji: 24.12.2024, data dostępu: 30.12.2024]
[10] Mozilla Developer Network, Using media queries, https://developer.mozilla.org/enUS/docs/Web/CSS/CSS_media_queries/Using_media_queries [data ostatniej aktualizacji: 02.01.2025, data dostępu: 07.01.2025]
[11] Pallets, Dokumentacja Flask, https://flask.palletsprojects.com/en/stable/ [data dostępu: 12.12.2024]
[12] Python Software Foundation, Threading — Thread-based parallelism, https://docs.python.org/3/library/threading.html [data dostępu: 11.01.2025]
[13] Raspberry Pi Ltd, Dokumentacja Raspberry Pi 4 B, https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-product-brief.pdf [data publikacji: 04.2024, data dostępu 10.12.2024]
[14] Socket.IO, Dokumentacja Socket.IO, https://socket.io/docs/v4/ [data dostępu: 15.12.2024]