System IIoT do zbierania danych pomiarowych

Todo list

Spis treści

1 Wstęp

Systemy informatyczne znajdują obecnie szerokie zastosowanie w wielu przedsiębiorstwach. Do zadań takich systemów należą przede wszystkim: automatyzacja procesów, ułatwienie zarządzania zasobami (ludzkimi, materialnymi, finansowymi, dokumentacją, wiedzą), gromadzenie informacji na temat prowadzonej działalności na potrzeby raportowania, nadzoru, optymalizacji i rozwoju strategii, a także usprawnienie komunikacji z klientami i kontrahentami. Korzyści wynikające z wdrożenia rozwiązań informatycznych wykorzystuje się również w środowisku przemysłowym. Obok wymienionych zadań (obejmujących szerszy zakres działalności przedsiębiorstwa) podstawową rolą przemysłowych systemów informatycznych jest wspomaganie prowadzenia procesów przemysłowych (produkcji przemysłowej). Istotnym aspektem działania takich systemów jest zbieranie danych pomiarowych. Dane gromadzone przez zainstalowaną w systemie produkcji aparaturę pomiarową, jak i wprowadzane przez ludzi lub inne systemy, determinują działanie układów automatyki, umożliwiają nadzór, diagnostykę, regulację, monitorowanie i wizualizację procesów przemysłowych. Ponadto stosunkowo duża liczba gromadzonych danych poddana zaawansowanemu przetwarzaniu (ang. *big data*) może dostarczyć cennej wiedzy w kontekście prowadzenia biznesu. Wymaganiem charakterystycznym stawianym wobec systemów informatyki przemysłowej jest konieczność działania w ramach ograniczeń nakładanych przez specyficzne środowisko. Należą do nich: determinizm działań i determinizm czasowy, niezawodne funkcjonowanie (odporne na często niesprzyjające warunki), bezpieczeństwo danych i urządzeń [23].

Zbieranie znacznej ilości danych w celu ich dalszego przetwarzania wymienia się wśród głównych zadań Internetu Rzeczy (ang. *Internet of Things*, w skrócie: IoT). Jest to koncepcja, w myśl której przedmioty wyposażone w czujniki i/lub elementy wykonawcze są podłączane do Internetu. Systemy wykorzystujące Internet Rzeczy stały się szczególnie popularne w ostatnich latach dzięki czynnikom ułatwiającym ich realizację: znaczący rozwój systemów komputerowych małej i średniej skali oraz urządzeń mobilnych, upowszechniony dostęp do Internetu, stopniowe wdrażanie protokołu IPv6, duże postępy w dziedzinie eksploracji danych, uczenia maszynowego i inteligencji obliczeniowej, a także dynamiczny rozwój usług opartych na chmurze obliczeniowej [35]. Dziedzina zastosowań IoT jest szeroka, gdyż obejmuje ona urządzenia o wielu różnych przeznaczeniach. Powszechne stało się używanie określenia „inteligenty” (ang. *smart*) wobec tego typu przedmiotów lub środowisk, w których są one używane. W ten sposób powstały koncepcje inteligentnych gadżetów i urządzeń gospodarstwa domowego, rolnictwa, logistyki, systemów opieki zdrowotnej, budynków, a nawet miast. Do obszarów wykorzystania IoT zalicza się także przemysł. Koncepcja ta nosi nazwę Przemysłowego Internetu Rzeczy (ang. *Industrial Internet of Things*, w skrócie IIoT). Głównym przeznaczeniem IIoT jest gromadzenie wiedzy na temat procesu przemysłowego,

a w konsekwencji — zwiększanie jego efektywności [68].

Celem niniejszej pracy jest zaprojektowanie systemu informatycznego wraz z prototypem, którego podstawowym zadaniem jest zbieranie i składowanie danych pomiarowych. Obszarem zastosowań systemu jest środowisko przemysłowe. W realizacji projektu kładzie się nacisk na wykorzystanie technologii IoT oraz chmury obliczeniowej. Źródłem danych w systemie jest aparatura pomiarowa (czujniki) funkcjonująca w ramach systemów automatyki i IoT, inne systemy informatyczne wdrożone w przedsiębiorstwie, a także pracownicy obsługujący proces przemysłowy (mogą oni monitorować jego działanie i wprowadzać wyniki obserwacji do systemu, jak również wyzwalać określone zdarzenia). Przedmiotem pracy jest ogólne opracowanie architektury systemu adresujące przedstawiony problem aplikacyjny. Stąd opisane w kolejnych rozdziałach rozwiązanie teoretyczne abstrahuje od szczegółowych przpyadków użycia, konkretnych urządzeń technicznych czy technologii. Oczekiwaną cechą systemu jest możliwość dostosowania do różnych konfiguracji sprzętu i oprogramowania stosowanych w IoT i dedykowanych rozwiązaniach przemysłowych. Jedną z możliwych implementacji i jednocześnie weryfikacją zaproponowanej koncepcji systemu jest wykonany prototyp. Na jego potrzeby zdefiniowano kilka ściślej sprecyzowanych scenariuszy zastosowania, dokonano wyboru urządzeń i określonych usług chmurowych, jak również bibliotek i narzędzi programistycznych.

W rozdziale 2 przedstawiono przegląd dostępnej literatury związanej z podjętą problematyką. W pierwszej kolejności omówiono z osobna poszczególne zagadnienia, które obejmuje temat pracy, przybliżając w ten sposób jego dziedzinę. Następnie zebrane informacje zestawiono w kontekście realizowanego zadania projektowego, wskazano możliwe problemy i potencjalne rozwiązania wraz z zarysem architektury systemu. Opisano również kilka dostępnych na rynku rozwiązań komercyjnych, które adresują podobny zakres funkcjonalny. W rozdziale 3 sprecyzowano wymagania funkcjonalne i niefunkcjonalne, które ma realizować projektowany system oraz zdefiniowano ogólne przypadki użycia i przykładowe scenariusze wykorzystania systemu implementowane w ramach prototypu. Rozdział 4 zawiera szczegółowy opis architektury zaprojektowanego systemu i jego implementacji w postaci prototypu. Przedstawiono poszczególne komponenty systemu oraz omówiono ich funkcje i sposoby wzajemnej interakcji. W rozdziale 5 zamieszczono informacje dotyczące wdrażania i używania systemu, a także wymagań sprzętowych i programowych. Poruszono również kwestie bezpieczeństwa. Rozdział 6 zawiera opis stosowanej metodyki testowania oraz przedstawia wyniki przeprowadzonych testów z uwzględnieniem napotkanych problemów. W rozdziale 7 zamieszczono podsumowanie i wnioski, które nasunęły się po zakończeniu prac nad projektem. Uzyskany rezultat zestawiono z postawionymi celami i określonymi wymaganiami. Rozważono również kierunki dalszego rozwoju systemu.

2 Analiza tematu

W niniejszym rozdziale przedstawiono rozważania teoretyczne dotyczące projektowanego systemu opierające się na analizie dostępnej literatury. Realizowany temat dotyczy zasadniczo czterech zagadnień: systemów do zbierania danych pomiarowych, systemów informatycznych w przemyśle, IoT i IIoT. Do każdego z nich odnoszą się odpowiednio kolejne podrozdziały przedstawiające podstawowe pojęcia, problemy i rozwiązania w danej dziedzinie. W ostatnim podrozdziale dokonano zestawienia zebranych wiadomości w kontekście celu pracy.

2.1 Systemy informatyczne do zbierania danych pomiarowych

Zgodnie z encyklopedią **system informatyczny** to zespół systemów komputerowych, sieci oraz oprogramowania, których celem zastosowania jest przetwarzanie informacji [72]. **Przetwarzaniem informacji** nazywa się proces akwizycji, utrwalania, udostępniania, organizacji, interpretacji i wizualizacji informacji [30], [69]. Warto w tym miejscu doprecyzować pojęcia **informacji** oraz **danych**. Dane są pewną skodyfikowaną reprezentacją faktów. Natomiast dane, które podlegają intepretacji i nabierają znaczenia stają się informacją [25].

Jednym z możliwych źródeł informacji w systemie informatycznym mogą być **systemy do zbierania danych pomiarowych** (system akwizycji danych, ang. *Data Acquisition Systems*, w skrócie: DAQ). Są to systemy, których rolą jest gromadzenie w formie cyfrowej danych opisujących zjawiska fizyczne. Systemy DAQ tworzą między innymi następujące komponenty o odpowiednich przeznaczeniach:

- czujniki (sensory, ang. *sensors*) — przetwarzanie zjawisk fizycznych na analogowy sygnał elektryczny,
- przetworniki analogowo-cyfrowe — konwersja sygnału analogowego na cyfrowy,
- sprzęt komputerowy i oprogramowanie — przetwarzanie informacji.

Systemy DAQ stanowią jedno z głównych narzędzi stosowanych przez naukowców i inżynierów na potrzeby testów, pomiarów, a także zadań automatyzacji [18].

2.2 Systemy informatyczne w przemyśle

W podrozdziale podjęto zagadnienie systemów informatycznych wdrażanych w środowisku przemysłowym. Omówiono kolejno: pojęcia podstawowe, fundamentalne informacje dotyczące systemów rozproszonych i czasu rzeczywistego oraz zaprezentowano przykładową architekturę systemów przemysłowych.

2.2.1 Podstawowe zagadnienia

Podstawowym zagadnieniem odnoszącym się do środowsika przemysłowego jest pojęcie **procesu przemysłowego** nazywane też **procesem technologicznym**. Jest to ciąg określonych zjawisk fizykochemicznych mających na celu wytworzenie produktu. Dla zwiększenia efektywności, bezpieczeństwa i kontroli nad procesami przemysłowymi praktykuje się wspomaganie ich systemami informatycznymi [23].

W dziedzinie informatyki przeymsłowej wykorzystuje się tzw. **urządzenia aparatury kontrolno-pomiarowej i automatyki** (w skrócie: AKPiA). Stanowią one pomost pomiędzy procesem przemysłowym a systemem informatycznym. Wśród urządzeń AKPiA można wyróżnić następujące grupy [23]:

- układy **inicjatorów** (czujniki, sensory, ang. *sensors*) — rejestrowanie stanu procesu,
- układy **wykonawcze** (ang. *actuators*) — modyfikacja stanu procesu,
- układy **mieszane** (ang. *mixed*) — jednoczesna rejestracja i modyfikacja stanu procesu.

Zestaw urządzeń AKPiA można zamodelować zbiorem informacji, który nazywa się **obiektem przemysłowym** [23].

W ostatnich latach na popularności zyskał pomysł wykorzystania w przemyśle systemów cybernetyczno-fizycznych (ang. *Cyber-Physical Systems*). Koncepcja ta polega na osadzaniu komputerów o niewielkich rozmiarach w zwykłych obiektach fizycznych, a w kontekście przemysłowym — w elementach obsługujących procesy przemysłowe. Mówi się wówczas o tzw. inteligencji wbudowanej. W obrębie CPS spotykane jest określenie analogiczne (nieco bardziej ogólne) do obiektu przemysłowego — tzw. **bliźniaka cyfrowego** (ang. *Digital Twin*). Jest to cyfrowe odwzorowanie cech i atrybutów rzeczywistego obiektu [23], [68].

Omawiając tematykę związaną z systemami przemysłowymi nie sposób pominąć kwestię warunków otoczenia. Środowisko przemysłowe charakteryzuje się zwiększoną uciążliwością. Wymienia się następujące **zaburzenia środowiskowe**, które mogą mieć wpływ na pracę systemu przemysłowego: zaburzenia otoczenia (np. temperatura, wilgotność, czynniki chemiczne), zaburzenia mechaniczne, przewodzone i elektromagnetyczne [23].

2.2.2 Systemy rozproszone czasu rzeczywistego

Systemy informatyczne w przemyśle często posiadają cechy **systemów rozproszonych**. W systemach tych nie występuje centralne urządzenie przetwarzające informacje, lecz składają się one z wielu jednostek przetwarzających o różnym zakresie funkcjonalności. Do zalet takich systemów zalicza się zwiększoną moc obliczeniową względem

systemów nierozproszonych, niezawodność oraz elastyczność (adaptacyjność, rekonfigurowalność) [23], [40]. W obrębie systemów przemysłowych w szczególności stosuje się **rozproszone systemy sterowania**, (ang. *Distributed Control Systems*, w skrócie: DCS). Do zadań takich systemów należą sterowanie i wizualizacja procesu przemysłowego.

Wymaganą cechą procesów przemysłowych jest **determinizm czasowy**. Postawienie wymagania determinizmu czasowego względem systemu przemysłowego oznacza, że jego odpowiedź na zdarzenia musi zachodzić zawsze w zdefiniowanym, skończonym czasie [23].

Systemy charakteryzujące się m. in. determinizmem czasowym nazywa się **systemami czasu rzeczywistego** (ang. *Real Time Systems*, w skrócie: RTS). Systemy posiadające cechy zarówno systemów rozproszonych, jak i systemów czasu rzeczywistego nazywa się **systemami rozproszonymi czasu rzeczywistego**. Zagadnieniem kluczowym w kontekście takich systemów jest **komunikacja**, którą umożliwiają sieci i protokoły. Na potrzeby przemysłu opracowano rozwiązania odpowiadające również na potrzebę działania w ścisłych ograniczeniach czasowych. Do przykładowych sieci należą przede wszystkim sieci polowe (ang. *fieldbus*) i Ethernet czasu rzeczywistego (Ethernet przemysłowy, ang. *Real-Time Ethernet*, w skrócie: RTE). Sieci RTE posiadają obecnie wiele technicznych realizacji. Krokiem w kierunku standaryzacji jest koncept sieci TSN (ang. *Time-Sensitive Networks*) rozwijany od ostatniej dekady w ramach specyfikacji sieci Ethernet należących do grupy IEEE 802. Dedykowane dla sieci przemysłowych protokoły to m. in. Profibus/Profinet, Modbus, EtherCAT, Ethernet Powerlink i inne [23], [40], [38], [21].

2.2.3 Architektura systemów informatycznych w przemyśle

Dziedzina funkcjonalna systemów informatycznych w przemyśle jest stosunkowo obszerna. Są one w stanie wspomagać zarówno przebieg procesu przemysłowego, jak i kompleksowo usprawniać procesy funkcjonujące na wyższych poziomach przedsiębiorstwa — np. te związane z zarządzaniem zasobami. W literaturze spotkać można różne modele przedstawiające zakres działania przemysłowych systemów informatycznych. Swego czasu popularny był tzw. model piramidowy (warstwowy, hierarchiczny) przedstawiony na rys. 1. Został on wyparty m. in. przez model RAMI 4.0, który opisano w rozdziale 2.4 w kontekście IIoT. W modelu piramidowym wyróżnia się następujące warstwy: [41], [23], [34], [35]:

- Warstwa produkcyjna — tworzą ją urządzenia automatyki przemysłowej oraz aparatury kontrolno-pomiarowej wraz z łączącymi je sieciami komputerowymi,
- Warstwa operacyjna — należą do niej systemy realizacji produkcji (ang. *Manufacturing Execution System*, w skrócie: MES) oraz systemy zbierania danych

i sprawowania kontroli SCADA/HMI (ang. *Supervisory Control and Data Acquisition, Human Machine Interface*),

- Warstwa biznesowa — należą do niej wysokopoziomowe systemy planowania zasobów przedsiębiorstwa ERP (ang. *Enterprise Resource Planning*), systemy analizy i produkcji SAP (ang. *Systems Analytics and Product*), systemy zarządzania relacjami z klientami CRM (ang. *Customer Relationship Management*) oraz wiele innych.

Rysunek 1: Model piramidowy informatycznych systemów przemysłowych.

2.3 Internet Rzeczy

W podrozdziale przedstawiono przegląd definicji, zastosowań, a także elementów tworzących IoT. Podrozdział zawiera również odniesienie do ważnego w kontekście omawianego zagadnienia przetwarzania w chmurze.

2.3.1 Definicje i zastosowania Internetu Rzeczy

Internet Rzeczy (ang. *Internet of Things*, w skrócie: IoT) posiada różne definicje. Jedną z podstawowych jest: sieć przedmiotów z wbudowanymi czujnikami, które są podłączone do Internetu [35]. Na nieco wyższym poziomie definiuje się IoT jako globalną infrastrukturę udostępniającą zaawansowane usługi poprzez połączenie przedmiotów z wykorzystaniem technologii informacyjnych. [35].

Głównym zastosowaniem Internetu Rzeczy jest monitorowanie świata rzeczywistego oraz wchodzenie z nim w interakcję, co może usprawnić działalność człowieka na wielu płaszczyznach. Wśród praktycznych zastosowań IoT wymienia się między innymi: sieci czujników i pomiary rozproszone — w szczególności z wykorzystaniem technologii bezprzewodowej sieci czujników (ang. *Wireless Sensor Networks*, w skrócie: WSN) lub też

bezprzewodowej sieci czujników i urządzeń wykonawczych (ang. *Wireless Sensor and Actuator Networks*, w skrócie: WSAN), urządzenia (gadżety) ubieralne, inteligentne budynki, miasta, logistyka, przemysł, a także marketing [20], [62]. Możliwości, jakie niesie IoT dla przemysłu zostały w sposób szczególny rozpatrzone w rozdziale 2.4.

2.3.2 „Rzeczy" w IoT

Wśród przedmiotów tworzących IoT wyróżnia się dwie kategorie: **czujniki oraz urządzenia wykonawcze**. Są to odpowiedniki aparatury kontrolno-pomiarowej wykorzystywanej w przemyśle (patrz: rozdział 2.2). Czujniki umożliwiają obserwację świata rzeczywistego i jej zapis w postaci cyfrowej. Urządzenia wykonawcze są w stanie zmienić stan układu rzeczywistego na podstawie otrzymanej informacji [61]. Przykładowe czujniki to: przyciski, przełączniki, czujniki ruchu, czujniki gazów, czujniki wibracji, czujniki temperatury, wilgoci, nacisku, ultradźwięków i wielu innych wielkości fizycznych. Przykłady urządzeń wykonawczych to: silniki, siłowniki liniowe, przekaźniki, zawory.

Ważnym elementem infrastruktury IoT są systemy wbudowane typu SoC (ang. *System on a Chip*). Są to układy elektroniczne, których poszczególne komponenty realizujące wymagane funkcjonalności są zintegrowane w ramach jednego, kompletnego układu scalonego. System typu SoC mogą zawierać m. in.: procesory lub mikrokontrolery, przetworniki analogowo-cyfrowe i cyfrowo-analogowe, pamięci, procesory graficzne, interfejsy komunikacyjne [35], [70]. Obok rozwiązań dedykowanych dużą popularnością cieszą się następujące platformy uniwersalne [35], [59], [71], [44], [8], [57], [19]:

- Arduino — Różne modele płytek rozwojowych wykorzystujące m. in. systemy SoC producenta Atmel oparte o mikrokontrolery AVR. Wśród zestawów płytek dostępne są moduły rozszerzeń posiadające przewodowe i bezprzewodowe interfejsy sieciowe;
- ESP — Systemy typu SoC producenta Espressif Systems zawierające m. in. wbudowane bezprzewodowe interfejsy sieciowe;
- Raspberry Pi — Zaawansowane płytki tworzące komputery jednopłytkowe (ang. *single-board computers*) wykorzystujące m. in. procesory ARM Cortex. Można na nich uruchomić system operacyjny (również z interfejsem graficznym). Posiadają liczne interfejsy komunikacyjne (w tym sieciowe);
- STM32 — Rodzina 32-bitowych mikrokontrolerów opartych na procesorach ARM Cortex. Platformę cechuje stosunkowo duża moc obliczeniowa, możliwość działania w trybie niskiego poboru energii oraz ograniczeniach czasowych;
- BeagleBone — Komputery jednopłytkowe podobne do RaspberryPi oparte na procesorach ARM. Można na nich uruchomić system operacyjny. Cechą wyróż-

niającą jest możliwość funkcjonowania w czasie rzeczywistym z wykorzystaniem programowalnej jednostki czasu rzeczywistego (ang. *Programmable Real-Time Unit*, w skrócie PRU).

Wymienione platformy uniwersalne mogą stanowić pomost pomiędzy główną siecią w systemie IoT (np. Ethernet lub WiFi) a innymi urządzeniami peryferyjnymi komunikującymi się za pośrednictwem uniwersalnych złącz pinowych GPIO albo interfejsów szeregowych, takich jak: SPI, I2C, RS232, USB [35].

2.3.3 „Internet" w IoT

Wymiana informacji w systemach IoT może odbywać się w sposób przewodowy lub bezprzewodowy. Stosowane modele komunikacji sieciowej bazują na standardowym modelu OSI bądź internetowym stosie TCP/IP. W warstwie fizycznej i łącza danych sieci IoT wykorzystuje się: Ethernet, WiFi, Bluetooth, sieci komórkowe, ZigBee, Z-Wave, NFC. W warstwie sieciowej stosuje się protokoły: IPv4, IPv6 oraz zoptymalizowaną pod IoT wersję IPv6 — 6LoWPAN. Protokoły warstwy transportowej to m. in. TCP i UDP. Warstwy sesji, prezentacji i aplikacji implementują powszechne używane internetowe protokoły, np. HTTP, RTP, SMTP. Wśród nich opracowano również protokoły dedykowane dla IoT — są to m. in. MQTT i CoAP [20], [35], [61].

W sieciach IoT rozróżnia się trzy następujące modele komunikacyjne [35]:

- Urządzenie – Urządzenie (ang. *Machine to Machine, Device to Device*, w skrócie: M2M) — Urządzenia komunikują się pomiędzy sobą bez potrzeby translacji czy wykonywania skomplikowanego przetwarzania danych.
- Urządzenie – Brama (ang. *Device to Gateway*) — Występuje, gdy zachodzi potrzeba translacji informacji wymienianej pomiędzy różnymi sieciami. Translacją zajmuje brama (ang. *gateway*), którą tworzy dedykowane oprogramowanie i/lub urządzenie;
- Urządzenie – Chmura (ang. *Device to Cloud*) — Występuje, gdy zachodzi potrzeba zaawansowanego przetwarzania danych z wykorzystaniem chmury (statystyka, eksploracja danych, big data, składowanie, itp.).

2.3.4 Usługi chmurowe

Przetwarzanie w chmurze (ang. *cloud computing*) pozwala na migrację części lub całości infrastruktury informatycznej do tzw. chmury. Wśród popularnych rozwiązań znajdują się: Microsoft Azure, Amazon AWS, Google Cloud, czy IBM Cloud. Usługi chmurowe udostępnianie przez wymienionych i wielu innych dostawców za pośrednictwem Internetu noszą miano **chmury publicznej**. Są one dostępne na życzenie

i posiadają elastyczny system rozliczeń, w którym koszta są zależne od faktycznie zużytych zasobów informatycznych (cykle procesorów, pamięć, wykorzystanie łącza, liczba zapytań). Do zalet takiego rozwiązania należą: łatwa skalowalność, konfigurowalność i duży wybór usług ułatwiających utrzymanie infrastruktury informatycznej. Spośród wad należy wymienić zależność od firm trzecich (dostawców chmury) oraz brak pełnej kontroli nad posiadaną infrastrukturą. Niektóre firmy rozwijają też podobne, lecz dostępne wyłącznie lokalnie, rozbudowane platformy infrastruktury informatycznej, zwane **chmurą prywatną**. Tego typu rozwiązania są kosztowne, lecz dają pełną kontrolę nad posiadanymi zasobami [61].

W ramach swoich usług dostawcy chmury udostępniają m.in.: przetwarzanie danych, analitykę, eksplorację danych, uczenie maszynowe, konteneryzację, relacyjne i nierelacyjne bazy danych, integrację aplikacji, hosting aplikacji i stron internetowych, maszyny wirtualne, narzędzia programistyczne, zarządzanie użytkownikami, a ponadto platformy do zarządzania systemami IoT [15], [6].

Zarządzanie dużą ilością danych jest podstawowym zadaniem systemów IoT. Usługi chmurowe umożliwiają zaawansowane przetwarzanie danych oferując jednocześnie skalowalność, zdalny dostęp, opłaty zależne od zapotrzebowania i gotowe rozwiązania programistyczne skracające czas rozwoju systemów [35], [42], [9], [43].

2.4 Internet rzeczy w przemyśle

Przemysłowym Internetem Rzeczy (ang. *Industrial Internet of Things*, w skrócie: IIoT) nazywa się rozwiązania IoT znajdujące zastosowanie w przemyśle. IIoT ma umożliwić lepsze zrozumienie procesów przemysłowych, a w związku z tym zapewnić większą wydajność i zrównoważenie produkcji. Innymi słowy, IIoT ma za zadanie ułatwić integrację **technologii operacyjnych** (ang. *Operation Technologies*, w skrócie: OT) oraz **technologii informatycznych** (ang. *Information Technologies*, w skrócie: IT) [68]. Technologie operacyjne to systemy nadzorujące przebieg procesów przemysłowych, zaś technologie informatyczne to systemy, których rolą jest jest gromadzenie i przetwarzanie informacji wartościowych z perspektywy przedsiębiorstwa [11].

Ze względu na konieczność funkcjonowania w specyficznym środowisku przemysłowym IIoT posiada pewne cechy odróżniające od standardowego (konsumenckiego) IoT. Wśród nich należy wymienić [68]:

- Rozwój o charakterze ewolucyjnym, nie rewolucyjnym — Wdrażając systemy IIoT należy się liczyć z dużą bezwładnością przemysłowego zaplecza technicznego [23];
- Infrastruktura komunikacyjna — Infrastruktura IoT jest bardziej elastyczna, podczas gdy infrastruktura IIoT wymaga dostosowania do bardziej ustrukturyzowanych modeli komunikacyjnych;

- Ograniczenia — W otoczeniu przemysłowym charakterystyczne są wysokie wymagania dotyczące ograniczeń czasowych (determinizmu), niezawodności, bezpieczeństwa danych oraz funkcjonowania w niesprzyjających warunkach środowiskowych.

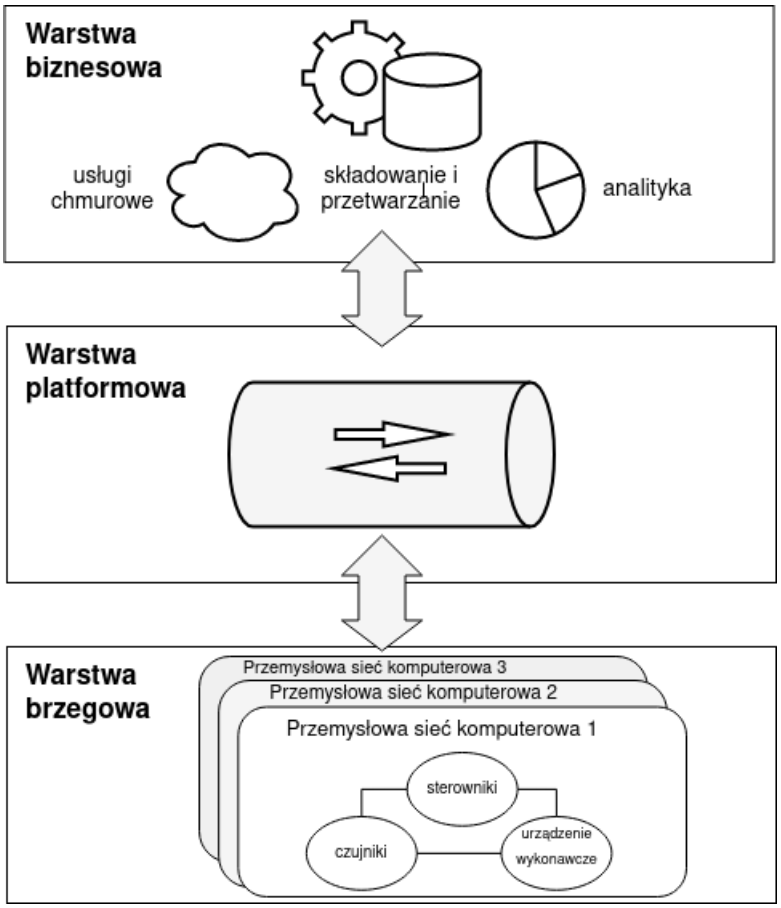
Szczególną uwagę należy poświęcić stwierdzeniu, że **intencją wykorzystania IIoT nie jest zastąpienie tradycyjnych systemów automatyzacji procesów przemysłowych**. Na tej płaszczyźnie istnieją bowiem dedykowane rozwiązania adresujące problem ograniczeń spotykanych w automatyce przemysłowej. Właściwym celem wdrażania systemów IIoT jest zwiększanie wiedzy na temat procesów przemysłowych, co w efekcie pozwala na ulepszenie ich wydajności [68].

Ze względu na tolerancję ograniczeń w systemie (m. in. opóźnień czasowych) można wyróżnić trzy poziomy jakości usług komunikacyjnych [61]:

- Dostarczanie z wykorzystaniem najlepszych możliwości (ang. *best effort*) — Komunikacja nie spełnia ścisłych ograniczeń czasowych, wykorzystywana jest maksymalna dostępna przepustowość. Jest to poziom jakości charakterystyczny dla usług internetowych;
- Dostarczanie w czasie gwarantowanym — Komunikacja spełnia pewne ograniczenia czasowe, których niedotrzymanie skutkuje spadkiem korzyści płynących z wykorzystawanych usług;
- Dostarczanie w czasie deterministycznym — Komunikacja powinna spełniać ścisłe ograniczenia czasowe, których niedotrzymanie jest równoznaczne z awarią systemu.

Powszechna infrastruktura tworząca Internet, jak i usługi w nim dostępne, są z reguły niedeterministyczne pod względem czasowym. Systemy IoT czy IIoT mogą więc funkcjonować zgodnie paradygmatem *best effort*. Aczkolwiek niekoniecznie należy odrzucać możliwość współpracy systemu z układem automatyki na poziomie lokalnym, zanim dane trafią do Internetu [36], [75].

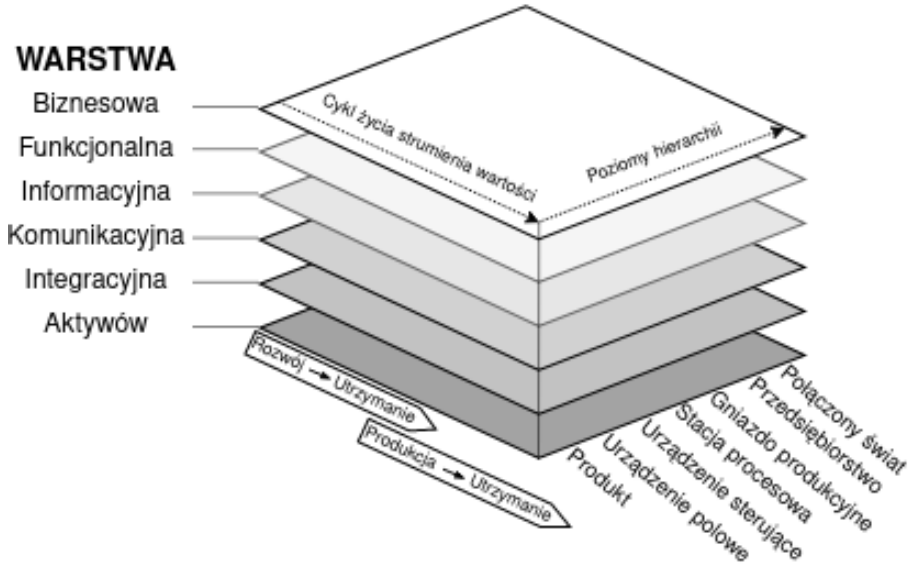
Dla IIoT zostały opracowane pewne wzorce architektoniczne, które stanowią szablon dla implementacji tego typu systemów. Obecnie popularny jest wzorzec trójwarstwowy, dla którego wyróżnia się warstwę brzegową (ang. *edge*), warstwę platformową (ang. *platform*) i warstwę biznesową (ang. *enterprise*). Model ten przedstawiono na rys. 2. Do warstwy brzegowej należą typowe urządzenia systemów automatyki przemysłowej, m.in. czujniki i urządzenia wykonawcze oraz sterowniki (np. klasy PLC) oraz łączące je sieci komputerowe. Warstwa platformowa umożliwia zarządzanie systemem IoT, akwizycję i agregację danych oraz ich trasowanie na wejścia aplikacji i usług warstwy biznesowej, które korzystają ze zgromadzonych danych w celu wspomagania analityki, monitorowania, zarządzania, archiwizacji, itd. [68], [51].



Rysunek 2: Trójwarstwowy wzorzec architektury systemów IIoT

W wyniku połączenia idei wywodzących się z systemów IoT, IIoT i CPS powstała koncepcja **Przemysłu 4.0** (niem. *Industrie 4.0*) reprezentująca tzw. czwartą rewolucję przemysłową. Domenę Przemysłu 4.0 opisuje na wysokim poziomie model architektury RAMI 4.0 (ang. *Reference Architectural Model Industrie 4.0*), któremu ustępuje obecnie przestarzały model piramidowy opisany w rozdziale 2.2.3. Architektura RAMI 4.0 została przedstawiona na rys. 3. Jest to model trójwymiarowy opisujący trzy podstawowe aspekty Przemysłu 4.0. Na osi o nazwie „Poziomy hierarchii“ umieszczono poziomy funkcjonowania jednostek produkcyjnych. Podział ten wywodzi się ze standardów IEC 62264 i ISA-95 (modelu piramidowego). Oś o nazwie „Cykl życia strumienia wartości“ reprezentuje cykl życia produktów i innych aktywów. W pierwszej fazie następuje projektowanie, rozwój i prototypownie (powstaje tzw. „typ produktu“), a w drugiej fazie rozpoczyna się produkcja (powstają tzw. „instancje produktu“). Pionowa oś przedstawia sześć warstw funkcjonowania systemów informatycznych w ramach Przemysłu 4.0. Każda z nich udostępnia usługi dla warstwy wyższej i używa usług warstwy niższej. Warstwa aktywów (zasobów) reprezentuje fizyczne urządzenia. Warstwa integracji jest pomostem między światem fizycznym a światem informacji. Warstwa komunikacji zapewnia jednolity format danych i sposób komunikacji. Warstwa informacji odpowiada za przetwarzanie danych. Warstwa funkcjonalna dostarcza środowiska do

realizacji modeli implementowanych w ramach warstwy biznesowej [68], [22], [27], [34].



Rysunek 3: Model architektury RAMI 4.0

2.5 Ujęcie tematu w świetle zgromadzonych informacji

Głównym zagadnieniem rozważanym w ramach projektu jest zbieranie danych pomiarowych w środowisku przemysłowym. Problem ten adresują technologie IIoT oraz usługi chmurowe. Umożliwiają one gromadzenie i przetwarzanie danych w systemie rozproszonym składającym się z szeroko pojętej aparatury kontrolno-pomiarowej (urządzeń cybernetyczno-fizycznych). W niniejszej pracy podjęto zadanie **zaprojektowania architektury** opisanego systemu oraz **realizacji jego prototypu**.

Sporządzenie koncepcji architektury systemu sprowadza się do wypracowania **ogólnego rozwiązania** dla klasy zadań, jaką jest zbieranie danych w środowisku przemysłowym. Zaproponowany model będzie więc abstrahować od konkretnych narzędzi, technologii, urządzeń czy szczegółowych przypadków użycia. Przykładową implementację modelu ma stanowić wykonany prototyp, na poziomie którego dobór technologii i aparatury będzie skonkretyzowany na podstawie analizy i porównań dostępnych narzędzi i usług informatycznych.

Modelem referencyjnym dla projektowanego systemu jest zaprezentowana na rys. 2 **trójwarstwowa architektura systemów IIoT**. Dla warstwy brzegowej przyjmuje się, że w systemie jest zapewniony dostęp do dwóch rodzajów urządzeń: przemysłowe urządzenia lub zespoły urządzeń aparatury kontrolno-pomiarowej pracujące w czasie rzeczywistym w ramach sieci przemysłowej oraz typowe dla IoT bezprzewodowe sieci

WSN/WSAN (patrz: rozdział 2.3) lub inne zespoły urządzeń zawierające czujniki komunikujące się z wykorzystaniem interfejsów i protokołów charakterystycznych dla IoT [14]. Implementację warstwy platformowej realizują rozwiązania oferowane przez dostawców chmury. Do warstwy biznesowej mogą należeć istniejące aplikacje wspomagające działalność przedsiębiorstwa (m. in. systemy klasy ERP) oraz mogące przejmować ich funkcje usługi chmurowe. Pewną lukę, element brakujący w proponowanym modelu stanowi konieczność dopasowania do różnych protokołów sieci przemysłowych i IoT. Istnieje również potrzeba integracji warstwy brzegowej i platformowej zaproponowanego modelu. Są to krytyczne zadania przypisywane systemom IIoT. Do rozwiązania przytoczonego problemu stosuje się tzw. **bramy IoT** (ang. *IoT gateways*). Są to zwykle urządzenia wraz z oprogramowaniem będące w stanie integrować zarówno protokoły warstwy fizycznej i łącza danych, jak i dokonywać translacji protokołów wyższych warstw (tzw. bramy semantyczne) wraz z możliwością udostępniania danych za pośrednictwem Internetu [39]. Bramy mogą realizować ponadto wstępne przetwarzanie danych odciążając w ten sposób usługi wyższych warstw. Jest to tzw. koncepcja **przetwarzania brzegowego** (ang. *edge computing*) [12]. W ogólniejszym rozumieniu bramy IoT mają stanowić pomost pomiędzy technologią operacyjną a technologią informatyczną [14], [68].

Otrzymano zatem podstawowy zarys projektowanego systemu. Składa się on z urządzeń pomiarowych udostępniających dane w ramach sieci przemysłowych oraz IoT. Zbieraniem danych zajmuje się brama (bramy) IoT, która przekazuje dane w spójnym, jednolitym formacie do usług chmurowych i innych aplikacji za pośrednictwem Internetu. Te z kolei mogą realizować zaawansowane przetwarzanie, składowanie, udostępnianie i wizualizację danych.

Wśród istniejących, komercyjnych rozwiązań o podobnej dziedzinie funkcjonalnej znaleziono następujące przykłady:

- Sensemetrics — Rozwiązanie amerykańskiej firmy Industrial IoT Solutions. Jest to system do akwizycji i zarządzania danymi [63];
- Nazca 4.0 — System rozwijany przez polską firmę APA Group zorientowany na Przemysł 4.0. Wśród użytych technologii producent wymienia IoT, inteligentne czujniki, przetwarzanie w chmurze, integrację systemów [50];
- IXON — Rozwiązanie IIoT holenderskiej firmy o tej samej nazwie. System składa się z platformy internetowej umożliwiającej nadzór, analitykę i wizualizację danych oraz dedykowanego routera/bramy IoT [33].

3 Specyfikacja wymagań

W niniejszym rozdziale przedstawiono szczegółową specyfikację wymagań stawianych wobec projektowanego systemu i prototypu. Dokumentacja składa się z wymagań funkcjonalnych i niefunkcjonalnych, przypadków użycia oraz scenariuszy biznesowych implementowanych w ramach prototypu.

3.1 Wymagania funkcjonalne

Do założonych ogólnych funkcjonalności systemu należą: zbieranie, przetwarzanie, składowanie, udostępnianie i wizualizacja danych pomiarowych, które omówiono w kolejnych podrozdziałach.

3.1.1 Zbieranie danych pomiarowych

Zbieranie (gromadzenie) danych pomiarowych to podstawowe zadanie projektowanego systemu. Pod pojęciem "dane pomiarowe" rozumie się opis obiektów i zjawisk fizycznych zapisany w postaci cyfrowej, który jest zwykle przyporządkowany do określonego punktu lub przedziału w czasie. Ciąg takich opisów nazywa się szeregiem czasowym [73]. Do możliwych źródeł danych pomiarowych należą: aparatura pomiarowa (czujniki, sensory), pamięć masowa, użytkownicy wchodzący w interakcję z systemem oraz inne systemy informatyczne. Projektowany system powinien **udostępniać interfejs wejściowy** umożliwiający zbieranie i przekazywanie danych pochodzących z wymienionych źródeł. Należy ponadto uwzględnić fakt, że urządzenia dostarczające dane mogą komunikować się na różne sposoby. Stąd konieczne jest, aby system był przystosowany do współpracy z popularnymi interfejsami oraz protokołami komunikacyjnymi.

3.1.2 Przetwarzanie danych

Realizowany system ma za zadanie umożliwić **przetwarzanie zgromadzonych danych pomiarowych** — zarówno na poziomie sprzętowym, jak i programowym. Wśród możliwych przykładów przetwarzania danych w systemie można wymienić: agregację danych (scalanie wyników, wyznaczanie statystyk), obliczanie kluczowych wskaźników efektywności prowadzonej działalności (ang. *Key Performance Indicators*, w skrócie: KPI), uczenie maszynowe (zadania klasyfikacji, prognozowanie, optymalizacja procesów), automatyzację zadań.

3.1.3 Składowanie danych

W ramach dziedziny biznesowej istnieje zwykle potrzeba przyszłego wykorzystania zebranych danych oraz wyników ich przetwarzania. Trwały zapis danych dokumentuje

procesy zachodzące w przedsiębiorstwie i udogadnia sprawowanie kontroli; pozwala na gromadzenie większej ich ilości i w konsekwencji pogłębioną, bardziej dokładną analizę. Zatem oczekuje się, aby projektowany system **realizował funkcjonalność składowania (archiwizacji) danych pomiarowych w bazach danych** i umożliwiał dostęp do nich w przyszłości.

3.1.4 Wizualizacja danych

Aby gromadzenie i przetwarzanie danych pomiarowych mogło przynieść jakiekolwiek korzyści dla przedsiębiorstwa niezbędne jest, aby dane otrzymywane na wyjściu realizowanego systemu posiadały reprezentację interpretowalną, łatwo przyswajalną dla człowieka. Możliwe jest wówczas dokonywanie dalszej analizy i zadań obserwacji, monitorowania czy zarządzania. System **powinien zatem udostępniać przyjazny dla człowieka interfejs (graficzny)**, w którym dane przedstawione są za pomocą tabel, wykresów, diagramów, plików o przejrzystej strukturze.

3.1.5 Udostępnianie danych

Z efektów działania projektowanego systemu bezpośrednio mogą korzystać nie tylko ludzie, lecz także inne systemy. Wymaga się więc, aby realizowany system **mógł udostępniać zgromadzone dane i wyniki ich przetwarzania na potrzeby zewnętrznych systemów (klientów)**, które realizują kolejne etapy przetwarzania danych.

3.2 Wymagania niefunkcjonalne

Rozdział zawiera opis następujących wymagań niefunkcjonalnych: wykorzystanie technologii IoT i przetwarzania w chmurze, dostosowanie do środowiska przemysłowego, ogólność rozwiązania, skalowalność i bezpieczeństwo danych. Szczegółowe rozważania dotyczące wymienionych założeń przedstawiono w ramach kolejnych podrozdziałów.

3.2.1 Wykorzystanie technologii IoT i przetwarzania w chmurze

Wykorzystanie Internetu Rzeczy to podstawowe założenie, które zostało ujęte w temacie niniejszej pracy i zaznaczone przy określeniu celu projektu. Do najważniejszych zastosowań IoT należy zbieranie dużej ilości danych z wielu różnych urządzeń. Użycie rozwiązań z tej dziedziny jest więc wyborem uzasadnionym. Zadaniem krytycznym w systemach IoT jest przetwarzanie zebranych danych. Problem ten podejmują przede wszystkim oferowane za pośrednictwem Internetu usługi chmurowe. Tak więc Internet Rzeczy i przetwarzanie w chmurze w stopniu znaczącym odpowiadają na większość wymagań funkcjonalnych przedstawionych w poprzednim rozdziale.

3.2.2 Dostosowanie do środowiska przemysłowego

Drugim założeniem podstawowym obejmującym zakres niniejszej pracy jest **przystosowanie projektowanego systemu do zastosowania w przemyśle.** W wielu przedsiębiorstwach istnieją już sprawdzone systemy automatyki wyposażone w aparaturę pomiarową. Jako że rozwój technologiczny w przemyśle ma z reguły charakter stopniowy, wymaga się, aby projektowany system **integrował się z obecnie funkcjonującymi systemami**, co oznacza konieczność posiadania odpowiednich interfejsów komunikacyjnych i implementowania określonych protokołów charakterystycznych dla rozproszonych systemów przemysłowych.

Jak już wspomniano w rozdziale 2, systemy kontrolujące procesy przemysłowe muszą zwykle działać w ramach twardych ograniczeń czasowych. Natomiast wykorzystanie technologii internetowych gwarantuje działanie na poziomie *best effort*. Celem ich stosowania w przemyśle nie jest jednak zastąpienie istniejących systemów automatyki, lecz ich integracja z systemami warstwy IT, które niekoniecznie operują w sztywnych ramach czasowych. Aczkolwiek, żeby zapewnić szerszą współpracę projektowanego systemu z istniejącymi rozwiązaniami przemysłowymi, należy **uwzględnić możliwość jego funkcjonowania w czasie rzeczywistym na poziomie lokalnym**, tzn. w ramach warstwy brzegowej (patrz: rozdział 2, rys. 2).

W środowisku przemysłowym oczekuje się zwiększonej niezawodności systemów zarówno na poziomie sprzętowym, jak i programowym. Choć w przypadku IIoT nie jest to zwykle wymaganie o stopniu tak krytycznym, jak w standardowych systemach automatyki, to w projekcie systemu należy uwzględnić w zakresie podstawowym zagadnienia **redundancji, diagnostyki i możliwości funkcjonowania w niesprzyjających warunkach środowiskowych**.

3.2.3 Ogólność rozwiązania

Pożądaną cechą projektowanego systemu jest jego uniwersalność. Wypracowane rozwiązanie powinno być możliwie niezależne od specyficznych urządzeń, narzędzi technicznych, oprogramowania, czy dostawców chmury. Takie podejście pozwala na implementację różnych przypadków biznesowych oraz dostosowanie do istniejącej w przedsiębiorstwach infrastruktury sprzętu i oprogramowania z uwzględnieniem indywidualnych preferencji, wymagań i posiadanych zasobów.

Integrację różnych systemów informatycznych na odpowiednich poziomach abstrakcji zapewniają sieci komputerowe, interfejsy i protokoły. Stąd **kluczowym aspektem projektowanego systemu — co już kilkukrotnie zostało podkreślone — jest dostarczenie wsparcia dla popularnych interfejsów, protokołów i modeli wymiany informacji stosowanych w IoT, przemyśle oraz aplikacjach internetowych i usługach chmurowych**.

Wraz z implementacją popularnych protokołów wymagane jest ponadto **zapewnienie jednolitego formatu danych na poziomie warstwy aplikacji**, który ma zagwarantować przenośność, łatwą wymianę danych pomiędzy współpracującymi systemami. Formaty te powinny być uprzednio specyfikowane, a projektowany system powinien być do nich zaadaptowany — analogicznie, jak w przypadku protokołów.

Obok przystosowania do różnych sposobów komunikacji ważnym kryterium uniwersalności systemu jest elastyczność oprogramowania. W systemie, w którym zbierane są dane z wielu różnych urządzeń, wymagania dotyczące ich przetwarzania podlegają częstej zmianie i ewolucji. **Oczekuje się zatem, że system będzie umożliwiał łatwą modyfikację i rozbudowę realizowanej logiki, jak również wymianę komponentów**. Ponadto należy uwzględnić fakt, że w zarządzaniu systemem przemysłowym niekoniecznie muszą uczestniczyć zawodowi programiści, z czym wiąże się potrzeba, aby **model programistyczny systemu był zrozumiały także dla specjalistów innych dziedzin techniki i analityków**.

3.2.4 Skalowalność

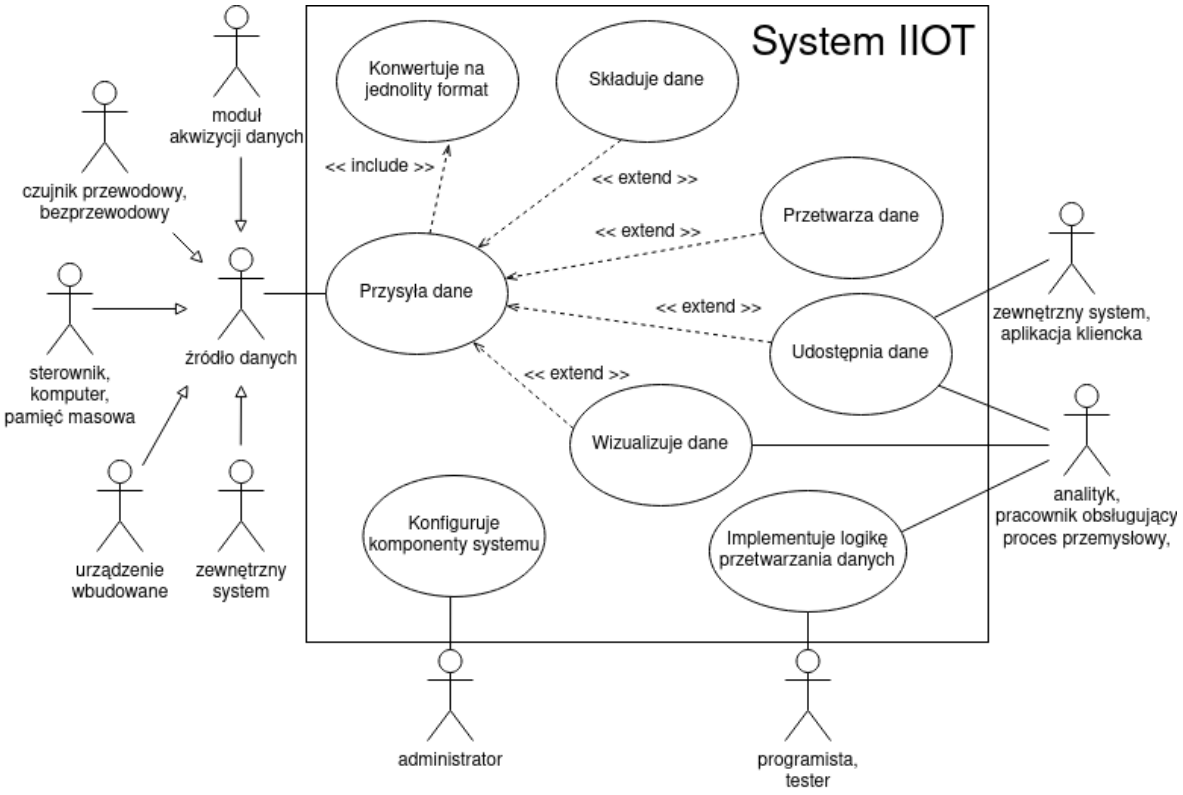
Projekt powinien przewidywać możliwość zmiany skali infrastruktury przedsiębiorstwa, w ramach której ma on być wdrożony. Wymaga się, aby system mógł być w stosunkowo prosty, szybki i tani sposób zaadaptowany do nowych warunków pracy, które są rezultatem zmieniającej się liczby obsługiwanych urządzeń i ilości koniecznych do przetworzenia danych. Dostosowanie systemu sprowadza się w głównej mierze do przydzielenia odpowiednich zasobów sprzętowych i programowych.

3.2.5 Bezpieczeństwo danych

Wymaganiem stawianym wobec większości systemów informatycznych jest bezpieczeństwo danych. Również w środowisku przemysłowym przetwarzane dane są zwykle traktowane jako poufne, zaś ograniczenie dostępu do systemów sterowania ma charakter krytyczny, gdyż mają one realny wpływ na bezpieczeństwo zasobów przedsiębiorstwa — w tym także i ludzi. W ramach projektu należy rozważyć w zakresie podstawowym kwestie bezpieczeństwa w aspekcie autoryzacji dostępu do systemu (w świecie rzeczywistym oraz cyfrowym), a także zapewnienia poufności z wykorzystaniem szyfrowania i metod ochrony przed kradzieżą.

3.3 Przypadki użycia

Ogólne ujęcie funkcjonalności projektowanego systemu z perspektywy użytkowników, zewnętrznych urządzeń i systemów modeluje diagram przypadków użycia przedstawiony na rys. 4. Na jego potrzeby zdefiniowano abstrakcyjne źródło danych, które



Rysunek 4: Diagram przypadków użycia systemu.

mogą stanowić zróżnicowane urządzenia pomiarowe, komputery, jak również inne systemy. Źródło przesyła dane do systemu, który dokonuje ich konwersji na pożądany format i umożliwia dalsze składowanie, przetwarzanie, udostępnianie i wizualizację. Dane są udostępniane na potrzeby innych systemów informatycznych (aplikacji klienckich). Są one także bezpośrednio prezentowane pracownikom obsługi procesu przemysłowego i analitykom w ramach wizualizacji. Wymienieni pracownicy we współpracy z programistami i testerami uczestniczą w procesie implementacji, modyfikacji i rozbudowy logiki przetwarzania danych. Administrator systemu zajmuje się jego konfiguracją i utrzymaniem poszczególnych jego komponentów. Praca ta polega na instalacji urządzeń warstwy sprzętowej, zarządzaniu użytkownikami i dostępem do systemu, integracji poszczególnych jego komponentów, ustawieniu parametrów połączeń sieciowych i protokołów, monitorowaniu i usuwaniu awarii.

3.4 Scenariusze wykorzystania systemu

Na potrzeby walidacji funkcjonalności realizowanego projektu dokonano wyboru przykładowych scenariuszy biznesowych, które mają zostać zaimplementowane przez prototyp. Przyjęto założenie, że w środowisku docelowym w obrębie lokalnej sieci są zainstalowane urządzenia pomiarowe, które komunikują się odpowiednio z wykorzystaniem protokołów MQTT, CoAP oraz Modbus/TCP. Pierwsze dwa są charakterystyczne

dla IoT, natomiast ostatni jest popularny wśród rozwiązań przemysłowych. System ma za zadanie integrować się z lokalną siecią i wymienionymi protokołami, z użyciem których gromadzone będą dane. Szczegółowy opis przyjętych scenariuszy biznesowych przedstawiono w kolejnych podrozdziałach.

3.4.1 Monitorowanie parametrów powietrza

W określonych pomieszczeniach zakładu produkcyjnego zainstalowano sieć bezprzewodowych czujników temperatury i wilgoci WiFi, która jest połączona z siecią lokalną. Czujniki potrafią komunikować się z wykorzystaniem protokołu MQTT/TCP. Do monitorowanych pomieszczeń należą: chłodnia, magazyn i hala produkcyjna. Prototyp systemu ma za zadanie zbierać, składować, udostępniać i wizualizować zmierzone parametry powietrza.

3.4.2 Monitorowanie stanu zużycia chemii przemysłowej

W zakładzie produkcyjnym rozlokowane są trzy zbiorniki (bufory) chemii przemysłowej. Zbiorniki są połączone z zewnętrznym zbiornikiem głównym, z którego uzupełniany jest ich stan. W celu automatycznego napełniania zbiorników wykorzystano sterownik klasy PLC podłączony do lokalnej sieci Ethernet, który otwiera albo zamyka odpowiednie zawory i nadzoruje pracę pomp. Odczytuje on dane z przepływomierzy pozwalające na monitorowanie stanu zużycia chemii w każdej z trzech stref oraz poziom cieczy w zbiorniku głównym. Zadaniem prototypu jest cykliczny odczyt wskazań dla każdego zbiornika znajdujących się w pamięci PLC z wykorzystaniem protokołu Modbus/TCP. Zebrane dane powinny być składowane w chmurze oraz wizualizowane.

3.4.3 Wyznaczanie parametru KPI

Na wyjściu trzech linii procesowych rozmieszczona jest aparatura przeznaczona do oceny jakości dostarczanych produktów. Urządzenie, w którym zapisywany jest wynik, jest podłączone do lokalnej sieci Ethernet. Informacja o spełnieniu albo niespełnieniu wymagań jakościowych jest przesyłana do serwera z wykorzystaniem CoAP/UDP w oparciu o architekturę REST. Zadaniem realizowanym przez prototyp jest odbiór przesyłanych rezultatów inspekcji, przetwarzanie i składowanie w chmurze. Przetwarzanie danych polega w tym przypadku na wyznaczaniu wartości parametru FTQ (ang. *First Time Quality*). Jest to jeden ze wskaźników KPI będący miarą zdolności linii do produkowania bez wad. FTQ to iloraz liczby produktów, które spełniły wymagania jakościowe i liczby wszystkich dostarczonych produktów wyrażony wzorem (1). Jego wartość podaje się zwykle w procentach [23]. Wskaźnik FTQ wyznaczany dla każdej z trzech rozpatrywanych linii procesowych powinien być na bieżąco wizualizowany.

$$FTQ = \frac{s}{n} \tag{1}$$

gdzie:

s — liczba produktów wyprodukowanych bez wad,

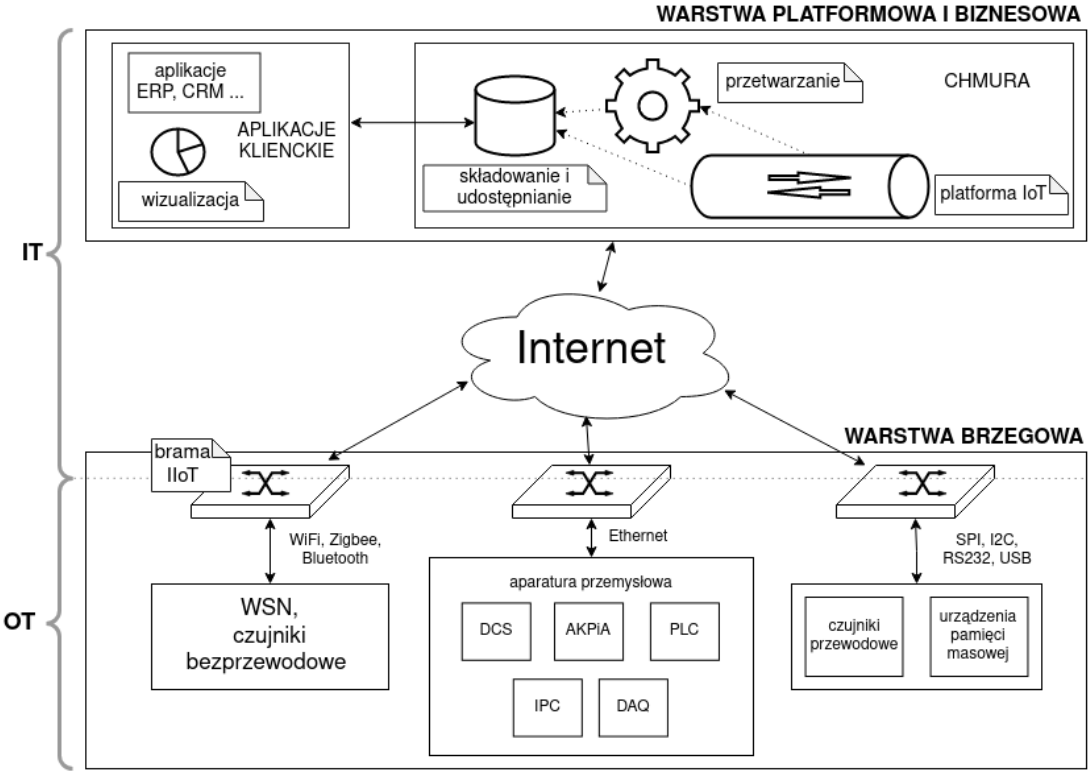
n — liczba wszystkich produktów wyprodukowanych w procesie.

4 Specyfikacja wewnętrzna

Niniejszy rozdział zawiera specyfikację ogólnego projektu systemu oraz opis szczegółów implementacji wykonanego prototypu.

4.1 Ogólna koncepcja rozwiązania

Proponowane rozwiązanie opiera się na trójwarstwowym wzorcu architektury systemów IIoT (patrz: rozdział 2.4, rys. 2), który zaadaptowano w celu praktycznego zastosowania, uwzględniwszy zdefiniowane wymagania projektowe oraz dostępne narzędzia i usługi. Wypracowaną koncepcję zobrazowano na rys. 5.



Rysunek 5: Ogólna koncepcja architektury systemu

Pierwszym punktem zbierania danych w systemie są bramy IoT (IIoT). Ich wykorzystanie pozwala na integrację zróżnicowanej pod względem technicznym aparatury pomiarowej. Bramy mają dostarczać odpowiednie interfejsy fizyczne, jak również dokonywać translacji protokołów pośrednicząc w ten sposób w wymianie danych pomiędzy warstwą brzegową i biznesową [14], [39], [12], [64], [36]. Realizują one także wstępne przetwarzanie w warstwie brzegowej i ułatwiają sprawowanie kontroli nad przepływem danych w systemie [13]. Idea stosowania bramy IoT w przemyśle umożliwia spełnienie głównego zadania IIoT, jakim jest połączenie technologii operacyjnych i informacyjnych [52]. Na omawianym rysunku zamieszczono trzy bramy, które integrują urządzenia odpowiednio w ramach sieci bezprzewodowej, przemysłowej sieci Ethernet oraz

interfejsów szeregowych. Jest to podział poglądowy. W zależności od indywidualnych potrzeb i dostępnego sprzętu możliwe są różne konfiguracje: pojedyncza brama pracująca w wielu sieciach, bramy dedykowane dla określonych sieci, hierarchiczna struktura bram.

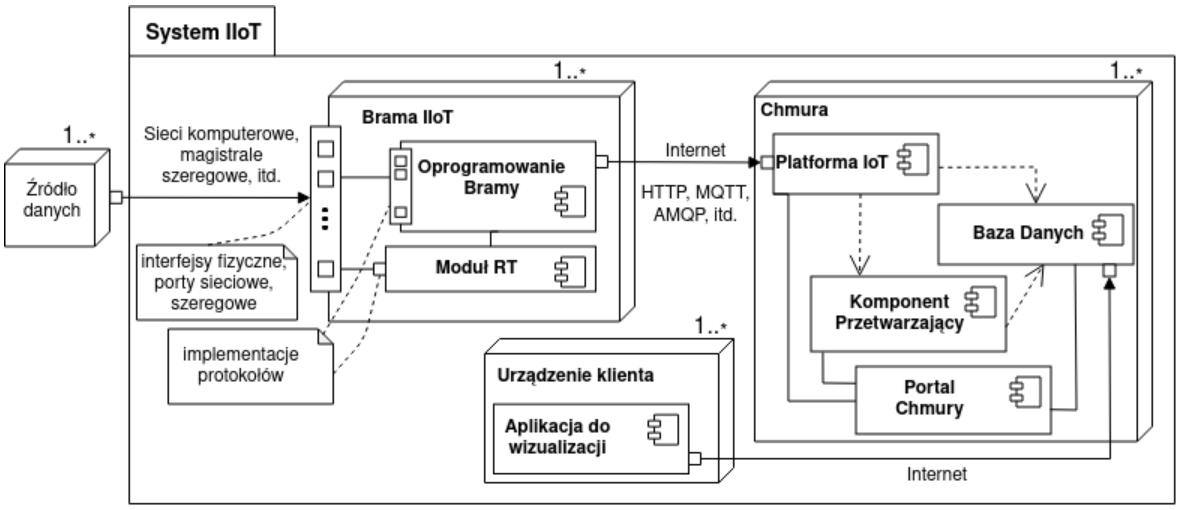
Za pośrednictwem Internetu brama przekazuje dane do warstw wyższych, z wykorzystaniem których odbywa się kolejny etap gromadzenia, a także przetwarzanie, składowanie i udostępnianie danych. Są to warstwy platformowa i biznesowa, których techniczną implementację umożliwiają odpowiednie usługi chmurowe popularnych dostawców: Microsoft, Amazon, Google, Intel, IBM. Możliwe jest też wdrożenie przez przedsiębiorstwo chmury prywatnej lub dedykowanej infrastruktury i oprogramowania. Do warstwy biznesowej należą także aplikacje klienckie, którym są udostępniane dane składowane w bazie. Wśród potencjalnych klientów należy wyszczególnić oprogramowanie, które realizuje zdefiniowane w ramach projektu wymaganie wizualizacji danych.

4.2 Komponentowy model architektury

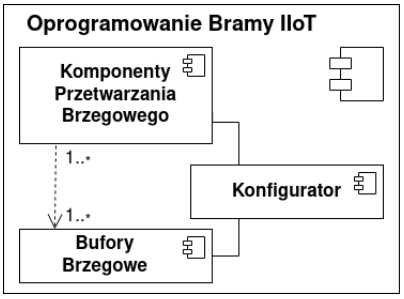
Wychodząc od koncepcji opisanej w poprzednim rozdziale opracowano formalny model architektury projektowanego systemu przedstawiony w postaci diagramu na rys. 6. Składa się on z licznych luźno powiązanych komponentów rozlokowanych na określonych platformach sprzętu i oprogramowania. Rozdział systemu na pojedyncze elementy składowe odpowiada na wymaganie uniwersalności, zapewnia lepszą konfigurowalność i skalowalność. Modularny charakter systemu udogadnia także niezależną modyfikację i rozbudowę poszczególnych podzespołów. Biorąc pod uwagę fakt, że system obejmuje szeroką dziedzinę (od warstwy procesowej po wysokopoziomowe aplikacje IT) i może być utrzymywany przez zespoły o różnych specjalnościach, jest to aspekt szczególnie ważny. Środkiem współpracy grup komponentów osadzonych w `Bramie IIoT`, `Chmurze` i na `Urządzeniach Klienckich` jest Internet i popularne protokoły internetowe: HTTP, FTP, MQTT, AMQP, WebSocket, TLS.

4.2.1 Brama IIoT

Zamieszczona na diagramie z rys. 6 `Brama IIoT` to urządzenie (system komputerowy) posiadające odpowiednie porty (interfejsy fizyczne, porty sieciowe, porty szeregowe), przez które może komunikować się z aparaturą pomiarową reprezentowaną przez `Źródło danych`. Zbieranie danych w jednym miejscu, translację protokołów, przetwarzanie brzegowe i komunikację z komponentami udostępnianymi za pośrednictwem Internetu realizuje dedykowane `Oprogramowanie Bramy`. Jest to komponent złożony składający z `Komponentów Przetwarzania Brzegowego` realizujących określone funkcje przetwarzające oraz `Buforów Brzegowych`, które umożliwiają agregację i zachowanie jednolitego formatu danych wyjściowych. Szczegóły komponentu `Oprogramowania`



Rysunek 6: Komponentowy model architektury systemu



Rysunek 7: Model oprogramowania bramy IIoT

`Bramy` przedstawiono na rys. 7.

W ramach bramy może funkcjonować również sprzętowo-programowy moduł czasu rzeczywistego RT (ang. *Real-Time*), którego celem jest współpraca z lokalnym systemem czasu rzeczywistego.

W systemie przewiduje się istnienie wielu źródeł danych, jak i możliwe jest wykorzystanie wielu bram. Ponadto żeby zapewnić większą niezawodność i skalowalność, można rozważyć budowę klastra bram z redundancją i równoważeniem obciążenia. Jako sprzętową realizację bramy w literaturze spotyka się komputer jednopłytkowy RaspberryPi, a także bramki z serii Siemens Simatic IOT2000 [52], [76], [77], [64]. Można też wziąć pod uwagę wykorzystanie platformy BeagleBone Black, czy dedykowanych rozwiązań producentów Cisco, Dell, Huawei, Hewlett Packard, NXP i innych [31]. Do realizacji oprogramowania można posłużyć się narzędziami przeznaczonymi do integracji urządzeń IoT. Jednym z najpopularniejszych programów jest darmowe narzędzie Node-RED [1], [12], [76], [52], [77]. Alternatywne bezpłatne rozwiązania to m.in. n8n.io, Total.js Flow. Wśród ofert komercyjnych dostępne są przykładowo platformy Crosser, Bosch ProSyst [53], [49], [74], [17], [31].

4.2.2 Chmura

Przedstawiony na rys. 6 diagram o nazwie `Chmura` może w rzeczywistości odpowiadać chmurze publicznej, prywatnej lub innej dedykowanej infrastrukturze sprzętu i oprogramowania. W niniejszym projekcie w szczególności rozpatruje się wariant z chmurą publiczną, której usługi są dostępne za pośrednictwem Internetu. Pozwalają one na techniczną realizację postawionych wymagań przetwarzania, składowania i udostępniania danych. Umieszczone wewnątrz diagramu komponenty zostały dobrane pod kątem zdefiniowanych wymagań funkcjonalnych. `Platforma IoT` to komponent, który odbiera dane przychodzące z urządzeń (bram IIoT). Umożliwia również zarządzanie nimi: dodawanie, usuwanie, modyfikację uprawnień, a także monitoring i diagnostykę. Dane z platformy mogą zostać bezpośrednio składowane w `Bazie Danych`. Możliwe jest też przekazanie ich na wejście `Komponentu Przetwarzającego`. Pod tym pojęciem rozumiane są wszelkie usługi chmurowe realizujące tzw. bezserwerowe przetwarzanie danych (ang. *serverless computing*). Nie oznacza to, że serwer nie istnieje, lecz że jest on niejako „zakryty" przed programistą, który skupia się jedynie na dostarczeniu logiki w postaci wsadowej [58]. Usługi przetwarzania bezserwerowego dokonują ewentualnego odczytu i zapisu do `Bazy Danych`. Wśród usług oferowanych przez liderów rynku chmurowego — Amazon Web Services (AWS) i Microsoft Azure [24] — dokonano przeglądu przykładowych, które mogą być użyte do implementacji opisanych komponentów. W poniższych punktach oddzielono znakiem „ | " odpowiednio nazwy usług AWS i Azure, które mają podobny zakres funkcjonalny [2].

Przykładowe usługi użyteczne do realizacji `Platformy IoT`:

- IoT | IoT Hub — Brama w chmurze umożliwiająca komunikację z urządzeniami w bezpieczny i skalowalny sposób,
- IoT Things Graph | Digital Twins — Tworzenie cyfrowych modeli systemów fizycznych umożliwiających zbieranie danych ze świata rzeczywistego.

Przykładowe usługi użyteczne do realizacji `Bazy Danych`:

- Simple Storage Services | Blob Storage — Magazyn do składowania, udostępniania, archiwizowania i tworzenia kopii zapasowej obiektów,
- RDS | SQL Database — Relacyjne bazy danych,
- DynamoDB | CosmosDB — Nierelacyjne bazy danych,

Przykładowe usługi użyteczne do realizacji `Komponentów Przetwarzania`:

- Lambda | Functions — Bezserwerowe, skalowalne funkcje implementowane, z wykorzystaniem języków programowania: Javascript, Python, Java, C#,

- SageMaker | Machine Learning — Uczenie maszynowe,
- Kinesis Analytics | Stream Analytics — Tworzenie potoków przetwarzania strumieniowego dla dużej ilości danych z wykorzystaniem języka SQL.

4.2.3 Aplikacja kliencka

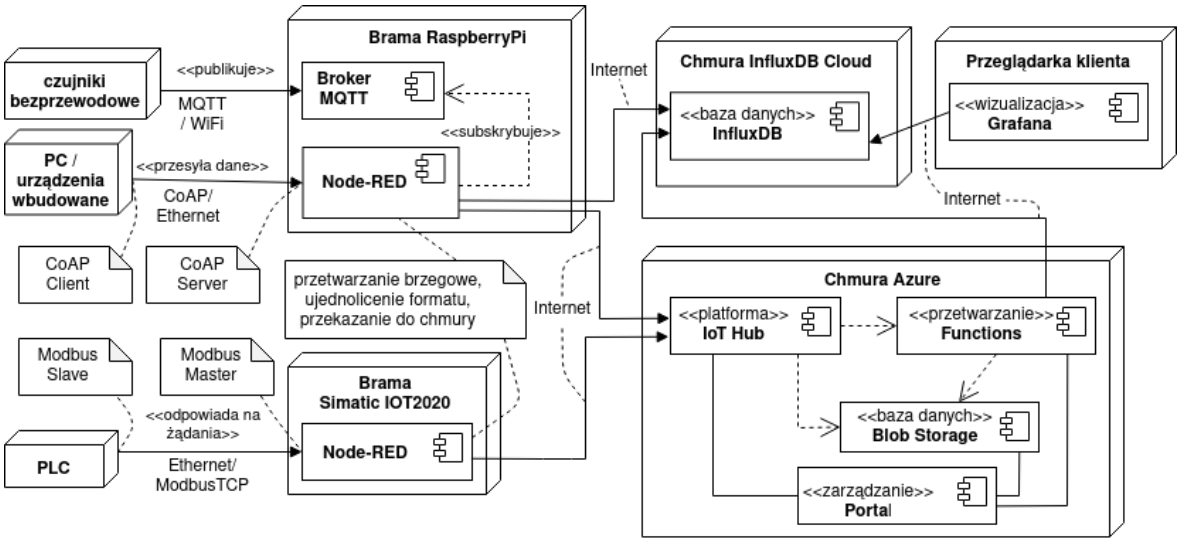
Przedstawione na rys. 6 `Urządzenie Klienta` może w rzeczywistości być komputerem osobistym, urządzeniem mobilnym, serwerem lub usługą chmurową. Na urządzeniu jest wdrożone `Oprogramowanie Klienta`. Jest to komponent, który korzysta za pośrednictwem Internetu z danych udostępnianych przez `Chmurę`. W kontekście niniejszej pracy rozpatruje się w szczególności aplikacje wizualizujące dane i wspomagające analitykę. Wśród potencjalnie użytecznych narzędzi dostępne są darmowe rozwiązania *open source*, które pozwalają na tworzenie modularnych paneli wizualizacji w ramach aplikacji internetowej (przeglądarkowej): Grafana, Freeboard, Kibana, Dash. Na rynku oferowane są także komercyjne platformy analityczne: PowerBI, Tableau, Grow.

4.3 Specyfikacja wewnętrzna prototypu

Na podstawie architektury systemu opisanej w poprzednim rozdziale wykonano prototyp, który implementuje scenariusze użycia zdefiniowane w rozdziale 3.4. Model zrealizowanego prototypu przedstawiono na rys. 8. System składa się z dwóch bram IIoT: Siemens Simatic IOT2020 i RaspberryPi. Źródła danych zostały zasymulowane przez komputery, na których uruchomiono specjalnie wykonane oprogramowanie generujące zadane przebiegi. Sporządzone narzędzie do symulacji danych opisano szerzej w rozdziale 6. Na potrzeby translacji protokołów i przetwarzania brzegowego w systemach obu bram zainstalowano platformę Node-RED. W celu dalszego zbierania, przetwarzania, składowania i udostępniania danych użyto usług chmury Azure. Na potrzeby bieżącej wizualizacji danych wykorzystano bazę danych szeregów czasowych InfluxDB osadzoną w chmurze InfluxDB Cloud. Zebrane tam dane pomiarowe są cyklicznie odczytywane i wizualizowane przez narzędzie Grafana. W kolejnych rozdziałach omówiono poszczególne elementy systemu.

4.3.1 Bramy IIoT — Simatic IOT2020 i RaspberryPi

Wyboru urządzenia IOT2020 dokonano ze względu na spełnienie wymaganych standardów przemysłowych, o czym zapewnia producent. Posiada ono certyfikat UL/CE. Płytka elektroniczna jest umieszczona w obudowie, która jest przystosowana do montażu na popularnych w przemyśle szynach DIN i zapewnia ochronę o stopniu IP20. Bramka ma wbudowane interfejsy USB, Ethernet oraz moduł rozszerzeń GPIO. W systemie komputerowym Intel Quark x1000 400 MHz, 512 MB RAM można uruchomić



Rysunek 8: Model implementacji prototypu

system Linux z obrazu na karcie microSD. Producent dostarcza obraz systemu zbudowany na dystrybucji Yocto Linux dedykowanej dla urządzeń wbudowanych. Udostępniona jest także wersja z modułem czasu rzeczywistego i sterownikami Profinet, dzięki której brama IIoT może funkcjonować z lokalnym systemem czasu rzeczywistego [66], [65], [64].

Jako że brama IOT2020 zapewnia lepsze przystosowanie do środowiska przemysłowego, postanowiono wykorzystać ją do zbierania danych ze sterownika PLC, który monitoruje stan zużycia chemii przemysłowej. Komunikacja odbywa się w ramach sieci Ethernet z wykorzystaniem protokołu Modbus/TCP. Modbus to protokół, w którym występuje węzeł nadrzędny (ang. *Master*) oraz węzły podrzędne (ang. *Slave*). Węzeł nadrzędny może dokonywać odczytu i zapisu do węzła podrzędnego z wykorzystaniem odpowiednich funkcji kodowanych w ramce. Pierwotnie protokół był zaprojektowany dla połączeń szeregowych, z czasem zaczęto używać go także na stosie TCP/IP kapsułkując ramki Modbus w ramce TCP [23], [40]. W rozpatrywanej konfiguracji brama pełni rolę węzła nadrzędnego, który cyklicznie odczytuje zawartość rejestrów węzła podrzędnego (sterownika PLC).

Minikomputer RaspberryPi nie spełnia ściśle wymagań przemysłowych, natomiast ze względu na dostępne wsparcie użyto go na potrzeby prototypownia i rozwoju. Jest to bowiem platforma bardzo popularna, posiadająca liczną społeczność zwolenników. RaspberryPi jako bramę IoT/IIoT można spotkać również w literaturze [52], [76], [77]. W projekcie użyto modelu 3 B, który posiada m. in. czterordzeniowy procesor o taktowaniu 1.2 GHz, 1 GB RAM, interfejsy USB, Ethernet, Bluetooth, GPIO. Z karty microSD można uruchomić dedykowaną dystrybucję systemu Linux RaspberryPi OS lub inną.

Bramę RaspberryPi wykorzystano do komunikacji z siecią bezprzewodowych czuj-

ników mierzących parametry powietrza w określonych pomieszczeniach oraz z komputerami zainstalowanymi przy trzech stacjach procesowych, które przesyłają informacje o ocenie jakości wytworzonych produktów. Komunikacja z czujnikami odbywa się z wykorzystaniem sieci WiFi i protokołu MQTT. MQTT (ang. *Message Queue Telemetry Transport*) to lekki pod względem wykorzystania zasobów protokół polegający na wymianie komunikatów. Komunikacja jest oparta na wzorcu publikuj-subskrybuj (ang. *publish-subscribe*). Klienci łączą się do centralnego serwera (pośrednika, ang. *broker*), za pośrednictwem którego mogą wysyłać i odbierać komunikaty pod konkretnym adresem (tematem, ang. *topic*). Tematy tworzą hierarchiczną strukturę analogiczną do URL. MQTT wykorzystuje TCP w warstwie transportowej [61]. W omawianym scenariuszu czujniki publikują dane pomiarowe pod tematami zarezerwowanymi odpowiednio dla pomieszczenia chłodni, magazynu i hali produkcyjnej. Brama jest subskrybentem, który nasłuchuje na każdym z trzech tematów.

Wymiana danych z komputerami, które zbierają informacje na temat jakości produktów odbywa się za pośrednictwem sieci Ethernet i protokołu CoAP. CoAP (ang. *Constrained Application Protocol*) to alternatywa dla HTTP przeznaczona dla urządzeń o ograniczonych zasobach. Wykorzystuje m. in. protokół UDP w warstwie transportowej oraz krótsze nagłówki w formie binarnej. CoAP jest zorientowany na zasoby umożliwiając ich tworzenie, odczyt, aktualizację i usuwanie. Analogicznie do HTTP realizowany jest model komunikacji klient-serwer z użyciem metod GET, POST, PUT, DELETE, itd. [35], [61]. W realizowanym przypadku biznesowym brama jest serwerem, a trzy komputery przy stacji procesowej klientami, które przesyłają dane metodą POST.

4.3.2 Oprogramowanie bram — Node-RED

Komponent `Oprogramowania` obu bram (patrz: rys. 7 , rozdział 4.2) zaimplementowano z wykorzystaniem narzędzia Node-RED. Jest to darmowe oprogramowanie typu *open source* funkcjonujące w środowisku Node.js. Jego celem jest integracja interfejsów fizycznych oraz interfejsów programowania aplikacji. W Node-RED stosuje się paradygmat programowania opartego na przepływie danych. Program posiada postać wizualną, w ramach której łączy się węzły (ang. *nodes*) realizujące określone funkcjonalności. Pomiędzy wyjściem i wejściem kolejnych węzłów przekazywany jest komunikat (wiadomość, ang. *message*) w powszechnym formacie JSON (ang. *JavaScript Object Notation*). Węzły mogą być prostymi lub złożonymi funkcjami, a nawet rozbudowanymi bibliotekami napisanymi w języku JavaScript. Wizualny model programowania odpowiada na wymaganie uniwersalności, jako że może być on łatwo zrozumiały dla specjalistów różnych dziedzin pracujących w przemyśle [1]. Node-RED to serwer, którego panel administracyjny i środowisko programistyczne udostępniane są w ramach aplikacji przeglądarkowej. Wyboru opisanego narzędzia dokonano ze względu na ela-

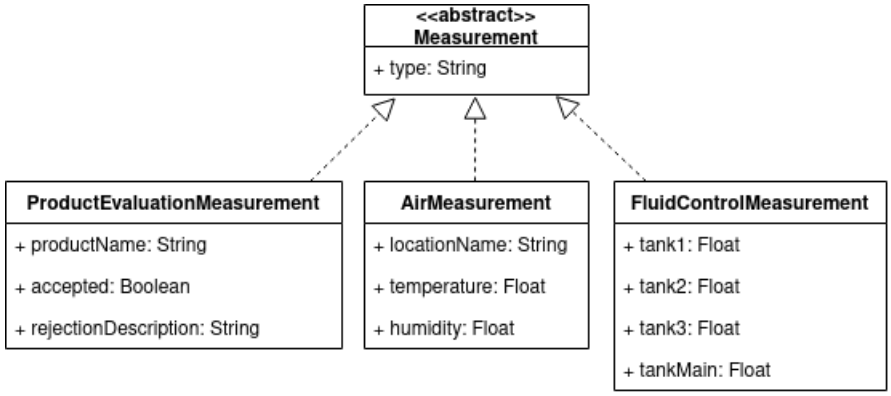
styczność sposobu implementacji logiki i wsparcie w postaci wielu bibliotek umożliwiających translację protokołów. Dystrybucja w formie *open source* daje większą kontrolę nad używanym oprogramowaniem, zaś licencja Apache 2.0 pozwala na jego bezpłatne wykorzystanie również w celach komercyjnych [53]. Node-RED jest stosowane w IoT stosunkowo często [12], [76], [52], [77].

Zadaniem bramy zdefiniowanym w obrębie wymagań jest zapewnienie jednolitych struktur danych stosowanych w warstwie aplikacji na poziomie wszystkich systemów informatycznych w przedsiębiorstwie. W tym celu proponuje się wykorzystanie formatu JSON. Za pomocą notacji JSON Schema możliwe jest formalne określanie struktur danych, które powinny być zaimplementowane przez wszystkie systemy.

Struktury danych niezbędne do realizacji przypadków biznesowych przedstawiono na rys. 9. Zdefiniowano następujące klasy:

- `Measurement` — Klasa abstrakcyjna reprezentująca pomiar, który ma określony typ, np. pomiar powietrza. Jawna informacja o typie może być zbędna na poziomie języka programowania, lecz musi być zawarta w obiekcie JSON, który podlega serializacji;
- `FluidControlMeasurement` — Reprezentuje informację o poziomie zużycia cieczy w strefach 1, 2, 3 oraz o poziomie cieczy w zbiorniku głównym;
- `AirMeasurement` — Reprezentuje pomiar powietrza. Posiada kolejno pola informujące o lokacji, wartości temperatury i wilgotności;
- `ProductEvaluationMeasurement` — Reprezentuje ocenę jakości produktu. Posiada kolejno pola informujące o nazwie produktu, spełnieniu wymagań (tak/nie — `true`/`false`) i ewentualnej przyczynie niespełnienia wymagań.

Przykładową definicję struktury `AirMeasurement` w notacji JSON Schema przedstawiono na listingu 1.



Rysunek 9: Struktury danych używane w warstwie biznesowej systemu

Listing 1: Przykład definicji obiektu JSON w notacji JSON Schema dla struktury `AirMeasurement`

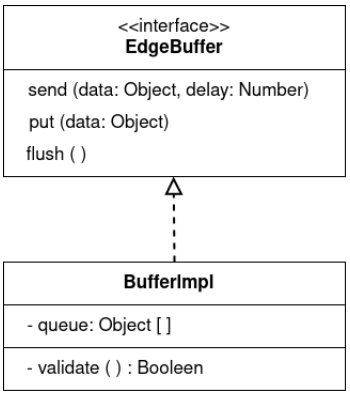
```
{
      "anyOf": [
               {
                 "type": "object",
                 "required": [
                   "type",
                   "locationName",
                   "temperature",
                   "humidity"
                 ],
                 "properties": {
                   "type": {
                     "type": "string"
                   },
                   "locationName": {
                     "type": "string"
                   },
                   "temperature": {
                     "type": "number"
                   },
                   "humidity": {
                     "type": "number"
                   }
                 }
               }
      ]
}
```

4.3.3 Bufor Brzegowy w Node-RED

Na potrzeby realizacji komponentu `Bufora Brzegowego` (patrz: rozdział 4.2.1) zaprojektowano własny węzeł Node-RED o nazwie `EdgeBuffer`. Jego model przedstawiono na rys. 10. Bufor ma być w domyśle umieszczany tuż przed węzłem wysyłającym dane na zewnątrz systemu. Jego rolą jest sprawdzenie, czy format przekazanych danych jest zgodny ze strukturami zdefiniowanymi za pomocą notacji JSON Schema. Ma on także umożliwić kumulowanie danych w celu ich agregacji. Bufor posiada następujące metody publiczne:

- `send` — przepisanie danych z wejścia na wyjście z opcjonalnym opóźnieniem i obowiązkową uprzednią walidacją,
- `put` — składowanie danych wejściowych w buforze po uprzedniej walidacji,
- `flush` — przekazanie składowanych danych na wyjście i zwolnienie bufora.

W ramach implementacji `BufferImpl` zdefiniowano prywatną kolejkę `queue`, która umożliwia przekazanie zbuforowanych danych na wyjście w kolejności FIFO. Prywatna metoda `validate` realizuje logikę walidacji formatu danych wejściowych i zwraca wartość `true`, gdy dane są poprawne. W konfiguracji Node-RED przewidziano parametr



Rysunek 10: Model `Bufora Brzegowego`

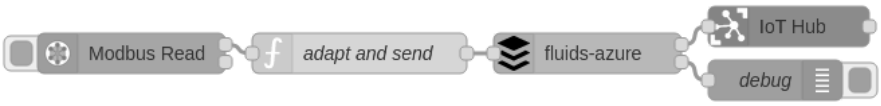
określający ścieżkę do pliku zawierającego definicje prawidłowych struktur danych w notacji JSON Schema.

4.3.4 Zbieranie informacji o stanie zużycia chemii przemysłowej

Diagram przepływu Node-RED realizujący logikę zbierania danych ze sterownika PLC przedstawiono na rys. 11. Implementację protokołu Modbus/TCP dostarcza biblioteka `node-red-contrib-modbus`. Jednym z jej komponentów jest węzeł `Modbus Read`, który implementuje węzeł nadrzędny (w sensie protokołu Modbus) umożliwiający cykliczny odczyt danych ze sterownika PLC będącego węzłem podrzędnym. Przyjęto, że dane o zużyciu cieczy przemysłowej w strefach 1, 2, 3 są zapisane w kolejnych rejestrach 40001, 40002, 40003 jako liczby całkowite określające zużycie w litrach. Podobnie informacja o poziomie cieczy w zbiorniku głównym zapisana jest w rejestrze 40004. Węzeł `Modbus Read` skonfigurowano do użycia funkcji Modbus `FC 3: Read Holding Registers` w celu odczytu danych z czterech sąsiednich rejestrów, począwszy od adresu 40001. Odczyt wykonywany jest co 3 sekundy. Funkcja `adapt and send` przekształca otrzymaną na wejściu tablicę wartości na oczekiwaną strukturę `FluidControlMeasurement` i przekazuje dane do bufora wywołując metodę `send`. Węzeł `fluids-azure` to `Bufor Brzegowy`, którego rola sprawdza się w rozpatrywanym przypadku jedynie do sprawdzenia poprawności otrzymanej na wejściu struktury i natychmiastowym przekazaniu jej na wyjście. Na końcu umieszczono węzeł `IoT Hub` pochodzący z biblioteki `node-red-contrib-azure-iot-hub`, który implementuje interfejs usługi IoT Hub chmury Azure. Umożliwia on przekazywanie danych do chmury za pośrednictwem Internetu. Węzeł `debug` wyświetla w konsoli informacje o ewentalnych błędach, w tym m. in. komunikaty bufora o niepowodzeniu walidacji.

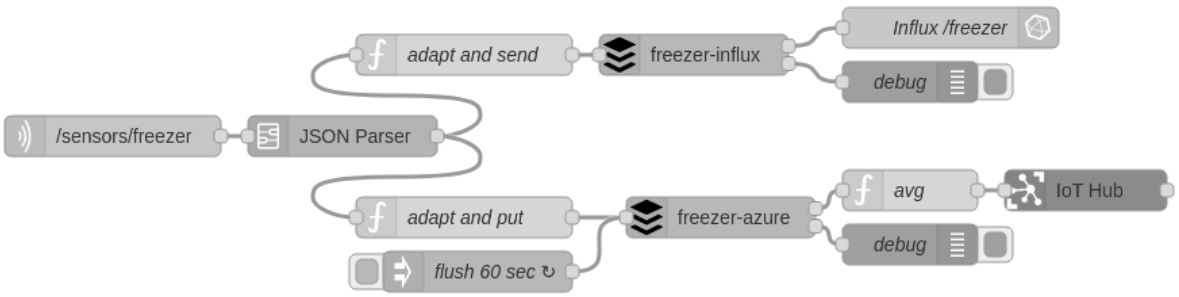
4.3.5 Zbieranie pomiarów parametrów powietrza

Na rys. 12 zamieszczono diagram przepływu wykonujący gromadzenie pomiarów parametrów powietrza w ramach protokołu MQTT. Serwer pośredniczący w wymia-



Rysunek 11: Diagram przepływu Node-RED realizujący odczyt danych ze sterownika PLC

nie komunikatów (broker) zralizowano z wykorzystaniem darmowego oprogramowania Eclipse Mosquitto zainstalowanego na RaspberryPi [46]. Węzeł `/sensors/freezer` to subskrybent MQTT dostępny w ramach standardowej biblioteki Node-RED nasłuchujący na temacie o tej samej nazwie. Publikacji komunikatów dokonują w tym przypadku czujniki rozmieszczone w chłodni. Identyczne diagramy, lecz z dopasowanymi tematami `/sensors/warehouse` i `/sensors/production` opracowano dla czujników osadzonych odpowiednio w magazynie i hali produkcyjnej. Węzeł `JSON Parser` deserializuje nadesłane wiadomości z postaci znakowej na obiekt JSON. Funkcja `adapt and put` mapuje dane otrzymane z czujników na strukturę `AirMeasurement` i składa w buforze `freezer-azure`, który standardowo waliduje format danych na wejściu. Węzeł `flush 60 sec` co minutę wywołuje metodę `flush` zwalniającą bufor. Funkcja agregująca `avg` wylicza średnią z zebranych wartości i wysła pojedynczą wiadomość do serwisu IoT Hub. Funkcja `adapt and send` dostosowuje format danych do bazy InfluxDB, która wymaga przekazania prostego obiektu JSON składającego się jedynie z pól temperatury i wilgotności. Tylko te dane telemetryczne są wykorzystywane do wizualizacji. Na buforze `freezer-influx` wywoływana jest metoda `send`. Dalej dane przekazywane są do węzła `influx /freezer` pochodzącego z biblioteki `node-red-contrib-influxdb`. Przekazuje on otrzymane dane pomiarowe do chmury InfluxDB Cloud dołączając do nich atrybut `measurement` określający rodzaj pomiaru — w tym przepadku jego wartość to `"freezer"`. Dla pozostałych monitorowanych pomieszczeń używane są wartości `"warehouse"` oraz `"production"`.



Rysunek 12: Diagram przepływu Node-RED realizujący zbieranie danych dostarczanych z czujników mierzących parametry powietrza

4.3.6 Zbieranie danych do wyznaczenia parametru KPI

Algorytm przetwarzania danych zbieranych w ramach protokołu CoAP z trzech stacji kontroli jakości przedstawiono na rys. 13. Węzeł `[POST] /ftq` pochodzi z biblioteki `node-red-contrib-coap` i implementuje serwer CoAP, a ściślej — obsługę punktu końcowego `/ftq`, na który klienty zainstalowane przy stacjach przesyłają wyniki ewaluacji. Funkcja `CoAP OK` wysyła pozytywną odpowiedź na zapytanie i przekazuje otrzymane dane do kolejnych węzłów. Dalszy przebieg algorytmu jest zbliżony do wcześniej opisanych przypadków, gdzie dane są walidowane przez bufor i wysyłane do serwisu IoT Hub.

Rysunek 13: Diagram przepływu Node-RED realizujący zbieranie danych dostarczanych ze stacji kontroli jakości

4.3.7 Użycie chmury Azure

Chmura Microsoft Azure udostępnia usługi, które pozwalają na realizację wszystkich komponentów zdefiniowanych na potrzeby projektu (patrz: rozdział 4.2.2). Z kolei Node-RED posiada dobre wsparcie dla Azure w postaci licznych bibiliotek. Ponadto Microsoft oferuje studentom darmową subskrypcję *Azure for Students* dającą bezpłatny dostęp do określonego przez dostawcę zbioru usług [3]. Stąd platforma Azure została wybrana do prototypowania.

4.3.8 Zbieranie danych w chmurze — Azure IoT Hub

Jako `Platformy IoT` użyto usługi IoT Hub. Pozwala ona na rejestrowanie wielu urządzeń IoT, zarządzanie ich uprawnieniami, zbieranie danych i wysyłanie wiadomości do urządzeń. W portalu zarejestrowano obie wykorzystane bramy IIoT, co przedstawia zrzut ekranu zamieszczony na rys. 14. W ustawieniach usługi IoT Hub zdefiniowano także trasowanie (ang. *message routing*) na wejście usług Blob Storage i Functions pełniących rolę komponentu `Bazy Danych` i `Kompunentu Przetwarzającego`.

Device ID	Status	Last Status Update (UTC)	Authentication Type
RaspberryPi	Enabled	--	Sas
IoT2020	Enabled	--	Sas

Rysunek 14: Zrzut ekranu przedstawiający tabelę urządzeń zarejestrowanych w usłudze IoT Hub

4.3.9 Składowanie w chmurze i udostępnianie — Azure Blob Storage

Baza Blob Storage umożliwia elastyczne przechowywanie danych zebranych w usłudze IoT Hub w formie obiektów JSON zapisanych w strukturze plików i folderów, którą można swobodnie przeglądać w portalu Azure (interfejs graficzny przypomina menadżera plików). W prototypowym systemie baza Blob Storage przechwuje zagregowane wartości parametrów temperatury `AirMeasurement`, aktualne wskazania przepływomierzy i czujnika poziomu cieczy `FluidControlMeasurement`, a także liczbę produktów wytworzonych bez wad i liczbę wszystkich produktów na potrzeby wyznaczania wartości parametru FTQ. Pobieranie i zapis danych przez systemy zewnętrzne umożliwia interfejs programistyczny REST z wykorzystaniem protokołu HTTP [10] W ten sposób realizowane są wymagania składowania i udostępniana danych pomiarowych.

4.3.10 Przetwarzanie w chmurze — Azure Functions

Dane zbierane za pośrednictwem usługi IoT Hub są także przekazywane na wejście usługi Azure Functions realizującej wymaganie przetwarzania danych. Umożliwia ona impelemtację prostych funkcji bezserwerowych w wybranych językach programowania. Programista zajmuje się jedynie implementacją logiki, zaś dostarczenie niezbędnych zasobów informatycznych i ich skalowanie zapewnia chmura. W ramach prototypu napisano funkcję w języku Javascript, która wyznacza wartość parametru FTQ. Na wejściu przyjmuje ona wiadomość informującą o wyniku oceny jakości nowo wytworzonego produktu w postaci `ProductEvaluationMeasurement`. Z bazy Blob Storage pobierane są ostatnie wartości liczników produktów wyprodukowanych bez wad oraz wszystkich produktów. Odpowiedni licznik jest inkrementowany i wyznaczana jest nowa wartość parametru FTQ na podstawie wzoru 1 zamieszczonego w rozdziale 3.4. Następnie aktualizowane są wpisy w bazie i wynik jest przesyłany do chmury InfluxDB na potrzeby bieżącej wizualizacji. Kod źródłowy funkcji przedstawiono na listingu 2. Zdefiniowano także drugą, prostszą funkcję pomocniczą, która przekazuje wartości parametru `FluidControlMeasurement` do bazy InfluxDB. Zbierająca wartości tego parametru brama IOT2020 nie wspiera bowiem na chwilę obecną oprogramowania umożliwiającego bezpośredni przesył danych z Node-RED do chmury InfluxDB, stąd konieczne było wykorzystanie pośrednika w psotaci Azure IoT Hub i Functions.

Listing 2: Kod źródłowy funkcji wyznaczającej aktualną wartość parametru FTQ osadzonej w usłudze Azure Functions

```
1 IoTHubMessages.forEach(message => {
2       message = JSON.parse(message);
3
4       // akceptacja wiadomosci o okreslonym typie
5       if (message.type !== 'product-evaluation-result') {
6           return;
7       }
```

```
8
9       // wczytanie danych z Blob Storage
10      input = context.bindings.inputBlob;
11      var ftqParams;
12
13      // sprawdzenie, ktorego produktu dotyczy wiadomosc
14      switch(message.productName) {
15          case 'PRODUCT-A':
16              ftqParams = input.prodA;
17              break;
18          case 'PRODUCT-B':
19              ftqParams = input.prodB;
20              break;
21          case 'PRODUCT-C':
22              ftqParams = input.prodC;
23              break;
24      }
25
26      // inkrementacja licznikow
27      incrementFtqCounters(ftqParams, message.accepted);
28
29      // wyznaczenie wartosci ftq
30      const ftq = ftqParams.positive / ftqParams.all * 100;
31
32      // przeslanie wyniku do bazy InfluxDB
33      sendToInflux(message.productName, ftq, context);
34
35      // zapis wyniku do Blob Storage
36      context.bindings.outputBlob = input;
37  });
```

4.3.11 Wizualizacja danych — InfluxDB i Grafana

Do realizacji wymagania wizualizacji danych użyto bazy danych InfluxDB. Jest to baza zoptymalizowana pod kątem przechowywania szeregów czasowych (ang. *Time Series Database*). InfluxDB to bezpłatne oprogramowanie dostępne na licencji MIT, które można zainstalować na dowolnej maszynie z systemem Linux, Mac OS X, Windows. Dostępna jest także płatna usługa chmurowa o nazwie InfluxDB Cloud firmy InfluxData. Usługa ta oferuje hosting bazy w chmurze wraz z panelem zarządzania dostępnym z przeglądarki jako aplikacja internetowa. Osadzenie bazy w chmurze umożliwia łatwą konfigurację i skalowanie „na żądanie“ bez konieczności zajmowania się szczegółami technicznymi. Dostawca przewodział darmowy program startowy, co wykorzystano w niniejszym projekcie na potrzeby prototypowania. Dane pomiarowe są składowane w InfluxDB jako rekordy w tabelach pogrupowanych w konternery — tzw. wiadra (ang. *buckets*). Do podstawowych kolumn należą: znacznik czasowy pomiaru (`_time`), wartość parametru (`_value`), nazwa parametru (`_field`) i nazwa pomiaru (`_measurement`) [28], [29]. Zrzuty ekranów z tabel InfluxDB przechowujących pomiary wilgotności i temperatury przedsawiono na rysunkach 15 i 16.

Z bazą InfluxDB dobrze integruje się Grafana — darmowa aplikacja do wizuali-

_time	_value	_field	_measurement
2021-01-10 20:00:00 GMT+1	7.729460655319148	temperature	freezer
2021-01-10 20:30:00 GMT+1	4.725326039999998	temperature	freezer
2021-01-10 21:00:00 GMT+1	3.6516577798882666	temperature	freezer
2021-01-10 21:30:00 GMT+1	6.0660865298969044	temperature	freezer

Rysunek 15: Zrzut ekranu przedstawiający tabelę InfluxDB zawierającą szeregi czasowe pomiaru temperatury

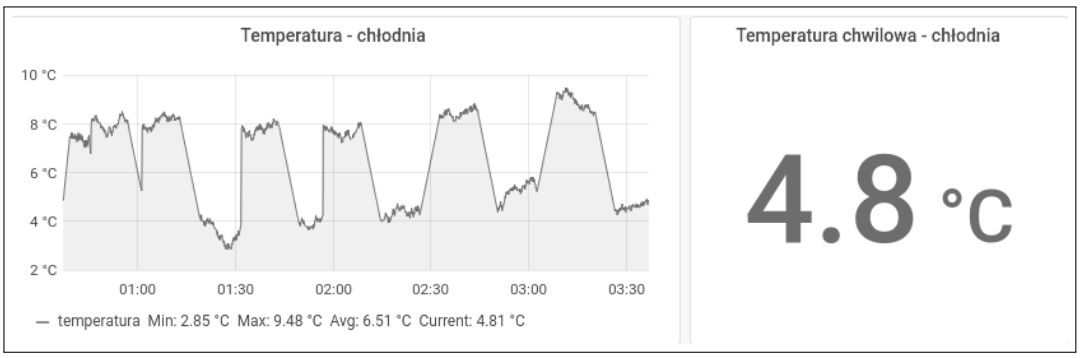
_time	_value	_field	_measurement
2021-01-10 20:00:00 GMT+1	59.647006765957435	humidity	freezer
2021-01-10 20:30:00 GMT+1	45.603607511111086	humidity	freezer
2021-01-10 21:00:00 GMT+1	44.6352671620112	humidity	freezer
2021-01-10 21:30:00 GMT+1	58.20440930927834	humidity	freezer

Rysunek 16: Zrzut ekranu przedstawiający tabelę InfluxDB zawierającą szeregi czasowe pomiaru wilgotności

zacji danych dostępna na licencji Apache 2.0. Podobnie jak w przypadku InfluxDB, możliwa jest lokalna instalacja lub skorzystanie z płatnej usługi chmurowej. W ramach prototypu narzędzie zostało zainstalowane na komputerze klienckim. Jest to serwer, który udostępnia interfejs graficzny w postaci aplikacji przeglądarkowej, w ramach której można tworzyć dowolne diagramy, wykresy, tabele, wskaźniki i swobodnie rozbudowywać modularne panele zarządzania. Istnieje także możliwość tworzenia własnych komponentów graficznych i wtyczek. Grafana komunikuje się z bazą InfluxDB za pośrednictwem Internetu z wykorzystaniem deklaratywnego języka zapytań Flux. Przykładowe zapytanie przedstawiono na listingu 3. Wykonywana jest selekcja rekordów z kontenera o nazwie `bucket`, w których wartość pola `_measurement` to `freezer`, a pola `_field` — `temperature`. Zakres wyników ograniczany jest do trzech ostatnich godzin, a wartości są agregowane z wykorzystaniem funkcji wyznaczającej średnią (`mean`) w obrębie interwałów pięciosekundowych. Wynik zapytania może być wizualizowany przez wybrany komponent graficzny. Na rys. 17 zamieszczono zrzut ekranu będący fragmentem panelu Grafana, przedstawiający wykres i wskaźnik wartości wizualizujące wynki zapytania z listingu 3.

Listing 3: Przykładowe zapytanie w języku Flux

```
1 from(bucket:"bucket")
2       |> range(start:-3h)
3       |> filter(fn:(r) =>
4             r._measurement == "freezer" and
5             r._field == "temperature"
6       )
7       |> aggregateWindow(every: 5s, fn: mean)
```



Rysunek 17: Zrzut ekranu z panelu Grafana wizualizujący wyniki zapytania przedstawione na listing 3

5 Specyfikacja zewnętrzna

W niniejszym rozdziale przedstawiono opis sposobu uruchomienia systemu, wymieniono kategorie użytkowników i dokonano rozważań na temat bezpieczeństwa.

5.1 Uruchomienie systemu

Wyróżniono cztery etapy uruchamiania, wdrażania i konfiguracji systemu: uruchomienie i konfiguracja bram IIoT, konfiguracja usług w chmurze, instalacja i konfiguracja narzędzia do wizualizacji, integracja bram IIoT z usługami chmurowymi. Kolejne podrozdziały dotyczą poszczególnych zagadnień. Specyfikacja ma charakter ogólny, zaś szczegółowe wskazówki do implementacji należy zaczerpnąć z dokumentacji narzędzi i urządzeń. Aktualne odnośniki (styczeń 2021 r.) do odpowiednich dokumentów zamieszczono w przypisach do niektórych akapitów.

5.1.1 Uruchomienie i konfiguracja bram IIoT

W celu uruchomienia urządzeń pełniących rolę bram IIoT należy w pierwszej kolejności przygotować środowisko ich funkcjonowania, tj. wydzielić miejsce ich instalacji zapewniając łatwy dostęp i odpowiedni poziom zabezpieczeń przed czynnikami niesprzyjającymi lub osobami nieuprawnionymi. Następnie konieczne jest dostarczenie wymaganego źródła zasilania określonego w specyfikacji urządzenia. Kolejny etap polega na konfiguracji oprogramowania (w tym ewentualnego systemu operacyjnego) i połączeń sieciowych.

W ramach prototypu użyto dwóch bramy IIoT - Simatic IOT2020 i Raspberry Pi 3 B. Brama IOT2020 wymaga stałego napięcia zasilania mieszczącego się w zakresie 9 - 36V. Pobór prądu wynosi 1.4A. Dla RaspberryPi producent zaleca użycie zasilacza sieciowego microUSB o napięciu wyjściowym 5.1V i prądzie 2.5A. Przy wyborze źródła zasilania należy zwrócić uwagę na rodzaj obudowy. Warto rozważyć produkty posiadające odpowiedni stopień ochrony (np. IP20) i możliwość montażu na szynie DIN (co jest rozwiązaniem popularnym w przemyśle). Dla każdej z bram producenci przygotowali dedykowaną dystrybucję systemu Linux. Obrazy systemów należy najlepiej pobrać z oficjalnych stron i zainstalować na kartach microSD o minimalnej pojemności 8 GB. Po umieszczeniu karty w odpowiednim slocie urządzenia i podłączenia zasilania system uruchamia się samoczynnie. Następnie należy dokonać podstawowej konfiguracji systemu operacyjnego: podłączenia i konfiguracji interfejsów sieciowych, wprowadzenia użytkowników i nadania praw, instalacja preferowanych narzędzi [32], [66], [60].

Podstawą oprogramowania bram w przyjętej implementacji prototypu jest narzędzie Node-RED wymagające zainstalowanego środowiska Node.js. W przypadku obu obrazów systemów oprogramowanie to jest dostępne domyślnie. Dla RaspberryPi warto

rozważyć jego natychmiastową aktualizację. W przypadku IOT2020 aktualizacja może być utrudniona, ze względu na ograniczone wsparcie Node.js dla procesora Intel Quark x1000. W tym przypadku najlepiej jest śledzić aktualności na forum producenta [67]. Serwer Node-RED należy uruchomić zgodnie z dokumentacją. Domyślny port, na którym nasłuchuje serwer to 1880. Po wpisaniu w przeglądarce lokalnego adresu i portu w oknie pojawia się widok aplikacji do zarządzania serwerem Node-RED. Aplikacja umożliwia tworzenie i wdrażanie programów, dodawanie bibliotek, importowanie i eksportowanie projektu. Instalację bibliotek (w szczególności własnych, lokalnych węzłów) umożliwia także menadżer pakietów `npm`. W zależności od integrowanych interfejsów i protokołów konieczne jest ustawienie określonych parametrów połączeń, np. adres serwera pośrednika (brokera) w przypadku protokołu MQTT [54].

Dalsze czynności polegają na rozwoju i utrzymaniu bram w zależności od wymagań biznesowych. Działania te sprowadzają się m. in. do dołączania aparatury pomiarowej dostarczającej dane z wykorzystaniem dostępnych interfejsów, implementacji logiki przetwarzania brzegowego w ramach Node-RED, instalacji dodatkowego oprogramowania (w przypadku wykonanego prototypu na RaspberryPi zainstalowano przykładowo serwer pośrednika MQTT — Mosquitto [47]).

5.1.2 Konfiguracja usług w chmurze

Każda chmura posiada nieco inny sposób konfiguracji i model zarządzania. Czynności administracyjne, które należy przeprowadzić, żeby korzystać z usług chmurowych, to zwykle: rejestracja konta organizacji i użytkowników, wybór modelu rozliczeniowego (subskrypcji), konfiguracja i wzajemna integracja zasobów, które można postrzegać jako instancje usług. Przykładowe zasoby to bazy danych, funkcje bezserwerowe, maszyny wirtualne, itp.

Na potrzeby prototypu postanowiono użyć chmury Azure wraz z usługami: IoT Hub, Blob Storage, Functions. Po założeniu konta i uzyskaniu dostępu do portalu należy stworzyć grupę zasobów i dołączyć do niej nowy zasób w ramach każdej z wymienionych usług. Konfiguracja zasobu IoT Hub polega m. in. na zarejestrowaniu urządzeń (w tym przypadku bram IIoT). W celu skorzystania z usługi Blob Storage należy stworzyć zasób typu Storage Account i zdefiniować kontenery na dane zgodnie z zapotrzebowaniem. Przechodząc do wybranego kontenera można przeglądać składowane dane, które są zorganizowane w strukturę folderów i plików. Użycie usługi Azure Functions wymaga stworzenia zasobu typu Function App. Nowe funkcje można implementować przez panel zasobu Function App za pośrednictwem edytora w przeglądarce (tylko dla języka JavaScript). Dla większej wygody kod źródłowy można rozwijać lokalnie w jednym ze wspieranych języków programowania i przesyłać do chmury z wykorzystaniem narzędzia Function Core Tools. Żeby umożliwić wyzwalanie funkcji przez wiadomości (dane) zbierane w usłudze IoT Hub, a także zapis danych do bazy

Blob Storage, należy w ustawieniach IoT Hub skonfigurować trasowanie (integrację zasobów) przez zakładkę o nazwie Message Routing [5], [7], [4].

W implementacji projektu postanowiono także wykorzystać bazę InfluxDB osadzoną w chmurze InfluxDB Cloud. Po założeniu konta w usłudze i wybraniu modelu rozliczeniowego można od razu przejść do zarządzania zainstalowaną i uruchomioną już bazą za pośrednictwem aplikacji przeglądarkowej. Panel umożliwia m. in. definiowanie kontenerów na dane, edycję ustawień, realizację zapytań do bazy wraz z podstawową wizualizacją wyników [29].

5.1.3 Instalacja i konfiguracja narzędzia do wizualizacji

Na rynku dostępnych jest wiele darmowych i płatnych narzędzi do wizualizacji danych. Sposób ich instalacji i konfiguracji jest zróżnicowany. Mogą to być mianowicie aplikacje przeznaczone do instalacji lokalnej na komputerze osobistym, statyczne albo dynamiczne aplikacje internetowe (ang. *Static Web Applications / Single Page Applications*) z oprogramowaniem serwera (ang. *backend*), a także aplikacje na platformy mobilne i urządzenia wbudowane.

Do realizacji prototypu wybrano aplikację Grafana, jako że jest to darmowe, modularne (łatwe w rozbudowie) narzędzie, które domyślnie wspiera bazę InfluxDB. Aplikację postanowiono zainstalować lokalnie na komputerze osobistym z systemem Linux pobrawszy pakiet instalacyjny ze strony producenta. Alternatywnie Grafana może być zainstalowana na maszynie wirtualnej w chmurze albo można skorzystać bezpośrednio z płatnej usługi chmurowej producenta o nazwie Grafana Cloud (analogicznie jak w przypadku InfluxDB Cloud). Sposób użytkowania aplikacji jest podobny do Node-RED. Działający w tle serwer udostępnia aplikację przeglądarkową na domyślnym porcie 3000, w ramach której można konfigurować użytkowników, dodawać źródła danych i tworzyć panele wizualizacji. Żeby móc odczytywać dane z bazy InfluxDB należy w ustawieniach dodać źródło danych typu InfluxDB, podać adres URL bazy, nazwę organizacji i kontenera, a także wpisać znacznik (ang. *token*), który można wygenerować w panelu zarządzania bazą InfluxDB. Znacznik to specjalny, poufny ciąg znaków, który umożliwia klientom (aplikacjom i/lub użytkownikom) uwierzytelnianie i autoryzację [26], [29].

5.1.4 Integracja bram IIoT z usługami chmurowymi

W celu połączenia urządzeń z usługami chmurowymi należy w pierwszej kolejności dokonać ich rejestracji w panelu zarządzania chmurą, a następnie zaimplementować wymagany protokół po stronie urządzenia lub skorzystać z dedykowanej biblioteki będącej wygodną w użyciu fasadą dla protokołu. Bibliotekę należy odpowiednio skonfigurować, co odbywa się zwykle przez określenie parametrów połączenia: adresu IP albo URL,

portu, danych uwierzytelniających, itd.

Użyte na potrzeby prototypu oprogramowanie bram IIoT w postaci Node-RED posiada w swoim katalogu darmowe biblioteki umożliwiające połączenie zarówno z usługą Azure IoT Hub (`node-red-contrib-azure-iot-hub`), jak i bazą InfluxDB osadzoną w chmurze (`node-red-contrib-influxdb`). Węzeł IoT Hub wymaga dostarczenia danych połączenia w postaci nazwy hosta, identyfikatora urządzenia i klucza dostępu, które można znaleźć w ustawieniach zarejestrowanego urządzenia w portalu Azure. Węzeł służący do połączenia z InfluxDB Cloud wymaga dostarczenia tych samych danych, co w przypadku integracji aplikacji Grafana (patrz: rozdział 5.1.3) [55], [29], [5].

5.2 Kategorie użytkowników

Na diagramie przypadków użycia (patrz: rozdział 3.3, rys. 4) przedstawiono użytkowników systemu wraz z możliwym zakresem ich działań. W systemie wyróżniono następujące kategorie użytkowników:

- Administratorzy — Są to pracownicy zarządzający infrastrukturą systemu. Do ich zadań należy instalacja, konfiguracja i utrzymanie urządzeń, zarządzanie sieciami komputerowymi, dbanie o bezpieczeństwo (monitorowanie, wykrywanie awarii, aktualizacja oprogramowania i certyfikatów), zarządzanie zasobami w chmurze i monitorowanie kosztów, zarządzanie użytkownikami, administracja urządzeniami klienckimi. W zależności od skali systemu administratorzy mogą pełnić obowiązki w obrębie całego systemu lub poszczególnych jego warstw i komponentów;
- Programiści i testerzy — Implementują i testują logikę przetwarzania danych osadzaną na urządzeniach warstwy brzegowej oraz w chmurze;
- Analitycy, pracownicy obsługujący proces przemysłowy i inni — Kategoria odnosi się do wszystkich użytkowników, którzy korzystają z wników zbierania, przetwarzania i wizualizacji danych, a także uczestniczą w rozwoju logiki przetwarzania danych.

5.3 Kwestie bezpieczeństwa

W ramach podrozdziału rozpatrzono zagadnienia niezawodności, ograniczenia dostępu i szyfrowania komunikacji w systemie.

5.3.1 Niezawodność

Problematyka niezawodności systemu posiada dwa aspekty — sprzętowy i programowy. Żeby zapewnić poprawne działanie w warstwie sprzętowej, warto w pierwszej

kolejności dokładnie przemyśleć dobór urządzeń. Ich specyfikację należy zestawić z warunkami, w których potencjalnie przyjdzie im pracować. W otoczeniu przemysłowym mogą występować bowiem różne zagrożenia: zakłócenia elektromagnetyczne, mechaniczne i chemiczne mające realny wpływ na pracę urządzeń elektronicznych. Ponadto dla uzyskania większego poziomu bezawaryjności można rozważyć instalację źródeł awaryjnego zasilania, wprowadzenie urządzeń nadmiarowych (redundantnych) przejmujących zadania uszkodzonych urządzeń, a także układów typu *watchdog* wykrywających awarie [23].

Drogą do osiągnięcia niezawodności w warstwie programowej jest przede wszystkim sporządzanie odpowiedniej liczby testów automatycznych oraz przeprowadzenie testów manualnych na poziomie jednostkowym, integracyjnym i systemowym. W tym przypadku również należy również rozważyć użycie programów i/lub urządzeń typu *watchdog* wykrywających awarie i wyzwalające przewidziane reakcje systemu — na przykład ponowne uruchomienie urządzeń, próba ponownego nawiązania połączenia, itp.

5.3.2 Ograniczenie dostępu

Kwestie zapewnienia dostępu do systemu wyłącznie osobom i aplikacjom do tego uprawnionym można również rozpatrywać na płaszczyźnie sprzętowej i programowej. Dla warstwy sprzętowej należy rozważyć wydzielenie specjalnych pomieszczeń dla używanych urządzeń, do których dostęp będzie kontrolowany i monitorowany przez wyznaczony do tego personel i odpowiednie systemy monitoringu. Ograniczenie dostępu do płaszczyzny programowej polega przede wszystkim na zastosowaniu metod uwierzytelniania i autoryzacji oraz wykorzystaniu sieciowych mechanizmów zabezpieczeń (zapora ogniowa, oprogramowanie antywirusowe, itd.). Uwierzytelnianie i autoryzacja z wykorzystaniem identyfikatorów i haseł powinny znaleźć zastosowanie zarówno w warstwie brzegowej, jak i biznesowej systemu. Wymagane jest, aby hasła posiadały odpowiedni stopień skomplikowania i były często aktualizowane [37].

Uwierzytelnianiu i autoryzacji w obrębie zrealizowanego prototypu podlega dostęp do: systemów operacyjnych bram IIoT, aplikacji Node-RED, portali zarządzania chmurą Azure i InfluxDB Cloud, komputerów klienckich i aplikacji Grafana. Komponenty integrujące się z chmurą (Node-RED i Grafana) uzyskują dostęp na podstawie wspomnianego już znacznika — specjalnego ciągu znaków pełniącego rolę klucza. Znaczniki można w każdej chwili dezaktywować. Analogicznie przedstawia się sytuacja z kontami użytkowników w chmurach. Nieco więcej wysiłku może wymagać administracja kontami na poziomie urządzeń przedsiębiorstwa. W niektórych przypadkach możliwe jest wykorzystanie mechanizmu pojedynczego logowania (ang. *Single Sign-On*, w skrócie: SSO) zintegrowanego z chmurą, np. w ramach usług Azure Active Directory czy AWS Directory Service. Dla podniesienia poziomu bezpieczeństwa należy

także rozważyć wprowadzenie mechanizmów ochrony dostępu na poziomie protokołów używanych do komunikacji w warstwie brzegowej, np. ustawienie hasła w serwerze pośredniczącym MQTT.

5.3.3 Szyfrowanie komunikacji

W celu zwiększenia ochrony poufności danych przesyłanych za pośrednictwem sieci komputerowych należy wdrożyć odpowiednie mechanizmy szyfrowania. Jest to aspekt szczególnie ważny, gdy ma miejsce wymiana danych w ramach sieci publicznej — Internetu. W kontekście realizowanego projektu dotyczy to połączenia bram IIoT z usługami chmurowymi. Wśród dostępnych rozwiązań wyróżnia się m. in. użycie protokołu kryptograficznego TLS w połączeniu ze standardowymi protokołami HTTP, MQTT, itd. Przydatne może okazać się również wykorzystanie mechanizmu tunelowania w postaci usługi VPN (ang. *Virtual Private Network*). Powinno się także rozważyć szyfrowanie komunikacji na poziomie sieci warstwy brzegowej. W tym przypadku można jednak napotkać pewne problemy, gdyż nie wszystkie urządzenia są w stanie implementować algorytmy kryptograficzne ze względu na ograniczone zasoby, jak również nie wszystkie protokoły posiadają specyfikację odnoszącą się do zagadnień bezpieczeństwa [23], [35].

Wykorzystane na potrzeby prototypu chmury Azure i InfluxDB narzucają szyfrowanie komunikacji z wykorzystaniem protokołu TLS. Narzędzia Node-RED i Grafana umożliwiają konfigurację certyfikatów SSL i tym samym użycie protokołu HTTPS do komunikacji z serwerem aplikacji w ramach przeglądarki. Specyfikacje protokołów MQTT, CoAP i Modbus/TCP również przewidują użycie protokołu TLS albo DTLS (w przypadku, gdy protokołem warstwy transportowej jest UDP) [48], [45], [16].

6 Weryfikacja i walidacja

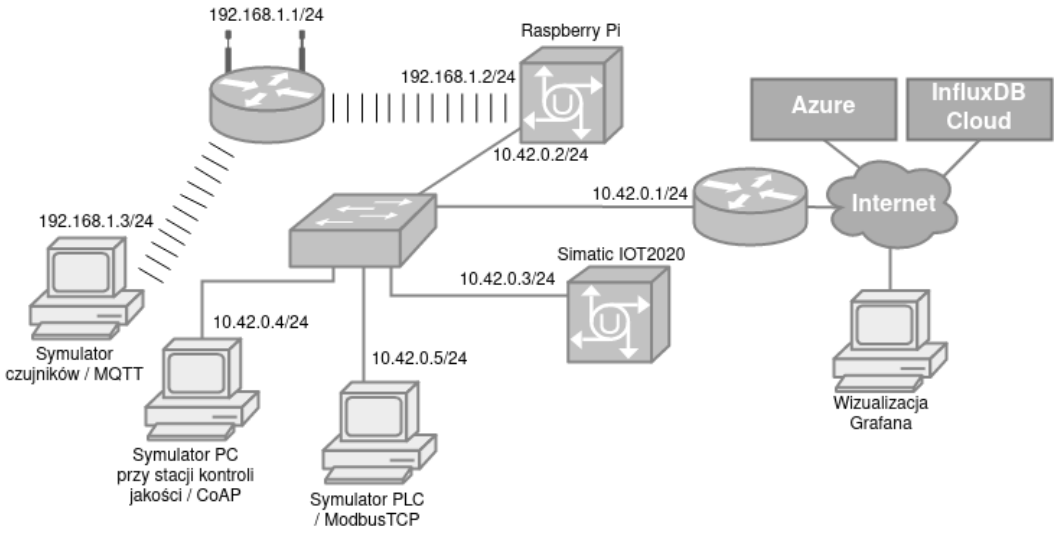
W niniejszym rozdziale opisano dwa rodzaje testów przeprowadzonych w celu weryfikacji i walidacji działania systemu: testy jednostkowe oraz systemowe.

6.1 Testy jednostkowe

W celu weryfikacji poprawności działania autorskich komponentów, tj. funkcji bezserwerowych uruchamianych w chmurze Azure oraz Bufora Brzegowego i funkcji przetwarzających Node-RED zaimplementowano testy jednostkowe. Żeby uruchomić i zautomatyzować przeprowadzanie testów posłużono się darmowymi narzędziami: dedykowanym modułem `node-red-node-test-helper` w przypadku Node-RED oraz narzędziem Jest w przypadku funkcji Azure implementowanych w języku JavaScript.

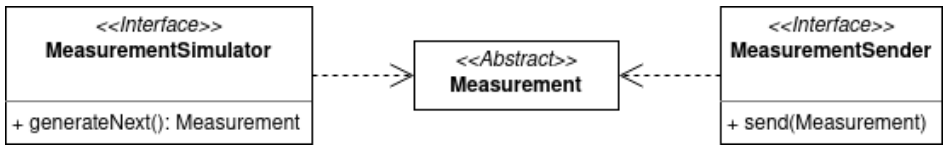
6.2 Testy systemowe

Na potrzeby weryfikacji poprawności działania całego systemu oraz sprawdzenia, czy spełnia on zdefiniowane wymagania (walidacja), uruchomiono w warunkach domowych zrealizowany prototyp systemu o konfiguracji przedstawionej na rys. 18. Zbudowano sieć komputerową na bazie Ethernet i WiFi składającą się z dwóch podsieci: 10.42.0.0/24 przeznaczonej dla urządzeń przewodowych oraz 192.168.1.0/24, w ramach której komunikują się urządzenia bezprzewodowe. Bramy RaspberryPi oraz Simatic IOT2020 nie pełnią w tej konstelacji roli bram w sensie sieci, lecz w sensie funkcjonalnym bram IIoT, czyli gromadzenia danych, przetwarzania brzegowego, translacji protokołów i przekazywania danych do chmury. Właściwą bramą sieciową udostępniającą połączenie z Internetem jest w systemie urządzenie o adresie 10.42.0.1.



Rysunek 18: Schemat ideowy systemu uruchomionego na potrzeby testów systemowych

Na komputerach o adresach 192.168.1.3, 10.42.0.4 i 10.42.0.5 zainstalowano i uruchomiono autorskie oprogramowanie symulujące urządzenia pomiarowe, które generuje kolejne dane i przesyła je do bram z wykorzystaniem odpowiednich protokołów. Program napisano w języku Java. Jego stosunkowo prostą architekturę przedstawiono na rys. 19. Rozszerzanie działania symulatora odbywa się przez implementację interfejsów i klasy abstrakcyjnej znajdujących się na diagramie. Implementacje interfejsu `MeasurementSimulator` mają za zadanie generować kolejne wartości pomiarów w postaci obiektów typu `Measurement`. Otrzymane dane są wysyłane do odbiorcy z wykorzystaniem implementacji interfejsu `MeasurementSender`. Do implementacji protokołów wykorzystano darmowe biblioteki: Eclipse Paho (MQTT), Apache PLC4J (Modbus/TCP), Eclipse Californium (CoAP).



Rysunek 19: Diagram przedstawiający architekturę oprogramowania symulatora danych pomiarowych

Implementacja symulatora parametrów powietrza dla każdego z trzech pomieszczeń pozwala na zdefiniowanie listy kroków symulacji, które są wykonywane przez określoną liczbę wywołań metody `generateNext`. Dla każdego kroku należy wybrać jeden z trybów pracy:

- `OSCILLATE` — oscylacja (przyrosty pseudolosowe, w ograniczonym przedziale),
- `DROP` — spadek (o pseudolosową wartość z ograniczonego przedziału),
- `GROW` — wzrost (o pseudolosową wartość z ograniczonego przedziału).

W ten sposób możliwe jest definiowanie przebiegów o konkretnym kształcie, który jednocześnie sprawia wrażenie naturalnego ze względu na pewien stopień losowości. Wygenerowane wartości temperatury i wilgotności są przesyłane cyklicznie do serwera pośredniczącego MQTT znajdującego się pod adresem 192.168.1.2:1883 (RaspberryPi).

Symulator danych dotyczących jakości wytworzonych produktów przyjmuje jako parametr oczekiwaną wartość parametru FTQ, która określa prawdopodobieństwo wystąpienia produktu spełniającego wymagania jakościowe. Wartość wskaźnika zbliża się więc w każdym kroku symulacji do zadanej. Wygenerowane dane są przesyłane cyklicznie do serwera CoAP znajdującego się pod adresem 192.168.1.2:5683 (RaspberryPi.)

Symulator stanu przepływomierzy i poziomu cieczy w zbiorniku głównym generuje w każdym kroku symulacji nowy stan dla przepływomierzy. Od poziomu cieczy w zbiorniku głównym odejmowana jest następnie wartość ostatniego przyrostu zużycia cieczy w każdej ze stref. Zbiornik jest automatycznie uzupełniany po przekroczeniu

zdefiniowanej wartości minimalnej. Symulator pełni zatem jednocześnie funkcję atrapy sterownika PLC. Cztery wygenerowane wartości są zapisywane cyklicznie do rejestrów węzła podrzędnego Modbus/TCP nasłuchującego pod adresem 10.42.0.5:502 (na tym samym komputerze, co symulator). Dane z atrapy PLC są odczytywane przez bramę Simatic IOT2020 będącą węzłem nadrzędnym, znajdującą się pod adresem 10.42.0.3.

Testy systemu polegały na obserwacji wizualizacji w narzędziu Grafana i weryfikacji, czy otrzymane wyniki są zgodne z wartościami wygenerowanymi przez symulator. W pierwszej kolejności skonfigurowano symulatory do wygenerowania wartości stałych, zapisanych „na sztywno“, żeby jednoznacznie stwierdzić poprawność. Następnie uruchomiono opisane wcześniej przebiegi symulacji na czas trzech godzin. Oczekiwano następujących rezultatów:

- określony kształt przebiegu wartości temperatury i wilgoci dla każdej ze stref (odpowiednio: prostokątny, trapezoidalny i trójkątny),
- zbliżanie się wartości parametrów FTQ do zadanych,
- stopniowy wzrost wskazań stanu przepływomierzy i adekwatny poziom cieczy w zbiorniku głównym.

Obok obserwacji wizualizacji dokonano także przeglądu danych zapisywanych w bazach Azure Blob Storage i InfluxDB.

Opisane testy przeprowadzano wielokrotnie, a z każdą iteracją wykrywano pewne błędy wynikające z:

- błędnej konfiguracji połączeń sieciowych (nieprawidłowy adres IP, maska, brama sieciowa),
- błędów w implementacji logiki przetwarzania w warstwie brzegowej,
- błędnej konfiguracji węzłów odpowiedzialnych za integrację protokołów i komunikację z chmurą,
- niepoprawnej konfiguracji trasowania wiadomości z Azure IoT Hub do innych usług,
- błędów w logice funkcji bezserwerowych,
- błędów w składni języka zapytań Flux,
- niepoprawnej konfiguracji komponentów w aplikacji Grafana.

W ostateczności wszystkie wymienione usterki zostały poprawione i więcej nieprawidłowości nie stwierdzono.

Przeprowadzone testy pozwoliły na weryfikację działania systemu, a także na walidację — sprawdzenie, czy spełnione są zdefiniowane wymagania. Prototypowa konfiguracja realizuje zadania zbierania, przetwarzania, składowania, udostępniania i wizu-

alizacji danych pomiarowych. Wykorzystano określone rozwiązania IoT i usługi chmurowe. Użycie bramy Simatic IOT2020 i protokołu Modbus pozwoliło na dostosowanie do środowiska przemysłowego w podstawowym zakresie. Oprogramowanie Node-RED realizuje wymaganie ogólności, jako że umożliwia integrację różnych urządzeń i protokołów. Usługi chmurowe pozwalają na stosunkowo proste skalowanie zasobów. Zapewniono także podstawowy poziom bezpieczeństwa — dostęp do komponentów systemu wymaga uwierzytelnienia z użyciem hasła, zaś komunikacja za pośrednictwem Internetu odbywa się z wykorzystaniem mechanizmów szyfrowania.

7 Podsumowanie

W ramach projektu podjęto zadanie wypracowania ogólnej koncepcji systemu wraz z przykładową implementacją w postaci prototypu, którego przeznaczeniem jest zbieranie danych pomiarowych w środowisku przemysłowym, a także ich składowanie, przetwarzanie, udostępnianie i wizualizacja. Przyjęto założenie, że należy w tym celu wykorzystać możliwości oferowane przez IoT i chmurę obliczeniową. W rozdziale 4.1 przedstawiono uniwersalny model architektury systemu, który opracowano na podstawie spotykanego w literaturze trójwarstwowego wzorca systemów IIoT oraz referencyjnego modelu architektury RAMI 4.0 (patrz: rozdział 2.4). Zaproponowany model posiada strukturę komponentową podzieloną na trzy grupy ze względu na miejsce wdrożenia: środowisko przemysłowe (bramy IIoT), chmura i urządzenia klienckie. Każdy z komponentów jest odpowiedzialny za określony zakres funkcjonalny, może być swobodnie wymieniany, modyfikowany i posiadać różne implementacje. Dla poszczególnych elementów architektury przedstawiono potencjalne realizacje na podstawie przeglądu dostępnych urządzeń, rozwiązań technicznych i narzędzi informatycznych.

Żeby umożliwić weryfikację działania systemu, na potrzeby prototypu zdefiniowano konkretne scenariusze biznesowe, które podlegały implementacji: pomiar parametrów powietrza (temperatury i wilgotności), wyznaczanie wskaźnika jakości produkcji na podstawie wyników ewaluacji produktów, wizualizacja stanu zużycia chemii przemysłowej. Wykorzystano urządzenia RaspberryPi i Siemens Simatic IOT2020, które pełnią rolę bram IIoT. Zbieranie danych pomiarowych w systemie odbywa się z wykorzystaniem sieci Ethernet oraz WiFi. Do komunikacji użyto charakterystycznych dla IoT protokołów MQTT i CoAP, a także popularnego w przemyśle protokołu Modbus. Integrację różnych interfejsów i protokołów umożliwia darmowe oprogramowanie Node-RED zainstalowane na bramach IIoT. Realizuje ono przetwarzanie brzegowe i przesył danych do chmury za pośrednictwem Internetu. Pozwala na elastyczną modyfikację i rozbudowę logiki przetwarzania. W warstwie biznesowej systemu użyto chmury Microsoft Azure. W celu gromadzenia danych przesyłanych z bram zastosowano usługę IoT Hub, zaś dalsze przetwarzanie, składowanie i udostępnianie danych realizują usługi

Functions i Blob Storage oraz baza danych szeregów czasowych InfluxDB osadzona w chmurze InfluxDB Cloud. Do wizualizacji danych posłużono się darmowym narzędziem Grafana, które integruje się z bazą InfluxDB i umożliwia tworzenie wykresów, diagramów, tabel, wskaźników, jak również własnych komponentów graficznych.

Wśród napotkanych wyzwań projektowych należy przede wszystkim wymienić konieczność połączenia wiedzy ze stosunkowo szerokiej dziedziny, którą stanowią: urządzenia wbudowane, systemy pomiarowe, sieci komputerowe, przemysłowe systemy informatyczne, aplikacje internetowe, IoT i chmura. Zasadniczym problemem technicznym okazała się być integracja komponentów systemu, co można wnioskować na podstawie wychwyconych błędów wymienionych w rozdziale 6.2. Dobrą decyzją było prototypownie z wykorzystaniem mikrokomputera RaspberryPi w pierwszej kolejności. Choć nie jest to urządzenie dedykowane dla przemysłu, to jednak stosunkowo duża moc obliczeniowa i dostępne wsparcie programistyczne przyniosły korzyści w warunkach testowych, rozwojowych. Brama IOT2020 jest przeznaczona do zastosowań przemysłowych, lecz nie posiada tak dużej mocy obliczeniowej oraz wsparcia dla najnowszych wersji oprogramowania (m. in. Node.js), co znacząco wydłużyło czas implementacji i wdrażania, jak również wymusiło zrezygnowanie z niektórych bibliotek Node-RED. Zaobserwowane korzyści wynikające z zastosowania chmury to przede wszystkim brak konieczności utrzymywania infrastruktury informatycznej, która może być swobodnie zmieniana, rozbudowywana za pośrednictwem portalu.

Komponentowy model architektury systemu i wykorzystanie uniwersalnych technologii stwarza możliwości do dalszego rozwoju. Jedną z nich jest dołączenie do prototypu prawdziwych urządzeń aparatury przemysłowej, jak i IoT. Jeżeli wybrane urządzenia warstwy brzegowej posiadają techniczną zdolność do pracy w czasie rzeczywistym (np. Simatic IOT2020, BeagleBone), można spróbować zintegrować je z działaniem lokalnego systemu automatyki. Przed wdrożeniem systemu w miejscu docelowym koniecznie należy przeprowadzić testy w środowisku możliwie najlepiej odwzorowującym środowisko produkcyjne. Warto rozpatrzyć także zagadnienia niezawodności i bezpieczeństwa: rozbudować mechanizmy szyfrowania danych i ograniczenia dostępu w warstwie brzegowej, wprowadzić redundancję sprzętową, układy typu *watchdog*, ewentualnie stworzyć klaster urządzeń przetwarzających (bram IIoT) wraz z równoważeniem obciążenia. Wykorzystując dostępne usługi chmurowe można zrealizować również bardziej zaawansowane przetwarzanie danych: uczenie maszynowe i analizę *big data*. Dane składowane w chmurze mogą być ponadto udostępniane na rzecz aplikacji „szytych na miarę“, które pozwalają na realizację w zasadzie dowolnej funkcjonalności. Dostępne rozwiązania technologiczne umożliwiają także przepływ informacji w kierunku od chmury do warstwy brzegowej, co może być pożyteczne w wielu scenariuszach, jednak jednocześnie podnosi poziom zagrożenia, jako że systemy zewnętrzne mogłyby mieć wpływ na działanie układów sterowania, zaś sieć publiczna ułatwia dostęp dla potencjalnych

przestępców. Implementacji takiego modelu należy zatem dokonywać bardzo rozważnie. Przedstawiona architektura opiera się na koncepcji bram IIoT jako pośrednika komunikacji. Istnieje podejście alternatywne, w myśl którego przetwarzanie brzegowe odbywa się w chmurze, zaś urządzenia komunikują się z nią bezpośrednio. Trzeba jednak zwrócić uwagę na fakt, że nie wszystkie urządzenia są w stanie implementować kompletne stosy protokołów. Brak elementu pośredniczącego w postaci bramy utrudnia sprawowanie kontroli nad systemem. Stąd dla przemysłu można rozważyć rozwiązanie co najwyżej o charakterze hybrydowym [56].

Bibliografia

[1] Davide Ancona, Giorgio Delzanno, Lorenzo Benvenuto i Gianluca Gambari. „Flow Programming: A Flexible way to bring the Internet of Things into the Lab". W: *Session 2: Adaptation and Personalization in Computer Science Education (APCSE 2020)* (2020).

[2] *AWS to Azure services comparison*. Microsoft. URL: `https://docs.microsoft.com/en-us/azure/architecture/aws-professional/services#ai-and-machine-learning` (term. wiz. 18. 12. 2020).

[3] *Azure for Students*. Microsoft Azure. URL: `https://azure.microsoft.com/en-us/free/students/` (term. wiz. 20. 12. 2020).

[4] *Azure Functions Documentation*. Microsoft Azure. URL: `https://docs.microsoft.com/en-us/azure/azure-functions/` (term. wiz. 11. 01. 2021).

[5] *Azure IoT Hub Documentation*. Microsoft Azure. URL: `https://docs.microsoft.com/en-us/azure/iot-hub/` (term. wiz. 11. 01. 2021).

[6] *Azure products*. Microsoft Azure. URL: `https://azure.microsoft.com/en-us/services/` (term. wiz. 15. 11. 2020).

[7] *Azure Storage Documentation*. Microsoft Azure. URL: `https://docs.microsoft.com/en-us/azure/storage/` (term. wiz. 11. 01. 2021).

[8] *BeagleBone*. BeagleBoard.org Foundation. URL: `http://beagleboard.org/` (term. wiz. 15. 11. 2020).

[9] *Benefits of a Cloud Platform in the IoT*. AVSystem. URL: `https://www.avsystem.com/blog/iot-cloud-platform/` (term. wiz. 15. 11. 2020).

[10] *Blob service REST API*. Microsoft. URL: `https://docs.microsoft.com/en-us/rest/api/storageservices/blob-service-rest-api` (term. wiz. 20. 12. 2020).

[11] Emilie Bonnetto, Bernard Yannou, Gwenola Bertoluci, Vincent Boly i Jorge Alvarez. „A categorization of customer concerns for an OT front-end of innovation process in an IT/OT convergence context". W: *International Design Conference* (2016).

[12] Claudio Botta, Leonardo Pierangelini i Luca Vollero. „IoT Gateways for Industrial and Medical Applications: Architecture and Performance Assessment". W: *IEEE International Workshop on Metrology for Industry 4.0 & IoT* (2020).

[13] Ren-gen Huang Chang-le Zhong Zhen Zhu. „Study on the IOT Architecture and Gateway Technology". W: *14th International Symposium on Distributed Computing and Applications for Business Engineering and Science* (2015).

[14] Hao Chen i Heng Li Xueqin Jia. „A brief introduction to the IoT gateway". W: *IET International Conference on Communication Technology and Application (ICCTA 2011)* (2011).

[15] *Cloud Products*. Amazon Web Services. URL: https://aws.amazon.com/products/?nc2=h_ql_prod_fs_f (term. wiz. 15. 11. 2020).

[16] *CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets*. Internet Engineering Task Force (IETF). URL: https://tools.ietf.org/id/draft-ietf-core-coap-tcp-tls-11.html (term. wiz. 11. 01. 2021).

[17] *Crosser*. Crosser. URL: https://crosser.io/contact/ (term. wiz. 18. 12. 2020).

[18] Maurizio Di Paolo Emilio. *Data Acquisition Systems. From Fundamentals to Applied Design*. Springer, 2013.

[19] *Difference Between Raspberry Pi 3 and BeagleBone Black*. EDUCBA. URL: https://www.educba.com/raspberry-pi-3-vs-beaglebone-black/ (term. wiz. 15. 11. 2020).

[20] Vlad M.Trifa Dominique D.Guinard. *Internet rzeczy. Budowa sieci z wykorzystaniem technologii webowych i Raspberry Pi*. Helion, 2017.

[21] Max Felser. „Real Time Ethernet: standardization and implementations". W: *IEEE International Symposium on Industrial Electronics* (2010).

[22] Francisco Fraile, Raquel Sanchis i Raul Poler Angel Ortiz. „Reference Models for Digital Manufacturing Platforms". W: *Applied Sciences 9(20):4433* (2019).

[23] Piotr Gaj. *Wybrane zagadnienia projektowania systemów informatyki przemysłowej*. Studia Informatica : quarterly ; vol. 37, nr 4B (128). Silesian University of Technology Press, 2016.

[24] *Gartner Magic Quadrant for Cloud Infrastructure and Platform Services*. Gartner. 2020. URL: https://www.gartner.com/en/documents/3989743/magic-quadrant-for-cloud-infrastructure-and-platform-ser (term. wiz. 18. 12. 2020).

[25] Mariusz Grabowski i Agnieszka Zając. „Dane, informacja, wiedza – próba definicji". W: *Zeszyty Naukowe nr 798 Uniwersytetu Ekonomicznego w Krakowie* (2009).

[26] *Grafana documentation*. Grafana Labs. URL: https://grafana.com/docs/grafana/latest/ (term. wiz. 11. 01. 2021).

[27] Martin Hankel i Bosch Rexroth. „The Reference Architectural Model Industrie 4.0 (RAMI 4.0)". W: *ZVEI Industrie 4.0* (2015).

[28] *InfluxDB Cloud*. InfluxData Inc. URL: https://www.influxdata.com/products/influxdb-cloud/ (term. wiz. 20. 12. 2020).

[29] *InfluxDB Documentation.* InfluxData Inc. URL: https://docs.influxdata.com/influxdb/v2.0/ (term. wiz. 20. 12. 2020).

[30] *Information science.* Encyclopædia Britannica. URL: https://www.britannica.com/science/information-science (term. wiz. 01. 12. 2020).

[31] *IoT Gateways. Market Map.* Postscapes. URL: https://www.postscapes.com/iot-gateways/ (term. wiz. 18. 12. 2020).

[32] *IOT2000 Starter Guide and useful information.* Siemens AG. URL: https://support.industry.siemens.com/tf//WW/en/posts/iot2000-starter-guide-and-useful-information/155652?page=0&pageSize=10 (term. wiz. 11. 01. 2021).

[33] *IXON. All-in-one Industrial IoT solution.* IXON. URL: https://www.ixon.cloud/all-in-one-industrial-iot-solution (term. wiz. 15. 11. 2020).

[34] Sabina Jeschke, Christian Brecher, Houbing Song i Danda B.Rawat. *Industrial Internet of Things. Cybermanufacturing Systems.* Springer, 2017.

[35] Aleksandr Kapitonov, Dmitrii Dobriborsci, Igor Pantiukhin, Valerii Chernov, Raivo Sell, Rim Puks, Mallor Kingsepp, Agris Nikitenko, Karlis Berkolds, Anete Vagale, Rudolfs Rumba, Piotr Czekalski, Krzysztof Tokarz, Oleg Antemijczuk, Jarosław Paduch, Raivo Sell, Salvatore, Rustem Dautov, Riccardo Di Pietro, Antonino Longo Minnolo, Blanka Czekalska, Małgorzata Wiktorczyk i Ritankar Sahu. *Introduction to the IoT.* RTU Press, 2019.

[36] Aditi khandelwal, Ishita Agrawal, Malaserene I., S.Sankar Ganesh i Rajeev Karothia. „Design and implementation of an industrial gateway: Bridging sensor networks into IoT". W: *International conference on Electronics, Communication and Aerospace Technology (ICECA)* (2019).

[37] J. Kirupakar i S. Mercy Shalinie. „Situation Aware Intrusion Detection System Design for Industrial IoT Gateways". W: *Second International Conference on Computational Intelligence in Data Science* (2019).

[38] Oliver Kleineberg i Axel Schneider. *Time-Sensitive Networking For Dummies.* John Wiley & Sons, Inc, 2018.

[39] Viacheslav Kulik i Ruslan Kirichek. „The Heterogeneous Gateways in the Industrial Internet of Things". W: *10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)* (2018).

[40] Andrzej Kwiecień. *Analiza przepływu informacji w komputerowych sieciach przemysłowych.* Studia Informatica, 1642-0489, Zeszyty Naukowe Politechniki Śląskiej. Informatyka. Wydawnictwo Politechniki Śląskiej, 2002.

[41] Roman Kwiecień. *Komputerowe systemy automatyki przemysłowej.* Helion, 2013.

[42] Theo Lynn, John G. Mooney, Pierangelo Rosati i Grace Fox. *Measuring the Business Value ofCloud Computing*. PALGRAVE STUDIES IN DIGITAL BUSINESS AND ENABLING TECHNOLOGIES. Palgrave Macmillan, 2020.

[43] Zaigham Mahmood, Eldar Sultanow i Alina Chircu. *The Internet of Things in the Industrial Sector*. Computer Communications and Networks. Springer, 2019.

[44] *Mikrokontrolery STM32 - po które z nich warto sięgnąć?* Elektronika B2B. URL: https://elektronikab2b.pl/technika/50000-mikrokontrolery-stm32-po-ktore-z-nich-warto-siegnac (term. wiz. 15. 11. 2020).

[45] *MODBUS/TCP Security. Protocol Specification.* Schneider Electric USA, Inc. URL: https://modbus.org/docs/MB-TCP-Security-v21_2018-07-24.pdf (term. wiz. 11. 01. 2021).

[46] *Mosquitto*. Eclipse Foundation. URL: https://mosquitto.org/ (term. wiz. 19. 12. 2020).

[47] *Mosquitto Documentation*. Eclipse Foundation. URL: https://mosquitto.org/documentation/ (term. wiz. 11. 01. 2021).

[48] *mosquitto-tls man page*. Eclipse Foundation. URL: https://mosquitto.org/man/mosquitto-tls-7.html (term. wiz. 11. 01. 2021).

[49] *n8n.io*. n8n.io. URL: https://n8n.io/ (term. wiz. 18. 12. 2020).

[50] *Nazca 4.0*. APA Group. URL: https://www.apagroup.pl/en/nazca-4-0 (term. wiz. 15. 11. 2020).

[51] Galia Novakova Nedeltcheva i Elena Shoikova. „Models for Innovative IoT Ecosystems". W: *IEEE 5th World Forum on Internet of Things (WF-IoT)* (2019).

[52] Phuc Nguyen-Hoang i Phuoc Vo-Tan. „Development An Open-Source Industrial IoT Gateway". W: *19th International Symposium on Communications and Information Technologies (ISCIT)* (2019).

[53] *Node-RED*. OpenJS Foundation. URL: https://nodered.org/ (term. wiz. 18. 12. 2020).

[54] *Node-RED Documentation*. OpenJS Foundation. URL: https://nodered.org/docs/ (term. wiz. 11. 01. 2021).

[55] *Node-RED Library*. OpenJS Foundation. URL: https://flows.nodered.org/ (term. wiz. 11. 01. 2021).

[56] Deepak Priyadarshi i Ashutosh Behura. „Analysis of Different IoT Protocols for Heterogeneous Devices and Cloud Platform". W: *International Conference on Communication and Signal Processing* (2018).

[57] *PRU-ICSS Resources*. BeagleBoard.org Foundation. URL: http://beagleboard.org/pru (term. wiz. 15. 11. 2020).

[58] *Przetwarzanie bezserwerowe*. Microsoft Azure. URL: https://azure.microsoft.com/pl-pl/overview/serverless-computing/ (term. wiz. 18. 12. 2020).

[59] *Raspberry Pi*. Raspberry Pi Foundation. URL: https://www.raspberrypi.org/ (term. wiz. 15. 11. 2020).

[60] *Raspberry Pi Documentation*. Raspberry Pi Foundation. URL: https://www.raspberrypi.org/documentation/ (term. wiz. 11. 01. 2021).

[61] Ammar Rayes i Samer Salam. *Internet of Things From Hype to Reality. The Road to Digitization*. Springer, 2019.

[62] Margaret Rouse. *WSAN (wireless sensor and actuator network)*. TechTarget. URL: https://internetofthingsagenda.techtarget.com/definition/WSAN-wireless-sensor-and-actuator-network (term. wiz. 15. 11. 2020).

[63] *Sensmetrics*. Industrial IoT Solutions. URL: https://i-iotsolutions.com/sensemetrics/ (term. wiz. 15. 11. 2020).

[64] Feng Shu, Hanhua Lu i Yin Ding. „Novel Modbus Adaptation Method for IoT Gateway". W: *IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC 2019)* (2019).

[65] *Siemens Simatic IOT2000 Real-Time Profinet Driver*. Siemens. URL: https://support.industry.siemens.com/tf/WW/en/posts/profinet-driver-for-iot2000-released/200420?page=0&pageSize=10 (term. wiz. 19. 12. 2020).

[66] *SIEMENS SIMATIC IOT2020*. RS Components Sp. z o.o. URL: https://pl.rs-online.com/web/generalDisplay.html?id=siemens-simatic-iot2020&intcmp=PL-WEB-_-BP-PB3-_-Apr-17-_-siemens (term. wiz. 19. 12. 2020).

[67] *Simatic IOT2000 Support Forum*. Siemens AG. URL: https://support.industry.siemens.com/tf/ww/en/conf/60/ (term. wiz. 11. 01. 2021).

[68] Emiliano Sisinni, Abusayeed Saifullah, Song Han, Ulf Jennehag i Mikael Gidlund. „Industrial Internet of Things: Challenges, Opportunities and Directions". W: *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, VOL. 14, NO. 11* (2018).

[69] Vladimir Slamecka. *Information processing*. Encyclopædia Britannica. URL: https://www.britannica.com/technology/information-processing (term. wiz. 01. 12. 2020).

[70] *SoC*. TechTerms. URL: https://techterms.com/definition/soc (term. wiz. 15. 11. 2020).

[71] *STM32 32-bit Arm Cortex MCUs*. STMicroelectronics. URL: https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html (term. wiz. 15. 11. 2020).

[72] *System informatyczny*. Wydawnictwo Naukowe PWN. URL: https://encyklopedia.pwn.pl/haslo/system-informatyczny;3982203.html (term. wiz. 01. 12. 2020).

[73] *Time Sersies And Forecasting*. Encyclopædia Britannica. URL: https://www.britannica.com/science/statistics/Residual-analysis#ref367512 (term. wiz. 01. 12. 2020).

[74] *Total.js Flow*. Total Avengers. URL: https://www.totaljs.com/flow/ (term. wiz. 18. 12. 2020).

[75] Alex Vakaloudis i Christian O'Leary. „A framework for rapid integration of IoT Systems with industrial environments". W: *International Conference on Big Data and Internet of Things (BDIOT2017)* (2017).

[76] Ahmad Zainudin, Mohamad Fahmi Syaifudin i Nanang Syahroni. „Design and Implementation of Node Gateway with MQTT and CoAP Protocol for IoT Applications". W: *4th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)* (2019).

[77] Atefeh Zare i M. Tariq Iqbal. „Low-Cost ESP32, Raspberry Pi, Node-Red, and MQTT Protocol Based SCADA System". W: *IEEE* (2020).