

Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH



Instytut Telekomunikacji

Praca dyplomowa inżynierska

na kierunku Telekomunikacja
w specjalności Techniki Teleinformatyczne

Czujnik wideo z obsługą LoRaWAN do monitorowania zajętości urządzeń

Marcin Ruta

Numer albumu 303952

promotor
mgr inż. Marcin Golański

WARSZAWA 2024

Czujnik wideo z obsługą LoRaWAN do monitorowania zajętości urządzeń

Streszczenie. Praca ma charakter wdrożeniowy, opisuje projekt oraz implementację systemu służącego do wykrywania zajętości urządzeń na siłowni. Założenia i wymagania stawiane przed systemem wynikały z problemów napotykanych przez użytkowników siłowni. Celem utworzonego projektu jest próba rozwiązania jednego z głównych powodów narzeków klientów siłowni, czyli niskiej dostępności ich ulubionych maszyn. W pracy opisano istniejące sposoby wykrywania obecności człowieka, oraz przeanalizowano istniejące na rynku rozwiązania, które pozwalają sprawdzać występowanie ludzi w określonych miejscach. Utworzony system można podzielić na dwie główne części. Pierwszą z nich jest urządzenie, które po zamontowaniu w docelowym miejscu jest w stanie badać i określać zajętość, a następnie przesyłać uzyskane informacje dalej z pomocą sieci LoRaWAN. Zbudowany został prototyp, korzystający z Raspberry Pi, który zbierając dane z otoczenia z pomocą kamery jest w stanie wykryć zajętość dzięki skorzystaniu z modeli uczenia maszynowego i analizy obrazu. Druga część natomiast ma za zadanie odbierać informacje z sieci LoRaWAN i udostępniać je użytkownikom w formie aplikacji webowej. Stworzony prototyp urządzenia umieszczono w rzeczywistym środowisku badawczym i przeprowadzono szereg testów. Uzyskane wyniki testów pozwoliły na zweryfikowanie poprawności implementacji oraz umożliwiły wyciągnięcie wniosków i wskazanie potencjalnych kierunków rozwoju systemu.

Słowa kluczowe: Internet Rzeczy, Sieć LoRaWAN, Detekcja osób, Internet Rzeczy w przemyśle

LoRaWAN-enabled video sensor for monitoring device occupancy

Abstract. The paper describes the design and implementation of a system for detecting the occupancy of equipment in a gym. The requirements for the system are the result from problems encountered by gym users. The purpose of the created project is to try to solve one of the main reasons for complaints of the gym customers, namely the low availability of their favorite machines. The work describes existing ways of detecting human presence, and analyzes existing solutions on the market to check the presence of people in certain places. The created system can be divided into two main parts. The first is a device that, when installed in a target location, is able to survey and determine occupancy, and then transmit the obtained information further with the help of a LoRaWAN network. A prototype has been built, using a Raspberry Pi, which, by collecting data from the environment with the help of a camera, is able to detect occupancy through the use of machine learning models and image analysis. The second part, on the other hand, is designed to receive information from LoRaWAN and make it available to users in the form of a web application. The created prototype device was placed in a real environment and a series of tests were conducted. The results of the tests made it possible to verify the correctness of the implementation and lead to conclusions and potential directions for the development of the system.

Keywords: Internet of Things, LoRaWAN network, Human detection, Internet of Things in industry

Spis treści

1. Wprowadzenie	9
1.1. Zdefiniowanie problemu	9
1.2. Cel pracy	9
1.3. Układ pracy	9
2. Analiza stanu wiedzy oraz istniejących rozwiązań	10
2.1. Metody wykrywania osób	10
2.1.1. Wykorzystanie sensorów w detekcji ludzi	10
2.2. Uczenie maszynowe	11
2.3. Wykorzystanie algorytmów detekcji obiektów	14
2.4. Sieć LoRaWan	16
3. Proponowane rozwiązanie	18
3.1. Funkcjonalność rozwiązania	18
3.1.1. Wymagania	18
3.1.2. Wymagania funkcjonalne aplikacji webowej	18
3.2. Architektura systemu	20
3.3. Projekt urządzenia badającego zajętość	21
3.4. Projekt sieci LoRaWAN	22
3.5. Projekt części chmurowej	23
4. Implementacja rozwiązania	26
4.1. Budowa urządzenia badającego zajętość	26
4.1.1. Implementacja modułu komunikacji z nadajnikiem LoRa	27
4.1.2. Implementacja modułu wykrywania zajętości	28
4.1.3. Konfiguracja sieci VPN	29
4.2. Implementacja sieci LoRaWAN	30
4.3. Implementacja części chmurowej	32
4.3.1. Baza danych	32
4.3.2. Mikrousługa danych	33
4.3.3. Aplikacja wyzwalająca przetwarzanie danych	35
4.3.4. Menadżer sekretów	35
4.3.5. Magazyn plików	36
4.4. Aplikacja webowa	37
4.4.1. Proces uwierzytelniania	37
4.4.2. Zarządzanie kontem	39
4.4.3. Przeglądanie listy sal i urządzeń	41
4.4.4. Zarządzanie siłownią	42
4.4.5. Przeglądanie wykresów zajętości	44
5. Testy przeprowadzone w środowisku	46

0. Spis treści

5.1. Stanowisko pomiarowe	46
5.2. Wyniki eksperymentu	48
5.3. Przeprowadzone testy	48
5.3.1. Test poprawnego zmiany statusu zajętości	48
5.3.2. Test poprawnej zmiany awatara	50
5.4. Test dodawania i edycji urządzeń	51
6. Podsumowanie	54
Bibliografia	55
Wykaz symboli i skrótów	58
Spis rysunków	59
Spis tabel	60
Spis załączników	60
Spis listingów	60

1. Wprowadzenie

1.1. Zdefiniowanie problemu

Rosnąca świadomość znaczenia aktywności fizycznej spowodowała wzrost liczby ludzi uprawiających regularnie sport. Z roku na rok populacja wybierająca ten sposób spędzania czasu stopniowo się powiększa. Szacuje się, że procent ludności Polski uprawiających sport w roku 2008 wynosił 37.5%, natomiast w 2016 wzrósł do 46.4%, a w roku 2020 wynik wyniósł 47% [1]. Spośród tych, którzy praktykują aktywność fizyczną aż 5.2% respondentów wskazało kluby fitness bądź siłownie jako miejsce odbywanie ćwiczeń [2]. Ludzie ćwiczący na siłowni wskazują na szereg problemów związanych z uczęszczaniem na zajęcia fitness, wśród nich wymieniają takie jak problemy z czystością, brak zainteresowania klientem przez pracowników, niewystarczająca komunikacja czy też utrudnienia związane z brakiem łatwego dostępu do sprzętu w związku z przepelniением [3]. To właśnie próba rozwiązania ostatniego z wymienionych jest problemem podjętym w tej pracy. W celu jego rozwiązania zaprojektowany i stworzony został system sprawdzania zajętości urządzeń znajdujących się na salach siłowni.

1.2. Cel pracy

Celem pracy jest opracowanie, implementacja oraz przetestowanie systemu badania zajętości urządzeń na siłowni, pozwalającego na oznaczenie stanu maszyn na siłowni oraz prezentacje uzyskanych wyników.

1.3. Układ pracy

Praca składa się z sześciu rozdziałów. Rozdział pierwszy "Wprowadzenie" opisuje problem, cel oraz układ pracy. Rozdział drugi o tytule "Analiza stanu wiedzy oraz istniejących rozwiązań" traktuje o metodach wykrywania ludzi przez urządzenia, przeglądzie podobnych do przedmiotu pracy rozwiązań jak i o sposobie działania sieci LoRaWAN. W rozdziale trzecim "Proponowane rozwiązanie" opisana została architektura systemu oraz projekt poszczególnych elementów systemu przedstawionego w pracy. Pokazane są w nim narzędzia i technologie wykorzystane do utworzenia całego systemu. Rozdział czwarty "Implementacja rozwiązania" zawiera szczegółowy opis tego w jaki sposób zostały zaimplementowane kolejne części systemu. W rozdziale piątym "Testy przeprowadzone w środowisku" znajduje się opis stanowiska badawczego, jak i wyniki przeprowadzonych testów. Pracę kończy rozdział szósty "Podsumowanie". Zawarte są w nim wnioski oraz możliwe kierunki dalszego rozwoju projektu.

2. Analiza stanu wiedzy oraz istniejących rozwiązań

2.1. Metody wykrywania osób

Problem wykrywania obecności ludzi w danym miejscu jest stale rozwiązywany na nowo. Potrzeba wykrywania i analizowania przestrzeni pod względem przebywania w niej osób pojawia się w wielu niezależnych od siebie dziedzinach i branżach, przykładowo sieci supermarketów wykorzystują dane o obecności klientów w różnych miejscach sklepu dla celów optymalizacyjnych w postaci lepszego rozmieszczenia produktów, które docelowo generującego większy zysk. Innym przykładem takiej potrzeby jest wykorzystanie technologii pozwalającej na wykrywanie pieszych idących po chodnikach w celu poprawy bezpieczeństwa jak i analizy ścieżek wybieranych przez nich. Dzięki nim możliwe staje się wskazanie elementów infrastruktury miejskiej wymagającej remontu bądź też przebudowy. Kolejnym przykładem jest potrzeba detekcji osób znajdujących się w samochodzie w celu zapewnienia bezpieczeństwa pasażerom poprzez generowanie ostrzeżenia o niesapiętych pasach bezpieczeństwa. Powstałe metody detekcji są zróżnicowane, co pozwala je dopasować do konkretnego zapotrzebowania. Jedne polegają na wykorzystaniu sensorów różnego rodzaju, natomiast główną i najczęściej wykorzystywaną metodą jest analiza obrazu za pomocą algorytmów i modeli.

2.1.1. Wykorzystanie sensorów w detekcji ludzi

Do detekcji ludzi można wykorzystać sensory, czyli urządzenia lub elementy, które są zdolne do wykrywania, mierzenia lub rejestrowania fizycznych lub chemicznych właściwości otoczenia lub substancji. Przykładowe wartości jakie te narzędzia są w stanie badać to temperatura, ciśnienie, natężenie światła, natężenie dźwięku, wilgotność, przyspieszenie, położenie, stężenie substancji chemicznych i wiele innych.

Jednym z sensorów, który można wykorzystać jest czujnik badający pojemność kondensatora. W pracy [4] opisana została implementacja systemu monitorowania i sprawdzania stanowisk w bibliotece. Potrzeba wytworzenia takiego systemu wynikała z popularnego zjawiska "hoggingu" występującego w bibliotekach uniwersyteckich polegającego na zostawianiu przykładowo plecaka, laptopa, bądź odzienia wierzchniego w celu zajęcia miejsca, tak aby w momencie odejścia na przerwę, nikt nie zajął poprzednio zajmowanej przestrzeni. W celu sprawdzenia czy dane stanowisko jest zajmowane autorzy rozwiązania wykorzystali czujnik zbudowany z kondensatora w postaci folii miedzianej wraz z połączonym sensorem MPR121 badającym zmiany pojemności kondensatora. Następnie za pomocą testów utworzony został algorytm wykorzystujący mechanizm przesuwającego się okna pozwalający na detekcję zarówno ludzi jak i zostawionych na blacie rzeczy. Kolejnym sensorem wykorzystanym w tej samej pracy jest detektor podczerwieni, czyli urządzenia służącego do wykrywania i rejestrowania promieniowania podczerwonego (IR) w zakresie elektromagnetycznym. Promieniowanie podczerwone jest niewidoczne dla ludzkiego oka, ale jest obecne w postaci ciepła emitowanego przez obiekty. Razem z detektorem

wykorzystana została dioda IR i całość jest w stanie wykrywać ludzi bądź też przedmioty korzystając z mechanizmu działania sonarów. To znaczy że dioda IR emisuje światło, które następnie dociera do obiektów czy też ludzi, jest odbijane i wraca z powrotem padając na fotodiodę, na której odkłada się napięcie proporcjonalne do ilości promieniowania na nią padającej. Następnie wyniki analizowane są przez stworzony algorytm, który pozwala określić jaki spośród czterech stanów (zajęte przez ludzi, zajęte przez laptopa, zajęte przez inny obiekt, wolne) jest aktualnym stanem stanowiska.

Innym z czujników wykorzystywanym powszechnie w celu wykrycia ludzi jest czujnik ciśnienia. Używany jest w systemach detekcji pasażerów w samochodach osobowych. Wraz z wynalezieniem poduszek powietrznych, a następnie spopularyzowaniem ich użycia w samochodach osobowych, pojawiła się potrzeba określenia czy na fotelu pasażera znajduje się w danym momencie jakaś osoba. Wynika to z faktu iż, wybuch poduszki powietrznej w przypadku, gdy na siedzisku znajduje się dziecko, bądź też miejsce jest puste może być tragiczny w skutkach. Aby zaradzić temu problemowi powstały systemy detekcji pasażerów. Najczęściej korzystają one z sensorów potrafiących badać ciśnienie, które zamontowane są w siedziskach foteli. Pomiary zdobyte za ich pomocą analizowane są przez elektroniczny moduł kontrolny i pozwalają uzyskać informację o wadze jak i pozycji osoby siedzącej. Następnie moduł ten jest w stanie podjąć decyzję o wyłączeniu, bądź też dostosowaniu mocy z jaką jest w stanie wybuchnąć poduszki powietrznej [5].

2.2. Uczenie maszynowe

Uczenie maszynowe jest to gałąź nauki o sztucznej inteligencji jak i informatyki. Skupia się ona na wykorzystaniu zarówno algorytmów, jak i zbiorów danych w celu naśladowania sposobu, w jaki uczą się ludzie, aby stopniowo poprawiać skuteczność i jakość algorytmów. Uczenie maszynowe i sztuczna inteligencja opierająca się na nim jest wykorzystywana szeroko w dzisiejszym świecie. Rozwój algorytmów, jak i budowa nowych modeli korzystających z istniejących metod uczenia maszynowego są jednymi z najważniejszych kierunków rozwoju stawianych przed dziedziną nauk jaką jest informatyka. Znajduje zastosowanie w takich miejscach jak:

- **Rozpoznawanie mowy** - pozwala na detekcję mowy, a także przekształcenie jej na tekst. Wykorzystując przetwarzania języka naturalnego, modele pozwalają na komunikacje pomiędzy komputerem a człowiekiem z pomocą mowy. Przykładem wykorzystania rozpoznawania mowy są asystenci osobisci tacy jak Siri autorstwa Apple, czy Alexa Amazona.
- **Obsługa klienta** - polega na możliwości tworzenia wirtualnych konsultantów w postaci botów, z którymi klient jest w stanie przeprowadzić rozmowę i uzyskać pomoc. Przykładowo w branży e-commerce szeroko wykorzystywane są boty, które potrafią odpowiedzieć na pytania użytkowników związane z zamówieniami, terminem dostawy, a także potrafią pomóc w dobraniu odpowiednich produktów.

2. Analiza stanu wiedzy oraz istniejących rozwiązań

- **Wizja komputerowa** - jest to szereg zadań pozwalających komputerom na wyciągnięcie istotnych informacji ze zdjęć czy filmów. Wykorzystywane są w tym celu sieci neuronowe powstałe w wyniku uczenia maszynowego. Wizja komputerowa wykorzystywana jest w radiologii, pozwalając na wykrywanie groźnych zmian w ludzkim ciele, bądź też w branży motoryzacyjnej w samochodach autonomicznych.
- **Detekcja oszustw** - uczenie maszynowe pozwala na analizę dziesiątek tysięcy transakcji internetowych. Wśród znajdują się transakcje z wykorzystaniem fałszywych kont, kart kredytowych, oraz prób oszustwa. Odpowiednio wyuczone modele pozwalają na detekcję takich oszustw i ich zatrzymywanie.

W procesie uczenia się modelu uczenia wyróżnić trzy kluczowe elementy:

Funkcja błędu funkcja pozwalająca na ocenę wyniku uzyskanego przez model poprzez porównywanie wyniku uzyskiwanego przez model do faktycznego i znanego wyniku dla tych samych danych testowych. Pozwala na ocenę jakości modelu.

Proces optymalizacji modelu model ucząc się, nadaje odpowiednie wagę danym wejściowym, następnie porównuje uzyskane wyniki dla zbioru danych testowych, z przygotowanymi prawidłowymi wynikami i stara się autonomicznie modyfikować wagę dla parametrów, tak aby zminimalizować różnice. Proces powtarza się do czasu uzyskania określonego na początku, progu skuteczności algorytmu.

Proces decyzyjny proces polegający na wykonanie przez model predykcji bądź też klasyfikacji. Model na podstawie danych wejściowych jest w stanie dokonać przybliżonego wyniku bazując na poznanych wcześniej wzorach w danych.

Modele uczenia maszynowego można podzielić na:

Uczenie nadzorowane metoda uczenia polegająca na dostarczeniu uczącemu się algorytmowi zbioru danych treningowych, który zawiera dołączone rozwiązanie problemu. Mogą to być odpowiednie etykiety, bądź klasy. Z pomocą tej metody rozwiązywane są dwa typy problemów, problem regresji czyli przewidywania wartości i problem klasyfikacji czyli przewidywania klas. Przykładem problemu regresji jest określenie wartości nieruchomości na podstawie jej lokalizacji, roku budowy, powierzchni. Przykładem problemu klasyfikacji jest na przykład określenie czy pacjent jest chory na daną chorobę na podstawie jego wyników krwi. Przykładami algorytmów uczenia maszynowego, które są w stanie rozwiązać problem regresji są:

- regresja liniowa
- drzewo regresyjne
- sieci neuronowe
- regresja wielomianowa

Natomiast przykładowymi algorytmami służącymi do klasyfikacji są:

- drzewa decyzyjne
- las losowy

- sieci neuronowe
- metoda k-najbliższych sąsiadów

Uczenie nienadzorowane metoda uczenia polegająca na dostarczeniu uczącemu się algorytmowi zbioru danych treningów, które nie są w żaden sposób oznaczone. Zadaniem algorytmu jest znalezienie powiązań między danymi. Jednym z typów wykorzystywanych algorytmów są algorytmy analizy skupień, pozwalają one na zgrupowanie danych w konkretne skupiska. Przykładowo rozwiązywanym problemem, może być pogrupowanie osób chodzących do kina w konkretne grupy docelowe. Innym rodzajem wykorzystywanych algorytmów są algorytmy wizualizujące. Pozwalają one na pokazanie zbioru danych w postaci dwu lub trzy wymiarowej co pozwala odnaleźć pewne zależności między nimi. W nienadzorowanym uczeniu maszynowym, wykorzystywane są algorytmy takie jak:

- jednoklasowa maszyna wektorów nośnych
- metoda k-średnich
- jądrowa analiza głównych składowych
- hierarchiczna analiza skupień

Uczenie przez wzmacnianie metoda uczenia polegająca na uczeniu modelu poprzez interakcje ze środowiskiem. Model wykorzystując zgromadzone informacje ma za zadanie prowadzić interakcje ze środowiskiem. W odróżnieniu do poprzednich metod, algorytmowi nie są dostarczone dane testowe, tylko środowisko, z którego uczący się algorytm musi samodzielnie zebrać dane. Celem algorytmu jest takie działanie, aby zwiększyć maksymalnie uzyskaną nagrodę. Nagroda jest to postawiony przed algorymem cel, do którego zmaksymalizowania powinien dążyć. Przykładowo, dla algorytmu uczącego się grać w szachy, środowiskiem będzie gra w szachy wraz z jej zasadami, a nagrodą będzie wygranie partii. W tej metodzie uczenia wyróżniane są następujące elementy:

- **Środowisko** - symulacja, bądź zadanie, z którym algorytm wchodzi w interakcję.
- **Agent** - element, który prowadzi interakcje z przygotowanym środowiskiem. Jego celem jest zmaksymalizowanie nagrody, poprzez nauczenie się najbardziej korzystnych zachowań w środowisku. Za decyzje, jakie podejmuje agent odpowiada polityka, czyli funkcja, która zwraca akcję. Najczęściej rolę polityki pełni sieć neuronowa.
- **Bufor** - magazyn danych, pozwalających na przetrzymywanie danych zbieranych przez agenta w trwającej iteracji symulacji. Zgromadzone dane wykorzystywane są do późniejszego wytrenowania modelu, aby ten z każdą iteracją poprawiał swój wynik.

Uczenie półnadzorowane metoda uczenia polegająca na dostarczeniu danych testowych algorytmowi, które są tylko częściowo oznaczone. Proces oznaczania danych etykietami jest dość czasochłonny i kosztowny. Stąd powstała metoda, która jest połączeniem

2. Analiza stanu wiedzy oraz istniejących rozwiązań

metody nienadzorowanej i nadzorowanej. Łączy ona algorytmy znane z uczenia nadzorowanego i nienadzorowanego i implementuje je w sieć neuronową. Dzięki temu możliwe jest wytrenowanie modelu wykorzystując niewielką ilość oznaczonych danych w stosunku do tych nieoznaczonych.

Uczenie przyrostowe i wsadowe to kryterium podziału systemów uczenia maszynowego, bierze pod uwagę w jaki sposób nadsyłane mogą być dane. W systemie przyrostowym, dane służące do nauki, mogą być nadsyłane sekwencyjnie w trakcie działania modelu. Natomiast w systemie wsadowym, aby dotrenować model, należy zaprzestać jego działania i przejść do etapu uczenia poprzez zasilenie go paczką nowych danych. Dopiero po zakończeniu etapu nauki można wznowić działanie modelu.

2.3. Wykorzystanie algorytmów detekcji obiektów

Detekcja obiektów to termin opisujący szereg zadań, które pozwalają na identyfikację obiektów z użyciem wizji komputerowej. Pozwala przede wszystkim na identyfikację i oznakowywanie obiektów w materiale źródłowym, którym może być zdjęcie, film, czy nawet materiał na żywo. Najczęściej wykorzystywane są gotowe wyuczone modele i dzięki nim do dokonania detekcji wystarczy zasilać układ danymi wejściowymi aby uzyskać gotowy wynik w postaci obrazu z nałożonymi oknami wyznaczającymi odseparowane i ponazywane obiekty. To właśnie wspomniane okna są jednym z kluczowych elementów całego procesu, pozwalają one na odseparowanie wykrytego obiektu od tła, a także nadanie mu odpowiedniej etykiety. Przybierają kształt czworokąta i mogą na siebie nachodzić jeżeli wykryte obiekty również na siebie nachodzą.

Zadaniami wizji komputerowej są:

Klasyfikacja obrazów proces polegający na przypisaniu kategorii zwanej klasą do danego obrazu. W praktyce pozwala określić co jest ukazane na danym obrazie, ale nie jest w stanie określić lokalizacji obiektu.

Segmentacja proces polegający na grupowaniu pikseli o podobnych cechach. Docelowo pozwala on na identyfikację obiektów na obrazie.

Lokalizacja obiektów proces polegający na znajdywaniu lokalizacji jednego bądź też kilku obiektów znajdujących się na danym obrazie.

Detekcja obiektów nie jest możliwa bez modeli stworzonych do wykonywania tego zadania. Modele tworzone są w procesie nadzorowanej nauki maszynowej wykorzystującym tysiące uprzednio oznaczonych za pomocą odpowiedniej etykiety zdjęć. Z pomocą przychodzą gotowe zbiory danych takie jak COCO. Każdy z obrazów zbiorze musi mieć dopasowany do niego plik opisowy zawierający krawędzi obiektów wraz z ich klasami.

Do nauki modeli pozwalających na detekcję najczęściej wykorzystuje się algorytmy z dwóch rodzin. Pierwszą z nich jest rodzina algorytmów R-CNN, która została zaproponowana w 2014 roku [6]. Algorytm ten składa się z trzech głównych kroków. Pierwszy nazywa

się selektywnym wyszukiwaniem, znajduje on grupy pixeli, które być może reprezentują jakiś obiekt. Grupy te nazywane są regionami zainteresowania. Wynikiem tego etapu jest utworzenie około dwóch tysięcy regionów zainteresowania dla każdego z obrazów. Następnie każdy z wyselekcjonowanych regionów przekształcany jest, tak aby uzyskać zunifikowany rozmiar, a potem przepuszczony jest przez konwolucyjną sieć neuronową, która następnie jest w stanie wyodrębnić cechy każdego z regionów i zapisać je w wektorze zawierającym wartości liczbowe. Ostatnim krokiem jest analiza wyodrębnionych wektorów cech przez algorytm klasyfikujący pozwalający na przypisanie odpowiedniej klasy wektorowi. Jednym z możliwych wyników jest przypisanie klasy "tło", jeżeli dany region nie zawiera żadnego obiektu. Z używaniem algorytmu R-CNN wiąże się kilka problemów. Po pierwsze początkowy etap jest pracochłonny i złożony obliczeniowo, wyznaczenie dwóch tysięcy regionów nie jest szybkim procesem. Następnie wyodrębnianie cech ze wszystkich RoI jest równie czasochłonne ze względu na liczbę obliczeń, których musi dokonać sieć neuronowa. Z tego względu metody R-CNN nie nadają się do zastosowań przetwarzania obrazów na żywo. Ostatnim z problemów jest fakt, że na cały model składają się trzy odseparowane procesy przez co trudno zintegrować obliczenia i poprawić ich szybkość.

Model R-CNN został ulepszony i na jego podstawie powstał rozwinięty model zwany szybkim R-CNN [7]. Poprawa polegała na połączeniu warstw selekcji obrazów i wyodrębniania cech w jeden model uczenia maszynowego. Zmieniły się natomiast wymagane dane wejściowe, pierwotny model wymagał tylko obrazów, a nowa wersja wymaga obrazów wraz z potencjalnymi wyselekcjonowanymi regionami. Do modelu szybkiego R-CNN wprowadzona została warstwa pozwalająca na wyodrębnianie cech wszystkich z proponowanych regionów w trakcie jednego przebiegu przez sieć w odróżnieniu do poprzedniej wersji gdzie każdy z regionów obliczany był oddziennie. Zaowocowało to w znaczącą poprawę szybkości, natomiast potrzeba selekcjonowania regionów nadal pozostała, przez co model szybkiego R-CNN nadal nie nadawał się do zastosowania w przetwarzaniu obrazów na żywo.

Następną fazą rozwoju było wprowadzenie szybszego modelu R-CNN [8]. Udało się w nim zwalczyć problem odseparowanej selekcji regionów poprzez włączenie algorytmu wyodrębniania cech do sieci służącej detekcji obiektów. Model jako dane wejściowe przyjmuje obrazy, a jego wynikiem jest lista wykrytych klas wraz z wyznaczonymi granicami, wewnętrznych których znajdują się wykryte obiekty. Sukces szybszego R-CNN polega na dodaniu sieci propozycji regionów, która z map cech wytworzonych przez konwolucyjną sieć neuronową jest w stanie zaproponować listę ramek ograniczających, w których mogą znajdować się obiekty. Pozwoliło to osiągnąć o wiele większą wydajność, która prawie zaczęła pozwalać na przetwarzanie obrazów na żywo. Modele z rodziny R-CNN są nadal stale rozwijane i ulepszane.

Drugą rodziną modeli używaną powszechnie w detekcji obiektów jest rodzina YOLO. Wykorzystuje ona głębokie uczenie, które pozwala na uzyskanie polepszonej szybkości i

2. Analiza stanu wiedzy oraz istniejących rozwiązań

dokładności detekcji. Głównym udoskonaleniem modeli z rodziny YOLO jest integracja detekcji obiektów i ich klasyfikacji w jedną kompletną sieć neuronową. Zamiast oddziennie wyodrębniać cechy i regiony, model YOLO dokonuje niezbędnych obliczeń w jednym przejściu przez sieć, stąd wzięła się jego nazwa "you only look once" co oznacza z języka angielskiego "patrzysz tylko raz". Jego szybkość jest wystarczająco do obsługi obrazów na żywo. Jest stale rozwijany i powstają jego coraz nowsze wersje, które zwiększą szybkość jak i dokładność. W chwili obecnej powstała jego wersja numerowana 7 [9].

2.4. Sieć LoRaWan

LoRaWAN jest siecią należącą do rodzin sieci LPWAN charakteryzujących się bezprzewodowym połączeniem na duże dystanse z urządzeniami końcowymi. Pozwalają one na wymianę krótkich wiadomości i na transmisję o niskiej przepustowości. Sieci te wykorzystywane są przede wszystkim w monitorowaniu i zarządzaniu inteligentnymi miastami jak i w zastosowaniach industrialnych opartych o urządzenia Internetu Rzeczy. W porównaniu z innymi bezprzewodowymi sieciami, LPWAN korzysta z urządzeń o wiele mniejszym zużyciu energii kosztem ilości przenoszonych informacji. Typowo zasięg takich sieci wynosi około dziesięciu kilometrów, a dzięki prostocie, oraz lekkości protokołów użytych w komunikacji urządzenia końcowe jak i sprzęt obsługujący sieć jest stosunkowo tani w porównaniu do innych rozwiązań. Poprzez niskie zużycie energii, urządzenia mogą być zasilane bateriami a to pozwala na ich działanie nawet do kilu lat. Przykładami LPWAN są LoRaWAN, Wi-SUN, NB-IOT, Sigfox. LoRa natomiast jest warstwą fizyczną sieci LoraWAN. Wykorzystuje modulację CSS, polegającą na nadawaniu sygnałów o zmiennej częstotliwości. Cechą charakterystyczną jest to, że zmiana częstotliwości sygnału odbywa się liniowo. Dzięki temu, sygnał nabiera odporności na zakłócenia selektywne, efekt Doplera oraz ogranicza błąd synchronizacji częstotliwości w odbiorniku.

Typowa architektura sieci LoRaWAN składa się z następujących elementów:

Urządzenia końcowe urządzenia będące w stanie otrzymywać i wysyłać informacje za pomocą wiadomości zmodulowanych LoRa

Bramki urządzenia pośredniczące w wymianie wiadomości pomiędzy serwerem sieciowym a urządzeniami końcowymi. Komunikują się z nimi za pomocą LoRa.

Serwer sieciowy część software'u uruchomiona w chmurze zarządzająca całą siecią

Serwer aplikacyjny część software'u pozwalająca na przetwarzanie danych uzyskanych z urządzeń końcowych jak i wysyłanie do nich poleceń

Urządzenia końcowe komunikują się z najbliższymi bramkami, natomiast one połączone są z pomocą Internetu do serwera sieciowego. Specyfiką LoRaWAN jest to, że używają protokołu typu ALOHA, a konkretnie jego wariantu z podziałem dziedziny czasu na konkretne szczeliny. Jego idea polega na tym, iż nadajniki mogą wysyłać wiadomości, kiedy tego potrzebują, natomiast stacje bazowe, muszą zaczekać na początek szczeliny czasowej

2. Analiza stanu wiedzy oraz istniejących rozwiązań

i tylko wtedy mogą zacząć nadawanie. Długość pakietu musi być zgodna z czasem trwania pojedynczej szczeliny. Urządzenia końcowe mogą retransmitować pakiety kilkukrotnie w celu zagwarantowania przesłania wiadomości do sieci. Wiadomości wysyłane są w eter i kilka bramek może odebrać tą samą wiadomość. Następnie bramki przekazują otrzymane komunikaty do serwera sieciowego, który przeprowadza proces deduplikacji i zapisuje pojedynczą kopię wiadomości zamiast wszystkich otrzymanych.

3. Proponowane rozwiązanie

3.1. Funkcjonalność rozwiązania

Zaprojektowany system służy do badania zajętości urządzeń na siłowni wraz z możliwością analizowania i wyświetlania wyników za pomocą interaktywnej aplikacji webowej.

Jednym z celów systemu jest zwiększenie zadowolenia klientów z usług oferowanych przez placówki, w których zakupili oni karnet poprzez zwiększenie zadowolenia z pobytu na siłowni, dzięki lepszemu dobraniu godzin treningu, a także większej dostępności maszyn. Drugim z celów jest dostarczenie właścicielom, bądź też osobom zarządzającym siłowniami informacji na temat średniej zajętości konkretnych urządzeń, czyli tego jak często są one wykorzystywane. Dostarczenie takich informacji pozwala na dokonywanie lepszych zakupów nowego sprzętu dla obiektu, jak i również wymianę urządzeń praktycznie nieużywanych na cieszące się większą popularnością. Opracowany projekt przeznaczony jest dla siłowni średniej, bądź dużej wielkości ze względu na stosunkowo wysoki koszt zakupu i instalacji urządzeń badających zajętość.

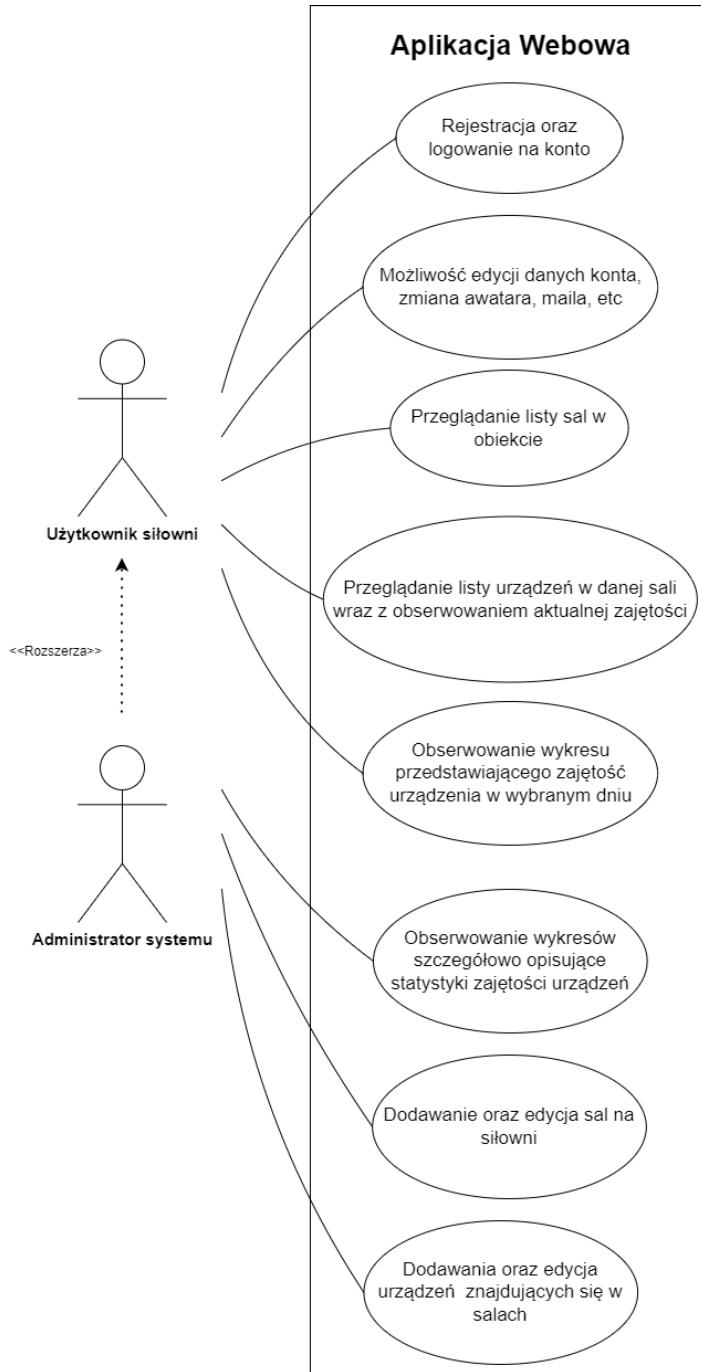
3.1.1. Wymagania

Założenia oraz wymagania stawiane przed systemem badania zajętości urządzeń prezentują się w następujący sposób:

- stworzenie urządzenia będącego czujnikiem IoT pozwalającego na wykrywanie zajętości stanowisk na siłowni
 - wykorzystanie kamery jako sensora badającego stan stanowiska
 - określanie zajętości na podstawie informacji zebranych z kamery
 - przesyłanie zbadanego stanu z wykorzystaniem sieci LoRaWAN do środowiska sieciowego
- stworzenie aplikacji webowej dostarczającej wizualnej interpretacji wyników badań zajętości

3.1.2. Wymagania funkcjonalne aplikacji webowej

Diagram przypadków użycia aplikacji webowej przedstawiono na rysunku 3.1



Rysunek 3.1. Diagram przypadków użycia aplikacji webowej

Aplikacja w założeniach ma pozwalać na:

Rejestracje oraz logowanie się na konto użytkownik jest w stanie się zarejestrować oraz zalogować na wcześniej utworzone konto

Edycje danych konta użytkownik jest w stanie edytować podane podczas rejestracji przez siebie dane takie jak adres e-mail czy też jest w stanie zamieścić, bądź zmienić avatar przypisany do konta

3. Proponowane rozwiązanie

Przeglądanie listy sal w obiekcie aplikacja udostępnia widok wyświetlający poszczególne sale i ich szczegóły

Przeglądanie listy urządzeń w danej sali wraz z obserwowaniem aktualnej zajętości aplikacja udostępnia widok pozwalający na przeglądanie urządzeń znajdujących się w wybranej sali. Kluczowym elementem tego widoku jest pokazanie aktualnej dostępności wybranego urządzenia.

Obserwowanie wykresu przedstawiającego zajętość urządzenia w wybranym dniu system udostępnia widok pozwalający na obserwowanie wykresu pokazującego jak przebiegał status zajętości wybranego urządzenia w trakcie wybranego przez użytkownika dnia.

Obserwowanie wykresów szczegółowo opisujące statystyki zajętości urządzeń system udostępnia administratorom widok pozwalający na obserwowanie wykresów obrazujących dogłębne statystyki na temat zajętości maszyn

Dodawanie oraz edycja sal na siłowni aplikacja udostępnia administratorom widok pozwalający na edytowanie oraz dodawanie nowych sal znajdujących się w danej placówce

Dodawania oraz edycja urządzeń znajdujących się w salach aplikacja udostępnia administratorom widok pozwalający na edytowanie oraz dodawanie nowych maszyn znajdujących się w danej sali

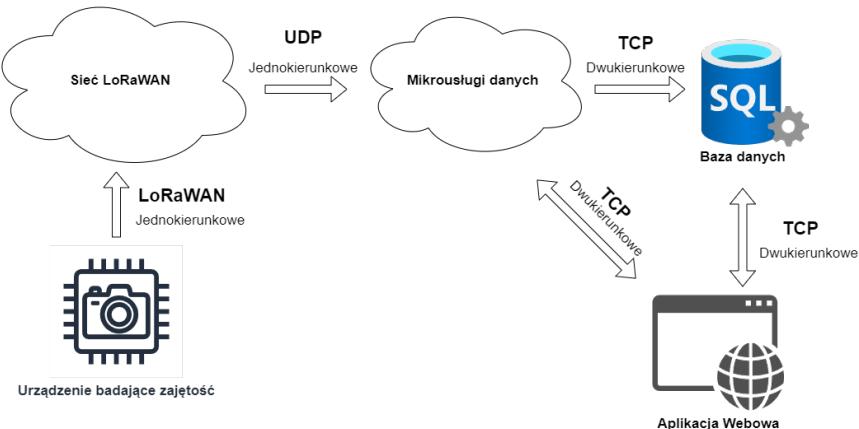
3.2. Architektura systemu

Cały projekt systemu można podzielić na trzy odrębne części. Wśród nich wymienić można:

- Urządzenie badające zajętość
- Sieć LoRaWAN wraz z konfiguracją części chmurowej sieci
- Część chmurową zawierającą w sobie zarówno aplikacje webową, jak i mikrousługi danych, bazę danych i inne niezbędne elementy

Część związana z urządzeniem badającym zajętość wymaga dobrania odpowiednich komponentów, następnie ich konfiguracji oraz montażu, tak aby wszystkie elementy można było traktować jako odrębne urządzenie. Część związana z siecią LoRaWAN wymaga doboru dostawcy sieci, następnie jej konfiguracji i zapewnienia komunikacji pomiędzy urządzeniami końcowymi a siecią LoRaWAN jak i zapewnienie komunikacji pomiędzy siecią LoRaWAN a częścią chmurową całego projektu. Część chmurową wymaga stworzenia aplikacji, w tym wypadku nazywanej mikrousługą danych, która będzie w stanie przyjmować komunikaty z sieci LoRaWAN i umieszczać je w bazie danych, która jest kolejnym elementem wymagającym stworzenia. Trzecim komponentem jest natomiast aplikacja webowa pozwalająca na wyświetlanie zgromadzonych danych jak i zarządzanie urządzeniami.

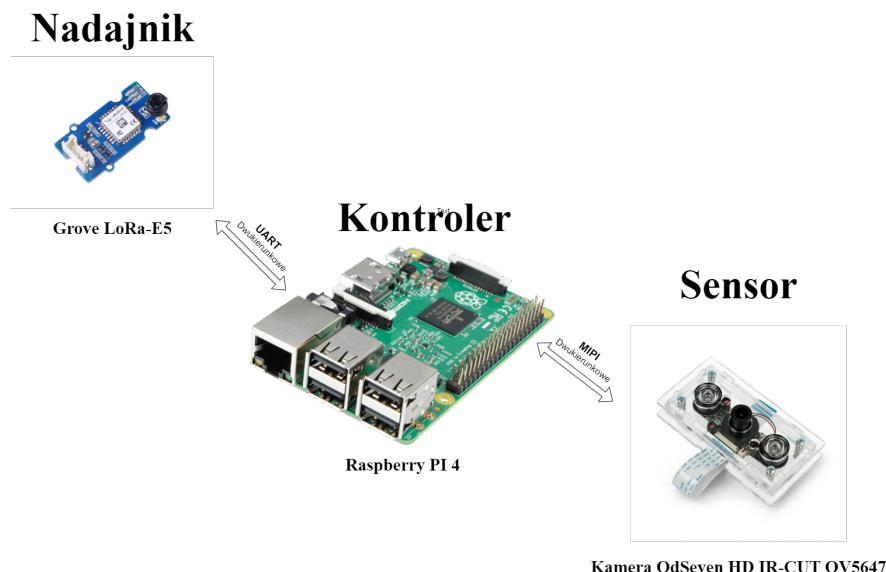
Wysokopoziomową architekturę systemu przedstawiono na rysunku 3.2



Rysunek 3.2. Wysokopoziomowa architektura systemu

3.3. Projekt urządzenia badającego zajętość

Schemat połączenia głównych modułów urządzenia widoczny jest na rysunku 3.3



Rysunek 3.3. Schemat budowy urządzenia

Na gotowe urządzenie składają się trzy elementy. Pierwszym z nich jest kontroler, którego zadaniem jest odbieranie sygnałów z sensora, dokonywanie niezbędnych obliczeń i analizy uzyskanych danych z sensora. Po wykonaniu tychże obliczeń jest w stanie przesłać wynik z pomocą nadajnika. W projekcie zdecydowano się na wykorzystanie **Raspberry Pi 4** jako kontrolera [10]. Kluczowym w wyborze tej konstrukcji była jej stosunkowo duża moc obliczeniowa biorąc pod uwagę niewielki rozmiar. Niemniej ważna była również

3. Proponowane rozwiązanie

łatwość implementacji z wykorzystaniem tej platformy jak i szeroka dostępność informacji publikowanych w Internecie na temat programowania tego typu kontrolerów.

Drugim z elementów jest sensor. Założeniem projektu było wykorzystanie czujnika w postaci kamery w celu dokonywania dokładniejszych pomiarów. Wybrana została kamera **OdSeven HD IR-CUT OV5647** [11]. Charakteryzuje się ona dodatkiem w postaci dwóch diod IR zamontowanych w samym module kamery. Dzięki nim nawet w słabych warunkach oświetleniowych, takich które dość często potrafią panować na siłowniach można uzyskiwać dokładny obraz. A to właśnie dokładny obraz pozwala na trafne określanie zajętości. Przy jej wyborze ważna była również kompatybilność z kontrolerem, jako że kamera OdSeven jest urządzeniem dedykowanym do współpracy z Raspberry Pi 4.

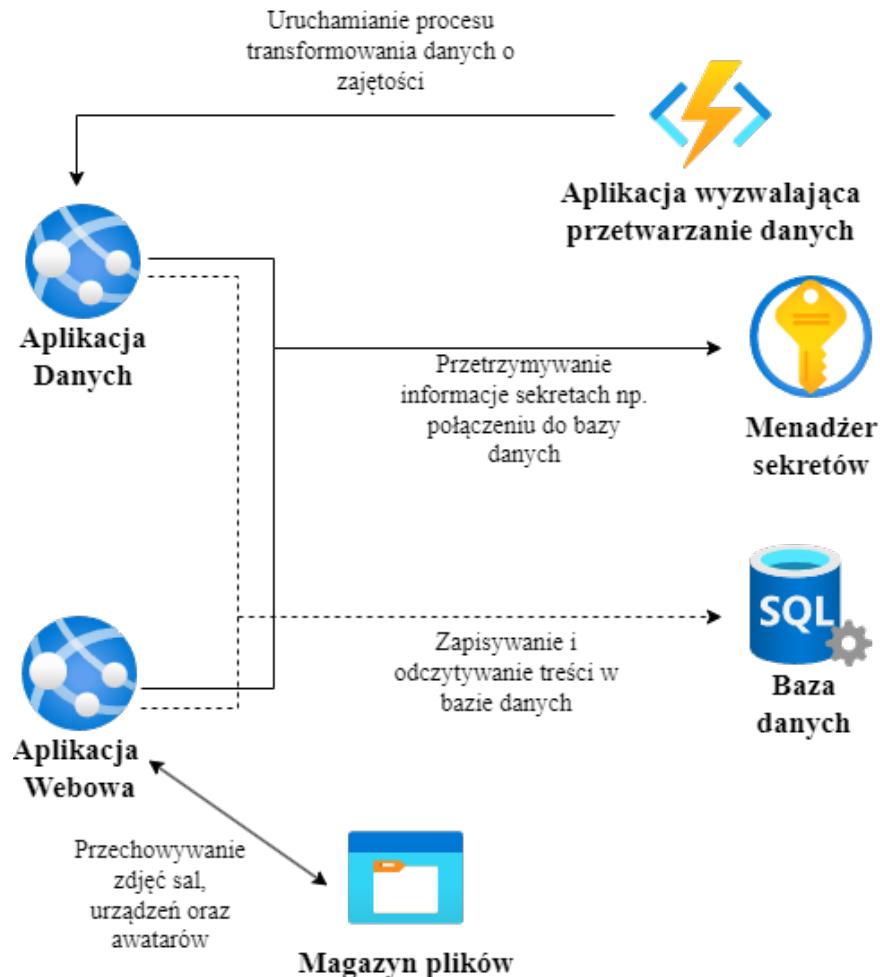
Ostatnim z elementów jest nadajnik. Służy on do komunikacji urządzenia z resztą systemu znajdującego się w chmurze. Zgodnie z założeniami do komunikacji należało skorzystać z sieci LoRaWAN, w tym celu zdecydowano się na wybranie zewnętrznego modułu **Grove - LoRa-E5**. Komunikacja z tym modułem odbywa się z pomocą interfejsu UART i specjalnych komend AT [12].

3.4. Projekt sieci LoRaWAN

Projekt zakłada użycie sieci LoRaWAN do przesyłu informacji z sensorów do części chmurowej. Jako dostawcę wybrano sieć The Things Network [13] stworzoną i utrzymywianą przez firmę The Things Industries. Na jej wybór złożyło się kilka czynników, są to m.in. duże pokrycie terenu Polski przez sieć co pozwala na korzystanie z już istniejących bramek, co w efekcie ogranicza koszt wprowadzenia całego rozwiązania, gdyż nie jest wymagane posiadanie własnej bramki. Kolejnym czynnikiem jest fakt, że korzystanie z sieci TTN jest darmowe. Konfiguracja sieci TTN wymaga stworzenia aplikacji w konsoli chmury The Things Network. Po jej utworzeniu należy dodać i skonfigurować urządzenia końcowe poprzez wprowadzenie w aplikacji ich numerów DevEUI, czyli 64-bitowych globalnie unikalnych numerów pozwalających na identyfikację poszczególnych urządzeń w sieci LoRaWAN. W procesie dołączania urządzeń do sieci, wysyłają one wiadomość, która zawiera ich DevEUI, jak i AppEUI, czyli również 64-bitowy globalnie unikalny numer pozwalający na identyfikację poszczególnych aplikacji w sieci LoRaWAN. Kolejnym elementem wymagającym konfiguracji jest stworzenie translatora wiadomości przesyłanych przez urządzenia. Takie wymaganie wynika z faktu iż komunikaty przesyłane do bramek mają formę binarną, która nie jest wystarczająco czytelna. Ostatnim elementem jest wybór sposobu integracji chmury TTN z mikrousługą danych. Zdecydowano się na wybranie modułu Webhooks [14]. Pozwala on na bezpośrednie wykonywanie zapytań HTTP do mikrousługi danych, która z kolei będzie w stanie przyjmować takowe zapytania i umieszczać dane o zajętości w bazie danych. Oprócz konfiguracji chmury TTN, należy także uzupełnić konfigurację urządzeń o wygenerowany numer AppEUI.

3.5. Projekt części chmurowej

Architekturę części chmurowej można zobaczyć na rysunku 3.4



Rysunek 3.4. Schemat części chmurowej

Jako dostawce usług chmurowych wybrano Microsoft Azure czyli kompleksową platformę chmurową oferowaną przez firmę Microsoft. Stanowi ona zbiór zintegrowanych usług obliczeniowych, sieciowych, analitycznych, pozwalających na przechowywanie danych oraz rozwiązań dla biznesu. Azure umożliwia użytkownikom tworzenie, wdrażanie i zarządzanie różnymi typami aplikacji i usług w chmurze, z elastycznymi możliwościami skalowania w górę lub w dół w zależności od potrzeb. Platforma Azure oferuje również narzędzia do zarządzania danymi, tworzenia sztucznej inteligencji, Internetu Rzeczy (IoT) oraz integracji z innymi usługami i rozwiązaniami Microsoft. Dzięki swojej skalowalności, niezawodności i bogatej funkcjonalności, Azure jest szeroko wykorzystywane w różnych sektorach i branżach do tworzenia innowacyjnych rozwiązań opartych na chmurze [15].

Część chmurowa dzieli się na sześć elementów. Pierwszym z nich jest aplikacja danych, która pozwala na wprowadzenie raportu o aktualnym stanie maszyny na siłowni, jak i na uruchomienie procesu transformacji raportów z dnia w bazie danych do postaci

3. Proponowane rozwiązanie

pozwalającej na tworzenie szczegółowych wykresów. Do jej utworzenia wykorzystano środowisko ASP.NET Core [16]. Skorzystano także z biblioteki Entity Framework Core [17] jako ORM czyli biblioteki służącej do łatwiejszego komunikowania się z bazą danych poprzez automatyczne mapowanie obiektów z języka danego środowiska (w tym wypadku jest to język C#) na struktury w bazie relacyjnej. Do wdrożenia aplikacji na środowisko testowe wykorzystano Azure App service czyli usługę platformy Microsoft Azure dostarczającą gotową infrastrukturę uruchomieniową aplikacji w chmurze [18].

Następnym kluczowym elementem jest baza danych. Zdecydowano się na wybranie bazy Microsoft SQL Server ze względu na łatwość integracji ze platformą Azure jak i ze środowiskiem .NET. Do wdrożenia na środowisko testowe wykorzystano Azure SQL Server czyli usługę, która pozwala na utworzenie wirtualnego serwera bazodanowego [19].

Kolejną częścią projektu systemu działającą w chmurze jest menadżer sekretów czyli narzędzie lub usługa, która służy do bezpiecznego przechowywania, zarządzania i udostępniania poufnych informacji, takich jak hasła, klucze API, certyfikaty, tokeny dostępu i inne tajne dane. W tym projekcie przechowywane są w niej takie informacje jak hasło do bazy danych czy też unikalny klucz do jednej z wykorzystanych bibliotek. Jako menadżer sekretów służy Azure Key Vault, czyli usługa dostarczana przez platformę Microsoft Azure [20].

Czwartym komponentem jest magazyn plików w którym gromadzone są pliki graficzne zarówno obrazujące maszyny, jak i rozplanowanie sal. W tymże magazynie projekt zakłada umieszczenie także awatarów użytkowników. Do pełnienia jego roli wybrano Azure Blob Storage [21], czyli usługę przechowywania obiektów w chmurze Microsoft Azure, umożliwiającą skalowalne, trwałe i wysoko dostępne przechowywanie dużej ilości danych, takich jak pliki, obrazy czy multimedia.

Przed ostatnim elementem jest aplikacja pozwalająca na wyzwalanie procesu przetwarzania danych o zajętości z konkretnych dni. Wymogiem jest aby proces ten uruchamiał się raz dziennie o określonej porze. Wdrożenia aplikacji na platformie Azure wyposażone są w mechanizm, który polega na tym, że w przypadku zmniejszonej ilości zapytań do aplikacji, bądź też całkowitego ich braku, wprowadza wdrożoną aplikację w stan uśpienia. Sprawia to, że aplikacja nie jest w stanie wykonywać zaplanowanych zadań z pomocą wewnętrznych modułów służących do planowania i zarządzania poleceniami. Z tego powodu zdecydowano się wykorzystać zewnętrzne źródło generujące zapytania do aplikacji o określonej porze. Do utworzenia aplikacji generującej cykliczne zapytania wybrano usługę Azure Functions [22]. Usługa ta umożliwia tworzenie i uruchamianie drobnych, jednofunkcyjnych, bezstanowych funkcji w reakcji na różne zdarzenia, takie jak wywołania HTTP, zmiany w bazie danych czy pojawienie się nowego pliku w Azure Blob Storage. W tym projekcie utworzona aplikacja będzie wykonywała raz dziennie o określonej porze zapytanie HTTP w kierunku aplikacji danych.

Ostatnim, ale nie najmniej ważnym elementem jest aplikacja webowa spełniająca

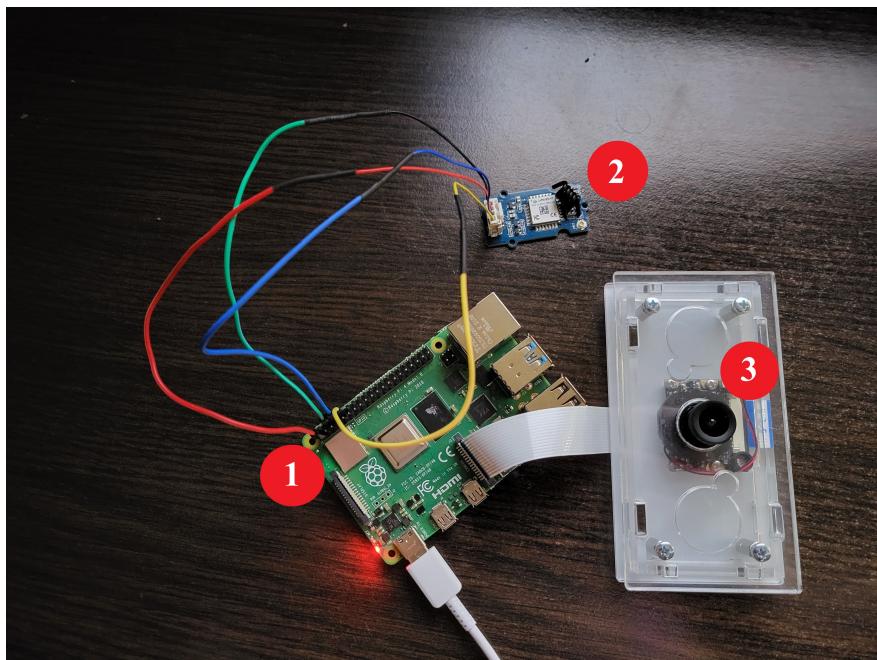
założone w podrozdziale 3.1.2 wymagania funkcjonalne. Zadecydowano na utworzenie jej wykorzystując środowisko ASP.NET Core a konkretnie ASP.NET Core Blazor [23] w modelu Server Side Application. Blazor jest frameworkm służącym do tworzenia aplikacji internetowych. Jego cechą charakterystyczną jest możliwość pisania kodu zarówno po stronie klienta, jak i serwera z wykorzystaniem języka C#. Wybrano model hostingu Server Side Application ze względu na niewielką docelową ilość użytkowników. Wszystkie obliczenia w tym modelu wykonywane są na serwerze w odróżnieniu od innych aplikacji webowych. Pozwala ta na o wiele szybsze ładowanie stron, gdyż ilość danych do pobrania przez użytkownika jest o wiele mniejsza porównując do innych modeli. Podobnie jak w aplikacji danych jako bibliotekę ORM wykorzystano Entity Framework Core. W celu ułatwienia tworzenia komponentów graficznych i funkcjonalnych takich jak przyciski, widoki list, grafiki zdecydowano się na wykorzystanie biblioteki komponentów Blazorise [24]. Pozwala ona na użycie gotowych, wydajnych, przetestowanych pod względem jakości komponentów, co pozwala na łatwiejsze tworzenie estetycznych i funkcjonalnych interfejsów. Do pomocy przy tworzeniu wykresów postanowiono wykorzystać inną bibliotekę komponentów, którą została biblioteka Syncfusion [25].

4. Implementacja rozwiązania

4.1. Budowa urządzenia badającego zajętość

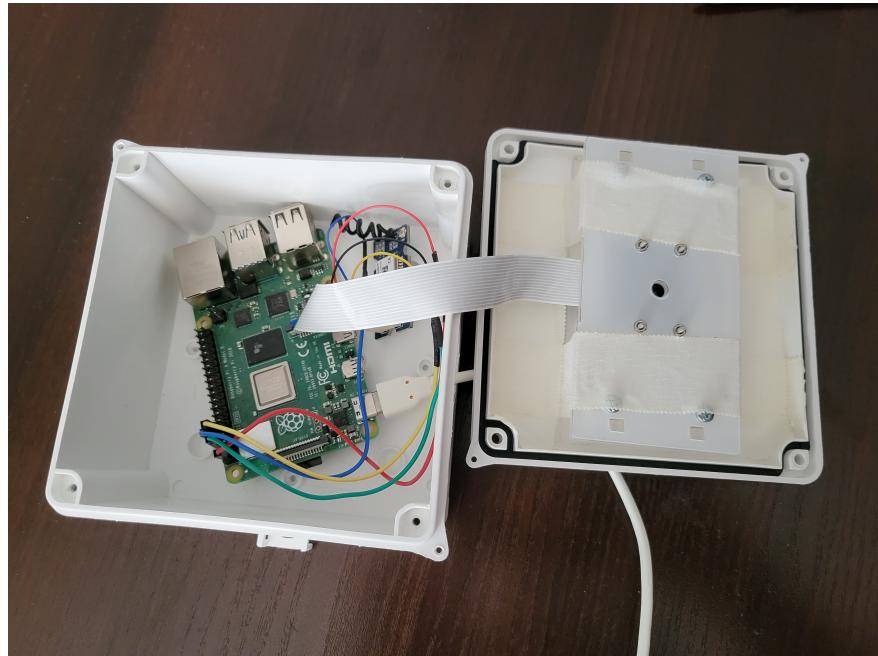
Pierwszym krokiem w budowie urządzenia była początkowa konfiguracja kontrolera, w tym celu wybrano Raspberry Pi 4. Polegała ona na zainstalowaniu systemu Raspberry Pi OS [26] na karcie SD z pomocą narzędzia Raspberry Pi Imager. Następnym krokiem było przyłączenie kamery oraz nadajnika LoRa do kontrolera. Do połączenia kamery wykorzystywany jest specjalny taśmowy przewód, który wpasowuje się w dedykowane do kamer gniazdo znajdujące się na Raspberry Pi. Nadajnik Grove LoRa-E5 wymaga komunikacji z pomocą interfejsu UART, jak i dostarczenia zasilania. Do tych celów wykorzystano wyprowadzenia GPIO znajdujące się na płytce. Po wstępnej konfiguracji i połączeniu wszystkich elementów urządzenie prezentowało się w sposób widoczny na rysunku 4.1. Widać na nim odpowiednio:

1. Kontroler Raspberry Pi 4
2. Nadajnik Grove LoRa-E5
3. Kamerę OdSeven HD IR-CUT OV5647



Rysunek 4.1. Testowa konfiguracja urządzenia

Po stworzeniu i przetestowaniu skryptów służących odpowiednio do wykrywania zajętości, jak i do komunikacji z modułem LoRa z pomocą interfejsu UART, urządzenie zostało usadowione w obudowie, którą została skrzynka montażowa. Wnętrze skrzynki montażowej po osadzeniu urządzenia widoczne jest na rysunku 4.2.



Rysunek 4.2. Wnętrze skrzynki montażowej

Skompletowane i w pełni gotowe do zamontowania w docelowym miejscu urządzenie widoczne jest na rysunku 4.3.



Rysunek 4.3. Gotowe urządzenie badające zajętość

4.1.1. Implementacja modułu komunikacji z nadajnikiem LoRa

Moduł komunikacji z nadajnikiem stworzony został w formie klasy posiadającej cztery metody. Wykorzystany został do tego celu język programowania Python [27]. Do komuni-

4. Implementacja rozwiązania

kowania się z użyciem interfejsu UART zdecydowano na skorzystanie z biblioteki WiringPi [28]. Natomiast do formatowania wiadomości wysyłanych z pomocą LoRa użyto biblioteki PyCayenneLPP [29].

Pierwsza z funkcji **connect** pozwala na inicjację modułu LoRa, ustawienie poprawnego AppEui, wybranie odpowiedniego wykorzystywanego pasma częstotliwości, wybór obsługiwanych kanałów LoRa jak i sposobu dołączania do sieci, którym w tym wypadku jest OTAA. Sposób ten polega na tym, że urządzenie wysyła żądanie dołączenia do sieci LoRa poprzez przekazanie swojego unikalnego identyfikatora DevEUI oraz losowego numeru DevNonce, a następnie, po weryfikacji, otrzymuje klucze sesji do bezpiecznej komunikacji z siecią. Po konfiguracji funkcja wysyła do urządzenia komendy, które nakazują urządzeniu wysłać żądanie dołączenia do sieci. Próba dołączenia do sieci powtarzana jest tyle razy ile zostało to określone w argumencie funkcji.

Drugą funkcją jest natomiast funkcja **join** pozwalająca na pojedyncze przesłanie komendy do nadajnika z nakazem próby dołączenia do sieci LoRaWAN.

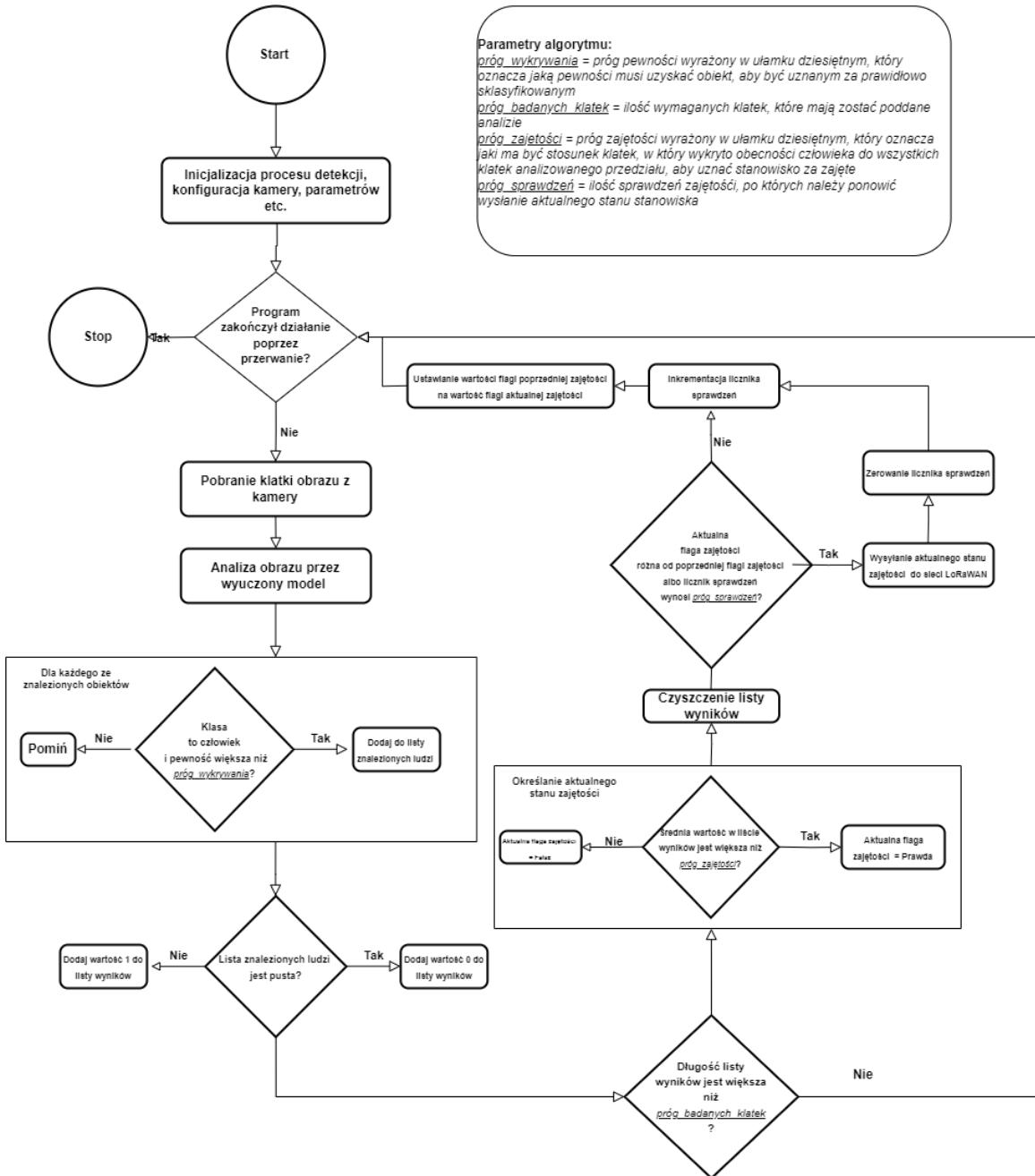
Trzecia funkcja **joinloop** pozwala powtórzyć przesłanie komendy o dołączeniu do sieci określoną w argumencie ilość razy. Po każdej próbie program odczuwa 10 sekund.

Ostatnią funkcją jest **sendstatus** pozwala ona przesłanie flagi zajętości określonej w argumencie do nadajnika LoRa, który to następnie wyśle wiadomości do bramki. Flaga umieszczana jest w ramce Lpp a następnie przekształcana do formy bitowej i przesyłana do nadajnika LoRa z komendą nakazującą przesłanie wiadomości do sieci LoRaWAN.

Cała treść klasy LoraConnector znajduje się w dodatku Załącznik 1.

4.1.2. Implementacja modułu wykrywania zajętości

Moduł przygotowano w formie skryptu, który wykonuje nieskończoną pętle i działa zgodnie ze schematem widocznym na rysunku 4.4. Jako bazę i wzór działania programu wykrywającego zajętość wykorzystano program przygotowany przez Alasdaira Allana [30]. Zmodyfikowano jego działanie tak, aby działał zgodnie z założeniami projektu. W gotowym skrypcie wykorzystano bibliotekę PiCamera 2 [31] do obsługi kamery połączonej z Raspberry Pi. Do manipulowania i przetwarzania obrazu uzyskanego z kamery wykorzystano bibliotekę OpenCV [32]. Do wykrywania obiektów na wstępnie przetworzonych obrazach z kamery wykorzystano bibliotekę TensorFlow Lite [33] wraz z wyuczonym modelem sieci neuronowej Mobilenet w wersji drugiej. Do komunikacji z nadajnikiem LoRa wykorzystano klasę LoraConnector opisaną w 4.1.1. Cała treść gotowego skryptu znajduje się w dodatku Załącznik 2



Rysunek 4.4. Algorytm działania modułu detekcji

4.1.3. Konfiguracja sieci VPN

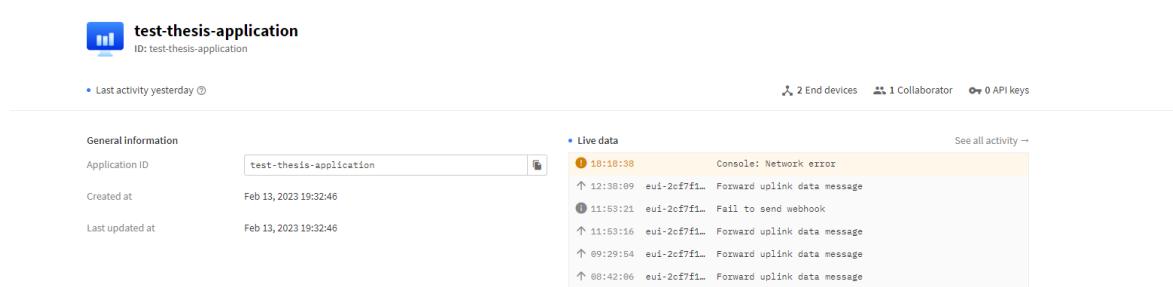
W celu zapewnienia zdalnego dostępu do zainstalowanego urządzenia w środowisku testowym zdecydowano się na wykorzystanie sieci VPN. Pozwala to na łączenie się z kontrolerem Raspberry Pi zdalnie z pomocą SSH. Jako dostawcę sieci VPN wybrano ZeroTier [34]. Po konfiguracji i stworzeniu sieci w aplikacji webowej dostawcy i zainstalowaniu klienta VPN na urządzeniu będącym czujnikiem zajętości i następnym dołączeniu do stworzonej sieci, możliwe się stało tworzenie sesji SSH pomiędzy urządzeniami połączonymi do tej

4. Implementacja rozwiązania

sieci niezależnie od tego do jakich sieci lokalnych podłączone są urządzenia próbujące nawiązać połaczenie.

4.2. Implementacja sieci LoRaWAN

Pierwszym krokiem było stworzenie aplikacji w konsoli chmury TTN. Utworzona została aplikacja o nazwie test–thesis–application co można obejrzeć na rysunku 4.5.



Rysunek 4.5. Utworzona aplikacja w chmurze TTN

Kolejnym krokiem było dodanie urządzeń i ich konfiguracja. Dodane i skonfigurowane urządzenia widać na rysunku 4.6.

The screenshot shows the TTN end device configuration interface. At the top, it says 'Applications > test-thesis-application > End devices'. Below that, there's a search bar, an 'Import end devices' button, and a 'Register end device' button. A table lists two end devices: 'eui-2cf7f1203230c670' and 'eui-2cf7f1203230d0a3'. Each device row contains columns for ID, Name, DevEUI, JoinEUI, and Last activity. The first device has a DevEUI of '2C F7 F1 20 32 30 C6 70' and a JoinEUI of '00 00 00 00 00 00 00 00'. Its last activity was '1 hr. ago'. The second device has a DevEUI of '2C F7 F1 20 32 30 08 A3' and a JoinEUI of '00 00 00 00 00 00 00 00'. Its last activity was 'Never'.

Rysunek 4.6. Skonfigurowane urządzenia

Następnym krokiem było utworzenie translatora wiadomości, którego zadaniem jest tłumaczenie wiadomości, które przychodzą z urządzeń końcowych z formatu binarnego na format, który dostarcza więcej informacji. Jeżeli wiadomość z urządzenia jest ma postać "000101" oznacza to, że dane urządzenie ma status zajętości oznaczony jako zajęty. W przypadku każdej innej wiadomości, status zajętości ustawiany jest jako wolny. Kod translatora widać na listingu 4.2.

```
1 function decodeUplink(input) {  
2     var decoded={};  
3     try{  
4         var result = String.fromCharCode.apply(null, input.bytes);  
5         if(result=='000101'){  
6             decoded.occupancy = true;  
7         }  
8     else{  
9         decoded.occupancy = false;  
10    }  
11    } catch (err) {  
12        decoded.message ='Decoder: ' + err.name + ' : ' + err.message;  
13    }  
14  
15    return {  
16        data: decoded,  
17        warnings: [] ,  
18        errors: []  
19    };  
20}
```

Listing 1. Translator wiadomości LoRa

Ostatnim krokiem w procesie konfigurowania sieci TTN było stworzenie integracji między siecią LoRaWAN a mikrourługą danych. Do tego celu posłużył jeden z komponentów dostępnych w chmurze TTN, którym jest moduł integracji w formie webhooku. Pozwala on na wykonywanie zapytań HTTP w określonych momentach, takich jak na przykład udane dołączenie do sieci przez urządzenie czy przesłanie wiadomości przez urządzenie. W przypadku tego projektu wykorzystano właśnie możliwość emitowania zapytań HTTP, gdy sieć otrzyma wiadomość od urządzenia końcowego. Wymagane jest określenie bazowej ścieżki i poszczególnych ścieżek dla każdego ze zdarzeń. Pełna konfiguracja utworzonego webhooka widoczna jest na rysunku 4.7

4. Implementacja rozwiązania

The screenshot shows the configuration interface for a webhook. At the top, there's a breadcrumb navigation: Applications > test-thesis-application > Webhooks > Edit. The main title is "Edit webhook". A descriptive text explains that webhooks allow sending application-related messages to specific HTTP(S) endpoints and scheduling downlinks to end devices, with a link to the "Webhooks guide".

General settings

- Webhook ID ***: azure-integration
- Webhook format ***: JSON
- Base URL ***: https://availabilityreportapi.azurewebsites.net/api
- Downlink API key**: (empty)
- Request authentication**: Use basic access authentication (basic auth)
- Additional headers**: + Add header entry
- Filter event data**: + Add filter path

Enabled event types

For each enabled event type an optional path can be defined which will be appended to the base URL.

- Uplink message**: /AvailabilityReport/TTN
An uplink message is received by the application
- Normalized uplink**: /AvailabilityReport/TTN
A normalized uplink payload

Rysunek 4.7. Skonfigurowany Webhook

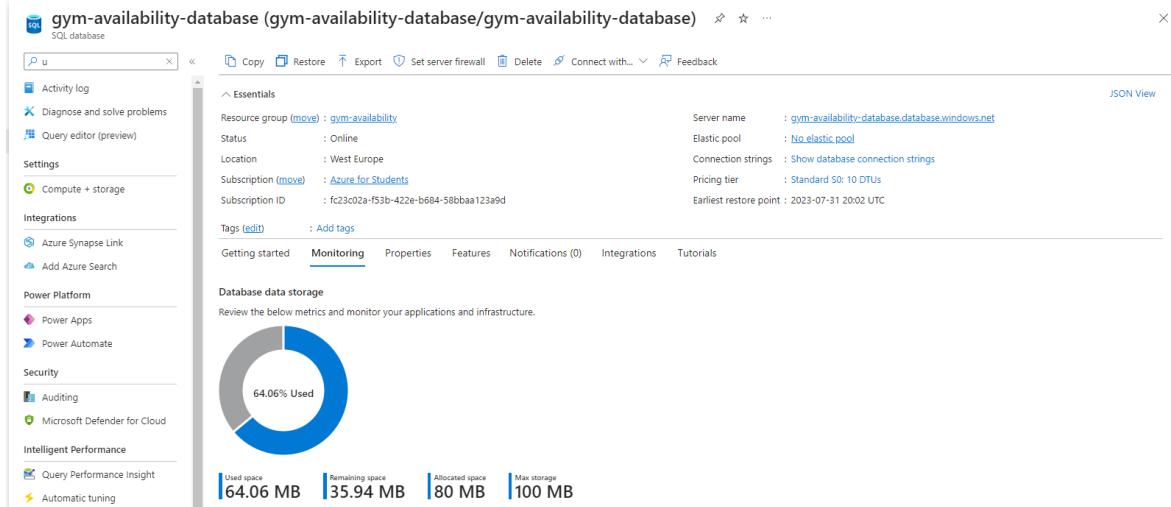
4.3. Implementacja części chmurowej

Implementacja części chmurowej wymagała utworzenia i konfiguracji poszczególnych usług na platformie Azure, a także zaprojektowania bazy danych i stworzenia poszczególnych aplikacji.

4.3.1. Baza danych

Dane o zajętości urządzeń oraz informację o salach, stanowiskach na siłowni czy o ustawieniu stanowiska przechowywane są w bazie danych opartej na silniku Microsoft SQL. Umieszczona została w chmurze z pomocą usługi Azure SQL Server. Na rysunku 4.8 widoczna jest utworzona baza danych w tej usłudze. Dla projektu przygotowano

sześć encji. W Tabelach 4.1-4.6 przedstawione zostały poszczególne encje wraz z opisami atrybutów.



Rysunek 4.8. Widok utworzonej bazy danych w usłudze Azure SQL Server

Tabela 4.1. Tabela Gyms

Atrybut	Typ	Wymagania	Opis
Id	int	Obowiązkowy - klucz	Number identyfikujący
Name	nvarchar	Obowiązkowy	Nazwa siłowni

Tabela 4.2. Tabela GymRooms

Atrybut	Typ	Wymagania	Opis
Id	int	Obowiązkowy - klucz	Number identyfikujący
Name	nvarchar	Obowiązkowy	Nazwa pokoju
Floor	nvarchar	Obowiązkowy	Nazwa piętra
RoomFileLink	nvarchar	Opcjonalny	Odnośnik do zdjęcia pokoju
GymId	int	Obowiązkowy - klucz obcy	Referencja do siłowni
createdAt	datetime	Opcjonalny	Znacznik czasu stworzenia
updatedAt	datetime	Opcjonalny	Znacznik czasu modyfikacji

4.3.2. Mikrousługa danych

Stworzono mikrousługę danych, która udostępnia dwa punkty końcowe. Cała aplikacja wdrożona została do chmury z pomocą usługi Azure App Service i dostępna jest pod adresem <https://availabilityreportapi.azurewebsites.net/api>. Widok wdrożonej aplikacji na platformie Azure pokazany jest na rysunku 4.9.

4. Implementacja rozwiązania

Tabela 4.3. Tabela Machines

Atrybut	Typ	Wymagania	Opis
Id	int	Obowiązkowy - klucz	Number identyfikujący
Name	nvarchar	Obowiązkowy	Nazwa Maszyny
Description	nvarchar	Opcjonalny	Opis maszyny
ImageFileLink	nvarchar	Opcjonalny	Odnośnik do zdjęcia maszyny
DeviceEUI	nvarchar	Opcjonalny	Numer devEUI czujnika
createdAt	datetime	Opcjonalny	Znacznik czasu stworzenia
updatedAt	datetime	Opcjonalny	Znacznik czasu modyfikacji

Tabela 4.4. Tabela MachinePlacements

Atrybut	Typ	Wymagania	Opis
Id	int	Obowiązkowy - klucz	Number identyfikujący
Description	nvarchar	Opcjonalny	Opis ustawienia
MachineId	int	Obowiązkowy - klucz obcy	Referencja do maszyny
GymRoomId	int	Obowiązkowy - klucz obcy	Referencja do sali

Tabela 4.5. Tabela AvailabilityReports

Atrybut	Typ	Wymagania	Opis
Id	int	Obowiązkowy - klucz	Number identyfikujący
MachineId	int	Obowiązkowy - klucz obcy	Referencja do maszyny
Timestamp	datetime	Opcjonalny	Znacznik czasu wystąpienia stanu
CurrentState	nvar	Obowiązkowy	Bieżący stan maszyny
PreviousState	nvar	Opcjonalny	Poprzedni stan maszyny

Tabela 4.6. Tabela AvailabilityReportFactSt

Atrybut	Typ	Wymagania	Opis
Id	int	Obowiązkowy - klucz	Number identyfikujący
Occupancy	nvarchar	Opcjonalny	Stan maszyny
MachineId	int	Obowiązkowy - klucz obcy	Referencja do maszyny
GymRoomId	int	Obowiązkowy - klucz obcy	Referencja do sali
Timestamp	time	Opcjonalny	Znacznik czasu wystąpienia stanu
Date	datetime	Opcjonalny	Data wystąpienia stanu
FactId	datetime	Obowiązkowy - klucz obcy	Referencja do raportu

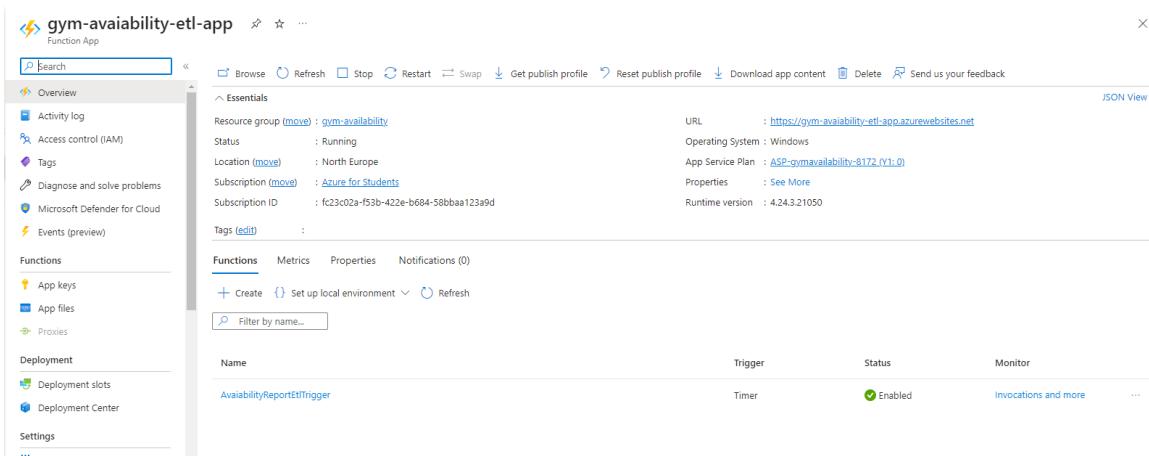
Rysunek 4.9. Mikrorusługa danych w usłudze Azure App Service

Pierwszym udostępnianym punktem końcowym jest punkt **/AvaiabilityReport/TTN**. Korzysta on z metody POST i odpowiedzialny jest on za umieszczanie raportów o dostępności w bazie danych. To właśnie on jest wywoływany przez moduł Webhook sieci TTN. Najpierw pobiera on dane o maszynie w bazie danych, do której przypisany jest numer devEUI urządzenia z którego przyszła wiadomość nowym stanie. Następnie tworzony jest nowy wpis w tabeli AvaiabilityReports.

Drugim udostępnionym punktem końcowym jest punkt **/AvaiabilityReport**. Korzysta on z metody GET i odpowiada za uruchomienie procesu przekształcania danych z tabeli AvaiabilityReports i umieszczenia ich w tabeli AvaiabilityReportFactSt w taki sposób aby dla każdej minuty od godziny 6 do godziny 23 w dniu uruchomienia procesu pojawiały się wpisy z oznaczeniem zajętości każdej maszyny. Potrzeba przetwarzania raportów zaistniała, gdyż urządzenia raportują stan swojej zajętości tylko w przypadku zmiany jej stanu, bądź też gdy aktualny stan utrzymuje się przez dłuższy czas. Dane w tak nieregularnej postaci są trudne do przedstawienia na wykresie. Kod całej aplikacji dostępny jest w załączniku Załącznik 3.

4.3.3. Aplikacja wyzwalająca przetwarzanie danych

Do stworzenia aplikacji wyzwalającej przetwarzanie danych wykorzystano usługę Azure Functions. Powstała w tym celu prosta aplikacja, która wykonuje zapytanie na punkt końcowy **/AvaiabilityReport** mikrousługi danych codziennie o godzinie 23. Wdrożona aplikacja w chmurze Azure widoczna jest na rysunku 4.10. Kod całej aplikacji dostępny jest w załączniku Załącznik 4.



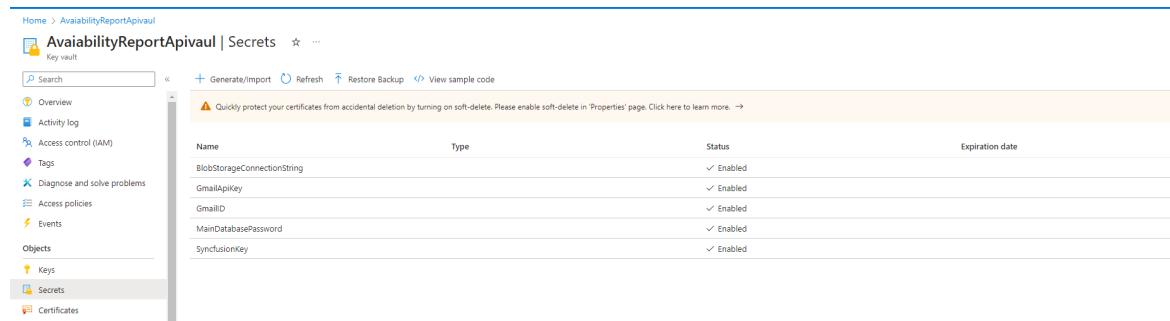
Rysunek 4.10. Widok aplikacji wyzwalającej przetwarzanie danych w usłudze Azure Functions

4.3.4. Menadżer sekretów

W celu przetrzymywania wrażliwych danych takich jak hasło do bazy danych, czy klucz do konta e-mail używanego do potwierdzania kont użytkowników utworzono menadżer

4. Implementacja rozwiązania

sekretów w postaci usługi Azure Key Vault. Na rysunku 4.11 widoczny jest skonfigurowany menadżer wraz z przechowywanymi sekretami.



The screenshot shows the Azure Key Vault interface for the 'AvailabilityReportApivaul' vault. On the left, there's a navigation sidebar with options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Access policies, Events, Objects, Keys, Secrets, and Certificates. The 'Secrets' option is selected. The main area displays a table of secrets:

Name	Type	Status	Expiration date
BlobStorageConnectionString		✓ Enabled	
GmailApiKey		✓ Enabled	
GmailID		✓ Enabled	
MainDatabasePassword		✓ Enabled	
SyncfusionKey		✓ Enabled	

Rysunek 4.11. Widok usługi Azure Key Vault

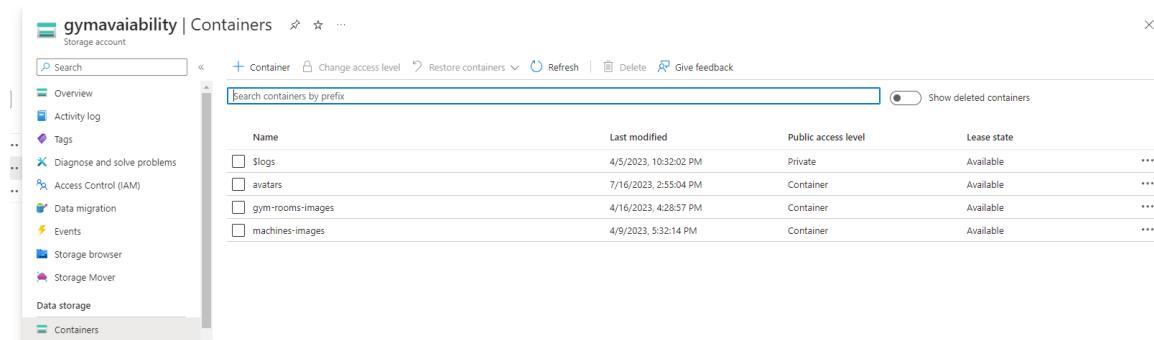
Aby aplikacje miały możliwość wydobycia wartości z menadżera sekretów, należało skonfigurować moduł łączący się z nim. Jego konfiguracja polegała na inicjalizacji poprzez dodanie fragmentu kodu widocznego na listingu 4.3.4 do pliku Program.cs w przypadku mikrousługi danych, jaki i aplikacji webowej.

- 1 var keyVaultEndpoint = new Uri(Environment.GetEnvironmentVariable("VaultUri"));
- 2 builder.Configuration.AddAzureKeyVault(keyVaultEndpoint,
- 3 new DefaultAzureCredential());

Listing 2. Konfiguracja komunikacji z menadżerem sekretów

4.3.5. Magazyn plików

Aplikacja webowa wymaga magazynowania plików takich jak zdjęcia maszyn, sal na siłowni, czy avatarów użytkowników. Do tego celów wykorzystano usługę Azure Blob Storage. Na rysunku 4.12 widoczny jest skonfigurowany magazyn wraz z utworzonymi folderami zwartymi kontenerami, które służą do przechowywania plików.



The screenshot shows the Azure Blob Storage interface for the 'gymavailability' storage account. On the left, there's a navigation sidebar with options like Overview, Activity log, Tags, Diagnose and solve problems, Access Control (IAM), Data migration, Events, Storage browser, Storage Mover, Data storage, and Containers. The 'Containers' option is selected. The main area displays a table of containers:

Name	Last modified	Public access level	Lease state	...
Slogs	4/5/2023, 10:32:02 PM	Private	Available	...
avatars	7/16/2023, 2:55:04 PM	Container	Available	...
gym-rooms-images	4/16/2023, 4:28:57 PM	Container	Available	...
machines-images	4/9/2023, 5:32:14 PM	Container	Available	...

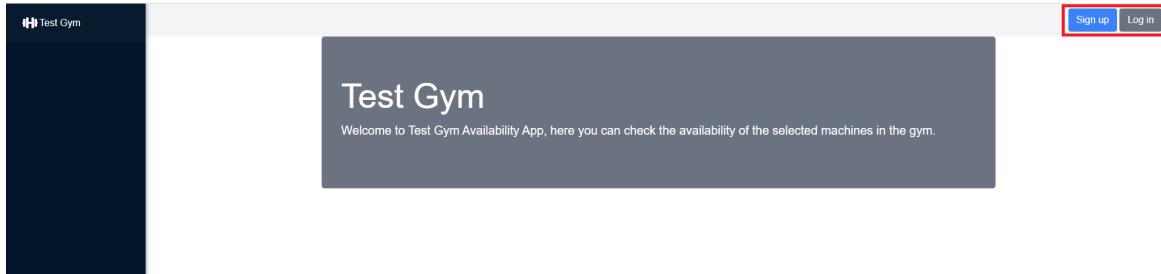
Rysunek 4.12. Widok usługi Azure Blob Storage

4.4. Aplikacja webowa

Stworzona została aplikacja webowa spełniająca wymagania postawione w rozdziale 3.1.2. Jako główny framework służący do budowy aplikacji wykorzystano ASP.NET Core Blazor. Utworzony serwis zapewnia dwie perspektywy dostępu, jedną dla zwykłych użytkowników siłowni, drugą dla administratorów/pracowników siłowni. Użytkownicy są w stanie logować się, rejestrować, zmieniać dane konta, wyświetlać informacje o salach i sprawdzać statusy urządzeń. Pracownicy są dodatkowo w stanie dodawać i modyfikować informacje o salach jak i maszynach.

4.4.1. Proces uwierzytelniania

Z poziomu strony głównej aplikacji, użytkownik ma do wyboru dwa przyciski, jeden służący do logowania z napisem “Log in” i drugi “Sign up” pozwalający na rejestrację. Do obsługi tych dwóch akcji skorzystano z framework'u ASP.NET Core Identity [35]. Na rysunku 4.13 widoczna jest strona główna z zaznaczonymi czerwoną ramką wspomnianymi przyciskami.



Rysunek 4.13. Widok ekranu głównego aplikacji

Na rysunku 4.14 i rysunku 4.15 widoczne są odpowiednio ekran logowania jak i rejestracji. W przypadku obu użytkownik musi wprowadzić adres email i hasło. Po ich wprowadzeniu podczas rejestracji i naciśnięciu przycisku “Register” użytkownikowi pokazywana jest wiadomość widoczna na rysunku 4.16. Informuje ona o potrzebie potwierdzenia konta poprzez wejście w link, który znajduje się w wiadomości email, która trafia na adres wskazany przez użytkownika. Jej treść widoczna jest na rysunku 4.17. Do wysyłania wiadomości do użytkowników wykorzystano konto Gmail, które pozwala na wysyłanie maili z pomocą aplikacji. Utworzona została do tego celu klasa EmailSenderService.cs. Jej treść jest dostępna w repozytorium aplikacji dostępnym w załączniku Załącznik 5.

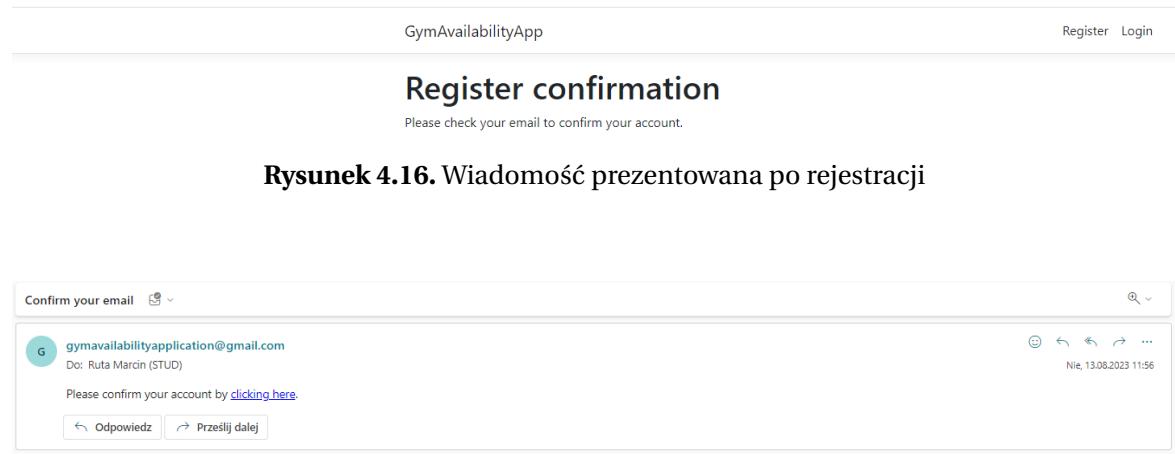
4. Implementacja rozwiązania

The screenshot shows the 'Log in' page of the GymAvailabilityApp. At the top right are 'Register' and 'Login' links. Below them is a heading 'Log in' and a sub-instruction 'Use a local account to log in.' There are two input fields: 'Email' containing 'ulyon@gmail.com' and 'Password' containing several dots. A 'Remember me?' checkbox is checked. A large blue 'Log in' button is centered below the inputs. Below the button are three links: 'Forgot your password?', 'Register as a new user', and 'Resend email confirmation'.

Rysunek 4.14. Widok ekranu logowania

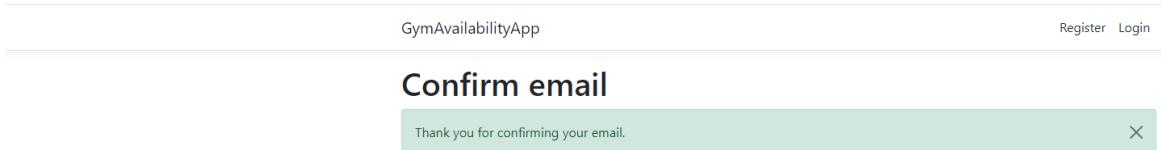
The screenshot shows the 'Register' page of the GymAvailabilityApp. At the top right are 'Register' and 'Login' links. Below them is a heading 'Register' and a sub-instruction 'Create a new account.' There are three input fields: 'Email', 'Password', and 'Confirm Password'. A large blue 'Register' button is centered below the fields.

Rysunek 4.15. Widok ekranu rejestracji



Rysunek 4.17. Treść wiadomości email

Wchodząc w link znajdujący się w mailu użytkownik przekierowany jest do strony przedstawiającej komunikat o poprawnej rejestracji, widoczna jest ona na rysunku 4.18, a samo konto użytkownika zostaje zweryfikowane i potwierdzone w bazie danych.

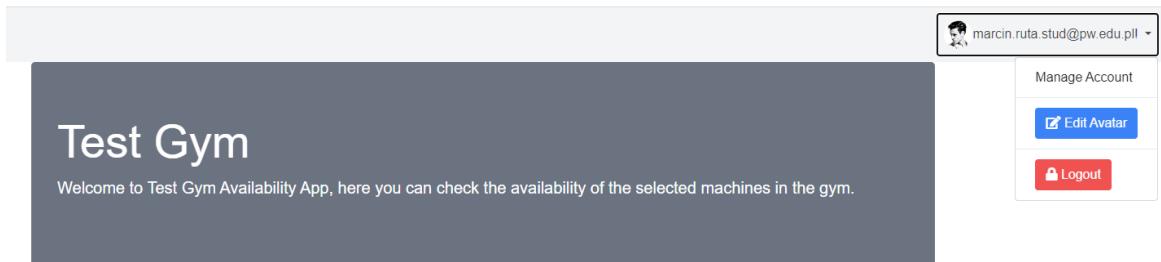


Rysunek 4.18. Widok strony z komunikatem o poprawnej rejestracji

Następnie użytkownik może się zalogować korzystając z danych wykorzystanych podczas rejestracji. Po ich wprowadzeniu i naciśnięciu przycisku “Log in” kierowany jest na stronę główną aplikacji.

4.4.2. Zarządzanie kontem

Użytkownik z poziomu ekranu głównego jest w stanie przejść do sekcji zarządzania kontem. Po kliknięciu w nazwę użytkownika, bądź też awatar w prawym górnym rogu aplikacji, użytkownikowi prezentowana jest lista przycisków odpowiedzialnych kolejno za: przejście do sekcji zarządzania, otwarcie dialogu zmiany awatara, wylogowanie się. Lista ta widoczna jest na rysunku 4.19.



Rysunek 4.19. Widok opcji dotyczących konta

Wciśnięcie przycisku “Manage Account” skutkuje przekierowaniem użytkownika do widoku aplikacji przedstawionego na rysunku 4.20 pozwalającego na zmianę hasła, adresu email przypisanego do konta czy wpisaniem, bądź zmianą numeru telefonu.

4. Implementacja rozwiązania

GymAvailabilityApp

Hello marcin.ruta.stud@pw.edu.pl! Logout

Manage your account

Change your account settings

Profile

Email

Password

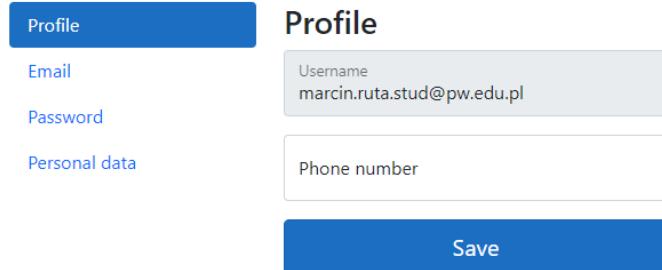
Personal data

Profile

Username
marcin.ruta.stud@pw.edu.pl

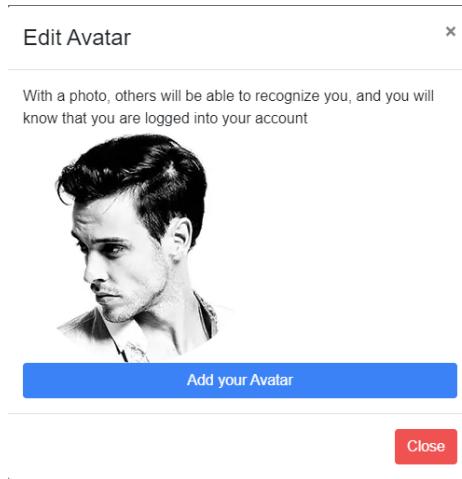
Phone number

Save



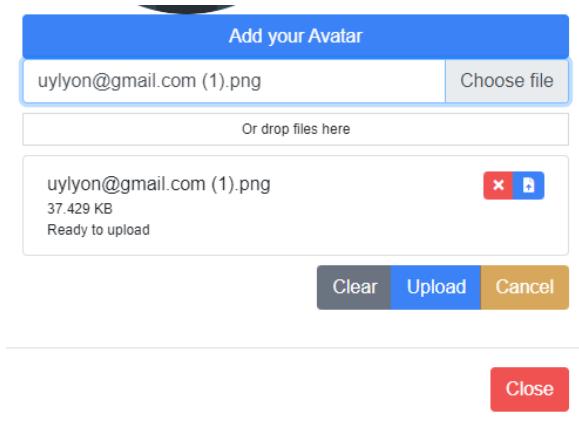
Rysunek 4.20. Widok zarządzania kontem

Po naciśnięciu przycisku “Edit Avatar“ otwierany jest dialog edycji awatara. Jest on widoczny na rysunku 4.21



Rysunek 4.21. Dialog edycji awatara

Po naciśnięciu przycisku “Add your Avatar“ pojawia się pole pozwalające na umieszczenie pliku zawierające zdjście, które użytkownik chciałby umieścić w aplikacji. Po wybraniu pliku z dysku, użytkownik może usnąć wybrany plik, albo zdecydować się na przesłanie pliku co będzie równoznaczne zmianie awatara. Na rysunku 4.22 widoczny jest widok po wybraniu pliku będącego awatarem. Przycisk “Upload“ odpowiada za przesłanie awataru na serwer.

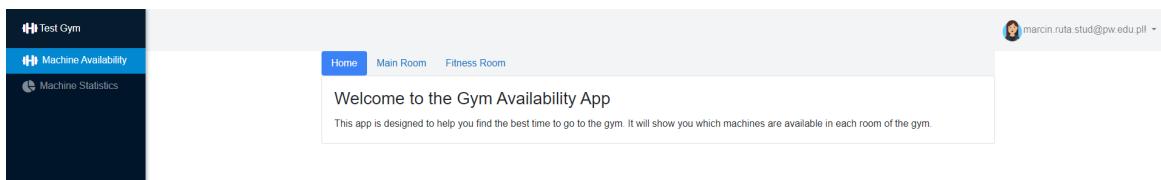


Rysunek 4.22. Przesłanie pliku zawierającego awatar

Wciśnięcie przycisku “Logout“ powoduje wylogowanie użytkownika i przeładowanie strony.

4.4.3. Przeglądanie listy sal i urządzeń

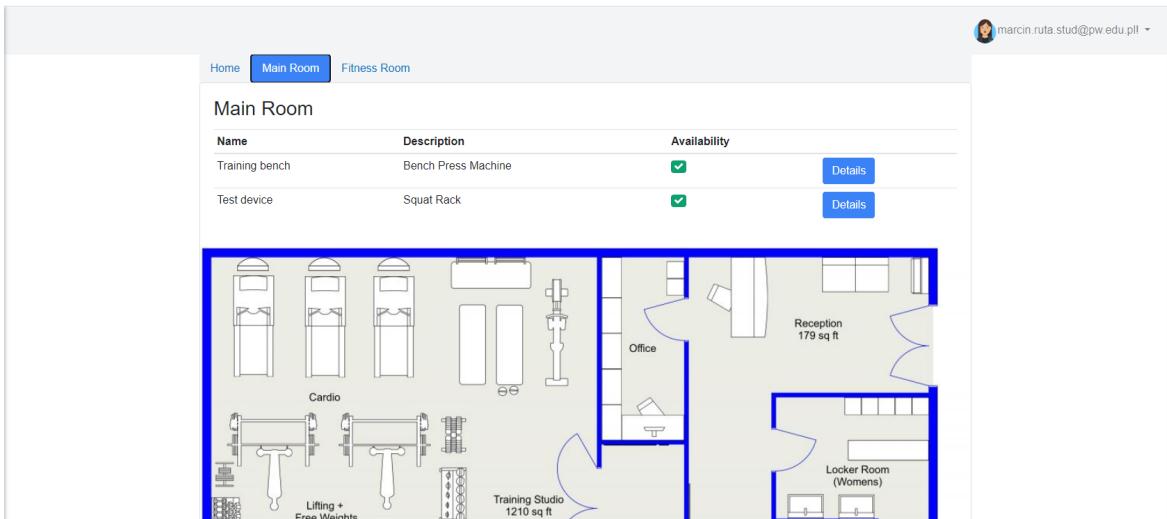
Użytkownik jest w stanie przejść do widoku aplikacji, który prezentuje każdą z sal dostępnych na siłowni, wraz z urządzeniami dostępnymi w poszczególnych salach. Do przejścia do tego widoku służy przycisk “Machine Availability“ znajdujący się w menu nawigacyjnym po lewej stronie aplikacji. Na rysunku 4.23 widoczny jest ekran główny widoku pozwalającego na przeglądanie listy sal.



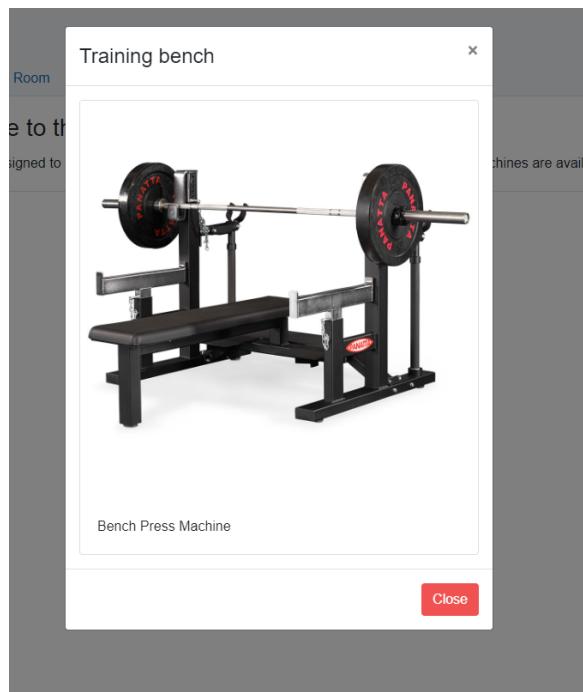
Rysunek 4.23. Podstawowa sekcja widoku przeglądania listy sal

W środkowej części użytkownik może wybierać pomiędzy zakładkami oznaczającymi poszczególne sale znajdujące się na siłowni. Na rysunku 4.24 przedstawiony jest ekran aplikacji po wybraniu jednej z sal. Widnieją na nim urządzenia przypisane do danej sali wraz z ich opisem i stanem zajętości widocznym w sekcji Availability. Użytkownik może podejrzeć szczegóły każdego z urządzeń klikając przycisk “Details“. Szczegóły urządzenia widoczne są na rysunku 4.25.

4. Implementacja rozwiązania



Rysunek 4.24. Widok przedstawiający wybraną salę



Rysunek 4.25. Widok przedstawiający szczegóły maszyny

4.4.4. Zarządzanie siłownią

Użytkownicy, którzy posiadają rolę administratora mają dostęp do widoku pozwalającego na dodawanie, edytowanie oraz usuwanie sal i maszyn znajdujących się w siłowni. Do przejścia do tego widoku służy przycisk "Manage Gym" znajdujący się w menu nawigacyjnym po lewej stronie aplikacji. Na rysunku 4.26 widoczny jest ekran widoku pozwalającego na zarządzanie siłownią, a konkretnie sekcja zarządzania salami.

#	Name	Floor	Image
2	Main Room	Second Floor	✓
3	Fitness Room	First Floor	✓

Rysunek 4.26. Widok zarządzania salami

Użytkownik jest w stanie dodać salę po naciśnięciu przycisku “New”, jest to widoczne na rysunku 4.27. Kolejną możliwą akcją jest wybór konkretnej sali i możliwość jej usunięcia przez przycisk “Delete”, edycji po naciśnięciu przycisku “Edit” co jest widoczne na rysunku 4.28, jak i dodanie bądź edycja zdjęcia sali poprzez przycisk “Add Image”. Jego przyciśnięcie otwiera dialog przypominający ten znajdujący się w widoku zmiany awatara. Widoczny jest on na rysunku 4.29.

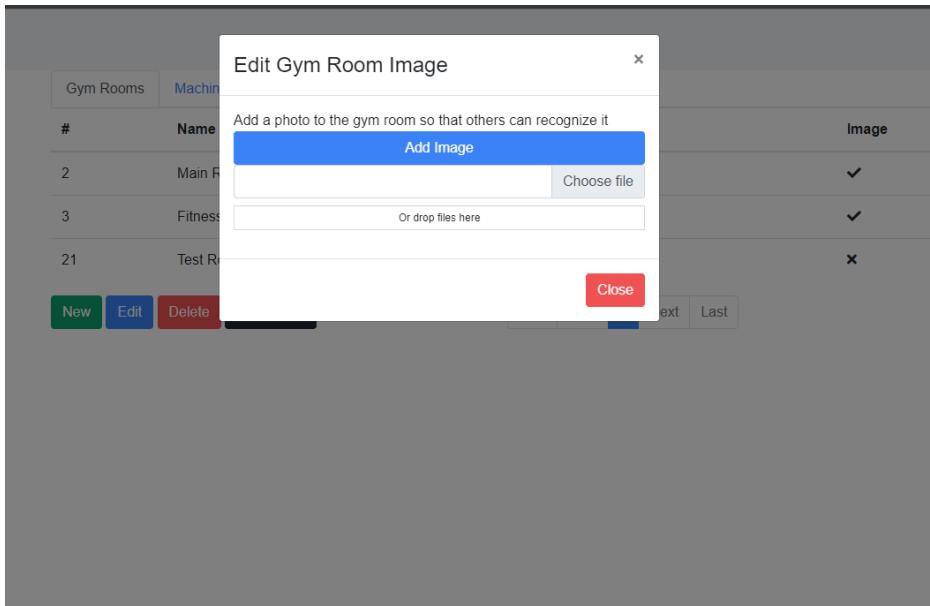
#	Name	Floor	Image
	Name	Floor	
	<input type="text"/>	<input type="button" value="First Floor"/>	

Rysunek 4.27. Widok dodawania sali

#	Name	Floor	Image
2	Main Room	Second Floor	✓
3	Fitness Room	First Floor	✓

Rysunek 4.28. Widok edytowania sali

4. Implementacja rozwiązania



Rysunek 4.29. Dialog edycji zdjęcia sali

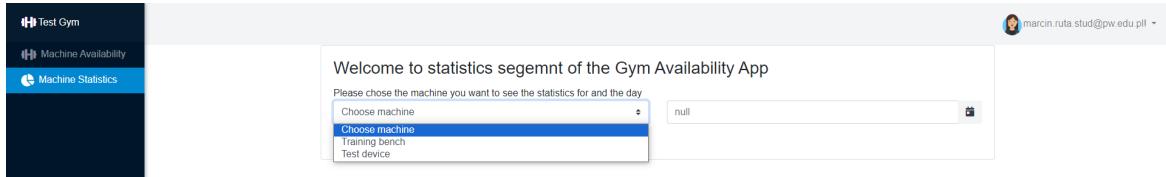
Analogiczna sekcja z takimi samymi funkcjonalnościami jak dla zarządzania salami została przygotowana do celów zarządzania maszynami. Jest ona widoczna na rysunku 4.30.

#	Name	Device EUI	Description	Gym Room	Image	Placement
1	Training bench	2CF7F1203230D0A3	Bench Press Machine	Main Room	✓	Near window
2	Test device	2CF7F1203230C670	Squat Rack	Main Room	✓	Near door

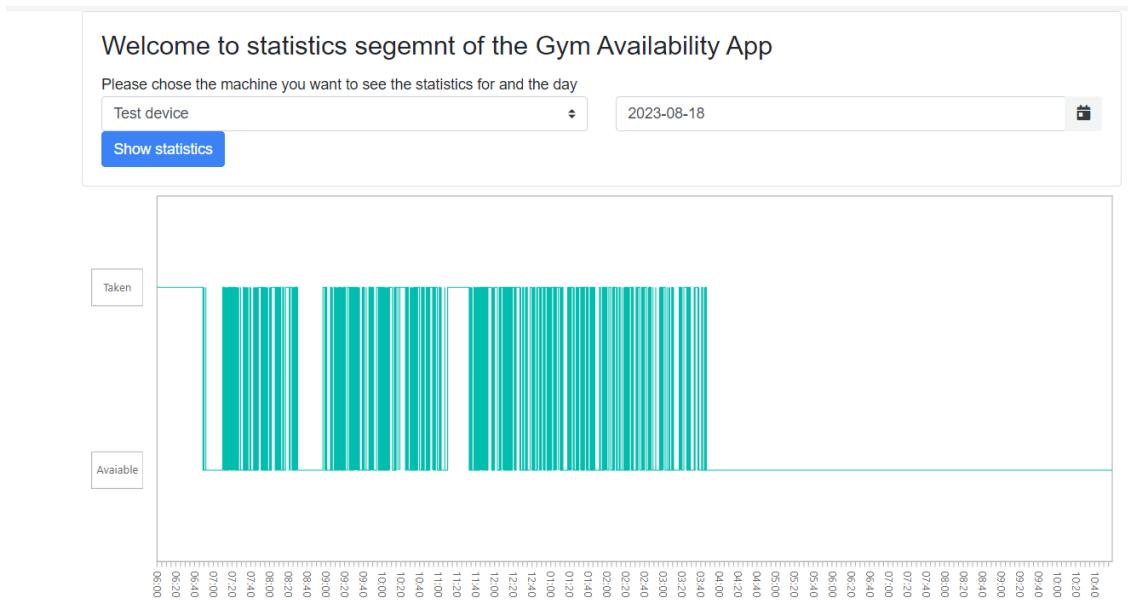
Rysunek 4.30. Widok zarządzania maszynami

4.4.5. Przeglądanie wykresów zajętości

Użytkownik jest w stanie przejść do widoku aplikacji, który pozwala na przeglądanie wykresów zajętości urządzeń w konkretnym dniu. Do przejścia do tego widoku służy przycisk "Machine Statistics" znajdujący się w menu nawigacyjnym po lewej stronie aplikacji. Na rysunku 4.31 widoczny jest ekran pozwalający na wybór konkretnej maszyny, jak i daty, dla których wygenerowany ma zostać wykres. Wygenerowany wykres widoczny jest natomiast na rysunku 4.32.



Rysunek 4.31. Widok wyboru urządzenia i daty



Rysunek 4.32. Widok wygenerowanego wykresu

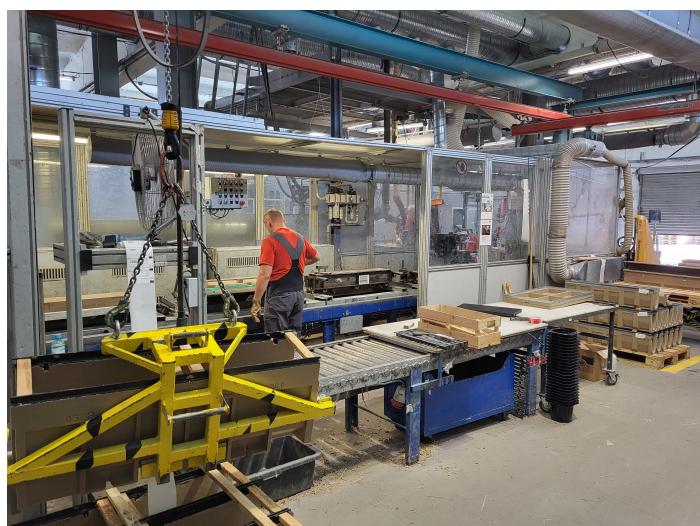
5. Testy przeprowadzone w środowisku

Przygotowany system poddany został testom, które miały na celu sprawdzić poprawność jego działania poprzez montaż urządzenia sprawdzającego zajętość w środowisku rzeczywistym. Dzięki temu sprawdzono możliwości urządzenia i tego jak skutecznie jest w stanie wykrywać obecność człowieka, oraz to czy wiadomości o statusie stanowiska poprawnie przesyłane są przez sieć LoRaWAN i przekazywane do bazy danych, a następnie wyświetlane w aplikacji webowej.

5.1. Stanowisko pomiarowe

Pierwszym docelowym miejscem, w którym planowano umieścić urządzenie była siłownia. Skontaktowano się z kilkoma placówkami, natomiast wystąpił problem w postaci niechęci użytkowników siłowni do bycia obserwowanym przez kamery, co spowodowało odmowę przeprowadzenia eksperymentów. Stąd zdecydowano się na wybór innego miejsca prowadzenia doświadczenia. Urządzenie służące do wykrywania zajętości zdecydowano zamontować na hali produkcyjnej jednego z przedsiębiorstw zajmującego się budową konstrukcji wykorzystywanych do odwadniania. Podobnie jak w problemie opisany w tej pracy dotyczącym potrzeby sprawdzania zajętości urządzeń na siłowniach, tak na wspomnianej hali istnieje potrzeba sprawdzania obecności pracowników przy jednym ze stanowisk. Sam mechanizm działania systemu jest tożsamy, stąd decyzja o montażu urządzenia w budynku przedsiębiorstwa. Dodatkowo pozwala to przetestować projekt oraz prototyp w szerszym kontekście do zastosowań przemysłowych jako system do sprawdzania ogólnej zajętości, a nie tylko zajętości urządzeń na siłowni.

Urządzenie zawieszono na przeciwko stanowiska wymagającego obserwacji. Widoczne jest ono na rysunku 5.1. Sposób montażu urządzenia i jego lokalizacja pokazana jest na rysunku 5.2. Obraz z kamery czujnika przedstawiony jest na rysunku 5.3.

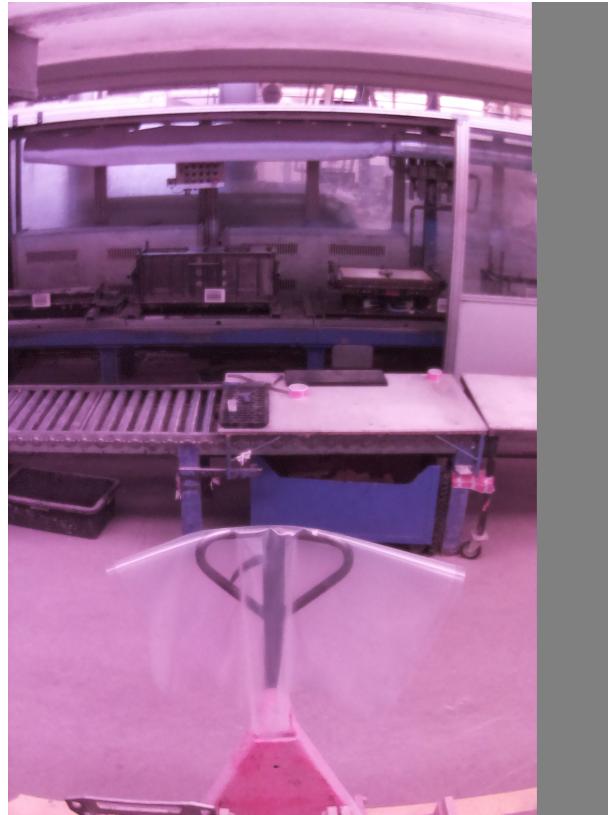


Rysunek 5.1. Stanowisko na hali produkcyjnej

5. Testy przeprowadzone w środowisku



Rysunek 5.2. Zamontowane na hali urządzenie

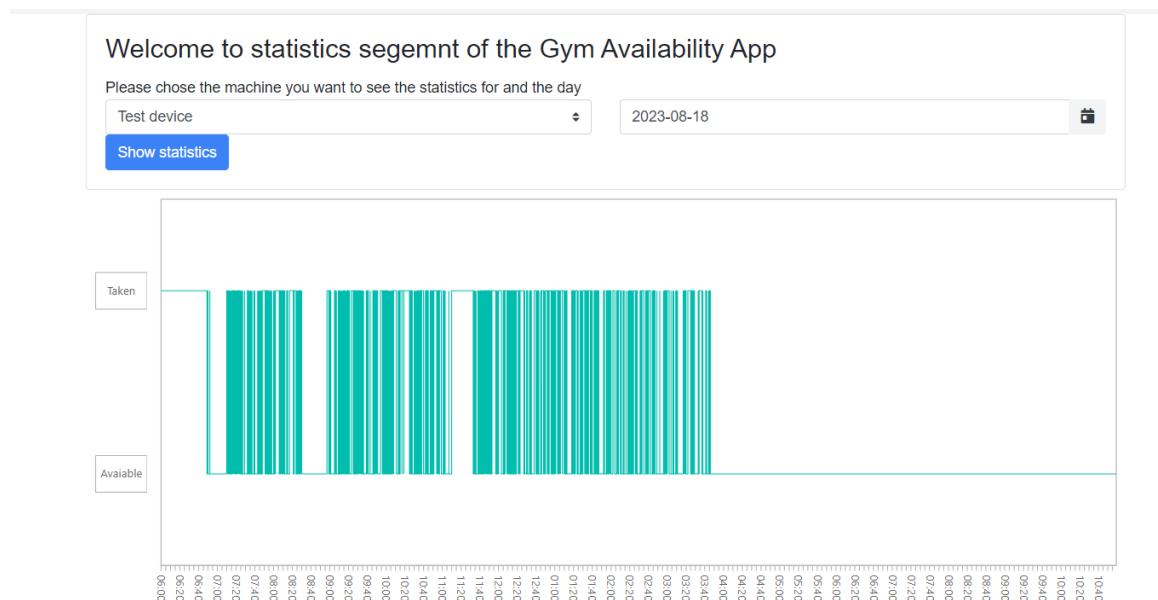


Rysunek 5.3. Obraz z kamery czujnika

5. Testy przeprowadzone w środowisku

5.2. Wyniki eksperymentu

W wyniku zamontowania urządzenia na hali produkcyjnej powstały dane, które można obserwować na wykresie w aplikacji webowej. Widoczny jest on na rysunku 5.4. Widać na nim ciągłe zmieniającą się zajętość stanowiska, a także dłuższy interwał w okolicach godziny 10, podczas którego stanowisko nie wykrywana była obecność człowieka. W tym samym czasie pracownicy na hali mają przerwę, co oznacza że system poprawnie zadziałał oznaczając stanowisko jako puste. Ciągłe zmiany zajętości wynikają z przemieszania się pracowników po stanowisku, wychodzenia poza obszar kamery, a także z niedoskonałości algorytmu wykrywania, który momentami błędnie określa zmianę stanu.



Rysunek 5.4. Wykres na podstawie zebranych wyników

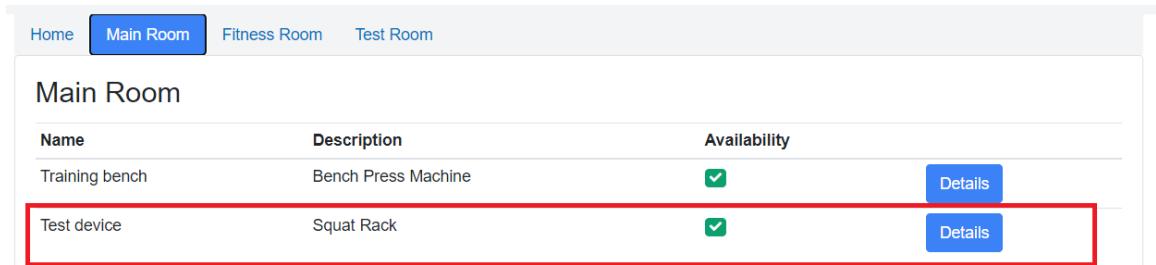
5.3. Przeprowadzone testy

5.3.1. Test poprawnego zmiany statusu zajętości

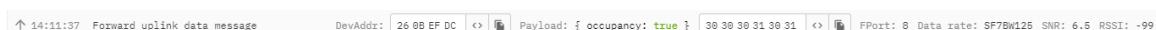
Scenariusz przeprowadzonego testu wymagał wykonania następujących kroków:

1. Zalogowanie się na konto użytkownika w aplikacji webowej.
2. Sprawdzenie aktualnego stanu urządzenia w aplikacji webowej.
3. Zmiana stanu urządzenia poprzez pojawienie się osoby w badanym miejscu.
4. Sprawdzenie dziennika logów sieci LoRaWAN.
5. Sprawdzenie wpisu w bazie danych.
6. Ponowne sprawdzenie aktualnego stanu urządzenia w aplikacji webowej.

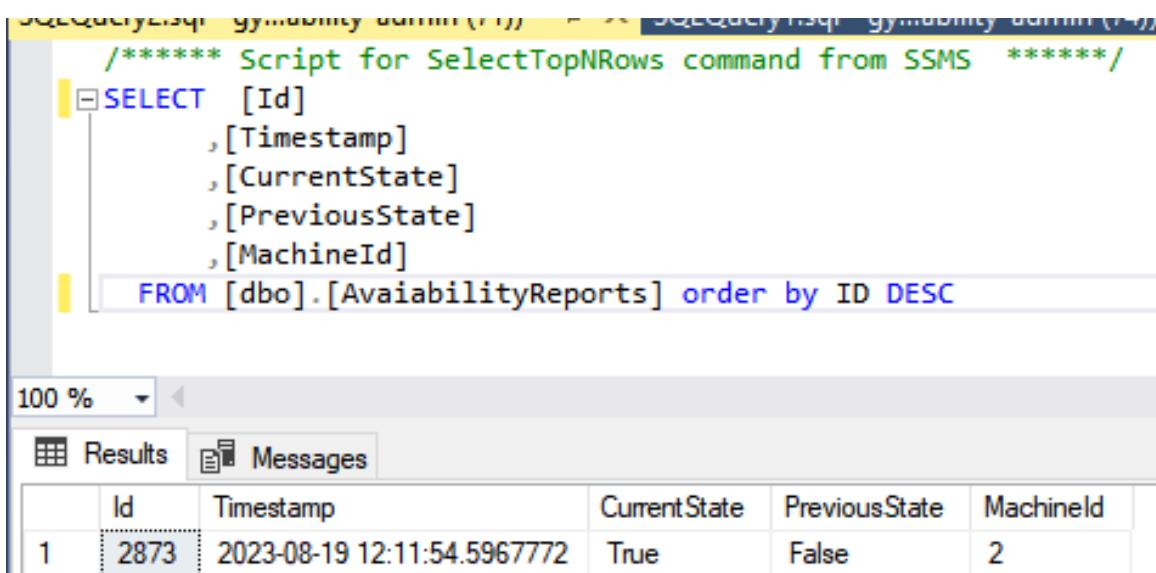
Na rysunkach 5.4 - 5.7 przedstawione zostały wyniki poszczególnych kroków opisywanego scenariusza.



Name	Description	Availability	
Training bench	Bench Press Machine	<input checked="" type="checkbox"/>	<button>Details</button>
Test device	Squat Rack	<input checked="" type="checkbox"/>	<button>Details</button>

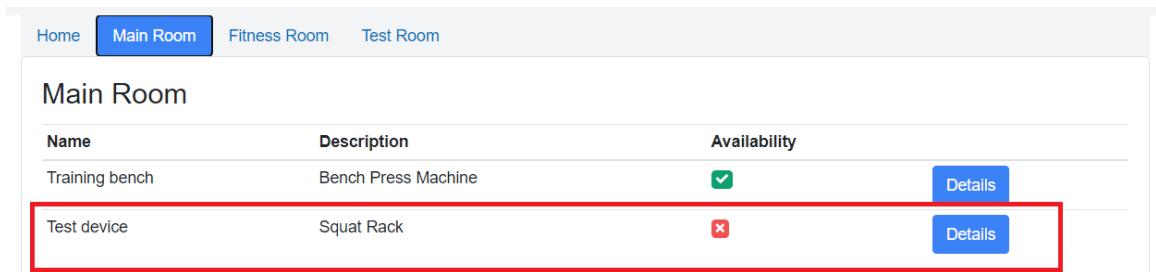
Rysunek 5.5. Początkowy stan urządzenia


↑ 14:11:37 Forward uplink data message DevAddr: 26 0B EF DC <> Payload: { occupancy: true } 30 30 30 31 30 31 <> FPort: 8 Data rate: SF7BW125 SNR: 6.5 RSSI: -99

Rysunek 5.6. Sprawdzenie dziennika logów sieci LoRaWAN


```
***** Script for SelectTopNRows command from SSMS *****
SELECT [Id]
      ,[Timestamp]
      ,[CurrentState]
      ,[PreviousState]
      ,[MachineId]
 FROM [dbo].[AvailabilityReports] order by ID DESC
```

	Id	Timestamp	CurrentState	PreviousState	MachineId
1	2873	2023-08-19 12:11:54.5967772	True	False	2

Rysunek 5.7. Sprawdzenie wpisu w bazie danych


Name	Description	Availability	
Training bench	Bench Press Machine	<input checked="" type="checkbox"/>	<button>Details</button>
Test device	Squat Rack	<input type="checkbox"/>	<button>Details</button>

Rysunek 5.8. Końcowy stan urządzenia w aplikacji webowej

Test zakończył się powodzeniem zgodnie z oczekiwany wynikiem. Aplikacja webowa była w stanie poprawnie wyświetlić zmieniony stan.

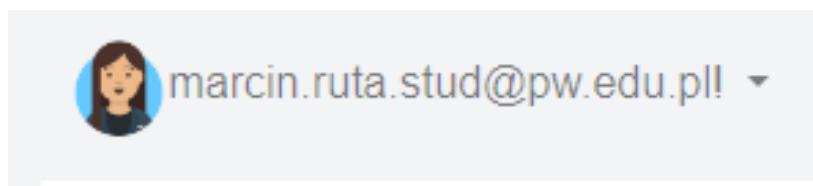
5. Testy przeprowadzone w środowisku

5.3.2. Test poprawnej zmiany awatara

Scenariusz przeprowadzonego testu wymagał wykonania następujących kroków:

1. Zalogowanie się na konto użytkownika w aplikacji webowej.
2. Sprawdzenie aktualnego avatara w aplikacji webowej.
3. Sprawdzenie pliku zawierającego avatar w magazynie plików.
4. Zmiana avatara z pomocą edytora avatarów.
5. Wyczyszczenie cookies w przeglądarce.
6. Sprawdzenie avatara po zmianie w aplikacji webowej.
7. Sprawdzenie pliku avatara w magazynie plików po zmianie.

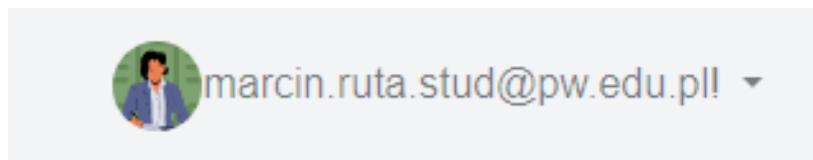
Na rysunkach 5.8 - 5.12 przedstawione zostały wyniki poszczególnych kroków opisywanego scenariusza.



Rysunek 5.9. Początkowy awatar w aplikacji webowej

A screenshot of the Azure Storage Blobs interface. The left sidebar shows a tree structure with "Container" selected under "avatars". The main area displays a list of blobs. A single blob is selected, showing its details: Name: "marcin.ruta.stud@pw.edu.pl.png", Modified: "8/13/2023, 2:25:59 PM", Access tier: "Hot (Inferred)", Archive status: "Not yet archived", Blob type: "Block blob", Size: "37.43 KB", Lease state: "Available". A red box highlights the "Modified" timestamp.

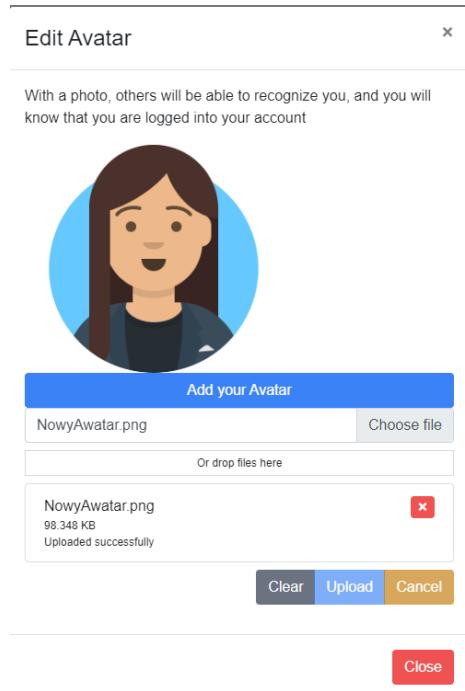
Rysunek 5.10. Początkowy awatar w magazynie plików



Rysunek 5.11. Awatar w aplikacji webowej po zmianach

A screenshot of the Azure Storage Blobs interface, identical to Rysunek 5.10 in terms of the UI and blob list. The selected blob's details are shown: Name: "marcin.ruta.stud@pw.edu.pl.png", Modified: "8/19/2023, 3:06:07 PM", Access tier: "Hot (Inferred)", Archive status: "Not yet archived", Blob type: "Block blob", Size: "98.35 KB", Lease state: "Available". A red box highlights the "Modified" timestamp.

Rysunek 5.12. Awatar po zmianie w magazynie plików



Rysunek 5.13. Dialog zmiany awatara

Test zakończył się powodzeniem zgodnie z oczekiwany wynikiem. Zmiana awatara zakończyła się powodzeniem.

5.4. Test dodawania i edycji urządzeń

Scenariusz przeprowadzonego testu wymagał wykonania następujących kroków:

1. Zalogowanie się na konto administratora w aplikacji webowej.
2. Przejście do sekcji zarządzania siłownią.
3. Dodanie nowego urządzenia.
4. Sprawdzenie wpisu w bazie danych dotyczącego utworzonego urządzenia.
5. Edycja utworzonego urządzenia.
6. Sprawdzenie wpisu w bazie danych dotyczącego edytowanego urządzenia.
7. Usunięcie poprzednio utworzonego urządzenia.
8. Potwierdzenie usunięcia wpisu w bazie danych.

Na rysunkach 5.13 - 5.18 przedstawione zostały wyniki poszczególnych kroków opisywanego scenariusza.

5. Testy przeprowadzone w środowisku

Gym Rooms Machines

#	Name	Floor	Image
	Name	Floor	
	<input type="text"/>	First Floor	<input type="button" value="▼"/>

Save Cancel

Rysunek 5.14. Dodanie nowej maszyny w aplikacji webowej

```
***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP (1000) [Id]
    ,[Name]
    ,[ImageFileLink]
    ,[Description]
    ,[DeviceEUI]
    ,[createdAt]
    ,[updatedAt]
FROM [dbo].[Machines]
```

Id	Name	ImageFileLink	Description	DeviceEUI	createdAt	updatedAt
1	Training bench	https://gymavailability.blob.core.windows.net/mac...	Bench Press Machine	2CF7F1203230D0A3	2023-05-04 17:40:26.137	2023-08-13 13:59:44.897
2	Test device	https://gymavailability.blob.core.windows.net/mac...	Squat Rack	2CF7F1203230C670	2023-05-04 17:40:26.137	2023-08-13 13:59:56.080
3	Test Scenario	NULL	Test Description	ABCDEFGHIJKLMNP	2023-08-19 13:38:17.080	2023-08-19 13:38:17.080

Rysunek 5.15. Dodana maszyna w bazie danych

Gym Rooms Machines

#	Name	Device EUI	Description	Gym Room	Image	Placement
1	Training bench	2CF7F1203230D0A3	Bench Press Machine	Main Room	<input checked="" type="checkbox"/>	Near window
2	Test device	2CF7F1203230C670	Squat Rack	Main Room	<input checked="" type="checkbox"/>	Near door

Name Device EUI

Edit Scenario ABCDEFGHIJKLMNOP

Description Gym Room

Edit Description Main Room

Placement

Edit Placement

New Edit Delete Add Image Save Cancel

First Prev 1 Next Last 1 - 3 of 3 items

Rysunek 5.16. Edytowanie maszyny w aplikacji webowej

```

SQLQuery3.sql - gy...ability-admin (74)  X gym-availability-da...AvailabilityReports      SQLQuery2.sql - gy...ability-admin (71))* 
/*===== Script for SelectTopNRows command from SSMS =====*/
[SELECT TOP (1000) [Id]
 , [Name]
 , [ImagefileLink]
 , [Description]
 , [DeviceEUI]
 , [createdAt]
 , [updatedAt]
FROM [dbo].[Machines]

100 %  <  Results  Messages

```

	Id	Name	ImagefileLink	Description	DeviceEUI	createdAt	updatedAt
1	1	Training bench	https://gymavailability.blob.core.windows.net/mac...	Bench Press Machine	2CF7F1203230D0A3	2023-05-04 17:40:26.137	2023-08-19 13:50:06.770
2	2	Test device	https://gymavailability.blob.core.windows.net/mac...	Squat Rack	2CF7F1203230C670	2023-05-04 17:40:26.137	2023-08-13 13:59:56.080
3	12	Edit Scenario	NULL	Edit Description	ABCDEFGHIJKLMNPQ	2023-08-19 13:38:17.080	2023-08-19 13:51:04.917

Rysunek 5.17. Edytowana maszyna w bazie danych

#	Name	Device EUI	Description	Gym Room	Image	Placement
1	Training bench	2CF7F1203230D0A3	Bench Press Machine	Main Room	✓	Near window
2	Test device	2CF7F1203230C670	Squat Rack	Main Room	✓	Near door

New Edit Delete Add Image First Prev 1 Next Last 1 - 2 of 2 items

Rysunek 5.18. Usunięta maszyna w aplikacji webowej

```

SQLQuery3.sql - gy...ability-admin (74)  X gym-availability-da...AvailabilityReports      SQLQuery2.sql - gy...ability-admin (71))* 
/*===== Script for SelectTopNRows command from SSMS =====*/
[SELECT TOP (1000) [Id]
 , [Name]
 , [ImagefileLink]
 , [Description]
 , [DeviceEUI]
 , [createdAt]
 , [updatedAt]
FROM [dbo].[Machines]

100 %  <  Results  Messages

```

	Id	Name	ImagefileLink	Description	DeviceEUI	createdAt	updatedAt
1	1	Training bench	https://gymavailability.blob.core.windows.net/mac...	Bench Press Machine	2CF7F1203230D0A3	2023-05-04 17:40:26.137	2023-08-19 13:50:06.770
2	2	Test device	https://gymavailability.blob.core.windows.net/mac...	Squat Rack	2CF7F1203230C670	2023-05-04 17:40:26.137	2023-08-13 13:59:56.080

Rysunek 5.19. Usunięta maszyna w bazie danych

Test zakończył się powodzeniem zgodnie z oczekiwany wynikiem. Udało się po- myślnie dodać urządzenie, następnie przeprowadzić jego edycję, a na końcu poprawnie usunąć.

6. Podsumowanie

W ramach pracy inżynierskiej zrealizowany został projekt systemu pozwalającego na sprawdzanie zajętości urządzeń na siłowni wraz z jego implementacją. Pozwoliło to na osiągnięcie celu pracy. Zgodnie z nim powstał prototyp urządzenia, które po zamontowaniu przed stanowiskiem na siłowni jest w stanie sprawdzać, czy aktualnie ktoś z niego korzysta, a następnie przesyła tą informację do sieci LoRaWAN. Kolejnym powstalym elementem w ramach wypełnienia celu pracy jest infrastruktura sieciowa w postaci konfiguracji sieci TTN, utworzenia bazy danych i innych niezbędnych usług wymaganych w projekcie. Ostatnim stworzonym komponentem jest aplikacja webowa.

Zaimplementowany system pozwala na badanie zajętości stanowisk przez stworzone urządzenie, które wykrywa obecność człowieka z pomocą modeli uczenia maszynowego analizujące obrazy z kamery zainstalowanej w urządzeniu. Następnie przesyła uzyskany wynik oznaczający czy analizowany obszar jest zajęty, bądź też nie, do sieci LoRaWAN. Utworzony serwer aplikacyjny w sieci TTN przekazuje wiadomości o stanie zajętości dalej do mikrousługi danych, która to odpowiada za umieszczanie raportów w bazie danych. Z bazy danych korzysta natomiast powstała aplikacja webowa. Pozwala ona na wyświetlanie wspomnianych informacji w czasie rzeczywistym. Ponadto udostępniane są funkcjonalności takie jak zarządzanie siłownią, poprzez dodawanie i edytowanie maszyn oraz sal, możliwość rejestracji i logowania się użytkowników, sprawdzanie historii zajętości maszyn, czy edycja awataru użytkownika.

Przeprowadzone zostały także testy wraz z eksperymentem polegającym na zamontowaniu prototypu urządzenia w hali produkcyjnej. Dzięki nim, możliwe było sprawdzenie czy system działa zgodnie z założeniami oraz nie posiada znaczących błędów w implementacji. Wyniki pokazały, że system spełnia wymagania postawione na początkowym etapie pracy.

Powstały system posiada wiele obszarów, które mogą zostać poddane rozwojowi. Jednym z nich jest udoskonalenie algorytmu sprawdzającego zajętość poprzez dokładniejsze dostosowanie parametrów takich jak procent pewności, czy wykrytym obiektem jest człowiek, liczba analizowanych klatek, minimalny procent klatek w których wykryto człowieka pozwalający na stwierdzenie że urządzenie jest zajęte. Kolejnym usprawnieniem jest ułatwienie konfiguracji urządzenia. Aplikacja webowa posiada też kilka funkcjonalności, które potencjalnie usprawniłyby działanie systemu. Są to na przykład dodanie większej ilości generowanych wykresów czy wprowadzenie automatycznych sugestii godzin, w których najlepiej uczęszczać na siłownię.

Bibliografia

- [1] U. S. w Rzeszowie, *Obiekty służące poprawie kondycji fizycznej w 2020 r.* Dostęp zdalny (26.06.2023): https://stat.gov.pl/download/gfx/portalinformacyjny/pl/defaultaktualnosci/5495/15/1/1/obiekty_sluzace_poprawie_kondycji_fizycznej_w_2020_r.pdf, 2021.
- [2] P. E. George, *How Occupant Classification Systems Work*, Dostęp zdalny (03.07.2023): <https://auto.howstuffworks.com/car-driving-safety/safety-regulatory-devices/ocs.htm>, 2008.
- [3] J. Weller, *9 Common Gym Complaints and How to Deal With Them*. Dostęp zdalny (26.06.2023): <https://www.glofox.com/blog/gym-complaints/>, 2021.
- [4] Y. L. H. H. Nguyen N. Gulati i R. K. Balan, “Real-time Detection of Seat Occupancy and Hogging”, w *IoT-App '15: Proceedings of the 2015 International Workshop on Internet of Things towards Applications: Seoul, Korea, November 1*, ACM, 2015.
- [5] R. C. Koons, “Sobel on Gödel’s Ontological Proof”, Dostęp zdalny (25.04.2019): <http://www.robkoons.net/media/69b0dd04a9d2fc6dffff80b4ffffd524.pdf>, 2005.
- [6] R. Girshick, J. Donahue, T. Darrell i J. Malik, *Rich feature hierarchies for accurate object detection and semantic segmentation*, 2014. arXiv: 1311.2524 [cs.CV].
- [7] R. Girshick, *Fast R-CNN*, 2015. arXiv: 1504.08083 [cs.CV].
- [8] S. Ren, K. He, R. Girshick i J. Sun, *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*, 2016. arXiv: 1506.01497 [cs.CV].
- [9] C.-Y. Wang, A. Bochkovskiy i H.-Y. M. Liao, “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors”, *arXiv preprint arXiv:2207.02696*, 2022.
- [10] R. P. Ltd, *Raspberry Pi 4*, Dostęp zdalny (09.07.2023): <https://www.raspberrypi.com/documentation/>, 2023.
- [11] OdSeven, *OdSeven HD IR-CUT OV5647*, Dostęp zdalny (09.07.2023): <https://odseven.com/products/odseven-raspberry-pi-3-model-b-camera-kit-5mp-focal-adjustable-night-version-camera>, 2023.
- [12] SpeedStudio, *Grove - Wio-E5*, Dostęp zdalny (09.07.2023): https://wiki.seeedstudio.com/Grove_LoRa_E5_New_Version/, 2022.
- [13] T. T. Industries, *The Things Network*, Dostęp zdalny (03.08.2023): <https://www.thethingsnetwork.org/>, 2023.
- [14] T. T. Industries, *The Things Network*, Dostęp zdalny (05.08.2023): <https://www.thethingsindustries.com/docs/integrations/webhooks/>, 2023.
- [15] Microsoft, *What is Azure*, Dostęp zdalny (09.07.2023): <https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure/>, 2023.
- [16] Microsoft, *ASP.NET core documentation*, Dostęp zdalny (09.07.2023): https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-7.0&WT.mc_id=dotnet-35129-website/, 2023.

6. Bibliografia

- [17] Microsoft, *Entity Framework Core*, Dostęp zdalny (09.07.2023): <https://learn.microsoft.com/en-us/ef/core/>, 2023.
- [18] Microsoft, *Azure App Service*, Dostęp zdalny (09.07.2023): <https://azure.microsoft.com/en-us/products/app-service/>, 2023.
- [19] Microsoft, *What is Azure SQL Database?*, Dostęp zdalny (09.07.2023): <https://learn.microsoft.com/en-us/azure/azure-sql/database/sql-database-paas-overview?view=azuresql-db>, 2023.
- [20] Microsoft, *Azure Key Vault basic concepts*, Dostęp zdalny (09.07.2023): <https://learn.microsoft.com/en-us/azure/key-vault/general/basic-concepts>, 2023.
- [21] Microsoft, *Introduction to Azure Blob Storage*, Dostęp zdalny (10.07.2023): <https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction>, 2023.
- [22] Microsoft, *Azure Functions documentation*, Dostęp zdalny (10.07.2023): <https://learn.microsoft.com/en-us/azure/azure-functions/>, 2023.
- [23] Microsoft, *ASP.NET Core Blazor*, Dostęp zdalny (10.07.2023): <https://learn.microsoft.com/en-us/aspnet/core/blazor/hosting-models?view=aspnetcore-7.0>, 2023.
- [24] Megabit, *Blazorise documentation*, Dostęp zdalny (10.07.2023): <https://blazorise.com/docs>, 2023.
- [25] Syncfusion, *Syncfusion documentation*, Dostęp zdalny (10.07.2023): <https://www.syncfusion.com/blazor-components>, 2023.
- [26] R. P. Ltd, *Raspberry Pi OS documentation*, Dostęp zdalny (05.08.2023): <https://www.raspberrypi.com/documentation/computers/os.html>, 2023.
- [27] P. S. Foundation, *Python Documentation*, Dostęp zdalny (05.08.2023): <https://docs.python.org/3/>, 2023.
- [28] WiringPi-Python repository, Dostęp zdalny (05.08.2023): <https://github.com/WiringPi/WiringPi-Python>, 2023.
- [29] PyCayenneLPP documentation, Dostęp zdalny (05.08.2023): <https://pypi.org/project/pycayennelpp/>, 2023.
- [30] A. Allan, *A TensorFlow Lite example for Picamera2 on Raspberry Pi*, Dostęp zdalny (06.08.2023): https://github.com/raspberrypi/picamera2/blob/main/examples/tensorflow/real_time_with_labels.py, 2023.
- [31] *The Picamera2 Library*, 2023.
- [32] OpenCV Python Library, Dostęp zdalny (06.08.2023): <https://pypi.org/project/opencv-python/>, 2023.
- [33] *TensorFlow Lite documentation*, Dostęp zdalny (06.08.2023): https://www.tensorflow.org/lite/api_docs, 2023.
- [34] *ZeroTier documentation*, Dostęp zdalny (06.08.2023): <https://zerotier.atlassian.net/wiki/spaces/SD/overview>, 2023.

- [35] R. Anderson, *Introduction to Identity on ASP.NET Core*, Dostęp zdalny (11.08.2023):
<https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-7.0&tabs=visual-studio>, 2022.

Wykaz symboli i skrótów

- AppEUI** – ang. *Application Extended Unique Identifier*
COCO – ang. *Common objects in Context*
CSS – ang. *Chirp Spread Spectrum*
DevEUI – ang. *Device Extended Unique Identifier*
HTTP – ang. *Hypertext Transfer Protocol*
IR – ang. *Infrared*
LoRaWAN – ang. *Long Range Wide Area Network*
LPWAN – ang. *Low Power Wide Area Network*
ORM – ang. *Object Relational Mapper*
R-CNN – ang. *Region-based Convolutional Neural Network*
RoI – ang. *Regions of Interest*
TTN – *The Things Network*
YOLO – ang. *You Only Look Once*

Spis rysunków

3.1	Diagram przypadków użycia aplikacji webowej	19
3.2	Wysokopoziomowa architektura systemu	21
3.3	Schemat budowy urządzenia	21
3.4	Schemat części chmurowej	23
4.1	Testowa konfiguracja urządzenia	26
4.2	Wnętrze skrzynki montażowej	27
4.3	Gotowe urządzenie badające zajętość	27
4.4	Algorytm działania modułu detekcji	29
4.5	Utworzona aplikacja w chmurze TTN	30
4.6	Skonfigurowane urządzenia	30
4.7	Skonfigurowany Webhook	32
4.8	Widok utworzonej bazy danych w usłudze Azure SQL Server	33
4.9	Mikrousługa danych w usłudze Azure App Service	34
4.10	Widok aplikacji wyzwalającej przetwarzanie danych w usłudze Azure Functions	35
4.11	Widok usługi Azure Key Vault	36
4.12	Widok usługi Azure Blob Storage	36
4.13	Widok ekranu głównego aplikacji	37
4.14	Widok ekranu logowania	38
4.15	Widok ekranu rejestracji	38
4.16	Wiadomość prezentowana po rejestracji	38

4.17 Treść wiadomości email	38
4.18 Widok strony z komunikatem o poprawnej rejestracji	39
4.19 Widok opcji dotyczących konta	39
4.20 Widok zarządzania kontem	40
4.21 Dialog edycji awatara	40
4.22 Przesłanie pliku zawierającego avatara	41
4.23 Podstawowa sekcja widoku przeglądania listy sal	41
4.24 Widok przedstawiający wybraną salę	42
4.25 Widok przedstawiający szczegóły maszyny	42
4.26 Widok zarządzania salami	43
4.27 Widok dodawania sali	43
4.28 Widok edytowania sali	43
4.29 Dialog edycji zdjęcia sali	44
4.30 Widok zarządzania maszynami	44
4.31 Widok wyboru urządzenia i daty	45
4.32 Widok wygenerowanego wykresu	45
5.1 Stanowisko na hali produkcyjnej	46
5.2 Zamontowane na hali urządzenie	47
5.3 Obraz z kamery czujnika	47
5.4 Wykres na podstawie zebranych wyników	48
5.5 Początkowy stan urządzenia	49
5.6 Sprawdzenie dziennika logów sieci LoRaWAN	49
5.7 Sprawdzenie wpisu w bazie danych	49
5.8 Końcowy stan urządzenia w aplikacji webowej	49
5.9 Początkowy avatar w aplikacji webowej	50
5.10 Początkowy avatar w magazynie plików	50
5.11 Avatar w aplikacji webowej po zmianach	50
5.12 Avatar po zmianie w magazynie plików	50
5.13 Dialog zmiany awataru	51
5.14 Dodanie nowej maszyny w aplikacji webowej	52
5.15 Dodana maszyna w bazie danych	52
5.16 Edytowanie maszyny w aplikacji webowej	52
5.17 Edytowana maszyna w bazie danych	53
5.18 Usunięta maszyna w aplikacji webowej	53
5.19 Usunięta maszyna w bazie danych	53

Spis tabel

4.1 Tabela Gyms	33
4.2 Tabela GymRooms	33
4.3 Tabela Machines	34
4.4 Tabela MachinePlacements	34
4.5 Tabela AvaiabilityReports	34
4.6 Tabela AvaiabilityReportFactSt	34

Spis załączników

1. Kod modułu obsługującego nadajnik Lora	61
2. Kod modułu sprawdzającego zajętość	62
3. Kod mikrousługi danych	69
4. Kod aplikacji wyzwalającej przetwarzanie danych	69
5. Kod aplikacji webowej	69

Spis listingów

1 Translator wiadomości LoRa	31
2 Konfiguracja komunikacji z menadżerem sekretów	36
3 Klasa LoraConnector.py	61

Załącznik 1. Kod modułu obsługującego nadajnik Lora

```
1 import wiringpi
2 import time
3 from cayennelpp import LppFrame
4
5 class LoraConnector:
6
7     def __init__(self, stack):
8         self.stack = stack
9         wiringpi.wiringPiSetup()
10        self.serial = wiringpi.serialOpen('/dev/serial0', 9600)
11
12    def connect(self, retries):
13        wiringpi.serialPuts(self.serial, 'AT+ID=AppEui,0000000000000000')
14        time.sleep(1.1)
15        wiringpi.serialPuts(self.serial, 'AT+ID=DevEui')
16        time.sleep(1.1)
17        wiringpi.serialPuts(self.serial, 'AT+DR=EU868')
18        time.sleep(1.1)
19        wiringpi.serialPuts(self.serial, 'AT+CH=NUM,0-2')
20        time.sleep(1.1)
21        wiringpi.serialPuts(self.serial, 'AT+MODE=LWOTAA')
22        time.sleep(1.1)
23        wiringpi.serialPuts(self.serial, 'AT+KEY=APPKEY,\\"ED032FC3A30F76D68F97')
24        time.sleep(1.1)
25        wiringpi.serialPuts(self.serial, 'AT+ID')
26        time.sleep(1.1)
27        for i in range(retries):
28            wiringpi.serialPuts(self.serial, 'AT+JOIN')
29            time.sleep(10)
30
31    def join(self):
32        wiringpi.serialPuts(self.serial, 'AT+JOIN')
33
34    def send_status(self, status):
35        frame = LppFrame()
36        if(status):
37            frame.add_digital_output(0, 1)
38        else:
```

```

39             frame.add_digital_output(0, 0)
40         buffer = bytes(frame)
41         msg= 'AT+MSG=' +buffer.hex()
42         print(msg)
43         time.sleep(1.1)
44         wiringpi.serialPuts(self.serial, msg)
45         time.sleep(2.1)
46
47     def join_loop(self, retries):
48         for retrie in range(0,retries):
49             wiringpi.serialPuts(self.serial, 'AT+JOIN')
50             time.sleep(10)
51
52 if __name__ == '__main__':
53     lora = LoraConnector('TTN')
54     #lora.connect(1)
55     #lora.join()
56     #lora.join_loop(5)
57     lora.send_status(True)

```

Listing 3. Klasa LoraConnector.py

Załącznik 2. Kod modułu sprawdzającego zajętość

```

1  #!/usr/bin/python3
2  # Copyright (c) 2022 Raspberry Pi Ltd
3  # Author: Alasdair Allan <alasdair@raspberrypi.com>
4  # SPDX-License-Identifier: BSD-3-Clause
5
6  import argparse
7  import cv2
8  import numpy as np
9  import tflite_runtime.interpreter as tflite
10 import time
11 from statistics import mean
12 from LoraConnector import LoraConnector
13 from datetime import datetime
14 from picamera2 import MappedArray, Picamera2, Preview
15
16 old_print = print

```

```

17
18 def timestamped_print(*args , **kwargs):
19     old_print(datetime.now() , *args , **kwargs)
20
21 print = timestamped_print
22
23
24 normalSize = (640 , 480)
25 lowresSize = (320 , 240)
26 org =(50,50)
27 font = cv2.FONT_HERSHEY_SIMPLEX
28 rectangles = []
29 availabilityArray = []
30 frames_to_check = 180
31 availabilityText = "false"
32 curr_availability = False
33 prev_availability = False
34 loraUsage = False
35 availability_test_counter = 0
36 test_counter_threshold = 500
37 classification_percentage = 0.5
38 detection_threshold = 0.8
39
40 def ReadLabelFile(file_path):
41     with open(file_path , 'r') as f:
42         lines = f.readlines()
43     ret = {}
44     for line in lines:
45         pair = line.strip().split(maxsplit=1)
46         ret[int(pair[0])] = pair[1].strip()
47     return ret
48
49 def calculateAvailability(availabilityArray):
50     if mean(availabilityArray) > detection_threshold:
51         return True
52     else:
53         return False
54
55 def DrawRectangles(request):

```

```

56
57     with MappedArray(request, "main") as m:
58         global avaiabilityText, avaiabilityArray, curr_avaiability,
59             prev_avaiability
60         global avaiability_test_counter, test_counter_threshold, frames_to_check
61         global detection_threshold, classification_percentage
62         if rectangles:
63             avaiabilityArray.append(1)
64             for rect in rectangles:
65                 rect_start = (int(rect[0] * 2) - 5, int(rect[1] * 2) - 5)
66                 rect_end = (int(rect[2] * 2) + 5, int(rect[3] * 2) + 5)
67                 cv2.rectangle(m.array, rect_start, rect_end, (0, 255, 0, 0))
68                 if len(rect) == 5:
69                     text = rect[4]
70                     cv2.putText(m.array, text, (int(rect[0] * 2) + 10, int(rect[1] *
71                         font, 1, (255, 255, 255), 2, cv2.LINE_AA)
72         else:
73             avaiabilityArray.append(0)
74         if(len(avaiabilityArray) > frames_to_check):
75
76             curr_avaiability = calculateAvaiability(avaiabilityArray)
77             print("Previous avaiability", prev_avaiability)
78             print("Current avaiability", curr_avaiability)
79             print("Current frame", avaiability_test_counter)
80             avaiabilityText = str(curr_avaiability)
81             avaiabilityArray = []
82             if(((curr_avaiability != prev_avaiability)
83         or avaiability_test_counter == test_counter_threshold)and loraUsage):
84                 lora.send_status(curr_avaiability)
85                 avaiability_test_counter = 0
86                 print("sending lora packet")
87                 avaiability_test_counter +=1
88                 prev_avaiability = curr_avaiability
89     def InferenceTensorFlow(image, model, output, label=None):
90         global rectangles, avaiabilityArray, classification_percentage
91
92         if label:
93             labels = ReadLabelFile(label)
94         else:

```

```

95     labels = None
96
97     interpreter = tflite.Interpreter(model_path=model, num_threads=4)
98     interpreter.allocate_tensors()
99     image = cv2.rotate(image, cv2.ROTATE_90_COUNTERCLOCKWISE)
100    input_details = interpreter.get_input_details()
101    output_details = interpreter.get_output_details()
102    height = input_details[0]['shape'][1]
103    width = input_details[0]['shape'][2]
104    floating_model = False
105    if input_details[0]['dtype'] == np.float32:
106        floating_model = True
107
108    rgb = cv2.cvtColor(image, cv2.COLOR_GRAY2RGB)
109    initial_h, initial_w, channels = rgb.shape
110
111    picture = cv2.resize(rgb, (width, height))
112
113    input_data = np.expand_dims(picture, axis=0)
114    if floating_model:
115        input_data = (np.float32(input_data) - 127.5) / 127.5
116
117    interpreter.set_tensor(input_details[0]['index'], input_data)
118
119    interpreter.invoke()
120
121    detected_boxes = interpreter.get_tensor(output_details[0]['index'])
122    detected_classes = interpreter.get_tensor(output_details[1]['index'])
123    detected_scores = interpreter.get_tensor(output_details[2]['index'])
124    num_boxes = interpreter.get_tensor(output_details[3]['index'])
125    rectangles = []
126    for i in range(int(num_boxes)):
127        top, left, bottom, right = detected_boxes[0][i]
128        classId = int(detected_classes[0][i])
129        score = detected_scores[0][i]
130        if score > classification_percentage and classId == 0:
131            xmin = left * initial_w
132            ymin = bottom * initial_h
133            xmax = right * initial_w

```

```

134         ymax = top * initial_h
135         box = [xmin, ymin, xmax, ymax]
136         rectangles.append(box)
137         if labels:
138             print(labels[classId], 'score = ', score)
139             rectangles[-1].append(labels[classId])
140         else:
141             print('score = ', score)
142
143     def main():
144         global loraUsage, lora, frames_to_check
145         global availability_test_counter, test_counter_threshold
146         global detection_threshold, classification_percentage
147         parser = argparse.ArgumentParser()
148         parser.add_argument('--model', help='Path of the detection model.', required=True)
149         parser.add_argument('--label', help='Path of the labels file.')
150         parser.add_argument('--output', help='File path of the output image.')
151         parser.add_argument('--preview', help='Show preview')
152         parser.add_argument('--lora', help='Use Lora')
153         parser.add_argument('--frames', help='How many frames should be checked in availability calculation')
154         parser.add_argument('--dp', help='What percentage should detected object be')
155         parser.add_argument('--ap', help='What percentage or frames person should be in')
156         parser.add_argument('--it', help='How many checkes should result in resending the status')
157         args = parser.parse_args()
158
159         if (args.output):
160             output_file = args.output
161         else:
162             output_file = 'out.jpg'
163             if (args.label):
164                 label_file = args.label
165             else:
166                 label_file = None
167             if (args.preview):
168

```

```

173         preview = eval(args.preview)
174     else:
175         preview = False
176     if (args.lora):
177         loraUsage = eval(args.lora)
178     else:
179         loraUsage = False
180     if (args.frames):
181         frames_to_check = eval(args.frames)
182     else:
183         frames_to_check = 180
184     if (args.ap):
185         detection_threshold = eval(args.ap)
186     else:
187         detection_threshold = 0.8
188     if (args.dp):
189         classification_percentage = eval(args.dp)
190     else:
191         classification_percentage = 0.5
192     if (args.it):
193         test_counter_threshold = eval(args.it)
194     else:
195         test_counter_threshold = 500
196
197     if(loraUsage):
198         lora = LoraConnector("TTN")
199         picam2 = Picamera2()
200         if preview:
201             picam2.start_preview(Preview.QTGL)
202             config = picam2.create_preview_configuration(main={"size": normalSize},
203             lores={"size": lowresSize, "format": "YUV420"})
204             # config["transform"] = libcamera.Transform(vflip=1)
205             picam2.configure(config)
206
207             stride = picam2.stream_configuration("lores")["stride"]
208             picam2.post_callback = DrawRectangles
209
210             picam2.start()
211

```

```
212     while True:  
213         buffer = picam2.capture_buffer("lores")  
214         grey = buffer [: stride * lowresSize[1]].reshape((lowresSize[1], stride))  
215         _ = InferenceTensorFlow(grey, args.model, output_file, label_file)  
216  
217     if __name__ == '__main__':  
218         main()
```

Załącznik 3. Kod mikrousługi danych

Repozytorium w serwisie Github: github.com/marcinRuta/AvailabilityReportApi

Załącznik 4. Kod aplikacji wyzwalającej przetwarzanie danych

Repozytorium w serwisie Github: github.com/marcinRuta/AvailabilityReportFunctionsApp

Załącznik 5. Kod aplikacji webowej

Repozytorium w serwisie Github: github.com/marcinRuta/GymAvailabilityApp