

**Politechnika Warszawska**

W Y D Z I A Ł E L E K T R Y C Z N Y



Instytut Elektrotechniki Teoretycznej i Systemów Informacyjno-Pomiarowych

# Praca dyplomowa inżynierska

na kierunku Informatyka Stosowana  
w specjalności Inżynieria danych i multimedia

Konstrukcja i oprogramowanie robota wyposażonego w kamerę do monitorowania budynku

**Jordan Parviaainen**

numer albumu 319084

promotor  
dr hab. inż. Marcin Kołodziej, prof.

WARSZAWA 2025



# **Konstrukcja i oprogramowanie robota wyposażonego w kamerę do monitorowania budynku**

## **Streszczenie**

Praca dotyczy konstrukcji i oprogramowania mobilnego robota wyposażonego w kamerę, wykorzystującego Raspberry Pi jako platformę sprzętową. Celem pracy było stworzenie funkcjonalnego zdalnie sterowanego robota, który może być wykorzystany do monitorowania budynku.

We wstępie przedstawiono cel i zakres pracy, jak i motywację wybór tematyki. Przybliżono problematykę nowoczesnych metod monitoringu i zaprezentowano istniejące rozwiązania naukowe oraz komercyjne.

W kolejnej części opisano sprzęt, który został użyty do zbudowania robota. Opisano wykorzystane elementy konstrukcyjne – gotowy zestaw podwozia z silnikami elektrycznymi. Przedstawiono nakładkę Motor Driver Hat, która pozwala na sterowanie silnikami i zasilanie Raspberry Pi. Opisano również zasilanie akumulatorowe, które zastosowano w robocie. Zamieszczono schemat całego układu. Przedstawiono również koszty związane z budową robota.

Trzecia część pracy dotyczy oprogramowania. Opisano definicję wymagań za pomocą diagramu przypadków użycia. Przedstawiono architekturę systemu, w tym komunikację między poszczególnymi modułami. Wykorzystano diagram komponentów do przedstawienia zależności pomiędzy mikroserwisami po stronie backendu i ich połączenia z frontendem. Opisano architekturę wdrożenia systemu na Raspberry Pi i wykorzystanie wirtualnej sieci prywatnej ZeroTier. Przedstawiono mikroserwis użytkowników i ustawień poprzez wymienienie wystawionych przez niego endpointów z funkcjonalnościami i ukazanie architektury na diagramie klas. Przedstawiony został również mikroserwis sterowania silnikami robota, który wystawia serwer WebSocket pozwalający na komunikację w czasie rzeczywistym z robotem. Kolejnym mikroserwisem jest serwer kamery, który obsługuje strumienianie obrazu z kamery na żywo. Ostatni opisany mikroserwis backendowy to serwis archiwizacji zdjęć, który wysyła zdjęcia z kamery na zdalny serwer SFTP. Dalej przybliżono stronę internetową, która pozwala na zdalne sterowanie robotem.

Kolejna część pracy dotyczy testów. Opisano testy oprogramowania, które były przeprowadzane w trakcie wytwarzania. Wymieniono testy jednostkowe, integracyjne, end-to-end i manualne. Spisane zostały scenariusze testowe robota, które były wykonywane po zakończeniu budowy sprzętu i oprogramowania. Przedstawiono wyniki testów, które wykazały, że robot spełnia swoje zadanie.

Ostatnia część pracy to podsumowanie. Stwierdzono, że cel pracy został zrealizowany. Wskazano, że część sprzętowa ma charakter prototypowy, ale system informatyczny jest gotowy do dalszego rozwoju. Przedstawiono możliwości rozwoju projektu w przyszłości.

**Słowa kluczowe:** robot, kamera, monitorowanie, Raspberry Pi, sterowanie, aplikacja webowa, Python



# **Constructing and programming a camera equipped robot for indoor surveillance**

## **Abstract**

The thesis is about the construction and software development of a mobile robot equipped with a camera, using Raspberry Pi as the hardware platform. The aim of the thesis was to create a functional remotely controlled robot that can be used for building surveillance.

The introduction presents the purpose and scope of the work, as well as the motivation behind choosing this topic. It discusses the challenges of modern monitoring methods and presents existing scientific and commercial solutions.

The next section describes the hardware used to build the robot. It details the components used in the construction — a ready-made chassis kit with electric motors. The Motor Driver Hat extension, which enables motor control and powers the Raspberry Pi, is described. The battery power supply used in the robot is also covered. A schematic of the entire system is included, along with the costs associated with building the robot.

The third section focuses on the software. The requirements are defined using a use case diagram. The system architecture is presented, including communication between its individual modules. A component diagram is used to show dependencies between backend microservices and their connection to the frontend. The system deployment architecture on the Raspberry Pi and the use of the ZeroTier virtual private network are described. The user and settings microservice is detailed, listing its exposed endpoints with functionalities and illustrating its architecture with a class diagram. The motor control microservice is presented, which exposes a WebSocket server enabling real-time communication with the robot. Another microservice is the camera server, which handles live video streaming from the camera. The final backend microservice described is the photo archiving service, which sends camera photos to a remote SFTP server. The web interface allowing remote control of the robot is also introduced.

The next section covers testing. It describes software tests conducted during development, including unit, integration, end-to-end, and manual tests. Test scenarios for the robot, performed after completing the hardware and software, are listed. The test results are presented, showing that the robot fulfills its intended purpose.

The final section is the conclusion. It states that the thesis objective was achieved. It notes that the hardware is of a prototype nature, but the IT system is ready for further development. Future development opportunities for the project are outlined.

**Keywords:** robot, camera, surveillance, Raspberry Pi, control, web application, Python



# Spis treści

|   |           |
|---|-----------|
| <b>1 Wstęp</b>  | <b>9</b>  |
| 1.1 Cel i zakres pracy . . . . .                          | 9         |
| 1.2 Problematyka . . . . .                                | 9         |
| 1.3 Powiązane publikacje naukowe . . . . .                | 10        |
| 1.4 Istniejące rozwiązania komercyjne . . . . .           | 10        |
| 1.5 Motywacja . . . . .                                   | 11        |
| <b>2 Sprzęt</b>   | <b>13</b> |
| 2.1 Cel . . . . .   | 13        |
| 2.2 Korpus . . . . .                                      | 13        |
| 2.3 Elektronika . . . . .                                 | 14        |
| 2.4 Zasilanie . . . . .                                   | 15        |
| 2.5 Schemat układu . . . . .                              | 16        |
| 2.6 Złożona konstrukcja . . . . .                         | 17        |
| 2.7 Koszty . . . . .                                      | 17        |
| <b>3 Oprogramowanie</b>                                   | <b>20</b> |
| 3.1 Definicja wymagań . . . . .                           | 20        |
| 3.2 Architektura systemu . . . . .                        | 21        |
| 3.3 Wdrożenie . . . . .                                   | 22        |
| 3.4 Serwis użytkowników i ustawień . . . . .              | 23        |
| 3.5 Serwis sterujący silnikami robota . . . . .           | 24        |
| 3.6 Serwis kamery . . . . .                               | 26        |
| 3.7 Serwis archiwizacji zdjęć . . . . .                   | 26        |
| 3.8 Strona internetowa . . . . .                          | 26        |
| 3.9 Moduł joysticka do sterowania ruchem robota . . . . . | 29        |
| <b>4 Testy</b>  | <b>31</b> |
| 4.1 Testy oprogramowania . . . . .                        | 31        |
| 4.2 Testy robota . . . . .                                | 31        |
| <b>5 Podsumowanie</b>                                     | <b>33</b> |

|                                  |           |
|----------------------------------|-----------|
| 5.1 Realizacja celu . . . . .    | 33        |
| 5.2 Możliwości rozwoju . . . . . | 33        |
| <b>Bibliografia</b>              | <b>35</b> |
| <b>Spis rysunków</b>             | <b>37</b> |

# Rozdział 1

## Wstęp

### 1.1 Cel i zakres pracy

Celem pracy było zbudowanie mobilnego robota, który może być wykorzystany do monitorowania budynku. Podstawowym jego wyposażeniem jest kamera. Robot jest zdolny do poruszania się na kołach.

Obsługa odbywa się przez aplikację webową (przeglądarkową). Oprogramowanie pozwala na zdalne sterowanie pojazdem bez kontaktu wzrokowego dzięki strumieniowaniu obrazu z kamery na żywo. Dane wizualne z kamery są gromadzone i regularnie wysyłane na zdalny nośnik. Aplikacja jest zabezpieczona przed nieupoważnionymi osobami i pozwala na zarządzanie dostępem użytkowników.

### 1.2 Problematyka

Monitoring miejsc, obiektów itp. to powszechna praktyka w miejscach zarówno publicznych, jak i prywatnych. Jest to jeden ze środków ochrony ludzi i mienia. Jak opisuje artykuł[13] opublikowany w *Wall Street Journal*, masowy monitoring prędko się upowszechnił po traumatycznej tragedii zamachu na wieże World Trade Center 11 września 2001 roku. Obecnie przemysł nadzorowania za pomocą kamer jest wyceniany na ok. 74 miliardy USD[3] i przewiduje się roczny wzrost na poziomie 12%.

Ogląd na stan obecny i kierunek rozwoju systemów do wideo monitoringu przedstawiają artykuły naukowe: [2] oraz [12]. Najbardziej powszechnym systemem monitoringu jest sieć zainstalowanych na stałe kamer. Kiedyś były to zamknięte obwody, do których obserwowania trzeba było stale zatrudniać ochroniarzy. Dziś powszechnie są tzw. kamery IP ze stałym podłączeniem do internetu, coraz częściej bezprzewodowym. Otwiera to drogi do nowych możliwości w zakresie archiwizacji, przetwarzania i podejmowania decyzji na podstawie danych wideo. Coraz częściej sięga się po zaawansowane metody przetwarzania obrazu wykorzystujące uczenie maszynowe, sztuczną inteligencję. Stawia się też na zastosowanie szerokiej gamy sensorów oprócz samych kamer.

Jednym z innowacyjnych rozwiązań dotyczących monitoringu są mobilne roboty. Nie jest to (jeszcze) podejście głównego nurtu, ale pojawiają się głosy mówiące o potrzebie takich rozwiązań, jak i same konstrukcji. Zaletami takich rozwiązań są m.in.

- zwiększenie obszaru pokrytego monitoringiem – statycznie zamontowane kamery mają martwe pola,
- zmniejszenie zasobów potrzebnych na nadzór dużych powierzchni – wiele kamer statycznych może zastąpić jeden mobilny robot.

Na konferencji *2010 IEEE Workshop on Advanced Robotics and its Social Impacts* tematem jednego z wystąpień było *Intelligent surveillance and security robot systems*[4]. Zaprezentowanych zostało kilka typów nowatorskich, inteligentnych rozwiązań do monitoringu, oto opis jednego z nich:

A mobile patrol robot named ‘STAR’, with a unique single pivot design, is driven by four independently steered wheels, which enables changing the direction on the spot. The mobile robot is designed for surveillance over a vast range of fields with the guidance by a GPS sensor and a scanning laser range finder.

Ta praca wpisuje się w dziedzinę podobnych rozwiązań.

### 1.3 Powiązane publikacje naukowe

Przeszukując literaturę naukową w tematyce mobilnych robotów nadzorujących, można zauważyc, że nie jest to popularna dziedzina i wiele prac przedstawia jedynie projekty i/lub niezbyt mocno rozwinięte prototypy. Tutaj pokrótko przedstawię kilka z ciekawzych pomysłów.

W [11] opisany jest prototyp jeżdżącego robota wyposażonego w kamerę opartego o platformy Raspberry Pi i Arduino. Jego oprogramowanie pozwala na sterowanie poprzez stronę internetową, jak i na automatyczne wykrywanie nieprawidłowości w nadzorowanym otoczeniu za pomocą algorytmów analizy obrazu.

Inna konstrukcja tego typu została opisana w [7]. Robot oparty jest o popularny mikrokomputer Raspberry Pi 4, który steruje silnikami i przetwarza dane z sensorów – czujnika ruchu PIR i kamery. Jedną z funkcjonalności jest wykrywanie twarzy i wysyłanie powiadomień o wykryciu nieznanych osób.

W [10] przedstawiony jest jeżdżący robot z kamerą, oparty na Raspberry Pi i na mikrokontrolerze Arduino. Ma on funkcję autonomicznego patrolowania terenu wraz ze zdolnością do omijania przeszkód dzięki zamontowanym czujnikom odległości.

### 1.4 Istniejące rozwiązania komercyjne

Mimo, że obszar mobilnych robotów nadzorujących może się wydawać nierożwinięty i niedojrzały, to na rynku istnieje kilka firm, których działalność opiera się na tego typu rozwiązaniami.

Jedną z nich jest SMP Robotics. Jej flagową linią produktów są mobilne roboty przeznaczone do nadzoru terenów zewnętrznych. Są to pojazdy wyposażone w kamerę zamontowaną na słupku,

przykładowy model przedstawiony na zdjęciu 1. Oferują m.in.: w pełni autonomiczny patrol,



Rysunek 1. Robot patrolowy firmy SMP Robotics

sterowanie głosowe, rozpoznawanie twarzy w odległości do 50 metrów i system automatycznego ładowania akumulatorów.

Firma Robotnik produkuje roboty do różnych zastosowań, w tym model *RB-Watcher*. Jest on przeznaczony do autonomicznego nadzoru wewnętrz budynków, jak i na zewnątrz. Jego wyposażenie obejmuje bispektralną kamerę, GPS i mikrofon. Do orientacji w przestrzeni wykorzystuje algorytmy typu SLAM (*Simultaneous localization and mapping*). Budową przypomina on wspomnianego wyżej robota 1.4 od SMP Robotics, ale jest wyraźnie niższy. Jest on przedstawiony na zdjęciu 2.

Podobne rozwiązanie oferuje firma Knightscope. Na stronie internetowej produktu chwalą się rezultatami zastosowania w postaci:

- zmniejszenia liczby zgłaszanych przestępstw o 46%,
- zwiększenia liczby aresztów o 27%,
- zmniejszenia liczby wystawionych mandatów o 68%.

Robot firmy Knightscope wygląda inaczej od przedstawionych wcześniej konkurencyjnych produktów – widać to na zdjęciu 3.

Mobilne roboty nadzorujące oferują także firmy: Cobalt AI, Running Brains Robotics, Super Droid Robots i inne.

## 1.5 Motywacja

Obszar mobilnych robotów sprawujących nadzór wydaje się mieć duży potencjał. Jednocześnie rozwiązania z tej dziedziny nie należą do kanonu narzędzi typowo stosowanych do monitoringu. Dlatego jest to nisza, która pozostawia w mojej opinii nadal pole do eksploracji.

Zakres tej pracy jest ciekawy ze względu na konieczną fuzję kilku dziedzin inżynierii – informatyki, elektroniki i mechaniki. Jako że jednak moją specjalizacją jest informatyka, to najbardziej skupiam

## Rozdział 1. Wstęp

się na części związanej z oprogramowaniem robota. W tym obszarze widzę kilka wyzwań związanych z:

- cyberbezpieczeństwem – zabezpieczenie dostępu do aplikacji webowej, danych użytkowników,
- przetwarzaniem danych w czasie rzeczywistym – przy sterowaniu robotem,
- przetwarzaniem multimedialnych – strumieniowanie obrazu z kamery do wielu użytkowników.

Wyzwania te czynią tę pracę okazją do rozwoju.



**Rysunek 2.** Robot patrolowy RB Watcher firmy Robotnik



**Rysunek 3.** Robot patrolowy K5 firmy Knightscope

## Rozdział 2

# Sprzęt

### 2.1 Cel

Celami tej części projektu były:

- zaplanowanie fizycznej budowy,
- zaprojektowanie układu elektronicznego,
- wybranie i pozyskanie odpowiednich części,
- fizyczne skonstruowanie maszyny.

### 2.2 Korpus

Robot z założenia miał poruszać się na kołach, więc nieodzowne były silniki i podwozie. Ze względu na brak doświadczenia w konstrukcji pojazdów, zdecydowano się na dobranie gotowego rozwiązania. Na rynku znaleziono produkt będący zestawem do budowy robota jeżdżącego. Składa się on z:

- dwóch podstaw o rozmiarze 26 x 15 cm,
- czterech silników prądu stałego z przekładnią zasilanych napięciem 6V, o momencie obrotowym 0,78 Nm i poborze prądu 0,19 - 1,0 A,
- czterech kół z gumowymi oponami o średnicy 6,5 cm,
- elementów montażowych – śrubki, dystanse, nakrętki itp.

Zawartość zestawu przedstawiona jest na zdjęciu 4.

Zdecydowano, że biorąc pod uwagę masę konstrukcji, cztery koła jezdne nie są potrzebne i dwa przednie powinny wystarczyć. Na tył robota przeznaczono obrotowe kółko podporowe. Dzięki takiemu rozwiązaniu zostaje więcej miejsca na umiejscowienie innych komponentów pomiędzy platformami. Liczne otwory montażowe w podwoziu zostały wykorzystane do umocowania różnych elementów maszyny.

## 2.3 Elektronika

Wymagania stawiane względem oprogramowania robota były dosyć wysokie – strumieniowanie obrazu z kamery, serwowanie aplikacji internetowej, gromadzenie danych. Jednocześnie potrzebny był interfejs do sterowania silnikami, czy zbierania danych z czujników. Platformą, która łączy świat pełnoprawnych komputerów i elektroniki jest popularne Raspberry Pi. Konkretnym modelem wybranym do realizacji tego projektu jest Raspberry Pi 3B+. Najważniejsze elementy specyfikacji:

- wymiary 85 x 56 mm,
- 64-bitowy procesor ARM o taktowaniu 1.4GHz oraz 1GB RAM,
- karta graficzna Cortex-A53,
- karty Wi-Fi i Ethernet,
- cztery porty USB 2.0,
- porty: 4 x USB, HDMI, RCA, DSI (do wyświetlacza), CSI (do kamery), MicroSD,
- 40 pinów GPIO,
- napięcie zasilania 5V.

Pewną bolączkę stanowił dobór układu zasilania. W porównaniu do np. Arduino, wadą Raspberry Pi jest mała tolerancja zakresu napięcia zasilania – +- 5%. Ponadto nie posiada układów chroniących przed przepięciami, więc nie jest trudno spalić to urządzenie projektując nieodpowiedni układ lub zwyczajnie będąc nieostrożnym. Wybrane do projektu silniki elektryczne wymagają wyższego



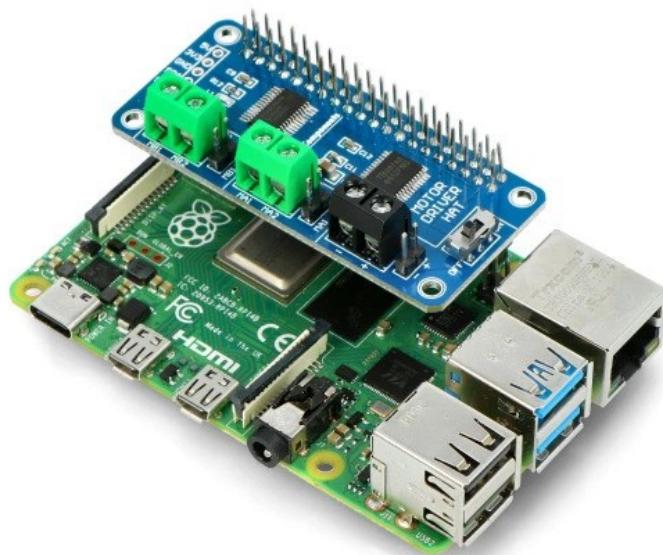
Rysunek 4. Gotowa platforma do budowy robota jeżdżącego

napięcia - 6V. Dodatkowym utrudnieniem jest charakterystyka silników prądu stałego, które generują zakłócenia na linii zasilania i mają wysoki prąd rozruchowy.

Z powyższych względów postawiono na gotowy układ projektowany pod Raspberry Pi. Jest to nakładka *Motor Driver Hat SKU21789* od firmy SB Components, która zawiera:

- dwukanałowy sterownik silników DC o napięciu 6 - 12 V o poborze prądu do 3A,
- generator sygnału PWM o rozdzielczości 12 bitów,
- regulator napięcia 5V do zasilania Raspberry Pi,
- interfejs komunikacyjny I2C.

Komponent ten, zamontowany na Raspberry Pi, przedstawiony jest na zdjęciu 5.



Rysunek 5. Nakładka *Motor Driver Hat SKU21789* na Raspberry Pi

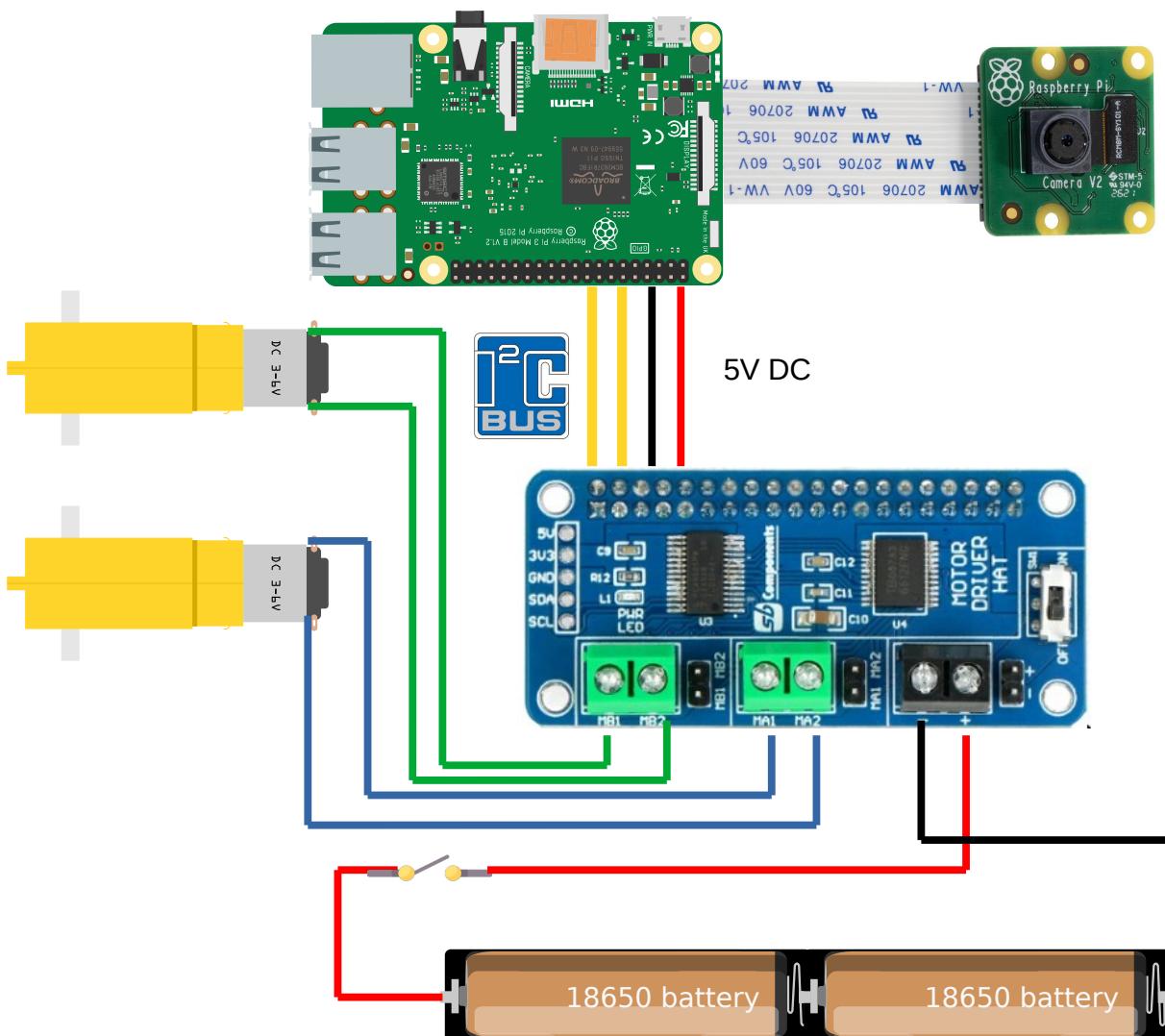
## 2.4 Zasilanie

Ze względu na mobilność robota wybrano zasilanie akumulatorowe. Dzięki zastosowaniu wyżej opisanego układu 2.3, zarówno płytę Raspberry Pi z periferiami, jak i silniki można zasilić z tego samego źródła. Początkowo próbowało zastosować zwykłe baterie AA 1,5V, połączone szeregowo w celu uzyskania źródła napięcia 6V. To rozwiązanie okazało się nieefektywne – Raspberry Pi nawet nie było w stanie w całości przejść sekwencji bootowania. Prawdopodobnie wykorzystane baterie AA nie miały wystarczającej wydajności prądowej.

Ostatecznie zastosowano dwie baterie litowo-jonowe 3,7V typu 18650 o pojemności 3,4 Ah połączone szeregowo. Mają one wydajność prądową klasy 2C, co oznacza, że można z nich pobierać nawet 6,4 A prądu. Rozwiążanie to sprostało wyzwaniu jednoczesnego zasilenia mikrokomputera z peryferiami i silników. Dodatkową stabilność zapewniło podłączenie równolegle dwóch kondensatorów elektrolitycznych o łącznej pojemności 4800  $\mu$ F w celu złagodzenia chwilowych spadków napięcia.

## 2.5 Schemat układu

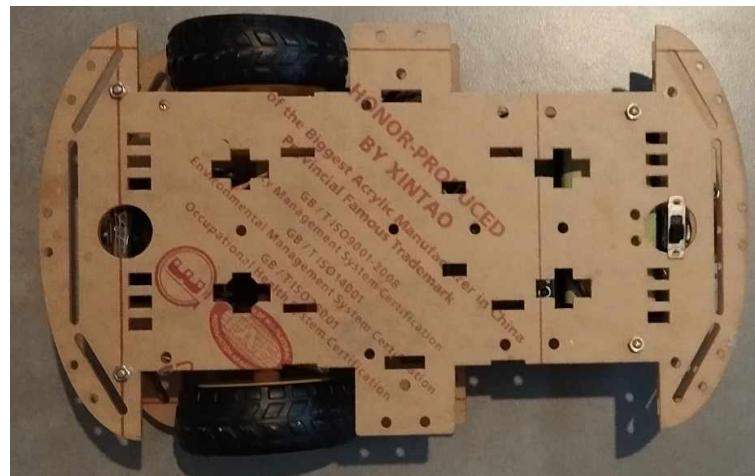
Połączenie części elektronicznych jest przedstawione na schemacie 6.



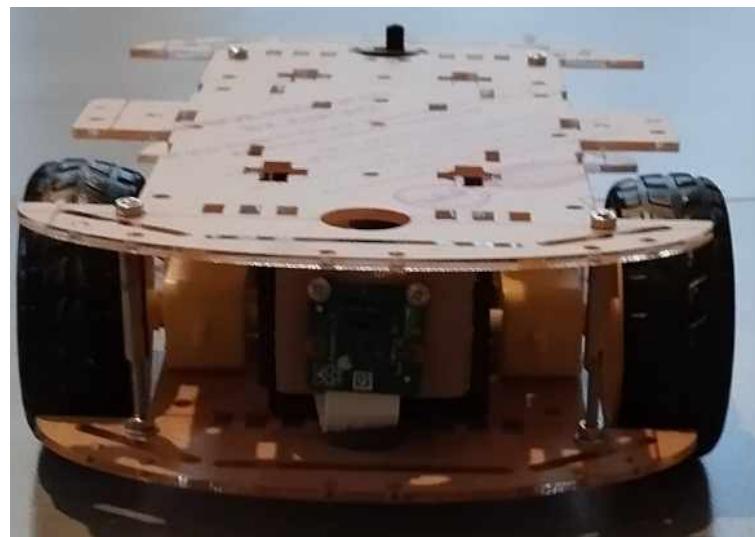
Rysunek 6. Schemat połączeniowy

## 2.6 Złożona konstrukcja

Po złożeniu podwozia i zamontowaniu połączonych elementów elektronicznych konstrukcja była ukończona. Efekt końcowy przedstawiają zdjęcia robota z góry 7, z przodu 8, z prawego boku 10, z lewego boku 9 i z tyłu 11.



Rysunek 7. Zdjęcie robota z góry



Rysunek 8. Zdjęcie robota z przodu

## 2.7 Koszty

Kosztorys przedstawiający wydatki na poszczególne elementy robota, na podstawie sklepu internetowego Botland:

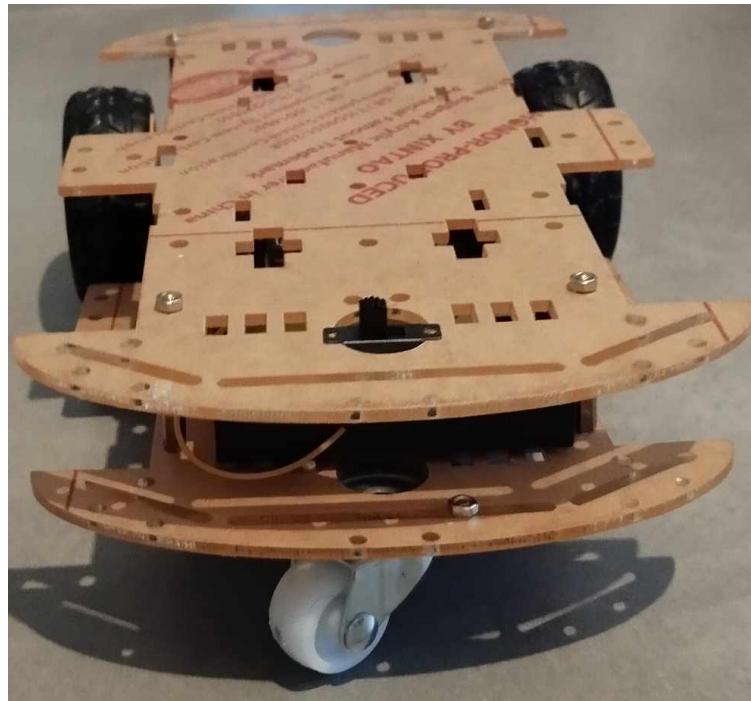
- zestaw podwozia z silnikami – 55 zł,



Rysunek 9. Zdjęcie robota z lewej strony



Rysunek 10. Zdjęcie robota z prawej strony



Rysunek 11. Zdjęcie robota z tyłu

- nakładka Motor Driver Hat – 95 zł,
- Raspberry Pi 3B+ – 180 zł,
- baterie litowo-jonowe – 2 x 21 zł,
- kamera Raspberry Pi Camera V2 – 83 zł,
- przewody i inne drobne elementy – ok. 15 zł.

Całkowity koszt wyniósł ok. 500 zł.

## Rozdział 3

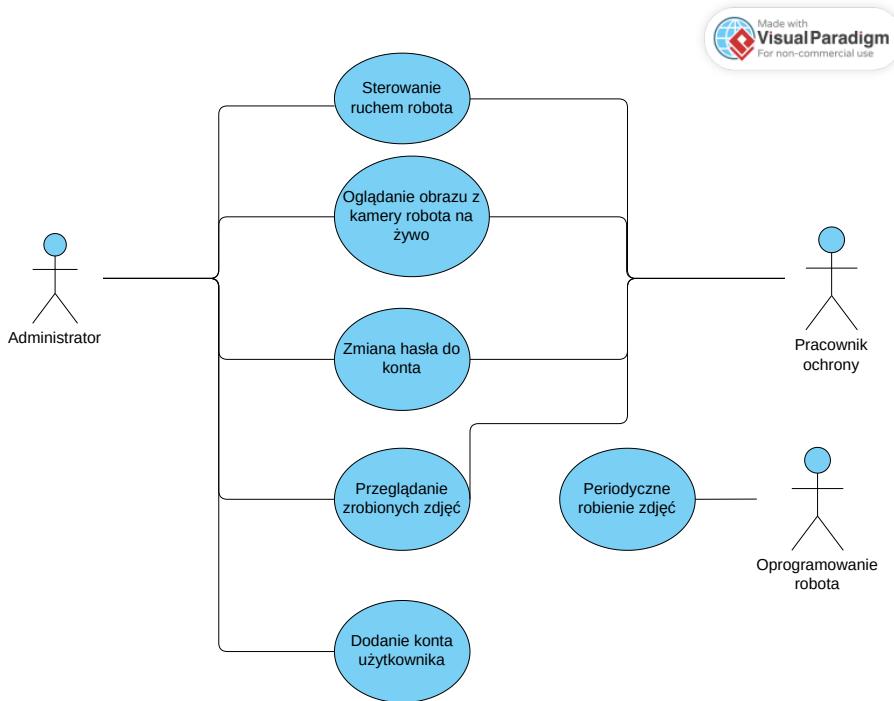
# Oprogramowanie

### 3.1 Definicja wymagań

Celem tej części projektu było zaprojektowanie i napisanie oprogramowania robota. Określono, że będą dwa rodzaje aktorów ludzkich wchodzących w interakcje z systemem:

- administrator – jest tylko jeden i ma dostęp do wszystkich ustawień i zarządzania pozostałymi użytkownikami;
- pracownik ochrony – ma dostęp do kluczowych funkcjonalności;

Funkcjonalności, jakie ma posiadać system zostały zdefiniowane na diagramie przypadków użycia 12.



Rysunek 12. Diagram przypadków użycia

## 3.2 Architektura systemu

Zdecydowano się na interfejs użytkownika w formie aplikacji przeglądarkowej (webowej). Rozwiązanie to pozwala na dostęp do systemu bez instalacji żadnego oprogramowania. Ponadto taka forma aplikacji pozwala na dostęp przez szeroką gamę urządzeń końcowych – komputery, smartfony, tablety itp.

Zgrubny podział aplikacji jest na:

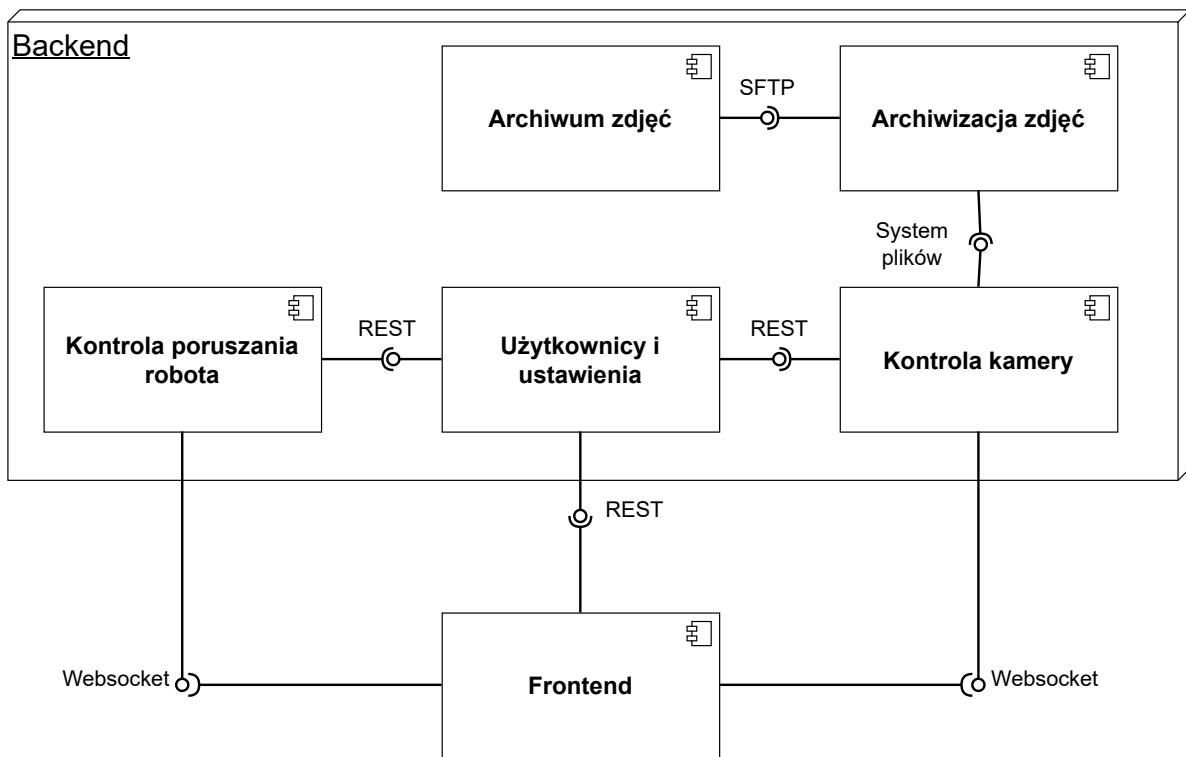
- Frontend

Aplikacja uruchamiana w przeglądarce użytkownika. Prezentuje dane przychodzące z robota użytkownikowi. Przesyła do backendu dane pochodzące od użytkownika.

- Backend

Aplikacja działająca na sprzęcie robota. Wysyłająca dane do frontendu i przetwarzająca przychodzące dane. Steruje zachowaniem robota.

Na backendzie ze względu na różnorodną i wielowarstwową funkcjonalność aplikacji zdecydowano się na architekturę mikroserwisową. Wysokopoziomowa komunikacja modułów między sobą ukazana jest na diagramie komponentów 13.



Rysunek 13. Diagram komponentów

### 3.3 Wdrożenie

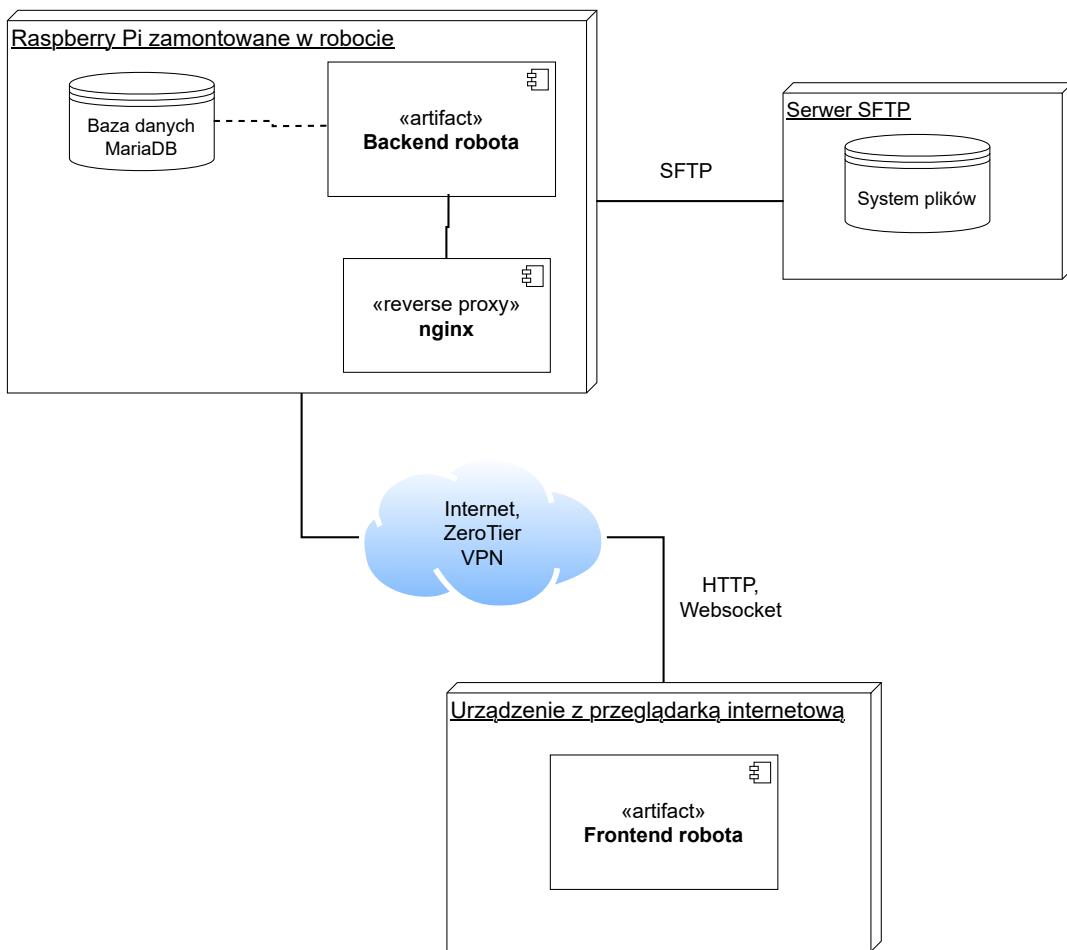
Wdrożona aplikacja ma działać na sprzęcie samego robota i na urządzeniach końcowych użytkowników. Wykorzystując możliwości minikomputera Raspberry Pi stawiany jest na nim serwer backendu i frontendu. Jedynym wykorzystaniem zewnętrznej infrastruktury jest zewnętrzny serwer na pliki, gdzie archiwizowane są zdjęcia. Architektura wdrożeniowa przedstawiona jest na schemacie 14.

We wdrożeniu wszystkie moduły backendowe są schowane za reverse proxy, które stanowi odpowiednio skonfigurowany serwer HTTP *nginx*. Do uruchamiania całej aplikacji napisany został skrypt w bashu. Skrypt ten z kolei jest uruchamiany przez specjalnie napisany serwis *systemd*<sup>1</sup>.

Jeśli chodzi o architekturę sieciową, to robot jest połączony do internetu przez bezprzewodową sieć Wi-Fi. Ponadto, administrator może podłączyć maszynę do wirtualnej sieci prywatnej. *ZeroTier* jest popularnym tego typu oprogramowaniem wykorzystanym w tym projekcie. Dzięki temu rozwiązaniu użytkownicy robota mogą korzystać z systemu robota w jakimkolwiek miejscu z dostępem do internetu. Jednocześnie *ZeroTier* szyfruje cały ruch sieciowy od końca do końca, co gwarantuje bezpieczeństwo i prywatność.

---

<sup>1</sup>*systemd* to menadżer systemu i usług w wielu dystrybucjach systemu operacyjnego GNU/Linux.



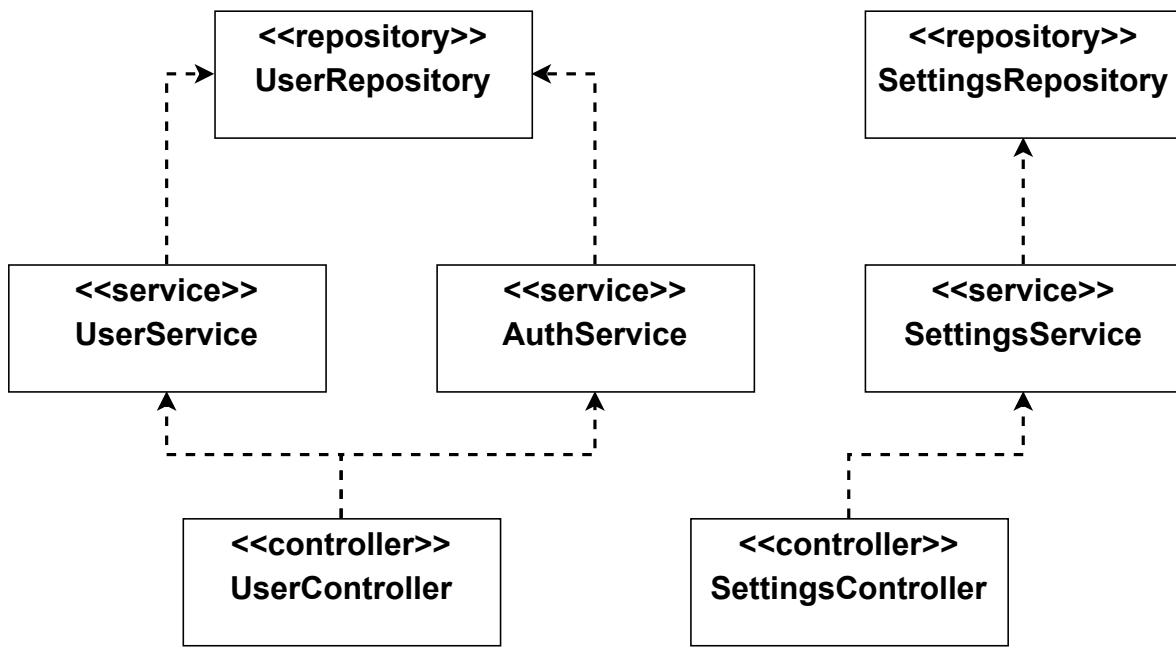
Rysunek 14. Diagram wdrożenia

### 3.4 Serwis użytkowników i ustawień

Program ten jest serwerem HTTP wystawiającym API typu REST do komunikacji z frontendem. Przechowuje dane w lokalnej, relacyjnej bazie danych - *MariaDB*. Aplikacja jest napisana w języku Python i wykorzystuje popularny framework do aplikacji internetowych - *Flask*. Architektura wprowadza podział na warstwy:

- kontrolery – definiują endpointy REST i obsługę zapytań;
- serwisy – zawierają funkcjonalność i logikę biznesową;
- repozytoria – zajmują się przechowywaniem i odczytywaniem danych.

Klasy i zależności pomiędzy nimi ukazane są na diagramie 15.



Rysunek 15. Diagram klas serwisu użytkowników i ustawień

Lista endpointów z opisem:

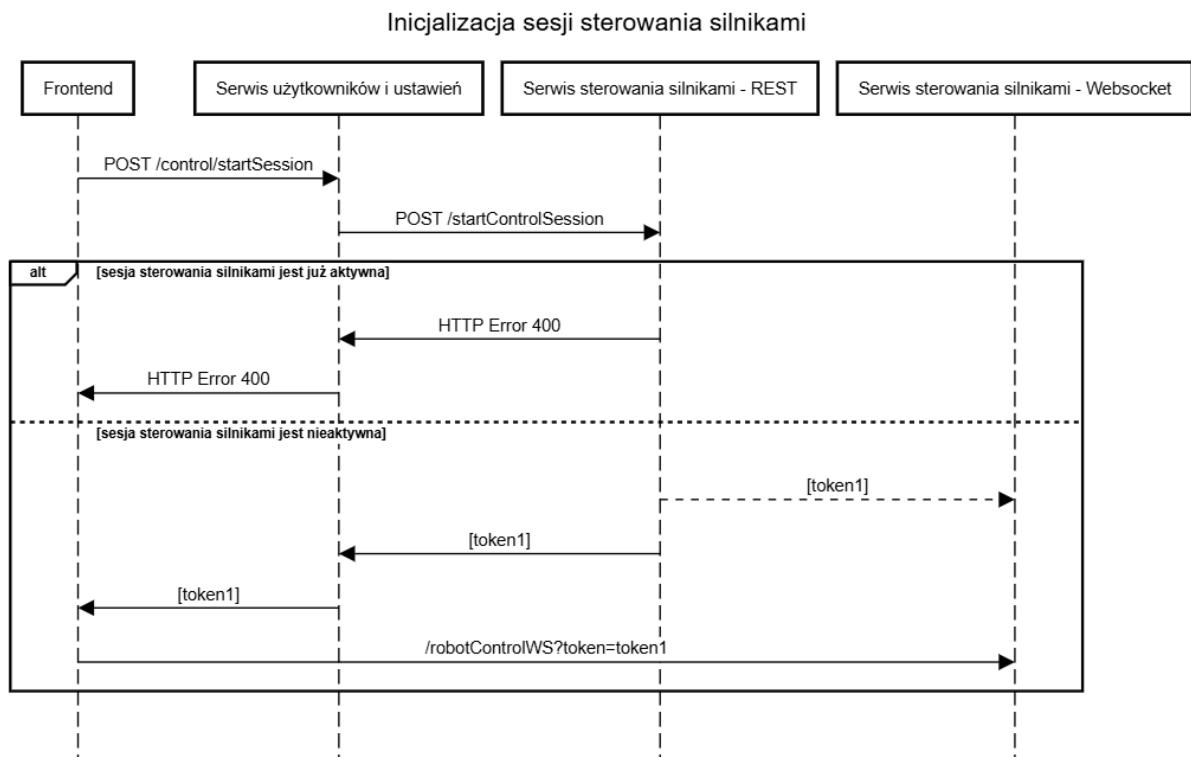
- /add\_user – dodawanie użytkownika;
- /users – listowanie użytkowników;
- /users[id] – operacje na konkretnym użytkowniku;
- /login – zalogowanie użytkownika;
- /logout – wylogowanie użytkownika;
- /check\_session – zweryfikowanie sesji użytkownika;
- /change\_password – zmiana hasła użytkownika;
- /control/startSession – rozpoczęcie sesji sterowania robotem;
- /settings – wylistowanie/zmiana ustawień;
- /zerotier – połączenie do sieci ZeroTier;
- /sftp – wylistowanie, zmiana ustawień transferu zdjęć przez SFTP;
- /ip – informacja o adresach IP robota.

### 3.5 Serwis sterujący silnikami robota

Główną funkcjonalnością tego programu jest przetwarzanie przychodzących komend sterowania silnikami robota i wykonywania ich. Aplikacja napisana jest w języku Python i wystawia serwer Websockets, wykorzystując nowoczesny framework do aplikacji internetowych *FastAPI*. Websockets to popularny protokół komunikacji używany w przeglądarkach internetowych. Umożliwia ciągłą dwustronną komunikację w ramach jednego połączenia TCP.

Program serwuje też niewielkie REST API w celach zarządzania sesją. Ten interfejs nie jest dostępny z zewnątrz, jedynie dla serwisu użytkowników 3.4, który pośredniczy w inicjalizacji sesji tylko jeśli użytkownik jest już zautoryzowany. Wtedy wydawany jest token sesji, który służy do autoryzacji do serwera Websocket. Jeśli jakiś użytkownik ma już aktywną sesję sterowania silnikami, to nikt inny nie może takiej sesji zacząć do czasu aż sesja zostanie zakończona. Proces ten przedstawiony jest na diagramie sekwencji 16.

Serwer Websocket i REST są uruchamiane jako oddzielne procesy wymieniające się danymi przez kolejkę asynchroniczną.



Rysunek 16. Diagram sekwencji inicjalizacji sesji sterowania silnikami

Wiadomości przesyłane przez Websocket mają popularny format JSON. Przykładowa wiadomość:

```
{
    "type": "motorControl",
    "params": {
        "leftMotor": {
            "thrust": 50
        },
        "rightMotor": {
            "thrust": -25
        }
    }
}
```

}

Wiadomość ta żąda włączenia 50% mocy w lewym silniku i -25% w prawym. W przypadku braku nowych wiadomości od klienta w przeciągu pół sekundy, silniki zostają zatrzymane. Komunikacja po interfejsie I<sup>2</sup>C odbywa się za pomocą dostarczonej przez producenta sterownika silników 2.3 biblioteki dla języka Python.

### 3.6 Serwis kamery

Funkcją tego programu jest strumieniowanie obrazu z kamery robota na żywo oraz regularne robienie zdjęć. Jest on, podobnie jak serwis sterujący silnikami 3.5, napisany w Pythonie i wykorzystuje framework *FastAPI*. Jako że do Raspberry Pi jest podpięta dedykowana kamera, to program może korzystać z dobrzej dedykowanej biblioteki w języku Python - *picamera*.

Program strumieniuje obraz w formacie *MotionJPEG*. Jest to prosty format kompresji wideo typu intraframe – każda klatka kompresowana jest oddzielnie w formacie JPEG. Protokołem opakowującym jest HTTP. Taki format strumieniowania wideo jest powszechnie wspierany przez przeglądarki internetowe. Aplikacja wspiera rozsyłanie materiału do wielu klientów naraz. Autoryzacja zapytań odbywa się przez ciaścęko sesji, które jest weryfikowane przez serwis użytkowników i ustawień 3.4.

### 3.7 Serwis archiwizacji zdjęć

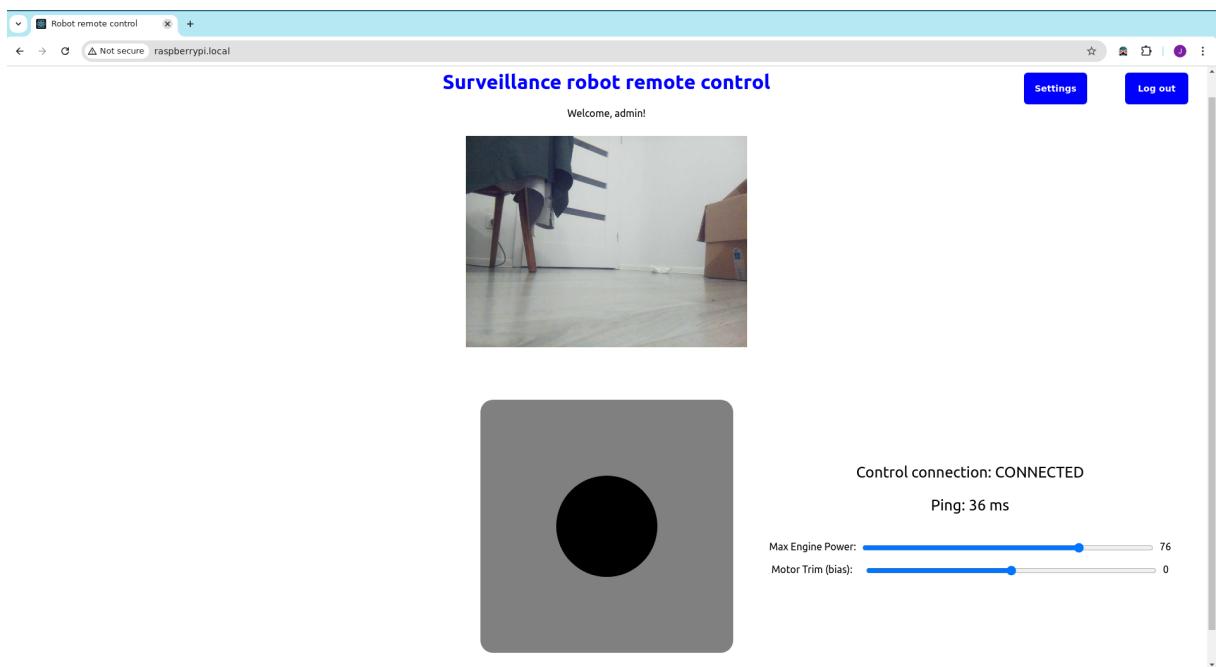
Rolą tego programu jest archiwizacja zrobionych zdjęć na zewnętrznym serwerze. Aplikacja ta jest napisana w języku Python. Pakuje wszystkie zdjęcia z podanego katalogu do formatu ZIP. Następnie przesyła skompresowany plik na zdalny serwer za pomocą protokołu SFTP. Namiary i dane autoryzacyjne są pobierane z lokalnej bazy danych – z tej samej, z której korzysta serwis użytkowników i ustawień 3.4.

We wdrożeniu program ten ma serwis uruchomieniowy *systemd*. Uruchamiany jest okresowo korzystając z systemu *timer-ów systemd*.

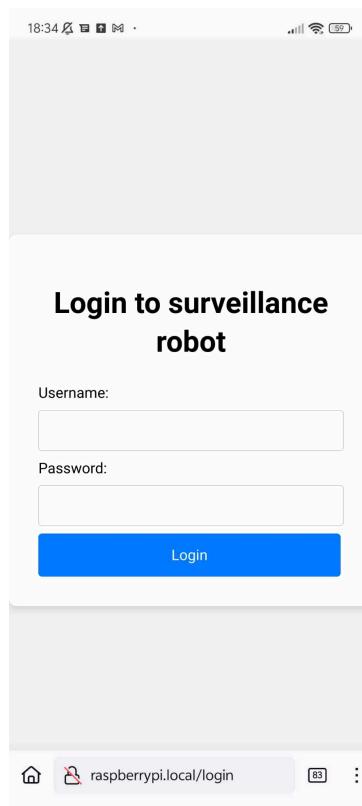
### 3.8 Strona internetowa

Oprogramowanie pełniące rolę interfejsu użytkownika to strona internetowa. Pierwsza rzecz, którą widzi użytkownik po jej załadowaniu to ekran logowania, który jest pokazany na zrzucie ekranu 17. Po pomyślnym zalogowaniu ukazuje się strona główna. Na niej jest wyświetlany podgląd z kamery robota. Poniżej, automatycznie uruchamia się sesja i interfejs sterowania robotem. Na górze są przyciski do przejścia do widoku ustawień lub wylogowania się. Elementy te są zaprezentowane na zrzutach ekranu strony głównej na dużym ekranie: 18 i na ekranie telefonu: 19.

### 3.8. Strona internetowa

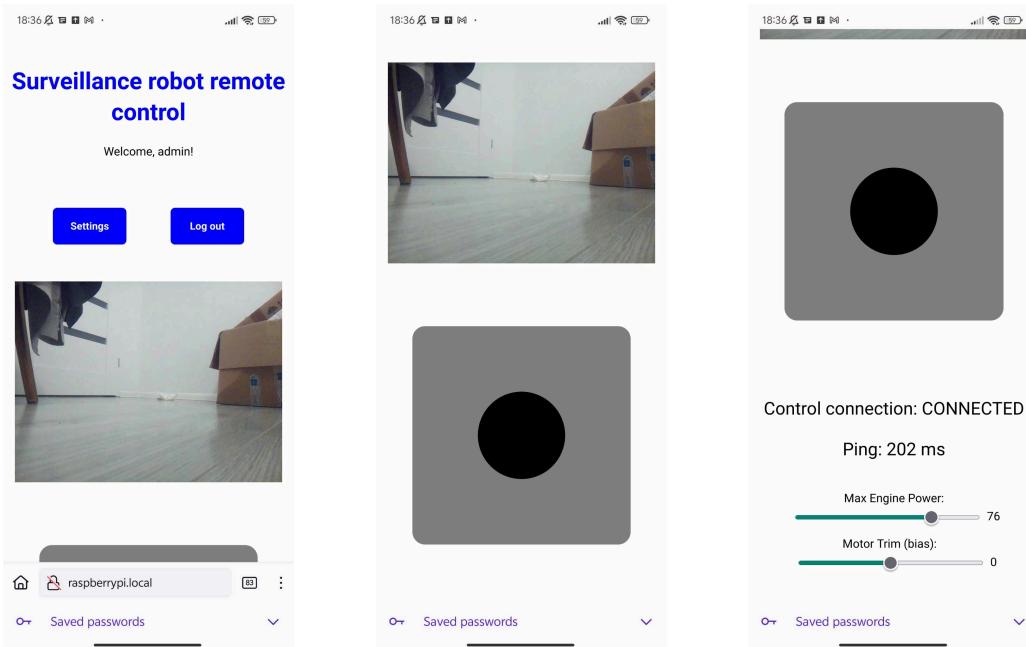


Rysunek 18. Zrzut ekranu strony głównej na komputerze



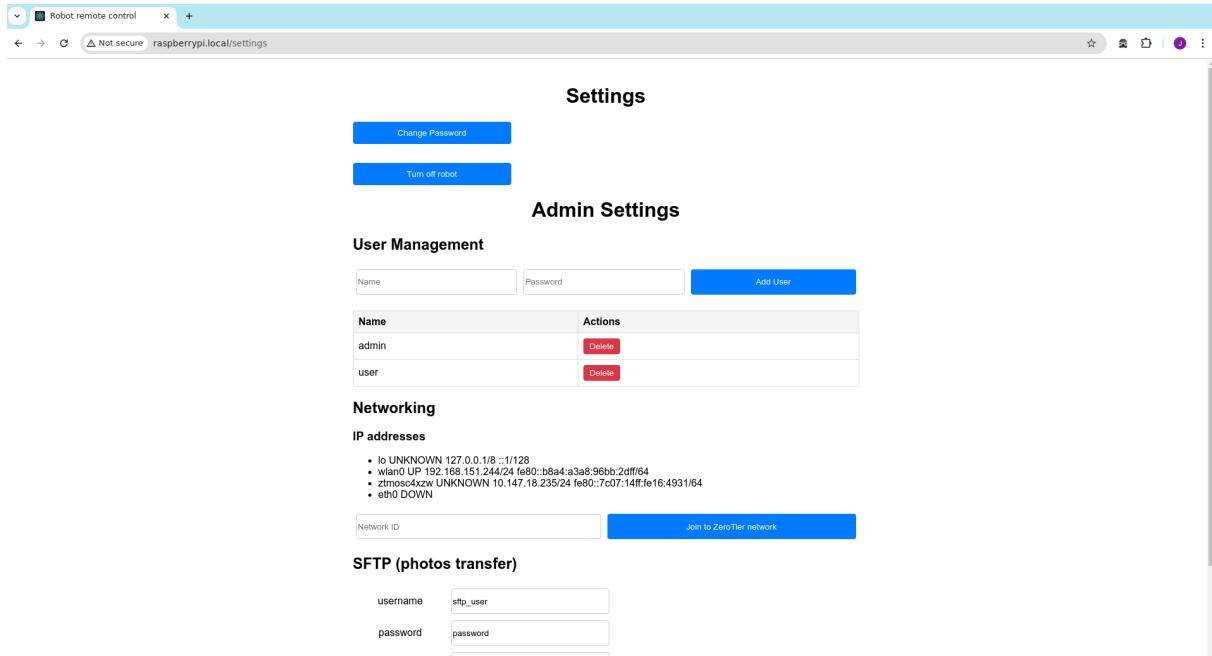
Rysunek 17. Zrzut ekranu strony logowania na telefonie

### Rozdział 3. Oprogramowanie

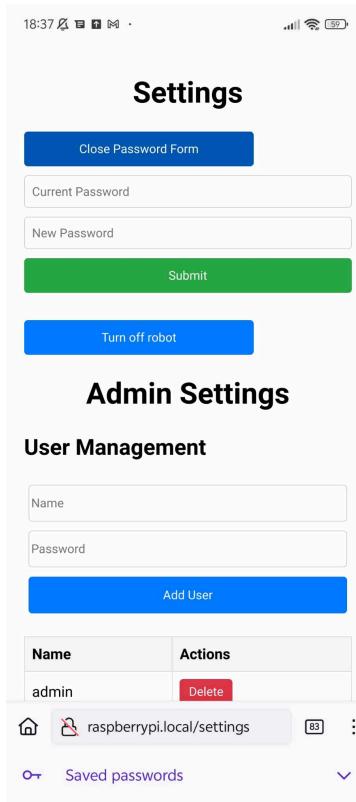


Rysunek 19. Zrzut ekranu strony głównej na telefonie

Po kliknięciu odpowiedniego przycisku na stronie głównej otwiera się strona ustawień. Jej wygląd jest ukazany na zrzutach ekranu w wersji na dużym ekranie: 20 i na ekranie telefonu: 21.



Rysunek 20. Zrzut ekranu strony ustawień na komputerze



Rysunek 21. Zrzut ekranu strony ustawień na telefonie

Z jej poziomu zwykły użytkownik może zmienić hasło do swojego konta bądź wyłączyć robota.

Administrator może:

- dodać użytkownika,
- usunąć użytkownika,
- odczytać adresy IP na wszystkich interfejsach sieciowych robota,
- dodać robota do sieci VPN *ZeroTier*
- odczytać i zmienić dane dostępowe serwera SFTP, gdzie archiwizowane są zdjęcia.

Jeśli chodzi o technologię wykonanie, to nie było tutaj możliwości szerokiego wyboru języków – standardem są HTML i CSS do definicji wyglądu oraz JavaScript do zaprogramowania zachowania. Został wykorzystany najpopularniejszy framework webowy dla Javascript-u – *React*.

## 3.9 Moduł joysticka do sterowania ruchem robota

Kod strony internetowej odczytuje pozycję joysticka i na jej podstawie oblicza wysterowanie silników. Logika sterowania opiera się o algorytm, który przekłada pozycję joysticka na prędkość obrotową kół. Do skręcania stosowana jest metoda sterowania różnicowego. Im większy wychył joysticka do przodu/tyłu, tym większa prędkość obrotowa kół. Im większy wychył joysticka w lewo/prawo, tym większa różnica prędkości obrotowej kół. Kod odpowiedzialny za te obliczenia: 1.

```
1 function rescaleJoystickPos(posValue) {
2     const pos = Math.abs(posValue) * 100
3     if (pos <= JOYSTICK_DEADZONE) return 0
4     else {
5         return (pos - JOYSTICK_DEADZONE) / (100 - JOYSTICK_DEADZONE) * 100
6     }
7 }
8 function handleJoystickMove(position) {
9     const power = rescaleJoystickPos(position.y)
10    const turn = rescaleJoystickPos(position.x)
11    let outsideWheelSpeed = power
12    let insideWheelSpeed = (100 - turn) * power / 100
13    if (position.y < 0) {
14        outsideWheelSpeed *= -1
15        insideWheelSpeed *= -1
16    }
17    if (position.x > 0) {
18        leftMotThrust.current = outsideWheelSpeed
19        rightMotThrust.current = insideWheelSpeed
20    } else {
21        leftMotThrust.current = insideWheelSpeed
22        rightMotThrust.current = outsideWheelSpeed
23    }
24 }
```

**Listing 1.** Kod obliczający moc silników na podstawie wychylenia joysticka

Parametry wysterowania są wysyłane do serwera platformy nieustannie co stały interwał czasu.

# Rozdział 4

## Testy

### 4.1 Testy oprogramowania

W czasie wytwarzania oprogramowanie było testowane przez 3 rodzaje testów automatycznych:

- jednostkowe - sprawdzające poprawność logiki kodu,
- integracyjne - weryfikujące poprawność zachowania poszczególnych mikroserwisów,
- end-to-end - symulujące zachowanie użytkownika.

Testy integracyjne serwisu użytkowników i ustawień 3.4 zostały napisane z wykorzystaniem biblioteki *Flask testing*. W przypadku serwisów napisanych za pomocą framework'a FastAPI wykorzystano jego wbudowane biblioteki do testowania.

Testy end-to-end zostały zbudowane za pomocą nowoczesnego narzędzia przeznaczonego do tego celu – Cypress. Użyto języka Javascript.

Ponadto zostały przeprowadzone szerokie testy manualne, których scenariusze zostały dopasowane do przypadków użycia zdefiniowanych w wymaganiach 12. Aby można było testować oprogramowanie bez uruchamiania całego robota, klasy odpowiadające za sterowanie silnikami i zbieranie obrazu z kamery zostały zastąpione klasami pozorowanymi (ang. *mock*).

### 4.2 Testy robota

Po ukończeniu budowy sprzętu i oprogramowania przystąpiono do sprawdzania działania całego rozwiązania. Powtórzono napisane wcześniej automatyczne testy end-to-end, tym razem na w pełni wdrożonym systemie.

Głównym przypadkiem użycia robota jest zdalne sterowanie przez internet. Zostało to przetestowane przez autora projektu i osobę postronną w następujących scenariuszach:

Scenariusz 1 – sterowanie robotem w sieci lokalnej. Operator był podłączony na komputerze do tej samej sieci bezprzewodowej Wi-Fi, co robot. Średnie opóźnienie (ping) wynosił ok. 15 ms.

Scenariusz 2 – sterowanie robotem przez internet na dobrym łączu. Operator był podłączony na komputerze do innej sieci niż robot. Sieć robota i operatora były połączone światłowodem. Średnie opóźnienie (ping) wynosił ok. 30 ms.

Scenariusz 3 – sterowanie robotem przez internet na łączu mobilnym. Operator łączył się z robotem na telefonie korzystając z bezprzewodowej sieci komórkowej 4G LTE. Średnie opóźnienie (ping) wynosił ok. 60 ms.

Robot był umieszczony na płaskiej podłodze w biurze, gdzie były typowe dla tego środowiska przeszkody.

Subiektywne odczucia po testach były takie, że robot jest w stanie spełniać swoją rolę. Jednak sterowanie nie było łatwe i wymagało przyzwyczajenia. Często ruchy robota wydawały się zbyt gwałtowne. Zmniejszenie parametru maksymalnej mocy silników pomagało w płynniejszej jeździe. W scenariuszu 3 trochę przeszkadzało opóźnienie transmisji obrazu z kamery, szczególnie, że jest on ograniczony do 10 klatek na sekundę.

Robot nie radzi sobie z pokonywaniem wysokich progów i innych podobnych przeszkód. Jeszcze większym oczywistym problemem są drzwi lub schody.

Kolejnym testem było zmierzenie zużycia energii elektrycznej. W tym celu zmierzono natężenie prądu pobieranego z akumulatorów multimetrem. Stwierdzono, że w bezruchu pobierane jest ok. 0.35 A. W trakcie ruchu silników z wypełnieniem PWM 50% natężenie wynosiło ok. 0.7 A. Na podstawie tych danych i pojemności akumulatorów 3.4 Ah obliczono, że bez jazdy robot może działać przez około 8 godzin.

# Rozdział 5

## Podsumowanie

### 5.1 Realizacja celu

Cel pracy uznaje się za zrealizowany. Efektem końcowym jest mobilny robot z kamerą, który jest sterowany przez stronę internetową. Posiada on założone funkcjonalności, które czynią go zdatnym do wykorzystania do monitoringu budynku.

Trzeba jednak przyznać, że część sprzętowa została zrealizowana tanim kosztem i ma charakter raczej prototypowy. Przez to, praktyczne zastosowanie robota jest ograniczone. Na przykład, słabo radzi sobie z pokonywaniem przeszkód.

Za to zbudowany system informatyczny ma wiele mocnych stron i jest gotowy do dalszego rozwoju. Komunikacja między stroną internetową a robotem działa sprawnie, a interfejs jest intuicyjny i responsywny. Architektura systemu jest modularna i pozwala na łatwe dodawanie nowych funkcjonalności.

### 5.2 Możliwości rozwoju

W przyszłości można rozwinąć projekt w kilku kierunkach:

- Dodanie elementów autonomiczności. Robot mógłby samodzielnie patrolować zadany obszar lub ścieżkę. Można by wykorzystać np. algorytmy typu SLAM do mapowania otoczenia na podstawie obrazu z kamery. Duże wartości dodałyby funkcjonalność samodzielnego dokowania do stacji ładowania.
- Dołożenie większej liczby czujników. Wartość robota jako urządzenia nadzorującego urosłaby, gdyby mógł on zbierać więcej informacji o otoczeniu. Można by dodać kamery termowizyjne, mikrofony.
- Zwiększenie mobilności. Robot mógłby być bardziej wszechstronny, gdyby mógł pokonywać większe przeszkody.
- Wydłużenie czasu pracy. Obecnie robot może działać przez około 8 godzin. Można tę wartość łatwo zwiększyć przez zastosowanie większych akumulatorów.



# Bibliografia

- [1] Banks, A. i Porcello, E., *Learning React: Modern Patterns for Developing React Apps*. O'Reilly Media, 2020, ISBN: 9781492051695. adr.: <https://books.google.pl/books?id=tDjrDwAAQBAJ>.
- [2] Elharrouss, O., Almaadeed, N. i Al-Maadeed, S., „A review of video surveillance systems”, *Journal of Visual Communication and Image Representation*, t. 77, s. 103116, 2021, ISSN: 1047-3203. DOI: <https://doi.org/10.1016/j.jvcir.2021.103116>. adr.: <https://www.sciencedirect.com/science/article/pii/S1047320321000729>.
- [3] Grand View Research, Inc., „Video Surveillance Market Size, Share & Trends Analysis Report By Component (Hardware, Software), By Vertical (Commercial, Residential), By System (Analog Video Surveillance, IP Video Surveillance), By Region, And Segment Forecasts, 2025 - 2030”, Grand View Research, Inc., spraw. tech., 2024.
- [4] Kim, K., Bae, S. i Huh, K., „Intelligent surveillance and security robot systems”, w *2010 IEEE Workshop on Advanced Robotics and its Social Impacts*, 2010, s. 70–73. DOI: [10.1109/ARSO.2010.5679624](https://doi.org/10.1109/ARSO.2010.5679624).
- [5] Lombardi, A., *WebSocket: Lightweight Client-Server Communications*. O'Reilly Media, 2015, ISBN: 9781449369255. adr.: <https://books.google.pl/books?id=4ZWJCgAAQBAJ>.
- [6] Massé, M., *REST API Design Rulebook*. O'Reilly, 2012, ISBN: 9781449317904. adr.: <https://books.google.pl/books?id=e1r0jwEACAAJ>.
- [7] Meddeb, H., Abdellaoui, Z. i Houaidi, F., „Development of surveillance robot based on face recognition using Raspberry-Pi and IOT”, *Microprocessors and Microsystems*, t. 96, s. 104728, 2023, ISSN: 0141-9331. DOI: <https://doi.org/10.1016/j.micpro.2022.104728>. adr.: <https://www.sciencedirect.com/science/article/pii/S0141933122002575>.
- [8] Mwaura, W., *End-to-End Web Testing with Cypress: Explore techniques for automated front-end web testing with Cypress and JavaScript*. Packt Publishing, 2021, ISBN: 9781839215636. adr.: <https://books.google.pl/books?id=Er0YEAAAQBAJ>.
- [9] Ramalho, L., *Fluent Python*, 1st. O'Reilly Media, Inc., 2015, ISBN: 1491946008.
- [10] Rani, V. U., Sridevi, J. i Sai, P. M., „Web Controlled Raspberry Pi Robot Surveillance”, w *2021 International Conference on Sustainable Energy and Future Electric Transportation (SEFET)*, 2021, s. 1–5. DOI: [10.1109/SeFet48154.2021.9375666](https://doi.org/10.1109/SeFet48154.2021.9375666).

## *Bibliografia*

---

- [11] Shin, M. S., Kim, B. C., Hwang, S. M. i Ko, M. C., „Design and implementation of IoT-based intelligent surveillance robot”, *Studies in Informatics and Control*, t. 25, nr. 4, s. 422, 2016.
- [12] Tsakanikas, V. i Dagiuklas, T., „Video surveillance systems-current status and future trends”, *Computers & Electrical Engineering*, t. 70, s. 736–753, 2018, ISSN: 0045-7906. DOI: <https://doi.org/10.1016/j.compeleceng.2017.11.011>. adr.: <https://www.sciencedirect.com/science/article/pii/S0045790617311813>.
- [13] Valentino-Devries, J., Angwin, J. i Stecklow, S., „Document Trove Exposes Surveillance Methods”, *Wall Street Journal*, list. 2011, <<https://www.wsj.com/articles/SB10001424052970203611404577044192607407780>>.

# Spis rysunków

|    |  |    |
|----|--|----|
| 1  | Robot patrolowy firmy SMP Robotics . . . . .                         | 11 |
| 2  | Robot patrolowy <i>RB Watcher</i> firmy Robotnik . . . . .           | 12 |
| 3  | Robot patrolowy <i>K5</i> firmy Knightscope . . . . .                | 12 |
| 4  | Gotowa platforma do budowy robota jeżdżącego . . . . .               | 14 |
| 5  | Nakładka <i>Motor Driver Hat SKU21789</i> na Raspberry Pi . . . . .  | 15 |
| 6  | Schemat połączeniowy . . . . .                                       | 16 |
| 7  | Zdjęcie robota z góry . . . . .                                      | 17 |
| 8  | Zdjęcie robota z przodu . . . . .                                    | 17 |
| 9  | Zdjęcie robota z lewej strony . . . . .                              | 18 |
| 10 | Zdjęcie robota z prawej strony . . . . .                             | 18 |
| 11 | Zdjęcie robota z tyłu . . . . .                                      | 18 |
| 12 | Diagram przypadków użycia . . . . .                                  | 20 |
| 13 | Diagram komponentów . . . . .  | 21 |
| 14 | Diagram wdrożenia . . . . .  | 23 |
| 15 | Diagram klas serwisu użytkowników i ustawień . . . . .               | 24 |
| 16 | Diagram sekwencji inicjalizacji sesji sterowania silnikami . . . . . | 25 |
| 18 | Zrzut ekranu strony głównej na komputerze . . . . .                  | 27 |
| 17 | Zrzut ekranu strony logowania na telefonie . . . . .                 | 27 |
| 19 | Zrzut ekranu strony głównej na telefonie . . . . .                   | 28 |
| 20 | Zrzut ekranu strony ustawień na komputerze . . . . .                 | 28 |
| 21 | Zrzut ekranu strony ustawień na telefonie . . . . .                  | 29 |