



POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Imię i nazwisko dyplomanta: Dominik Stec

Nr albumu: s12623

Kierunek studiów: Informatyka

Rodzaj studiów: niestacjonarne

Praca dyplomowa

Temat pracy: Soft Drive – system IT wspomagający jazdę motocyklem

Temat w języku angielskim: Soft Drive – IT system supporting motorcycling

Opiekun pracy: dr hab. Bartosz Marcinkowski

Streszczenie:

Celem realizowanego zakresu projektu jest konstrukcja prototypu system IT, składającego się z dwóch modułów: części sprzętowej oraz oprogramowania, które mają zapewnić lepszą orientację kierowcy motocykla odnośnie zagrożeń bezpieczeństwa ruchu drogowego podczas prowadzenia pojazdu. Prewencja mająca na celu bezpieczne prowadzenie pojazdu została uzyskana poprzez monitorowanie istotnych z punktu widzenia kierowcy danych od za pośrednictwem elektronicznych sensorów i sprzętowej warstwy pośredniej w postaci systemu wbudowanego, realizowanej przez minikomputer, transmitującego rejestrowane dane do aplikacji mobilnej systemu w celu ich przetworzenia i prezentacji na ekranie smartfona w formie animacji działającej w czasie rzeczywistym. Projekt zrealizowano stosując metodykę ewolucyjno-przyrostową. W celu zwiększenia niezawodności rozwiązania wykonano testy prototypu oraz zastosowano mechanizmy kontroli odczytanych z sensora danych. Produktem realizacji projektu jest przygotowany do dalszego rozwoju i wdrożenia system IT wspomagający bezpieczne prowadzenie pojazdu motocyklowego.

Gdańsk, 2021 rok

Oświadczenie autora pracy dyplomowej

Świadomy odpowiedzialności prawnej oświadczam, że niniejszą pracę dyplomową w zakresie przedstawionym przeze mnie wykonałem samodzielnie i nie zawiera ona treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że praca w przedstawionym przeze mnie zakresie nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu ukończenia studiów wyższych.

Oświadczam ponadto, że niniejsza wersja pracy dyplomowej jest identyczna z załączoną wersją elektroniczną.



Karta dyplomowego projektu inżynierskiego

Temat projektu: Soft Drive - System IT wspomagający jazdę motocyklem	Akronim: SD Data ustalenia tematu: 20.10.2019 r.
Opiekun: dr hab. Bartosz Marcinkowski	Konsultanci: dr inż. Stanisław Szejko
Cele projektu: Utworzenie prototypu systemu IT dedykowanego kierowcom pojazdów motocyklowych, składającego się z podzespołów elektronicznych i oprogramowania, w celu ułatwienia kierującemu pojazdem poruszanie się w rzeczywistych warunkach ruchu drogowego, dzięki zmniejszeniu ryzyka wystąpienia kolizji drogowej, poprzez informowanie na bieżąco kierowcy o sytuacjach zagrażających bezpieczeństwu ruchu drogowego.	
Rezultaty projektu: Aplikacja mobilna z graficzną animacją w systemie Android z zapewnionym dostępem do internetu poprzez operatora sieci GSM, skomunikowana bezprzewodowo z oprogramowanym minikomputerem Raspberry Pi sterującym pracą czujnika odległości i modułem łączności WiFi oraz elementami prototypu dostosowanymi do testowania i instalacji na pojeździe samochodowym i zapewnionym niezależnym od pojazdu zasilaniem wraz z dołączoną dokumentacją całego projektu.	
Miary sukcesu: Realizacja ustalonych założeń projektowych w wymaganym czasie i przy posiadanych zasobach.	
Ograniczenia: Realizacja projektu nie zakłada prezentacji działania systemu zainstalowanego na pojeździe motocyklowym w rzeczywistych warunkach ruchu drogowego.	

Wykonawcy	Numer albumu	Specjalizacja	Tryb studiów:
Dominik Stec	s12623	Aplikacje Internetowe	niestacjonarne

Data ukończenia projektu:	Recenzent:
----------------------------------	-------------------

Oceny projektu

Semestr dyplomowy I

<i>Wykonawcy</i>	<i>Nr albumu</i>	<i>Ocena sem. I</i>	<i>Data oceny</i>	<i>Podpis</i>
Dominik Stec	s12623			

Semestr dyplomowy II

<i>Wykonawcy</i>	<i>Numer albumu</i>	<i>Ocena promotora</i>	<i>Podpis</i>	<i>Ocena I recenzenta</i>	<i>Podpis</i>	<i>Ocena II recenzenta</i>	<i>Podpis</i>	<i>Data oceny</i>
Dominik Stec	s12623							

Spis treści

1 Wstęp	8
1.1 Kontekst projektu	8
1.2 Opis problemu	8
1.2.1 Społeczne aspekty realizacji projektu	8
1.2.2 Uzasadnienie zapotrzebowania realizacji projektu	10
1.2.3 Motywacja celów projektu	12
1.3 Zakres pracy	14
1.4 Realizacja pracy	14
1.5 Rezultaty	15
1.6 Podziękowania	15
2 Dokument Założeń Wstępnych	16
2.1 Zakres etapu projektowania	16
2.2 Cele projektu	19
2.3 Zakres systemu	19
2.4 Wymagania jakościowe i inne	20
2.5 Wizja konstrukcyjna	21
2.6 Ograniczenia	21
2.6.1 Produktowe	21
2.6.2 Projektowe	21
3 Wstępny Plan Projektu	22
3.1 Założenia projektu	22
3.1.1 Cele projektu	22
3.1.2 Spodziewane produkty	22
3.1.3 Udziałowcy	23
3.1.4 Uwarunkowania, ograniczenia, założenia strategii	23
3.1.5 Priorytety	24
3.1.6 Analiza zagrożeń	24
3.1.7 Budżet	24
3.2 Wizja rozwiązania	25
3.3 Technologia i zamierzone środowisko	25
3.4 Planowany proces wytwarzania	25
3.4.1 Wybrany sposób dokumentowania	25
3.4.2 Wybór strategii prowadzenia prac	26
3.4.3 Schemat procesu wytwarzania	29
3.5 Zakładane zasoby	31
4 Projekt Systemu	32
4.1 Architektura	32
4.1.1 Użyte wzorce projektowe	32

4.1.2	Architektura systemu	33
4.1.3	Dekompozycja systemu na podsystemy	34
4.1.4	Opis interakcji pomiędzy podsystemami	41
4.2	Wykorzystane elementy	46
4.2.1	Część sprzętowa	46
4.2.2	Część programowa	47
4.3	Projekt rozwiązań sprzętowych	48
4.4	Projekt interfejsu	48
4.4.1	Interfejs użytkownika	48
4.4.2	Założenia konstrukcji interfejsu	48
4.4.3	Projekt ekranów	49
4.5	Projekt bazy danych	51
4.5.1	Schemat implementacyjny bazy	51
4.5.2	Model logiczny dla bazy	51
4.5.3	Specyfikacja tabeli bazy	54
5	Wdrożenie rozwiązania	55
5.1	Pierwsza konfiguracja systemu	55
5.1.1	Opis wymagań wstępnych	55
5.1.2	Przygotowanie środowiska instalacyjnego systemu Raspbian	56
5.1.3	Pierwsze uruchomienie systemu Raspbian	65
5.1.4	Problem z automatycznym logowaniem do systemu	70
5.2	Podłączenie kamery	73
5.2.1	Instalacja kamery	73
5.2.2	Konfiguracja kamery	76
5.2.3	Instalacja chłodzenia	81
5.3	Podłączenie skanera obiektów	82
5.3.1	Zasadność zastosowania w projekcie	82
5.3.2	Pierwsze uruchomienie skanera	82
5.3.3	Oprogramowanie producenta	84
5.3.4	Modyfikacja kodu producenta	85
5.3.5	Konfiguracja środowiska obsługi REST API	85
5.3.6	Automatyzacja uruchamiania skryptów startowych	86
5.4	Podłączenie czujnika odległości	88
5.4.1	Konstrukcja oprogramowania w języku C++	89
5.4.2	Konstrukcja oprogramowania w języku Python	93
5.4.3	Problemy z wdrożeniem	95
5.5	Aplikacja mobilna	102
5.6	Przepływ sterowania w aplikacji	103
5.6.1	Aktywacja wymaganych usług	103
5.6.2	Ustalenie adresu IP serwera	106
5.6.3	Usługa REST API	109
5.6.4	Baza danych	111
5.6.5	Graficzny interfejs użytkownika	112
5.7	Prototyp części sprzętowej	117
5.7.1	model	117
5.7.2	Wdrożenie	118
5.8	Laminat PCB	120
6	Testowanie rozwiązań	121

6.1	Testy jednostkowe	121
6.1.1	Testy inicjalizacji systemu	123
6.1.2	Testy usługi rest	125
6.1.3	Testy bazy danych	126
6.2	Refaktoryzacja implementacji	127
6.3	Implementacja po refaktoryzacji	128
6.4	Testy użyteczności	133
7	Opis przyrostów	137
7.1	Część sprzętowa	137
7.2	Kryteria akceptacji części sprzętowej	138
7.3	Dokumentacja	138
7.4	Kryteria akceptacji dokumentacji	140
7.5	Oprogramowanie	141
7.6	Kryteria akceptacji oprogramowania	141
7.7	Testowanie	142
7.8	Wdrożenie	142
7.9	Kryteria akceptacji wdrożenia	143
8	Podsumowanie	144
8.1	Nakłady pracy	144
8.1.1	Dokumentacja	147
8.1.2	Część sprzętowa	148
8.1.3	Oprogramowanie	148
8.1.4	Prototypowanie	149
8.1.5	Testowanie	149
8.2	Potencjał komercjalizacyjny	150
8.2.1	Analiza Value Proposition Canvas	150
8.2.2	Analiza TAM SAM SOM	151
8.2.3	Kanały dotarcia	152
8.2.4	Porównanie na tle konkurencji	153
8.2.5	Szanse i zagrożenia wdrożenia produktu	155
8.2.6	Specyfika rynku	155
8.2.7	Strategia cenowa	157

Rozdział 1

Wstęp

Wstępnie zarysowany zakres przeprowadzonych prac nad realizacją projektu.

1.1 Kontekst projektu

Realizowany projekt posiada charakter interdyscyplinarny, łącząc dziedzinę nauk informatyki z dziedziną nauk elektroniki, skupiając się jednak przede wszystkim na aspektach dziedzinowych informatyki.

W nawiązaniu do obszarów dziedzinowych elektroniki realizowany projekt skupia się na dobraniu kompatybilnych podzespołów elektronicznych i skomunikowaniu ich w celu stworzenia jednolitego systemu gotowego do programowej konfiguracji i wdrożenia na nich oprogramowania.

W kontekście informatyki wykorzystano wiedzę dziedzinową w zakresie systemów operacyjnych, programowania, sieci komputerowych, projektowania i metod dokumentowania wdrażanych rozwiązań.

1.2 Opis problemu

Charakterystyka i motywacje analizowanego problemu.

1.2.1 Społeczne aspekty realizacji projektu

Uczestnicy ruchu drogowego, niejednokrotnie są narażeni na wypadki drogowe, u których podłożem często jest nieuwaga kierującego lub idąc tym tropem brak jasno sprecyzowanych, bezpośrednich danych odnośnie aktualnie panujących uwarunkowań drogowych, a więc informacji o charakterze krytycznym, na podstawie których podejmowane są decyzje kierujących.

Efektem niedostatecznej informacji w najgorszym przypadku są kolizje drogowe, które w przypadku użytkowników pojazdów jednośladowych często oznacza głęboką niepełnosprawność lub śmierć. Objęci tym zagrożeniem są również inni uczestnicy ruchu drogowego¹.

Często pojawienie się informacji ostrzegawczych, wymagających zwiększenia uwagi kierującego lub skupienia się na powiązanym z ostrzegawczym komunikatem aspekiem, dotyczącym uczestnictwa w ruchu drogowym może być czynnikiem zmniejszającym ryzyko wystąpienia kolizji, poprzez zwiększenie świadomości o warunkach panujących na drodze, a więc trafniejsze podjęcie decyzji czy wręcz pohamowanie skłonności do brawurowego zachowania u niektórych

¹KRBRD, <http://www.krbrd.gov.pl/pl/tag/statystyki-wypadkow-drogowych> [dostęp: 04.09.2021]

kierowców poprzez odpowiednie ostrzeżenia i uruchomienie myślenia o konsekwencjach podejmowanych działań i wyobraźni odnośnie skutków tych działań. Właściwe podejście do rzetelnie otrzymywanych informacji o warunkach prowadzenia pojazdu buduje odpowiedzialne i bezpieczne zachowania kierowców, zmniejszając tym samym ryzyko wystąpienia kolizji drogowych².

Kolejnym aspektem jest poprawa komfortu prowadzenia pojazdu i zwiększone poczucie bezpieczeństwa prowadzącego, poprzez większą świadomość odnośnie panujących na drodze warunków. Zmniejsza się tym samym czynnik wywołujący stres w postaci niedostatecznej informacji odnośnie podejmowanych decyzji dodatkowo pod presją czasu i odpowiedzialności³. Nadmiar stresu jest współcześnie jedną z form chorób cywilizacyjnych, zmniejszenie czynników stresogennych tam gdzie jest to w ramach zdrowego rozsądku możliwe jest zjawiskiem pożądany⁴.

Pożądane, jest także aby osoby z niewielkim doświadczeniem w prowadzeniu pojazdów jednośladowych, mogły poruszać się po drogach w środowisku zwiększonego dostępu do informacji odnośnie specyfiki aktualnie panujących warunków drogowych i stanu swojego pojazdu. W ten sposób będą mogły podejmować trafniejsze decyzje podczas jazdy oraz zdobywać doświadczenie, w oparciu o ważne z punktu widzenia kierującego informacje, których pozyskanie bez wspomagania współczesną technologią informacyjną byłoby trudne.

Poruszone kwestie, zarysowują horyzont problemu bezpiecznej, odpowiedzialnej i komfortowej jazdy pojazdem jednośladowym, którego rozwiązaniem może być odpowiednio zaprogramowany i zautomatyzowany komputerowy system informacyjny, służący monitorowaniu warunków drogowych i stanu pojazdu, w tym w czasie rzeczywistym, wspierając w ten sposób podejmowanie trafnych decyzji przez kierującego pojazdem jednośladowym w rzeczywistych warunkach drogowych.

Istniejące rozwiązania są z całą skutecznością wdrażane w branży motoryzacyjnej, ale dotyczą w znacznej mierze samochodów osobowych oraz ciężarowych i autokarów⁵. Sytuacja pojazdów jednośladowych, w tym porównaniu wydaje się zaniedbana. Wyjątek mogą stanowić nowe i drogie motocykle, które często pozostają poza zasięgiem kierowców o przeciętnym systemie wynagrodzeń, w szczególności dotyczy to osób młodych, których zarobki nie są wystarczające aby zdobyć się na taki zakup⁶. W dodatku szeroką grupą użytkowników zainteresowanych przemieszczaniem się przy pomocy motocykli są właśnie osoby młode⁷. W konsekwencji w typowym ruchu ulicznym możemy spotkać kierowców motocykli używanych, a wiec ze starszą technologią i budżetowymi, których systemy wspomagające prowadzenie pojazdu mogą być ograniczone do minimum⁸.

W nawiązaniu do istniejącego zapotrzebowania na systemy komputerowe wspomagające prowadzenie pojazdów jednośladowych, okazuje się, że nie muszą być one integralną i fabryczną częścią konstrukcyjną pojazdu. Współcześnie możliwa jest, bezwzględnie przy zachowaniu norm

²INFOR, <https://mojafirma.infor.pl/moto/prawo-na-drodze/predkosc-i-fotoradary/319047,Fotoradary-a-bezpieczenstwo-na-drogach.html>[dostęp: 04.09.2021]

³Encyklopedia PWN, <https://encyklopedia.pwn.pl/haslo/stres;3980346.html> [dostęp: 0.09.2021]

⁴Medycyna Praktyczna, <https://www.mp.pl/pacjent/neurologia/aktualnosci/197635,stres-pogarsza-pamiec-i-zmniejsza-mozg> [dostęp: 04.09.2021]

⁵poradnik Biznesu, <http://www.poradnikbiznesu.info/samochod-w-firmie/innowacje-w-motoryzacji> [dostęp: 04.09.2021]

⁶Wynagrodzenia, <https://wynagrodzenia.pl/artykul/podsumowanie-ogolnopolskiego-badania-wynagrodzen-w-2018-roku> [dostęp: 04.09.2021]

⁷CEPIK, <http://www.cezik.gov.pl/statystyki> [dostęp: 04.09.2021]

⁸CEPIK, <http://www.cezik.gov.pl/statystyki> [dostęp: 04.09.2021]

dotyczących bezpieczeństwa, instalacja zewnętrznego systemu monitorującego stan pojazdu i analizującego warunki drogowe, przetwarzającego, prezentującego te dane kierującemu lub automatyzującego inne procesy według zadanych wymagań.

Okazuje się, że komputerowy system wspomagania jazdy motocyklem, nie musi być drogi w projektowaniu, procesie wytwórczym i wdrożeniu, ze względu na relatywnie niskie ceny podzespołów elektronicznych oraz ich modułowość przyspieszającą realizację projektu i ułatwiającą komunikację pomiędzy modułami, które są ze sobą kompatybilne⁹. Przykładem takiego rozwiązania jest wykorzystanie minikomputera „Raspberry Pi” i zestawu sensorów o zgodnym interfejsie komunikacyjnym.

Uniwersalnością użytego rozwiązania jest wykorzystanie urządzenia telekomunikacyjnego z systemem Android jako elementu zapewniającego warstwę prezentacji widoków, warstwę telekomunikacyjną i warstwę rejestracji sensorycznej. Przy takim założeniu możliwe jest wykorzystanie modułów komunikacji oraz czujników, w które wyposażony jest telefon komórkowy lub tablet. Zwiększa się w ten sposób funkcjonalność systemu oraz upraszcza część projektową i techniczną realizację projektu, zmniejsza się koszt wytworzenia i wdrożenia oraz poprawia się dostępność aplikacji.

Ograniczeniem uniwersalności funkcji realizowanych przez komputerowy system wspomagania jazdy pojazdem jednośladowym jest konieczność indywidualnego montażu podzespołów systemu w konkretnym pojeździe. Fakt ten zmniejsza jego dostępność dla przeciętnego użytkownika, wymagając przynajmniej elementarnej wiedzy technicznej, w celu przeprowadzenia instalacji.

Produkt jest adresowany dla każdego kierowcy pojazdów jednośladowych, który chciałby prowadzić pojazd ze zwiększoną świadomością odnośnie panujących na drodze warunków oraz stanu pojazdu, w celu poprawy bezpieczeństwa warunków panujących na drodze oraz zwiększenia komfortu prowadzonego pojazdu.

1.2.2 Uzasadnienie zapotrzebowania realizacji projektu

W dzisiejszej dobie, przy istniejącej, dotąd niespotykanej w historii ludzkości możliwości prze mieszczania się osób, przy wykorzystaniu współczesnych środków transportu, w szczególności pojazdów silnikowych, można zaobserwować pozytywne i negatywne skutki jakie przyniósł nam rozwój technologii w branży transportu i logistyki.

Realizowany przeze mnie projekt jest motywowany próbą złagodzenia negatywnych skutków zjawisk występujących w ruchu ulicznym, w szczególności zagrożeniom bezpieczeństwa osób będących uczestnikami tego ruchu, aspirując jednocześnie do zwiększenia udziału aspektów pozytywnych nad negatywnymi, które można w specyfice ruchu drogowego wspólnie zaobserwować.

Temat bezpieczeństwa ruchu drogowego jest szeroką dziedziną. W realizowanych projekcie skupiono się na jednym z aspektów tej tematyki, wychodząc na przeciw potrzebie poprawy bezpieczeństwa ruchu drogowego, które statystycznie na przestrzeni lat ulega stopniowej poprawie, dzięki większej świadomości uczestników ruchu drogowego, rozwiniętej infrastrukturze drogowej i lepszym środkom transportu, biorąc pod uwagę większy ilościowy udział ogólnej liczby pojazdów poruszających się po drogach¹⁰, jednakże istnieją nisze, które w większym stopniu

⁹Elecena, ceny elementów elektronicznych, <https://elecena.pl/report> [dostęp 04.09.2021].

¹⁰PZPM, Polski Związek Przemysłu Motoryzacyjnego, <https://www.pzpm.org.pl/Rynek-motoryzacyjny/Park-pojazdow-zarejestrowanych/Tabele-Park-pojazdow-zarejestrowanych-w-Polsce-1990-2018>

wymagają zainteresowania aspektem bezpieczeństwa ruchu pojazdów i komfortu jazdy.

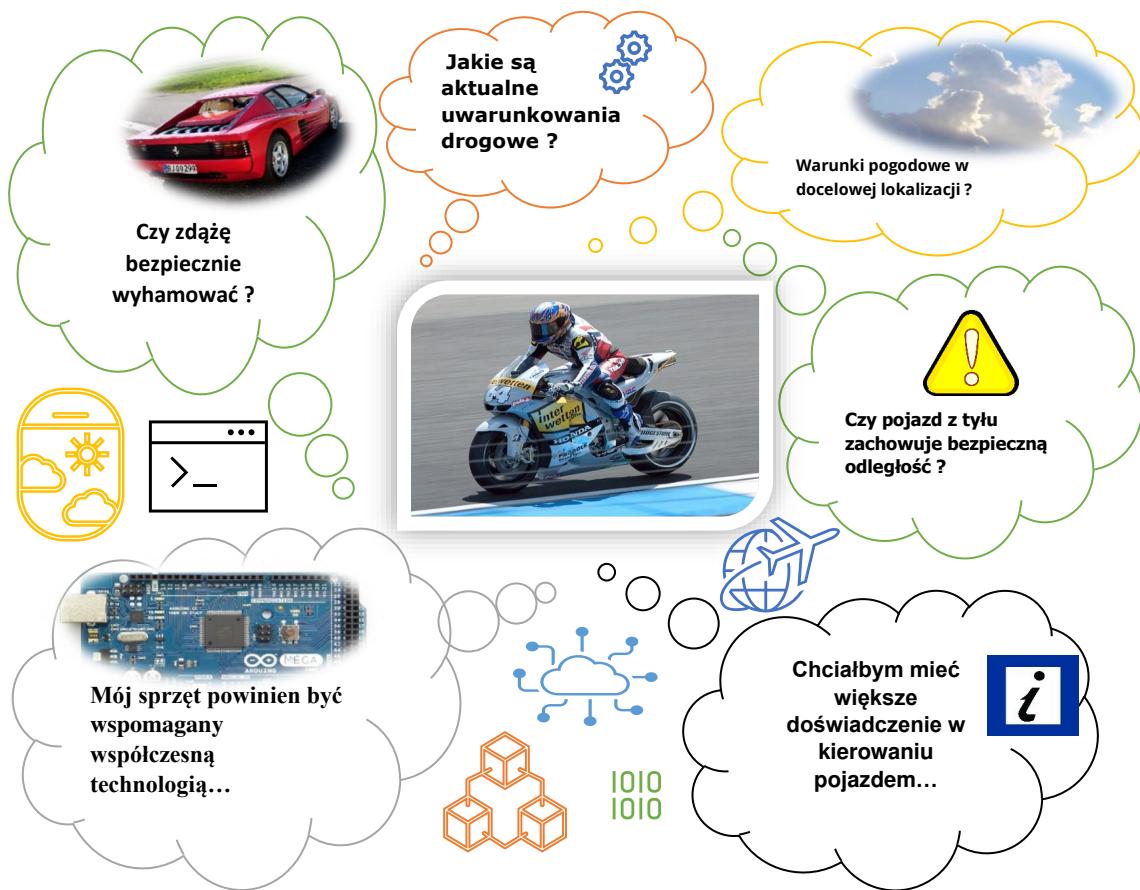
Opisywaną grupą są kierowcy pojazdów motocyklowych¹¹, którym realizowany projekt jest dedykowany, starając się wyjść na przeciw zarysowanym problemom.

Wzbogacony rysunek, obrazuje w reprezentacji graficznej funkcjonalność systemu określonego w kontekście użytkownika docelowego i jego oczekiwaniami związanymi z użytkowaniem systemu.

1.2.3 Motywacja celów projektu

Pomysł na projekt powstał w wyniku zainteresowania prowadzeniem pojazdów motocyklowych, szczególnie tych modeli, które pozbawione są nowoczesnych, bardziej zaawansowanych rozwiązań technologicznych montowanych fabrycznie. Drugim motywem powstania projektu było zainteresowanie informatyką, szczególnie dziedzinami inżynierii oprogramowania oraz działami związanymi z elektroniką. Efektem opisanych zainteresowań był pomysł na połączenie nowoczesnych rozwiązań elektronicznych i oprogramowania z pojazdem motocyklowym w celu poprawy szeroko rozumianej ergonomii jazdy pojazdem. Zawężając tematykę ergonomii jazdy motocyklem, zdecydowano się na wspomaganie jazdy kierowcy motocykla, wykorzystując nowoczesne technologie informatyczne. Priorytetową przesłanką dla wspomagania jazdy kierującego motocyklem wydał się aspekt poprawy bezpieczeństwa ruchu drogowego. W wyniku przedstawionej analizy powstał pomysł na system IT poprawiający ergonię i zwiększający bezpieczeństwo kierowcy pojazdu motocyklowego oraz pośrednio innych uczestników ruchu drogowego.

¹¹Policja, <https://policja.pl/pol/aktualnosci/190688,Bezpieczenstwo-motocyklistow-na-drodze.html> [dostęp: 23.05.2021].



Rysunek 1.1: Rich Picture, [źródło: własne]

1.3 Zakres pracy

Zakres prac nad realizacją projektu obejmuje utworzenie prototypu systemu IT, składającego się z:

1. Minikomputera Raspberry Pi
2. Elektronicznego czujnika odległości
3. Smartfona z dostępem do internetu
4. Oprogramowania użytych podzespołów elektronicznych
5. Aplikacji mobilnej

W zakresie prac projektu jest uwzględnione utworzenie dokumentacji.

1.4 Realizacja pracy

W celu realizacji celów projektowych, zastosowano metodykę iteracyjno-przyrostową. Pierwsza faza realizacji projektu miała charakter projektowy. W początkowym podejściu, w celu zarysowania planistycznych ram realizacji projektu utworzono sekcje: Wstęp, Dokument Założeń Wstępnych, Wstępny Plan Projektu;. W kolejnych podejściach realizacja projektu miała charakter wdrożeniowy. Na tym etapie powstawały moduły prototypu systemu zgodnie z założoną metodą i z uwzględnieniem założeń projektowych.

Podczas realizacji projektu wykorzystano następujące języki programowania:

1. Język C++
2. Język Python
3. Język Java
4. Język skryptowy BASH

Podczas realizacji projektu wykorzystano następujące narzędzia:

1. Android Studio
2. Edytor programistyczny MS Code
3. Plugins do MS Code
4. REST Client w przeglądarce Firefox - Open RESTED Tab
5. Program Fritzing
6. Program FlatCAM
7. Program Candle
8. Kompilator g++
9. Interpreter języka python
10. Pakiet SDK języka Java
11. Terminal systemu Linux
12. Procesor tekstu Word
13. Edytor graficzny Paint
14. Program PowerPoint
15. Program do składania tekstu Latex

Podczas realizacji projektu wykorzystano następujące systemy operacyjne:

1. Raspbian
2. Windows 10
3. Windows 11 preview

Podczas realizacji projektu wykorzystano następujące podzespoły elektroniczne:

1. Laptop
2. Raspberry Pi
3. Czujnik odległości
4. Skaner obiektów w przestrzeni dwuwymiarowej
5. Wideo rejestrator
6. Płytnica prototypowa
7. Adapter USB - UART
8. Przewody połączeniowe
9. Przewody zasilające
10. Bateria zasilająca
11. Maszyna do graverowania CNC
12. Laminat PCB
13. Kondensatory elektrolityczne

Podczas realizacji projektu dla celów utworzenia dokumentacji wykorzystano:

1. Cyfrowy aparat fotograficzny smartfona
2. Zrzuty ekranu z monitorów

1.5 Rezultaty

W wyniku realizacji projektu, poprzez utworzony prototyp systemu IT, osiągnięto rezultaty o oddziaływaniu:

1. Praktycznym
 - (a) Wspomaganie kierowców podczas jazdy motocyklem
 - (b) Zwiększenie komfortu jazdy motocyklem
 - (c) Zwiększenie świadomości odnośnie specyficznych uwarunkowań ruchu drogowego
2. Teoretycznym
 - (a) Zmniejszenie potencjalnej liczby wypadków drogowych z udziałem motocyklistów
 - (b) Poprawa wizerunku motocyklistów, jako kierowców zabiegających o bezpieczeństwo na drodze
 - (c) Zmniejszenie liczby ryzykowanych zachowań motocyklistów
3. Rozwojowym
 - (a) Możliwość dalszej rozbudowy projektu
 - (b) Możliwość komercjalizacji produktu
 - (c) Udostępnienie produktu społeczności skupionej wokół idei wolnego oprogramowania w celu dalszego rozwoju i popularyzacji projektu

1.6 Podziękowania

Jako student i dyplomant chciałbym wyrazić swoją wdzięczność Kadrze dydaktycznej Polsko-Japońskiej Akademii Technik Komputerowych w Gdańsku za okazane wsparcie i pomoc merytoryczną w przygotowaniu niniejszej pracy dyplomowej.

Rozdział 2

Dokument Założeń Wstępnych

Wstępne założenia projektowe.

2.1 Zakres etapu projektowania

Zakresem realizowanego projektu jest podjęcie działań, w oparciu o ustalone we Wstępie dokumentu:

1. kontekst projektu
2. społeczne aspekty rozwiązania
3. uzasadnienie realizacji projektu
4. zakres realizowanych prac
5. sposób realizacji prac
6. spodziewane rezultaty realizacji projektu

oraz ustalone w dalszych częściach dokumentu:

1. specyfikacje udziałowców
2. oczekiwane produkty realizacji projektu

Zakres projektu obejmuje:

1. Projektowanie realizacji rozwiązania w Dokumencie Założeń Wstępnych:
 - (a) Realizację etapów cyklu życia projektu od rozpoczęcia do zakończenia prac projektowych.
 - (b) Kontekst systemu, określa system, jako produkt realizacji projektu oraz systemy zewnętrzne z nim powiązane w tym charakter osadzenia systemu-produktu w określonym środowisku działania - kontekście. Kontekst systemu specyfikuje ponadto rodzaje użytkowników systemu, ich uprawnienia dostępu i ich liczbę, a także współdzielone zasoby systemu. Opis użytych wzorców i standardów wykorzystanych do wdrożenia i uruchomienia systemu również określony jest przez jego kontekst.
 - (c) Zakres systemu, określa funkcjonalność systemu, czyli dostarczane przez działający system usługi wraz z opisem sposobu ich wykorzystania przez odbiorców lub inne systemy. Zakres systemu wyróżnia również te zachowania i funkcjonalności, których system nie powinien pełnić.
 - (d) Wymagania jakościowe, określa:
 - i. ochrona - sposób użytkowania zapewniający długotrwałe, bezpieczne, niezawodne i wydajne wykorzystanie systemu,
 - ii. bezpieczeństwo - warunki, które system powinien spełniać aby nie narażać otoczenia na zagrożenie stanu bezpieczeństwa,
 - iii. przenośność - zapewnia możliwość zmiany środowiska działania systemu przy zachowaniu pełnionej funkcjonalności i spełnieniu wymagań jakościowych,
 - iv. elastyczność - określa stopień w jakim system zachowuje udostępnione usługi i

- wymagania jakościowe podczas zmieniających się warunkach otoczenia, w których pracuje,
- v. konfigurowalność - określa podatność programową na modyfikację pełnionych usług lub możliwości indywidualizacji zachowania systemu udostępnione użytkownikom,
 - vi. niezawodność, określa odporność systemu na błędy generowane zarówno od wewnętrznych warstw systemu, przez działający system, jak i od strony warstw zewnętrznych, w wyniku interakcji z użytkownikami lub innymi systemami,
 - vii. wydajność, zapewnia działanie systemu w sposób zapewniający akceptowalne z punktu widzenia użytkowników i innych systemów czasy jego odpowiedzi na ich żądania wyników działania udostępnionych usług, nawet podczas zwiększonego stanu kolejkowania tych żądań, powstałego w wyniku dużej z punktu widzenia systemu liczby generowanych żądań przez oczekujących odpowiedzi odbiorców, na skutek przeciążenia systemu.
- (e) Wizja konstrukcyjna, obejmuje planowane użycie konkretnych rozwiązań technologicznych i architektonicznych służących realizacji celów projektowych.
- (f) Ograniczenia, mają wpływ na kształt projektowanego systemu. Uniemożliwiają spełnienie planowanych założeń projektowych lub wymagają zastosowania rozwiązań alternatywnych w celu ich realizacji. Czynnikiem ograniczającym mogą być: Ograniczenia produktowe:
- i. ograniczenia prezentacji danych przez możliwości interfejsu,
 - ii. mało funkcjonalny dostępowy interfejs do oferowanych usług,
 - iii. wymagające warunki środowiska działania systemu,
- Ograniczenia projektowe:
- i. zbyt mała ilość czasu na realizację przyjętych założeń projektowych,
 - ii. zbyt duże wymagania odnośnie kompetencji lub możliwości personalnych i interpersonalnych udziałowców projektu,
 - iii. brak wymaganych do realizacji projektu rozwiązań sprzętowych,
 - iv. brak odpowiedniego oprogramowania,
 - v. niewystarczające środki finansowe w stosunku do założonych wymagań.

2. Projekt realizacji rozwiązania we Wstępny Planie Projektu:

- (a) Założenia projektu:
- i. Cel powstania projektu oraz pomysłu na rozwiązanie przedstawionego problemu
 - ii. Powstałe w wyniku zaplanowanej realizacji projektu spodziewane produkty
 - iii. Udziałowców projektu - zleceniodawców projektu i jego wykonawców oraz bezpośrednich i pośrednich użytkowników gotowego systemu
 - iv. Specyfika wybranej strategii realizacji projektu i jej pozytywne i negatywne uwarunkowania mające wpływ na realizację projektu, a także uzasadnienie wyboru danej strategii realizacji projektu
 - v. Hierarchia założonych celów projektowych pod względem ich istotności z punktu widzenia realizacji projektu
 - vi. Określenie głównych zagrożeń skuteczności realizacji projektu i rozmiaru ich oddziaływanego oraz gałęzi dziedzinowej, której dotyczą wraz z przyczyną zaistnienia ryzyka a także oszacowanie prawdopodobieństwa materializacji zagrożeń i stopnia wpływu na realizowany projekt wraz z zaplanowaniem przeciwdziałania zaistniałym zagrożeniom
 - vii. Oszacowany finansowy koszt realizacji projektu
- (b) Wizja rozwiązania:
- i. Planowanie użycia wybranych technologii w celu realizacji projektu oraz kon-

- cepcja ich zestawienia i konfiguracji w celu zarysowania kształtu systemu wraz z wykorzystaniem standardów dla planowanego rozwiązania
- ii. Charakterystyka technologii i środowiska wytwórczego użytych przy wdrażaniu celów projektowych oraz sposobu produkcji
 - iii. Charakterystyka technologii i środowiska produkcyjnego wdrożonego systemu oraz sposobu wdrożenia
- (c) Proces wytwarzania:
- i. Wybór metodyki projektowej w wyniku analizy potrzeb projektowych oraz selekcji porównawczej spośród istniejących alternatywnych strategii prowadzenia prac projektowych z uzasadnieniem dokonanego wyboru metodyki
 - ii. Opis planowanego przebiegu procesu projektowego według wybranej strategii prowadzenia prac wraz z określeniem kolejnych etapów procesu z określeniem zamierzonych do osiągnięcia celów oraz zadań do wykonania aby je zrealizować z uwzględnieniem powiązań zarysowanych działań. Efektem planowania są ścisłe określone, mierzalne produkty
 - iii. Wyszczególnienie kolejnych etapów realizacji powinno uwzględnić dodatkowo kryteria akceptacji dla uzyskanego efektu realizacji etapu oraz oszacowanie czynników ryzyka powodzenia realizacji etapu, ponadto powinny zostać określone zasady zarządzania i sposoby postępowania w celu zapewnienia określonej jakości dla produktów etapu
 - iv. Przeprowadzenie testów dla uzyskanych w wyniku realizacji etapu produktów pozwoli zlokalizować w celu eliminacji czynniki zagrażające spełnieniu wymaganiom jakościowym produktów
- (d) Infrastruktura:
- i. Wyszczególnienie adekwatnych do przyjętych założeń projektowych i zaplanowanego procesu wytwórczego zasobów, którymi dysponujemy lub wymaganymi do pozyskania jako nieodzownymi dla postawionych wymagań projektowych. Wyszczególnione zasoby charakteryzują:
 - A. wymiar nakładu poniesionej przez człowieka pracy
 - B. środki i narzędzia w postaci wymaganego sprzętu
 - C. środki i narzędzia w postaci wymaganego oprogramowania
 - D. wymagana infrastruktura dla użytych środków i produktów
 - E. posiadana wiedza i umiejętności
 - F. dostępność zewnętrznych usług
 - G. koszty finansowe
 - H. określony czas
 - ii. Wyszczególnienie infrastruktury technicznej
 - iii. Sposób prowadzenia dokumentacji
3. Projekt realizacji rozwiązania w Specyfikacji Wymagań Systemowych:
- (a) Projekt w kontekście
- i. Udziałowcy - rozumiani jako podmioty ożywione - osoby, społeczeństwo lub nieożywione - systemy, urządzenia, regulacje prawne, mające wpływ na projekt lub będące pod wpływem rozwiązań projektowych, w tym sensie użycia infrastruktura techniczna jest udziałowcem nieożywionym, ponieważ narzuca swoiste wymagania rozwiązaniom projektowym.
 - ii. Klienci zewnętrzni - stanowią grupę zleceniodawców, podwykonawców i dostawców.
 - iii. Charakterystyka użytkowników - określa prawa dostępu do poszczególnych fragmentów systemu

- (b) Wymagania
- i. Wymagania ogólne i dziedzinowe - wyszczególniają przyjęte zasadnicze założenia projektowe w zakresie ogólnym, biznesowym i dziedzinowym wraz z określonym kontekstem.
 - ii. Wymagania funkcjonalne:
 - A. Nazwa funkcji - wyszczególniają funkcjonalność i usługi systemu.
 - B. Interfejs z otoczeniem - wyszczególniają wymagania korelacji interfejsów istniejącego systemu z interfejsem usług dostarczanych przez zewnętrzne systemy współpracujące i powiązane funkcjonalnie z systemem projektowanym.
 - iii. Wymagania niefunkcjonalne - wyszczególniają skonkretyzowany lub mierzalny sposób działania systemu, opisując wymagania jakościowe lub kategorie ergonomiczne: łatwość użycia, intuicyjność, adaptowalność.
 - iv. Wymagania na środowisko docelowe - wyszczególniają docelowe środowisko pracy systemu.
 - v. Wymagania dotyczące procesu wytwarzania - wyszczególniają sposób przyjętej realizacji projektu, wybraną metodykę, planowany czas i plan prac.
- (c) Kryteria akceptacji rozwiązania - specyfikują stopień i kryteria akceptacji gotowego produktu przez klienta na podstawie założonych wymagań funkcjonalnych i niefunkcjonalnych, kategoryzowanych zgodnie ze zdefiniowanymi priorytetami, w celu porównania zgodności określonych tutaj priorytetów z celami projektowymi i ustalenia stopnia pokrycia i zgodności między nimi
- 4. Projekt systemu
 - 5. Wdrożenie rozwiązań projektowych
 - 6. Testowanie rozwiązań
 - 7. Dokumentowanie procesu realizacji projektu
 - 8. Prezentacja efektów realizacji projektu

2.2 Cele projektu

Celem projektu jest wytworzenie systemu IT, przy wykorzystaniu założonej w projekcie metodyki ewolucyjno-przyrostowej, realizując metodycznie cele określone w zakresie projektu.

2.3 Zakres systemu

Funkcjonalność systemu zapewnia usługi:

- 1. Prezentacji graficznych animacji na ekranie smartfona.
- 2. Połączenia z internetem mobilnym.
- 3. Stale włączonego ekran smartfona.
- 4. Przetwarzania danych numerycznych otrzymanych z sensorów na grafikę komputerową.
- 5. Stałej łączności pomiędzy podsystemami.
- 6. Zapisu rejestrowanych danych.
- 7. Odczytu danych z Rest API.
- 8. Przetwarzania sygnałów w sensorów na dane numeryczne.
- 9. Uruchomienia skryptów przy starcie systemu.
- 10. Utworzenia sieci lokalnej.
- 11. Wysyłania danych przez Rest API.

Następujące zdarzenia nie powinny wystąpić w systemie:

1. Brak rozruchu startowego (bootowania).
2. Nieładujący się system operacyjny.
3. Brak połączenia z internetem.
4. Brak załadowania programów startowych.
5. Błędy podczas działania systemu operacyjnego.
6. Błędy generowane przez oprogramowanie wdrażanego systemu.
7. Brak komunikacji z oprzyrządowaniem elektronicznym.
8. Niestabilność działającego systemu operacyjnego.
9. Brak wydajności systemu.

2.4 Wymagania jakościowe i inne

Produkt powinien spełniać wymagania:

1. Ochrony: Elementy montażowe produktu powinny być wykonane z odpornego materiału. Instalacja zasilania powinna być rozprowadzona w sposób zabezpieczający przed uszkodzeniem mechanicznym. Smartfon powinien być ochroniony przed wpływem warunków atmosferycznych. Elementy elektroniczne powinny być zabezpieczone przed wpływem warunków atmosferycznych.
2. Bezpieczeństwa: Instalacja produktu na pojeździe powinna być stabilna. Instalacja zasilania powinna być zabezpieczona przed zwarciem. System nie powinien generować błędnych odczytów.
3. Przenośności: Aplikacja powinna być dostępna dla smartfonów działających w systemie Android. Produkt powinien dać się zamontować na wszystkich rodzajach motocykli.
4. Elastyczności: Urządzenia elektroniczne powinny pracować prawidłowo w zmiennych warunkach atmosferycznych. System powinien być użyteczny w zmiennych warunkach oświetlenia za dnia i w nocy.
5. Konfigurowalności: System nie powinien podlegać modyfikacjom ze strony użytkownika. Modyfikacje dopuszcza się jedynie w wizualnej warstwie prezentacji widoków.
6. Niezawodności: System nie powinien generować błędów. System powinien być dostępny bez przerw podczas użytkowania.
7. Wydajności: Aplikacja mobilna nie powinna mieć opóźnień w czasie prezentacji treści. Cześć sprzętowa powinna zapewniać przesył danych bez opóźnień, na żądanie. Komunikacja bezprzewodowa pomiędzy podsystemami nie powinna mieć opóźnień.
8. Inne: W rozwiązaniu problemu projektowego powinno się wykorzystywać standaryzowanych protokołów, wzorców projektowych oraz zalecanych przez społeczność praktyk. Systemy wydanego do użytku urządzenia nie powinny być aktualizowane.

2.5 Wizja konstrukcyjna

1. Instalacja systemu Raspbian na karcie pamięci SD
2. Instalacja pamięci SD w minikomputerze Raspberry Pi.
3. instalacja sterownika adaptera USB UART w systemie Windows 10.
4. Komunikacja z interfejsem szeregowym Raspberry Pi wykorzystując adapter USB UART.
5. Nawiązanie połączenia z Raspberry Pi wykorzystując program PuTTY.
6. Nawiazanie połączenia z internetem w systemie Raspbian.
7. Konfiguracja wstępnych ustawień systemu Raspbian.
8. Podłączenie urządzeń peryferyjnych do minikomputera.
9. Podłączenie sensorów do minikomputerów.
10. Instalacja wymaganych sterowników.
11. Instalacja wymaganych bibliotek.
12. Uruchomienie oprogramowania sterującego pracą sensorów.
13. uruchomienie oprogramowania udostępniającego dane odczytane z sensorów.
14. Automatyzacja procesu uruchamiania oprogramowania przy starcie systemu.
15. Utworzenie systemowej sieci lokalnej.
16. Testowanie odczytu danych.
17. Utworzenie aplikacji mobilnej.
18. Udostępnienie usługi hotspot ze smartfona.
19. Połączenie z siecią internet poprzez operatora GSM.
20. Utworzenie serwisu odczytującego dane z sensorów.
21. Utworzenie serwisu odczytującego dane pogodowe.
22. Utworzenie serwisu zapamiętującego dane.
23. Utworzenie kontrolerów przetwarzających odczytane dane.
24. Utworzenie widoków prezentujących graficznie przetworzone dane.
25. Udostępnienie konfiguracji wyświetlania warstwy wizualnej użytkownikowi.
26. Testowanie wdrażanych rozwiązań.
27. Opracowanie schematu połączeń płytka PCB.
28. Wdrożenie elektronicznych prototypów na płycie PCB.
29. Zaprojektowanie modelu obudów i mocowania podzespołów.
30. Utworzenie obudów dla użytych podsystemów.
31. Utworzenie mocowania dla wytworzonych produktów.
32. Opracowanie schematu instalacji elektrycznej dla produktu.

2.6 Ograniczenia

2.6.1 Produktowe

Prezentacja danych w formie graficznej wymaga użycia rozwiązań, które narzuca IDE Android Studio.

Komunikacja pomiędzy podzespołami produktu jest ograniczona standardami protokołu HTTP. Widoczność animacji warstwy prezentacji jest uzależniona od wielkości ekranu smartfona.

Implementowane funkcjonalności są ograniczone dostępnym API bibliotek.

Dostęp do zewnętrznych danych jest ograniczony dostępnym w zewnętrznych serwisach Rest API.

Odporność na warunki atmosferyczne produktów projektu jest ograniczona brzegowymi wartościami określonymi w notach katalogowych jego podzespołów.

Działanie systemu jest uzależnione od dostępu do sieci internet.

2.6.2 Projektowe

Realizacja zakresu projektu jest ograniczona czasem.

Rozdział 3

Wstępny Plan Projektu

3.1 Założenia projektu

Klientami zrealizowanego projektu są kierowcy pojazdów jednośladowych oraz każda osoba zainteresowana poprawą bezpieczeństwa poruszania się pojazdów na drodze.

Realizacja projektu umożliwia zaplanowaną konstrukcję, przetestowanie i prezentację komputerowego systemu do motocykla, zautomatyzowanego w celu pozyskania informacji o specyficznych uwarunkowaniach ruchu drogowego.

Projekt realizowany będzie przy użyciu metodologii ewolucyjno-przyrostowej.

Priorytetem projektu jest ukończenie jego realizacji w wyznaczonych terminie. Kolejnym priorytetem jest spełnienie założonych wymagań projektowych.

Projekt realizowany jest samodzielnie.

3.1.1 Cele projektu

Realizowany projekt obejmuje następujące etapy twórcze, wykorzystując metodologię ewolucyjno-przyrostową:

1. Utworzenie wstępnej dokumentacji projektu.
2. Utworzenie dokumentacji przedstawiającej problem.
3. Utworzenie dokumentacji rozwiązania.
4. Realizację założeń projektowych w celu przyrostowego wytworzenia produktu.
5. Dodawanie przyrostów dokumentacji na podstawie postępów realizacji założeń projektowych wytwarzanego produktu.
6. Prezentację gotowego systemu.

3.1.2 Spodziewane produkty

Specyfikacja powstały produktów projektu jest uzależnione od przebiegu aktualnie realizowanego przyrostu. W wyniku planowej realizacji projektu spodziewane są następujące produkty:

1. System wbudowany dla przedniej części pojazdu.
2. Aplikacja mobilna systemu Android.
3. Prototyp.
4. Zasilanie systemów.
5. Dokumentacja projektu.

3.1.3 Udziałowcy

1. Polsko-Japońska Akademia Technik Komputerowych
2. Promotor
3. Recenzent
4. Konsultanci
5. Dyplomant
6. Przedstawiciele branży IT
7. Kierowcy motocykli
8. Pozostali uczestnicy ruchu drogowego
9. Komercyjni klienci produktu
10. Pozostali zainteresowani

3.1.4 Uwarunkowania, ograniczenia, założenia strategii

Motywacją wyboru metodologii ewolucyjno-przyrostowej była próba zachowania elastyczności przy realizacji prac projektowych oraz wykorzystanie przewag, które dana metodologia miała nad pozostałymi w danym zastosowaniu.

Przy wykorzystaniu opisanego podejścia opracowano pierwszą wersję dokumentacji w Dokumencie Założeń Wstępnych i Wstępny Planie Projektu oraz konstrukcję pierwszych wersji produktów - wstępnie skonfigurowanego minikomputera Raspberry Pi oraz wstępna wersję aplikacji mobilnej.

W dalszym etapie podjęto próbę przyrostowej realizacji wybranych etapów projektu według opisanego w Dokumencie Założeń Wstępnych zakresie. Próba wdrożenia wszystkich opisanych w zakresie wymagań zakończyła się niepowodzeniem, ponieważ w wyniku podjętej analizy ilości zasobów czasu potrzebnego w realizację wszystkich założeń w konfrontacji z rzeczywistym tematem prowadzenia prac i określonym terminem realizacji, wykazano brak wystarczającej ilości czasu.

Podczas etapu projektowania rozważano zmianę wybranej metodyki w celu lepszego dopasowania do charakteru planowanych prac. Rozważano metodologie: kaskadową, model V i hybrydową Model v został analizowany w celu organizacji procesu testowania rozwiązań, jednakże nie został wykorzystany w projekcie, z powodu zbyt dużego rozbudowania procesu testowania oraz trudności w synergii wymagań modelu v z wdrażaną już strategią ewolucyjno-przyrostową.

Ze względu na nieadekwatność planowanych prac nad projektem z alternatywnymi metodologiami oraz zbyt dużą próbę dowolności w kształtowaniu procesu realizacji projektu, a także ograniczenia i zbytnie wymagania narzucone przez te metodyki w stosunku do realnych wymagań dotyczących realizacji projektu, postanowiono nie zmieniać strategii realizacji projektu. Po konsultacji doradczej z udziałowcami projektu, dotyczącej wyboru odpowiedniej metodyki, odpowiednim podejściem okazało się podejście ewolucyjno-przyrostowe.

Na wstępie prowadzonych prac wykorzystano model ewolucyjno-przyrostowy w celu ogólnego zarysowania ram realizowanego projektu. Metodologią opracowano rozdziały: Wstęp, Dokument Założeń Wstępnych i Wstępny Plan Projektu. Zakres prac objął:

1. Uzupełnienie dokumentacji wg zarysowanego uprzednio spisu treści.
2. Uzupełnienie treści brakujących rozdziałów.
3. Przeredagowanie merytorycznie błędnych treści rozdziałów.
4. Zmianę struktury dokumentu.
5. Uzupełnienie istniejącej treści rozdziałów.

Z powodu ograniczeń czasowych zrezygnowano z uszczegóławiania specyfikacji wymagań systemowych na rzecz rozbudowy zakresu projektu systemu opisanego oddzielnym rozdziałem. Rozdziały dokumentacji bliższe wdrożeniu rozwiązania: Projekt Systemu, opis wdrożenia rozwiązania, specyfikacja diagramów, testowanie rozwiązań, prototypowanie, powinny być dekomponowane na mniejsze fragmenty, które zgodnie z przyjętą metodologią powinny być rozwijane przyrostami.

Planowana jest dalsza realizacja projektu wybraną metodą ewolucyjno-przyrostową. Ze względu na możliwość przyrostowego, a nie iteracyjnego podejścia, możliwy jest rozwój wybranych części zakresu projektu, niekoniecznie za jednym rozbudowanym podejściem i niekoniecznie w kaskadowej, sekwencyjnej kolejności. Możliwe są ewolucyjne przyrosty zakresów projektu, w sposób niezależny od stopnia powiązania ze sobą rozwijanych zakresów projektu. Takie rozwiązanie eliminuje wady użycia podejścia iteracyjnego w realizowanym podejściu, czyli konieczność doprowadzenia do rozbudowanej finalnej wersji dużego iterowanego zakresu i jego powiązań.

Wybrana metodologia zapewnia większą elastyczność w wyborze rodzaju, sposobu, ilości i momentu wykonania testów, które w kontekście projektu powinny być realizowane po zbudowaniu prototypu gotowego do testowania, a więc pod koniec trwania czasu realizacji projektu, zapewniając skuteczniejsze testowanie i większy zakres doboru metod testowych.

Termin realizacji projektu ustalono na 3 września 2021 r.

3.1.5 Priorytety

1. Wytworzenie gotowej do prezentacji finalnej wersji modułów sprzętowych.
2. Wytworzenie gotowej do prezentacji finalnej wersji aplikacji mobilnej.
3. Utworzenie kompletnej dokumentacji.
4. Pokrycie rozwiązania testami w 100 % założonym zakresie
5. Zestawienie prototypu.
6. Utworzenie elektrycznej instalacji zasilającej produkty.
7. Realizacja założeń projektu w 100 % założonym zakresie.

3.1.6 Analiza zagrożeń

l.p.	Czynnik ryzyka	Główne zagrożenia i ich kontekst	Prawdopodobieństwo wystąpienia, skutki wystąpienia, wpływ na projekt	Planowane przeciwdziałanie
1	Czas	Przekroczenie terminu realizacji projektu.	Ze względu na powstałą dysproporcję między terminem realizacji projektu a zaplanowanym zakresem projektu, prawdopodobieństwo przekroczenia terminu oszacowano na 50 %.	Przesunięcie terminu realizacji projektu.

3.1.7 Budżet

Budżetem przeznaczonym na realizację projektu jest 1500 zł. Budżet powinien pokryć zapotrzebowanie na potrzebne z punktu widzenia projektu urządzenia¹. Najdroższy moduł sprzętowy stanowi czujnik odległości, który można kupić za 700 zł. (stan na dzień 03.09.2021 r.). Kolejny zakup dotyczy minikomputera. Posiadany budżet jest elastyczny więc nie posiada charakteru wysokiego ryzyka. Planowane koszty realizacji projektu są łatwe do oszacowania i stabilne.

¹Kamami, sklep z elektronika, <https://kamami.pl/czujniki-odleglosci/562733-lidar-lite-v3-laserowy-czujnik-odleglosci-0-40-m-sen-14032.html> [dostęp 03.09.2021]

3.2 Wizja rozwiązania

W celutworzenia systemu zostaną przeprowadzone wielokrotne podejścia strategią ewolucyjno-przyrostową w celu realizacji wymagań zakresu systemu.

Oddany do użytku system stanowią:

1. Aplikacja mobilna.
2. Obudowany system wbudowany dla przedniej części pojazdu.
3. Prototyp gotowego systemu

3.3 Technologia i zamierzone środowisko

Środowisko wytwórcze zapewniające wymaganą infrastrukturę techniczną dla realizacji celów projektu jest realizowane w warunkach domowych.

Środowiskiem docelowym jest pojazd samochodowy z zainstalowanym prototypem produktu. Wdrożony system użytkowany będzie w rzeczywistych warunkach ruchu drogowego. Możliwe jest rozszerzenie zakresu projektu i rozbudowa o nowe funkcjonalności produktu w przyszłości.

3.4 Planowany proces wytwarzania

Metodyczne podejście do procesu wytwarzania.

3.4.1 Wybrany sposób dokumentowania

Produktem początkowej fazy realizacji projektu, była wstępnie określona dokumentacja, składająca się z: Karty Projektu, Dokumentu Założeń Wstępnych, Wstępnego Planu Projektu oraz udokumentowanego przebiegu realizacji początkowego wdrożenia produktu wg przyjętych założeń projektowych. Dokumentację prowadzono z użytkując procesor tekstu MS Word.

Następnie budowa dokumentacji obejmująca: Kartę Projektu, Wstęp, Dokument Założeń Wstępnych, Wstępny Plan Projektu przebiegła z wykorzystaniem wybranej metodologii projektu z jednoczesną aktualizacją treści dokumentów. Opisane dokumenty w wyniku zastosowania strategii kaskadowej w celach edycyjnych są uznane za kompletne i nie wymagają dalszych modyfikacji.

Realizacja dalszych etapów dokumentacji obejmuje: Projekt Systemu, Wdrożenie rozwiązania, Specyfikację diagramów, Testowanie rozwiązań, Harmonogram prac, Podsumowanie, Bibliografię oraz Załączniki. Realizacja sposobu prowadzenia dokumentacji przebiegać będzie wg metodologii ewolucyjno-przyrostowej. Kolejność opracowywanych sekcji dokumentu nie jest w góry ustalona i wynika z podejmowanych na bieżąco decyzji na podstawie aktualnego przebiegu realizacji etapów projektu. Dokumentacja będzie realizowana w przyrostach, umożliwiających wielokrotną aktualizację treści dokumentów, również w tych samych sekcjach.

Wraz z rozwojem dokumentacji jako infrastrukturę dokumentacyjną wykorzystano technologię składu tekstu Latex.

Ukończenie rozdziału z harmonogramem prac systemu oznacza kompletność dokumentacji i ukończenie prac.

W celach utworzenia dokumentacji rozwiązania używano następujących rozwiązań:

1. Technologia składania tekstu Latex
2. Edytor Texstudio
3. Biblioteki dla systemu Latex
4. Procesor tekstu MS Word

3.4.2 Wybór strategii prowadzenia prac

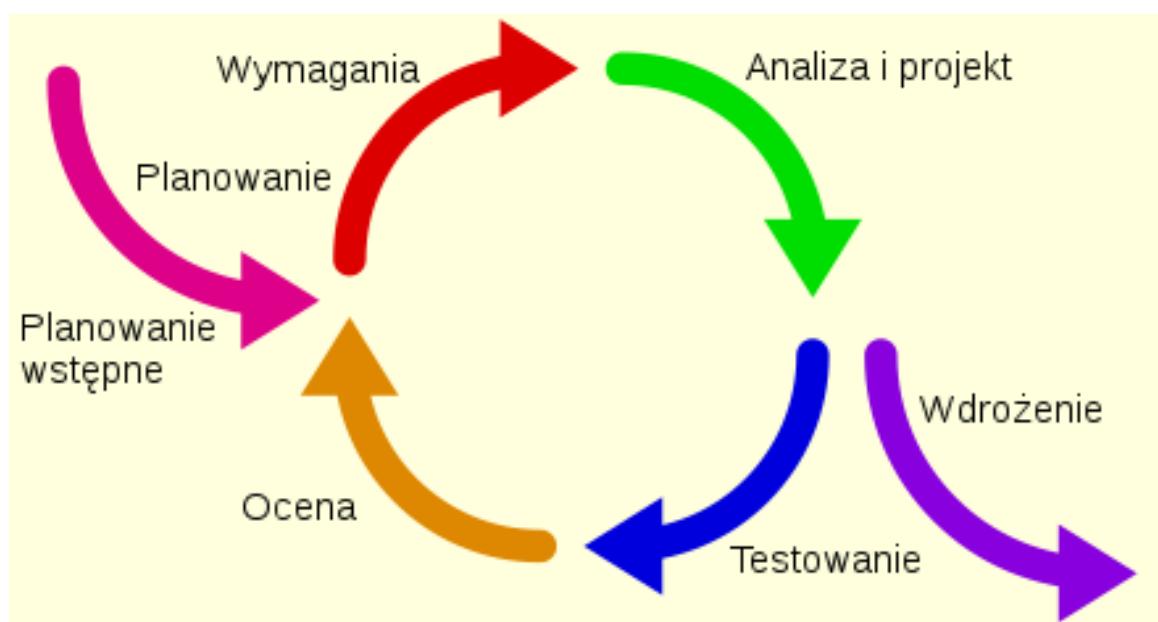
Opis wybranej strategii realizacji prac.

Model ewolucyjno-przyrostowy i iteracyjny

Metodologia ewolucyjno-przyrostowa umożliwia wielokrotne podejście do opracowywanych zagadnień, zapewniając ich przyrost, który nie musi mieć finalnego charakteru².

W przeciwnieństwie do modelu iteracyjnego, który zakłada ukończenie opracowania w kolejnych cyklach, które dotyczą ściśle określonego fragmentu pola działań³.

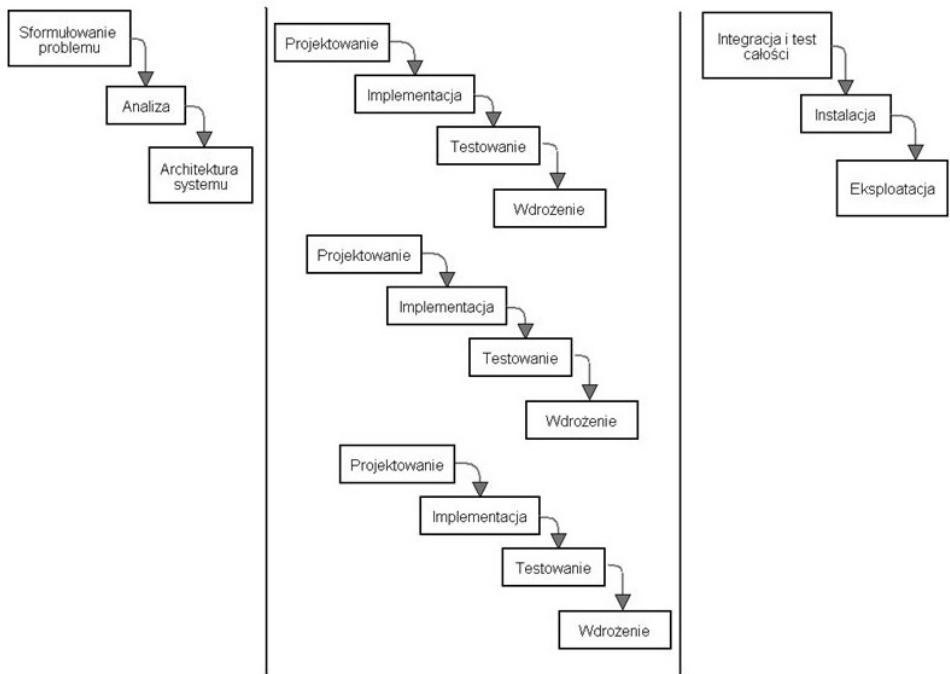
Przedstawiona różnica jest podstawową motywacją wyboru metodologii ewolucyjno-przyrostowej w celu realizacji projektu. Specyfika projektu wymaga większej elastyczności w rozwijaniu cechujących go funkcjonalności, które są zgrupowane w różnych, oddzielnych, ale powiązanych funkcjonalnie podsystemach. Rozwój pojedynczego systemu może generować potrzebę modyfikacji pozostałych oraz rozwój pojedynczego systemu może warunkować rozwój kolejnych. Modyfikacja w tych podsystemach, może poprzez sprzężenie zwrotne wymagać modyfikacji w systemie początkowo rozwijanym. Takie relacje pomiędzy podsystemami w zależności od rozwijanych funkcjonalności pojedynczych podsystemów wymaga metodyki, która nie zakłada utworzenia funkcjonalności od początku do końca, ale umożliwia częściowy jej rozwój i możliwość powrotu do dalszego rozwijania podsystemu w wyniku stopniowego rozwijania pozostałych podsystemów tą samą metodyką. Metodologia ewolucyjno-przyrostowa, zapewnia więc większą swobodę we wprowadzaniu modyfikacji w projekcie, niż model iteracyjny.



Rysunek 3.1: Schemat modelu iteracyjnego, źródło: <https://images.app.goo.gl/oBeJb6auKiyh6QK86> [dostęp 27.05.2021]

²Infona, Portal Komunikacji Naukowej, <https://www.infona.pl/resource/bwmeta1.element.baztech-article-BPC6-0001-0086/tab/summary> [dostęp: 27.05.2021].

³Yadda, Baza danych o zawartości polskich czasopism technicznych, <https://yadda.icm.edu.pl/baztech/element/bwmeta1.element.baztech-0ee28f97-19ac-4960-8d41-224057de95e7>, str. 58-59, 65 [dostęp: 27.05.2021].



Rysunek 3.2: Schemat modelu ewolucyjno-przyrostowego, źródło: <https://slideplayer.pl/slide/404100/> [dostęp 03.09.2021]

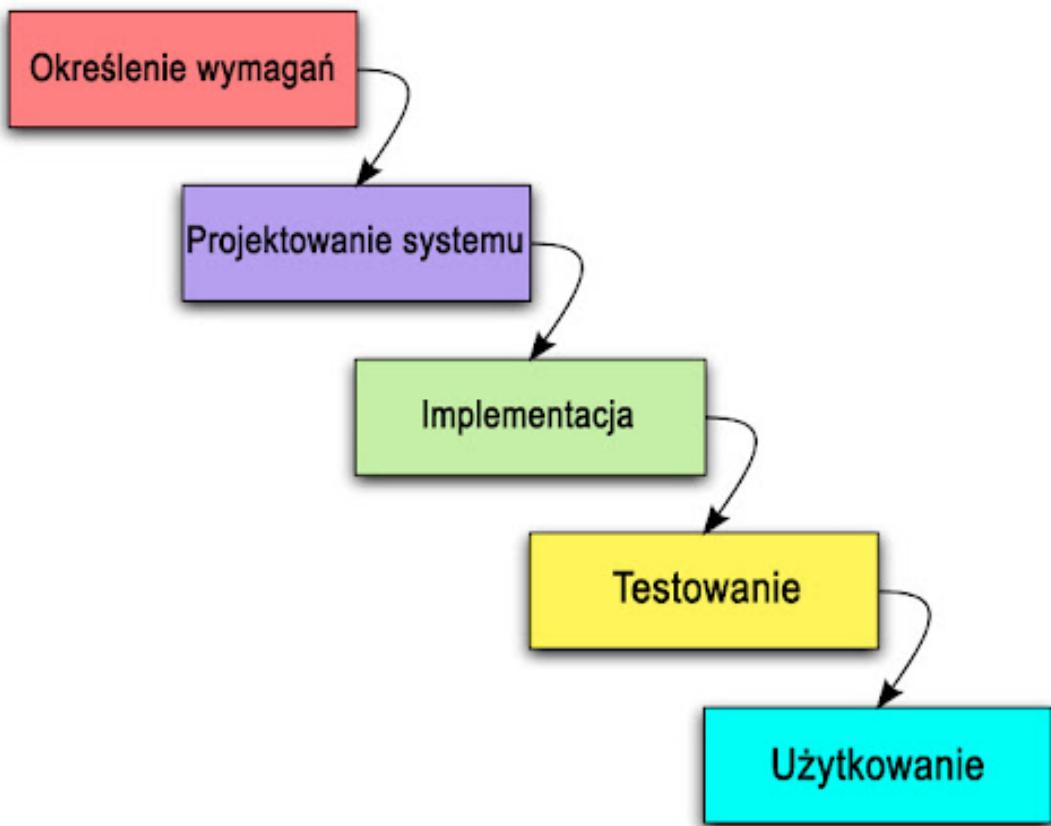
Model ewolucyjno-przyrostowy został wybrany do realizacji projektu ze względu na elastyczność we wprowadzaniu usprawnień, modyfikacji i nowych rozwiązań. Umożliwia realizację projektu w przyrostach, jako małych fragmentach, dzięki czemu łatwiejsza jest dokładniejsza realizacja poszczególnych etapów. Łatwiejsze jest zarządzanie małymi fragmentami przyrostów niż całym dużym projektem. Zmniejsza się koszt realizacji projektu, ze względu na brak konieczności wielokrotnego przechodzenia przez kolejne etapy iteracji. Ze względu na dekompozycję struktury oraz wielokrotność podejścia do opracowania zadania, zwiększa się jakość realizacji poszczególnych etapów, to wpływa na dokładność wykonania systemu komputerowego, a w konsekwencji jego wymagania jakościowe i inne, takie jak: ochrona, bezpieczeństwo, przenośność, elastyczność, konfigurowalność, niezawodność i wydajność.

Model kaskadowy

Model kaskadowy zakłada występujące kolejno po sobie usystematyzowane bloki funkcjonalne, których realizacja następuje w kolejności od pierwszego do ostatniego. Kolejna realizacja każdego bloku zakłada ukończenie bloków poprzednich. Po przejściu serii blokowych kaskad, następuje zakończenie projektu zgodnie z przyjętą metodologią kaskadową.⁴

Model kaskadowy był rozważany jako stosowana metodyka, w celu opracowania wstępnej wersji dokumentacji, skupiając się za aspekcie planowania założeń projektu i systemu. Motywem użycia modelu kaskadowego była sekwencyjna kolejność powiązanych ze sobą następujących kolejno sekcji dokumentu. Ze względu na uporządkowaną, sekwencyjną i powiązaną merytorycznie kolejność rozdziałów wydawał się trafnym podejściem do realizacji początkowej fazy tworzenia dokumentacji. Ograniczeniem tej metodyki był brak ponownego podejścia do rozwiązania problemu w przypadku konieczności rozszerzenia lub analizy już zrealizowanego etapu.

⁴Przegląd modeli cyklu życia oprogramowania, https://www.researchgate.net/profile/Rafal-Kasprzyk/publication/229596276_Przeglad_modeli_cyklu_zycia_oprogramowania/links/5b420974458515f71cb23497/Przeglad-modeli-cyklu-zycia-oprogramowania.pdf, str. 52-53 [dostęp: 27.05.2021].



Rysunek 3.3: Schemat modelu kaskadowego, źródło: https://fulmanski.pl/zajecia/zarzadzanie_firma/zajecia [dostęp 27.05.2021].

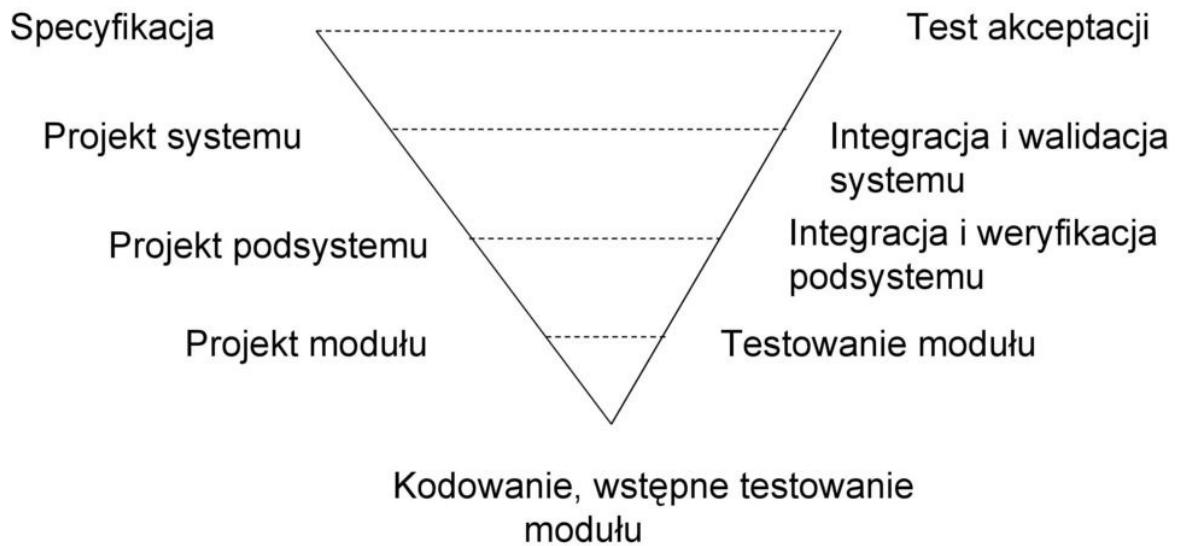
Model V

Model V jest pochodną od modelu kaskadowego i cechuje się dużym pokryciem testami każdego etapu kaskady realizowanego projektu.⁵. Model V został był rozważany w celu realizacji projektu ze względu na etap testowania wypracowanych w projekcie rozwiązań. Charakter realizowanego projektu, narzuca zachowanie jak największych wymagań dotyczących bezpieczeństwa stosowanych rozwiązań. Dotyczą one bezpieczeństwa ruchu drogowego i jego użytkowników, przekładając się na zdrowie i życie ludzi. Ponadto produktem zrealizowanego projektu jest bezobsługowy system komputerowy o cechach bezobsługowego systemu wbudowanego o wysokiej niezawodności. Model V został wybrany pomimo narżenia dużych kosztów związanych z fazą testowania. Atutami modelu V dziedzicznymi po modelu kaskadowym są: przejrzystość struktury, przewidywalność i łatwiejsze dostrzeżenie całego zakresu projektu.

W początkowym okresie realizacji projektu analizowano użycie modelu V, ze względu na istotny aspekt pokrycia testami opracowywanego rozwiązania przedstawionego problemu. Model V miał być wdrażany hybrydowo z metodyką iteracyjną. Po wyborze metodyki przyrostowej, postanowiono zrezygnować z modelu V, włączając wymagane testy w zakres projektu, realizując je przyrostowo, bez alternatywnych metodologii. Dzięki przyjętej strategii ominięto problem hybrydowego zestawienia bazującego na modelu kaskadowym, modelu V z metodyką ewolucyjno-przyrostową, która nie zakłada sekwencyjności, ani realizacji etapu od początku do końca.

⁵Przegląd modeli cyklu życia oprogramowania, https://www.researchgate.net/profile/Rafal-Kasprzyk/publication/229596276_Przeglad_modeli_cyklu_zycia_oprogramowania/links/5b420974458515f71cb23497/Przeglad-modeli-cyklu-zycia-oprogramowania.pdf, str. 54, 55 [dostęp: 27.05.2021].

Dodatkowo nie jest wymagane użycie wszystkich kategorii testów, więc realizacja metody V jest nadmiarowa w stosunku do wymagań projektu. Kolejnym powodem rezygnacji z modelu V jest wysoka, narzucona sztywność kolejności realizacji testów i podobnie jak w modelu kaskadowym, wymóg kompletności zestawów testów. Takie podejście zmniejszyłoby elastyczność realizacji projektu i byłoby trudne do zestawienia z przyjętymi uprzednio założeniami realizacji projektu.



Rysunek 3.4: Schemat modelu V, źródło: <https://images.app.goo.gl/57UVGrXw3d6cLaSw5> [dostęp 27.05.2021]

3.4.3 Schemat procesu wytwarzania

1. etap:
 - (a) Cel: Utworzenie wstępnej dokumentacji projektu.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: W podejściu trzech cykli udało uzyskać się wstępnie uzupełnioną dokumentację wstępną.
 - (d) Skutek: Zarysowano wstępnie: Dokument Założeń Wstępnych, Wstępny Plan Projektu.
 - (e) Plan: Planowane było dalsze rozwijanie dokumentacji w kolejnych cyklach.
 - (f) Uwagi: Ten punkt powstaje w trakcie tworzenia rozwiązania.
2. etap:
 - (a) Cel: Uściślenie wymagań we Wstępny Planie Projektu.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: Analiza podstawowej dokumentacji wstępnej.
 - (d) Skutek: Uszczegółowienie zakresu realizacji projektu.
 - (e) Plan: Planowany projekt systemu.
 - (f) Uwagi: Brak.

3. etap:
 - (a) Cel: Uścielenie wymagań we Wstępny Planie Projektu.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: Analiza podstawowej dokumentacji wstępnej.
 - (d) Skutek: Uszczegółowienie zakresu realizacji projektu.
 - (e) Plan: Planowany projekt systemu.
 - (f) Uwagi: Brak.
4. etap:
 - (a) Cel: Wydanie pierwszej wersji Projektu systemu.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: Wstępne zarysowanie projektowanego systemu.
 - (d) Skutek: Otrzymano ogólny zarys planowanego systemu.
 - (e) Plan: Planowane jest uścielenie wybranych fragmentów w projekcie systemu.
 - (f) Uwagi: Brak
5. etap:
 - (a) Cel: Uszczegółowienie jednego z aspektów w Projekcie Systemu.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: Projektowanie rozwiązania.
 - (d) Skutek: Otrzymano plan wdrożenia.
 - (e) Plan: Planowane jest wdrożenie planowanego rozwiązania.
 - (f) Uwagi: Brak
6. etap:
 - (a) Cel: Wdrożenie jednego z aspektów w Projekcie Systemu.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: Wdrożenie rozwiązania.
 - (d) Skutek: Otrzymano produkt.
 - (e) Plan: Planowane jest uścielenie wybranych fragmentów w projekcie systemu.
 - (f) Uwagi: Punkt wejścia dla kolejnego przyrostu
7. etap:
 - (a) Cel: Zestawienie prototypu.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: Łączenie wytworzonych produktów.
 - (d) Skutek: Otrzymano prototyp.
 - (e) Plan: Planowane jest testowanie prototypu.
 - (f) Uwagi:
8. etap:
 - (a) Cel: Testowanie prototypu.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: Testowanie wytworzonych produktów.
 - (d) Skutek: Poprawa jakości rozwiązania.
 - (e) Plan: Rozszerzenie funkcjonalności produktu.
 - (f) Uwagi:

3.5 Zakładane zasoby

1. Czas na realizację projektu
2. Budżet na pozyskanie wymaganej infrastruktury technicznej
3. Nadzór nad przebiegiem realizacji projektu
4. Umiejętności potrzebne do realizacji założeń projektowych
5. Wiedza dotycząca obsługi i integracji użytych technologii
6. Stanowisko pracy w celu realizacji założeń projektowych
7. Wymagana infrastruktura techniczna
8. Wymagana infrastruktura dokumentacyjna

Rozdział 4

Projekt Systemu

4.1 Architektura

Zawarcie opisu modułów wraz z powiązaniami składającymi na konstrukcję systemu¹. Architekturę tworzoną w ramach realizowanego projektu można podzielić na:

- architekturę sprzętową
- architekturę programową

4.1.1 Użyte wzorce projektowe

Dekompozycja systemu na trzy funkcjonalne bloki: modelu, widoku i prezentera była podstawą do utworzenia implementacji zgodnie ze wzorcem projektowym Model-View-Presenter (MVP), będącym pochodną od wzorca Model-View-Controller (MVC). Główna różnica pomiędzy tymi wzorcami i jednocześnie wyróżnik wzorca MVP to istnienie warstwy prezentera, który zawiera logikę biznesową sterującą widokiem i modelem, jednocześnie pośredniczącą w komunikacji pomiędzy tymi warstwami. Widok i model nie są bezpośrednio ze sobą skomunikowane. W realizowanej implementacji za widok odpowiadają klasy: MainViewActivity, FragmentActivity. Za warstwę modelu klasa DbManager z klasami pomocniczymi: DBHelper i CreateTable warstwę prezentera pełni klasa UiView z klasami pomocniczymi: Motorcycle, ForwardVehicle, UiElement, CLocation.

Podczas tworzenia implementacji wykorzystywano zasadę, aby nie powielać istniejącego kodu aplikacji w różnych miejscach w projekcie, ograniczając nadmiarowość tworzonego kodu oraz ułatwiając jego czytelność, jednocześnie łatwiej jest ponownie wykorzystywać istniejące już implementacje dla wybranych zastosowań, dlatego wykorzystano metodę "Nie powtarzaj siebie" (Don't Repeat Yourself) DRY, w praktyce tworząc kod oparty o stosowanie większej liczby metod.

Wykorzystując mechanizm dziedziczenia, w projektowanych klasach utworzono bazowy kontroler, zawierający bardziej niskopoziomową, mniej intuicyjną implementację, dzięki któremu możliwy był dostęp do niskopoziomowych operacji sprzętowych, przy jednoczesnym zagwarantowaniu mechanizmu dziedziczenia po sobie i rozszerzania oferowanej, podstawowej funkcjonalności o specyficzne zastosowania wysokopoziomowe. Możliwe było rozszerzanie podstawowej funkcjonalności przy jednoczesnym braku możliwości modyfikacji tej funkcjonalności. Takie podejście wpisuje się w treść zasady otwarte-zamknięte (Open-Closed-Principle) OCP. W projekcie zgodnie z tą zasadą realizowano dostęp do bazy danych, w którym warstwą pośrednią była klasa DBHelper, a klasą rozszerzającą funkcjonalność DbManager.

¹Żydzik K. Rak T. C# i MVC5, Helion 2015

Analogicznie zrealizowany został dostęp do usługi rest, tutaj warstwę pośrednią stanowiła klasa RestCtrl, a rozszerzeniem była klasa RestDistanceCtrl. Identyczne podejście zastosowano podczas implementacji elementów interfejsu graficznego, w którym rysowane obiekty takie jak samochód modelowany w klasie ForwardVehicle, dziedziczyły po podstawowej funkcjonalności klasy UiElement.

Podobna do OCP metodą jest zasada odwrócenia zależności (Dependency Inversion Principle) DIP, która podkreśla, że przepływ zależności pomiędzy warstwą abstrakcji a warstwą niskopoziomową, powinien przebiegać kierunku od warstwy niskopoziomowej, w kierunku warstwy abstrakcji, nie odwrotnie, dzięki temu warstwa niskopoziomowa jest odseparowana i hermetyczna na modyfikację, jednocześnie rozszerzalna. Według zasady DIP zrealizowano przepływ sterowania w klasach, wykorzystujących jednocześnie zasadę OCP.

Implementacja zrealizowana po procesie testowania posiada cechy zasady "Nie będziesz tego potrzebował" (You Ain't Gonna Need It) YAGNI, która sugeruje aby nie pisać niepotrzebnych, nadmiarowych implementacji, w celu zwiększenia użyteczności tworzonego kodu i oszczędzenia zasobów czasu w przypadku utworzenia niepotrzebnych implementacji. Stosując się do zasady YAGNI, powstały podczas pisania testów kod, poddany refaktoryzacji, nie zawiera nadmiarowych implementacji.

Zasady OCP i DIP są podzbiorem zasad SOLID.

W celu realizacji projektu wykorzystano następujące wzorce projektowe, implementacyjne i uznane tzw. dobre praktyki programowania:

- model MVP
- DRY
- OCP
- DIP
- YAGNI

4.1.2 Architektura systemu

W ramach utworzonej implementacji, składającej się na architekturę oprogramowania, wykonano dekompozycję systemu na moduły w celu:

- lepszej separacji funkcjonalnej odpowiedzialności za realizowane zadania
- lepszego zarządzania istniejącą implementacją
- większej podatności na rozbudowę
- łatwiejszej analizy
- większej podatności na testowanie

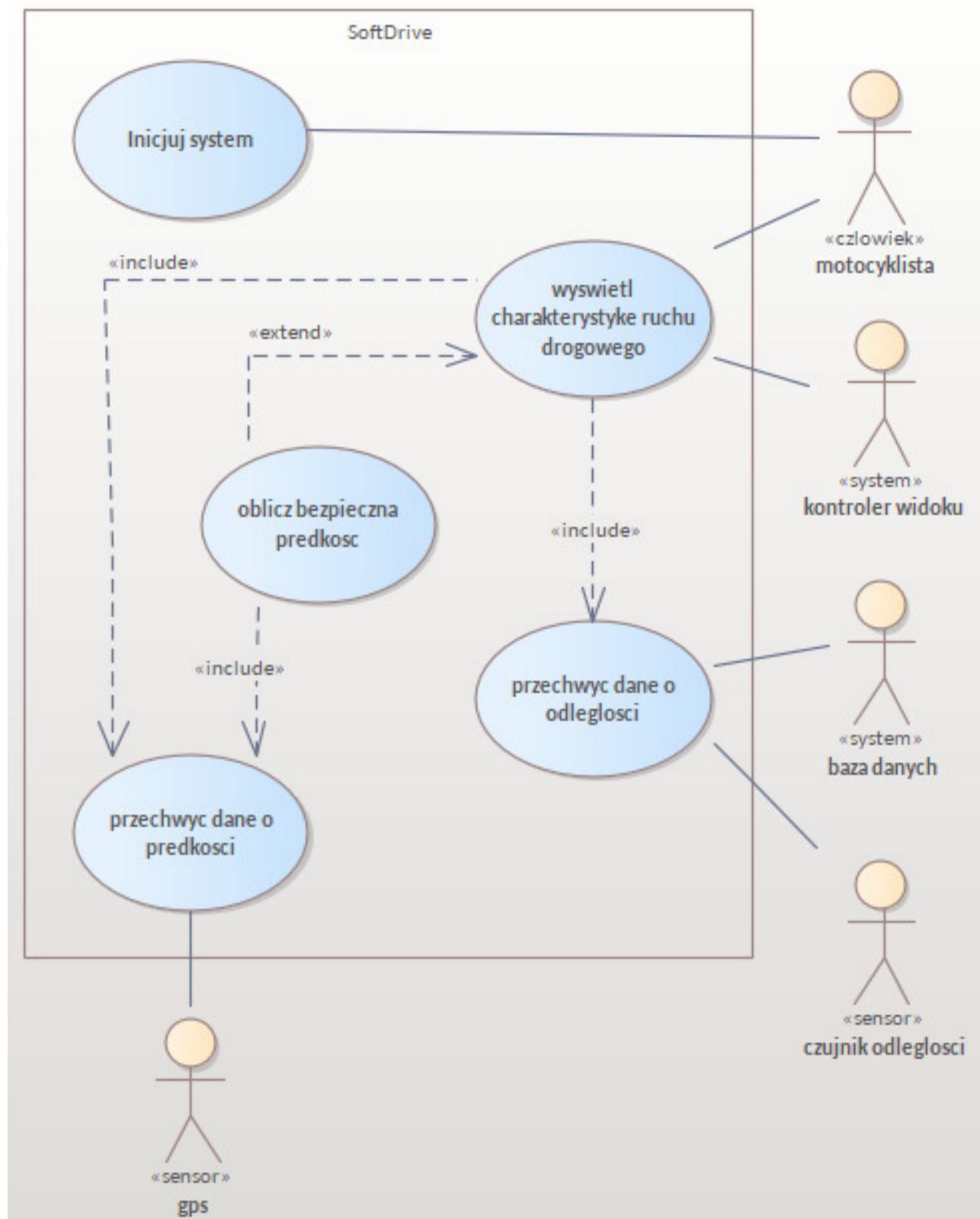
W tym celu kierowano się wytycznymi zasady separacji zagadnień (Separation of concern) SoC. Projekt systemu na charakter modułowy, w którym każdy element ma swoje charakterystyczne zastosowanie.

Całość realizowanego systemu, na który składają się moduły warstwy sprzętowej i oprogramowania, działa w architekturze klient - serwer. W nawiązaniu do specyfiki systemu, jako serwer działa część sprzętowa, udostępniając na żądanie klienta wartość odczytanej z czujnika odległości. Klientem w projektowanej architekturze jest aplikacja mobilna działająca na smartfonie, wysyłająca zapytania o wartość odległości do warstwy sprzętowej i odbierającą odpowiedzi od serwera.

4.1.3 Dekompozycja systemu na podsystemy

Diagram przypadków użycia

Graficzna reprezentacja przypadków użycia systemu.



Rysunek 4.1: Diagram przypadków użycia UML [źródło: własne]

Na diagramie motocyklista inicjalizuje system, który w następstwie wyświetla graficzna animację, charakteryzującą specyficzne uwarunkowania aktualnego ruchu drogowego, poprzez kontroler widoku, w tym celu przechwytywane są dane dotyczące odległości pojazdu poprzedzającego od pojazdu motocyklisty i zapisywane w bazie danych, jednocześnie przechwytywane są dane prędkości poruszania się motocyklisty, jeżeli motocyklista jedzie zbyt szybko lub odległość pomiędzy pojazdami jest zbyt mała obliczana jest bezpieczna prędkość jazdy i rysowana na graficznym interfejsie użytkownika.

Specyfikacja przypadków użycia

Dokumentacja poszczególnych przypadków użycia

1. Inicjuj system.

Nazwa: inicjuj system

Numer: 1

Poziom ważności: niski

Typ przypadku użycia: ogólny, niezbędny

Aktorzy: motocyklista

Krótki opis: Aktywacja internetu mobilnego oraz routera Wifi w telefonie.

Warunki wstępne: internet mobilny albo router Wifi nie są aktywne.

Warunki końcowe: Router Wifi powinien mieć ścisłe określona konfigurację.

Główny przepływ zdarzeń:

- (a) Motocyklista uruchamia aplikację.
 - (b) System sprawdza czy aktywny jest internet mobilny w telefonie.
 - (c) System wyświetla instrukcję jak uruchomić usługę mobilnego internetu użytkownikowi w przypadku wykrycia wyłączenia funkcji.
 - (d) Motocyklista po uruchomieniu routera Wifi potwierdza włączenie opcji w widoku aplikacji.
 - (e) System sprawdza czy aktywny jest router Wifi.
 - (f) System wyświetla instrukcję jak uruchomić i skonfigurować usługę routera Wifi użytkownikowi w przypadku wykrycia wyłączenia funkcji.
 - (g) Po ukończonej konfiguracji motocyklista potwierdza włączenie opcji w widoku aplikacji.
 - (h) Następuje uruchomienie rest serwisu.
-

Alternatywne przepływy zdarzeń:

- (c1) Włączony internet mobilny powoduje pominięcie wyświetlania widoku z instrukcją.
 - (f1) Włączony router Wifi powoduje pominięcie wyświetlania widoku z instrukcją.
-

Specjalne wymagania:

- (a) Poprawna konfiguracja routera Wifi.
-

Notatki i kwestie: Brak

2. Wyświetl charakterystykę ruchu drogowego.

Nazwa: Wyświetl charakterystykę ruchu drogowego

Numer: 2

Poziom ważności: wysoki

Typ przypadku użycia: szczegółowy, niezbędny

Aktorzy: motocyklista, kontroler widoku, baza danych, czujnik odległości, gps

Krótki opis: Wyświetlenie na wyświetlaczu widoku aplikacji z animacją charakteryzującą rzeczywistą odległość od pojazdu przed motocyklistą oraz aktualną prędkość jazdy. Dodatkowo wyświetlane jest ostrzeżenie o konieczności redukcji prędkości do w przypadku zbyt małej odległości lub zbyt dużej prędkości. W razie braku wykrycia poprawnej odległości wyświetlana jest informacja motocykliście. Wyświetlane są także wartości odczytanej odległości i prędkości.

Warunki wstępne: Odczyt odległości i prędkości musi być prawidłowy.

Warunki końcowe: Widok powinien być na bieżąco aktualizowany.

Główny przepływ zdarzeń:

- (a) Na wyświetlaczu widoczna jest animacja odwzorowująca wartość rzeczywistej odległości od pojazdu jadącego przed motocyklistą.
 - (b) Podczas zmiany odległości od pojazdu poruszającego się przed motocyklistą następuje aktualizacja wyświetlanej aktualnej odległości.
 - (c) Animacja z odwzorowaniem wartości odległości wyświetlana jest w sposób ciągły
 - (d) Wyświetlanie aktualnej wartości prędkości.
 - (e) Wyświetlanie aktualnej wartości odległości.
-

Alternatywne przepływy zdarzeń:

- (b1) Zmierzona odległość od pojazdu jadącego za motocyklistą jest zbyt mała.
 - (b2) Zmierzona prędkość jazdy pojazdu za motocyklistą jest zbyt duża.
 - (b3) Następuje wyświetlenie wartości bezpiecznej prędkości jazdy.
 - (c1) System wykrywa błąd odczytu odległości.
 - (c2) Na wyświetlaczu pojawia się widok z informacją o nieprawidłowym przetwarzaniu danych dotyczących odległości od pojazdu poruszającego się przez motocyklistą.
-

Specjalne wymagania:

- (a) Wysoka niezawodność systemu.
 - (b) Wysoka dokładność wskazań systemu.
 - (c) Intuicyjny i ergonomiczny interfejs użytkownika.
-

Notatki i kwestie: Brak

3. Oblicz bezpieczną prędkość.

Nazwa: Oblicz bezpieczną prędkość

Numer: 3

Poziom ważności: wysoki

Typ przypadku użycia: szczegółowy, niezbędny

Aktorzy: kontroler widoku, baza danych, gps, czujnik odległości

Krótki opis: Logika implementacji systemu obliczająca bezpieczną wartość prędkości poruszania się motocyklisty, na podstawie zmierzonej odległości od pojazdu przed motocyklistą i prędkości poruszania się motocyklisty.

Warunki wstępne: Kontroler widoku wykrywa zagrożenie zachowania bezpiecznej odległości poruszania się względem pojazdu jadącego przed motocyklistą przy określonej prędkości jazdy.

Warunki końcowe: Kontroler widoku nie wykrywa zagrożenia bezpieczeństwa w przypadku konieczności nagłego hamowania pojazdem motocyklisty.

Główny przepływ zdarzeń:

- (a) Kontroler widoku analizuje na bieżąco dane odległości od pojazdu jadącego przez motocyklistą i prędkości poruszania się motocyklisty.
 - (b) Kontroler widoku wykrywa zagrożenie bezpieczeństwa ruchu drogowego na podstawie analizowanych danych.
 - (c) Kontroler widoku zgłasza żądanie obliczenia bezpiecznej prędkości poruszania się motocyklisty w określonych warunkach odległości pomiędzy motocyklistą a pojazdem jadącym przed nim.
 - (d) Kontroler widoku przetwarza uzyskany wynik wartości bezpiecznej prędkości jazdy.
-

Alternatywne przepływy zdarzeń:

- (b1) Analiza danych nie wskazuje na zagrożenie bezpieczeństwa dla ruchu pojazdów.
-

Specjalne wymagania:

- (a) Wysoka niezawodność systemu.
 - (b) Prawidłowa wartość uzyskanych obliczeń sugerowanej prędkości jazdy.
-

Notatki i kwestie: Brak

4. Przechwyć dane o odległości.

Nazwa: Przechwyć dane o odległości

Numer: 4

Poziom ważności: ważny

Typ przypadku użycia: szczegółowy, niezbędny

Aktorzy: czujnik odległości, baza danych, kontroler widoku, gps

Krótki opis: Pomiar aktualnej wartości zmierzonej odległości z czujnika odległości w warstwie sprzętowej i wczytanie zmierzonej wartości do systemu aplikacji mobilnej w celu zapisania pobranej wartości w bazie danych.

Warunki wstępne: Wymagana jest poprawne połączenie sieciowe pomiędzy komponentami i prawidłowość wskazań sensora odległości wraz z zapewnionym dostępem do odczytywanych danych.

Warunki końcowe: System sensora odległości powinien zapewniać ponowny odczyt wartości odległości na żądanie systemu aplikacji mobilnej.

Główny przepływ zdarzeń:

- (a) System sensora odległości nasłuchuje zapytań o wartość pomiaru odległości.
 - (b) System aplikacji mobilnej wysyła zapytanie o aktualną wartość odległości.
 - (c) System przetwarza zapytanie i dokonuje pomiaru odległości.
 - (d) System sensora udostępnia w odpowiedzi do systemu aplikacji mobilnej zmierzona wartość odległości.
 - (e) System aplikacji mobilnej rejestruje odebraną wartość odległości.
 - (f) System aplikacji mobilnej zapisuje w bazie danych odebraną wartość odległości.
 - (g) System aplikacji mobilnej umożliwia odczyt zapisanej wartości odległości na żądanie.
-

Alternatywne przepływy zdarzeń: Brak

Specjalne wymagania:

- (a) Prawidłowa praca systemu czujnika odległości.
 - (b) Prawidłowe połączenie pomiędzy systemem czujnika odległości i systemem aplikacji mobilnej.
-

Notatki i kwestie: Komunikacja pomiędzy komponentem sprzętowym systemu a aplikacją mobilną działa na zasadzie mechanizmu żądanie - odpowiedź.

5. Przechwyć dane o prędkości.

Nazwa: Przechwyć dane o prędkości

Numer: 5

Poziom ważności: ważny

Typ przypadku użycia: szczegółowy, niezbędny

Aktorzy: czujnik odległości, baza danych, kontroler widoku, gps

Krótki opis: Pomiar aktualnej wartości zmierzonej prędkości z GPS w warstwie sprzętowej smartfona i wczytanie zmierzonej wartości do systemu aplikacji mobilnej w celu dalszego przetwarzania..

Warunki wstępne: Wymagana jest pozwolenie na dostęp do sensora GPS telefonu i prawidłowe wskazania sensora wraz z zapewnionym dostępem do odczytanych danych.

Warunki końcowe: System sensora GPS powinien aktualizować zmianę odczytu wartości prędkości w systemie aplikacji mobilnej.

Główny przepływ zdarzeń:

- (a) System aplikacji mobilnej nasłuchuje zmiany wartości pomiaru prędkości.
 - (b) GPS dokonuje pomiaru odległości.
 - (c) System aplikacji mobilnej rejestruje odebraną wartość prędkości.
-

Alternatywne przepływy zdarzeń: Brak

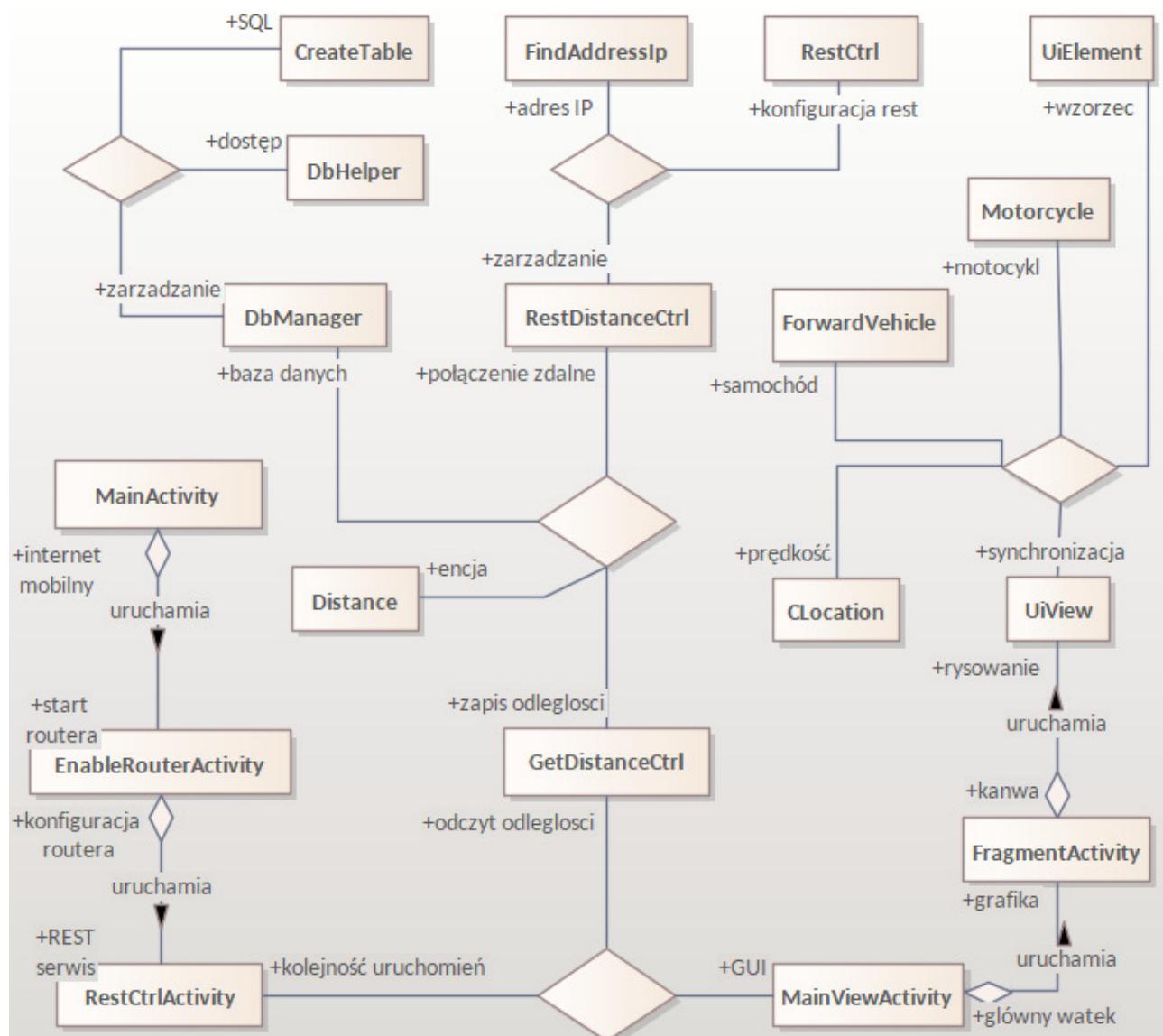
Specjalne wymagania:

- (a) Prawidłowa praca systemu czujnika GPS.
 - (b) Prawidłowe połączenie pomiędzy systemem czujnika odległości i systemem aplikacji mobilnej.
-

Notatki i kwestie: Brak

Diagram klas

Charakterystyka statyki systemu w postaci diagramu klas.

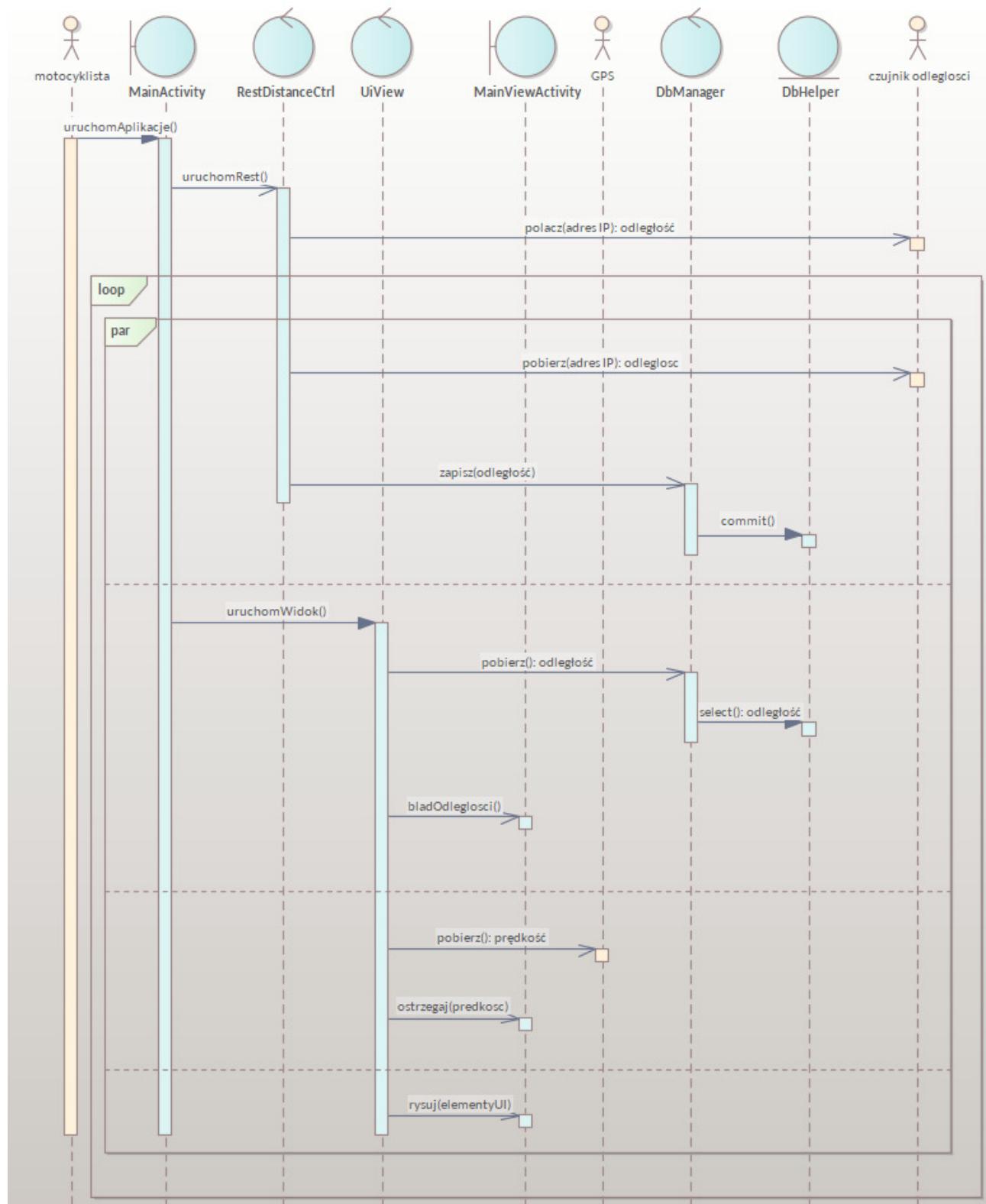


Rysunek 4.2: Diagram klas UML [źródło: własne]

4.1.4 Opis interakcji pomiędzy podsystemami

Diagram sekwencji

Opis dynamiki systemu w postaci diagramu sekwencji, z uwzględnieniem interakcji pomiędzy podsystemami.

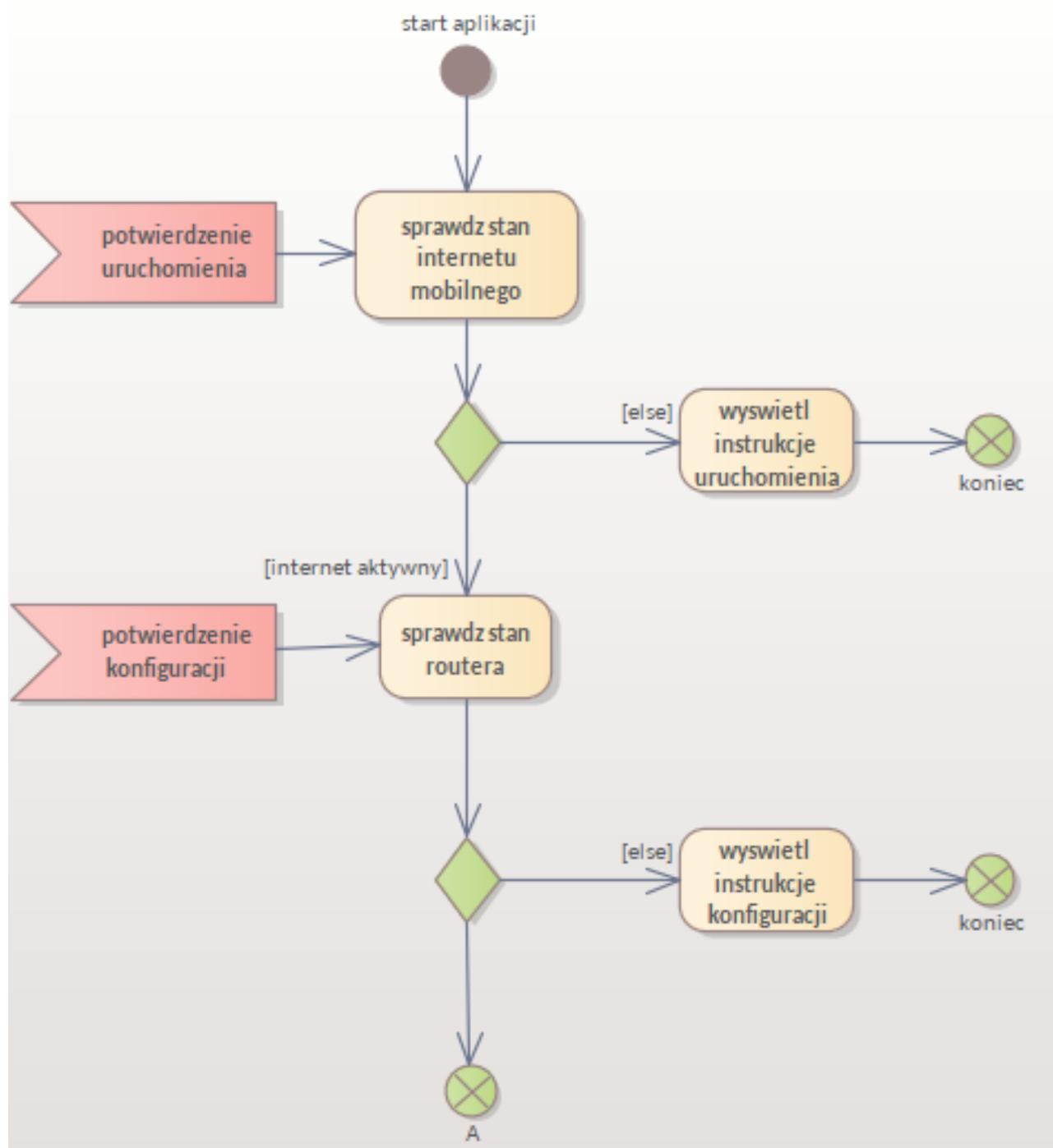


Rysunek 4.3: Diagram sekwencji UML systemu [źródło: własne]

1. Motocyklista inicjalizuje aplikację poprzez funkcjonalność klasy MainActivity
2. Z klasy MainActivity uruchomiona zostaje jako pierwsza asynchronicznie metoda uruchomRest() uruchamiająca w tle usługę rest
3. Z tej samej klasy uruchomiona zostaje jako druga, po interwale czasowym, metoda uruchomWidok() uruchamiająca w głównym wątku funkcjonalności odpowiedzialne za rysowanie widoku oraz za pobieranie danych odległości i ich przetwarzanie w celu ustalenia czy istnieje zagrożenie ruchu drogowego.
4. Istnieją cztery równolegle uruchomione, funkcjonalne wątki, z których każdy jest cyklicznie ponawiany przez cały czas pracy systemu
5. Pierwszy wątek określa proces odczytu danych o odległości z czujnika i zapis ich w bazie danych
6. Drugi wątek określa proces pobierania danych o odległości z bazy danych i kontrola błędów odczytu
7. Trzeci wątek określa proces odczytu danych o prędkości z czujnika i informowanie motocyklisty o zagrożeniu bezpieczeństwa
8. Czwarty wątek określa proces rysowania graficznych elementów interfejsu użytkownika

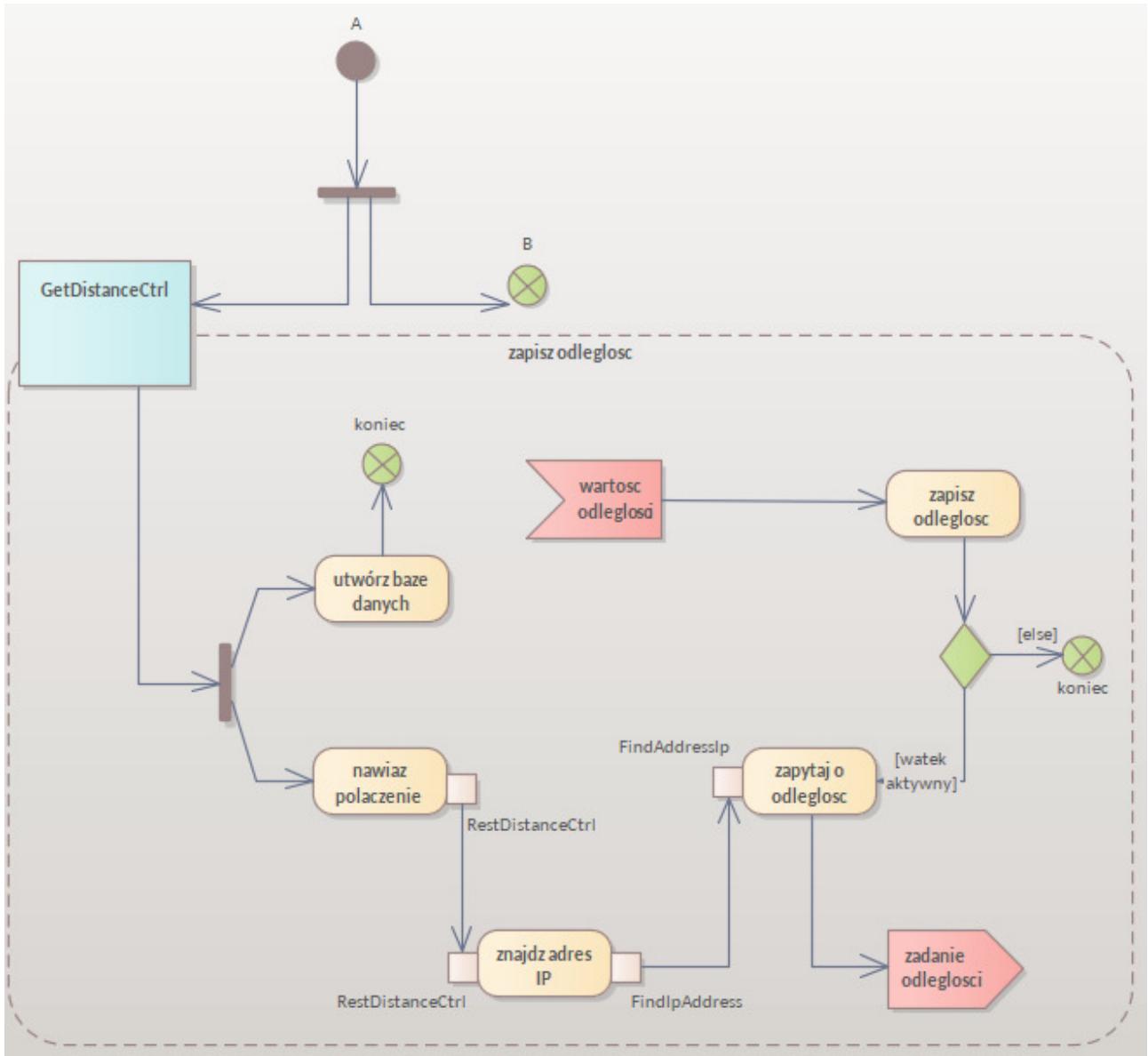
Diagram czynności

Opis dynamiki systemu w postaci diagramów czynności.



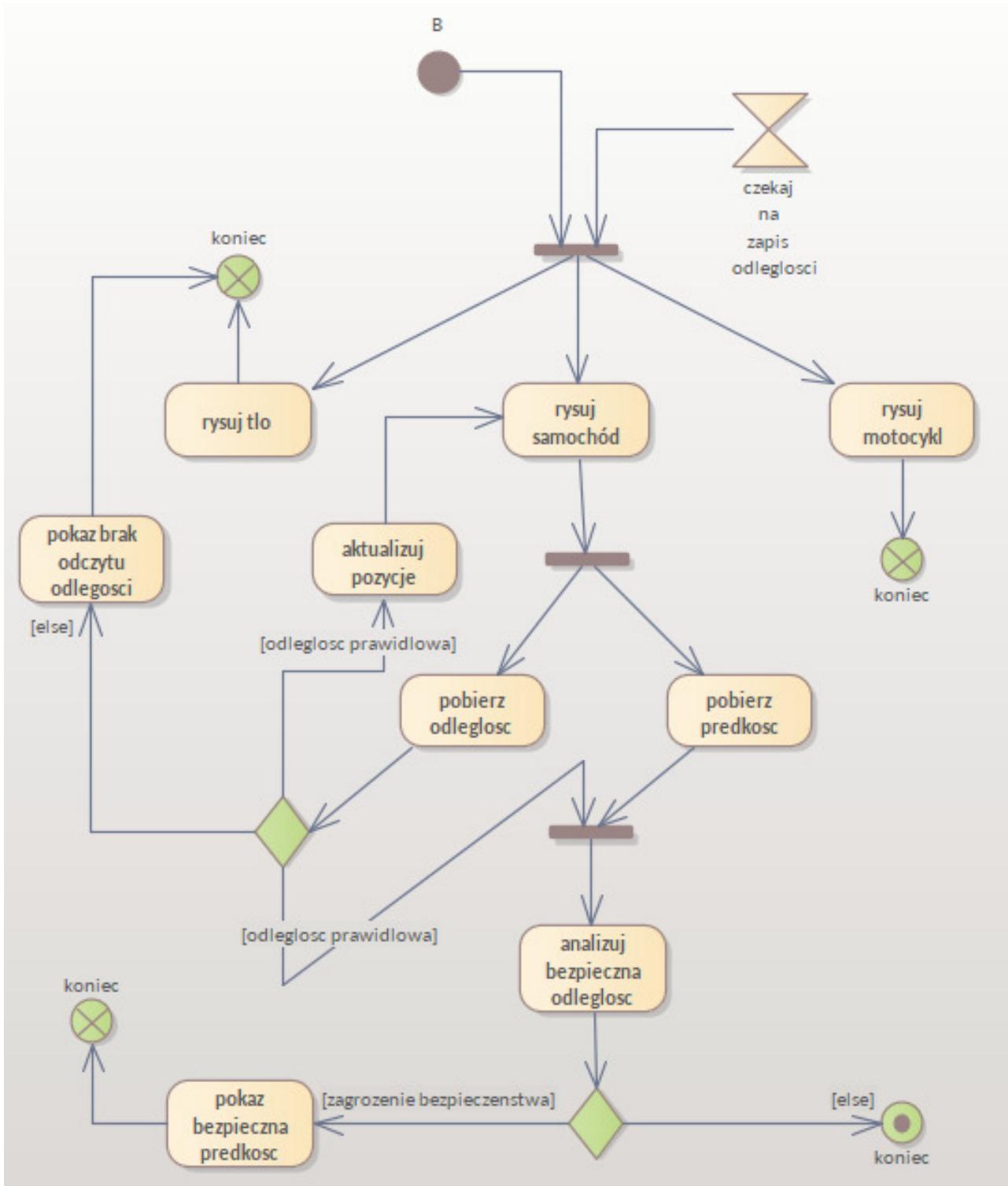
Rysunek 4.4: Diagram czynności UML wstępnej konfiguracji systemu [źródło: własne]

Przedstawiony przepływ sterowania dotyczy procesu inicjalizacji systemu



Rysunek 4.5: Diagram czynności UML odczytu odległości przez system [źródło: własne]

Przedstawiony przepływ sterowania dotyczy procesu pobierania wartości odległości z czujnika i jej zapisywania w bazie danych.



Rysunek 4.6: Diagram czynności UML rysowania GUI przez system [źródło: własne]

Przedstawiony przepływ sterowania dotyczy procesu rysowania i aktualizacji widoku interfejsu użytkownika.

4.2 Wykorzystane elementy

System jest podzielony na dwie architektury: sprzętową i programową

4.2.1 Część sprzętowa

- Minikomputery Raspberry Pi 3B+
- Warstwa sprzętowa tylnej części pojazdu została oprogramowana językiem Python.
- Konfigurację systemu Raspbian wykonano systemowym terminaliem wykorzystując język Bash.
- Użyto kompilatora g++ dla języka C++.
- Użyto standardowego interpretera języka Python.
- Jako edytora kodu wykorzystano edytor MS Code z dostępem do rozszerzeń.
- Wymagana była komunikacja z internetem poprzez dostępny moduł sieci bezprzewodowej WiFi.
- Adapter interfejsu USB UART
- Laserowy czujnik odległości
- Laserowy skaner obiektów w przestrzeni 2D
- Kamera szerokokątna
- Diody IR
- Kondensatory elektrolityczne
- Płytki prototypowe
- Przewody połączeniowe
- Dwa zasilacze DC
- Uniwersalny miernik wielkości elektrycznych
- Bateria
- Adapter interfejsu Slamtec STC-A0317
- Laminat PCB
- Frezarka CNC
- Frez 0.1 mm
- Wiertło 0.6 mm
- Piankowa taśma montażowa
- Przewody zasilające
- Dwie karty SD
- Trzy przewody micro USB
- Wtyk USB C
- Urządzenia peryferyjne
- Laptop
- Magistrala i2c
- Interfejs CSI
- Smartfon
- Łączność GSM
- Radiatory
- Router Wifi
- Moduł GPS smartfona
- Przewód ethernet
- Złącza goldpin
- Adapter gniazda rj45 do kostki przyłączeniowej
- Taśma izolacyjna
- Przewody połączeniowe
- Ściągacz izolacji

4.2.2 Część programowa

Aplikacja mobilna

- Protokół HTTP
- Plugin do Firefoxa RESTED generujący żądania HTTP
- Przeglądarka webowa Firefox
- System operacyjny Windows 10 i 11 Preview.
- system operacyjny Android.
- Android Studio IDE.
- Emulator AVD dla Android Studio.
- Język programowania Java.
- Testowanie jednostkowe biblioteką JUnit i Robolectric
- Wykorzystanie biblioteki Retrofit2.
- Pobieranie danych usługą Rest API.
- Użycie animacji w aktywnościach interfejsu.
- Wykorzystanie fragmentów w aktywnościach interfejsu.
- Użycie programu kreacji grafiki 2D GIMP.
- Edytor graficzny Paint
- Utworzenie i konfiguracja prywatnej sieci lokalnej.

Systemy wbudowane

- Użycie programu PuTTY
- Sterownik adaptera USB UART
- Protokół SSH
- Oprogramowanie Raspi-config
- Terminal systemowy
- System operacyjny Linux, dystrybucja Raspbian.
- Edytor kodu MS Code.
- Pluginsy do MS Code dla języka C++.
- Język programowania C++.
- Język skryptowy powłoki BASH.
- Kompilator g++.
- Moduł usługi Rest API.
- Usługa transmisji danych na żądanie z Rest API.
- Sterowniki systemowe dla sensora odległości.
- Konfiguracja automatycznego wyszukiwanie i połączenia z prywatną siecią lokalną.
- Konfiguracja automatycznego logowania i uruchamiania skryptów przy starcie systemu
- automatyczne wyszukiwanie i połączenie z prywatną siecią lokalną.
- Pluginsy do MS Code dla języka Python.
- Język programowania Python.
- Interpreter języka Python.
- Menadżer pakietów pip3 języka Python
- biblioteka Flask języka Python
- Środowisko graficzne PIXEL dla systemu Linux
- Menadżer pakietów apt-get

4.3 Projekt rozwiązań sprzętowych

Część sprzętowa realizowanego projektu powinna składać się z prototypu: z możliwością zasilania układu minikomputera, układem minikomputera i podłączeniem do sensora odległości oraz sensora odległości.

4.4 Projekt interfejsu

Planowane aspekty realizacji interfejsu użytkownika.

4.4.1 Interfejs użytkownika

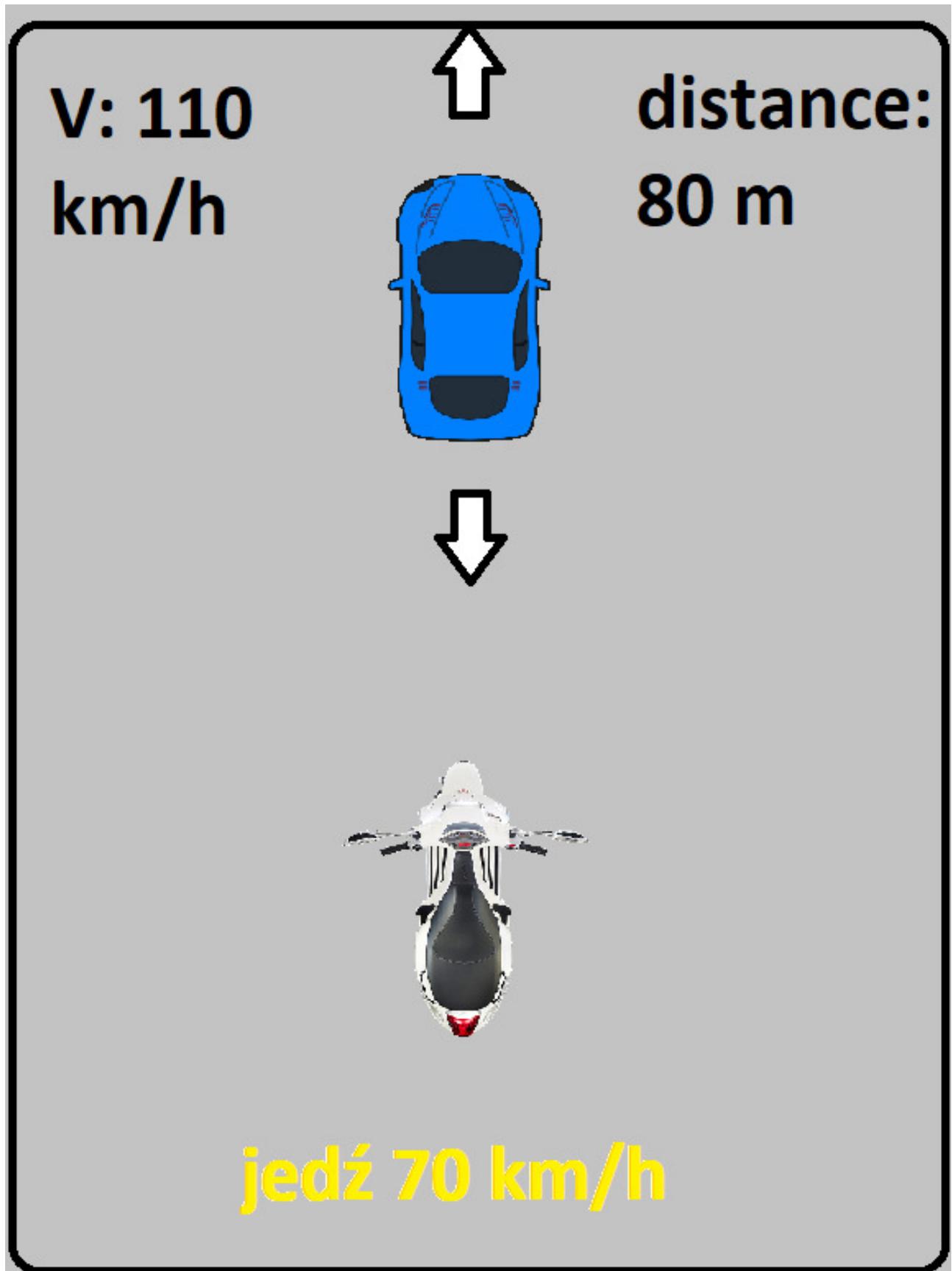
Na prototypowym interfejsie umieszczono na stałe wizerunek motocyklisty, przed nim w kierunku wertykalnym porusza się, zmieniając położenie, wizerunek samochodu poruszającego się przed motocyklistą. Podczas mierzenia odległości, zmienia się położenie samochodu, od odległości styku z przodem motocykla gdy zmierzona odległość jest minimalna, do wyłączenia wizerunku pojazdu, w przypadku pomiaru odległości przekraczającej zakres pomiarowy czujnika odległości. W górnej części ekranu wyświetlana jest na bieżąco prędkość poruszania się motocyklisty oraz zmierzona przednia odległość. W dolnej części ekranu pojawia się monit o zmniejszeniu prędkości jazdy do sugerowanej, w chwili wykrycia zagrożenia bezpieczeństwa ruchu pojazdem, przy aktualnych parametrach zmierzonej odległości i prędkości. W chwili braku zagrożenia monit nie jest widoczny. Tło animacji jest z stonowanym kolorze szarym.

4.4.2 Założenia konstrukcji interfejsu

Projektowany interfejs w celu zachowania bezpieczeństwa jazdy:

1. powinien być przejrzysty w celu trafnej sygnalizacji motocyklistie parametrów podróży
2. nie powinien wyświetlać elementów w sposób gwałtowny odwracających uwagę od prowadzenia pojazdu
3. powinien być w stonowanych kolorach aby niepotrzebnie nie przyciągał uwagi kierującego
4. powinien wyświetlać animację w sposób płynny
5. nie powinien umożliwiać interakcji z użytkownikiem
6. powinien wyłączyć się po przechwyceniu zdarzenia próby interakcji, aby nie była możliwa w czasie jazdy
7. powinien być cały czas włączony podczas użytkowania w celu większego komfortu użytkowania
8. powinien być wyświetlany z maksymalną jasnością w porze dziennej
9. powinien być wyświetlany z minimalną jasnością w porze nocnej
10. powinien mieć kolorystykę z przewagą odcieni szarości podczas użytkowania w nocy
11. nie powinien generować żadnych dźwięków
12. powinien mieć monotonny charakter animacji, w celu zwiększenia uwagi kierującego na otoczeniu ruchu drogowego
13. w przypadku wystąpienia błędu powinien sygnalizować powstanie błędu albo wyłączać się automatycznie, aby uniknąć błędnych wskazań motocyklistie, prowadzących do błędnych decyzji w ruchu drogowym

4.4.3 Projekt ekranów



Rysunek 4.7: Prototyp interfejsu systemu [źródło: własne]

Logo

info

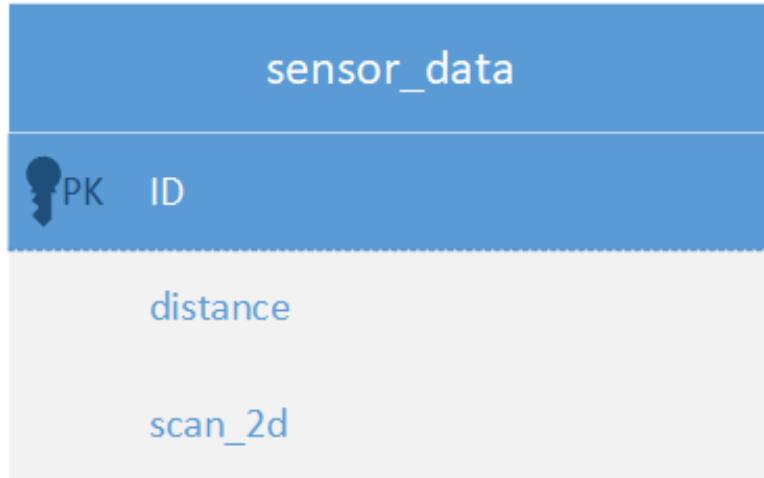
button

hidden message

Rysunek 4.8: Prototyp interfejsu inicjalizacji systemu [źródło: własne]

4.5 Projekt bazy danych

4.5.1 Schemat implementacyjny bazy



Rysunek 4.9: Schemat ERD bazy danych SensorData.db [źródło: własne]

4.5.2 Model logiczny dla bazy

implementacja modelu logicznego bazy danych

Nazewnictwo w bazie

Listing 4.1: Model logiczny nazewnictwa obiektów bazy danych

```
public class CreateTable {  
  
    public static class TableSensorData implements BaseColumns {  
        public static final String TABLENAME = "sensor_data";  
        public static final String COLUMN_NAME_DISTANCE = "distance";  
        public static final String COLUMN_NAME_SCAN_2D = "scan_2d";  
    }  
  
    public static final String SQL_CREATE_TABLE_SENSOR_DATA =  
        "CREATE_TABLE_" + TableSensorData.TABLENAME + "(" +  
        TableSensorData.ID + "_INTEGER_PRIMARY_KEY," +  
        TableSensorData.COLUMN_NAME_DISTANCE + "_INTEGER," +  
        TableSensorData.COLUMN_NAME_SCAN_2D + "_TEXT")";  
  
    public static final String SQL_DELETE_TABLE_SENSOR_DATA =  
        "DROP_TABLE_IF_EXISTS_" + TableSensorData.TABLENAME;  
}
```

Klasa CreateTable przechowuje w stałych polach zapytania języka SQL, w celu utworzenia na nich podstawie struktury bazy danych.

Warstwa pośrednicząca

Listing 4.2: Model logiczny warstwy pośredniczącej dla bazy danych

```
public class DBHelper extends SQLiteOpenHelper {  
  
    public static final int DATABASE_VERSION = 24;  
    public static final String DATABASE_NAME = "SensorData.db";  
  
    public DBHelper(Context context) {  
        // If you change the pl.pjatk.softdrive.database schema, you must increment the pl.pjatk.softdrive.database version.  
        super(context, DATABASE_NAME, null, DATABASE_VERSION);  
    }  
    public void onCreate(SQLiteDatabase db) {  
        db.execSQL(CreateTable.SQL_CREATE_TABLE_SENSOR_DATA);  
    }  
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
        db.execSQL(CreateTable.SQL_DELETE_TABLE_SENSOR_DATA);  
        onCreate(db);  
    }  
    public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
        onUpgrade(db, oldVersion, newVersion);  
    }  
}
```

Klasa DbHelper stanowi warstwę pośredniczącą w komunikacji pomiędzy klasą SQLiteOpenHelper, odpowiadającą za zarządzanie bazą danych, a kontrolerem DbManager utworzonym na potrzeby dostosowania implementacji aplikacji do wymagań komunikacyjnych bazy danych. W celu zresetowania bazy danych i skasowania jej zawartości należy zmienić wersję bazy w polu DATABASE_VERSION.

Kontroler

Listing 4.3: Model logiczny warstwy kontrolera dla bazy danych

```

public class DbManager extends DBHelper{

    public static final long MAX_ROW_COUNT = 200;
    private ExecutorService ex;
    private DBHelper dbHelper;
    private int distance;
    private String scan2d;

    public DbManager(Context context) {
        super(context);
        dbHelper = this;
        distance = 0;
        scan2d = "null";
        ex = Executors.newCachedThreadPool();
    }

    public int getDistance() {
        return distance;
    }

    public void setDistance(int distance) {
        this.distance = distance;
    }

    public String getScan2d() {
        return scan2d;
    }

    public void setScan2d(String scan2d) {
        this.scan2d = scan2d;
    }

    public boolean dbCommit() {

        // Gets the data repository in write mode
        SQLiteDatabase dbWrite = dbHelper.getWritableDatabase();

        // Create a new map of values, where column names are the keys
        ContentValues values = new ContentValues();
        values.put(CreateTable.TableSensorData.COLUMN_NAME_DISTANCE, distance);
        //values.put(CreateTable.TableSensorData.COLUMN_NAME_SCAN_2D, scan2d);

        // Insert the new row, returning the primary key value of the new row
        long newRowId = dbWrite.insert(CreateTable.TableSensorData.TABLE_NAME, null, values);
        Log.v("database", "insert_new_row");

        // clear database for more efficiency
        if (getRowCount(dbWrite) > MAX_ROW_COUNT) {
            clearDb(dbWrite);
            ContentValues valuesDel = new ContentValues();
            valuesDel.put(CreateTable.TableSensorData.COLUMN_NAME_DISTANCE, -1);
            dbWrite.insert(CreateTable.TableSensorData.TABLE_NAME, null, valuesDel);
            Log.v("database", "clear_database_rows");
        }

        dbWrite.close();

        return true;
    }

    private void clearDb(SQLiteDatabase dbWritableDatabase) {
        dbWritableDatabase.execSQL("delete from " + CreateTable.TableSensorData.TABLE_NAME);
    }

    public long getRowCount(SQLiteDatabase dbWritableDatabase) {
        long count = DatabaseUtils.queryNumEntries(dbWritableDatabase, CreateTable.TableSensorData.TABLE_NAME);
        return count;
    }

    public int getDbDistance() throws InterruptedException {
        SQLiteDatabase dbRead = dbHelper.getReadableDatabase();

        String[] projection = {
            CreateTable.TableSensorData.COLUMN_NAME_DISTANCE
        };

        Cursor cursor = dbRead.query(
            CreateTable.TableSensorData.TABLE_NAME,
            projection,           // The array of columns to return (pass null to get all)
            null,                // The columns for the WHERE clause
            null,                // The values for the WHERE clause
            null,                // don't group the rows
            null,                // don't filter by row groups
            null
        );
    }
}

```

```

int distance = -2;

try {
    // retry cursor read data
    int count = 0;
    while(! cursor.moveToLast()) {
        Thread.sleep(50);
        count++;
        if(count > 20) {
            // if too much retries
            cursor.close();
            dbRead.close();
            return -4;
        }
    }
    // read distance from db
    distance = cursor.getInt(cursor.getColumnIndex(CreateTable.TableSensorData.COLUMN_NAME_DISTANCE));
    cursor.close();
} catch (CursorIndexOutOfBoundsException e) {
    distance = -3;
}

dbRead.close();

return distance;
}

```

Menedżer bazy danych utworzony na potrzeby systemu, umożliwia w prosty dodanie wartości odległości do bazy danych oraz jej odczyt.

Aby dodać wartość dystansu do bazy, należy w pierwszej kolejności zmienić stan obiektu, przechowującego pamięć o wartości zmiennej do zapisania, metodą setDistance(int distance). Następnie przy użyciu metody dbCommit() wartość stanu pola obiektu jest zapisywana do bazy danych.

Aby zrealizować proces, najpierw należy uzyskać referencję do obiektu bazy danych, za pośrednictwem klasy DBHelper, z trybem zapisu do bazy. Następnie należy dokonać mapowania zapisywanych wartości na kolumny w bazie danych obiektem typu ContentValues, ten obiekt zapisujemy do bazy. Od tego momentu w bazie danych znajduje się zapisana wartość, jeżeli nie zostanie zgłoszony wyjątek.

W metodzie dbCommit() została dodatkowo zaimplementowana logika automatycznie kasująca istniejącą wpisy do bazy i wysyłającą do bazy wpis o wartości -1, zamiast domyślnie ustawionych jako górny limit dla bazy 200 encji. Taka logika ma na celu zachowanie odpowiedniej wydajności bazy danych, aby nie stała się wąskim gardłem wydajności systemu, którego działanie opiera się na cyklicznych wpisach i odczytach najnowszych wpisów do bazy w częstotliwością domyślnie ustawioną na 250 milisekund. Wartość -1 służy celom diagnostycznym systemu i przy tak ustawionej częstotliwości wpisów, jej wartość nie ma znaczącego wpływu na działający system. W celu odczytu najnowszego wpisu wartości odległości z bazy danych należy posłużyć się metodą getDbDistance(), która zwraca wartość odległości typu Integer.

W metodzie ustawiono tryb odczytu w referencji do obiektu bazy, przy użyciu warstwy pośredniczącej DBHelper.

Wykorzystując referencję, do bazy danych wysłano kwerendę z zapytaniem o wartość odległości, obiektem zwracanym jest typ Cursor.

Następuje odpytanie obiektu typu Cursor o posiadaną najnowszą wartość wpisu typu Integer według podanych przy wysłaniu do bazy wymagań, o wybór wszystkich wpisów o odległości w kolumnie. Odpytywanie z częstotliwością 50 milisekund trwa dopóki obiekt Cursor nie zwróci wyniku odległości, podczas gdy, przy każdym braku wartości zwracanego wyniku następuje ponowne zapytanie do bazy o wynik i obiekt typu Cursor jest odświeżany. w przypadku prawidłowego odczytu odległości zwracany jest wynik, w przypadku wyjścia podczas iteracji po przechowywanych danych na obiekcie typu Cursor poza dostępny zakres zwracana jest w wyniku wartość -3, gdy podjęto 20 prób odczytu z bazy danych z częstotliwością 50 milisekund i nie odczytano odległości zwracana jest wartość -4, ustawione częstotliwość i częstotliwość są modyfikowalne w celu optymalizacji systemu. Wartość -2 zwracana jest gdy nie nastąpi wejście do pętli while i rozpoczęcie procesu odpytywania obiektu typu Cursor.

Wykorzystane wartości ujemne odległości służą celom analitycznym podczas przeprowadzania procesu debugowania systemu.

4.5.3 Specyfikacja tabeli bazy

Projekt zakłada utworzenie relacyjnej bazy danych. W systemie Android domyślną bazą danych jest SQLite. Baza danych o nazwie SensorData.db zbudowana jest z jednej tabeli o nazwie sensor_data z trzema kolumnami:

- ID - PK, Integer, autoincrement
- distance - Integer
- scan_2d - Text

Klucz główny bazy jest automatycznie dopisywany wraz z commitami nowych encji i jest typu Integer. Na krotki distance i scan_2d nie są narzucone żadne ograniczenia, z wyjątkiem obligatoryjności typu danych.

Rozdział 5

Wdrożenie rozwiązania

Procesy wykonanych prac w celu realizacji rozwiązania.

5.1 Pierwsza konfiguracja systemu

Wstępna konfiguracja systemu.

5.1.1 Opis wymagań wstępnych

Wymagania wstępne procesu realizacji rozwiązania.

Wymagania sprzętowe

W celu realizacji wstępnej konfiguracji komunikacji urządzeń wykorzystano rozwiązania sprzętowe:

1. minikomputer Raspberry Pi 3 B+ dla przedniej strony pojazdu.
2. minikomputer Raspberry Pi 3 B+ dla tylnej strony pojazdu.
3. smartfon Samsung Galaxy Note 10+
4. platformę USB - UART PL2303 USB UART Board (typ A)
5. laptop Lenovo Ideapad S540
6. baterię powerbank Samsung Battery Pack

Wymagania programowe

W celu realizacji wstępnej konfiguracji komunikacji urządzeń wykorzystano rozwiązania programowe:

1. system operacyjny Windows 10 Home
2. system operacyjny Windows 11 Preview
3. system operacyjny Android w wersji 10
4. system operacyjny Raspbian z obrazu 2020-12-02-rpios-buster-armhf-lite.deb
5. sterownik adaptera usb uart dla systemu Windows PL2303 Prolific DriverInstaller v1.8.0
6. środowisko instalacyjne systemu operacyjnego Raspbian na karcie SD
7. program udostępniający terminal komunikacyjny dla połączenia szeregowego PuTTY w wersji 0.74 64 bit

5.1.2 Przygotowanie środowiska instalacyjnego systemu Raspbian

Prace inicjalizujące środowisko systemu.

konfiguracja adaptera USB UART

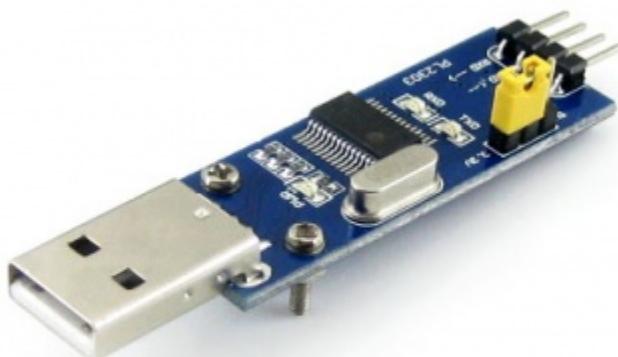
konfiguracja narzędzi dla wdrożenia rozwiązania.

Instalacja sterownika adaptera Komunikacja pomiędzy minikomputerem RaspberryPi a komputerem przenośnym została nawiązana dzięki połączeniu szeregowemu.

Aby połączenie szeregowe pomiędzy urządzeniami ustanowić w pierwszym etapie realizacji należało zainstalować sterownik do adaptera USB UART (rysunek 5.1) w systemie Windows. Po podłączeniu adaptera do portu USB, system Windows nie rozpoznaje tego urządzenia (rysunek 5.2). Wymagana jest instalacja sterowników urządzenia.

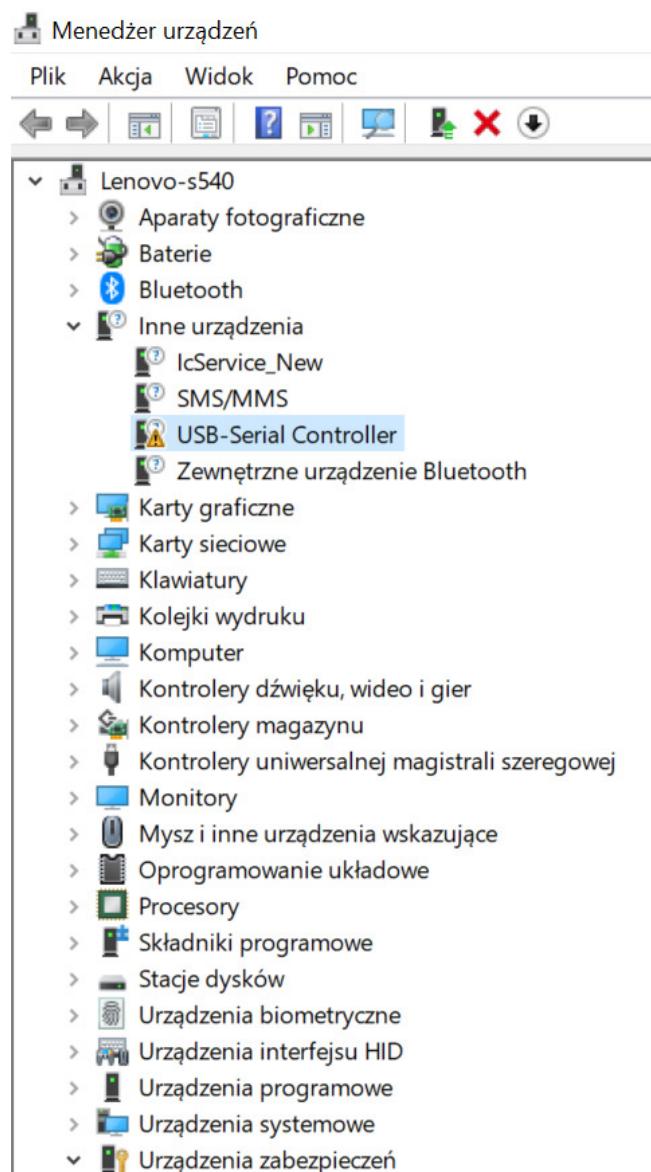
Usługa Windows Update nie zapewnia wsparcia dla użytego adaptera. Wymagane oprogramowanie jest dostępne na stronie producenta.

Proces instalacji jest zautomatyzowany i wymaga uruchomienia instalatora sterownika przy odłączonym urządzeniu, następnie po zakończonej instalacji i podłączeniu urządzenia system Windows powiąże adapter ze sterownikiem i umożliwi świadczenie usługi (rysunek 5.3).

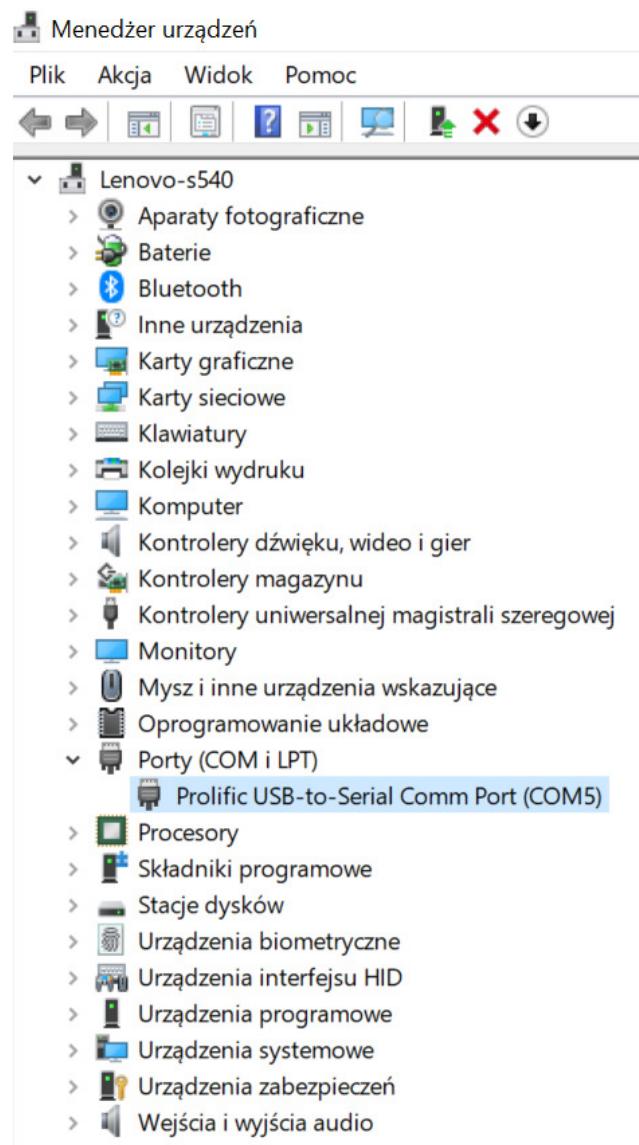


Rysunek 5.1: Adapter USB UART dla połączenia szeregowego.

źródło: <https://www.waveshare.com> [dostęp: 28.05.2021])



Rysunek 5.2: Nierozpoznane urządzenie USB Serial Controller w Menedżerze urządzeń systemu Windows [źródło: własne]



Rysunek 5.3: Rozpoznane urządzenie USB Serial Controller w Menedżerze urządzeń systemu Windows [źródło: własne]

Ustanowienie połączenia szeregowego Aby ustanowić połączenie szeregowego pomiędzy Raspberry Pi i komputerem przenośnym przy wykorzystaniu adaptera USB UART, w pierwszej kolejności należy nastawić pracę adaptera przy napięciu wynoszącym 3.3 Volt. Jest to maksymalna wielkość napięcia jakim operuje RPi przy wykorzystaniu połączenia szeregowego. Dostępne dla adaptera napięcie o wysokości 5 Volt uszkodziłoby moduł RPi. Wybór odpowiedniego napięcia jest możliwy do ustawienia przy użyciu zworki w kolorze żółtym (rysunek 5.1). Kolejnym etapem jest przewodowe połączenie adaptera z odpowiednimi wyprowadzeniami GPIO w RPi (rysunek 5.4). Do połączenia użyto 3 wyprowadzeń z adaptera:

1. wyprowadzenie masy (**GND**)
2. ścieżka wysyłania danych (**TXD**)
3. ścieżka odbioru danych (**RXD**)

Wyprowadzenia **GND** adaptera należy połączyć z **GPIO GDN** na płytce RPi

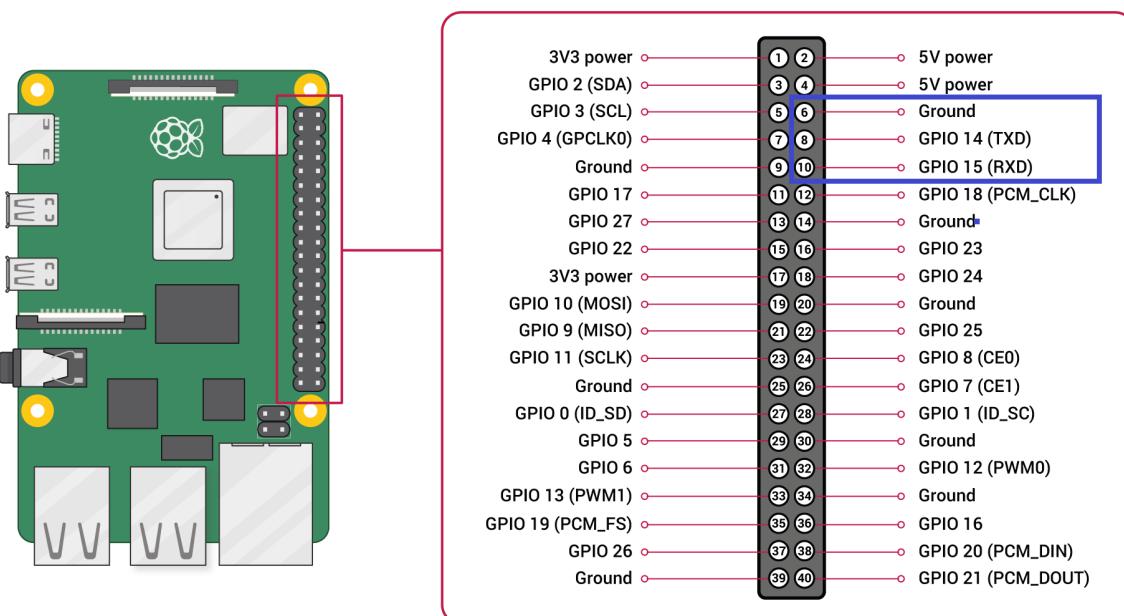
Wyprowadzenia linii sygnałowych należy połączyć naprzemiennie:

Wyprowadzenia **TXD** adaptera należy połączyć z **GPIO RXD** w RPi

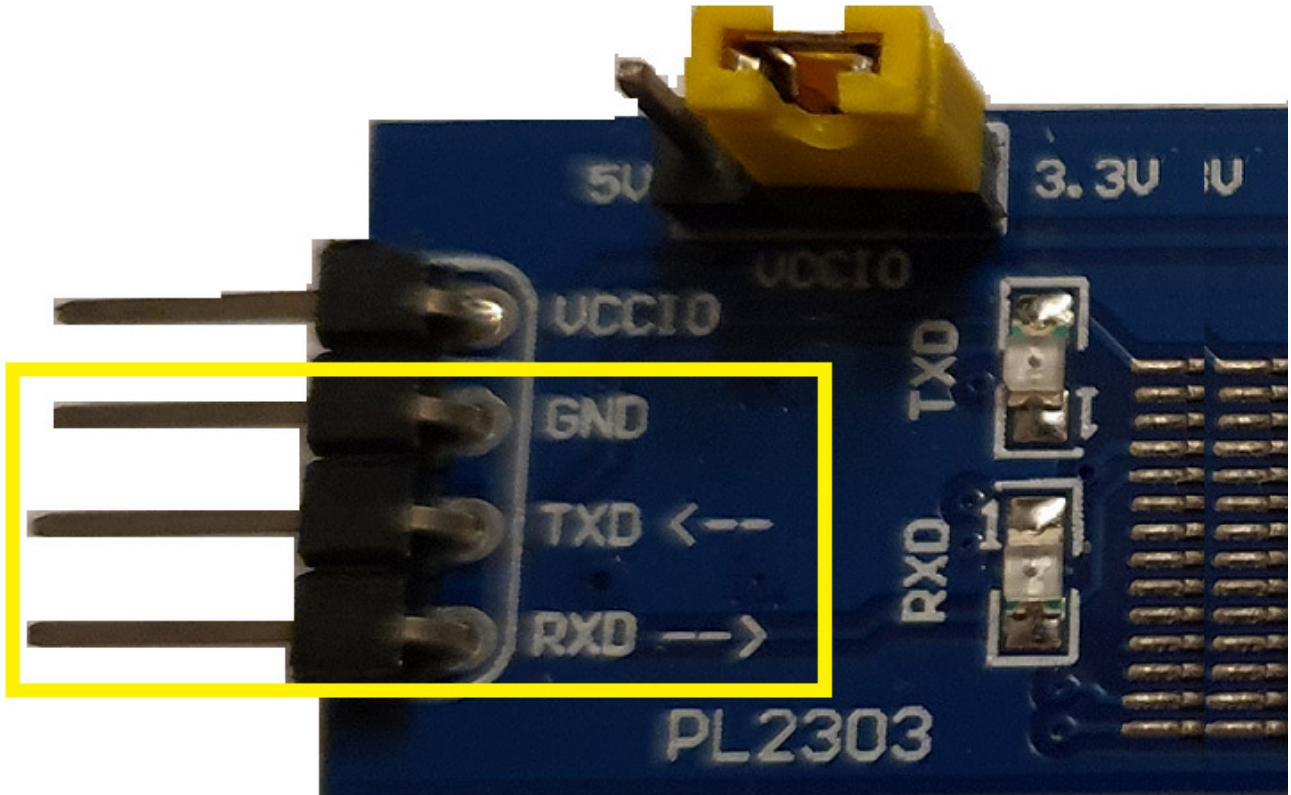
Wyprowadzenia **RXD** adaptera należy połączyć z **GPIO TXD** w RPi

W połączeniu wykorzystano 3 ścieżki sygnałowe z 4 dostępnych. Nie wykorzystano ścieżki VCC, odpowiadającej za stan wysoki sygnału napięcia. Połączony układ jest zasilany z portu USB komputera przenośnego (rysunek 5.5).

Zrealizowane połączenie sprzętowe stanowi podstawę dla konfiguracji programowej w celu ustanowienia połączenia szeregowego pomiędzy laptopem a RPi.



Rysunek 5.4: Specyfikacja GPIO RPi z zaznaczeniem linii dla połączenia szeregowego (źródło: <https://www.raspberrypi.org/documentation/usage/gpio/> [dostęp: 28.05.2021])



Rysunek 5.5: Oznaczenie ścieżek adaptera USB UART z zaznaczeniem ścieżek użytych dla połączenia szeregowego [źródło: własne]

Instalacja systemu Raspbian

Instalacja obrazu systemu Raspbian na karcie microSD W celu instalacji obrazu systemu Raspbian na karcie SD należy pobrać obraz z oficjalnej, dystrybucyjnej strony projektu. Obraz użyty w projekcie jest w wersji jądra: 5.4 i rozmiarze: 438MB. Użyto minimalnej wersji systemu: Lite, aby zwiększyć wydajność działania systemu, nieobciążonego niewymaganyimi pakietami z punktu widzenia projektu systemu oraz aby zwiększyć specjalizację systemu przez indywidualną instalację wymaganych z punktu widzenia projektu pakietów.

Aby zainstalować pobrany obraz systemu RPi użyto programu - "balenaEtcher". W programie należało wybrać obraz systemu jako źródło, umieszczoną w laptopie kartę SD jako cel i uruchomić proces. Nastąpiło auto formatowanie i partycjonowanie poza udziałem użytkownika. W wyniku powyższej operacji otrzymano czystą instalację systemu Raspbian w wersji Lite gotową do pierwszego uruchomienia na platformie RPi.

Konfiguracja systemu Raspbian przed pierwszym uruchomieniem

włączenie obsługi połączenia SSH Aby możliwa była komunikacja szeregowa, pozwolenie na połączenie *Secure Shell* (SSH) powinno być ustanowione po stronie RPi.

W celu realizacji tych założeń, jeszcze przed pierwszym uruchomieniem systemu w RPi, należy w partycji boot na karcie microSD z zainstalowanym systemem Raspbian umieścić plik bez rozszerzenia o nazwie *ssh*.

W systemie Windows 10 Home, należało w pierwszej kolejności odznać opcję *"ukryj rozszerzenia znanych typów plików"* dostępną poprzez menu opcje folderów (rysunek 5.6), takie ustawienie umożliwiło utworzenie pliku o nazwie *ssh*, bez rozszerzenia.

Obecność pliku *ssh* na partycji boot systemu Raspbian dopuszcza połączenia wykorzystujące protokół komunikacyjny *Secure Shell* (rysunek 5.7).

Opcje folderów

X

Ogólne Widok Wyszukiwanie

Widoki folderu



Możesz zastosować ten widok (taki jak Szczegóły lub Ikony) do wszystkich folderów tego typu.

Zastosuj do folderów

Resetuj foldery

Ustawienia zaawansowane:

- Pokaż zaszyfrowane lub skompresowane pliki NTFS w kolo
- Przywrć poprzednie okna folderów po zalogowaniu
- Ukryj chronione pliki systemu operacyjnego (zalecane)
- Ukryj konflikty scalania folderów
- Ukryj puste dyski
- Ukryj rozszerzenia znanych typów plików**
 - Ukryte pliki i foldery**
 - Nie pokazuj ukrytych plików, folderów ani dysków
 - Pokaż ukryte pliki, foldery i dyski
 - Uruchom okna folderów w osobnych procesach
 - Użyj Kreatora udostępniania (zalecane)
 - Użyj pół wyboru do zaznaczania elementów

Rysunek 5.6: Odznaczenie ustawienia "ukryj rozszerzenia znanych typów plików" w opcjach folderów systemu Windows 10 [źródło: własne]

The screenshot shows a Windows File Explorer interface with the following details:

- Toolbar:** Includes buttons for Plik (File), Narzędzia główne (Main Tools), Udostępnianie (Sharing), Widok (View), and various file operations like Wytnij (Cut), Kopiuje ścieżkę (Copy Path), Przenies do (Move to), Kopiuje (Copy), Usuń (Delete), Zmień nazwę (Change Name), Nowy folder (New Folder), Nowy (New), Właściwości (Properties), Otwórz (Open), Edytuj (Edit), and Historia (History).
- Address Bar:** Shows the path: boot (E:).
- File List:** A table showing the following files and their properties:

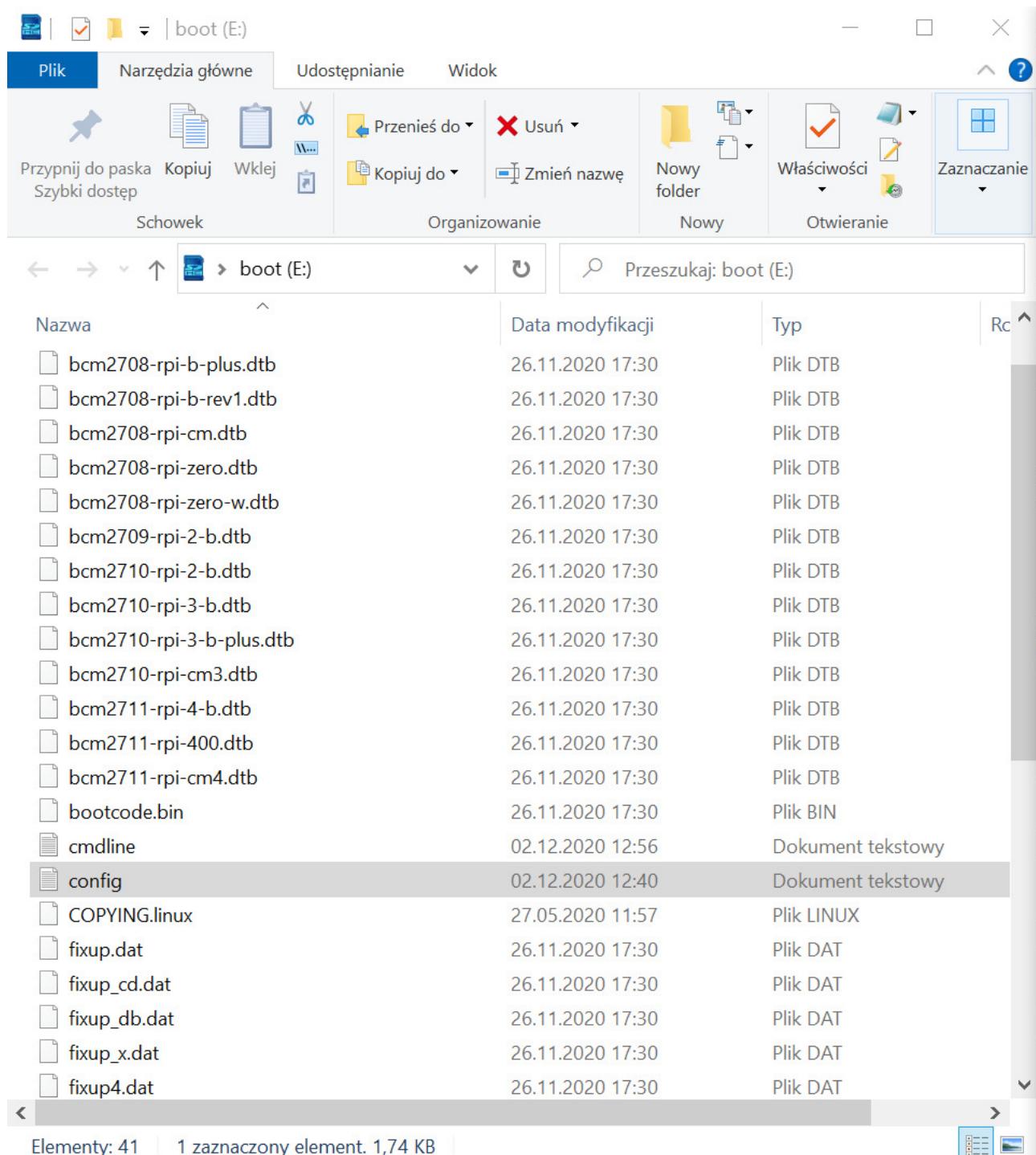
Nazwa	Data modyfikacji	Typ	Rozmiar
fixup.dat	26.11.2020 17:30	Plik DAT	8 KB
fixup_cd.dat	26.11.2020 17:30	Plik DAT	4 KB
fixup_db.dat	26.11.2020 17:30	Plik DAT	11 KB
fixup_x.dat	26.11.2020 17:30	Plik DAT	11 KB
fixup4.dat	26.11.2020 17:30	Plik DAT	6 KB
fixup4cd.dat	26.11.2020 17:30	Plik DAT	4 KB
fixup4db.dat	26.11.2020 17:30	Plik DAT	9 KB
fixup4x.dat	26.11.2020 17:30	Plik DAT	9 KB
issue.txt	02.12.2020 12:56	Dokument tekstowy	1 KB
kernel.img	26.11.2020 17:30	Plik obrazu dysku	5 328 KB
kernel7.img	26.11.2020 17:30	Plik obrazu dysku	5 651 KB
kernel7l.img	26.11.2020 17:30	Plik obrazu dysku	6 002 KB
kernel8.img	26.11.2020 17:30	Plik obrazu dysku	15 527 KB
LICENCE.broadcom	30.09.2020 13:00	Plik BROADCOM	2 KB
start.elf	26.11.2020 17:30	Plik ELF	2 869 KB
start_cd.elf	26.11.2020 17:30	Plik ELF	771 KB
start_db.elf	26.11.2020 17:30	Plik ELF	4 674 KB
start_x.elf	26.11.2020 17:30	Plik ELF	3 610 KB
start4.elf	26.11.2020 17:30	Plik ELF	2 162 KB
start4cd.elf	26.11.2020 17:30	Plik ELF	771 KB
start4db.elf	26.11.2020 17:30	Plik ELF	3 627 KB
start4x.elf	26.11.2020 17:30	Plik ELF	2 904 KB
ssh	05.12.2020 01:36	Plik	0 KB
- Status Bar:** Shows 'Elementy: 42' (Items: 42), '1 zaznaczony element' (1 selected item), and '0 B' (0 B).

Rysunek 5.7: Utworzony plik ssh zezwalający na obsługę protokołu komunikacyjnego ssh [źródło: własne]

Aktywacja interfejsu komunikacyjnego UART

Aby możliwe było połączenie szeregowe pomiędzy laptopem a RPi, w pliku config.txt na partycji boot, należy na samym końcu pliku umieścić wpis: **enable_uart=1**

Ten sposób otwiera się dostęp do asynchronicznej komunikacji nadawczo - odbiorczej wymaganej do ustanowienia połączenia szeregowego.



Rysunek 5.8: Konfiguracja pliku config.txt [źródło: własne]

```
*config — Notatnik
Plik Edycja Format Widok Pomoc
# DMT (computer monitor) modes
#hdmi_drive=2

# uncomment to increase signal to HDMI, if you have interference, blanking, c
# no display
#config_hdmi_boost=4

# uncomment for composite PAL
#sdtv_mode=2

#uncomment to overclock the arm. 700 MHz is the default.
#arm_freq=800

# Uncomment some or all of these to enable the optional hardware interfaces
#dtparam=i2c_arm=on
#dtparam=i2s=on
#dtparam=spi=on

# Uncomment this to enable infrared communication.
#dtoverlay= gpio-ir, gpio_pin=17
#dtoverlay= gpio-ir-tx, gpio_pin=18

# Additional overlays and parameters are documented /boot/overlays/README

# Enable audio (loads snd_bcm2835)
dtparam=audio=on

[pi4]
# Enable DRM VC4 V3D driver on top of the dispmanx display stack
dtoverlay=vc4-fkms-v3d
max_framebuffers=2

[all]
#dtoverlay=vc4-fkms-v3d

enable_uart=1
```

Rysunek 5.9: Dodanie wpisu otwierającego komunikację UART w pliku konfiguracyjnym config.txt [źródło: własne]

5.1.3 Pierwsze uruchomienie systemu Raspbian

Pierwsze uruchomienie wstępnie skonfigurowanego systemu.

Konfiguracja pliku wpa_supplicant.conf

Utworzenie pliku wpa_supplicant.conf umożliwia zautomatyzowanie połączenia RPi z zadeklarowaną siecią WiFi. Połączenie zostaje ustanowione automatycznie, bez udziału użytkownika, w dowolnym momencie, gdy tylko sieć WiFi o zadeklarowanej w skrypcie nazwie SSID i podanym haśle pojawi się w zasięgu wykrywania bezprzewodowej karty sieciowej RPi.

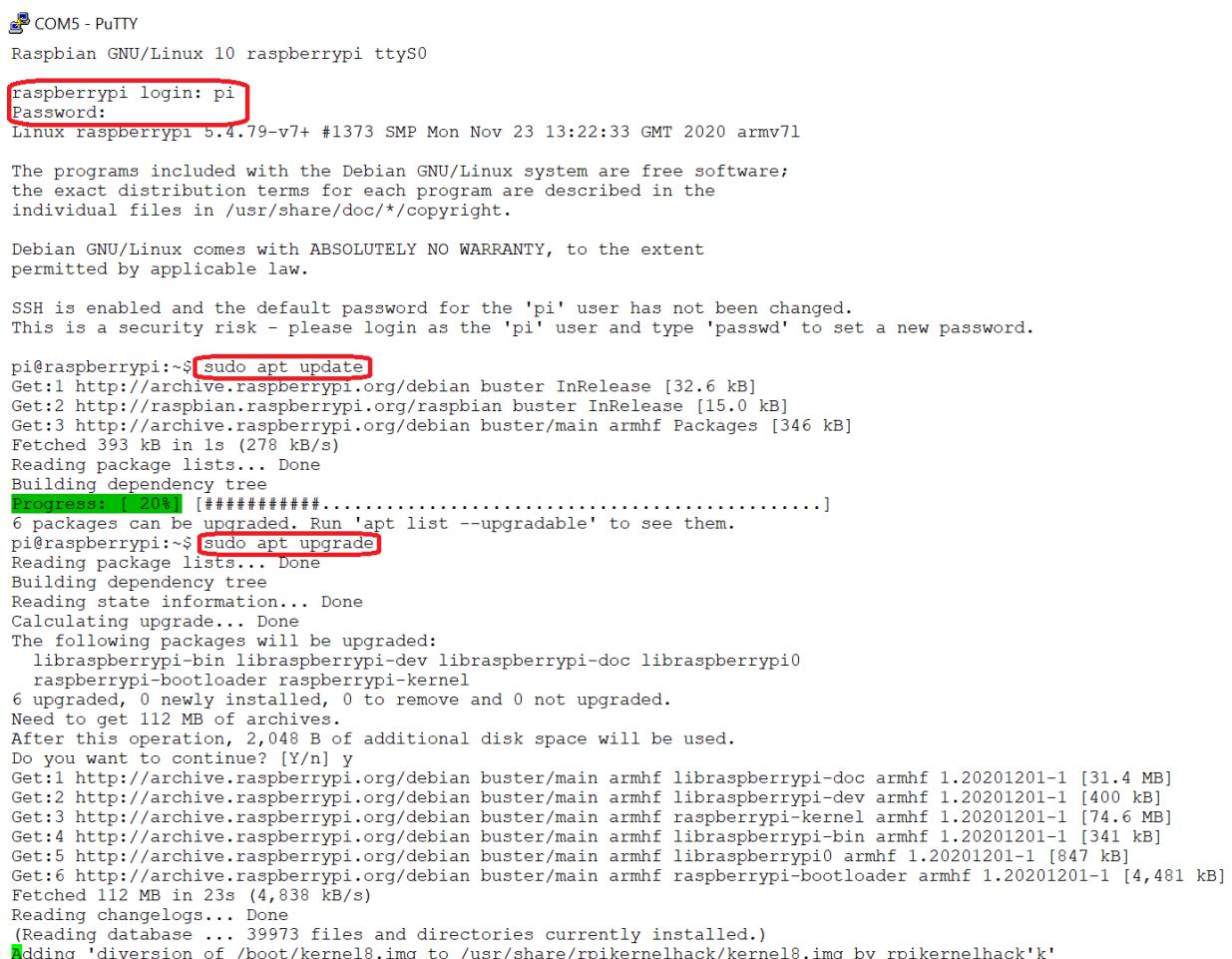
Listing 5.1: plik konfiguracyjny wpa_supplicant.conf

```
country=PL
update_config=1
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev

network={
    ssid="safedrivea"
    psk="safedrivea"
    key_mgmt=WPA-PSK
}
```

Aktualizacja systemu Raspbian

Aby zaktualizować system Raspbian należy zalogować się do systemu przy pomocy domyślnej nazwy użytkownika: **pi** oraz domyślnego hasła: **raspberry**. Następnie należy wpisać polecenie **sudo apt update**, które aktualizuje listę pakietów w repozytoriach systemu, po czym wpisać **sudo apt upgrade**, aby zaktualizować pakiety w systemie.



```
COM5 - PuTTY
Raspbian GNU/Linux 10 raspberrypi ttyS0

raspberrypi login: pi
Password: [REDACTED]
Linux raspberrypi 5.4.79-v7+ #1373 SMP Mon Nov 23 13:22:33 GMT 2020 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.

pi@raspberrypi:~$ sudo apt update
Get:1 http://archive.raspberrypi.org/debian buster InRelease [32.6 kB]
Get:2 http://raspbian.raspberrypi.org/raspbian buster InRelease [15.0 kB]
Get:3 http://archive.raspberrypi.org/debian buster/main armhf Packages [346 kB]
Fetched 393 kB in 1s (278 kB/s)
Reading package lists... Done
Building dependency tree
Progress: [ 20%] [#####]. ....
6 packages can be upgraded. Run 'apt list --upgradable' to see them.
pi@raspberrypi:~$ sudo apt upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
 libraspberrypi-bin libraspberrypi-dev libraspberrypi-doc libraspberrypi0
 raspberryi-bootloader raspberrypi-kernel
6 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 112 MB of archives.
After this operation, 2,048 B of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.raspberrypi.org/debian buster/main armhf libraspberrypi-doc armhf 1.20201201-1 [31.4 MB]
Get:2 http://archive.raspberrypi.org/debian buster/main armhf libraspberrypi-dev armhf 1.20201201-1 [400 kB]
Get:3 http://archive.raspberrypi.org/debian buster/main armhf raspberrypi-kernel armhf 1.20201201-1 [74.6 MB]
Get:4 http://archive.raspberrypi.org/debian buster/main armhf libraspberrypi-bin armhf 1.20201201-1 [341 kB]
Get:5 http://archive.raspberrypi.org/debian buster/main armhf libraspberrypi0 armhf 1.20201201-1 [847 kB]
Get:6 http://archive.raspberrypi.org/debian buster/main armhf raspberrypi-bootloader armhf 1.20201201-1 [4,481 kB]
Fetched 112 MB in 23s (4,838 kB/s)
Reading changelogs... Done
(Reading database ... 39973 files and directories currently installed.)
Adding 'diversion of /boot/kernel8.img to /usr/share/rpikernelhack/kernel8.img by rpikernelhack'!
```

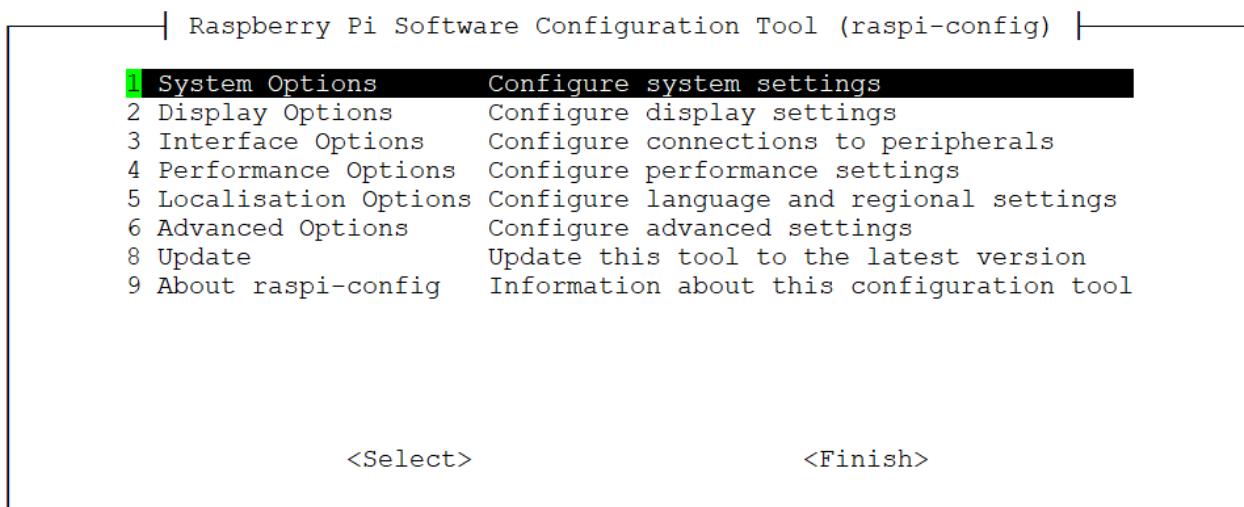
Rysunek 5.10: Aktualizacja pakietów w systemie Raspbian [źródło: własne]

Konfiguracja systemu narzędziem: *raspi-config*

Narzędzie raspi-config uruchamia się jako administrator polecienniem *sudo raspi-config*

 COM5 - PuTTY

Raspberry Pi 3 Model B Plus Rev 1.3



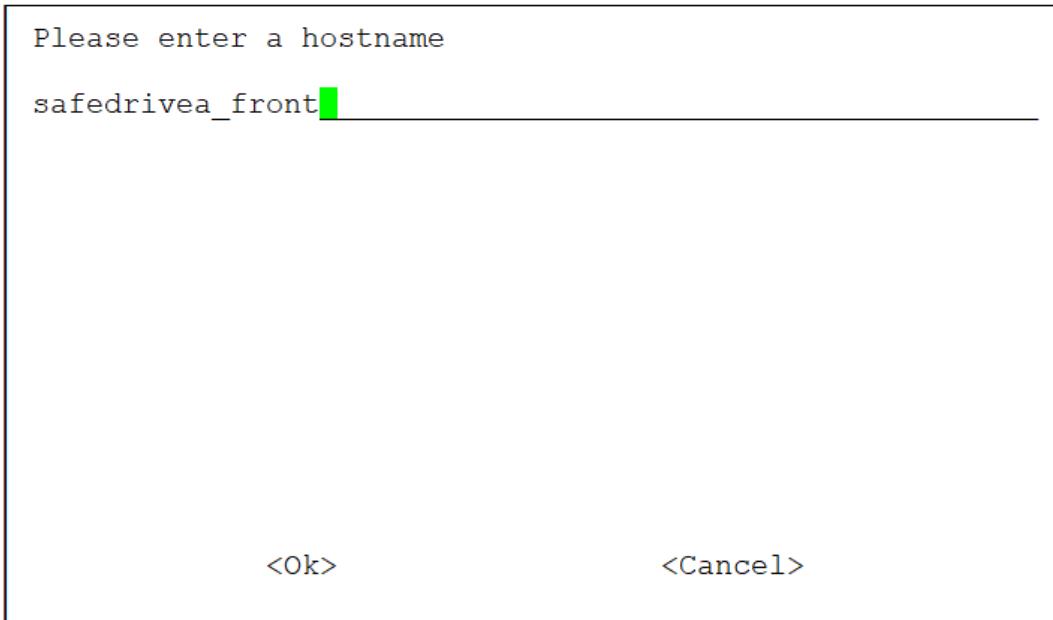
Rysunek 5.11: Menu główne narzędzia raspi-config [źródło: własne]

Narzędzie raspi-config umożliwia podstawowe ustawienia takie jak:

1. Nazwa hosta.

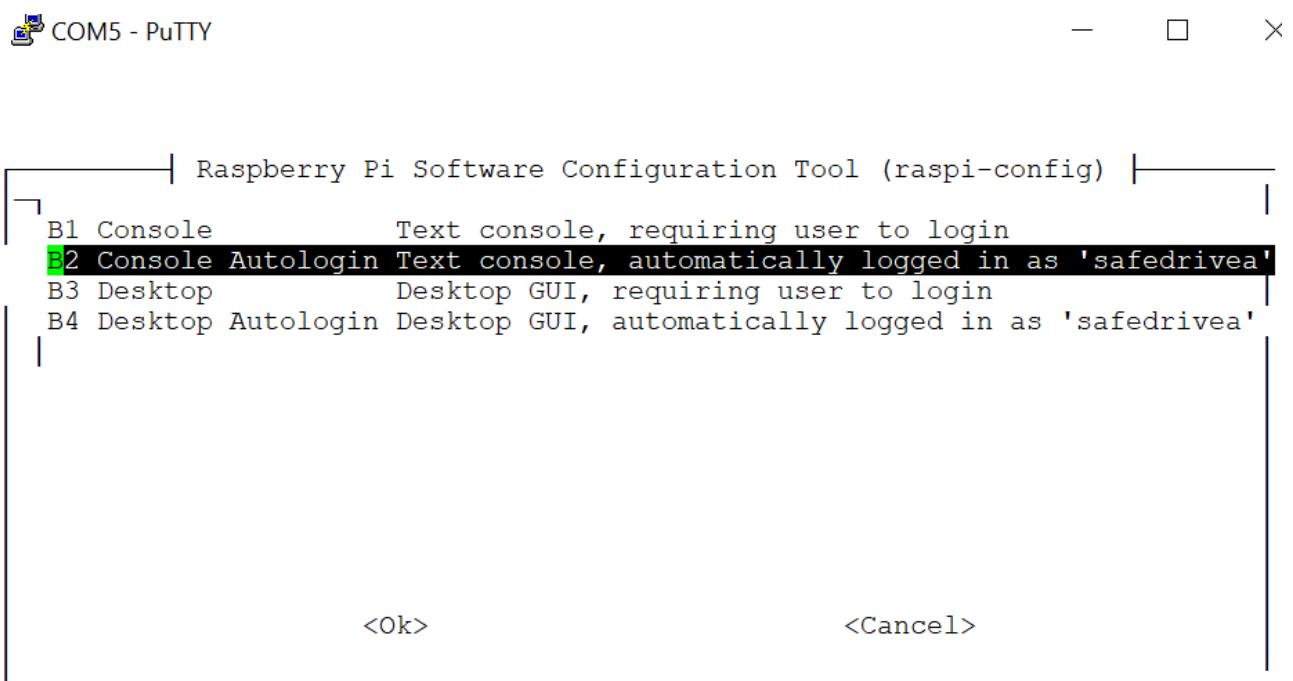
Nazwa hosta *safedriveafront* jest zarezerwowana dla urządzenia RPi działającego z przodu pojazdu. Nazwa *safedriveaback* jest przypisana hostowi działającemu z tyłu pojazdu.

 COM5 - PuTTY



Rysunek 5.12: Ustawienie nazwy hosta w raspi-config [źródło: własne]

- Automatyczne logowanie do terminala albo GUI po uruchomieniu systemu.
Ustawienie automatycznego logowania do terminala zamiast do graficznego środowiska użytkownika pozwala zwiększyć wydajność działania systemu i jego szybsze uruchamianie.



Rysunek 5.13: Ustawienie automatycznego logowania do systemu [źródło: własne]

- Oczekiwanie na ustanowienie połączenia internetowego przed całkowitym uruchomieniem systemu.

```
Would you like boot to wait until a network connection  
is established?
```

<Yes>

<No>

Rysunek 5.14: Wymuszenie połączenia z internetem przed załadowaniem systemu [źródło: własne]

4. Włączenie domyślnego interfejsu kamery.

Kamery do rejestracji ruchu ulicznego użyte w projekcie wykorzystują domyślny interfejs RPi w celu komunikacji, dlatego wymagane jest włączenie interfejsu.

| Raspberry Pi Software Configuration Tool (raspi-config) |

P1 Camera	Enable/disable connection to the Raspberry Pi Camera
P2 SSH	Enable/disable remote command line access using SSH
P3 VNC	Enable/disable graphical remote access using RealVNC
P4 SPI	Enable/disable automatic loading of SPI kernel module
P5 I2C	Enable/disable automatic loading of I2C kernel module
P6 Serial Port	Enable/disable shell messages on the serial connection
P7 1-Wire	Enable/disable one-wire interface
P8 Remote GPIO	Enable/disable remote access to GPIO pins

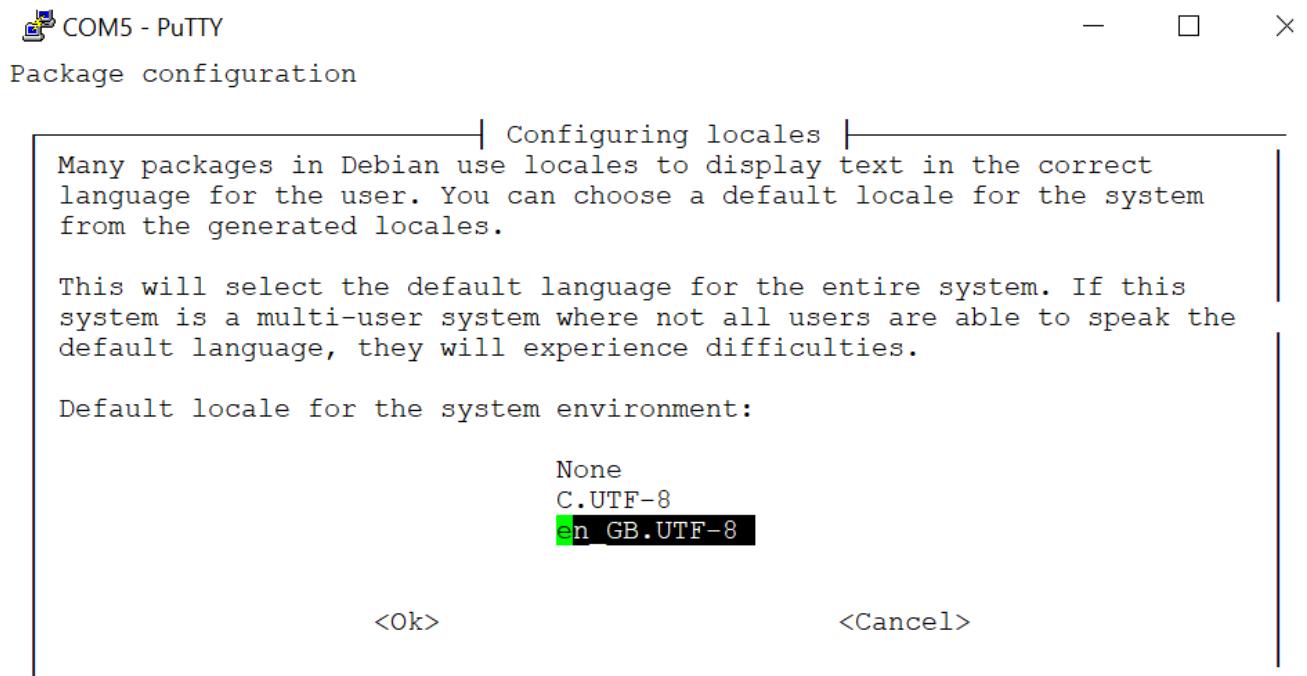
<Select>

<Back>

Rysunek 5.15: Włączenie domyślnego interfejsu kamery w RPi [źródło: własne]

5. Wybór strony kodowej znaków.

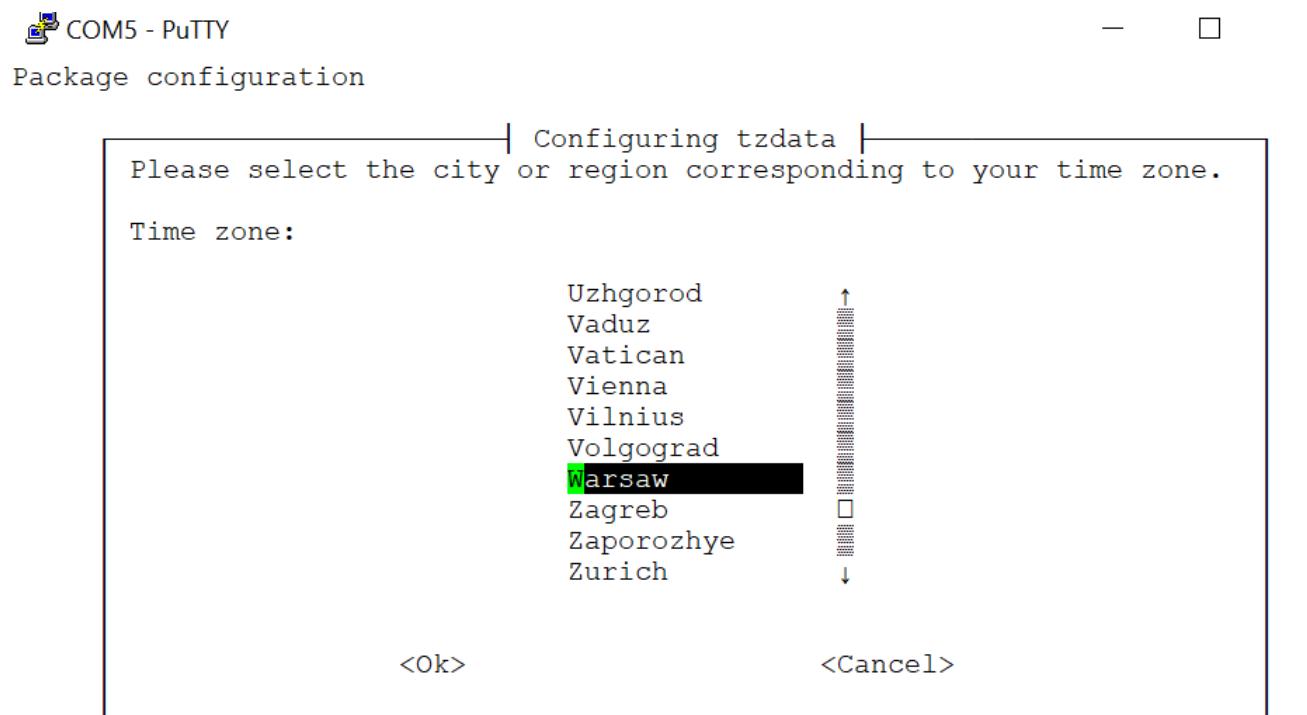
Wybraną stroną kodową znaków dla RPi jest zgodna z ASCII - system kodowania Unicode UTF-8.



Rysunek 5.16: Wybór strony kodowej znaków dla RPi [źródło: własne]

6. Wybór strefy czasowej

Wybrano lokalną dla Polski strefę czasową ze względu na zgodną synchronizację urządzeń z pozostałymi, działającymi lokalnie modułami systemu, szczególnie z ogólnie dostępnego w internecie API dla lokalnych warunków pogodowych.



Rysunek 5.17: Wybór strefy czasowej w RPi [źródło: własne]

Zmiana domyślnej nazwy, grupy i hasła użytkownika

W celu utworzenia nowej nazwy, grupy oraz hasła użytkownika posłużyono się kreatorem w powłoce BASH, wywołując kreator poleceniem: *sudo adduser*.

```
root@safedriveafront:~# users
root
root@safedriveafront:~# sudo adduser safedriveafront
Adding user `safedriveafront' ...
Adding new group `safedriveafront' (1001) ...
Adding new user `safedriveafront' (1001) with group `safedriveafront' ...
Creating home directory `/home/safedriveafront' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for safedriveafront
Enter the new value, or press ENTER for the default
    Full Name []: safedriveafront
    Room Number []: 0
    Work Phone []: 0
    Home Phone []: 0
    Other []: None
Is the information correct? [Y/n] y
root@safedriveafront:~# logout

Raspbian GNU/Linux 10 safedriveafront ttyS0

safedriveafront login: safedriveafront
Password:
Linux safedriveafront 5.4.79-v7+ #1373 SMP Mon Nov 23 13:22:33 GMT 2020 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
safedriveafront@safedriveafront:~$ █
```

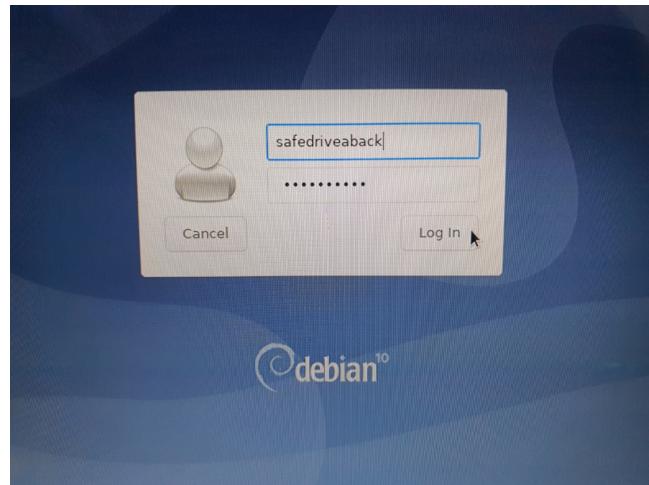
Rysunek 5.18: Utworzenie nazwy, grupy i hasła użytkownika [źródło: własne]

5.1.4 Problem z automatycznym logowaniem do systemu

W wyniku uruchomienia systemu Raspbian, po procesie wstępnej konfiguracji z ustawieniem automatycznego logowania użytkownika do tekstowego terminala systemu, nie następuje proces automatycznego logowania po starcie systemu.

System wymaga manualnego logowania poprzez ustaloną nazwę użytkownika i hasło. Takie podejście uniemożliwia zautomatyzowanie procesu użytkowania minikomputera jako systemu wbudowanego. Dodatkowo system blokuje możliwość uruchamiania planowanych wg założeń projektowych skryptów autostartu do czasu manualnego zalogowania się użytkownika do systemu.

Kolejnym problemem okazał się brak możliwości zalogowania do systemu poprzez graficzny interfejs użytkownika. Podczas próby uwierzytelnienia użytkownika za pomocą systemowego formularza logowania, formularz pojawiał się ponownie z pustymi polami tekstowymi, wymagając w nieskończonej pętli procesu autoryzacji.



Rysunek 5.19: Brak możliwości zalogowania z poziomu GUI [źródło: własne]

Możliwe okazało się logowanie do systemu poprzez interfejs tekstowy. W oparciu o analizę problemu, szukając rozwiązania zagadnienia wśród internetowej społeczności użytkowników dystrybucji Debian, posiadając dostęp do poleceń terminala systemu, postanowiono nadać uprawnienia do pliku: `.Xauthority`, użytkownikowi: `safedriveaback`, należącego do grupy: `safedriveaback` standardowym dla systemu Linux poleceniem: `chown`.

```

[ OK ] Started WPA supplicant.
[ OK ] Started dphys-swapfile - set up, mount/unmount, and delete a swap file.
[ OK ] Started LSB: Switch to ondemand cpu governor (unless shift key is pressed).
[ OK ] Started Configure Bluetooth Modems connected by UART.
[ OK ] Started Load/Save RF Kill Switch Status...
Starting Load/Save RF Kill Switch Status.
[ OK ] Created slice system-bthelper.slice.
Starting Bluetooth service...
[ OK ] Started Bluetooth service.
[ OK ] Started Raspberry Pi bluetooth helper.
[ OK ] Reached target Bluetooth.
Starting Hostname Service...
[ OK ] Started Hostname Service.
[ OK ] Started dhpcd on all interfaces.
[ OK ] Reached target Network.
Starting /etc/rc.local Compatibility...
Starting Permit User Sessions...
Starting OpenBSD Secure Shell server...
My IP address is 192.168.43.134
[ OK ] Started /etc/rc.local Compatibility.
[ OK ] Started Permit User Sessions.
Starting Hold until boot process finishes up...
Starting Light Display Manager...
[ OK ] Started OpenBSD Secure Shell server.

Raspbian GNU/Linux 10 safedriveaback tty1

safedriveaback login:
Password:
Last login: Sun Jan 31 15:40:11 CET 2021 on tty1
Linux safedriveaback 5.4.83-v7+ #1379 SMP Mon Dec 14 13:08:57 GMT 2020 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
safedriveaback@safedriveaback:~ $ sudo chown safedriveaback:safedriveaback .Xauthority

```

Rysunek 5.20: Próba przypisania użytkownikowi uprawnień do pliku .Xauthority [źródło: własne]

Plik .Xauthority znajduje się w katalogu domowym każdego użytkownika zarejestrowanego w systemie Linux i jest używany aby przechowywać uprawnienia do uruchomienia przez użytkownika sesji graficznego interfejsu użytkownika za pośrednictwem systemu okien X window, będącego graficznym komponentem systemu. W przypadku gdy zalogowany użytkownik nie jest właścicielem pliku .Xauthority, nie jest możliwe ustalenie uprawnień do uruchomienia graficznego interfejsu użytkownika.

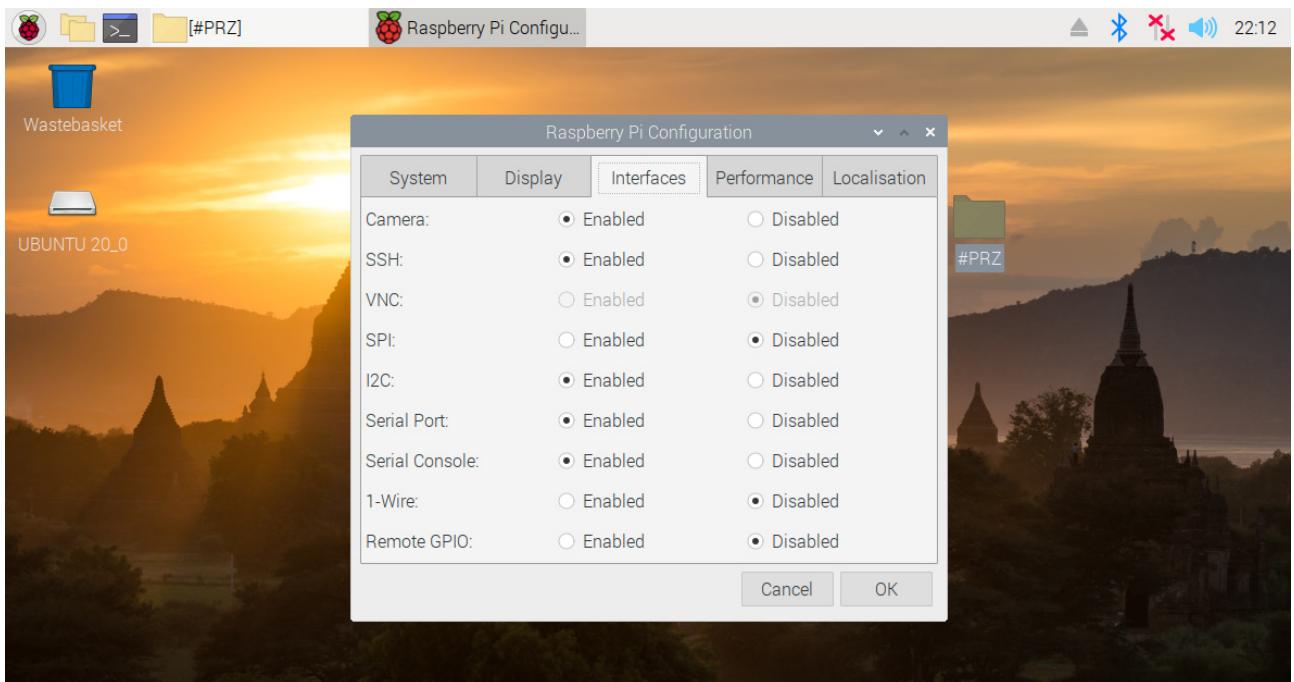
Powyzsze wnioski były przesłanką do nadania użytkowi uprawnień do pliku .Xauthority w celu zalogowania do GUI systemu. Zastosowane podejście okazało się nieskuteczne.

W wyniku zastosowania nieskutecznej metody i z powodu ograniczeń czasowych oraz po procesie analizy sugerowanych rozwiązań wśród społeczności użytkowników, postanowiono zmienić środowisko graficzne domyślnie zainstalowane z systemem Raspbian. Wybrano środowisko graficzne Pixel. W celu instalacji z poziomu linii komend sprowadzono polecenie: *sudo apt install raspberrypi-ui-mods*, następnie po procesie instalacji uruchomiono ponownie system.

Wykorzystując GUI środowiska Pixel możliwe było zalogowanie do systemu z poziomu graficznego interfejsu użytkownika. Dostępna była opcja automatycznego logowania użytkownika do systemu, zarówno do tekstopowego jak i graficznego interfejsu.

Napotkany problem automatycznego logowania został rozwiązany.

W celu sprawdzenia poprawności poprzedniej konfiguracji uruchmiono okno programu Raspberry Pi Configuration.



Rysunek 5.21: Okno podstawowej konfiguracji systemu Raspberry Pi [źródło: własne]

Uruchomione były następujące procesy:

1. Interfejs CSI kamery
2. Protokół komunikacyjny SSH
3. interfejs szeregowy I2C
4. Port szeregowy
5. Dostęp do konsoli przez port szeregowy

Pierwsza konfiguracja minikomputera Raspberry Pi przebiegła pomyślnie.

Procedurę konfiguracyjną powtórzono dla drugiego minikomputera Raspberry Pi z nawą użytkownika: safedriveafront w celu rozróżnienia obu systemów wbudowanych.

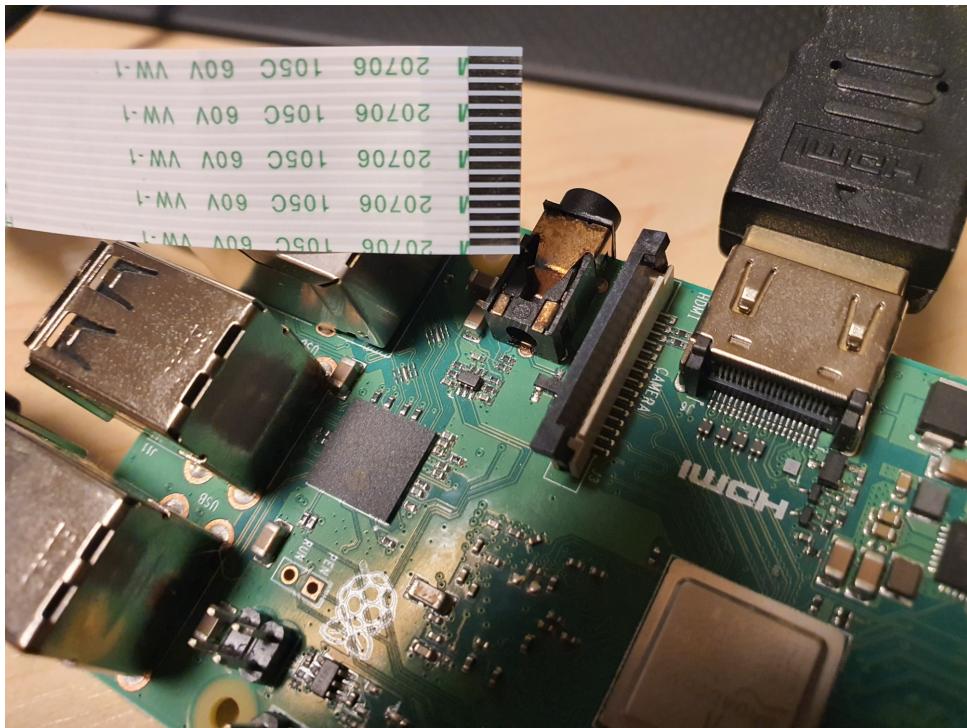
5.2 Podłączenie kamery

Moduł implementacji systemu kamery stanowi potencjalne rozszerzenie funkcjonalności dla aplikacji mobilnej obsługującej użytkę platformę Raspberry Pi i możliwe jest jego wykorzystanie w przyszłości, w celu rozwinięcia zakresu projektu.

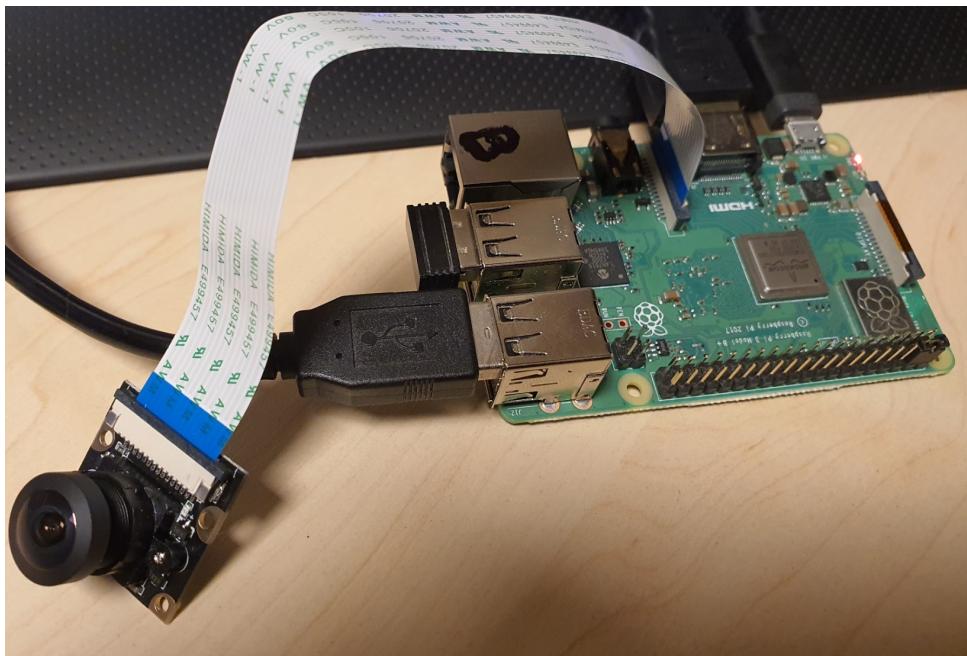
5.2.1 Instalacja kamery

Kamerę przygotowano do połączenia z minikomputerem mocując do kamery dwie diody IR, zapewniające lepszą widoczność rejestrowanego obrazu nocą.

Dostępną kamerę należało skomunikować z minikomputerem Raspberry Pi poprzez interfejs CSI. Interfejs CSI wymagał wcześniejszej aktywacji w pierwszym etapie konfiguracji minikomputera. Komunikacja podzespołów nastąpiła poprzez taśmę połączeniową i fabrycznie zainstalowany interfejs na płycie Raspberry Pi i w module sprzętowym kamery.



Rysunek 5.22: Interfejs CSI i taśma połączeniowa [źródło: własne]

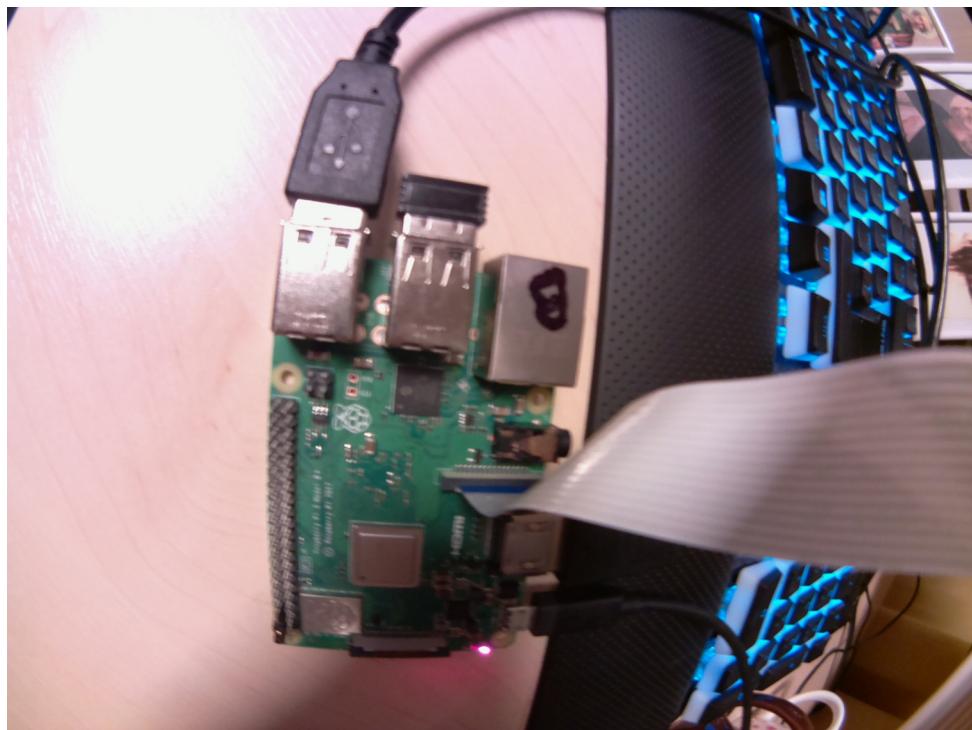


Rysunek 5.23: Połączenie między Raspberry Pi a kamerą [źródło: własne]

Zastosowana kamera jest fabrycznie kompatybilna i dedykowana dla środowiska modułów do Raspberry Pi. Producent kamery wydał oprogramowanie do obsługi kamery o nazwie: Motion, dostępne ze standardowych repozytoriów Linuxa.

W celu poprawnego działania modułu w systemie wymagana jest instalacja dodatkowego modułu jądra systemu polecением: `sudo modprobe bcm2835-v4l2`.

W celu instalacji biblioteki Motion należało wydać polecenie: `sudo apt-get install motion`. Uruchomienie zainstalowanego oprogramowania poleciением: `motion` tworzy instancję serwera udostępniającego zasób aktualnego widoku z kamery, dostępny z poziomu przeglądarki webowej pod adresem localhosta: 127.0.0.1



Rysunek 5.24: Widok z kamery przez przeglądarkę webową [źródło: własne]

5.2.2 Konfiguracja kamery

Zestawione połączenie z kamerą w domyślnej konfiguracji miało zbyt niską wydajność, powodując przycinanie się obrazu skutkiem zbyt niskiego wskaźnika klatek na sekundę FPS. Uruchomiono edycje pliku konfiguracyjnego: motion.conf dostępnego w lokalizacji: /etc/motion, w celu optymalizacji ustawień parametrów uzyskując jak najlepszą płynność i jakość obrazu.

```
safedriveaback@safedriveaback: ~
File Edit Tabs Help
GNU nano 3.2                               /etc/motion/motion.conf

# Rename this distribution example file to motion.conf
#
# This config file was generated by motion 4.1.1
# Documentation: /usr/share/doc/motion/motion_guide.html
#####
# Daemon
#####

# Start in daemon (background) mode and release terminal (default: off)
daemon off

# File to store the process ID, also called pid file. (default: not defined)
process_id_file /var/run/motion/motion.pid

#####
# Basic Setup Mode
#####

# Start in Setup-Mode, daemon disabled. (default: off)
setup_mode off

# Use a file to save logs messages, if not defined stderr and syslog is used. (default: not defined)
logfile /var/log/motion/motion.log

# Level of log messages [1..9] (EMG, ALR, CRT, ERR, WRN, NTC, INF, DBG, ALL). (default: 6 / NTC)
[ Read 740 lines ]
^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos      M-U Undo      M-A Mark Text   M-] To Bracket
^X Exit         ^R Read File    ^R Replace     ^U Uncut Text   ^T To Spell     ^G Go To Line   M-E Redo      M-0 Copy Text   M-[ Where Was
[safedriveaback...]
[ Pictures ]
```

Rysunek 5.25: Edycja pliku konfiguracyjnego motion.conf [źródło: własne]

Analizując różne wersje konfiguracji udostępnione przez społeczność deweloperów systemu Raspbian, zastosowano poniższe ustawienia:

1. stream_localhost off
 2. webcontrol_localhot off
 3. framerate 100
 4. stream_maxrate 10
 5. stream_port 8081
 6. substream_port 8081

W pliku: motion, w lokalizacji: /etc/default, zastosowano wpis: start_motion_daemon=yes start_motion_daemon=yes umożliwia uruchomienia usługi w tle, każdorazowo podczas startu systemu.

Struktura pliku motion.conf po konfiguracji jest następująca:

Listing 5.2: plik konfiguracyjny motion.conf

```
# Rename this distribution example file to motion.conf
#
# This config file was generated by motion 4.1.1
# Documentation: /usr/share/doc/motion/motion-guide.html

daemon on
process_id_file /var/run/motion/motion.pid

#####
# Basic Setup Mode
#####

setup_mode off
logfile /var/log/motion/motion.log
log_level 6
log_type all

#####
# Capture device options
#####

videodevice /dev/video0
v4l2_palette 17
; tunerdevice /dev/tuner0
input -1
norm 0
frequency 0
power_line_frequency -1
rotate 0
flip_axis none
width 640
height 480
framerate 1500
minimum_frame_time 0
; netcam_url value
; netcam_userpass value
netcam_keepalive off
; netcam_proxy value
netcam_tolerant_check off
rtsp_uses_tcp on
; mmalcam_name vc.ril.camera
; mmalcam_control_params -hf
auto_brightness off
brightness 0
contrast 0
saturation 0
hue 0

#####
# Round Robin (multiple inputs on same video device name)
#####

roundrobin_frames 1
roundrobin_skip 1
switchfilter off

#####
# Motion Detection Settings:
#####

threshold 1500
threshold_tune off
noise_level 32
noise_tune on
despeckle_filter EedDl
; area_detect value
; mask_file value
smart_mask_speed 0
lightswitch 0
minimum_motion_frames 1
pre_capture 0
post_capture 5
event_gap 60
max_movie_time 0
emulate_motion off

#####
# Image File Output
#####

output_pictures off
output_debug_pictures off
quality 100
picture_type jpeg

#####

ffmpeg_output_movies on
ffmpeg_output_debug_movies off
ffmpeg_bps 400000
ffmpeg_variable_bitrate 0
ffmpeg_video_codec mkv
ffmpeg_duplicate_frames true
timelapse_interval 0
timelapse_mode daily
timelapse_fps 30
timelapse_codec mpg
```

```

#####
# External pipe to video encoder
# Replacement for FFMPEG builtin encoder for ffmpeg_output_movies only.
# The options movie_filename and timelapse_filename are also used
# by the ffmpeg feature
#####

use_expipe off
;expipe mencoder --demuxer rawvideo --rawvideo w=%w:h=%h:i420 --ovc x264 --x264encopts bframes=4:frameref=1:subq=1:scenecut=-
avi -o %f.avi - -fps %fps
;expipe x264 - -input-res %wx%h --fps %fps --bitrate 2000 --preset ultrafast --quiet -o %f.mp4
;expipe mencoder --demuxer rawvideo --rawvideo w=%w:h=%h:fps=%fps --ovc x264 --x264encopts preset=ultrafast -of lavf -o %f.mp4
;expipe ffmpeg -y -f rawvideo -pix_fmt yuv420p -video_size %wx%h -framerate %fps -i pipe:0 -vcodec libx264 -preset ultrafa

#####
# Snapshots (Traditional Periodic Webcam File Output)
#####

snapshot_interval 0

#####
# Text Display
# %Y = year, %n = month, %d = date,
# %H = hour, %M = minute, %S = second, %T = HH:MM:SS,
# %v = event, %q = frame number, %t = camera id number,
# %D = changed pixels, %N = noise level, \n = new line,
# %i and %J = width and height of motion area,
# %K and %L = X and Y coordinates of motion center
# %C = value defined by text_event do not use with text_event!
# You can put quotation marks around the text to allow
# leading spaces
#####

locate_motion_mode off
locate_motion_style box
text_right %Y-%n-%d\n%T-%q
; text_left CAMERA %t
text_changes off
text_event %Y%a%d%H%M%S
text_double off
; exif_text %i%J/%K%L

#####
# Target Directories and filenames For Images And Films
# For the options snapshot_, picture_, movie_ and timelapse_filename
# you can use conversion specifiers
# %Y = year, %n = month, %d = date,
# %H = hour, %M = minute, %S = second,
# %v = event, %q = frame number, %t = camera id number,
# %D = changed pixels, %N = noise level,
# %i and %J = width and height of motion area,
# %K and %L = X and Y coordinates of motion center
# %C = value defined by text_event
# Quotation marks round string are allowed.
#####

target_dir /var/lib/motion
snapshot_filename %v-%Y%a%d%H%M%S-snapshot
picture_filename %v-%Y%a%d%H%M%S-%q
movie_filename %v-%Y%a%d%H%M%S
timelapse_filename %Y%a%d-timelapse

#####
# Global Network Options
#####

ipv6_enabled off

#####
# Live Stream Server
#####

# The mini-http server listens to this port for requests (default: 0 = disabled)
stream_port 8081

# 50% scaled down substream (default: 0 = disabled)
# substream_port 8082

# Quality of the jpeg (in percent) images produced (default: 50)
stream_quality 100

# Output frames at 1 fps when no motion is detected and increase to the
# rate given by stream_maxrate when motion is detected (default: off)
stream_motion off

# Maximum framerate for stream streams (default: 1)
stream_maxrate 100

# Restrict stream connections to localhost only (default: on)
stream_localhost off

# Limits the number of images per connection (default: 0 = unlimited)
# Number can be defined by multiplying actual stream rate by desired number of seconds
# Actual stream rate is the smallest of the numbers framerate and stream_maxrate
stream_limit 0

# Set the authentication method (default: 0)
# 0 = disabled
# 1 = Basic authentication
# 2 = MD5 digest (the safer authentication)
stream_auth_method 0

```

```

# Authentication for the stream. Syntax username:password
# Default: not defined (Disabled)
; stream_authentication username:password

# Percentage to scale the stream image for preview
# This is scaled on the browser side, motion will keep sending full frames
# Default: 25
; stream_preview_scale 25

# Have stream preview image start on a new line
# Default: no
; stream_preview_newline no

#####
# HTTP Based Control
#####

# TCP/IP port for the http server to listen on (default: 0 = disabled)
webcontrol_port 8080

# Restrict control connections to localhost only (default: on)
webcontrol_localhost off

# Output for http server, select off to choose raw text plain (default: on)
webcontrol_html_output on

# Authentication for the http based control. Syntax username:password
# Default: not defined (Disabled)
; webcontrol_authentication username:password

# Parameters to include on webcontrol. 0=none, 1=limited, 2=advanced, 3=restricted
# Default: 0 (none)
webcontrol_parms 0

#####
# Tracking (Pan/Tilt)
#####

track_type 0
track_auto off
;track_port /dev/ttyS0
;track_motorx 0
;track_motorx_reverse 0
;track_motory 1
;track_motory_reverse 0
;track_maxx 200
;track_minx 50
;track_maxy 200
;track_miny 50
;track_homex 128
;track_homey 128
track_iomodo_id 0
track_step_angle_x 10
track_step_angle_y 10
track_move_wait 10
track_speed 255
track_stepsize 40

quiet on
; on_event_start value
; on_event_end value
; on_picture_save value
; on_motion_detected value
; on_area_detected value
; on_movie_start value
; on_movie_end value
; on_camera_lost value
; on_camera_found value

#####
# Common Options for database features.
# Options require database options to be active also.
#####

; sql_log_picture on
; sql_log_snapshot on
; sql_log_movie off
; sql_log_timelapse off
event_time_stamp timestamp without time zone;
; sql_query insert into security(camera, filename, frame, file_type, time_stamp, event_time_stamp) values('%t', '%f', '%q', '%T')

#####
# Database Options
#####

; database_type value
; database_dbname value
; database_host value
; database_user value
; database_password value
; database_port value
; database_busy_timeout 0

#####
# Video Loopback Device (vloopback project)
#####

; video_pipe value
; motion_video_pipe value

```

```
#####
# camera config files - One for each camera.
# Except if only one camera - You only need this config file.
# If you have more than one camera you MUST define one camera
# config file for each camera in addition to this config file.
#####

; camera /etc/motion/camera1.conf
; camera /etc/motion/camera2.conf
; camera /etc/motion/camera3.conf
; camera /etc/motion/camera4.conf

#####
# Camera config directory
# Any files ending in '.conf' in this directory will be read
# as a camera config file.
#####

; camera_dir /etc/motion/conf.d
```

Modyfikacja konfiguracji zapewniała optymalny stosunek jakości do wydajności obrazu.

W lokalizacji /etc/motion/ w plikach konfiguracyjnych:

1. camera1-dist.conf
2. camera2-dist.conf
3. camera3-dist.conf
4. camera4-dist.conf

możliwa jest zmiana portu dla adresu localhosta, na którym nasłuchuje serwer udostępniający zasób obrazu z kamery. Używane wartości to kolejno: 8081, 8082, 8083, 8084.

W celu uruchomienia skonfigurowanej usługi należy wydać polecenie *sudo systemctl enable motion*

5.2.3 Instalacja chłodzenia

W wyniku instalacji i uruchomienia kamery, podczas zwykłego użytkowania systemu przy zadanych parametrach konfiguracyjnych, zauważono silne nagrzewanie się sekcji mikrokontrolera w minikomputerze Raspberry Pi.

Postanowiono zamontować radiator na sekcję mikroprocesora.



Rysunek 5.26: Instalacja radiatora na sekcji mikroprocesora i kontrolera sieci LAN [źródło: własne]

Rozwiążanie zmniejszy emisję ciepła przez układ poprzez rozpraszanie ciepła na ożebrowaniu radiatora.

5.3 Podłączenie skanera obiektów

Moduł implementacji systemu skanera obiektów stanowi potencjalne rozszerzenie funkcjonalności dla aplikacji mobilnej obsługującej użytką platformę Raspberry Pi i możliwe jest jego wykorzystanie w przyszłości, w celu rozwinięcia zakresu projektu.

5.3.1 Zasadność zastosowania w projekcie

Skaner obiektów w przestrzeni stanowi czujnik dostarczający informacji o znajdujących się w skanowanej przestrzeni obiektach, o określonej w tej przestrzeni pozycji i odległości od czujnika. Z punktu widzenia wymagań projektowych, zastosowanie skanera umożliwia obserwację w czasie rzeczywistym pojazdów znajdujących się za kierowcą motocykla. Rejestrowane dane o odległości i kształtach poruszających się z tyłu kierującego pojazdach, w danej chwili i w trybie symulacji rzeczywistego toru ruchu rejestrów pojazdów, umożliwia opracowanie funkcjonalności, która w reakcji za zbyt bliską odległość poruszających się z tyłu pojazdów, uruchomi system ostrzegający kierującego o zagrożeniu bezpieczeństwa.

Drugą funkcjonalnością zastosowanego rozwiązania jest uruchomienie prewencyjnego systemu bezpieczeństwa w sytuacji zbyt dużej prędkości nadjeżdżających z tyłu pojazdów w odniesieniu do prowadzącego motocykl użytkownika systemu.

W wyniku realizacji opisanych wymagań projektowych, wybór jako rozwiązania problemu przestrzennego skanera odległości staje się zasadne.

5.3.2 Pierwsze uruchomienie skanera

W celu uruchomienia laserowego skanera przestrzeni wykorzystano:

1. Adapter interfejsu komunikacji Slamtec STC-A0317 z gniazdem micro USB dla zestawienia połączenia z portem USB w minikomputerze Raspberry Pi.
2. Przewód z micro USB do USB.
3. Domyślne oprogramowanie producenta: Dave Astels dla firmy Adafruit z licencją MIT.

Uruchomienie skanera przestrzeni 2D jest sterowane oprogramowaniem zakodowanym w języku Python. W systemie Raaspbian wymagana była instalacja interpretera kodu języka Python przez wykonanie polecenia: `sudo apt-get install python3`.

W celu instalacji dodatkowych modułów dostępnych dla języka Python wymagana była instalacja menadżera pakietów: pip, wykonując polecenie `sudo apt-get install python3-pip`

Dostarczone przez producenta oprogramowanie wymagało doinstalowania biblioteki Pygame przy użyciu menadżera pakietów pip dla środowiska języka Python, posługując się poleceniem `python3 pip install pygame`.

Dodatkowy moduł, na podstawie którego opiera się działanie dostarczonego przez firmę Adafruit oprogramowania skanera wymagał instalacji poleceniem: `sudo python3 pip install adafruit-rplidar`.

Wymogiem uruchomienia kodu producenta była instalacja pakietu SDL poleceniem: `sudo apt-get install libsdl2-2.0-0` oraz instalacja modułu Pyserial poleceniem: `sudo python3 pip install pyserial`.

Kolejnym krokiem była kontrola dwóch zasobów w systemie Raspbian po podłączeniu skanera obiektów poprzez adapter. Następujące lokalizacje musiały istnieć w systemie: `/dev/fb0` oraz `/dev/ttyUSB0`. Do podanych lokalizacji istnieją odniesienia w kodzie źródłowym producenta. Dostarczone oprogramowanie po nakreślonej konfiguracji środowiska należało uruchomić poleceniem: `sudo python3 ./display_lidar_pi.py`.

Wynikiem działania domyślnego oprogramowania jest okienko systemowe z widokiem skanowanych w przestrzeni obiektów



Rysunek 5.27: Wynik działania skanera obiektów w przestrzeni [źródło: własne]

5.3.3 Oprogramowanie producenta

Listing 5.3: skrypt producenta dla skanera przestrzeni RPLidar:display_lidar_pi.py

```
"""
Consume LIDAR measurement file and create an image for display.

Adafruit invests time and resources providing this open source code.
Please support Adafruit and open source hardware by purchasing
products from Adafruit!

Written by Dave Astels for Adafruit Industries
Copyright (c) 2019 Adafruit Industries
Licensed under the MIT license.

All text above must be included in any redistribution.
"""

import os
from math import cos, sin, pi, floor
import pygame
from adafruit_rplidar import RPLidar

# Set up pygame and the display
os.putenv('SDL_FBDEV', '/dev/fb1')
pygame.init()
lcd = pygame.display.set_mode((320,240))
pygame.mouse.set_visible(False)
lcd.fill((0,0,0))
pygame.display.update()

# Setup the RPLidar
PORTNAME = '/dev/ttyUSB0'
lidar = RPLidar(None, PORTNAME)

# used to scale data to fit on the screen
max_distance = 0

#pylint: disable=redefined-outer-name, global-statement
def process_data(data):
    global max_distance
    lcd.fill((0,0,0))
    for angle in range(360):
        distance = data[angle]
        if distance > 0: # ignore initially ungathered data points
            max_distance = max([min([5000, distance]), max_distance])
        radians = angle * pi / 180.0
        x = distance * cos(radians)
        y = distance * sin(radians)
        point = (160 + int(x / max_distance * 119), 120 + int(y / max_distance * 119))
        lcd.set_at(point, pygame.Color(255, 255, 255))
    pygame.display.update()

scan_data = [0]*360

try:
    print(lidar.info)
    for scan in lidar.iter_scans():
        for (_, angle, distance) in scan:
            scan_data[min([359, floor(angle)])] = distance
    process_data(scan_data)

except KeyboardInterrupt:
    print('Stoping.')
    lidar.stop()
    lidar.disconnect()
```

5.3.4 Modyfikacja kodu producenta

W poniższym listingu utworzono endpoint dla usługi Rest API na adresie /api/rplidar. Port 5000 i adres ip localhosta są domyślnymi ustawieniami biblioteki Flask. W wyniku żądania GET protokołu HTTP na adresie: 127.0.0.1:5000/api/rplidar otrzymamy zwrócony aktualny wynik skanowania przestrzeni 2D.

Listing 5.4: przystosowany do REST API skrypt producenta dla skanera przestrzeni RPLidar: rest_server.py

```
from flask import Flask
import json
from math import floor
from adafruit_rplidar import RPLidar

app = Flask(__name__)

@app.route('/api/rplidar')
def get_distance():
    PORT_NAME = '/dev/ttyUSB0',
    lidar = RPLidar(None, PORT_NAME)

    scan_data = [0]*360
    for scan in lidar.iter_scans():
        lidar.stop()
        for _, angle, distance in scan:
            scan_data[min([359, floor(angle)])] = distance
    return json.dumps(scan_data)
    return 'rplidar_error'
```

5.3.5 Konfiguracja środowiska obsługi REST API

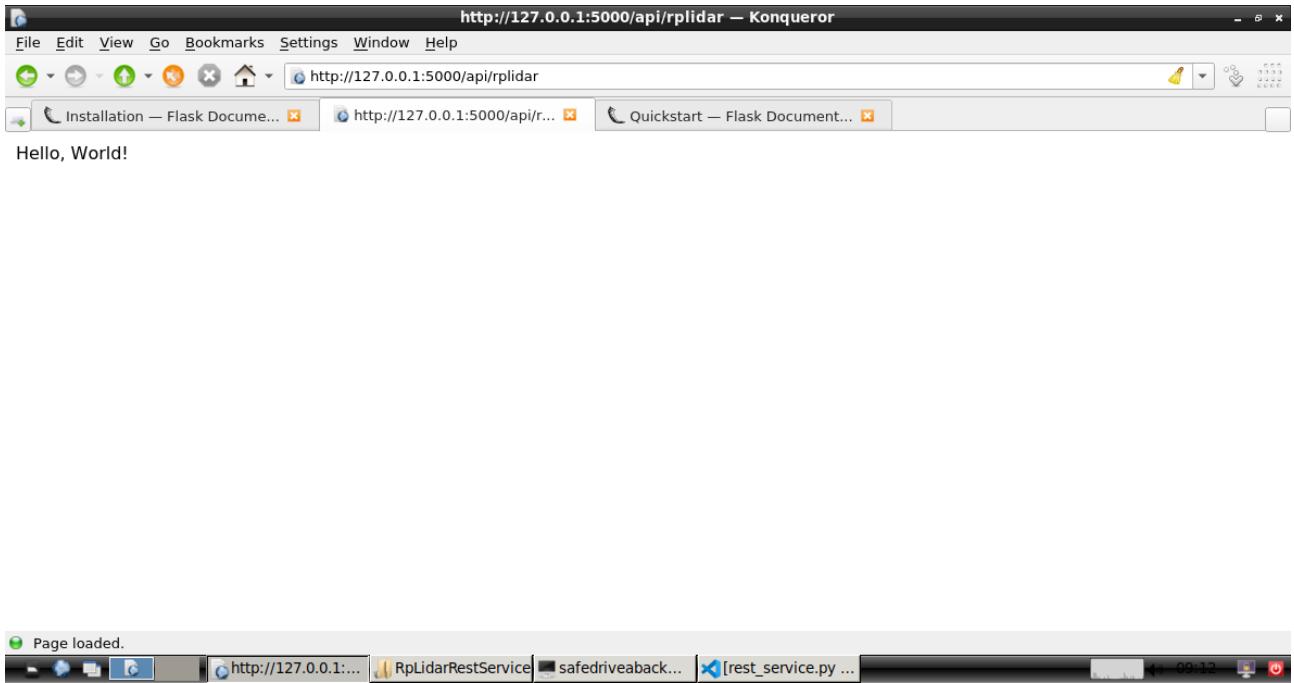
Celem etapu było utworzenie środowiska uruchamiania dla aplikacji webowych, ze względu na wykorzystanie technologii Rest API w kodzie operacyjnym skanera.

Realizując etap utworzono wirtualne środowisko uruchomieniowe dla nowo instalowanych zależności w systemie Raspbian. Dla języka Python i powiązanych z językiem pakietów istnieje narzędzie wirtualizacji o nazwie: venv. W celu instalacji środowiska venv należy użyć polecenia: *sudo apt-get install python3-venv*. W celu inicjalizacji środowiska należy użyć polecenia: *python3 venv venv*, następnie aby uruchomić proces wirtualizacji należy użyć polecenia: *. venv/bin/activate*. Po realizacji opisanej procedury znak zachęty zmieni się na (venv) świadcząc o prawidłowości konfiguracji.

Flask jest biblioteką języka Python, która umożliwia uruchomienie kontenera aplikacji webowych w bardzo podstawowej formie. W celu instalacji biblioteki należy użyć polecenia: *pip install flask*. Zaleca się instalację w środowisku venv w celu ograniczenia konfliktów zależności pomiędzy instalowanymi pakietami. Poleceniem: *export FLASK_APP=rest_service.py* wskazujemy bibliotekę, który kod chcemy uruchomić w kontenerze webowym. Poleceniem: *flask run* uruchamiamy kontener webowy z wskazaną aplikacją na porcie 5000 adresu ip localhosta.

```
Successfully installed Flask-1.1.2 Jinja2-2.11.3 MarkupSafe-1.1.1 Werkzeug-1.0.1 click-7.1.2 itsdangerous-1.1.0
(venv) safedriveaback@safedriveaback:~/Desktop/RpLidarRestService $ nano rest_service.py
(venv) safedriveaback@safedriveaback:~/Desktop/RpLidarRestService $ export FLASK_APP=rest_service.py
(venv) safedriveaback@safedriveaback:~/Desktop/RpLidarRestService $ flask run
 * Serving Flask app "rest_service.py"
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Rysunek 5.28: Uruchomienie biblioteki Flask [źródło: własne]



Rysunek 5.29: Nasłuchiwanie na endpointie /api/rplidar [źródło: własne]

5.3.6 Automatyzacja uruchamiania skryptów startowych

W celu automatyzacji procesu uruchamiania skryptów przy starcie systemu, uruchamiającego webowe środowisko konteneryzacji z aktywnym dostępowym Rest API utworzono skrypt startowy. Skrypt uruchamia środowisko konteneryzacji webowej z wdrożoną aplikacją, która poprzez usługę Rest API umożliwia odczyt współrzędnych odległości i rozmieszczenia w przestrzeni 2D zeskanowanych obiektów na żądanie GET protokołu HTTP w wewnętrznej sieci lokalnej całego systemu.

Listing 5.5: skrypt startowy systemu wbudowanego tylnego run_rest_server.sh

```
#!/bin/sh
export FLASK_APP=rest_server.py
echo "rplidar_rest_server_is_running"
python3 -m flask run --host=0.0.0.0 &
echo "rest_service_is_active"
```

W pliku systemowym .bashrc zlokalizowanym w katalogu domowym użytkownika dodano w końcowej części wpis wpis: `./run_rest_server.sh`. W efekcie plik ze skryptem startowym o nazwie `run_rest_server.sh` zostanie uruchomiony i wykonany po automatycznym zalogowaniu użytkownika do środowiska powłoki systemu. Aby rozwiązywanie działało należy nadać uprawnienia wykonywalne '+x' dla skryptu startowego. W tym celu wykonano polecenie: `/texttchmod 777 ./run_rest_server.sh` przypisując wszystkie możliwe uprawnienia wykorzystania skryptu przez system.

```
GNU nano 3.2 .bashrc

# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

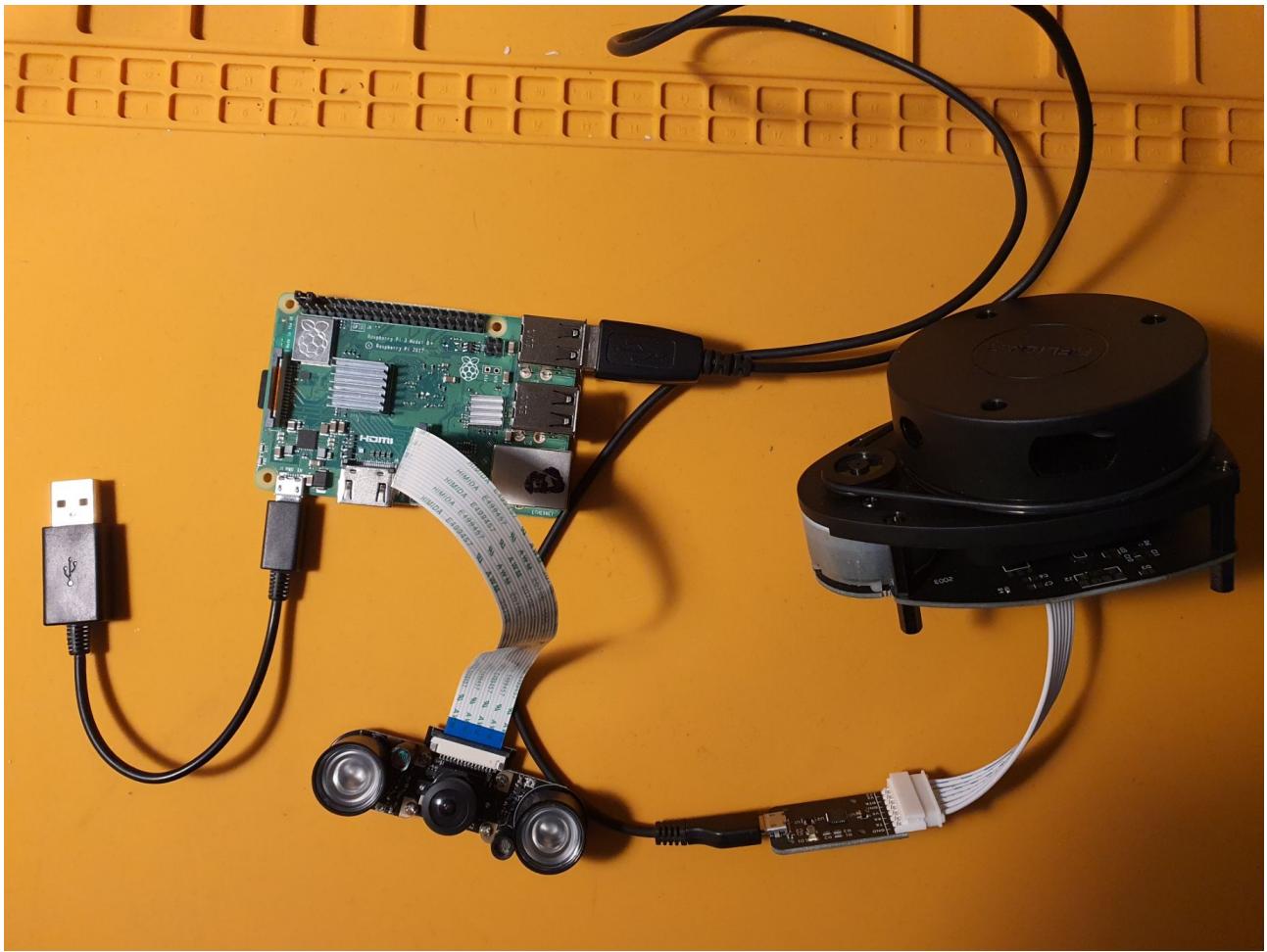
if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
    if [ -f /usr/share/bash-completion/bash_completion ]; then
        . /usr/share/bash-completion/bash_completion
    elif [ -f /etc/bash_completion ]; then
        . /etc/bash_completion
    fi
fi

./run_rest_server.sh
```

Rysunek 5.30: Uruchamianie skryptów przy starcie systemu [źródło: własne]

Prototyp tylnego systemu wbudowanego został ukończony.



Rysunek 5.31: Prototyp tylnego systemu wbudowanego [źródło: własne]

5.4 Podłączenie czujnika odległości

Opis czynności wykonanych podczas procesu wdrażania modułu czujnika odległości.

Zasadność zastosowania

Czujnik odległości umieszczony z przodu pojazdu umożliwia pomiar odległości w centymetrach, w czasie rzeczywistym, z optymalną do granicznej odległości wartością cykli zegarowych.

Pomiar jest dokonywany pomiędzy pojazdem motocyklisty, na którym sensor jest umieszczony, a obiektem znajdującym się przed pojazdem motocyklisty.

W praktycznym zastosowaniu pomiar dotyczy pojazdu znajdującego się przed motocyklistą w odległości do 40 metrów. Maksymalną odległość jaką jest w stanie zmierzyć czujnik jest 40 metrów, zgodnie z deklaracją producenta tego produktu w formie noty katalogowej.

Kombinacje wartości mierzonej odległości i aktualnej prędkości poruszania się pojazdu umożliwiają obliczenie minimalnej odległości pomiędzy motocyklistą, a pojazdem poprzedzającym wymaganej aby możliwe było bezpieczne wyhamowanie motocykla, w przypadku zaistnienia sytuacji drogowej wymagającej nagłego hamowania.

Z punktu widzenia bezpieczeństwa ruchu drogowego opisane zastosowanie czujnika odległości jest zasadne.

5.4.1 Konstrukcja oprogramowania w języku C++

W celu sprzętowej obsługi czujnika odległości, wykorzystano oprogramowanie w języku C++, utworzone i udostępnione na licencji open source Apache¹ przez producenta sprzętu - firmę Garmin. Udostępnione oprogramowanie jest możliwe do sklonowania w serwisie Github².

Do komplikacji kodu producenta, zapewniającego serwis odczytu danych odległości z czujnika, dodano bibliotekę uruchamiającą nasłuchujący serwer zapytań restowych poprzez dostępne API i udostępniający na zapytanie klienta odczytany w danej chwili pomiar odległości.

Implementacja usługi REST API powstała na bazie zdalnie dostępnych materiałów dotyczących programowania usług w języku C++³.

Na podstawie fuzji biblioteki producenta czujnika oraz instrukcji implementacji serwisu REST API w C++ powstał autorski projekt o następującej strukturze katalogów:

Name	Date modified	Type	Size
BasicController.cpp	15/12/2018 09:19	C++ Source File	1 KB
BasicController.h	15/12/2018 09:19	C Header Source File	1 KB
lidarlite_v3.cpp	12/03/2021 17:25	C++ Source File	15 KB
lidarlite_v3.h	22/04/2019 21:59	C Header Source File	3 KB
main.cpp	13/03/2021 01:14	C++ Source File	1 KB
main.out	09/08/2021 13:27	Wireshark capture file	1,240 KB
run_rest.sh	09/08/2021 14:07	Shell Script	1 KB
Service.cpp	13/03/2021 00:58	C++ Source File	2 KB
Service.h	13/03/2021 00:44	C Header Source File	1 KB
sh_script_exec.cpp	12/03/2021 12:14	C++ Source File	1 KB

Rysunek 5.32: Struktura plików projektu obsługi czujnika odległości [źródło: własne].

W celu komplikacji i uruchomienia powyższego programu utworzono skrypt w języku Bash, komplilujący projekt i uruchamiający go oraz informujący użytkownika systemu Raspbian o przeprowadzonych operacjach w oknie terminala.

Listing 5.6: skrypt powłoki bash: run_rest.sh

```
#!/bin/sh
echo "start_compilation"
g++ -I/home/safedriveafront/Desktop/RestServiceFrontSoftDrive /home/safedriveafront/Desktop/RestServiceFrontSoftDrive/main.cpp
echo "compilation_success"
echo "rest_server_is_active"
echo safedrivea | sudo -S /home/safedriveafront/Desktop/RestServiceFrontSoftDrive/main.out
```

¹Apache, <http://www.apache.org/licenses/LICENSE-2.0> [dostęp: 27.08.2021].

²GitHub, https://github.com/garmin/LIDARLite_RaspberryPi_Library [dostęp: 27.08.2021].

³Kompikownia, <https://www.kompikownia.pl/index.php/2018/07/26/rest-api-w-c-biblioteka-casablanca-cpprest/> [dostęp: 27.08.2021].

Przedstawiony skrypt uruchamiający komplikację dodano do usługi automatycznego wykonania przy starcie systemu Raspbian. Poprzez taką formę serwisu startowego systemu zapewniono bezobsługową pracę całego systemu przedniego czujnika odległości. Od tego momentu możliwe było uruchomienie zaimplementowanej funkcjonalności wyłącznie poprzez podłączenie zasilania do minikomputera. Skonstruowany system posiadał cechy systemu wbudowanego. Wymaganiem poprawnego startu usługi była konfiguracja systemu Raspbian zgodnie z opisem z rozdziału: "Pierwsze uruchomienie systemu", a także dostępność skonfigurowanego hotspota WiFi z poziomu sprzężonego układu - smartfona z dostępem do mobilnego internetu. Wykorzystane w poleceniu przełączniki: -lcpprest -lpthread -lboost_system -lssl -lcrypto -lstdc++ są wymagane do prawidłowej komplikacji kodu.

W celu uruchomienia skryptu wraz ze startem systemu Raspbian wykonano poniższe czynności:

1. Wykonanie polecenia: `sudo crontab -e`.
2. Wybór edytora tekstu - nano.
3. Na końcu otwartego pliku dodanie wpisu: `@reboot ./run_rest.sh`.
4. Zapis pliku i wyjście.
5. Nadanie dla skryptu uprawnień wykonalności poleceniem: `sudo chmod a+x run_rest.sh`.

W dalszym opisie struktury projektu zostały pominięte wszystkie pliki nagłówkowe języka C++. Następujące pliki w strukturze projektu nie zostały zmodyfikowane w wyniku łączenia bibliotek REST API i Lidar Lite:

- lidarlite_v3.cpp
- BasicController.cpp

Punktem wejścia programu jest plik main.cpp:

Listing 5.7: plik startowy: main.cpp

```
int main()
{
//read actual ip address
Sh_script_exec sh_exe = Sh_script_exec();
string ip = sh_exe.get_ip_addr();
ip.erase(remove(ip.begin(), ip.end(), '\n'), ip.end());
cout << "\n\nrest_server_ip:" << endl;
cout << ip;
cout << "\n\n";
//start rest service
Service serv(ip,"8080");
serv.setEndpoint("/api");
serv.accept().wait();
while(1==1)
{
sleep(1000);
}
return 0;
}
```

W przytoczonej sekwencji poleceń następują kolejno funkcjonalności:

1. Wczytanie aktualnego adresu IP minikomputera w prywatnej sieci LAN składającej się z urządzenia Raspberry Pi i smartfona.
2. Informacja dla użytkownika o wartości wczytanego adresu IP.
3. Uruchomienie nasłuchiwanego na serwerze REST API na endpointie /api i porcie 8080 dla wczytanego uprzednio adresu IP.
4. Wprowadzenie sekundowego opóźnienia w przetwarzaniu zapytań skierowanych do rest serwisu w celach większej wydajności i stabilności działania.

Plik main.cpp korzysta z klasy Service opisanej przez plik Service.cpp

Listing 5.8: plik Service.cpp

```
void Service::handleGet(http_request message) {
    vector<string> path = requestPath(message);
    if(path.empty()) {
        message.reply(status_codes::BadRequest);
    }
    else {
        if(path[0]==="distance") {

            //lidar logic
            LIDARLite_v3 myLidarLite;
            __u16 distance_lid;
            __u8 busyFlag;
            myLidarLite.i2c_init();
            myLidarLite.configure(0);

            busyFlag = myLidarLite.getBusyFlag();

            //rest logic
            if (busyFlag == 0x00) {
                myLidarLite.takeRange();
                distance_lid = myLidarLite.readDistance();

                std::cout << "distance:";
                printf("%d\n", distance_lid);
                std::cout << "\n\n";
            }

            json::value number;
            number["distance"] = json::value::number(distance_lid);
            message.reply(status_codes::OK, number);
        }
        else {
            message.reply(status_codes::BadRequest);
        }
    }
}
```

W przytoczonej powyżej sekwencji poleceń następują kolejno funkcjonalności:

1. Dla poprawnie skierowanego zapytania http do serwisu kredytowanego jest odpowiedź w formacie JSON przyjmującym postać distance : "wczytana odległość".
2. Wczytywana jest odległość z czujnika za pośrednictwem klasy LIDARLite_v3.
3. Użytkownik jest informowany o wczytanej odległości w terminalu systemowym.
4. Wysyłana jest odpowiedź w podanym formacie JSON z kodem odpowiedzi 200.

Przed uruchomieniem nasłuchującego zapytań serwera, potrzebna jest znajomość adresu IP, pod jakim serwer będzie dostępny dla hostów z zewnątrz. Za pobranie wartości adresu IP urządzenia odpowiada klasa zdefiniowana w pliku Sh_script_exec.cpp

Listing 5.9: plik Sh_script_exec.cpp

```
class Sh_script_exec {

private:
    std::string exec(const char* cmd) {
        std::array<char, 128> buffer;
        std::string result;
        std::unique_ptr<FILE, decltype(&pclose)> pipe(popen(cmd, "r"), pclose);
        if (!pipe) {
            throw std::runtime_error("popen() failed!");
        }
        while (fgets(buffer.data(), buffer.size(), pipe.get()) != nullptr) {
            result += buffer.data();
        }
        return result;
    }

public:
    std::string get_ip_addr() {
        const char* cmd = "ifconfig wlan0 | grep -v inet | awk '{ print $2 }' | head -1";
        std::string res = exec(cmd);
        return res;
    };
}
```

W przytoczonym listingu:

1. Metoda exec zwraca rezultat wykonania zadanego polecenia do powłoki Bash systemu Raspbian.
2. pole cmd składa się z utworzonego z 3 potoków zapytania filtrującego adres IP hosta wykonującego zapytanie na bazie wyświetlonej konfiguracji sieci tego hosta.
3. zwracany jest wynik w formie przefiltrowanego z konfiguracji sieciowej adresu IP hosta.

W wyniku uruchomienia sekwencji odczytów danych z czujnika otrzymano następujący widok terminala:

```
safedriveafront@safedriveafront: ~/  
File Edit Tabs Help  
344  
345  
346  
347  
346  
345  
346  
346  
345  
345  
346  
346  
344  
346  
346  
346  
345  
346  
345
```

Rysunek 5.33: Odczyt danych odległości z czujnika w centymetrach [źródło: własne].

W wyniku uruchomienia sekwencji odczytów danych z czujnika zakończonych powstaniem wyjątku otrzymano następujący widok terminala:

```
safedriveafront@safedriveafront: ~/De
File Edit Tabs Help
Failed to acquire bus access and/or talk to slave.
Failed to acquire bus access and/or talk to slave.
Failed to acquire bus access and/or talk to slave.
    0
Failed to acquire bus access and/or talk to slave.
Failed to acquire bus access and/or talk to slave.
Failed to acquire bus access and/or talk to slave.
    0
Failed to acquire bus access and/or talk to slave.
Failed to acquire bus access and/or talk to slave.
Failed to acquire bus access and/or talk to slave.
    0
Failed to acquire bus access and/or talk to slave.
Failed to acquire bus access and/or talk to slave.
Failed to acquire bus access and/or talk to slave.
    0
```

Rysunek 5.34: Odczyt danych odległości z czujnika zakończony niepowodzeniem [źródło: własne].

5.4.2 Konstrukcja oprogramowania w języku Python

W wyniku powstały problemów z wdrożeniem systemu w rzeczywistych warunkach pracy, wdrożono rozwiązanie polegające na zmianie biblioteki i języka kodowania z języka C++ na inny, zastępczy. Źródłem poszukiwań dla poszukiwań nowej implementacji był serwis GitHub, a kierunkiem poszukiwań prostota rozwiązania, w celu uniknięcia komplikacji kodu, wzorowana na implementacji skanera obiektów w przestrzeni 2d. Wynikiem spełniającym powyższe założenia okazała się biblioteka napisana w języku Python⁴.

Moduł obsługi czujnika importowany przez bibliotekę zainstalowaną w systemie poleceniem: `sudo pip3 install adafruit-circuitpython-lidarlite`. Moduł obsługi usługi REST API dostępny w bibliotece Flask zainstalowano w systemie poleceniem: `sudo pip3 install flask`.

Bieżąca implementacja usługi wykorzystująca obie biblioteki powstała przez dopisanie do implementacji z dokumentacji obsługi czujnika kodu zapewniającego obsługę REST API.

⁴GitHub, https://github.com/adafruit/Adafruit_CircuitPython_LIDARLite [dostęp: 27.08.2021].

Listing 5.10: plik run_rest.py

```
import board
import busio
import adafruit_lidarlite

from flask import Flask
import json

app = Flask(__name__)

i2c = busio.I2C(board.SCL, board.SDA)

sensor = adafruit_lidarlite.LIDARLite(i2c)

print("distance_rest_listener_running")

@app.route('/api/distance')
def getDistance():
    try:
        print((sensor.distance,))
        dist = sensor.distance
        dict = {'distance':dist}

        return json.dumps(dict)
    except RuntimeError as e:
        print(e)
        dist=-1
        dict = {'distance':dist}

    return json.dumps(dict)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080)
```

W przytoczonym listingu:

1. Zimportowano bibliotekę board odpowiedzialną za obsługę specyficznych dla Raspberry Pi funkcjonalności takich jak obsługa poszczególnych pinów.
2. Zimportowano bibliotekę busio odpowiedzialną za obsługę interfejsu połączenia szeregowego. Na potrzeby specyfikacji projektu będzie to połaczenie magistralą i2c.
3. Zimportowano biblioteki do obsługi czujnika odległości, usługi REST API i generowania danych w formacie JSON.
4. W implementacji ustawiono nasłuchiwanie na endpoint /api/distance.
5. Gdy dystans uda się odczytać zwracany jest w odpowiedzi za zapytanie protokołu HTTP słownik w postaci distance: "odczytana odległość". Taki format odpowiedzi jest zgodny z formatem zwracanym w poprzedniej implementacji w języku C++ i zapewnia zgodność z systemem aplikacji mobilnej generującej zapytania HTTP do układu czujnika odległości, pomimo zmiany implementacji w układzie czujnika.
6. Gdy odczytanie dystansu nie powiodło się zwracana jest wartość odległości wynosząca -1, stanowiąca sygnał o zaistniałym błędzie dla aplikacji mobilnej.
7. Serwer usługi REST nasłuchiwa na wszystkich możliwych dopasowaniach adresu IP hosta, reprezentowanych przez wartość 0.0.0.0. W przypadku skonstruowanego modułu czujnika dostępny jest jeden adres IP. Nasłuchiwanie na adresie jest kierowane do aplikacji przez port 8080, który stanowi zgodność wsteczną z poprzednią implementacją, dla obsługiwanej aplikacji mobilnej.

po przeprowadzeniu procesu zmiany implementacji serwisu czujnika odległości, aktualizacji wymagał wpis do skryptu startowego powłoki Bash, uruchamianego wraz ze startem systemu. Aktualizacja polegała za zastąpieniu poprzednich wpisów w listingu następującym: *echo safedrivea — sudo -S python3 /home/safedriveafront/lidar_lv3_python/run_rest.py*, w którym zawarte jest hasło użytkownika root oraz ścieżka dostępową do pliku wykonywalnego run_rest.py w systemie Raspbian.

Realizacja nowej implementacji skutkuje automatycznym i poprawnym uruchomieniem nasłuchiwania zapytań HTTP po stracie systemu.

```

[ 5.444866] sd 0:0:0:0: [sda] No Caching mode page found
[ 5.450651] sd 0:0:0:0: [sda] Assuming drive cache: write through

Raspbian GNU/Linux 10 safedriveafront tty1

safedriveafront login: safedriveafront (automatic login)

Last login: Wed Aug 18 11:10:30 CEST 2021 on ttys1
Linux safedriveafront 5.10.17-v7+ #1403 SMP Mon Feb 22 11:29:51 GMT 2021 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
[sudo] password for safedriveafront: distance rest listener running
 * Serving Flask app 'run_rest' (lazy loading)
 * Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)
-

```

Rysunek 5.35: Interfejs dostępny po starcie systemu czujnika odległości [źródło: własne].

5.4.3 Problemy z wdrożeniem

Napotkane przeszkody podczas wdrażania rozwiązania.

Zmiana biblioteki kodu

Po instalacji prototypu projektu w samochodzie osobowym, w celu przetestowania działania przedniego czujnika odległości, uzyskanie pomiaru wartości odległości z czujnika, okazało się problematyczne, z powodu braku odczytu spodziewanych wartości odległości miedzy czujnikiem, a obiekta mi przed pojazdem.

W wyniku analizy historii logów z minikomputera Raspberry Pi, podczas podłączonego, pracującego czujnika, w czasie testów wykonywanych w samochodzie osobowym, okazało się, że użyte oprogramowanie uruchomione na minikomputerze zgłasza błąd i następuje przerwanie procesu kontynuacji działania programu obsługiwany go przez system.

Wykonane testy polegały na kierowaniu podłączonego czujnika w stronę różnych obiektów zewnętrznych, z wnętrza samochodu i obserwacji zgłaszanych w konsoli systemu Raspbian logów i wartości zmierzanej przez czujnik odległości.

Użyte w celu obsługi sprzętowej czujnika oprogramowanie, zostało napisane w języku C++, przez producenta użytego czujnika - firmę Garmin. Na bazie oprogramowania udostępnionego przez producenta, na licencji open source, powstała autorska implementacja, w formie nieopublikowanego branch-a systemu git, w celu dopasowania jej do wymagań realizowanego projektu. Powodem powstania błędu była nietestowana biblioteka albo niepoprawna jej modyfikacja. Z powodu ograniczeń czasowych będących ograniczeniem zasobów fazy realizacji projektu, nie podjęto próby analizy głębszych przyczyn powstawania błędu. Jako rozwiązanie projektowe problemu, ustalone zostało przeszukanie zasobów otwarto źródłowego repozytorium serwisu GitHub, w celu znalezienia zastępczej biblioteki sterującej pracą czujnika odległości.

Zrezygnowano w istniejącego odgałęzienia biblioteki w języku C++ na rzecz analogicznej pod względem funkcjonalnym biblioteki w języku Python, produkcji firmy Adafruit. Biblioteka w języku Python okazała się wersją o znacznym ograniczeniu ilości kodu względem poprzedniej biblioteki. Po przeprowadzonych analogicznych testach pomiaru odległości z czujnika obiektów poza pojazdem, mierzonych z wnętrza pojazdu, przy unieruchomionym pojazdzie, system przestał stosować mechanizm przerwania na skutek powstającego wyjątku i pomiar odległości okazał się możliwy w trybie ciągłej pracy.

Debugowanie danych GPS

Projektowana w systemie Android aplikacja przetwarzająca dane z czujnika odległości pokazywała prawidłowy pomiar odległości na wyświetlaczu smartfona w warunkach domowego laboratorium. Aplikacja przetwarzająca dane odległości współpracowała z danymi dotyczącymi aktualnej prędkości przemieszczania się urządzenia. W warunkach laboratoryjnych nie zostały zaobserwowane jakiekolwiek anomalie uniemożliwiające stabilną pracę aplikacji.

Przystąpiono do testowania aplikacji w warunkach terenowych podczas jazdy samochodem osobowym. Podczas przemieszczania się pojazdem, aplikacja z widoczną w GUI wizualizacją pojazdu znajdującego się przed motocyklem i wyświetlonymi wartościami prędkości i odległości na interfejsie, wyłycała się chwilę po przemieszczeniu pojazdu, na odległość około 1 metra. Postanowiono przeprowadzić analizę logów generowanych podczas pracy aplikacji w systemie. W wygenerowanych logach dało się odczytać błędy dotyczące mechanizmu współbieżności. Proces debugowania był przeprowadzony w czasie 3 godzin, we wnętrzu pojazdu i wymagał dziesięciokrotniej próby zmiany położenia samochodu. Wnioskiem z przeprowadzonych analiz był znaleziony powód powstawania błędu, który wskazywał na błąd synchronizacji wątków, odpowiedzialnych za przetwarzanie danych odczytywanych z metody typu callback, obsługującej system czujnika GPS telefonu. Dlatego błąd ujawniał się podczas zmiany położenia, a nie w warunkach braku przemieszczania się urządzenia.

Rozwiązaniem problemu okazała się synchronizacja wszystkich metod w GUI deklaracją "synchronized" z języka Java oraz utworzenie nowych współbieżnych wątków w miejscach kodu, w których występowało oddziaływanie na generowanie widoku GUI. Dodatkowo jako główny wątek aplikacji został ustalony dynamicznie generowany graficzny interfejs użytkownika.

Po wprowadzeniu opisanych trzech poprawek w kodzie, zastosowano koncepcję mockowania dla odczytu danych z modułu GPS smartfona i w statecznych, domowych warunkach laboratoryjnych sprawdzono prawidłowość wyświetlania się GUI dla mockowanych, rosnących liniowo wartości prędkości i losowo odczytywanych rzeczywistych wskazań czujnika odległości. W czasie przeprowadzonych testów interfejs nie wykazywał nieprawidłowości w działaniu. Próby przemieszczanie się prototypu, przeprowadzone w samochodzie osobowym również nie wykazały błędu.

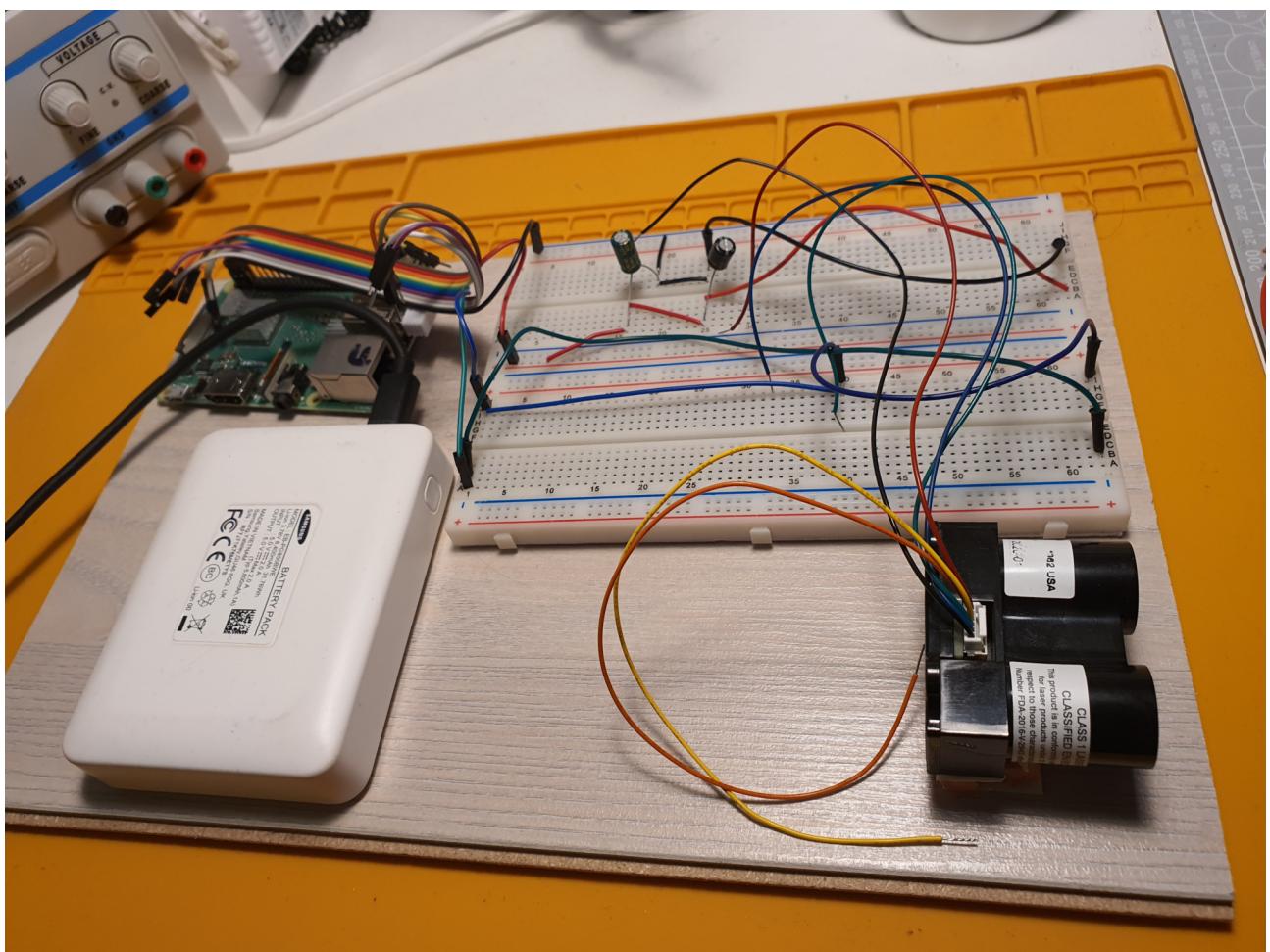
Zmiana punktu pomiarowego

Podczas wykonywanych pomiarów odległości z wnętrza samochodu osobowego, wynik pomiarów okazywał się nieregularny, ze zwiększoną tendencją do zgłaszania błędu pomiaru, niż jakikolwiek odczyt wartości odległości. Ten stan utrzymywał się zarówno podczas jazdy, jak i postoju pojazdem. W większości przypadków sensor gubił pomiar wielkości odległości. Początkowo za przyczynę problemu uznano błąd w oprogramowaniu i analiza oraz debugowanie oprogramowania pod kątem znalezienia przyczyny wymagała 5 dni pracy etatowej bez rezultatu i mało sprecyzowanych kombinacjach wariantów implementacji, które zostały pominięte ze względu na brak sprecyzowanych przesłanek ich użycia na zasadzie prób i błędów.

Powód powstania błędów odczytu odległości z czujnika okazał się bardziej trywialny i o charakterze sprzętowym. Czujnik działa na zasadzie generowania wiązki lasera przez emiter, która dosiąga obiektu, od którego odległość chcemy zmierzyć i jest po zjawisku odbicia optycznego od tego obiektu analizowana przez obiektyw czujnika w celu określenia odległości, w jakiej znalazły się obiekty od emitera⁵. Wiązka lasera jest podatna na zjawisko załamania wiązki światła. Najprawdopodobniej przyczyną sporadycznych odczytów odległości przez sensor było zjawisko załamania światła, które występowało na ukośnie ustawionej względem emitowanej wiązki szybie samochodu podczas pomiarów wykonywanych z wnętrza pojazdu. Ta hipoteza znajduje potwierdzenie po przeprowadzeniu testów z czujnikiem umieszczonym poza pojazdem i testów w domowych warunkach laboratoryjnych, w których czujnik wskazywał mierzone wielkości odległości, tylko sporadycznie zgłaszaając błąd odczytu.

Rozszerzenie konstrukcji prototypu

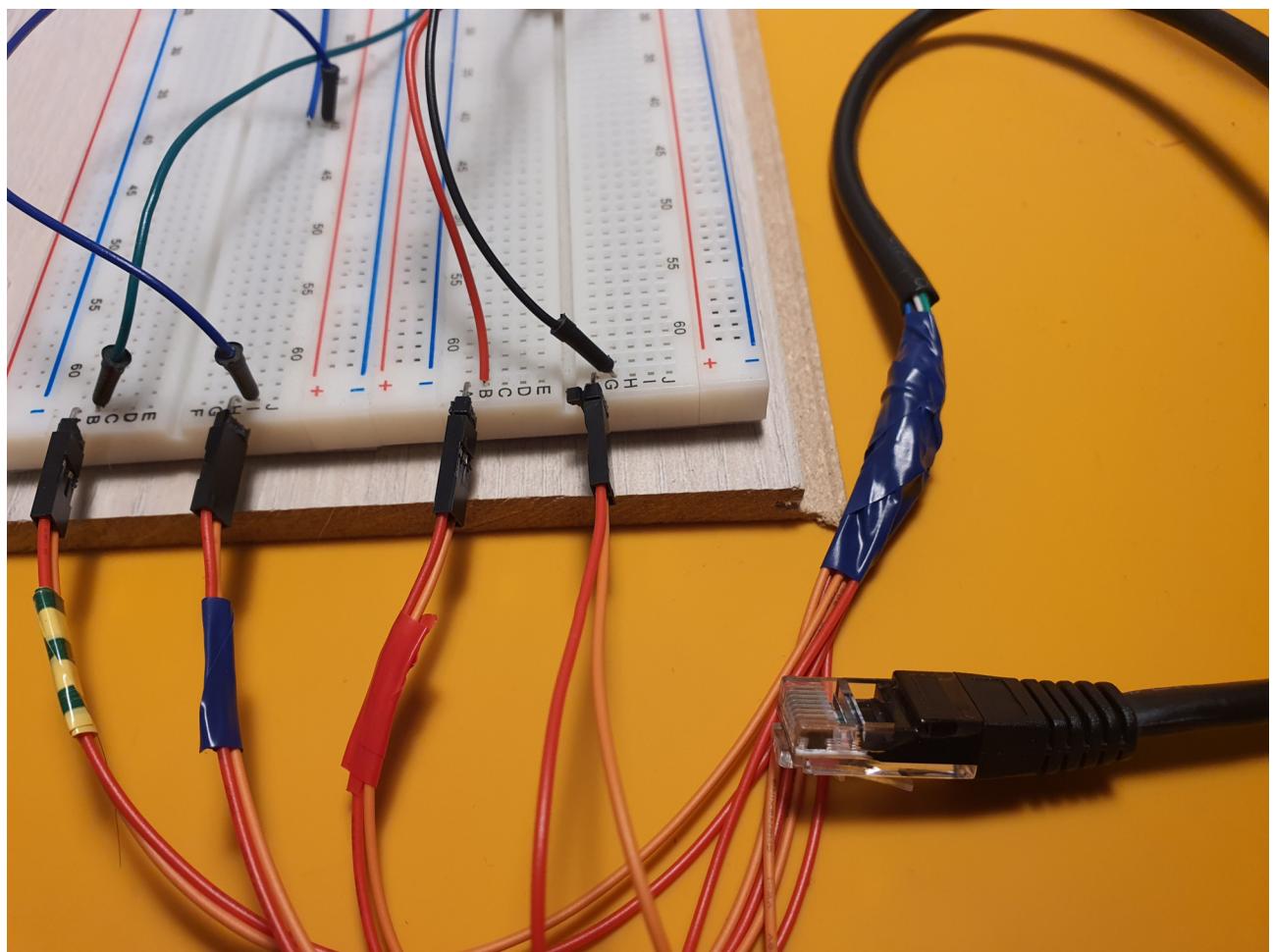
Skonstruowany prototyp części sprzętowej projektu, był przeznaczony do testów wewnętrz pojazdu samochodowego. Z powodu braku możliwości odczytu pomiarów odległości z wnętrza pojazdu, konieczne było przeprojektowanie prototypu aby pomiar był możliwy poza pojazdem, podczas jazdy, w sposób niezagrożający bezpieczeństwu ruchu drogowego.



Rysunek 5.36: Wersja prototypu przed modyfikacją [źródło: własne].

⁵AGH, <http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.baztech-article-AGH1-0025-0083> [dostęp: 27.08.2021].

Główna modyfikacja było umożliwienie montażu samego czujnika odległości poza pojazdem, podczas gdy pozostałe moduły prototypu pozostawać powinny wewnątrz pojazdu. W celu realizacji tej modyfikacji, postanowiono skonstruować przedłużenie przewodów połączeniowych pomiędzy czujnikiem a płytą stykową prototypu. Ze względu na małą średnicę wiązki dla ilości 4 przewodów postanowiono wykorzystać przewód Ethernet, w formie skrętki nieekranowanej o 8 przewodach, z których wykorzystano 4. Z płytki prototypowej wyprowadzono przedłużenia z jej pól skonstruowane z listwy połączeniowej typu goldpin. Do 4 wyprowadzeń goldpin z płytki stykowej wpięto po jednym wtyku, wyprowadzonym z przewodów skrętki sieciowej. Wykorzystano tylko przewody w kolorach jednolitych, w celu bardziej intuicyjnej identyfikacji poszczególnych połączeń oraz ich rozproszeniu we wtyku rj-45, który znajdował się po drugiej stronie skrętki, względem połączeń typu goldpin. Przy czujniku odległości znajdującym się na zewnątrz pojazdu, wykorzystano adapter wtyku rj-45 na listwę 8 wtyków ze śrubami mocującymi wtyki, do których wpięto przewód zakończony wtykiem rj-45 łączący z płytą stykową. Do wyprowadzeń adaptera, od strony listwy połączeniowej, wpięto 4 przewody wychodzące od czujnika odległości, podłączając je do odpowiednich przewodów jednokolorowych w skrętce, zgodnie ze specyfikacją konstrukcji prototypu i zachowania zgodności połączeń między czujnikiem a modułami prototypu.



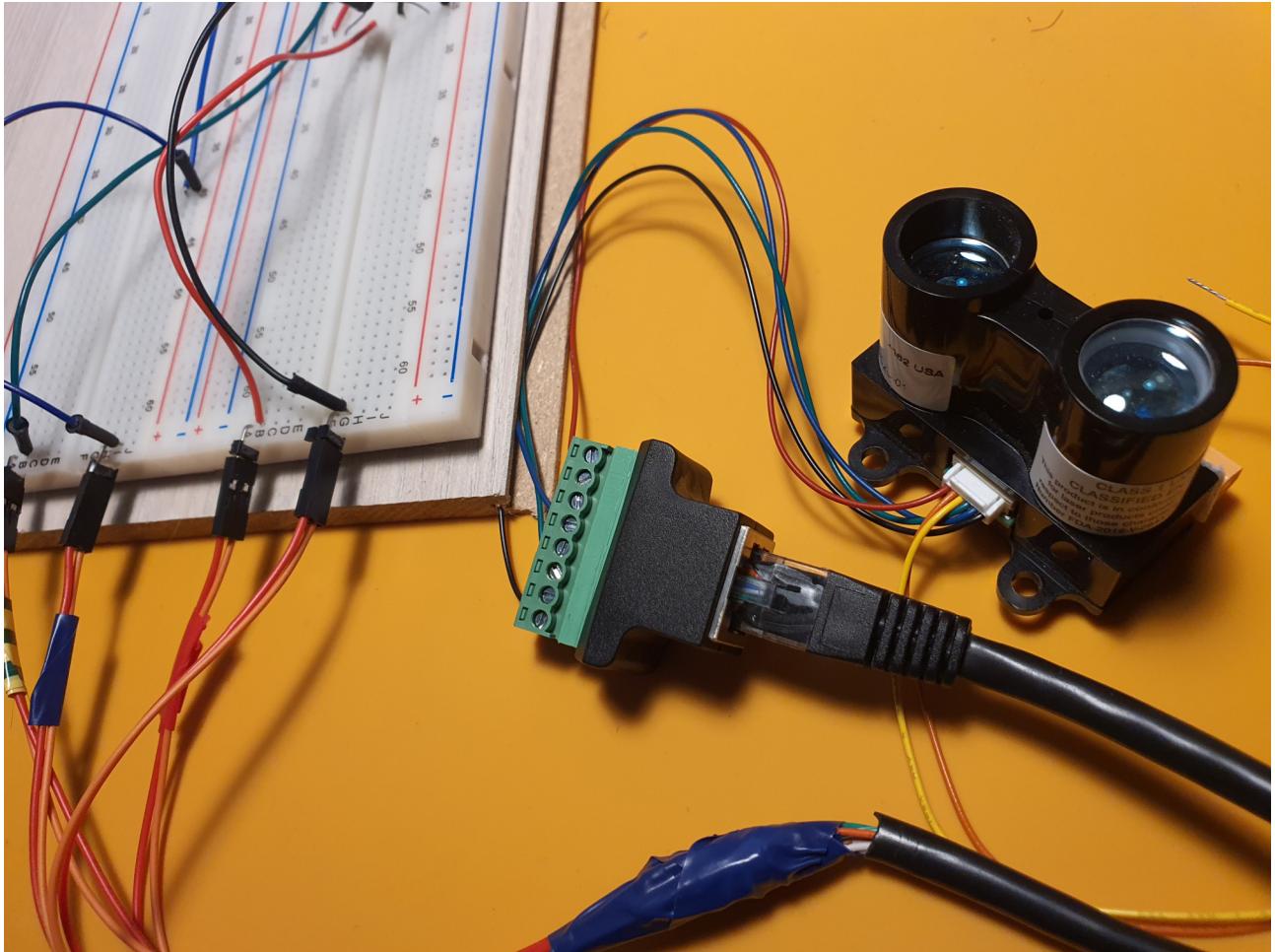
Rysunek 5.37: Przedłużanie przewodowego połączenia z czujnikiem [źródło: własne].

Ciągłość połączeń w obwodzie została sprawdzona uniwersalnym miernikiem wielkości elektrycznych.



Rysunek 5.38: Pomiar ciągłości obwodu [źródło: własne].

W tej formie prototypu możliwy był montaż czujnika poza kabiną pojazdu i pozostałych modułów prototypu wewnątrz pojazdu na czas testów.



Rysunek 5.39: Wersja prototypu po modyfikacji [źródło: własne].

Czujnik odległości, w celach bezpieczeństwa ruchu pojazdów na czas testów przymocowano do uchwytu pióra wycieraczki na pojeździe, unieruchamiając go plastikową opaską zaciskową jednokrotnego użytku. Przewód przedłużający przeprowadzono wzdułż pozostających w spo- czynku wycieraczezy szyby pojazdu i przez próg drzwiowy wprowadzono go do wnętrza pojazdu, w którym znajdowała się platforma z prototypem. Przewód przedłużający znajdującym się na zewnątrz pojazdu unieruchomiono dodatkowo opaskami zaciskowymi. Zamknięcie i otwieranie drzwi pojazdu było nieinwazyjne dla poprowadzonego przewodu.

Ograniczenia mierzonej odległości

Podczas przeprowadzania testów prototypu zainstalowanego w pojeździe, przy odczytach odległości od poprzedzających pojazdów zauważono, że czujnik wykazuje wysoką precyzyję pomiarów do odległości około 12 metrów, natomiast w przypadku obiektów zlokalizowanych w odległości pomiędzy 12 a 40 metrów od punktu pomiarowego, czujnik traci na częstotliwości wskazań odczytów, która staje się niewystarczająca aby zachować płynność animacji w GUI. Prawidłowych pomiarów odległości przekraczających wielkość 40 metrów nie zauważono. Prawdopodobną hipotezą wydaje się specyfika sprzętowa czujnika, który nie został skonstruowany do wskazań odległości w wysokim wskaźnikiem odświeżania wraz ze wzrostem mierzonej odległości. Zastosowanie czujnika o większej precyzyji pomiarów ze zwiększoną częstotliwością pomiarową, przekracza rama budżetowe realizowanego projektu.

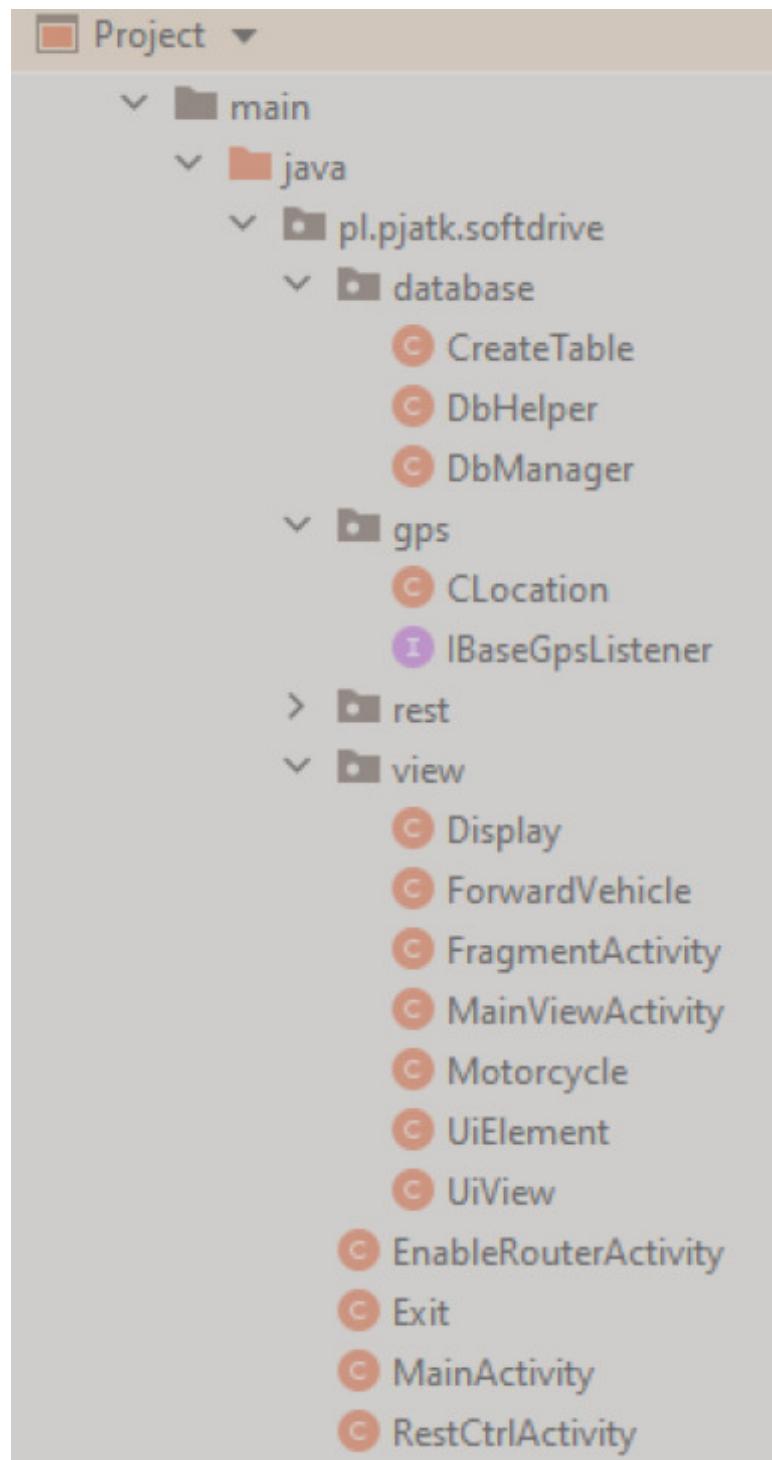
W celu rozwinięcia funkcjonalności realizowanych rozwiązań projektowych, w przyszłości możliwe jest rozbudowanie schematu o dodatkowy czujnik, który zapewniłby kompensację straconych wartości pomiarowych mierzonych przez obecnie zamontowany czujnik, zwiększąc dokładność i jakość pomiarów w mechanizmie synchronizacji współpracy tych sensorów oraz nadmiarowość użytych rozwiązań sprzętowych, w przypadku awarii jednego z czujników.

5.5 Aplikacja mobilna

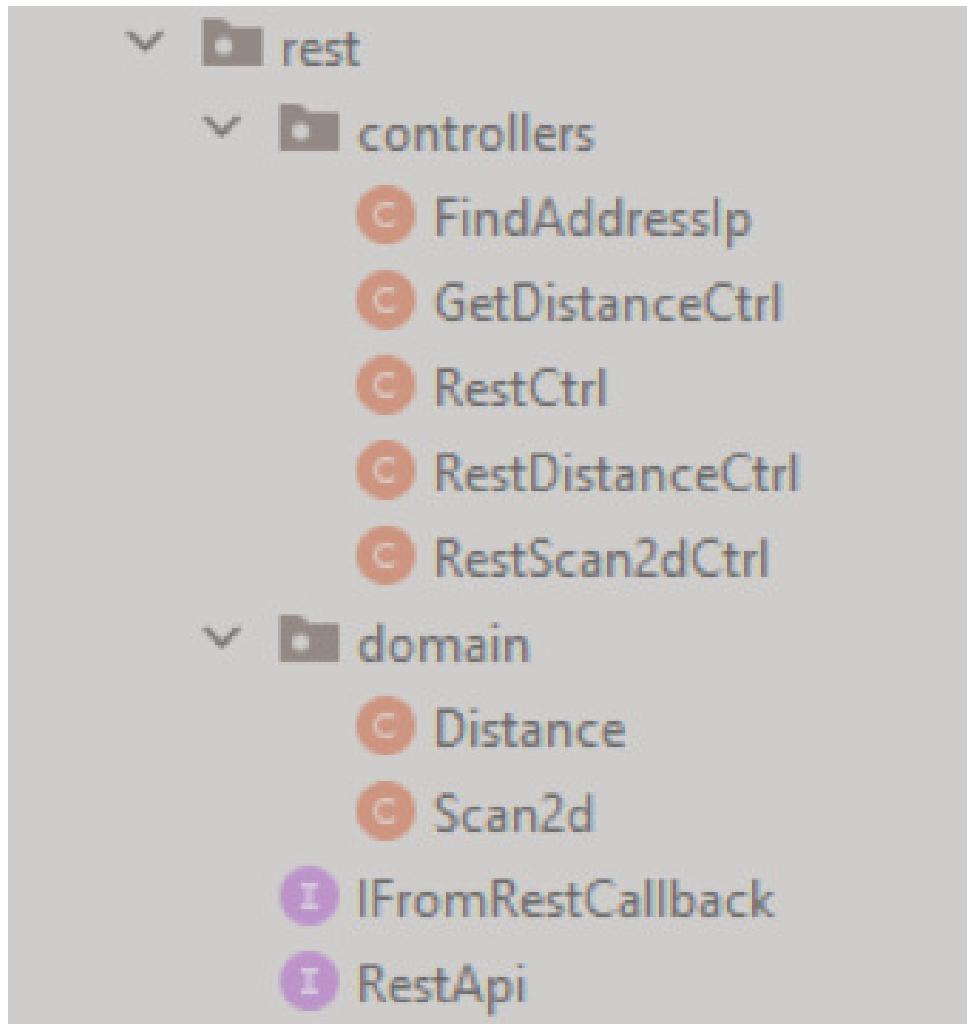
Charakterystyka aplikacji mobilnej projektowanego rozwiązania.

Struktura projektu aplikacji

Struktura plików projektu aplikacji mobilnej jest następująca:



Rysunek 5.40: Struktura plików projektu aplikacji mobilnej [źródło: własne].



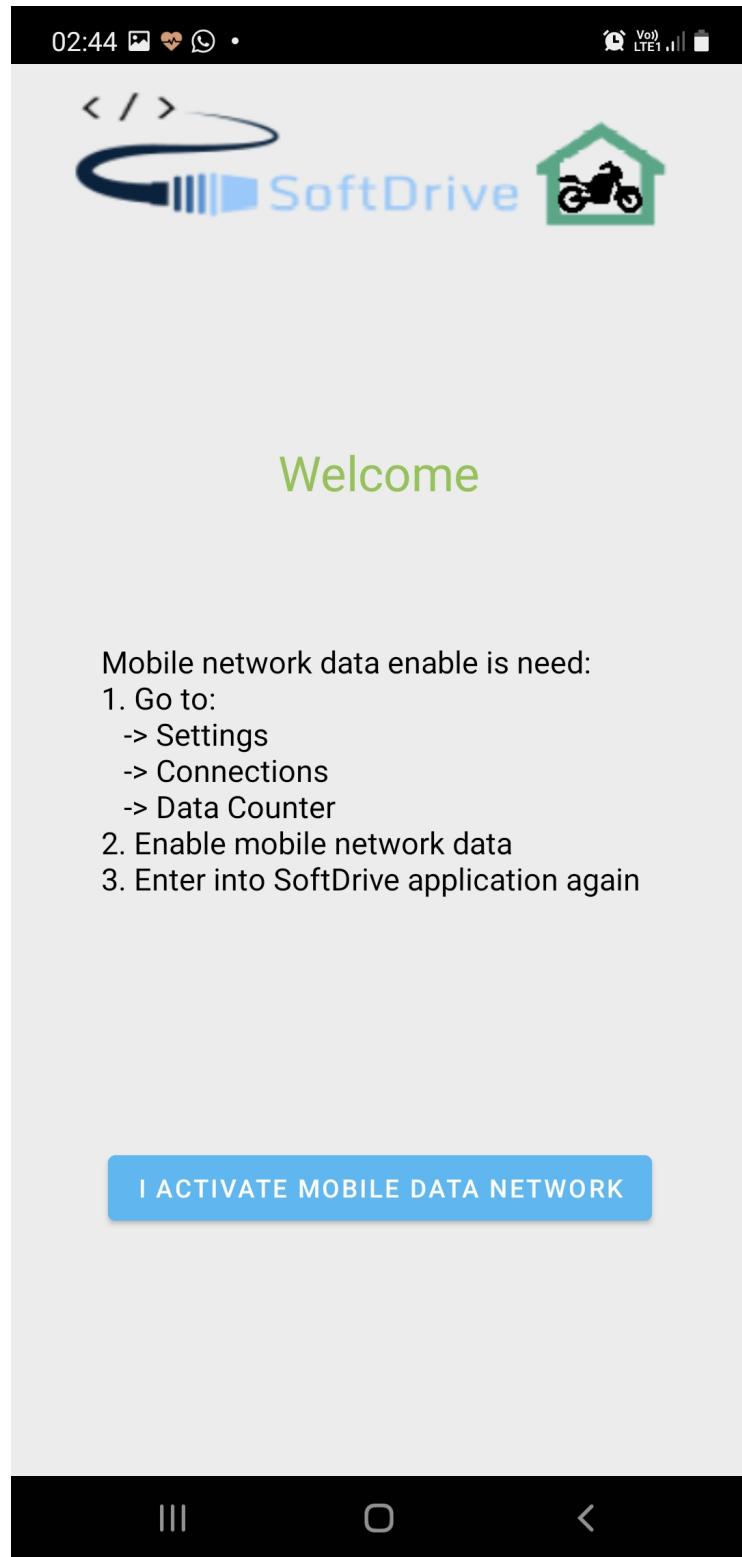
Rysunek 5.41: Struktura plików pakietu rest projektu aplikacji mobilnej [źródło: własne].

5.6 Przepływ sterowania w aplikacji

Opis kolejności aktywacji poszczególnych pakietów pod względem funkcjonalnym jest następujący:

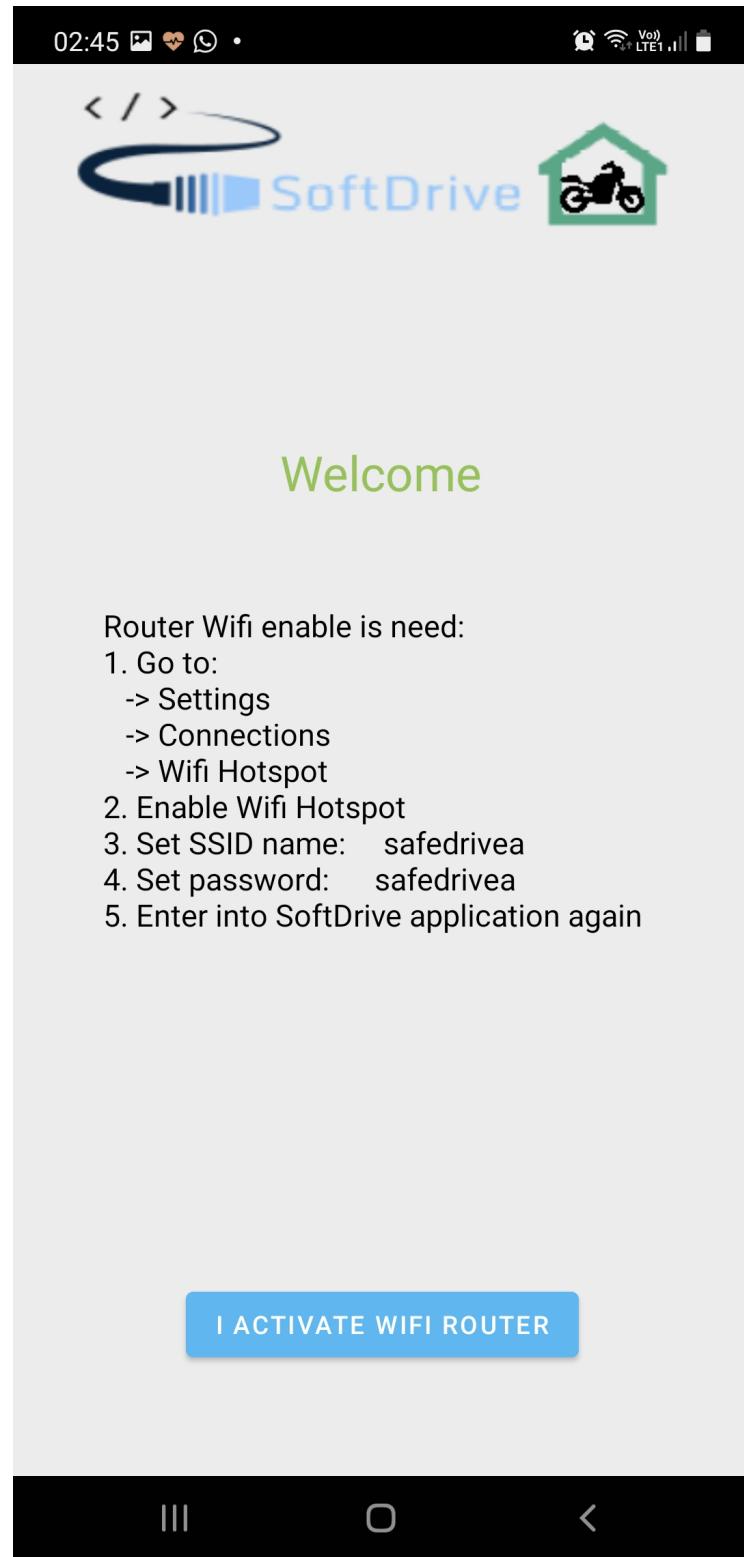
5.6.1 Aktywacja wymaganych usług

Puntem wejścia aplikacji jest klasa MainActivity, w tej klasie następuje aktywacja layoutu wymagającego od użytkownikałączenie mobilnego internetu, w przypadku wykrycia dezaktywacji tej funkcji.



Rysunek 5.42: Layout aplikacji z instrukcją uruchomienia mobilnego internetu [źródło: własne].

Gdy aktywny jest internet mobilny w urządzeniu, uruchomiona zostaje funkcjonalność klasy `EnableRouterActivity`, odpowiedzialna za wyświetlenie instrukcji konfiguracji hotspota WiFi, do którego połączy się automatycznie system czujnika odległości, w przypadku wykrycia dostępności tego punktu.



Rysunek 5.43: Layout aplikacji z instrukcją konfiguracji punktu dostępu WiFi [źródło: własne].

5.6.2 Ustalenie adresu IP serwera

Gdy router WiFi został skonfigurowany według podanej instrukcji, uruchomiona zostaje implementacja klasy RestCtrlActivity, która aktywuje notyfikację użytkownika dotyczącą wyrażenia przez niego zgodny na dostęp do sensora GPS podczas korzystania z aplikacji. Po otrzymaniu zgody od użytkownika, uruchamiana jest animacja loga aplikacji, podczas gdy w tle działającej animacji wyszukiwany jest adres IP hosta, którym jest system czujnika, w sieci lokalnej utworzonej pomiędzy modułem czujnika odległości a smartfonem. Wyszukiwanie adresu w sposób programowy jest niezbędne, ze względu na losowe dobieranie adresów przypisanych do urządzeń łączących się z hotspotem, na zasadzie działania serwera DHCP.

Początkowa implementacja na tym etapie działania nie wymagała programowego ustalenia adresu ip systemu czujnika odległości, ponieważ adres IP był przydzielony na stałe, jednorazowo w routerze WiFi. Po wprowadzeniu jednej z nowszych [obecny stan: 28.08.2021 r.] aktualizacji systemu Android, firma Google ustanowiła losowo generowany adres IP, przypisywany każdorazowo przy uruchomieniu hotspota. Wprowadzenie aktualizacji spowodowało konieczność programowego poszukiwania prawidłowego adresu. Poprawka w aktualizacji została wprowadzona w celu zwiększenia bezpieczeństwa systemu Android. Obecnie nie jest dostępne również API systemu Android, które w prosty sposób pozwalały na pobranie wartości adresu IP hosta w routerze. Czas poszukiwania adresu IP hosta ustalono na 10 sekund, według uśrednionych wyników obserwacji czasu trwania tego procesu.

Procesy poszukiwania prawidłowego adresu IP i uruchomienia GUI zostały wywołane w osobnych wątkach i działają niezależnie od siebie.

Po znalezieniu prawidłowego adresu IP urządzenia uruchomiany jest widok GUI określony klasą MainViewActivity w pakiecie view.

Listing 5.11: plik klasy RestCtrlActivity.java

```
/*
 * Check GPS permissions.
 * Initialize rest controller.
 * Run logo animation.
 * Run UI view
 * Control threads
 * @param savedInstanceState Android application Bundle
 */
@RequiresApi(api = Build.VERSION_CODES.Q)
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_ctrl_rest);

    if (ActivityCompat.checkSelfPermission(RestCtrlActivity.this,
        Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
        && ActivityCompat.checkSelfPermission(RestCtrlActivity.this,
        Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        Log.e("GPS", "NO_GPS_PERMISSION");

        ActivityCompat.requestPermissions(RestCtrlActivity.this, new String[] { android.Manifest.permission.ACCESS_FINE_LOCATION }, 100);
    }

    logoAnimation();

    e1 = Executors.newCachedThreadPool();
    Runnable rRest = new Runnable() {
        @RequiresApi(api = Build.VERSION_CODES.N)
        @Override
        public void run() {
            new GetDistanceCtrl().findRtrIpB4(1);
        }
    };

    e2 = Executors.newCachedThreadPool();
    Runnable rView = new Runnable() {
        @Override
        public void run() {
            try {
                Thread.sleep(10000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            Intent intent = new Intent(RestCtrlActivity.this, MainViewActivity.class);
            startActivity(intent);
        }
    };

    e1.execute(rRest);
    e2.execute(rView);
}
```

Poszukiwanie prawidłowej wartości adresu IP hosta jest realizowane przez klasę FindAddressIp w pakiecie rest/controllers. Ustalenie adresu rozpoczyna się od stałej wartości dwóch pierwszych bajtów adresu IP: 192.168. Następnie poszukiwany jest trzeci bajt adresu, który jest zmienny i stanowi część sieciową adresu sieci LAN, a nie hosta. Trzeci bajt adresu jest w takim przypadku wspólny dla adresu smartfona i systemu czujnika odległości. Odczytanie wartości adresu jest możliwe z poziomu polecenia: `ip addr show swlan0`. Składnia polecenia jest tożsama ze składnią poleceń systemem Linux. Na bazie jądra systemu Linux powstała komplikacja systemu Android. Otrzymane w wyniku wykonania polecenia zestawienie konfiguracji sieci jest filtrowane przez mechanizm dopasowania wzorca do tekstu: `inet 192.168.` wykorzystując metody klasy String, następnie wykorzystywany jest mechanizm wyrażeń regularnych, w celu uzyskania trzeciego bajtu adresu IP.

Listing 5.12: plik klasy FindAddressIp.java

```
/*
 * Try to find third byte of sender distance host IP address by shell command and regex pattern
 * @return Third byte from four of IP address
 */
@RequiresApi(api = Build.VERSION_CODES.N)
private String get3rdPartOfIp() {
    String ip3Byte = "";

    try {
        //shell command for show ip
        Process process = Runtime.getRuntime().exec("ip_addr_show_swlan0");
        BufferedReader in = new BufferedReader(new InputStreamReader(process.getInputStream()));
        String result = in.lines().collect(Collectors.joining());

        //cut shell command show result
        String subStr = "inet 192.168.";
        int ipEnd = result.lastIndexOf(subStr);
        String cutResult = result.substring(ipEnd + subStr.length());

        //get ip from cut shell command
        String regex = "^\\d{1,3}";
        Pattern r = Pattern.compile(regex);
        Matcher m = r.matcher(cutResult);

        if(m.find()) {
            ip3Byte = m.group(0);
            ip3Byte += ".";
            return ip3Byte;
        }
    } catch (IOException e) {
        Log.d("FinAddressIp", "ERROR_ip3byte");
        e.printStackTrace();
    }
    return null;
}
```

W pierwszej kolejności przed uruchomieniem GUI, w czasie 10 sekund poszukiwany jest adres IP czujnika odległości. Czwarty bajt adresu ip hosta jest wyszukiwany w klasie GetDistanceCtrl. Klasa uruchamia szereg 253 wywołań zapytań pod adres o znanych wartościach pierwszego, drugiego i trzeciego bajtu adresu IP. Czwarty bajt przypisany do hosta może przyjmować jedynie wartości od 2 do 254 włącznie, dlatego następuje wywołanie 253 zapytań pod adres, który w czwartym bajcie przyjmuje kolejno wartości od 2 do 254 i dla każdej z tych wartości następuje wysłanie zapytania GET protokołu HTTP. Tylko jedna wartość czwartego bajta może być prawidłowa, gdy logika klasy wykryje odpowiedź serwera usługi rest, zapisywany jest pełny adres IP tego serwera i wykorzystywany w dalszym przetwarzaniu i uzyskiwaniu danych odległości z czujnika. Gdy zapytanie skierowane na sprawdzany adres zwraca błąd, następuje zwiększenie wartości czwartego bajtu adresu o jeden i kolejne zapytanie jest kierowane na nowy adres IP. Odpytywanie adresów trwa do momentu przejścia pętli przez 253 iteracje. Każde z wywołań jest uruchamiane w osobnym wątku, w celu równoległego procesu wywołań, skracającego czas trwania poszukiwań adresu. Każde oczekiwanie na odpowiedź serwera kosztuje czas, szczególnie gdy adres serwera jest nieprawidłowy. Równoległość wywołań nie blokuje również głównego wątku aplikacji, działając w tle aplikacji.

Po uzyskaniu odpowiedzi z adresu serwera, w deklaracji wywołań zwrotnych (callback), w których są odczytywane wartości zmierzonych odległości, następuje zapis danych o odległości do bazy danych SQLite, po czym następuje interwał czasowy 250 milisekund i sekwencja odczytu odległości i zapisu jej do bazy danych jest powtarzana. Cykl odczytu i zapisu wartości odległości trwa przez cały czas działania aplikacji, jest uruchomiony w tle dzięki mechanizmowi współbieżności i wykorzystuje mechanizm wywołań zwrotnych w celu uzyskania wyniku zapytania skierowanego do serwera.

Listing 5.13: plik klasy GetDistanceCtrl.java

```

for ( ; endIp < 255 || ! isDone; ) {
    // run rest controller
    new RestDistanceCtrl(executorService, startIp, endIp, new IFromRestCallback() {

        @Override
        public void getScan2dResponse(Float[] value) {}

        @Override
        public void getDistanceResponse(Distance value) {}

        // Connected with remote distance sender, IP address found
        @Override
        public void getDistanceRouterIp(int partIpAddress) {

            byteIp4 = partIpAddress;
            executorService.shutdownNow();

            ScheduledExecutorService schedExecutor = Executors.newScheduledThreadPool(3);

            // run in continues thread iteration read from rest and write into database
            // distance value
            Runnable rtask = new Runnable() {
                @Override
                public void run() {

                    new RestDistanceCtrl(null, partIpAddress, partIpAddress, new IFromRestCallback() {

                        @Override
                        public void getScan2dResponse(Float[] value) {}

                        // get distance from sensor rest callback
                        // commit distance into database
                        @Override
                        public void getDistanceResponse(Distance value) {

                            db = new DbManager(getInstance());

                            db.setDistance(value.getDistance());
                            db.dbCommit();
                        }
                    });

                    @Override
                    public void getDistanceRouterIp(int partIpAddress) {}

                    @Override
                    public void getScan2dRouterIp(int partIpAddress) {}

                    }).prepareCall().call();
                };
            };

            // Cyclic thread for distance rest read and database write
            schedExecutor.scheduleAtFixedRate(rtask, 0, 250, TimeUnit.MILLISECONDS);
            isDone = true;
        }

        @Override
        public void getScan2dRouterIp(int partIpAddress) {}

        }).prepareCall().call();

        // control main loop with ratio
        startIp += ratio;
        endIp += ratio;
        if (endIp > 255) endIp = 255;
        if (startIp > 254) startIp = 254;
    }
}

```

5.6.3 Usługa REST API

W celu obsługi zapytań kierowanych do serwera i odpowiedzi uzyskiwanych z serwera wykorzystano bibliotekę Retrofit w wersji 2. Wykorzystanie biblioteki polega na przetwarzaniu danych uzyskanych w metodach wywołań zwrotnych w wyniku zapytania skierowanego do serwera poprzez REST API. Zapytanie jest skonfigurowane przez:

- Poprawny adres ip serwera wraz z protokołem, portem i ścieżką dostępową API.

Listing 5.14: plik interfejsu RestApi z ustawionym endpointem

```
@GET("api/distance")
Call<Distance> getDistanceEndpoint(@Header("Accept") String dataType);
```

- Mechanizm wywołań zwrotnych.

Listing 5.15: plik interfejsu IFromRestCallback przechwytyującego odpowiedzi serwera przez bibliotekę Retrofit

```
public interface IFromRestCallback {

    /**
     * Read distance from rest API
     * @param value distance domain object
     * @throws InterruptedException
     * @throws ExecutionException
     */
    void getDistanceResponse(Distance value) throws InterruptedException, ExecutionException;

    /**
     * Read fourth byte of remote host IP address
     * @param partIpAddress fourth byte of IP address
     */
    void getDistanceRouterIp(int partIpAddress);
}
```

- Implementację klasy domeny, mapującej otrzymane z serwera REST API wartości odpowiedzi.

Listing 5.16: plik interfejsu IFromRestCallback przechwytyującego odpowiedzi serwera przez bibliotekę Retrofit

```
/**
 * Distance mapping value from rest API
 * @author Dominik Stec
 */
public class Distance implements Serializable {

    @SerializedName("distance")
    @Expose
    public int distance;

    public int getDistance() {
        return distance;
    }

    public void setDistance(int distance) {
        this.distance = distance;
    }
}
```

- Logikę kontrolera klasy dziedziczącej po menadżerze biblioteki Retrofit i implementującej interfejs wywołań zwrotnych.

Listing 5.17: Klasa kontrolera usługi rest integrująca opisywane moduły

```
@RequiresApi(api = Build.VERSION_CODES.N)
public RestDistanceCtrl(Executor executor, int ipAddrStart, int ipAddrEnd, IFromRestCallback IFromRestCallback) {
    this.executor = executor;
    // callbacks initialization
    this.IFromRestCallback = IFromRestCallback;

    this.ipAddrStart = ipAddrStart;
    this.ipAddrEnd = ipAddrEnd;
    fourthIp = ipAddrStart;
    // full ip address preparation
    prepareIp(fourthIp);

    distance = new Distance();
}
```

```

/*
 * Concat parts of IP address for set full IP address
 * @param fourthIp four part of full IP address
 * @return this class reference to this class object
 */
@RequiresApi(api = Build.VERSION_CODES.N)
public RestDistanceCtrl prepareIp(int fourthIp) {
    ip = new FindAddressIp();
    thirdPartIp = ip.getIp();

    DISTANCE_URL = "";
    DISTANCE_URL += protocol;
    DISTANCE_URL += thirdPartIp;
    DISTANCE_URL += String.valueOf(fourthIp);
    DISTANCE_URL += portDistance;

    return this;
}

/**
 * Initialization Retrofit library components
 * @return this class reference to this class object
 */
public RestDistanceCtrl prepareCall() {
    start();
    return this;
}

/**
 * Start read data by rest callback
 */
public void call() {
    Call<Distance> call = restApiDistance.getDistanceEndpoint("application/json");
    call.enqueue(this);
}

/**
 * Read response from rest API and transfer it by callback interface
 * @param call object for manipulate call properties
 * @param response object for store response data from rest API
 */
@RequiresApi(api = Build.VERSION_CODES.N)
@Override
public void onResponse(Call<Distance> call, Response<Distance> response) {
    Log.v("Response_callback", "call_callback");

    // parse response to JSON
    getGson().toJson(response.body());

    if (response.isSuccessful()) {
        // run with thread
        if (executor != null) {
            executor.execute(new Runnable() {
                @RequiresApi(api = Build.VERSION_CODES.N)
                @Override
                public void run() {
                    // prepare correct ip of remote host with rest api
                    String ip3b = prepareIp(fourthIp).ip.getIp();

                    // set callback for fourth byte of correct IP address
                    IFromRestCallback.getDistanceRouterIp(fourthIp);
                    try {
                        // set callback for read distance from rest API remote host
                        IFromRestCallback.getDistanceResponse(response.body());
                    } catch (InterruptedException | ExecutionException e) {
                        e.printStackTrace();
                    }
                }
            });
            call.cancel();
        }
        return;
    }
}

// run without thread
} else {
    String ip3b = prepareIp(fourthIp).ip.getIp();
    IFromRestCallback.getDistanceRouterIp(fourthIp);

    try {
        IFromRestCallback.getDistanceResponse(response.body());
    } catch (InterruptedException | ExecutionException e) {
        e.printStackTrace();
    }

    call.cancel();
}

return;
}

/**
 * Run if IP address not exist
 * @param call object for manipulate call properties
 * @param t exception handler
 */

```

```

/*
@RequiresApi(api = Build.VERSION_CODES.N)
@Override
public void onFailure(Call<Distance> call, Throwable t){
    Log.e("try_find_IP", "IP_not_exist");

    // hold call reference for next IP address read try
    call = call.clone();

    // prepare for call rest API
    setDistancePartialUrl(String.valueOf(fourthIp));
    updateDistanceRetrofit();

    // renew with new IP address value
    fourthIp++;

    // end of ip address range
    if(fourthIp > this.ipAddrEnd) {
        call.cancel();
        return;
    }

    // prepare for call rest API and call
    this.prepareIp(fourthIp).prepareCall().call();
}

```

5.6.4 Baza danych

Obsługę bazy danych zapewnia menadżer z klasy DbManger.java udostępniając metody commitowania danych o odległości do bazy i odczytu ich wartości wraz z mechanizmem zapobiegającym przepełnieniu bazy danych, zwiększać wydajność jej obsługi. Metoda pobierająca dane zwraca w wyniku najbardziej aktualną wartość odległości. Po osiągnięciu 200 rekordów zapisu danych w bazie następuje czyszczenie bazy danych. Baza danych zapewnia mechanizm zgłaszania błędów poprzez mechanizm oznaczeń kodowych.

Listing 5.18: Klasa kontrolera bazy danych

```

public boolean dbCommit() {

    // Gets the data repository in write mode
    SQLiteDatabase dbWrite = dbHelper.getWritableDatabase();

    // Create a new map of values, where column names are the keys
    ContentValues values = new ContentValues();
    values.put(CreateTable.TableSensorData.COLUMN_NAME_DISTANCE, distance);
    //values.put(CreateTable.TableSensorData.COLUMN_NAME_SCAN_2D, scan2d);

    // Insert the new row, returning the primary key value of the new row
    long newRowId = dbWrite.insert(CreateTable.TableSensorData.TABLE_NAME, null, values);
    Log.v("database", "insert_new_row");

    // clear database for more efficiency
    if (getRowCount(dbWrite) > MAX_ROW_COUNT) {
        clearDb(dbWrite);
        ContentValues valuesDel = new ContentValues();
        valuesDel.put(CreateTable.TableSensorData.COLUMN_NAME_DISTANCE, -1);
        dbWrite.insert(CreateTable.TableSensorData.TABLE_NAME, null, valuesDel);
        Log.v("database", "clear_database_rows");
    }

    dbWrite.close();

    return true;
}

/**
 * Clear database commit entries
 * @param dbWritableDatabase SQLite database object
 */
private void clearDb(SQLiteDatabase dbWritableDatabase) {
    dbWritableDatabase.execSQL("delete_from_" + CreateTable.TableSensorData.TABLE_NAME);
}

/**
 * Numbers of inserted rows getter
 * @param dbWritableDatabase SQLite database object
 * @return Actual number of rows
 */
public long getRowCount(SQLiteDatabase dbWritableDatabase) {
    long count = DatabaseUtils.queryNumEntries(dbWritableDatabase, CreateTable.TableSensorData.TABLE_NAME);
    return count;
}

/**
 * Latest value of written distance from table getter
 * @return Newest value of distance
 * @throws InterruptedException
 */
public int getDbDistance() throws InterruptedException {
    SQLiteDatabase dbRead = dbHelper.getReadableDatabase();

    String[] projection = {

```

```

CreateTable.TableSensorData.COLUMN_NAME_DISTANCE
};

Cursor cursor = dbRead.query(
CreateTable.TableSensorData.TABLE_NAME,
projection,           // The array of columns to return (pass null to get all)
null,                // The columns for the WHERE clause
null,                // The values for the WHERE clause
null,                // don't group the rows
null,                // don't filter by row groups
null
);

int distance = -2;

try {
// retry cursor read data
int count = 0;
while(! cursor.moveToLast()) {
Thread.sleep(50);
count++;
if(count>20){
// if too much retries
cursor.close();
dbRead.close();
return -4;
}
}
// read distance from db
distance = cursor.getInt(cursor.getColumnIndex(CreateTable.TableSensorData.COLUMN_NAME_DISTANCE));
cursor.close();
} catch (CursorIndexOutOfBoundsException e) {
distance = -3;
}

dbRead.close();

return distance;
}

```

5.6.5 Graficzny interfejs użytkownika

Moduł graficznego interfejsu użytkownika uruchomiony jest w głównym wątku aplikacji, w celu uniknięcia błędów wynikłych z synchronizacją z usługami obsługi bazy danych lub odpytywania serwera o wartość aktualnej odległości. Zarządzanie usługą widoku jest zaimplementowane w klasie UiView. Klasa jest odpowiedzialna za rysowanie elementów graficznych interfejsu: motocyklistę, samochód przed motocyklistą, wskazania prędkości i odległości oraz tło. Logika dla rysowanego samochodu jest przechowywana w klasie ForwardVehicle, natomiast dla motocykla w klasie Motorcycle. Gdy wartość prędkości lub odległości ulegnie zmianie powinna nastąpić aktualizacja stanu klasy i ponowne rysowanie elementów graficznych. Rysowanie elementów na kanwie powinno odbywać się asynchronicznie w sposób prawie równoległy z synchronizacją wątków. Widok powinien pobierać wartości prędkości z bazy danych, zabezpieczając się obsługą błędów w przypadku wczytania nieprawidłowej wartości odległości. W klasie analizowana jest wartość prędkości poruszania się pojazdu oraz odległość od pojazdu jadącego z przodu w celu monitowania kierowcy o zbyt bliskiej odległości lub zbyt dużej prędkości jazdy w przypadku konieczności nagłego hamowania pojazdem. Każda zmiana odległości współrzędnych GPS powinna podlegać detekcji w celu pobrania aktualnej wartości prędkości jazdy motocyklisty. Naciśnięcie na ekran powinno spowodować zamknięcie aplikacji i powrót do ustawień domyślnych wyświetlacza urządzenia. Klasa Display pobiera aktualne wartości rozmiaru ekranu, na bazie których odbywa się skalowanie i rysowanie elementów kanwy. Implementacja interfejsu powstała na podstawie testów na wyświetlaczu smartfona Samsung Galaxy Note 10 Plus.

Listing 5.19: Klasa kontrolera GUI

```

/**
* View controller
* @author Dominik Stec
* @see SurfaceView
* @see SurfaceHolder.Callback
* @see Display
* @see ForwardVehicle
* @see Motorcycle
* configuration base come from following
* @link https://www.pearson.com/uk/educators/higher-education-educators/program/Deitel-Android-How-to-Program-International
*/
public class UiView extends SurfaceView implements SurfaceHolder.Callback {

/**
* Initialize object constants and values

```

```

/*
protected synchronized void initConstant() {
this.width = (int) (FORWARD_VEHICLE_WIDTH_PERCENT * displayWidth);
this.height = (int) (FORWARD_VEHICLE_HEIGHT_PERCENT * displayHeight);
this.motorHeight = (int) (MOTORCYCLE_HEIGHT_PERCENT * displayHeight);
textPaint.setTextSize((int) (TEXT_SIZE_PERCENT * screenHeight));
backgroundPaint.setColor(Color.GRAY);
tooFastAlarmPaint.setColor(Color.YELLOW);
ptConnAlert.setColor(Color.WHITE);
ptConnAlert.setTextSize(120);
}

/**
* Set distance of forward vehicle read from database
* @throws InterruptedException
*/
protected synchronized void updateActualDistance() throws InterruptedException {
distRegulator = db.getDbDistance() > 0 ? db.getDbDistance() : distRegulator;
forwardDist = distRegulator < 4000 ? distRegulator : 4000;
Log.v("car_distance_regulate", forwardDist + " meter");
}

/**
* Set actual motorcycle driver speed
* @param motorcycleSpeed speed of driver move
*/
protected synchronized void updateActualSpeed(float motorcycleSpeed) {
this.speed = motorcycleSpeed;
Log.v("actual_speed", speed + " m/s");
}

/**
* Set screen background for objects
* @param canvas draw object space
*/
protected synchronized void drawBackground(Canvas canvas) {
canvas.drawRect(0, 0, canvas.getWidth(), canvas.getHeight(), backgroundPaint);
}

/**
* Draw motorcycle object on canvas
* @param canvas draw object space
*/
protected synchronized void drawMotorcycle(Canvas canvas) {
motor = new Motorcycle(getContext(),
canvas,
(int) (MOTORCYCLE_WIDTH_PERCENT * displayWidth),
(int) (MOTORCYCLE_HEIGHT_PERCENT * displayHeight));
}

/**
* Forward vehicle initialize configuration
*/
protected synchronized void initForwardVehicle() {
int xCenter = displayWidth / 2 - width / 2;

forwardVehicle = new ForwardVehicle(
getContext(),
this,
Color.GREEN,
0,
xCenter,
10,
width,
height,
0,
(float) FORWARD_VEHICLE_SPEED_PERCENT * displayHeight
);
}

/**
* Show to big speed or to low distance alert
* @param canvas draw object space
*/
protected synchronized void drawSpeedAlert(Canvas canvas) {

boolean isTooFast = false;
if (speed > 0) isTooFast = isTooFast(speed, forwardDist/100);

if (isTooFast) {
int safeSpeed = getSafeSpeed(forwardDist/100);
tooFastAlarmPaint.setTextSize(140);
if(safeSpeed > 0 && toKmh(speed) - safeSpeed > 5) canvas.drawText("reduce:" + safeSpeed + " km/h", 30, 1600, tooFastAlarmPaint);
}
}

/**
* Check if speed or distance is dangerous
* @param speed speed of driver move
* @param distance distance between driver and forward vehicle
* @return true if situation is risk or false if it is safe
*/
protected boolean isTooFast(float speed, int distance) {
boolean isTooFast = false;

// 9 m/s^2
float breakLatency = 9f;
// V = at
float breakTime = speed / breakLatency;
// S = Vt
}

```

```

int breakDistance = (int) (speed * breakTime);
if (breakDistance > distance) isTooFast = true;

return isTooFast;
}

/**
* Calculate safe speed of move according to forward vehicle distance
* @param distance distance between driver and forward vehicle
* @return safe speed value
*/
protected int getSafeSpeed(int distance) {
// 9 m/s^2
float breakLatency = 9f;
// S = at^2 / 2
// 2S = at^2
// t = sqrt(2S / a)
float breakTime = (float) Math.sqrt((2 * distance) / breakLatency);
// V = S/t
if(breakTime == 0) return 0;
float safeSpeed = distance / breakTime;
safeSpeed = Math.round(safeSpeed);

return toKmh(safeSpeed);
}

/**
* Set position and draw forward vehicle
* @param canvas draw object space
*/
@RequiresApi(api = Build.VERSION_CODES.O)
protected synchronized void updateCarPosition(Canvas canvas) {
forwardVehicle.updateForwardVehiclePosition(forwardDist, motor.getyCoord(), motor.getHeight());
forwardVehicle.draw(canvas);

Log.v("car-distance-rest:", forwardDist + "m");
}

/**
* Draw actual speed on screen
* @param canvas draw object space
* @param text speed value to draw
*/
protected synchronized void drawSpeedMeter(Canvas canvas, String text) {
canvas.drawText("V:" + text + "km/h", 50, 100, textPaint);
}

/**
* Draw actual position between driver and forward vehicle
* @param canvas draw object space
* @throws InterruptedException
*/
protected synchronized void drawPositionMeter(Canvas canvas) throws InterruptedException {
canvas.drawText("dist:" + forwardDist, 600, 100, textPaint);

if (db.getDbDistance() < 0) {
++count;
if (count > 20) {
canvas.drawText(".. car-detection ..", 120, 300, ptConnAlert);
forwardVehicle.draw(canvas);
}
} else {
count = 0;
}
}

/**
* Called when surface is first created and start user interface view thread
* @param holder surface object holder
*/
@RequiresApi(api = Build.VERSION_CODES.O)
@Override
public void surfaceCreated(SurfaceHolder holder) {
Handler mHandler;

mHandler = new Handler(Looper.getMainLooper()) {
@Override
public void handleMessage(Message message) {
hideSystemBars();
uiViewThread = new UiViewThread(holder); // create thread
uiViewThread.start(); // start the game loop thread
};

executor.execute(new Runnable() {
@Override
public void run() {

Message message = mHandler.obtainMessage(1, "UI-handle");
message.sendToTarget();
});
};

}

@Override
public void onSizeChanged(int w, int h, int oldw, int oldh) {
}

// called when surface changes size
@Override
public void surfaceChanged(SurfaceHolder holder, int format,

```

```

int width, int height) {
}

// called when the surface is destroyed
@Override
public void surfaceDestroyed(SurfaceHolder holder) {

    /**
     * Called when the user touches the screen in this activity and exit application
     * @param e touch screen event handler
     * @return true if detect touch screen
     */
    @Override
    public boolean onTouchEvent(MotionEvent e) {
        ExitApp(e);
        return true;
    }

    /**
     * Parse from m/s to km/h
     * @param speed speed in m/s to parse
     * @return parsed speed in km/h
     */
    protected int toKmh(float speed) {
        return speed > 0 ? (int) (speed *= (0.001f / (1f / 3600f))) : (int) speed;
    }

    /**
     * hide system bars and app bar
     */
    protected void hideSystemBars() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT)
            setSystemUiVisibility(
                View.SYSTEM_UI_FLAG_LAYOUT_STABLE |
                View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION |
                View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN |
                View.SYSTEM_UI_FLAG_HIDE_NAVIGATION |
                View.SYSTEM_UI_FLAG_FULLSCREEN |
                View.SYSTEM_UI_FLAG_IMMERSIVE);
    }

    /**
     * Close application preparation
     * @param event touch screen event handler
     */
    protected void ExitApp(MotionEvent event) {
        executor.shutdownNow();
        schedExecutor.shutdownNow();

        Intent i = new Intent(getApplicationContext(), Exit.class);
        i.putExtra("EXTRA_EXIT", true);
        i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);
        getApplicationContext().startActivity(i);
    }

    /**
     * Thread subclass to control the UI view loop
     * @author Dominik Stec
     * @see IBaseGpsListener
     * configuration base come from following
     * @link https://www.pearson.com/uk/educators/higher-education-educators/program/Deitel-Android-How-to-Program-International
     */
    private class UiViewThread implements IBaseGpsListener{

        private SurfaceHolder surfaceHolder; // for manipulating canvas

        private ExecutorService es;
        private CLocation myGpsLocation;

        /**
         * initializes the surface holder
         * @param holder hold surface object
         */
        @RequiresApi(api = Build.VERSION_CODES.O)
        public UiViewThread(SurfaceHolder holder) {
            surfaceHolder = holder;

            initCLocation();

            schedExecutor = Executors.newScheduledThreadPool(20);
            es = Executors.newCachedThreadPool();

            nCurrentSpeed = -10f;
        }

        /**
         * Run main UI view thread with synchronized methods for control UI states
         */
        public void start() {

            Runnable rtask = new Runnable() {

                @RequiresApi(api = Build.VERSION_CODES.O)
                @Override
                public void run() {

                    Canvas canvas = null; // used for drawing

                    try {
                        // get Canvas for exclusive drawing from this thread

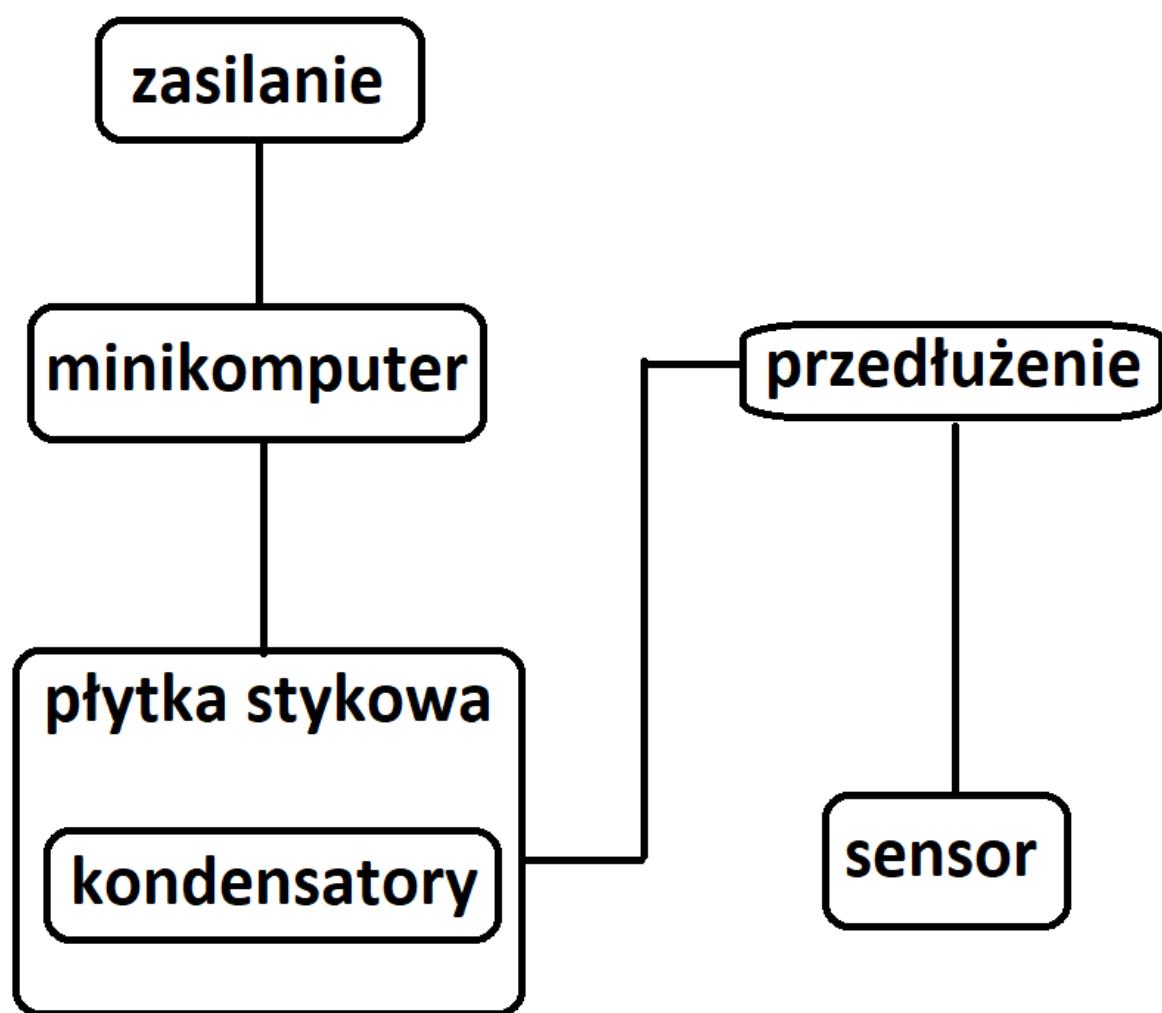
```


5.7 Prototyp części sprzętowej

5.7.1 model

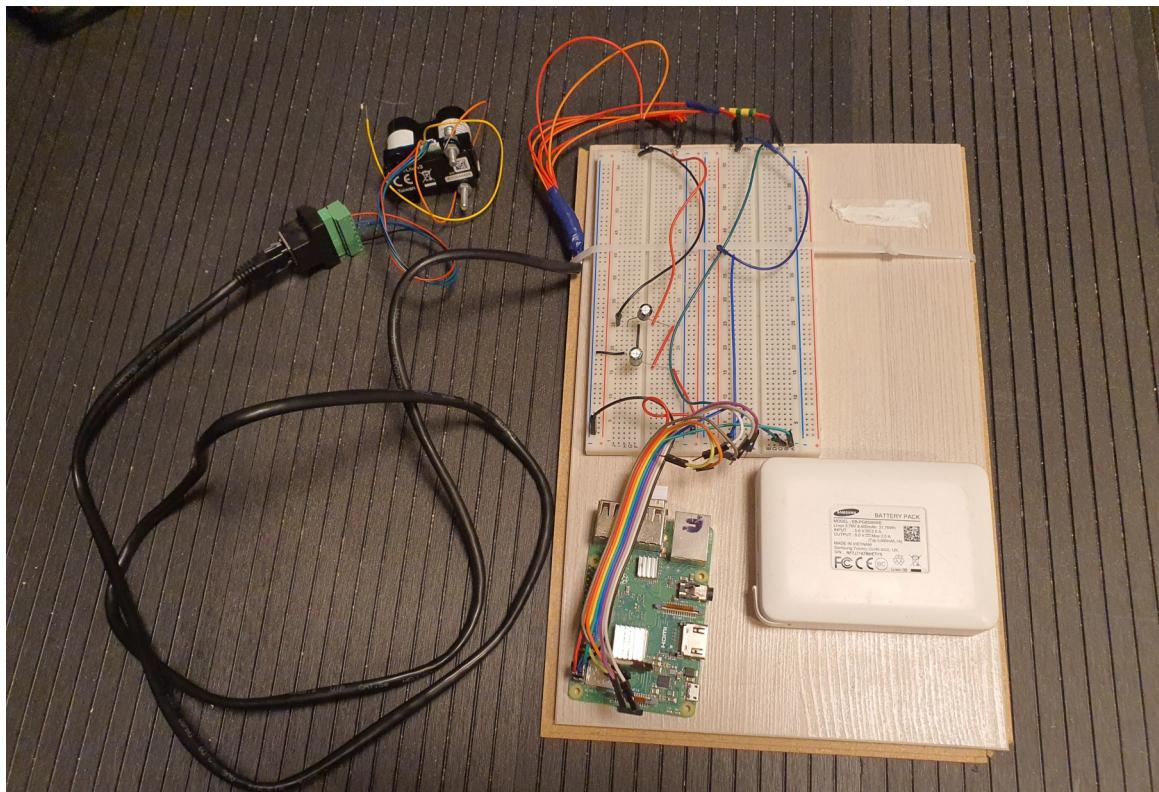
W celu konstrukcji prototypu wykorzystano model, na którym:

- uwzględniono zasilanie napięciem DC z baterii
- umieszczono minikomputera
- doprowadzono zasilania minikomputera
- podłączono czujnik przed skrętkę sieciową, w celu możliwie elastycznego montażu samego czujnika, w oderwaniu od pozostałych modułów prototypu
- podłączono czujnik do minikomputera poprzez warstwę pośrednią, składającą się z płytki stykowej z kondensatorami
- zrealizowano połączenie zasilania pomiędzy sensorem a minikomputerem przebiegającego przez równolegle połączone kondensatory w celu dobrania prawidłowej wartości pojemności, zgodnej z notą katalogową sensora.

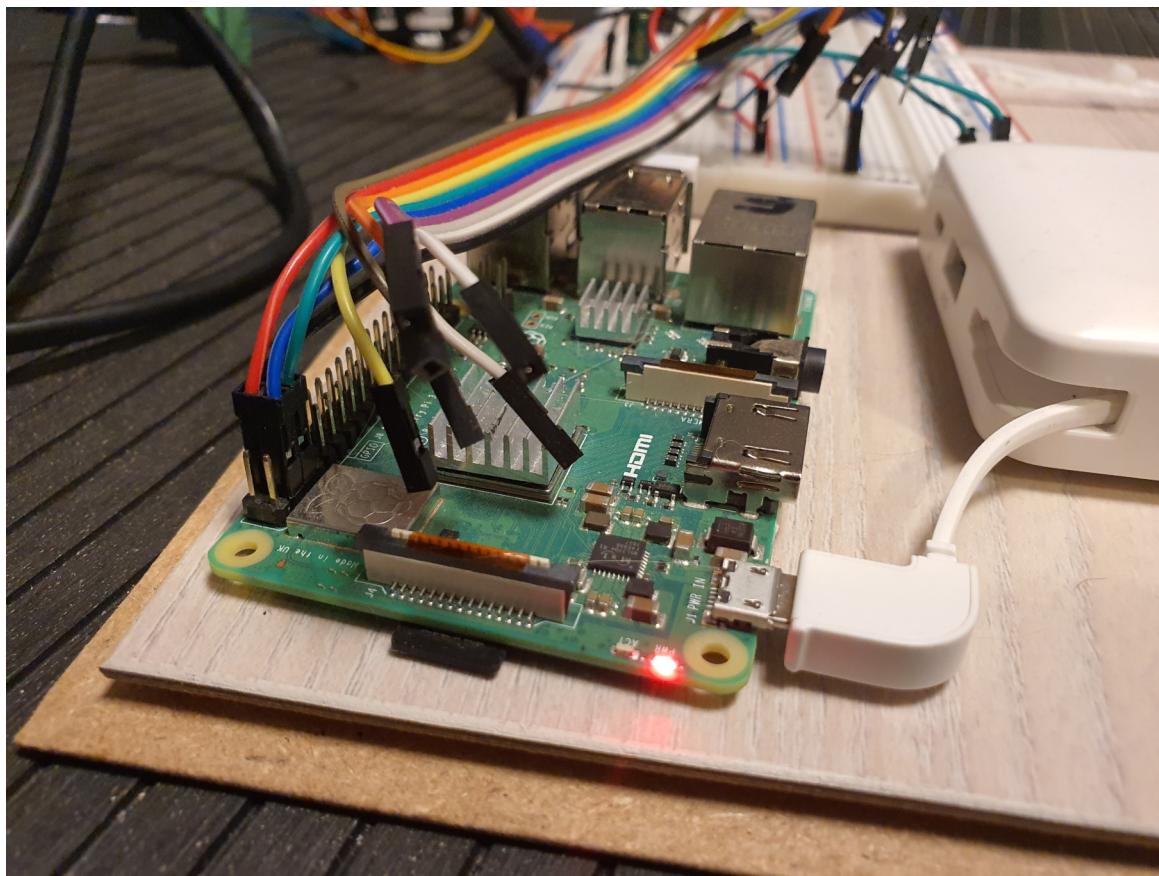


Rysunek 5.44: Projekt prototypu części sprzętowej [źródło: własne].

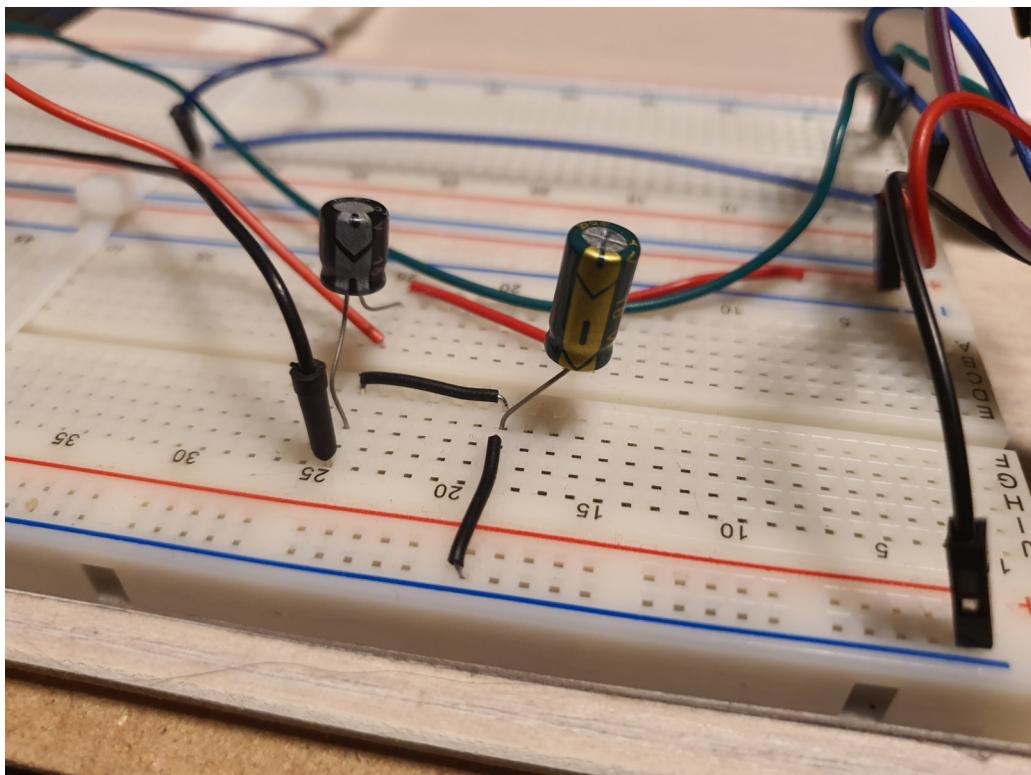
5.7.2 Wdrożenie



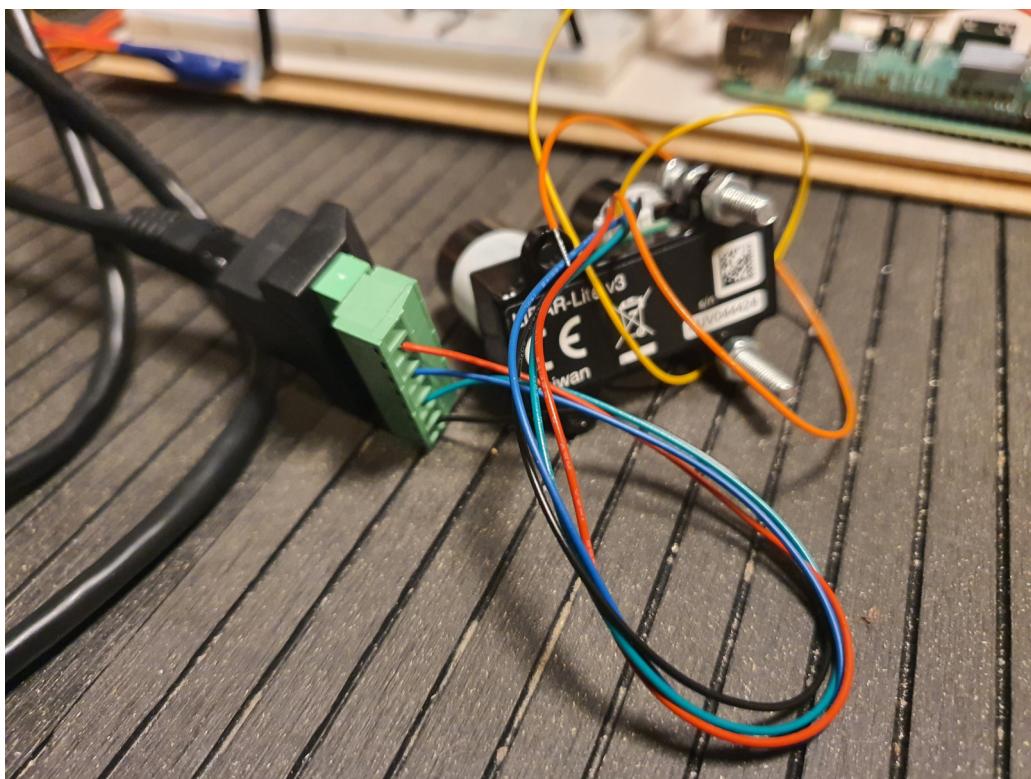
Rysunek 5.45: Konstrukcja prototypu części sprzętowej, widok całości [źródło: własne].



Rysunek 5.46: Konstrukcja prototypu części sprzętowej, widok połączeń [źródło: własne].

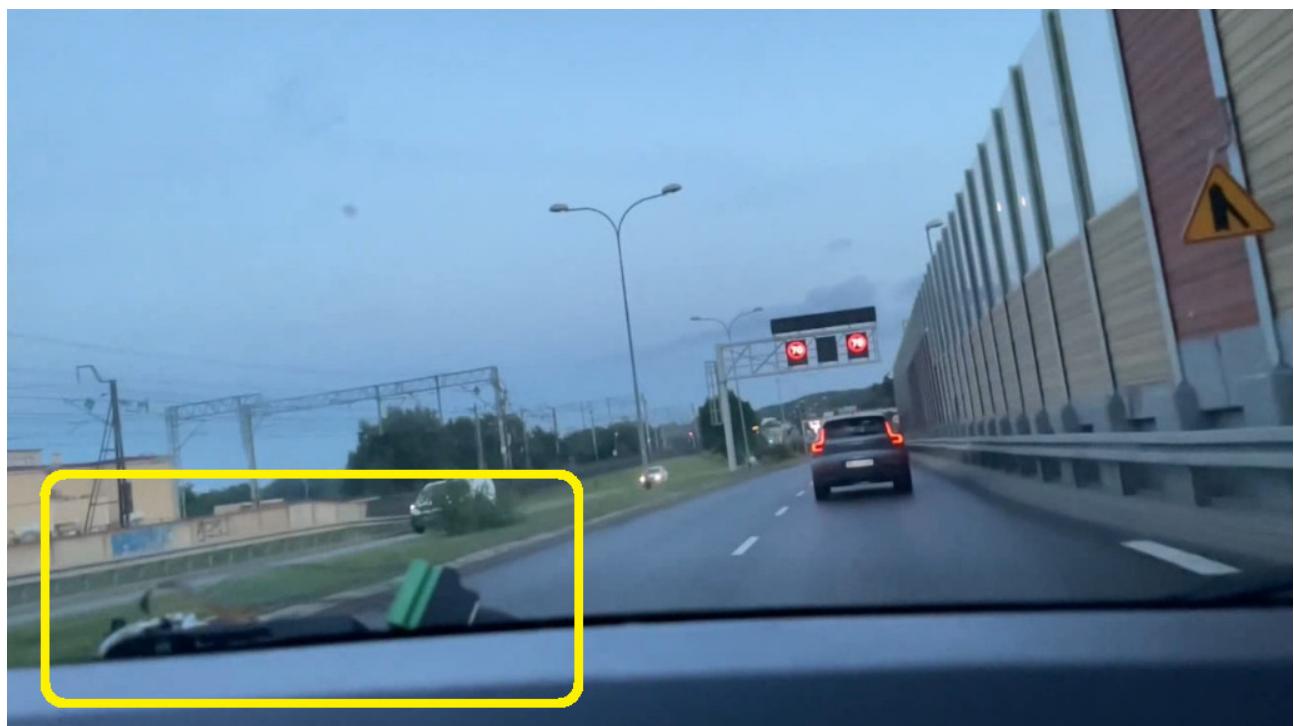


Rysunek 5.47: Konstrukcja prototypu części sprzętowej, widok kondensatorów [źródło: własne].



Rysunek 5.48: Konstrukcja prototypu części sprzętowej, widok czujnika z przedłużeniem połączenia [źródło: własne].

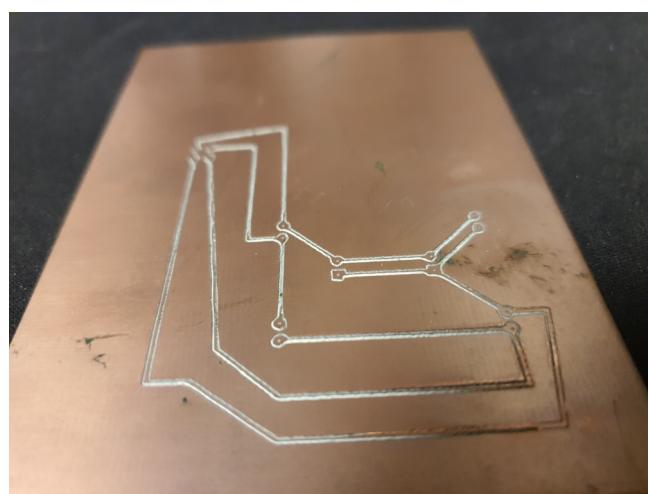
Instalacja czujnika na zewnątrz pojazdu z zastosowaniem przedłużenia sięgającego do środka pojazdu, łącząc czujnik z resztą prototypu.



Rysunek 5.49: Instalacja czujnika na zewnątrz pojazdu [źródło: własne]

5.8 Laminat PCB

Możliwy rozwój projektu przewiduje jego montaż na pojeździe w formie obudowanego urządzenia, jednym z etapów prac nad projektem było wygrawerowanie ścieżek w laminacie PCB z wykorzystaniem grawerki CNC sterowanej oprogramowaniem, zgodnie z konfiguracją połączeń w prototypie. Przewidywany dalszy rozwój projektu zakłada wykorzystanie laminatu w celach obudowania prototypu.



Rysunek 5.50: Laminat PCB z wygrawerowanymi w technologii CNC ścieżkami

We wdrożeniu opisano dodatkowo fragmenty realizacji pracy nie wchodzącej w zakres projektu, która możliwa była do zrealizowania w skutek posiadanych dodatkowych zasobów czasowych w celu zachęcenia i ułatwienia przyszłego rozwoju funkcjonalności produktu w poszerzonym zakresie.

Rozdział 6

Testowanie rozwiązań

Proces testowania rozwiązań przeprowadzony podczas realizacji projektu uwzględnia zasadę zgodności z piramidą testów, według której największe pokrycie przypadkami testowymi powinno dotyczyć testowania kodu aplikacji, następnie w mniejszej ilości powinny być uwzględnione testy integracyjne, na szczytce piramidy testów są testy użyteczności i interfejsu.

Ze względu na brak możliwości interakcji użytkownika z systemem poprzez GUI, zrezygnowano z realizacji testów interfejsu.

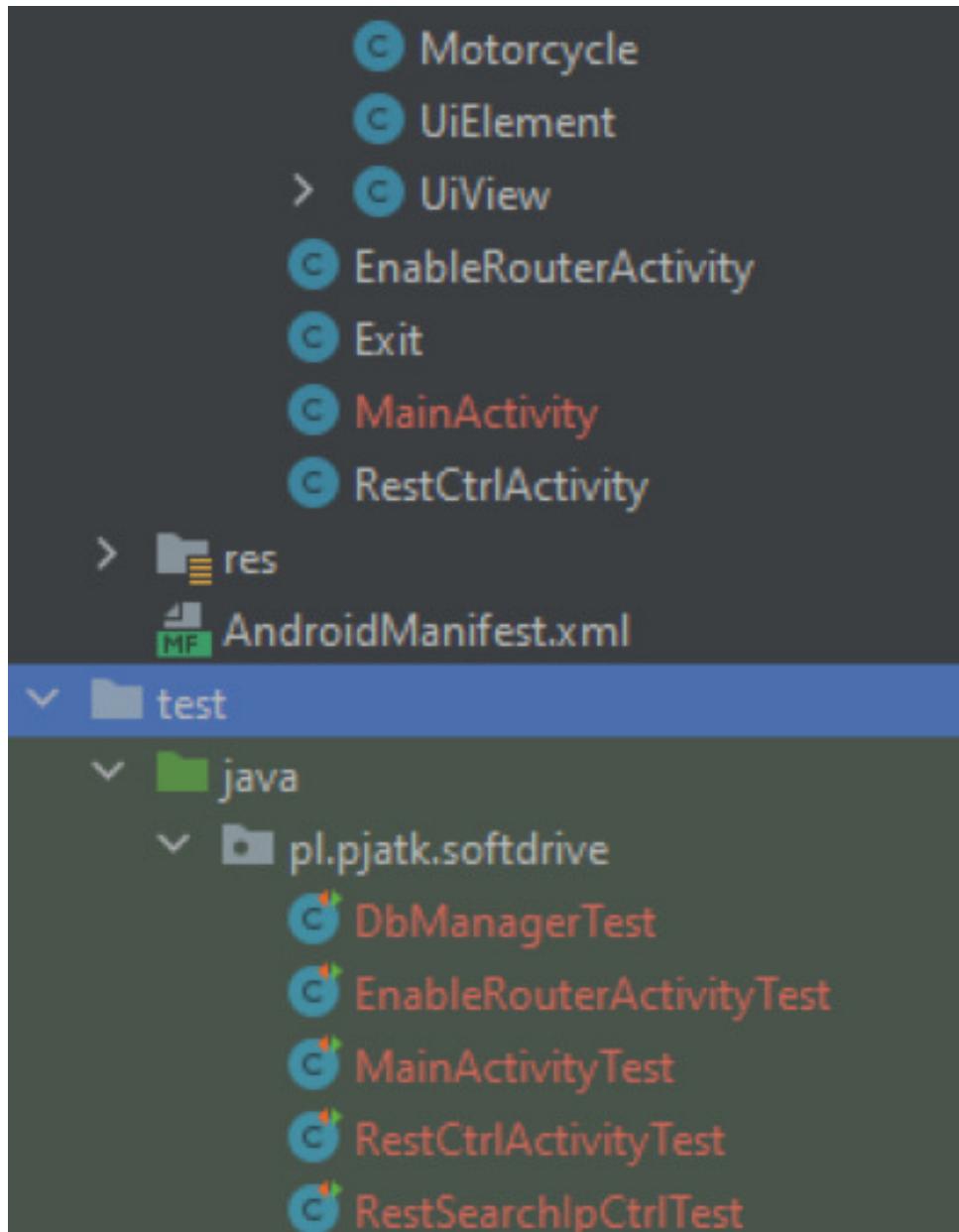
Dodatkowo specyfika realizowanego projektu dotycząca modularyzacji architektury na części sprzętowej i programowej, które pozostają w ścisłej zależności pod względem funkcjonalnym, była powodem syntetycznego ujęcia przypadków testów integracyjnych i użyteczności, w formie analizy zachowania systemu w rzeczywistych warunkach ruchu drogowego.

Dzięki zastosowanemu podejściu możliwe było uchwycenie błędów komunikacji pomiędzy modułami systemu, zgodnie z ideą testów integracyjnych, uwidaczniające się podczas testowania prototypu w kontekście jego użytkowania. Dodatkowo przeprowadzona została analiza użyteczności systemu pod kątem założonej funkcjonalności.

W celu przetestowania implementacji przeprowadzono testy jednostkowe. W celu przeprowadzenia testów użyteczności i integracyjnych, wykonano analizę zachowania systemu w rzeczywistych warunkach ruchu drogowego.

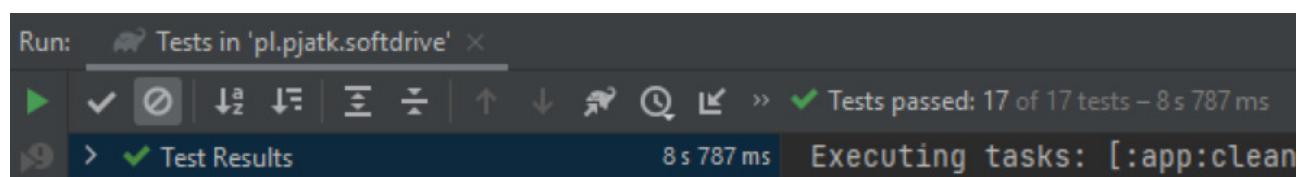
6.1 Testy jednostkowe

Przeprowadzone testy jednostkowe kodu aplikacji umożliwiły zmniejszenie liczby potencjalnych błędów, trudnych do uchwycenia podczas pisania kodu. Dodatkowo podejście jednoczesnego tworzenia implementacji i przypadków testowych umożliwiło zachowanie elastycznej struktury kodu. Podczas tworzenie implementacji i jednocześnie testów zwracano uwagę na szczegóły implementacji dotyczące stosowania dobrych praktyk i umożliwiające lepszą adaptację oraz rozbudowę istniejącego kodu. Napisane testy jednostkowe ułatwili dodatkowo lepsze zrozumienie zależności występujących w poszczególnych funkcjonalnych fragmentach jednostek kodu i pomiędzy tymi fragmentami. Wymienione aspekty realizacji i funkcjonalne tworzonej implementacji, mogłyby zostać pominięte, podczas rezygnacji z wdrożenia testów jednostkowych. Przypadki testowe w strukturze projektowanego systemu.



Rysunek 6.1: Struktura testów w projekcie [źródło: własne]

Pozytywny rezultat przeprowadzonych testów w środowisku Android Studio.



Rysunek 6.2: Wynik uruchomienia testów [źródło: własne]

6.1.1 Testyinicjalizacji systemu

Testy zapewniają mechanizm dostępu do metod prywatnych. Sprawdzona zostaje poprawność wykrywania aktywacji usługi internetu mobilnego. Następuje korekta działania metody wyświetlającej instrukcję użytkownikowi. Sprawdzona jest poprawność wyświetlania informacji o braku aktywacji internetu mobilnego, w przypadku próby potwierdzenia aktywacji. Poddano próbce prawidłowość aktywacji kolejnej aktywności po naciśnięciu przycisku.

Listing 6.1: Testy dla klasy MainActivity

```
@Test
public void whenMobileDataOnCheckMethod_thenReturnFalse() throws NoSuchMethodException, InvocationTargetException, IllegalAccessException
//GIVEN
Method privateisMobileDataEnabledFromLollipop = MainActivity.class.getDeclaredMethod("isMobileDataEnabledFromLollipop", Context.class);
privateisMobileDataEnabledFromLollipop.setAccessible(true);
//WHEN
boolean mobDataIsEnabled = (boolean) privateisMobileDataEnabledFromLollipop.invoke(activity, activity.getApplicationContext());
//THEN
assertFalse(mobDataIsEnabled);
}

@Test
public void whenContainsInstructionStringProperText_thenReturnTrue() throws NoSuchMethodException, InvocationTargetException, IllegalAccessException
//GIVEN
Method privategetInstructionString = MainActivity.class.getDeclaredMethod("getInstructionString", null);
privategetInstructionString.setAccessible(true);
//WHEN
String info = (String) privategetInstructionString.invoke(activity, null);
//THEN
assertTrue(info.contains("2. Enable mobile network data"));
}

@Test
public void whenInfoAlertInitialiseProper_thenReturnTrue() throws NoSuchMethodException, InvocationTargetException, IllegalAccessException
//GIVEN
Method privategetInfoAlertString = MainActivity.class.getDeclaredMethod(" getInfoAlertString", null);
privategetInfoAlertString.setAccessible(true);
//WHEN
String info = (String) privategetInfoAlertString.invoke(activity, null);
activity.initialiseInfoPrequisitions("?", info);
//THEN
assertTrue(activity.getMobileDataDisableAlert().contains("Mobile data are disable"));
}

@Test
public void whenBtnClickStartsNewActivity_givenBtnAndNewActivityReferences_thenStartNewActivity() {
//GIVEN
Intent expectedIntent = new Intent(activity.getApplicationContext(), EnableRouterActivity.class);
Button btn = (Button) activity.findViewById(R.id.data_enable_btn);
//WHEN
btn.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
activity.startActivity(expectedIntent);
}
});
btn.performClick();
Intent actual = shadowOf(activity.getApplication()).getNextStartedActivity();
//THEN
assertEquals(expectedIntent.getComponent(), actual.getComponent());
}
```

Sprawdzanie poprawności metody wykrywającej aktywację routera Wifi. Poddano próbie prawidłowość startu aktywności serwisu REST. Sprawdzono działanie wyświetlanego użytkownikowi informacji o braku aktywacji access pointa, po próbie potwierdzenia aktywacji przez użytkownika, w stanie jego dezaktywacji, pomimo nadania aplikacji uprawnienie do zasobów wymaganych do aktywacji.

Listing 6.2: Testy dla klasy EnableRouterActivity

```

@Test
public void whenCheckAccessPointIsEnable_givenAccessPointDisable_thenReturnFalse() throws NoSuchMethodException, InvocationTargetException, IllegalAccessException {
    //GIVEN
    Method privateisApEnable = EnableRouterActivity.class.getDeclaredMethod("isApEnable", null);
    privateisApEnable.setAccessible(true);
    //WHEN
    boolean isApEnable = (boolean) privateisApEnable.invoke(activity, null);
    //THEN
    assertFalse(isApEnable);
}

@Test
public void whenExpectedNewActivity_givenStartNewActivityMethod_thenStartRestCtrlActivity() {
    //GIVEN
    Intent expectedIntent = new Intent(activity.getApplicationContext(), RestCtrlActivity.class);
    //WHEN
    activity.startNextActivity();
    Intent actual = shadowOf(activity.getApplication()).getNextStartedActivity();
    //THEN
    assertEquals(expectedIntent.getComponent(), actual.getComponent());
}

@Test
public void whenClickButtonWithRouterOff_givenGrantAccessPermissions_thenShowInfoAlert() {
    //GIVEN
    Button btn = activity.findViewById(R.id.ap_enable_btn);
    ShadowApplication app = Shadows.shadowOf(activity.getApplication());
    app.grantPermissions(Manifest.permission.ACCESS_WIFI_STATE, Manifest.permission.ACCESS_NETWORK_STATE, Manifest.permission.READ_PHONE_STATE, Manifest.permission.ACCESS_WIFI_STATE, Manifest.permission.INTERNET);
    //WHEN
    btn.performClick();
    //THEN
    TextView txt = activity.findViewById(R.id.ap_data_dis);
    assertEquals(txt.getText().toString(), activity.getInfoAlertDefault());
}

```

6.1.2 Testy usługi rest

Testy umożliwiają sprawdzenie działania mechanizmu współpracy. Sprawdzono potwierdzenie przyznania dostępu do zasobu lokalizacji. Skontrolowano, czy zostały uruchomione w tle aplikacji, wątki: uruchamiające usługę wykrywania adresu ip modułu czujnika, odczytywania i zapisu odległości, inicjalizacji i aktualizacji widoku.

Listing 6.3: Testy dla klasy RestCtrlActivity

```
@Test
public void whenGrantLocationAccess_givenApplicationState_thenConfirmLocationAccess() {
    //GIVEN
    ShadowApplication app = Shadows.shadowOf(activity.getApplication());
    //WHEN
    app.grantPermissions(Manifest.permission.ACCESS_COARSE_LOCATION, Manifest.permission.ACCESS_FINE_LOCATION);
    //THEN
    assertTrue(activity.isLocationGrant());
}

@Test
public void whenExecuteThread_givenThreadInBackground_thenWorkInBackground_v1_IP() throws InterruptedException {
    //GIVEN
    int numberOfThreads = 1;
    ExecutorService service = Executors.newFixedThreadPool(1);
    CountDownLatch latch = new CountDownLatch(numberOfThreads);
    AtomicInteger counter = new AtomicInteger();
    //WHEN
    service.execute(() -> {
        activity.getIpSearchRunnable();
        counter.incrementAndGet();
        latch.countDown();
    });
    latch.await();
    //THEN
    assertEquals(numberOfThreads, counter.get());
}

@Test
public void whenExecuteThread_givenThreadInBackground_thenWorkInBackground_v2_distance() throws InterruptedException {
    //GIVEN
    int numberOfThreads = 1;
    ExecutorService service = Executors.newFixedThreadPool(1);
    CountDownLatch latch = new CountDownLatch(numberOfThreads);
    AtomicInteger counter = new AtomicInteger();
    //WHEN
    service.execute(() -> {
        activity.getDistanceReadRunnable(228, null);
        counter.incrementAndGet();
        latch.countDown();
    });
    latch.await();
    //THEN
    assertEquals(numberOfThreads, counter.get());
}

@Test
public void whenExecuteThread_givenThreadInBackground_thenWorkInBackground_v3_GUI() throws InterruptedException {
    //GIVEN
    int numberOfThreads = 1;
    ExecutorService service = Executors.newFixedThreadPool(1);
    CountDownLatch latch = new CountDownLatch(numberOfThreads);
    AtomicInteger counter = new AtomicInteger();
    //WHEN
    service.execute(() -> {
        activity.getGuiRunnable(0);
        counter.incrementAndGet();
        latch.countDown();
    });
    latch.await();
    //THEN
    assertEquals(numberOfThreads, counter.get());
}
```

Sprawdzono zachowanie metody podczas wyszukiwania adresu ip modułu sprzętowego przy braku mockowania wymaganych zależności. Sprawdzono poprawność konstrukcji adresu url dla pobierania danych w usłudze rest.

Listing 6.4: Testy dla klasy RestSearchIpCtrl

```
@Test
public void whenNoAction_givenDefaultValue_thenValueNotChange() {
    assertEquals(ctrl.getIp4Byte(), "none");
    assertEquals(ctrl.getFourthIp(), 1);
    assertEquals(ctrl.getIp(), 1);
}

@Test
public void whenBuildUrl_givenPartialIp_thenHaveCorrectParts() {
    //GIVEN
    String url;
    //WHEN
    url = ctrl.getPreparedUrl(224);
    //THEN
    assertTrue(url.contains("http"));
    assertTrue(url.contains("192.168."));
    assertTrue(url.contains(":8080"));
    assertTrue(url.contains("224"));
    assertTrue(url.contains("null"));
}
```

6.1.3 Testy bazy danych

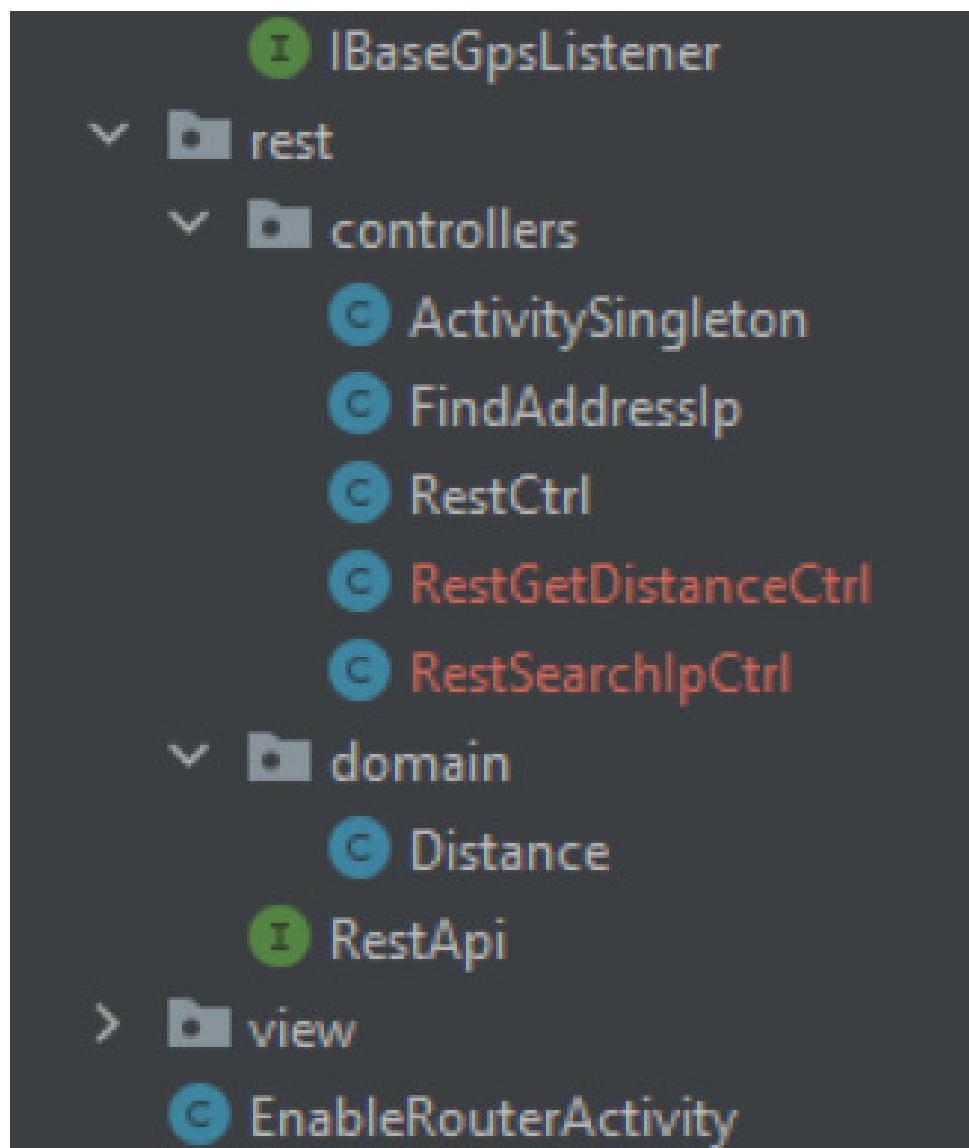
Przetestowano poprawność zapisu i odczytu wartości odległości w bazie danych.

Listing 6.5: Testy dla klasy DbManager

```
@Test
public void whenCommitDatabase_givenDistanceValue_thenReadSameFromDatabase() throws InterruptedException {
    //GIVEN
    db.setDistance(234);
    //WHEN
    db.dbCommit();
    //THEN
    assertEquals(db.getDbDistance(), 234);
}
```

6.2 Refaktoryzacja implementacji

W wyniku analizy składni i struktury istniejącej implementacji przy uwzględnieniu przeprowadzonych testów jednostkowych, postanowiono przeprowadzić refaktoryzację istniejącej implementacji w celu zwiększenia jakości i elastyczności utworzonej implementacji. Powstały kod aplikacji tworzono przy jednoczesnej analizie i kompozycji kolejnych przypadków testowych.



Rysunek 6.3: Zmiana struktury projektu aplikacji [źródło: własne]

6.3 Implementacja po refaktoryzacji

W nowej implementacji wykonano dekompozycję fragmentów kodu będących funkcjonalnie monolitami na większą liczbę metod, jednocześnie zrezygnowano z nadmiarowej implementacji dotyczącej funkcjonalności, które nie są w ramach projektu wykorzystywane. Na istniejącą formą implementacji miały wpływ testy jednostkowe, pisane w trakcie refaktoryzacji kodu.

Listing 6.6: Implementacja klasy MainActivity po refactoryzacji

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        initialiseInfoPrequisitions(getInstructionString(), getInfoAlertString());
        setNextIntentActivity(new Intent(this, EnableRouterActivity.class));

        setContentView(R.layout.activity_main);

        mobDataFlag = isMobileDataEnabledFromLollipop(this);

        if(mobDataFlag) {
            startActivity(getNextIntentActivity());
        }

        instructionTxtView = (TextView) findViewById(R.id.mob_data_instruction);
        instructionTxtView.setText(instructionTxt);

        enMobDataBtn = (Button) findViewById(R.id.data_enable_btn);
        enMobDataBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                mobDataFlag = isMobileDataEnabledFromLollipop(getApplicationContext());

                if(mobDataFlag) {
                    startActivity(getNextIntentActivity());
                } else {
                    TextView mobDataAlertTxtView = (TextView) findViewById(R.id.mob_data_dis);
                    mobDataAlertTxtView.setText(mobileDataDisableAlert);
                }
            }
        });
    }

    public void initialiseInfoPrequisitions(String instruction, String offInfoAlert) {
        this.instructionTxt = instruction;
        this.mobileDataDisableAlert = offInfoAlert;
    }

    @NotNull
    private String getInfoAlertString() {
        String infoAlert = "Mobile_data_are_disable\n" +
                "Please_follow_instruction_note";
        return infoAlert;
    }

    @NotNull
    private String getInstructionString() {
        String instructionTxt = "-----Mobile_network_data_enable_is_need:\n" +
                "-----1.Go_to:\n" +
                "----->_Settings\n" +
                "----->_Connections\n" +
                "----->_Data_Counter\n" +
                "-----2._Enable_mobile_network_data\n" +
                "-----3._Enter_into_SoftDrive_application_again";
        return instructionTxt;
    }

    private boolean isMobileDataEnabledFromLollipop(Context context) {
        boolean state = false;
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
            state = Settings.Global.getInt(context.getContentResolver(), "mobile_data", 0) == 1;
        }
        return state;
    }

    public boolean isMobDataFlag() {
        return mobDataFlag;
    }

    public void setMobDataFlag(boolean mobDataFlag) {
        this.mobDataFlag = mobDataFlag;
    }

    public String getInstructionTxt() {
        return instructionTxt;
    }

    public String getMobileDataDisableAlert() {
        return mobileDataDisableAlert;
    }

    public Intent getNextIntentActivity() {
        return nextIntentActivity;
    }
}
```

```

}

public void setNextIntentActivity(Intent nextIntentActivity) {
this.nextIntentActivity = nextIntentActivity;
}
}

```

Listing 6.7: Implementacja klasy EnableRouterActivity po refactoryzacji

```

public class EnableRouterActivity extends AppCompatActivity {

public String instructionTxt;
private Button enRouterActBtn;
String mobileDataDisableAlert;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_router_enable);

if(isApEnable()) {
startNextActivity();
} else {
showInstruction(getInstructionDefault());
confBtnListener(getInfoAlertDefault());
}
}

public void confBtnListener(String btnInfoOnClick) {
enRouterActBtn = (Button) findViewById(R.id.ap_enable_btn);
enRouterActBtn.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {

if (isApEnable()) {
startNextActivity();
} else {
showInfoAlert(btnInfoOnClick);
}
}
});
}

public void startNextActivity() {
Intent enRestAct = new Intent(getApplicationContext(), RestCtrlActivity.class);
startActivity(enRestAct);
}

public void showInfoAlert(String alertTxt) {
mobileDataDisableAlert = alertTxt;
TextView apRouterAlertTxtView = (TextView) findViewById(R.id.ap_data_dis);
apRouterAlertTxtView.setText(mobileDataDisableAlert);
}

@NotNull
public String getInfoAlertDefault() {
String mobileDataDisableAlert = "Wifi_router_is_disable\n" +
"Please_follow_instruction_note";
return mobileDataDisableAlert;
}

public void showInstruction(String instruction) {
instructionTxt = instruction;
TextView instructionTxtView = (TextView) findViewById(R.id.instruction_str);
instructionTxtView.setText(instructionTxt);
}

@NotNull
public String getInstructionDefault() {
String instructionTxt = "-----Router_Wifi_enable_is_need:\n" +
"-----1.Go_to:\n" +
"----->Settings\n" +
"----->Connections\n" +
"----->Wifi_Hotspot\n" +
"-----2.Enable_Wifi_Hotspot\n" +
"-----3.Set_SSID_name:_____safedrivea\n" +
"-----4.Set_password:_____safedrivea\n" +
"-----5.Enter_into_SoftDrive_application_again";
return instructionTxt;
}

private boolean isApEnable() {
WifiManager wifiManager = (WifiManager) getApplicationContext().getSystemService(WIFI_SERVICE);
boolean ret = false;
try{
Method isWifiApEnabledMethod = wifiManager.getClass().getMethod("isWifiApEnabled");
boolean isApEnable = (Boolean) isWifiApEnabledMethod.invoke(wifiManager);
ret = isApEnable;
} catch (Exception e) {
e.printStackTrace();
}
return ret;
}

public String getInstructionTxt() {
return instructionTxt;
}

public void setInstructionTxt(String instructionTxt) {
this.instructionTxt = instructionTxt;
}

```

```

}

public String getInfoAlertTxt() {
    return mobileDataDisableAlert;
}

public void setInfoAlertTxt(String mobileDataDisableAlert) {
    this.mobileDataDisableAlert = mobileDataDisableAlert;
}
}

```

Listing 6.8: Implementacja klasy ActivitySingleton po refactoryzacji

```

private static ActivitySingleton singleton;

@Override
public void onCreate() {
    super.onCreate();
    singleton = this;
}

public static synchronized ActivitySingleton getInstance() {
    return singleton;
}
}

```

Listing 6.9: Implementacja klasy FindAddressIp po refactoryzacji

```

public class FindAddressIp {

    private String ip = "192.168.";

    @RequiresApi(api = Build.VERSION_CODES.N)
    public FindAddressIp() {
        ip += get3rdByteOfIp();
    }

    public String getNetworkIpWithoutHost() {
        return ip;
    }

    @RequiresApi(api = Build.VERSION_CODES.N)
    private String get3rdByteOfIp() {

        String ip3Byte = "";

        try {
            //shell command for show ip
            String result = getShellReply();

            //cut shell command show result
            String cutResult = shellReplyFilterByIp(result);

            //get ip from cut shell command
            Matcher m = regexFilterResultByIp(cutResult);

            if(m.find()) {
                return extractIpByteFromRegex(m);
            }
        } catch (IOException e) {
            Log.d("FindAddressIp", "ERROR_ip3byte");
            e.printStackTrace();
        }
        return null;
    }

    @NonNull
    private String extractIpByteFromRegex(Matcher m) {
        String ip3Byte;
        ip3Byte = m.group(0);
        ip3Byte += ".";
        return ip3Byte;
    }

    @NonNull
    private Matcher regexFilterResultByIp(String cutResult) {
        String regex = "^\d{1,3}";
        Pattern r = Pattern.compile(regex);
        Matcher m = r.matcher(cutResult);
        return m;
    }

    @NonNull
    private String shellReplyFilterByIp(String result) {
        String subStr = "inet 192.168.";
        int ipEnd = result.lastIndexOf(subStr);
        String cutResult = result.substring(ipEnd + subStr.length());
        return cutResult;
    }

    @NonNull
    private String getShellReply() throws IOException {
        Process process = Runtime.getRuntime().exec("ip addr show wlan0");
        BufferedReader in = new BufferedReader(new InputStreamReader(process.getInputStream()));
        String result = in.lines().collect(Collectors.joining());
    }
}

```

```

return result;
}

```

Listing 6.10: Implementacja klasy RestCtrl po refactoryzacji

```

public class RestCtrl {

    @RequiresApi(api = Build.VERSION_CODES.N)
    public String getPreparedUrl(int hostPartIp) {
        FindAddressIp ipCtrl = new FindAddressIp();
        String networkIp = ipCtrl.getNetworkIpWithoutHost();

        String url = "";
        url += protocol;
        url += networkIp;
        url += String.valueOf(hostPartIp);
        url += portDistance;

        return url;
    }

    @RequiresApi(api = Build.VERSION_CODES.N)
    protected RestApi initRetrofitByUrl(String url) {
        return initRetrofit(initHttpHeader(),
            initGson(),
            initLogBuilder(),
            url);
    }

    @RequiresApi(api = Build.VERSION_CODES.N)
    protected RestApi initRetrofitByIp(int ip4Byte) {
        return initRetrofit(initHttpHeader(),
            initGson(),
            initLogBuilder(),
            getPreparedUrl(ip4Byte));
    }

    protected RestApi initRetrofit(OkHttpClient httpHeaderConf, Gson gson, OkHttpClient.Builder clientBuilder, String url) {
        Retrofit retrofit = new Retrofit.Builder()
            .client(httpHeaderConf)
            .addConverterFactory(GsonConverterFactory.create(gson))
            .baseUrl(url)
            .client(clientBuilder.build())
            .build();
        RestApi restApi = retrofit.create(RestApi.class);
        return restApi;
    }

    protected OkHttpClient initHttpHeader() {
        OkHttpClient httpHeaderConf = new OkHttpClient.Builder().addInterceptor(new Interceptor() {
            @Override
            public okhttp3.Response intercept(Chain chain) throws IOException {
                Request newRequest = chain.request().newBuilder()
                    .addHeader("Accept", "application/json")
                    .build();
                return chain.proceed(newRequest);
            }
        }).build();
        return httpHeaderConf;
    }

    protected OkHttpClient initLogBuilder() {
        OkHttpClient.Builder clientBuilder = new OkHttpClient.Builder();
        HttpLoggingInterceptor loggingInterceptor = new HttpLoggingInterceptor();
        loggingInterceptor.setLevel(HttpLoggingInterceptor.Level.BODY);
        clientBuilder.addInterceptor(loggingInterceptor);
        return clientBuilder;
    }

    private Gson initGson() {
        return new GsonBuilder()
            .setLenient()
            .create();
    }

    public Gson getGson() {
        return gson;
    }
}

```

Listing 6.11: Implementacja klasy RestGetDistanceCtrl po refactoryzacji

```

public class RestGetDistanceCtrl extends RestCtrl implements Callback<Distance> {

    @RequiresApi(api = Build.VERSION_CODES.N)
    public RestGetDistanceCtrl() {
        context = ActivitySingleton.getInstance();
        initDb(context);
    }

    @RequiresApi(api = Build.VERSION_CODES.N)
    public String getFullIpWithUrl(int ip4Byte) {
        ip = new FindAddressIp();
        String ipNetworkAddress = ip.getNetworkIpWithoutHost();

        super.distanceUrl = "";
        super.distanceUrl += protocol;

```

```

super.distanceUrl += ipNetworkAddress;
super.distanceUrl += String.valueOf(ip4Byte);
super.distanceUrl += portDistance;

return super.distanceUrl;
}

@RequiresApi(api = Build.VERSION_CODES.N)
public void startRest(int ip4Byte) {
String url = getFullIpWithUrl(ip4Byte);
super.restApi = super.initRetrofitByUrl(url);
call();
}

public void call() {
super.restApi.getDistanceEndpoint("application/json").enqueue(this);
}

@RequiresApi(api = Build.VERSION_CODES.N)
@Override
public void onResponse(Call<Distance> call, Response<Distance> response) {
Log.v("Response_callback", "call_callback");

if (response.isSuccessful()) {
assert response.body() != null;
commitDistanceDb(response.body());
call.cancel();
}
}

@RequiresApi(api = Build.VERSION_CODES.N)
@Override
public void onFailure(Call<Distance> call, Throwable t){
Log.e("rest_read_error","read_distance_failure");
}

public DbManager getDb () {
return db;
}

public void setDb (DbManager db){
this.db = db;
}

private void commitDistanceDb(Distance dist){
getDb().setDistance(dist.getDistance());
getDb().dbCommit();
}

private void initDb(Context context){
setDb(new DbManager(context));
}
}

```

Listing 6.12: Implementacja klasy RestSearchIpCtrl po refactoryzacji

```

public class RestSearchIpCtrl extends RestCtrl implements Callback<Distance> {

public RestSearchIpCtrl() {
fourthIp = 1;
ip4Byte = "none";
ip = 1;
}

@RequiresApi(api = Build.VERSION_CODES.N)
public void startSearchIp() throws IOException {
super.restApi = initRetrofitByIp(fourthIp);
getCall();
}

public void getCall() throws IOException {
super.restApi.getDistanceEndpoint("application/json").enqueue(this);
}

@RequiresApi(api = Build.VERSION_CODES.N)
public void startSearchIpLoop() throws IOException {
for(ip=1; ip<255; ip++) {
super.restApi = initRetrofitByIp(ip);
getCall();
}
}

@RequiresApi(api = Build.VERSION_CODES.N)
@Override
public void onResponse(Call<Distance> call, Response<Distance> response) {
Log.v("Response_IP_address", "find_IP_address");

if (response.isSuccessful()) {
setIp4Byte(String.valueOf(ip));
}
}

@RequiresApi(api = Build.VERSION_CODES.N)

```

```

@Override
public void onFailure(@NonNull Call<Distance> call, @NonNull Throwable t){
    Log.e("try_find_IP", "IP_not_exist");
}

public String getIp4Byte() {
    return ip4Byte;
}

public void setIp4Byte(String ip4Byte) {
    this.ip4Byte = ip4Byte;
}

public int getFourthIp() {
    return fourthIp;
}

public void setFourthIp(int fourthIp) {
    this.fourthIp = fourthIp;
}

public int getIp() {
    return ip;
}

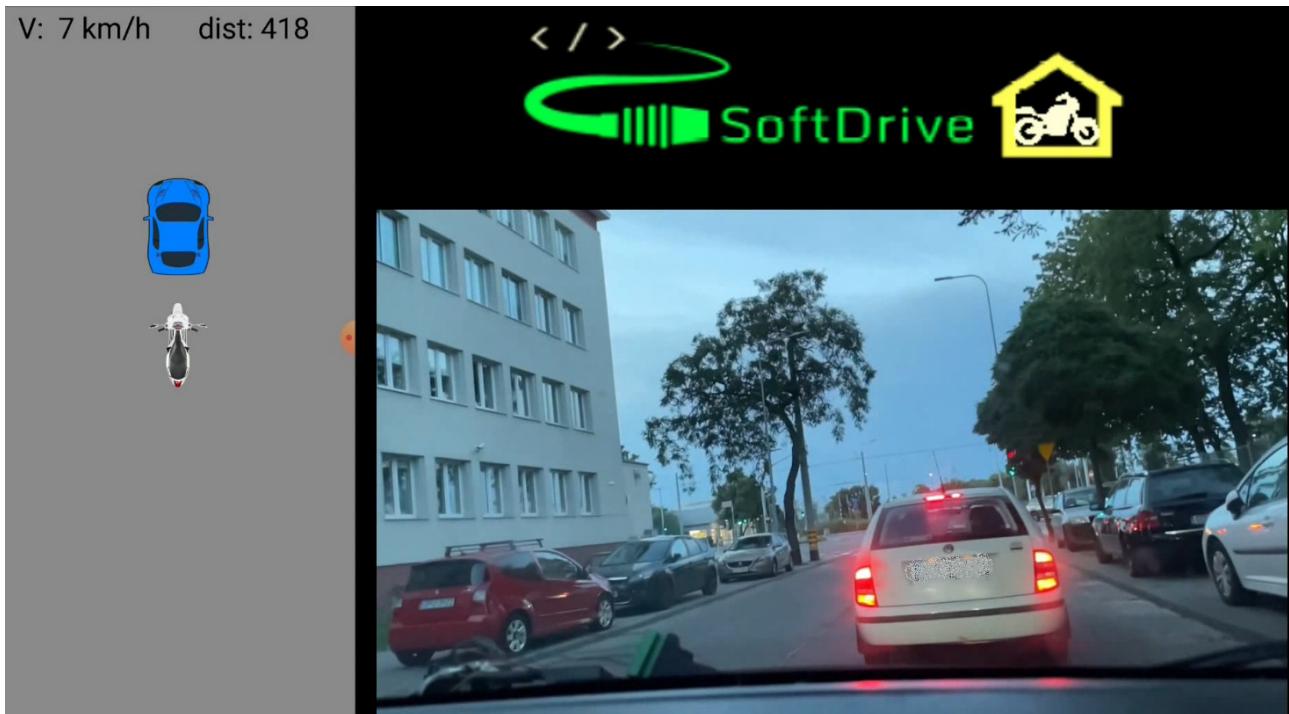
public void setIp(int ip) {
    this.ip = ip;
}

```

6.4 Testy użyteczności

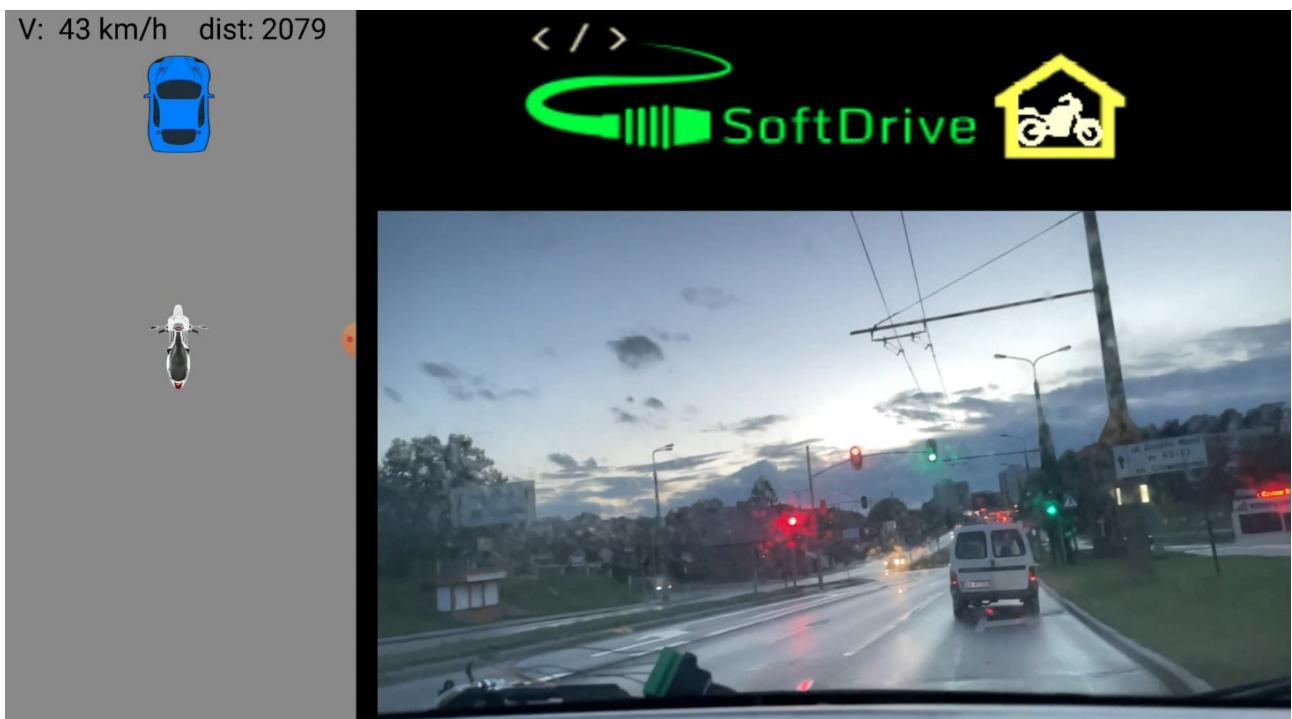
Weryfikacja czy wdrożone rozwiązanie spełnia wymagania użyteczności, polegała na kontrolnym teście zachowania się systemu w różnych przypadkach testowych wywoływanych w rzeczywistych warunkach ruchu drogowego w samochodzie osobowym z zamontowanym prototypem.

Test systemu podczas jazdy w bliskiej odległości od pojazdu jadącego z przodu.



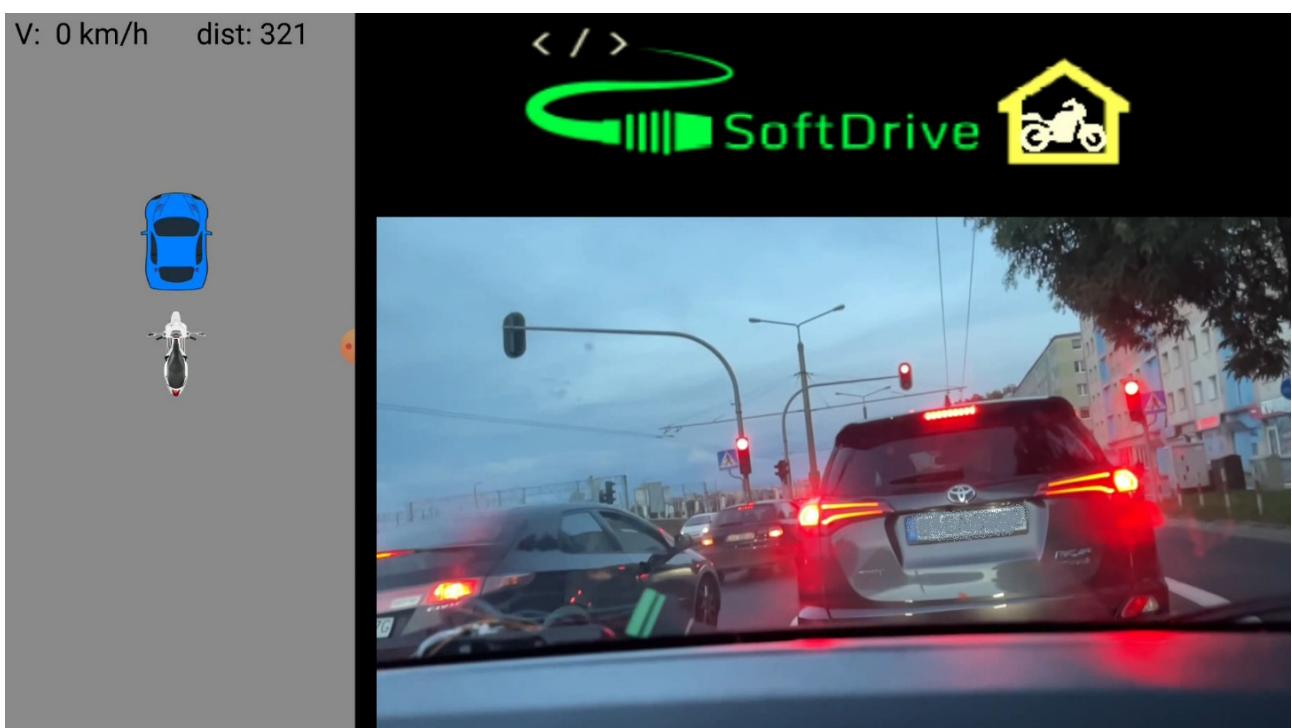
Rysunek 6.4: Praca systemu warunkach bliskiej odległości od pojazdu [źródło: własne].

Test systemu podczas jazdy w dalekiej odległości od pojazdu jadącego z przodu.



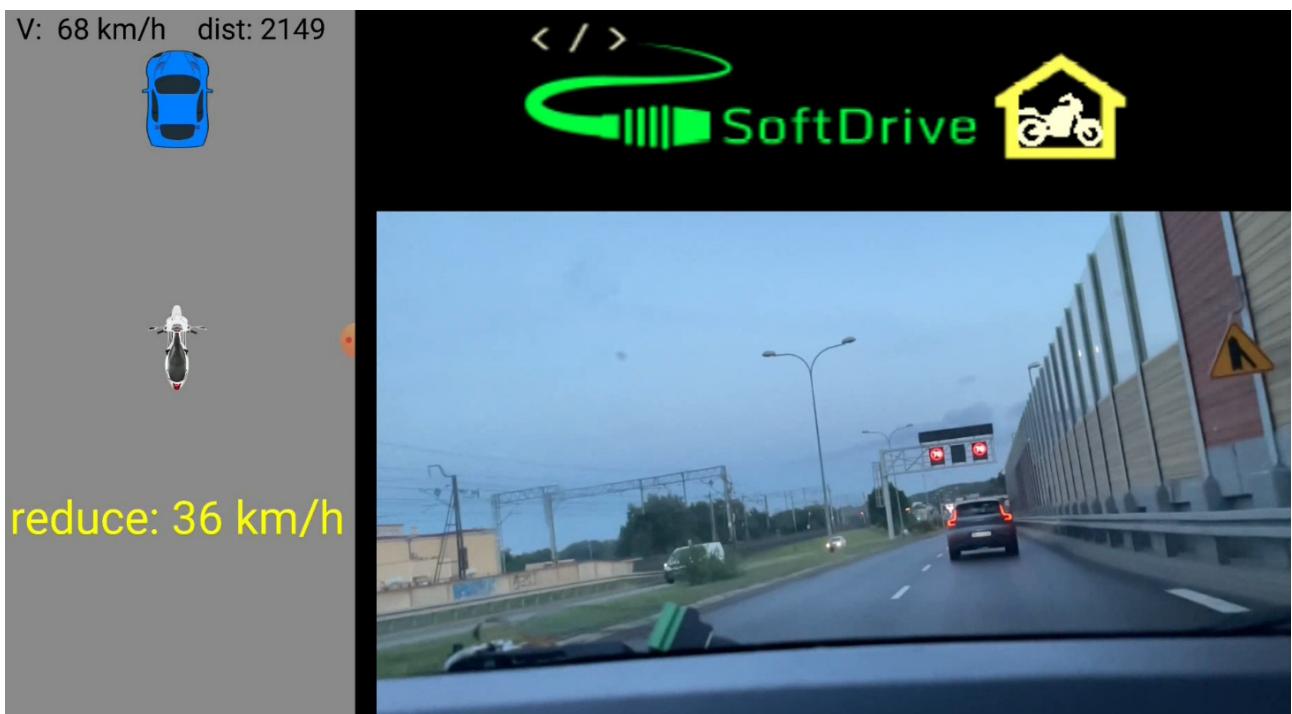
Rysunek 6.5: Praca systemu warunkach dalekiej odległości od pojazdu jadącego z przodu. [źródło: własne].

Test systemu podczas postoju z pomiarem odległości.



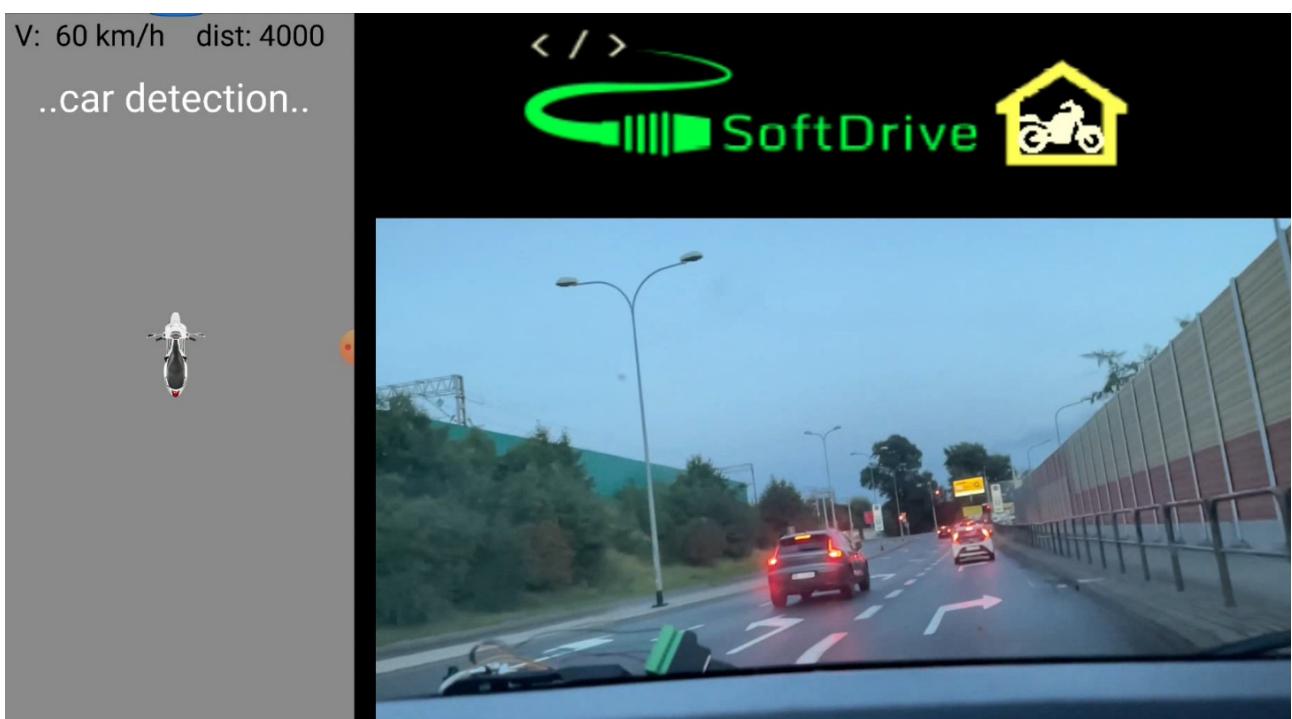
Rysunek 6.6: Praca systemu podczas postoju [źródło: własne].

Test pracy systemu podczas jazdy w warunkach zbyt małej odległości przy wysokiej prędkości.



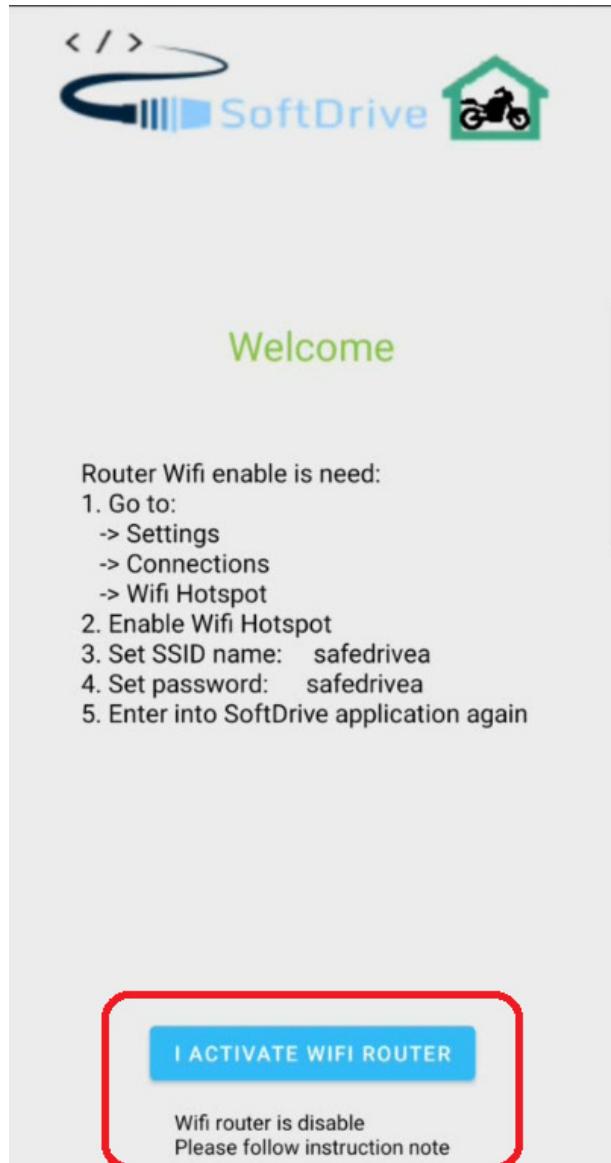
Rysunek 6.7: Praca systemu podczas detekcji zagrożenia ruchu drogowego [źródło: własne].

Test pracy systemu podczas problemów z odczytem odległości z sensora.



Rysunek 6.8: Praca systemu podczas braku odczytu odległości [źródło: własne].

Test graficznego interfejsu użytkownika podczas wstępnych ustawień systemu.



Rysunek 6.9: Weryfikacja poprawności interakcji z GUI [źródło: własne].

W każdym z przedstawionych przypadków testowych nie zaobserwowano nieprawidłowej pracy systemu, w każdym przypadku system działał stabilnie.

Podczas przeprowadzonych testów zaobserwowano częste problemy z odczytem odległości z czujnika przy odległościach przekraczających 10 m. System zgłaszał wówczas przejście do trybu wykrywania pojazdu. Problem nie pojawiał się przy odległościach mniejszych niż 10 m. Przy czyną powstającego ograniczenia jest konstrukcja sprzętowa czujnika odległości, który zgodnie z notą katalogową powinien odczytywać odległość do 40 metrów, jednak w specyficznych warunkach przeprowadzonych testów, przy mierzonych odległościach przekraczających 10 metrów, podczas często zmieniającej się pozycji obiektu, względem którego dokonywany jest pomiar oraz zmieniającej się bazowej pozycji, z której wykonywane są pomiary oraz przy częstotliwości dokonywanych pomiarów przekraczającej 250 milisekund czujnik odległości często gubił pomiar.

Rozdział 7

Opis przyrostów

Opis przebiegu realizacji pracy.

Podział realizacji pracy w kontekście spodziewanych produktów.

7.1 Część sprzętowa

1. etap:

- (a) Cel: Pierwsze uruchomienie laserowego czujnika odległości.
- (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
- (c) Proces: W dwóch podejściach udało się uruchomić laserowy czujnik odległości na urządzeniu Raspberry Pi, poprzez przewodowe połączenie podzespołów, instalację sterowników i bibliotek oraz wymaganą konfigurację wstępную na urządzeniu w celach testowych. Zamierzony efekt uzyskano dzięki dostępności oprogramowania w języku C++ udostępnionego przez społeczność skupioną wokół projektu producenta laserowego czujnika odległości.
- (d) Skutek: Otrzymano produkt: minikomputer Raspberry Pi z podłączonym laserowym skanerem odległości.
- (e) Plan: Planowane było udokumentowanie rozwiązania.
- (f) Uwagi: Brak.

2. etap:

- (a) Cel: w trzech podejściach zmodyfikowano oprogramowanie producenta czujnika odległości w celu udostępnienia odczytów danych o odległości w wewnętrznej sieci lokalnej systemu.
- (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
- (c) Proces: W dwóch podejściach zmodyfikowano oprogramowanie producenta dodając usługę Rest API i przypisano odczyty aktualnych wskazań do zdarzenia żądania GET poprzez protokół HTTP skierowanego do usługi REST aplikacji.
- (d) Skutek: Udostępniono usługę Rest API poprzez moduł aplikacji systemu laserowego czujnika odległości.
- (e) Plan: Planowane było udokumentowanie rozwiązania.
- (f) Uwagi: Brak.

3. etap:

- (a) Cel: Utworzenie prywatnej sieci lokalnej dla systemu
- (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
- (c) Proces: W dwóch podejściach podjęto próbę komunikacji urządzeń w ramach sieci Bluetooth. Podjęta próba nie przyniosła zamierzonego efektu. Podjęto drugą próbę komunikacji wykorzystując protokół WiFi z pozytywnym skutkiem.
- (d) Skutek: Utworzono prywatną wewnętrzną sieć lokalną dla urządzeń.
- (e) Plan: Planowane było udokumentowanie rozwiązania.

- (f) Uwagi: Brak.
4. etap:
- (a) Cel: Automatyzacja procesu uruchamiania oprogramowania sterującego pracą podłączonych sensorów z usługą Rest API przy starcie systemu.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: W dwóch podejściach utworzono skrypty startowe uruchamiające usługę Rest API dla odczytu danych z podłączonych sensorów na żądanie metodą GET protokołu HTTP. Następnie skonfigurowano automatyczne wykrywanie i łączenie się z wewnętrzną siecią lokalną systemu.
 - (d) Skutek: Powstały dwa systemy wbudowane nasłuchujące żądań protokołu HTTP ze strony aplikacji mobilnej w celu udostępnienia aktualnych wskazań czujników.
 - (e) Plan: Planowane było udokumentowanie rozwiązania.
 - (f) Uwagi: Brak.

7.2 Kryteria akceptacji części sprzętowej

1. Nawiązanie do etapu 1: Kryteria akceptacji:
Pierwsze i kolejne uruchomienia produktu nie generują błędów.
2. Nawiązanie do etapu 2: Kryteria akceptacji:
Możliwe jest uruchomienie oprogramowania sterującego pracą laserowego czujnika odległości i odczyt wskazań czujnika.
3. Nawiązanie do etapu 3: Kryteria akceptacji:
Urządzenia podłączone do sieci lokalnej komunikują się z bramą podsieci.
4. Nawiązanie do etapu 4: Kryteria akceptacji:
Możliwy jest odczyt danych o odległości poprzez żądanie GET protokołu HTTP z urządzeń przyłączonych do wewnętrznej sieci lokalnej systemu.

7.3 Dokumentacja

1. etap:
 - (a) Cel: Utworzenie rozdziału Wstęp w dokumentacji.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: W jednym podejściu uzupełniono dokumentację o rozdział Wstęp.
 - (d) Skutek: Powstał nowy rozdział: Wstęp
 - (e) Plan: Planowana jest dalsza praca nad dokumentacją.
 - (f) Uwagi: Brak.
2. etap:
 - (a) Cel: Aktualizacja Dokumentu Założeń Wstępnych.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: W jednym podejściu zaktualizowano Dokument Założeń Wstępnych.
 - (d) Skutek: Powstał aktualny merytorycznie Dokument Założeń Wstępnych.
 - (e) Plan: Planowana jest dalsza praca nad dokumentacją zgodnie z przyjętą metodologią kaskadową.
 - (f) Uwagi: Brak.
3. etap:
 - (a) Cel: Ustalenie zakresu projektu.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: W jednym podejściu ustalono w dokumentacji zakres realizowanego projektu.
 - (d) Skutek: Powstała nowa sekcja: zakres projektu.
 - (e) Plan: Planowana jest dalsza praca nad dokumentacją.
 - (f) Uwagi: Brak.
4. etap:

- (a) Cel: Aktualizacja sekcji Założeń projektowych dokumentu Wstępny Planu Projektu.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: W jednym podejściu zaktualizowano sekcję Założeń projektu w dokumencie Wstępny Plan Projektu.
 - (d) Skutek: Powstała aktualna merytorycznie sekcja dokumentu Wstpnego Planu Projektu.
 - (e) Plan: Planowana jest dalsza praca nad dokumentacją.
 - (f) Uwagi: Brak.
5. etap:
- (a) Cel: Aktualizacja sekcji Wizji rozwiązania, Planowanego procesu wytwarzania i Zakładanych zasobów dokumentu Wstpnego Planu Projektu.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: W jednym podejściu zaktualizowano sekcję Wizji rozwiązania, Planowanego Procesu wytwarzania i Zakładanych Zasobów w dokumencie Wstępny Plan Projektu.
 - (d) Skutek: Powstała aktualna merytorycznie sekcja dokumentu Wstpnego Planu Projektu.
 - (e) Plan: Planowana jest dalsza praca nad dokumentacją.
 - (f) Uwagi: Brak.
6. etap:
- (a) Cel: Wypełnienie dokumentu Projektu Systemu.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: W pięciu podejściach wypełniono wstępnie dokument Projektu Systemu.
 - (d) Skutek: Powstał aktualny merytorycznie dokument Projektu Systemu.
 - (e) Plan: Planowana jest dalsza praca nad dokumentacją i wdrożeniem.
 - (f) Uwagi: Brak.
7. etap:
- (a) Cel: Wypełnienie w dokumencie specyfikacji diagramów.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: W pięciu podejściach, stosując metodologię ewolucyjno-przyrostową wypełniono specyfikację diagramów.
 - (d) Skutek: Powstała aktualna merytorycznie specyfikacja diagramów.
 - (e) Plan: Planowana jest dalsza praca nad dokumentacją i wdrożeniem zgodnie z przyjętą metodologią ewolucyjno-przyrostową.
 - (f) Uwagi: Brak.
8. etap:
- (a) Cel: Wypełnienie dokumentu Wdrożenie rozwiązania.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: W czasie 5 miesięcy, stosując metodologię ewolucyjno-przyrostową wypełniono rozdział Wdrożenie rozwiązania.
 - (d) Skutek: Powstała aktualna merytorycznie treść rozdziału Wdrożenie rozwiązania.
 - (e) Plan: Planowana jest dalsza praca nad dokumentacją i wdrożeniem zgodnie z przyjętą metodologią ewolucyjno-przyrostową.
 - (f) Uwagi: Brak.
9. etap:
- (a) Cel: Wypełnienie dokumentu Testowanie rozwiązań.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: W czternastu podejściach, stosując metodologię ewolucyjno-przyrostową wypełnić rozdział Testowanie rozwiązań.
 - (d) Skutek: Powstała aktualna merytorycznie treść rozdziału Testowanie rozwiązań.

- (e) Plan: Planowana jest dalsza praca nad dokumentacją i wdrożeniem zgodnie z przyjętą metodologią ewolucyjno-przyrostową.
 - (f) Uwagi: Brak.
10. etap:
- (a) Cel: Wypełnienie dokumentu z podsumowaniem pracy.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: W sześciu podejściach, stosując metodologię ewolucyjno-przyrostową wypełnić rozdział z podsumowaniem pracy.
 - (d) Skutek: Powstała aktualna merytorycznie treść rozdziału z podsumowaniem pracy.
 - (e) Plan: Planowane jest zakończenie realizacji projektu.
 - (f) Uwagi: Brak.
11. etap:
- (a) Cel: Wypełnienie dokumentu o potencjale komercjalizacyjnym produktu.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: W dwóch podejściach, stosując metodologię ewolucyjno-przyrostową wypełniono rozdział Potencjał komercjalizacyjny.
 - (d) Skutek: Powstała aktualna merytorycznie treść rozdziału Potencjał komercjalizacyjny produktu.
 - (e) Plan: Planowana jest dalsza praca nad dokumentacją i wdrożeniem zgodnie z przyjętą metodologią ewolucyjno-przyrostową.
 - (f) Uwagi: Brak.
12. etap:
- (a) Cel: Aktualizacja struktury dokumentacji.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: W jednym podejściu uporządkowano część struktury dokumentacji poprzez usunięcie, modyfikację i dodanie wybranych sekcji.
 - (d) Skutek: Powstała nowa struktura dokumentacji.
 - (e) Plan: Planowana jest dalsza praca nad dokumentacją.
 - (f) Uwagi: Brak.
13. etap:
- (a) Cel: Sprawdzenie poprawności dokumentacji.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową..
 - (c) Proces: W dwóch podejściach, stosując metodologię ewolucyjno-przyrostową przeanalizowano poprawność dokumentacji.
 - (d) Skutek: Powstała aktualna merytorycznie dokumentacja projektu.
 - (e) Plan: Planowane jest zakończenie etapu dokumentowania pracy..
 - (f) Uwagi: brak.

7.4 Kryteria akceptacji dokumentacji

1. Nawiązanie do etapu 1: Kryteria akceptacji:

Powstały rozdział Wstęp jest kompletny i zawiera aktualną merytorycznie treść zgodną z resztą dokumentacji.

2. Nawiązanie do etapu 2: Kryteria akceptacji:

Dokumentacja ukończona w stopniu pozwalającym na rozpoczęcie tworzenia produktu podczas pierwszego przyrostu realizacji projektu.

3. Nawiązanie do etapu 3: Kryteria akceptacji:

Powstały zakres projektu opisuje wszystkie sekcje dokumentacji.

4. Nawiązanie do etapu 4: Kryteria akceptacji:

Powstała struktura dokumentacji jest spójna, logiczna i zgodna z wymaganiami dokumentacji projektowej.

5. Nawiązanie do etapu 5: Kryteria akceptacji:
Powstała sekcja dokumentu Wstępny Plan Projektu jest kompletna i zawiera aktualną merytorycznie treść zgodną z resztą dokumentacji.
6. Nawiązanie do etapu 6: Kryteria akceptacji:
Powstały dokument Projektu Systemu jest kompletny i zawiera aktualną merytorycznie treść zgodną z resztą dokumentacji.
7. Nawiązanie do etapu 7: Kryteria akceptacji:
Wypełniony rozdział Specyfikacja diagramów jest kompletny i zawiera aktualną merytorycznie treść zgodną z resztą dokumentacji.
8. Nawiązanie do etapu 8: Kryteria akceptacji:
Wypełniony rozdział Wdrożenie rozwiązania jest kompletny i zawiera aktualną merytorycznie treść zgodną z resztą dokumentacji.
9. Nawiązanie do etapu 9: Kryteria akceptacji:
Powstały produkt spełnia założenia projektu, pozytywnie przechodzi proces testowania, zapewnia wymaganą funkcjonalność, spełnia wymagania jakościowe i inne.
10. Nawiązanie do etapu 10: Kryteria akceptacji:
Wypełniony rozdział podsumowanie prac jest kompletny i zawiera aktualną merytorycznie treść zgodną z resztą dokumentacji, szczególnie z sekcją Etapy wytwarzania dokumentu Wstępnego Planu Projektu.
11. Nawiązanie do etapu 11: Kryteria akceptacji:
Wypełniony rozdział Podsumowanie jest kompletny i zawiera aktualną merytorycznie treść zgodną z resztą dokumentacji. Dodatkowo treść rozdziału zawiera syntezę komercjalizacji produktu ukończonego projektu.
12. Nawiązanie do etapu 12: Kryteria akceptacji:
Aktualizacja struktury dokumentacji ma na celu zapewnienie zachowania spójności pomiędzy poszczególnymi działami i eliminację błędów powstałych podczas realizacji projektu.
13. Nawiązanie do etapu 13: Kryteria akceptacji:
Dokumentacja jest kompletna, logicznie uporządkowana, nie zawiera błędów merytorycznych i strukturalnych oraz jest spójna.

7.5 Oprogramowanie

1. etap:
 - (a) Cel: Utworzenie aplikacji mobilnej
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: W czasie 3 miesięcy utworzono aplikację mobilną systemu Android.
 - (d) Skutek: Utworzenie aplikacji mobilnej.
 - (e) Plan: Planowane było udokumentowanie rozwiązania.
 - (f) Uwagi: Dostępna jest dokumentacja w postaci kodu źródłowego.

7.6 Kryteria akceptacji oprogramowania

1. Nawiązanie do etapu 1: Kryteria akceptacji:
Aplikacja mobilna instaluje się na emulatorze urządzenia z systemem Android i nie generuje błędów przy pierwszym i kolejnych uruchomieniach.
Zestawienie połączenia sieci lokalnej poprzez protokół Bluetooth, powinno zapewniać łączność pomiędzy urządzeniami. W przypadku braku łączności należy zmienić protokół sieciowy na inny o zbliżonej wymaganej funkcjonalności.
Automatyzacja procesu włączenia mobilnej transmisji danych w sieci komórkowej przy starcie aplikacji, powinna uruchomić usługę w tle działającej aplikacji. W przypadku

braku uruchomienia usługi w tle wymagane jest ręczne uruchomienie usługi przez użytkownika poprzez system operacyjny Android.

Automatyzacja procesu włączenia usługi udostępniania internetu poprzez hotspot WiFi przy starcie aplikacji, powinna uruchomić usługę w tle działającej aplikacji. W przypadku braku uruchomienia usługi w tle wymagane jest ręczne uruchomienie usługi przez użytkownika poprzez system operacyjny Android.

Automatyzacja procesu konfiguracji usługi udostępniania internetu poprzez hotspot WiFi przy starcie aplikacji, powinna skonfigurować zadaną nazwę użytkownika, hasło, rodzaj uwierzytelniania oraz typ pasma sieci. W przypadku konfiguracji w tle wymagane jest ręczne ustawienie przez użytkownika wymaganych konfiguracji poprzez system operacyjny Android.

7.7 Testowanie

1. typ testu:
 - (a) Testowane wdrożenie: Implementacja rozwiązania
 - (b) Rodzaj testu: Testy jednostkowe
 - (c) Przebieg procesu: przeprowadzenie refaktoryzacji istniejącego kodu
 - (d) wynik testu: Testy jednostkowe muszą być pozytywne w celu wdrażania dalszej implementacji, w przypadku negatywnego wyniku testu wymagana jest korekta implementacji i refaktoryzacja.
2. typ testu:
 - (a) Testowane wdrożenie: Poprawność rozwiązania
 - (b) Rodzaj testu: Testy walidacyjne
 - (c) Przebieg procesu: Analiza porównawcza cech otrzymanego produktu z założonymi w projekcie wymaganiami w rzeczywistym środowisku pracy systemu.
 - (d) wynik testu: W przypadku zgodności badanej cechy produktu z wymaganym założeniem projektowym wynik testu jest pozytywny.

Ze względu na bezobsługowy interfejs użytkownika aplikacji mobilnej systemu oraz bezobsługową specyfikę systemów wbudowanych realizowanego rozwiązania nie jest możliwe przeprowadzenie testów funkcjonalnych systemu.

7.8 Wdrożenie

1. etap:
 - (a) Cel: Wdrożenie projektowanych rozwiązań.
 - (b) Metoda: Wykorzystując metodologię ewolucyjno-przyrostową.
 - (c) Proces: W czasie 5 miesięcy, w sposób regularny, stosując metodologię ewolucyjno-przyrostową, wdrożono przyjęte rozwiązania zgodnie z Projektem systemu i aktualnymi założeniami projektowymi.
 - (d) Skutek: Powstał produkt realizacji projektu.
 - (e) Uwagi: Brak.
2. etap:
 - (a) Cel: Przygotowanie prezentacji systemu.
 - (b) Metoda: Wykorzystując techniki multimedialne
 - (c) Proces: Planowane jest utworzenie przebiegu prezentacji projektu.
 - (d) Skutek: Powstanie prezentacja zrealizowanego produktu.
 - (e) Plan: Brak.
 - (f) Uwagi: Brak

7.9 Kryteria akceptacji wdrożenia

1. Nawiązanie do etapu 1: Kryteria akceptacji:

Wdrożony system działa w sposób prawidłowy i stabilny oraz jest zgodny z wymaganiami projektowymi.

2. Nawiązanie do etapu 2: Kryteria akceptacji:

Na tym etapie produkt realizacji projektu powinien być ukończony a dokumentacja kompletna.

Rozdział 8

Podsumowanie

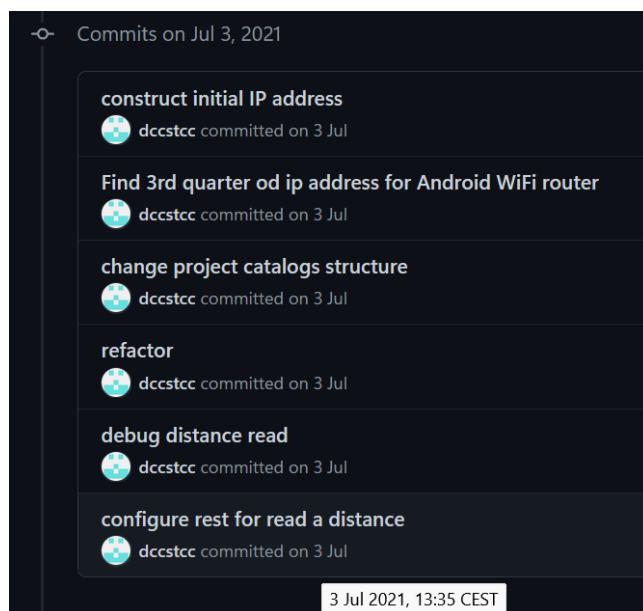
Podsumowanie przebiegu realizacji projektu.

8.1 Nakłady pracy

Z uwagi na samodzielny charakter realizacji prowadzonych prac w zakresie całego projektu, osobiście odpowiadam za utworzenie wszystkich komponentów teoretycznych i praktycznych powstałych podczas realizacji projektu.

Analizując wykonywane polecenia commit w repozytorium kodu serwisu github.com, odnoszące się do realizowanego projektu implementacji systemu z jednoczesnym uwarunkowaniem, że każda nowa grupa funkcjonalności była systematycznie commitowana w repozytorium lokalnym¹, możliwe jest przybliżone określenie czasu realizacji poszczególnych etapów projektu.

W czasie 60 dni zarejestrowanej historii realizacji projektu na prace programistyczne przeznaczono 25 dni, w okresie od 01.07.2021 r. do 01.09.2021 r. Na podstawie średniej ilości godzin w ciągu dnia przeznaczonych na implementację można obliczyć sumaryczny czas pracy wykonanej przy realizacji projektu.



Rysunek 8.1: Początkowy czas pracy nad implementacją w dniu 03.07.2021 r. [źródło: własne].

¹Github, https://github.com/dccstcc/SoftDrive_tests [dostęp: 02.09.2021].

The screenshot shows a list of commits from a GitHub repository. The commits are as follows:

- construct initial IP address** by **dccstcc** committed on 3 Jul at 3 Jul 2021, 16:01 CEST.
- Find 3rd quarter od ip address for Android WiFi router** by **dccstcc** committed on 3 Jul.
- change project catalogs structure** by **dccstcc** committed on 3 Jul.
- refactor** by **dccstcc** committed on 3 Jul.
- debug distance read** by **dccstcc** committed on 3 Jul.
- configure rest for read a distance** by **dccstcc** committed on 3 Jul.

Rysunek 8.2: Końcowy czas pracy nad implementacją w dniu 03.07.2021 r. [źródło: własne].

Czas przeznaczony na implementację w dniu 03.07.2021 r. wyniósł 2,5 godziny.

The screenshot shows a list of commits from a GitHub repository. The commits are as follows:

- add Log.v()** by **dccstcc** committed 19 days ago.
- add app icon** by **dccstcc** committed 19 days ago.
- add animation** by **dccstcc** committed 19 days ago.
- init logo animation** by **dccstcc** committed 19 days ago.
- autorun app modules onCreate** by **dccstcc** committed 19 days ago.
- init rest activity layout** by **dccstcc** committed 19 days ago.
- refactor** by **dccstcc** committed 19 days ago.

At the bottom right of the commit list, there is a timestamp: **14 Aug 2021, 09:01 CEST**.

Rysunek 8.3: Początkowy czas pracy nad implementacją w dniu 14.08.2021 r. [źródło: własne].

-o- Commits on Aug 14, 2021

- add display stay turn on**
dccstcc committed 19 days ago
- speed alert debug** 14 Aug 2021, 12:09 CEST
dccstcc committed 19 days ago
- prepare to use**
dccstcc committed 19 days ago
- debug app start**
dccstcc committed 19 days ago
- refactor**
dccstcc committed 19 days ago
- add Log.v()**
dccstcc committed 19 days ago

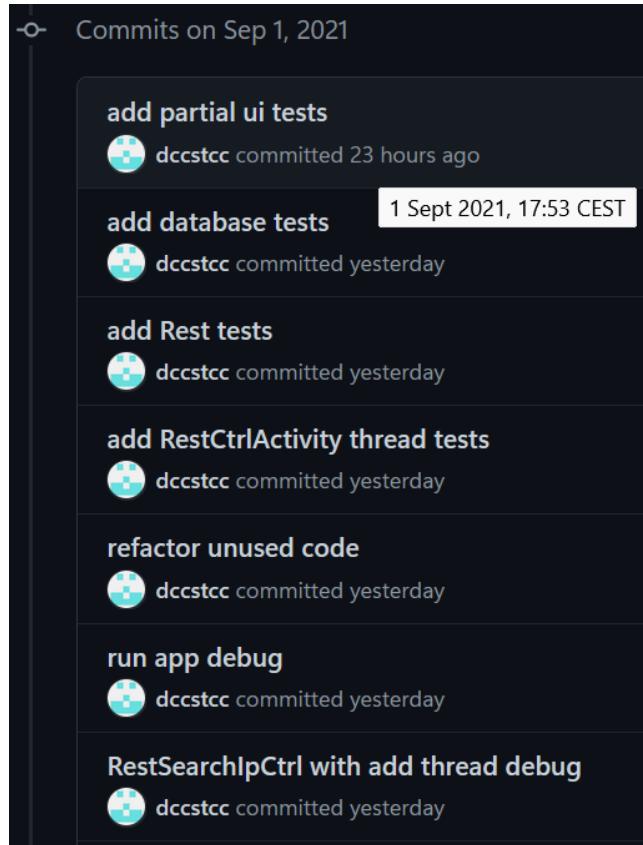
Rysunek 8.4: Końcowy czas pracy nad implementacją w dniu 14.08.2021 r. [źródło: własne].

Czas przeznaczony na implementację w dniu 01.09.2021 r. wyniósł 3 godziny.

- add RestCtrlActivity refactor**
dccstcc committed yesterday
- add FindAddressIp refactor**
dccstcc committed yesterday
- add RestCtrl partial refactor**
dccstcc committed 2 days ago
- add GetDistanceCtrl partial refactor**
dccstcc committed 2 days ago
- add RestCtrlActivity partial tests**
dccstcc committed 2 days ago
- add EnableRouterActivity tests**
dccstcc committed 2 days ago

1 Sept 2021, 02:31 CEST

Rysunek 8.5: Końcowy czas pracy nad implementacją w dniu 01.09.2021 r. [źródło: własne].

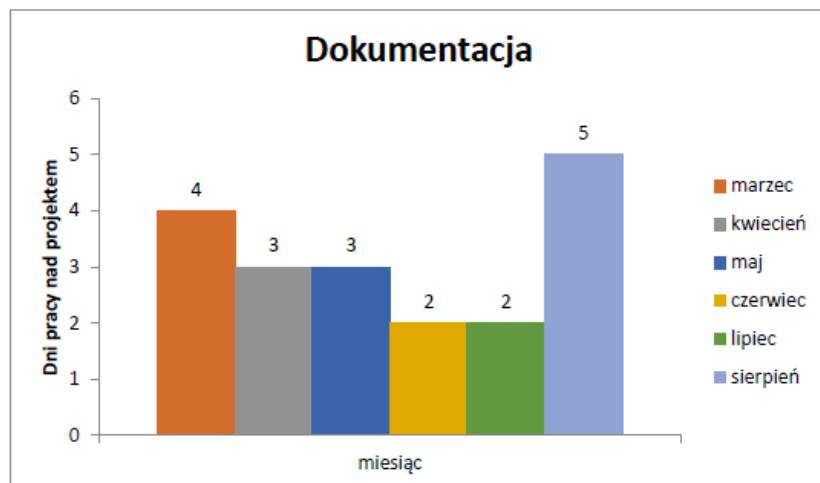


Rysunek 8.6: Końcowy czas pracy nad implementacją w dniu 01.09.2021 r. [źródło: własne].

Czas przeznaczony na implementację w dniu 01.09.2021 r. wyniósł 15 godzin.

Na wykazanej podstawie czasu pracy, przybliżony czas prac programistycznych w ciągu 1 dnia wynosi 7 godzin. W ciągu miesiąca na prace projektowe przeznaczono co 2 dzień.

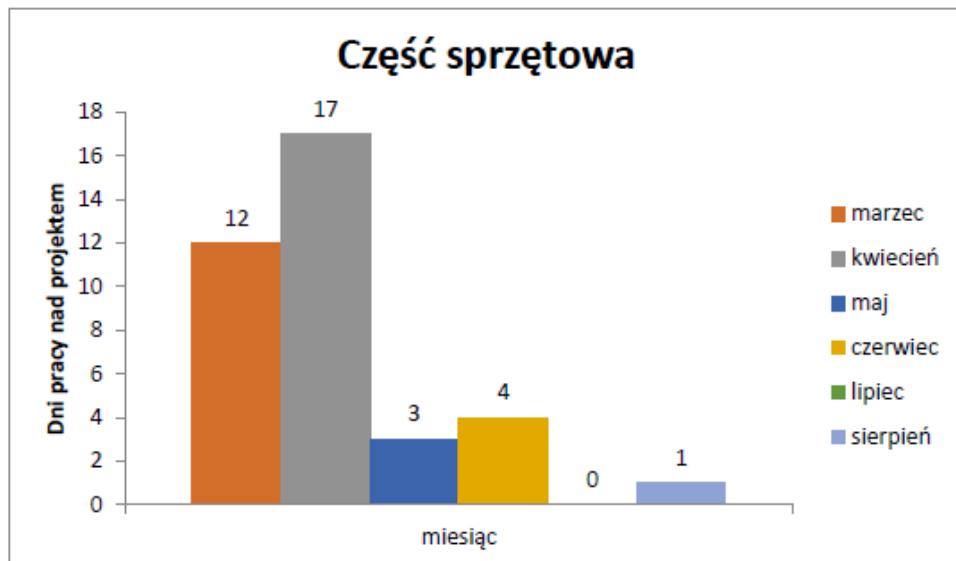
8.1.1 Dokumentacja



Rysunek 8.7: Wymiar czasu realizacji dokumentacji projektu [źródło: własne]

Realizacja dokumentacji na przestrzeni czasu 6 miesięcy była systematyczna, zgodnie z charakterem prac, wymagała systematycznej i przyrostowej realizacji.

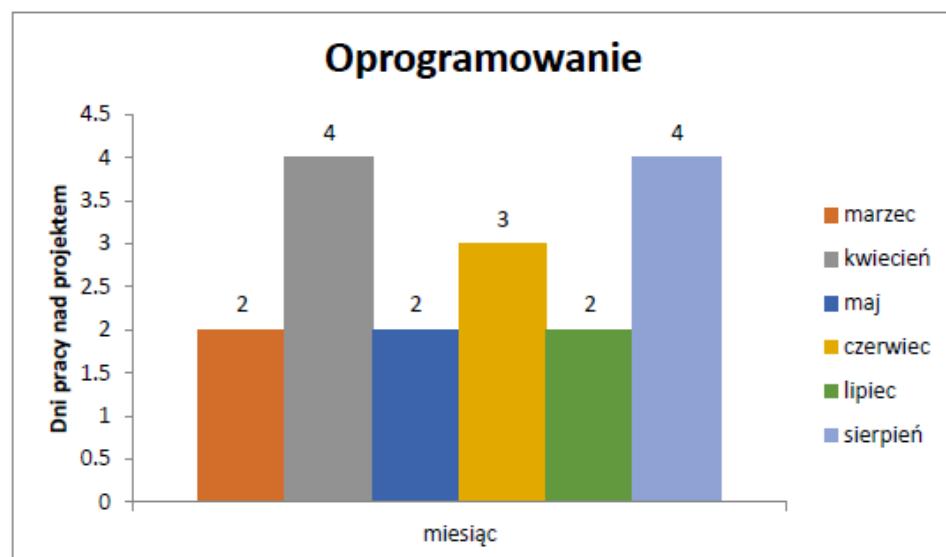
8.1.2 Część sprzętowa



Rysunek 8.8: Wymiar czasu realizacji części sprzętowej [źródło: własne]

Realizacja części sprzętowej pokrywała się z realizacją wstępnych założeń projektowych w dokumentacji w ciągu pierwszych 3 miesięcy realizacji projektu i została w tym okresie w znacznym stopniu zrealizowana.

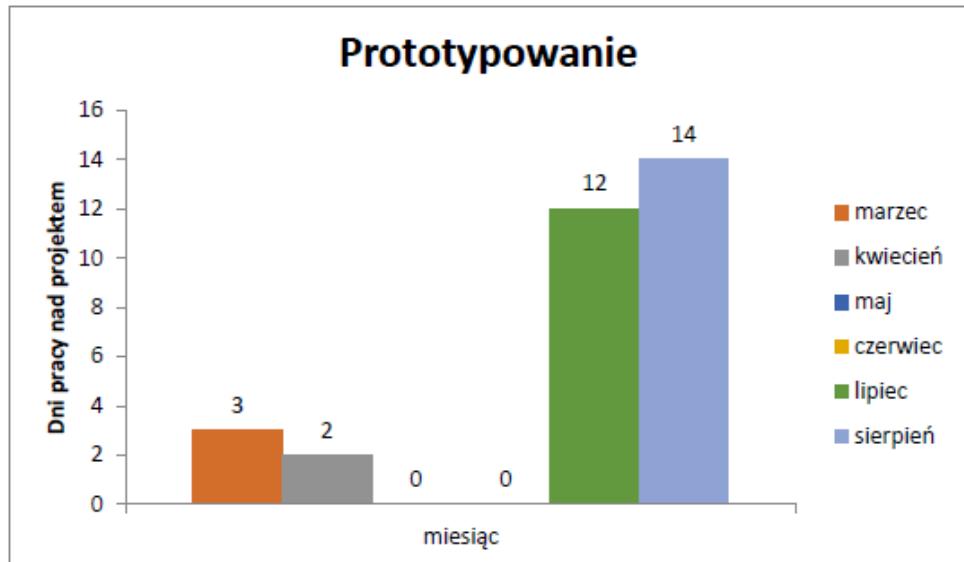
8.1.3 Oprogramowanie



Rysunek 8.9: Wymiar czasu realizacji oprogramowania [źródło: własne]

Realizacja oprogramowania wymiarem wymaganego czasu na realizację była zbliżona do charakterystyki czasowej tworzenia dokumentacji i zgodnie z przyjętą metodologią hybrydową: kaskadową i iteracyjno-przyrostową przez cały czas realizacji projektu była realizowana w małych przyrostach, które miały uporządkowaną kolejność.

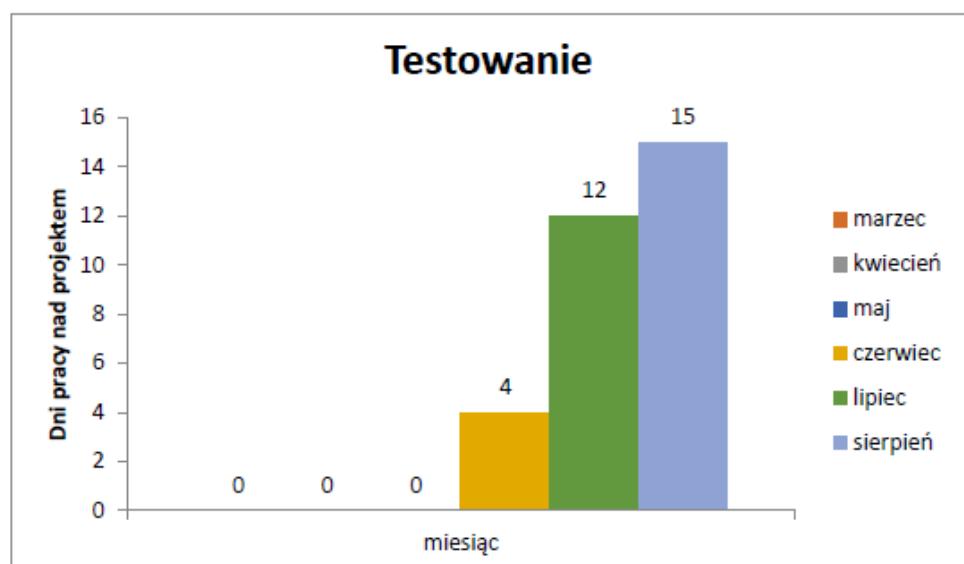
8.1.4 Prototypowanie



Rysunek 8.10: Wymiar czasu realizacji prototypu [źródło: własne]

Realizacja prototypu w charakterystyce czasu miała miejsce w początkowym okresie realizacji projektu, podczas intensyfikacji realizacji części sprzętowej, z którą funkcjonalnie była powiązana koniecznością zestawienia prototypu dla części sprzętowej. Najwięcej pracy nad prototypem było w końcowym okresie realizacji projektu kiedy wymagane było jednoczesne testowanie sprzętu w zestawieniu z oprogramowaniem.

8.1.5 Testowanie



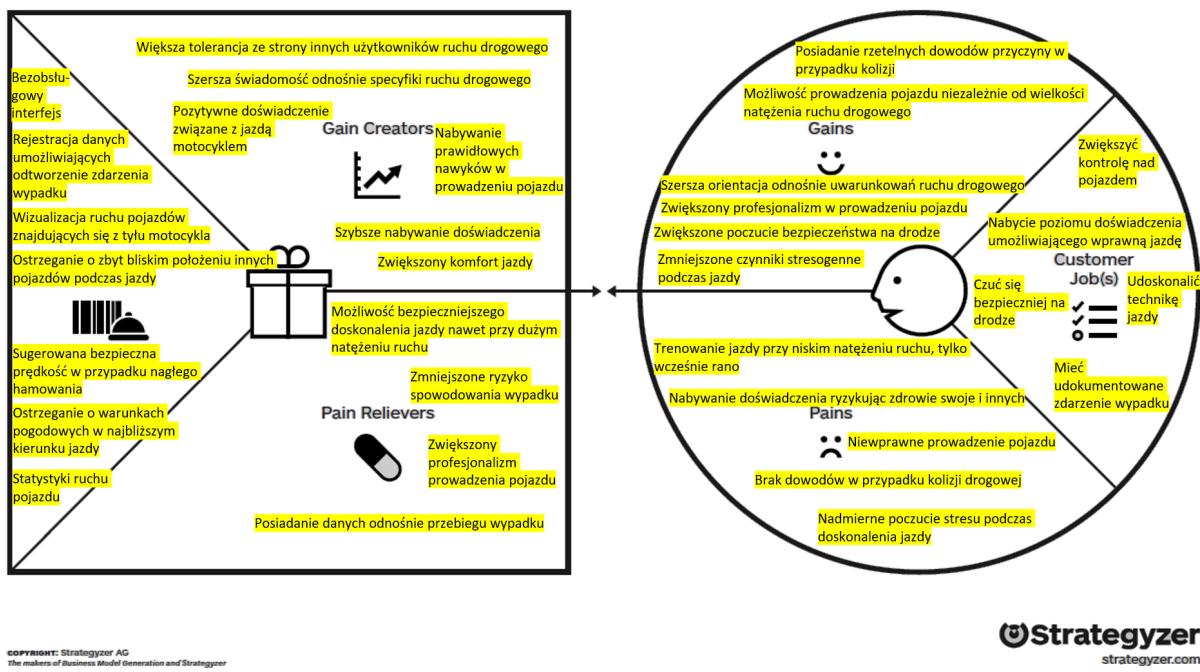
Rysunek 8.11: Wymiar czasu realizacji testów [źródło: własne]

Proces testowania miał ramy czasowe ustalone w końcowej fazie realizacji projektu, kiedy możliwe stało się testowanie zestawionego prototypu.

8.2 Potencjał komercjalizacyjny

Analiza biznesowa zrealizowanego przedsięwzięcia.

8.2.1 Analiza Value Proposition Canvas



Rysunek 8.12: Analiza Value Proposition Canvas [źródło: własne].

Z przeprowadzonej analizy Value Proposition Canvas wynika, że finalny produkt projektowanego systemu rozwiązuje bolączki początkowego użytkownika pojazdu motocyklowego, który chciałby w mniejszym wymagającym, niż standardowy sposób zdobyć większe doświadczenie jako kierowca.

Oferowane rozwiązanie, dzięki istniejącym funkcjonalnością (sekcja Products and Services) umożliwia zmniejszenie bolączek (sekcja Pains Relievers) użytkownika (sekcja Pains), związanych z wymaganiami stojącymi przed nim (sekcja Customers jobs).

Wychodząc na przeciw problemom użytkownika, dzięki usługom wdrażanego produktu, możliwe jest podniesienie jakości pozytywnych skutków (Gain Creators) wynikających z rozwiązania problemów i zaspokojenia potrzeb użytkownika (Gains).

8.2.2 Analiza TAM SAM SOM

Narzędzie analizy biznesowej wdrożenia produktu Value Proposition Canvas.

TAM: Interesariusze branży motocyklowej

SAM: Promotorzy urządzeń wspomagających jazdę motocyklem

SOM: Ośrodki szkolenia kierowców, Ośrodki egzaminujące na Prawo Jazdy, Portale internetowe

Istnieje 4983 ośrodków szkolenia kierowców w Polsce²

Ośrodki szkolenia kierowców same rejestrują się na stronie, stąd pochodzą statystyki. Za wiarygodnością danych przemawia aktualność oferty ośrodków, ze względu na aktywne poszukiwanie przez nie rynku zbytu. Za małą wiarygodnością danych przemawia brak naukowych reguł statystycznych przy ich pozyskiwaniu oraz nieoficjalne źródło tych danych.

OSK stanowią miejsce reklamy w postaci ulotek, plakatów, polecenia ze strony osób szkolących lub wdrożenia podczas nauczania jazdy. Osoby polecające otrzymują udział w 25% zysku ze sprzedaży produktu. Będą identyfikowane po kuponie zniżkowym w wysokości 5% wartości produktu, który przekaze nam klient z polecenia. Klient kupon otrzyma od instruktora z OSK.

Planowany jest wzrost liczby wizytowanych ośrodków w przypadku współpracy z pracownikami marketingu.

Istnieje 49 ośrodków egzaminacyjnych na Prawo Jazdy w Polsce³

Dane są wiarygodne, pomimo że dane pochodzą z nieoficjalnego źródła, za wiarygodnością danych przemawia charakter prawny ośrodków egzaminacyjnych w Polsce.

Wojewódzkie Ośrodki Ruchu Drogowego stanowią miejsce reklamy w postaci ulotek i plakatów z planowanymi docelowo stanowiskami reklamowymi z promującymi produkt pracownikami marketingu.

Zabiegi marketingowe w internecie szacuje się na 15 stron www tygodniowo.

Z powodu utrudnionego dotarcia bezpośrednio do potencjalnego klienta, wykorzystano środki pośrednie, z których potencjalny klient może korzystać: internet, ośrodki szkolenia kierowców, ośrodki egzaminacyjne na Prawo Jazdy.

W oparciu o analizę TAM SAM SOM i podejście bottom-up, w początkowym okresie jestem w stanie dotrzeć do:

1. 10 ośrodków szkolenia kierowców tygodniowo (2 dziennie).
2. 15 potencjalnych kanałów reklamy poprzez sieć www (3 dziennie).
3. 2 Ośrodków egzaminujących na Prawo Jazdy w miesiącu (1 na dwa tygodnie).

²Ranking szkół jazdy, <https://www.prawo-jazdy-360.pl/ranking-szkol-jazdy> [dostęp 04.09.2021]

³Ośrodki Egzaminacyjne na Prawo Jazdy w Polsce, http://www.fallcon.pl/s-Osrodkie_egzaminacyjne_WORD_w_Polsce-63.html [dostęp 04.09.2021].

8.2.3 Kanały dotarcia

Statystyki rejestracji wszystkich pojazdów w roku 2020 określają 1656383 nowych rejestracji, w tym zarejestrowano 109636 pojazdów motocyklowych.⁴

Udział liczby zarejestrowanych motocykli w ogólnej liczbie wszystkich pojazdów wynosi w przybliżeniu: 6,6% .

1. Kanał dotarcia: Ośrodki Szkolenia Kierowców

W kalkulacji przyjęto, że spośród wszystkich OSK w Polsce na Prawo Jazdy kat. A, AM, A1, A2 kształci 6,6 % spośród nich. Powstała przybliżona ilość 350 OSK bazuje na oszacowaniu ogólnej liczby zarejestrowanych liczby pojazdów jednośladowych w stosunku do wszystkich zarejestrowanych pojazdów w kraju.

Zakładając skuteczność na przybliżonym poziomie 2%, wskaźnik konwersji stanowi 2% w skali roku jestem w stanie dotrzeć pośrednio do potencjalnych klientów za pomocą 2% ośrodków szkolenia kierowców w ciągu pierwszego roku. Skuteczność 2% z 350 OSK stanowi w przybliżeniu 7 OSK spośród wszystkich 350 OSK.

Po 9 miesiącach wizytowane zostaną wszystkie OSK z ofertą szkoleń dla przyszłych motocyklistów. po tym czasie możliwa jest rewizytacja wcześniej odwiedzonych OSK i kontynuacja współpracy.

Przyjmując dalej wskaźnik konwersji na poziomie 2% oraz ustalając liczbę nowych kierowców motocykli w ciągu miesiąca w przybliżeniu na 9000 osób, co stanowi liczba rejestrowanych przez właścicieli pojazdów podzielona przez 12 miesięcy oraz przyjmując liczbę OSK na poziomie 350, można oszacować, że w każdym OSK w Polsce, każdego miesiąca szkoli się 26 motocyklistów. Uwzględniając wskaźnik 2% konwersji z liczby 26 przyszłych motocyklistów można w przybliżeniu oszacować, że każdego miesiąca zyskujemy jednego klienta.

Jeden OSK w ciągu miesiąca generuje dochód od jednego klienta. Siedem OSK jest w stałej współpracy, w tym zestawieniu co miesiąc notujemy sprzedaż siedmiu produktów.

- (a) W kalkulacji przyjęto fikcyjny udział ośrodków, które kształcą kierowców na kat. A, AM, A1, A2 spośród wszystkich OSK, ze względu na brak takich danych statystycznych.
- (b) W kalkulacji pominięto także udział kierowców kat. B, którzy mogą zarejestrować i prowadzić pojazd motocyklowy do 125 cm³ pojemności i 15 kW mocy silnika, ze względu na trudność w dokładniejszym oszacowaniu takich statystyk.
- (c) W kalkulacji pominięto liczbę kierowców, którzy posiadają już Prawo Jazdy uprawniające do jazdy motocyklem. Tacy kierowcy rejestrujący nowy motocykl nie są klientami OSK. Powodem pominięcia statystyk jest brak odpowiednich danych.
- (d) Uogólniono także liczbę klientów przypadających na każdy pojedynczy OSK, pomijając nierówności w liczbie klientów wynikające z konkurencji i marketingu charakterystycznych dla każdego OSK, powodem jest brak publicznych danych dla analizowanych OSK.

Posiadanie pominiętych zestawień umożliwiłoby dokładniejszą kalkulację liczby potencjalnych klientów projektowanego rozwiązania.

⁴CEPIK, <http://www.cezik.gov.pl/statystyki> [dostęp 04.09.2021].

2. Kanał dotarcia: Ośrodki egzaminacyjne

W przypadku działań marketingowych w Wojewódzkich Ośrodków Egzaminacyjnych na Prawo Jazdy, szkielet działań stanowi promocja w postaci ulotek i plakatów, z którymi potencjalni nabywcy zdający egzamin na Prawo Jazdy będą się mogli zapoznać. Wśród 49 ośrodków odwiedzanych z częstotliwością dwóch na miesiąc, po roku czasu wizytowana zostanie połowa wszystkich ośrodków.

Zakładając wskaźnik konwersji na poziomie 2% i szacowną liczbę osób przystępujących do egzaminu na poziomie 15% (wartość losowa, nie statystyczna, zakłada liczbę osób, które rejestrują pojazd, ale nie mają jeszcze zdanego egzaminu na Prawo Jazdy) z liczby wszystkich właścicieli rejestrujących swoje pojazdy jednośladowe można oszacować, że po roku czasu 15% z liczby 109636 osób rejestrujących pojazd w urzędzie, stanowiący w przybliżeniu 16000 osób, z uwzględnieniem 2% wskaźnika konwersji, wyniesie 320 klientów w ciągu roku.

Po drugim roku liczba klientów wyniesie dwukrotność liczby osób po pierwszym roku czyli 640 klientów, ze względu na wizytację nie odwiedzonej drugiej połowy ośrodków egzaminacyjnych.

Po pierwszym roku, w ciągu miesiąca zyskuję 26 nowych klientów miesięcznie, co stanowi wartość 320 klientów rocznie podzielona na 12 miesięcy.

Kalkulacja nie uwzględnia pokrywania się klientów zainteresowanych produktem w OSK i ośrodkach egzaminacyjnych jednocześnie.

3. Kanał dotarcia: Portale internetowe

Ten kanał dotarcia nie zostanie przedstawiony w postaci kalkulacji ze względu na złożoność kalkulacji i brak wiarygodnych danych statystycznych, jednakże całą kalkulację uwzględnia ten kanał dotarcia, z którego możliwe jest znalezienie nowych klientów, zwiększąc w ten sposób szansę utrzymania się na rynku, pomimo braku kalkulacji.

8.2.4 Porównanie na tle konkurencji

1. AIM technologies

Ze względu na zdominowanie zarejestrowanych pojazdów motocyklowych przez pojazdy używane o średniej wieku 18 lat, które w większości przypadków konstrukcyjnie nie posiadają jednostki ECU oraz elektronicznych systemów wspomagania jazdy, taki rynek zbytu produktu wydaje się mniej konkurencyjny i bardziej niszowy dla wdrażanego przeze mnie produktu. Na takim rynku istniejące rozwiązania w postaci desek rozdzielczych skomunikowanych z ECU, nie są konkurencyjne, ze względu na wiek pojazdów.

Czołowy producent tego typu rozwiązań firma „AIM”⁵, ma w swojej ofercie przeważnie rozwiązania bazujące na ECU, ma jednak również alternatywy dla starszych pojazdów w postaci sensorów i dostawek zwiększających funkcjonalność desek rozdzielczych dla pojazdów niewyposażonych w ECU.

⁵ AIM, <https://www.aimtechnologies.com/motorcycle/> [dostęp: 04.09.2021]

W przypadku starszych pojazdów oferowane przez konkurenta rozwiązanie nie uwzględnia prostych metod montażu i pomija aspekty bezpieczeństwa ruchu pojazdów, otwierając okno możliwości dla tego typu rozwiązań, które zapewnia wdrażany przeze mnie produkt, uzupełniając tą lukę, wykorzystując szansę zaistnienia na rynku z nowym produktem, pomimo istnienia silnego konkurenta.

Kolejną szansą jest niska cena oferowanego przeze mnie produktu w stosunku do rozwiązań firmy „AIM”. Mój produkt jest adresowany do użytkowników motocykli o średniej zamożności, nie uwzględnia na tym etapie wdrożenia współpracy korporacyjnej, w przeciwieństwie do firmy „AIM”, która takie usługi świadczy.

Docelowy klient mojego produktu w porównaniu z ofertą konkurencji, może po niego sięgnąć decydując się na prostotę instalacji, niższą cenę i funkcjonalność poprawy bezpieczeństwa, która nie jest oferowana przez podobne produkty. Przedstawione statystyki pokazują, że większość motocyklistów w Polsce posiada starsze modele pojazdów najprawdopodobniej ze względu ekonomicznych, dlatego oferta konkurencji może być dla nich nieosiągalna, zarówno pod względem ekonomicznym jak i technologicznym.

2. Urządzenia mobilne

Istnieją bezpłatne alternatywy dla wdrażanego produktu w postaci aplikacji mobilnych, które można pobrać np. ze sklepu Google Play⁶ ⁷ ⁸ w systemie Android. Aplikacje te jednak mają funkcjonalność ograniczoną tylko do możliwości połączenia z internetem i bazowych sensorów telefonu, dodatkowo wszystkie są podobne pod względem funkcjonalnym. Moje rozwiązanie rozszerza tę funkcjonalność o dostępne dodatkowe sensory, stanowiąc alternatywę.

3. Podobna architektura sprzętowa

Istnieje już rozwiązanie w postaci aplikacji mobilnej i bezprzewodowo skomunikowanych czujników zewnętrznych do instalacji na pojeździe⁹. Takie rozwiązania jest tożsame w oferowanym przeze mnie produkcie, jednakże to co wyróżnia moją ofertę to wyakcentowanie aspektu bezpieczeństwa w poruszaniu się po drodze.

⁶Luxhaminsurande, Ranking aplikacji dla motocyklistów, <https://www.lexhaminsurance.co.uk/blog/the-8-best-motorcycle-apps-for-ios-and-android/> [dostęp: 04.09.2021]

⁷Lowbrowcustoms, Porównanie aplikacji dla motocyklistów, <https://www.lowbrowcustoms.com/blogs/motorcycle-product-reviews/top-10-best-motorcylce-apps> [dostęp: 04.09.2021]

⁸DriveModeDashboard, Aplikacja dla motocyklistów, <https://www.drivemode dashboard.com/> [dostęp: 04.09.2021]

⁹<https://www.yougarage.net/>

8.2.5 Szanse i zagrożenia wdrożenia produktu

Ze względu na istniejące już w tym segmencie rynku podobne rozwiązania i silne marki np. „AIM”, dostępnymi kanałami aby zaistnieć na rynku z nowym produktem są:

1. rozszerzona o sensory zewnętrzne inne niż oferuje smartfon funkcjonalność będąca konkurencją dla bezpłatnych rozwiązań,
2. niższa cena, mniej skomplikowana instalacja oraz większa dostępność dla starszych pojazdów powinna stanowić niszową konkurencję dla firmy „AIM”,
3. akcent na poprawę bezpieczeństwa poruszania się po drodze odróżnia mój produkt od podobnych, bazujących na aplikacji i czujnikach zewnętrznych rozwiązaniach konkurencji.

Zagrożeniem dla zdrożenia produktu może być niski popyt na systemy bezpieczeństwa w motocyklu. Taki wniosek nasuwa się po analizie statystyk dotyczących bezpieczeństwa na drodze, które z jednej strony pokazują dużą potrzebę uwrażliwienia na ten aspekt, a z drugiej duże nadużycie ze strony kierowców w tej kwestii, a więc brak zainteresowania aby w praktyce stosować systemy ostrzegające o zagrożeniu bezpieczeństwa na drodze.

W odniesieniu do statystyk dotyczących niepełnoletnich kierowców motocykli, nabywcą ekonomicznym może być rodzina, a docelowym kierowca. Rodzice w trosce o dziecko mogą wymagać od nich używania produktu, który poprawi ich bezpieczeństwo na drodze.

Szansą powodzenia wdrożenia produktu na rynek mogą być akcje promujące bezpieczeństwo uczestników ruchu drogowego.

Nabywcą produktu może być również kierowca pragnący zwiększyć świadomość o stanie warunków bezpieczeństwa panujących na drodze.

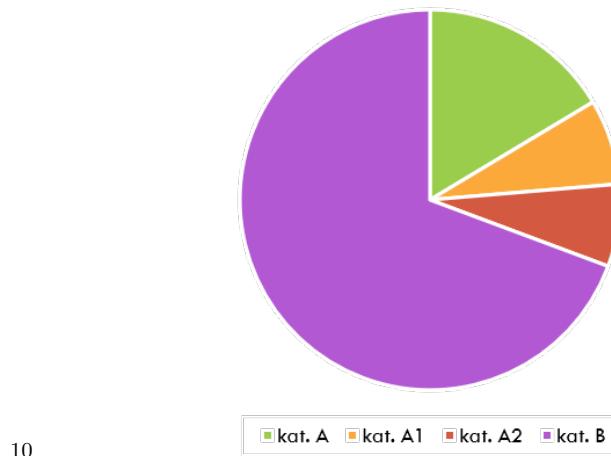
Nabywcą produktu może być również początkujący kierowca, który wykorzystałby wdrażany produkt w celu nabycia większego doświadczenia lub z troski o bezpieczeństwo własne i innych.

8.2.6 Specyfika rynku

Na rynku zbytu istnieją podobne rozwiązania od innych dostawców. Na podstawie przeprowadzonych analiz rynku z uwzględnieniem silnych stron konkurencji, można wyciągnąć następujące wnioski odnośnie przewag rynkowych oferowanego rozwiązania:

1. Isnieje niska rynkowa dla oferowanego rozwiązania.
2. Wykcentowanie aspektu bezpieczeństwa.
3. Dedykacja dla starszych pojazdów.
4. Dedykacja dla początkujących kierowców.
5. Dedykacja dla młodych kierowców.
6. Atrakcyjne cenowo rozwiązanie.

Za wyciągniętymi wnioskami przemawiają statystyki branży motocyklowej:

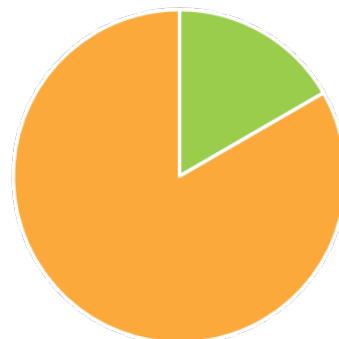


10

Rysunek 8.13: Utrata uprawnień do prowadzenia pojazdami wg kategorii Prawa Jazdy w 2020 r. [dostęp 04.09.2021]

11

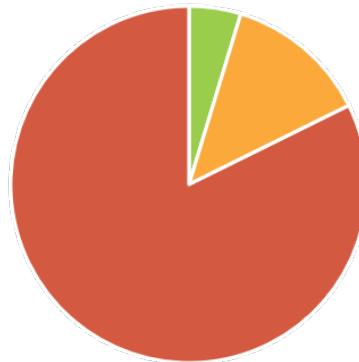
■ Kierowcy niepełnoletni ■ Kierowcy pełnoletni



Rysunek 8.14: Udział niepełnoletnich kierowców, którzy utracili Prawo Jazdy w 2020 r. [dostęp 04.09.2021]

12

■ z winy motocyklisty ■ bez winy motocyklisty ■ bez udziału motocyklisty



Rysunek 8.15: Udział kierujących pojazdami jednośladowymi w ogólnej liczbie wypadków w 2018 r. [dostęp 04.09.2021]

Średni wiek motocykla to 17,9 lat dla pojazdu posiadającego polisę OC w 2017 r.¹³

Isnieje niska rynkowa pośród istniejących rozwiązań konkurencji. Niszę tę stanowi akcent na aspekty poprawy bezpieczeństwa przez oferowany produkt.

Statystyki ruchu drogowego w zestawieniu z branżą motocyklową przemawiają za dużym zapotrzebowaniem na rozwiązania poprawiające bezpieczeństwo ruchu drogowego, szczególnie wśród kierowców młodych lub dopiero rozpoczynających jazdę motocyklem.

W dodatku oferowane rozwiązanie jest atrakcyjne cenowo w porównaniu do rozwiązań oferowanych przez konkurencję, jednakże akcent poprawy bezpieczeństwa nie napędza znacznie popytu produktu. Tematyka poprawy bezpieczeństwa ruchu drogowego nie jest bardzo popularna, choć potrzebna. Tą tezę zdają się potwierdzać przedstawione statystyki.

8.2.7 Strategia cenowa

Koszt wytworzenia produktu stanowi kwotę 2000 zł. Cenę rynkową ustala się na dwukrotność kosztów wytworzenia: 4000 zł. Najtańszy produkt głównego konkurenta - firmy "AIM" wynosi: 4000 zł. Wskaźnik LTV wynosi: 2000 zł. stanowi kwotę zysku Wskaźnik CAC wynosi 1/3 kwoty zysku, w przybliżeniu wynosi: 700 zł. - jest to kwota przeznaczona na promocję i marketing dla pojedynczego sprzedanego egzemplarza produktu.

Z poniesionych na marketing i promocję kosztów, po roku czasu, zyskujemy 7 miesięcznie klientów poprzez kanał dotarcia w postaci OSK i 26 klientów miesięcznie poprzez kanał dotarcia w postaci ośrodków egzaminujących. Kanał dotarcia w postaci e-commerce pominięto. Sumarycznie co miesiąc zyskujemy 33 nowych klientów. Każdy klient generuje zysk 2000 zł.

Miesięczny zysk wynosi w założonej analizie 66000 zł.

¹³Sejm RP, <http://www.sejm.gov.pl/sejm8.nsf/interpelacja.xsp?documentId=53DF1C3928200208C125812B004ABB4E> [dostęp 04.09.2021]

Bibliografia

1. Żydzik K. Rak T. C# i MVC5, Helion 2015
2. Barteczko K. JAVA Uniwersalne Techniki Programowania, PWN 2015
3. KRBRD, <http://www.krbrd.gov.pl/pl/tag/statystyki-wypadkow-drogowych> [dostęp: 04.09.2021]
4. INFOR, <https://mojafirma.infor.pl/moto/prawo-na-drodze/predkosc-i-fotoradary/319047,Fotoradar-a-bezpieczenstwo-na-drogach.html>[dostęp: 04.09.2021]
5. Encyklopedia PWN, <https://encyklopedia.pwn.pl/haslo/stres;3980346.html> [dostęp: 0.09.2021]
6. Medycyna Praktyczna, <https://www.mp.pl/pacjent/neurologia/aktualnosci/197635,stres-pogarsza-pamiec-i-zmniejsza-mozg> [dostęp: 04.09.2021]
7. Poradnik Biznesu, <http://www.poradnikbiznesu.info/samochod-w-firmie/innowacje-w-motoryzacji> [dostęp: 04.09.2021]
8. Wynagrodzenia, <https://wynagrodzenia.pl/artykul/podsumowanie-ogolnopolskiego-badania-wynagrodzen-w-2018-roku> [dostęp: 04.09.2021]
9. CEPIK, <http://www.cepik.gov.pl/statystyki> [dostęp: 04.09.2021]
10. Elecena, ceny elementów elektronicznych, <https://elecena.pl/report> [dostęp 04.09.2021]
11. PZPM, Polski Związek Przemysłu Motoryzacyjnego, <https://www.pzpm.org.pl/Rynek-motoryzacyjny/Park-pojazdow-zarejestrowanych/Tabele-Park-pojazdow-zarejestrowanych-w-Polsce-1990-2018> [dostęp: 23.05.2021]
12. Policja, <https://policja.pl/pol/aktualnosci/190688,Bezpieczenstwo-motocyklistow-na-drodze.html> [dostęp: 23.05.2021]
13. Kamami, sklep z elektroniką, <https://kamami.pl/czujniki-odleglosci/562733-lidar-lite-v3-laserowy-czujnik-odleglosci-0-40-m-sen-14032.html> [dostęp 03.09.2021]
14. Infona, Portal Komunikacji Naukowej, <https://www.infona.pl/resource/bwmeta1.element.baztech-article-BPC6-0001-0086/tab/summary> [dostęp: 27.05.2021]
15. Yadda, Baza danych o zawartości polskich czasopism technicznych, <https://yadda.icm.edu.pl/baztech-0ee28f97-19ac-4960-8d41-224057de95e7>, str. 58-59, 65 [dostęp: 27.05.2021]
16. Przegląd modeli cyklu życia oprogramowania, https://www.researchgate.net/profile/Rafal-Kasprzyk/publication/229596276_Przeglad_modeli_cyklu_zycia_oprogramowania/links/5b4209744585modeli-cyklu-zycia-oprogramowania.pdf, str. 52-53 [dostęp: 27.05.2021]
17. Apache, <http://www.apache.org/licenses/LICENSE-2.0> [dostęp: 27.08.2021]
18. GitHub, https://github.com/garmin/LIDARLite_RaspberryPi_Library [dostęp: 27.08.2021]
19. Kompikownia, <https://www.kompikownia.pl/index.php/2018/07/26/rest-api-w-c-biblioteka-casablanca-cpprest/> [dostęp: 27.08.2021]
20. GitHub Adafruit, https://github.com/adafruit/Adafruit_CircuitPython_LIDARLite [dostęp: 27.08.2021]

21. AGH, <http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.baztech-article-AGH1-0025-0083> [dostęp: 27.08.2021]
22. Github, https://github.com/dccstcc/SoftDrive_tests [dostęp: 02.09.2021]
23. Ranking szkół jazdy, <https://www.prawo-jazdy-360.pl/ranking-szkol-jazdy> [dostęp 04.09.2021]
24. Ośrodki Egzaminacyjne na Prawo Jazdy w Polsce, http://www.fallcon.pl/s-Osrodkiegzaminacyjne_W63.html [dostęp 04.09.2021]
25. CEPIK, <http://www.cezik.gov.pl/statystyki> [dostęp 04.09.2021]
26. AIM, <https://www.aimtechnologies.com/motorcycle/> [dostęp: 04.09.2021]
27. Luxhaminsurande, Ranking aplikacji dla motocyklistów, <https://www.lexhaminsurance.co.uk/blog/the-best-motorcycle-apps-for-ios-and-android/> [dostęp: 04.09.2021] Porównanie aplikacji dla motocyklistów, <https://www.lowbrowcustoms.com/blogs/motorcycle-product-reviews/top-10-best-motorcycle-apps> [dostęp: 04.09.2021]
28. Lowbrowcustoms, Porównanie aplikacji dla motocyklistów, <https://www.lowbrowcustoms.com/blogs/product-reviews/top-10-best-motorcycle-apps> [dostęp: 04.09.2021]
29. DriveModeDashboard, Aplikacja dla motocyklistów, <https://www.drivemodedashboard.com/> [dostęp: 04.09.2021]
30. Sejm Rp <http://www.sejm.gov.pl/sejm8.nsf/interpelacja.xsp?documentId=53DF1C3928200208C125> [dostęp 04.09.2021]

Spis rysunków

1.1	Rich Picture	13
3.1	Schemat modelu iteracyjnego	26
3.2	Schemat modelu ewolucyjno-przyrostowego	27
3.3	Schemat modelu kaskadowego	28
3.4	Schemat modelu V	29
4.1	Diagram przypadków użycia UML	34
4.2	Diagram klas UML	40
4.3	Diagram sekwenacji UML	41
4.4	Diagram czynności UML wstępnej konfiguracji	43
4.5	Diagram czynności UML odczytu odległości	44
4.6	Diagram czynności UML rysowania GUI	45
4.7	Prototyp interfejsu systemu	49
4.8	Prototyp interfejsuinicjalizacji systemu	50
4.9	Schemat bazy danych	51
5.1	Adapter USB UART	56
5.2	Niepoprawne wykrywanie adaptera USB-UART w systemie	57
5.3	Poprawne wykrywanie adaptera USB-UART w systemie	58
5.4	Specyfikacja GPIO w Raspberry Pi	59
5.5	Wyrowadzenia w adapterze USB-UART	60
5.6	Ustawienie widoczności ukrytych plików w systemie Windows	61
5.7	Konfiguracja dostępu do SSH	62
5.8	Ustawienia połączenia szeregowego	63
5.9	Konfiguracja ustawień połączenia szeregowego	64
5.10	Aktualizacja pakietów w systemie Raspbian	65
5.11	Narzędzie raspi-config	66
5.12	Nazwa hosta w raspi-config	66
5.13	Logowanie w raspi-config	67
5.14	Łączność internetowa w raspi-config	68
5.15	Kamera w raspi-config	68
5.16	Strona kodowa w raspi-config	69
5.17	Strefa czasowa w raspi-config	69
5.18	konfiguracja konta użytkownika w RPi	70
5.19	Problemy z logowaniem do GUI w RPi	71
5.20	Modyfikacja uprawnień pliku .Xauthority	72
5.21	GUI dla raspi-config	73
5.22	interfejs CSI w RPi	74
5.23	Podłączanie kamery do RPi	74
5.24	Obraz z kamery w RPi	75
5.25	Optymalizacja ustawień kamery	76
5.26	Radiatory w Rpi	81

5.27 Działanie skanera obiektów	83
5.28 Biblioteka Flask	85
5.29 Działanie biblioteki Flask	86
5.30 Skrypt startowy w RPi	87
5.31 Prototyp tylnego systemu wbudowanego	88
5.32 Struktura plików przedniego systemu wbudowanego	89
5.33 Odczyt odległości z czujnika	92
5.34 Błąd odczytu odległości z czujnika	93
5.35 Nasłuchiwanie odczytu odległości w RPi	95
5.36 Pierwsza wersja prototypu	97
5.37 przedłużanie przewodowego połączenia z czujnikiem	98
5.38 Zastosowanie miernika wielkości elektrycznych	99
5.39 Zmodyfikowany prototyp	100
5.40 Struktura projektu aplikacji	102
5.41 Struktura pakietu rest w aplikacji	103
5.42 Pierwszy layout aplikacji	104
5.43 Drugi layout aplikacji	105
5.44 Projekt prototypu realizacji projektu	117
5.45 Rzeczywisty prototyp	118
5.46 Połączenia w prototypie	118
5.47 Kondensatory w prototypie	119
5.48 Przedłużenie przewodowe w prototypie	119
5.49 Montaż czujnika	120
5.50 Laminat PCB z wygrawerowanymi w technologii CNC ścieżkami	120
 6.1 Testy w strukturze projektu	122
6.2 Wynik przeprowadzonych testów	122
6.3 nowa struktura projektu	127
6.4 Testy bliskiej odległości	133
6.5 Testy dalekiej odległości	134
6.6 Praca systemu podczas postoju	134
6.7 Praca systemu podczas detekcji zagrożenia ruchu drogowego	135
6.8 Praca systemu podczas braku odczytu odległości	135
6.9 Weryfikacja poprawności interakcji z GUI	136
 8.1 Początkowy czas pracy nad implementacją w dniu 03.07.2021 r.	144
8.2 Końcowy czas pracy nad implementacją w dniu 03.07.2021 r.	145
8.3 Początkowy czas pracy nad implementacją w dniu 14.08.2021 r.	145
8.4 Końcowy czas pracy nad implementacją w dniu 14.08.2021 r.	146
8.5 Końcowy czas pracy nad implementacją w dniu 01.09.2021 r.	146
8.6 Końcowy czas pracy nad implementacją w dniu 01.09.2021 r.	147
8.7 Wymiar czasu realizacji dokumentacji projektu	147
8.8 Wymiar czasu realizacji części sprzętowej	148
8.9 Wymiar czasu realizacji oprogramowania	148
8.10 Wymiar czasu realizacji prototypu	149
8.11 Wymiar czasu realizacji testów	149
8.12 Analiza Value Proposition Canvas	150
8.13 Utrata uprawnień do prowadzenia pojazdami wg kategorii Prawa Jazdy w 2020 r.	156
8.14 Udział niepełnoletnich kierowców, którzy utracili Prawo Jazdy w 2020 r.	156
8.15 Udział kierujących pojazdami jednośladowymi w ogólnej liczbie wypadków w 2018 r.	156

Kody źródłowe

4.1	Model logiczny nazewnictwa obiektów bazy danych	51
4.2	Model logiczny warstwy pośredniczącej dla bazy danych	51
4.3	Model logiczny warstwy kontrolera dla bazy danych	52
5.1	plik konfiguracyjny wpa_supplicant.conf	65
5.2	plik konfiguracyjny motion.conf	77
5.3	skrypt producenta dla skanera przestrzeni RPLidar: display_lidar_pi.py	84
5.4	przystosowany do REST API skrypt producenta dla skanera przestrzeni RPLidar: rest_server.py	85
5.5	skrypt startowy systemu wbudowanego tylnego run_rest_server.sh	86
5.6	skrypt powłoki bash: run_rest.sh	89
5.7	plik startowy: main.cpp	90
5.8	plik Service.cpp	91
5.9	plik Sh_script_exec.cpp	91
5.10	plik run_rest.py	94
5.11	plik klasy RestCtrlActivity.java	106
5.12	plik klasy FindAddressIp.java	107
5.13	plik klasy GetDistanceCtrl.java	108
5.14	plik interfejsu RestApi z ustawionym endpointem	109
5.15	plik interfejsu IFromRestCallback przechwytyującego odpowiedzi serwera przez bibliotekę Retrofit	109
5.16	plik interfejsu IFromRestCallback przechwytyującego odpowiedzi serwera przez bibliotekę Retrofit	109
5.17	Klasa kontrolera usługi rest integrująca opisywane moduły	109
5.18	Klasa kontrolera bazy danych	111
5.19	Klasa kontrolera GUI	112
6.1	Testy dla klasy MainActivity	123
6.2	Testy dla klasy EnableRouterActivity	124
6.3	Testy dla klasy RestCtrlActivity	125
6.4	Testy dla klasy RestSearchIpCtrl	126
6.5	Testy dla klasy DbManager	126
6.6	Implementacja klasy MainActivity po refactoryzacji	128
6.7	Implementacja klasy EnableRouterActivity po refactoryzacji	129
6.8	Implementacja klasy ActivitySingleton po refactoryzacji	130
6.9	Implementacja klasy FindAddressIp po refactoryzacji	130
6.10	Implementacja klasy RestCtrl po refactoryzacji	131
6.11	Implementacja klasy RestGetDistanceCtrl po refactoryzacji	131
6.12	Implementacja klasy RestSearchIpCtrl po refactoryzacji	132