Politechnika Łódzka
Wydział Elektrotechniki, Elektroniki, Informatyki i Automatyki
Instytut Informatyki Stosowanej

PRACA DYPLOMOWA MAGISTERSKA

System monitorowania położenia użytkownika dla potrzeb aplikacji mobilnej wykorzystującej rzeczywistość mieszaną (mixed reality)

System for monitoring user position in mobile mixed reality application

Piotr Marcińczyk
213381

Opiekun pracy:
dr inż. Piotr Duch

Łódź, wrzesień 2018

Spis treści

Spis skrótów **4**

Streszczenie **5**

Abstract **6**

1. Wprowadzenie **7**
1.1. Wstęp 7
1.2. Cele i zakres pracy 8

2. Istniejące rozwiązania **9**
2.1. Mieszana rzeczywistość 9
2.2. Sposoby śledzenia położenia 11
2.2.1. Bezwładnościowe 11
2.2.2. Optyczne 15
2.2.3. Inne metody 21

3. Zastosowane rozwiązania **23**
3.1. System DecaWave 23
3.2. Aplikacja serwerowa 30
3.3. Aplikacja mobilna 38

4. Testy działania **45**
4.1. Autopozycjonowanie 46
4.2. Pomiary statyczne 49
4.3. Pomiary dynamiczne 55
4.4. Testy subiektywne 59
4.5. Test zużycia energii elektrycznej 60

5. Wnioski **62**

Bibliografia **67**

Spis ilustracji **69**

Spis skrótów

AAR	Android Archive
AR	Augmented Reality
AV	Augmented Virtuality
FPS	Frames Per Second
IEEE	Institute of Electrical and Electronics Engineers
IMU	Inertial Measurement Unit
IR	Infrared Radiation
JSON	JavaScript Object Notation
JTAG	Joint Test Action Group
LCD	Liquid Crystal Display
LED	Light-Emitting Diode
LOS	Line-of-sight
MR	Mixed Reality
Ni-MH	Nickel Metal Hydride battery
NLOS	Non-line-of-sight
RR	Real Reality
SPI	Serial Peripheral Interface
SWD	Serial Wire Debug
TCP	Transmission Control Protocol
TDM	Time Division Multiplexing
ToF	Time of Flight
TWR	Two Way Ranging
UART	Universal Asynchronous Receiver and Transmitter
UDP	User Datagram Protocol
USB	Universal Serial Bus
UWB	Ultra Wide Band
VR	Virtual Reality

Streszczenie

Niniejsza praca magisterska opisuje realizację systemu monitorującego położenie użytkownika w świecie rzeczywistym za pomocą fal radiowych ultraszerokiego pasma (UWB) i następnie przekładającego ją do aplikacji rzeczywistości mieszanej (MR) działającej na urządzeniu mobilnym. Wyznaczanie pozycji odbywa się za pomocą zestawu **DecaWave TREK1000**, który umożliwia pomiar odległości między znajdującymi się w nim urządzeniami z dokładnością do kilku centymetrów, określając czas przebycia sygnału radiowego (ToF). Przy wykorzystaniu trzech z nich jako nieruchome punkty odniesienia oraz jednego umieszczonego przy użytkowniku, za pomocą trilateracji określane jest położenie, które następnie przekłada się na pozycję kamery w wirtualnej scenie. Oprogramowanie układów zestawu zostało zmodyfikowane na potrzeby pracy, aby uzyskać jak najlepszą wydajność lokalizacji. Przykładowa aplikacja mobilna została utworzona w środowisku **Unity** na system operacyjny **Android** przy wykorzystaniu frameworka **Google VR SDK**. Przeprowadzono również badania pozwalające stwierdzić w obiektywny sposób użyteczność opracowanego rozwiązania do zastosowań rzeczywistości mieszanej.

Słowa kluczowe: rzeczywistość wirtualna, rzeczywistość mieszana, lokalizacja w pomieszczeniu, urządzenia mobilne, łączność bezprzewodowa.

Abstract

The following master thesis describes realisation of a system for monitoring user position in a real world using ultra wide band radio waves and then transforming it to a mixed reality (MR) application running on a mobile device. Position determining is done by **DecaWave TREK1000** kit which allows measurements of the distance between its devices with the precision of a few centimeters by calculating time of flight of radio waves signal (ToF). By using three of them as stationary points of reference and one mounted on the user, the position is calculated using trilateration algorithm, which is then transformed into camera location on the virtual scene. The firmware of these devices has been modified for the needs of this thesis to obtain the best efficiency of ranging. A sample mobile application is created in **Unity** environment for **Android** operating system with **Google VR SDK**. There have been made several tests to check usability in a objective way of the developed solution for mixed reality applications.

Keywords: virtual reality, mixed reality, indoor localization, mobile devices, wireless communicaiton.

Rozdział 1

Wprowadzenie

1.1. Wstęp

> *„When people take off the headset, they immediately have a creative idea about what they can make in virtual reality, and a lot of them immediately want to get involved."*
>
> — Brendan Iribe, współzałożyciel Oculus VR[(1)]

Wirtualna rzeczywistość jest bardzo szybko rozwijającą się dziedziną informatyki oraz elektroniki. Na rynku ciągle pojawiają się coraz to nowsze rozwiązania pozwalające na zanurzenie w świecie generowanym przez komputery. Często dla zwiększenia interaktywności z wirtualną rzeczywistością wykorzystywane są kontrolery w postaci rękawic, kamer śledzących ruchy użytkownika, a nawet rozwiązania umożliwiające przenoszenie ruchów całego ciała do świata wirtualnego. Rosnąca moc obliczeniowa urządzeń przenośnych, takich jak smartfony i tablety pozwala wykorzystać je do wyświetlania wirtualnego świata. Ponadto często posiadają one wbudowane czujniki określające pozycję w przestrzeni, dzięki czemu, po umieszczeniu ich w specjalnych okularach, mogą służyć jako gotowe rozwiązanie przenoszące użytkownika w wirtualną rzeczywistość.

---

(1) TechCrunch: *Oculus CEO Brendan Iribe Donates $31M To Build VR Lab At His Alma Mater University Of Maryland*, 12.09.2014, `https://techcrunch.com/2014/09/11/brendan-iribe-center-for-computer-science/` (dostęp 05.09.2018)

1.2. Cele i zakres pracy

Celem niniejszej pracy jest przygotowanie aplikacji na urządzenia mobilne, która będzie generowała świat wirtualny i umożliwiała poruszanie się w nim przekładając rzeczywistą pozycję użytkownika do środowiska wirtualnego. Wyznaczenie pozycji w świecie rzeczywistym odbędzie się za pomocą fal radiowych, przy wykorzystaniu systemu DecaWave. Do określenia orientacji użytkownika w przestrzeni zostaną wykorzystane sensory wbudowane w urządzenie mobilne, na którym będzie działać wspomniana wcześniej aplikacja. Do najpopularniejszych z nich należą: żyroskop, akcelerometr oraz kompas. Odpowiednie połączenie danych otrzymywanych z tych czujników pozwala na płynne działanie, co następnie przekłada się na lepsze wrażenia w świecie wirtualnym.

Dodatkowym założeniem jest uzyskanie systemu, który będzie łatwy w konfiguracji i intuicyjny w obsłudze dla potencjalnego użytkownika końcowego. Praca zawiera ocenę działania uzyskanego systemu, możliwe rozwiązania poprawiające jego skuteczność, jak i alternatywne zastosowania, niezwiązane bezpośrednio z tematyką pracy.

Rozdział 2

Istniejące rozwiązania

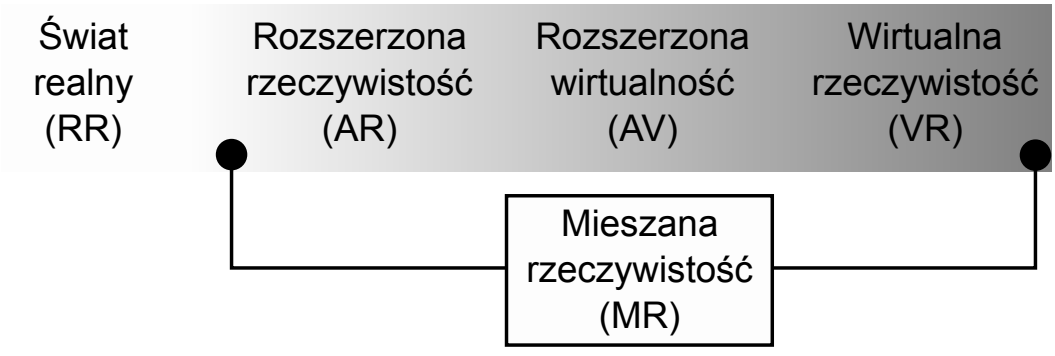
2.1. Mieszana rzeczywistość

Na rynku istnieje wiele systemów umożliwiających śledzenie pozycji oraz ruchów użytkownika w pomieszczeniu i następnie przeniesienie jej do świata wirtualnego[(1)]. Często dokonywany jest ich podział na wirtualną (*ang. VR – Virtual Reality*) i rozszerzoną rzeczywistość (*ang. AR – Augmented Reality*). Pierwsza z nich charakteryzuje się tym, że obraz jest w całości generowany przez komputer. Użytkownik patrząc przez specjalny zestaw gogli ze słuchawkami, odcina się całkowicie od świata realnego (*ang. RR – Real Reality*). Najczęściej każde z oczu otrzymuje osobny obraz przy wykorzystaniu miniaturowych ekranów i specjalnych soczewek. Dzięki temu uzyskuje się wrażenie głębi. Za pomocą czujników pozycyjnych takich jak żyroskop, akcelerometr i kompas określana jest orientacja głowy, co umożliwia przełożenie jej na ruchy kamery znajdującej się w wirtualnej scenie. Poruszanie i interakcja odbywa się przy użyciu kontrolerów typu gamepad, obsługiwanego za pomocą obydwu rąk lub dwóch oddzielnych na każdą dłoń osobno. Często ich ruch w przestrzeni także jest określany wbudowanymi czujnikami bezwładnościowymi (*ang. IMU – Inertial Measurement Unit*), a na wirtualnej scenie wyświetlane są odpowiedniki kończyn albo kursory.

(1) Road to VR: *Overview of Positional Tracking Technologies for Virtual Reality*, `https://www.roadtovr.com/overview-of-positional-tracking-technologies-virtual-reality/` (dostęp 05.09.2018)

Rozszerzona rzeczywistość z kolei, skupia się na wzbogacaniu rzeczywistości o komputerowo generowane obiekty. Interakcja z uzyskanym światem jest mocno ograniczona, a wykorzystanie pojedynczej kamery do przechwytywania sceny powoduje, że otrzymany obraz nie jest stereoskopowy. Miejsca, w których mają zostać umieszczone komputerowe modele oznacza się specjalnymi markerami/znacznikami w postaci wydrukowanego na kartce papieru wzoru. Alternatywnie do określenia orientacji otoczenia wykorzystuje się algorytmy śledzenia obrazu (*ang. optical flow*), co pozwala wyeliminować konieczność stosowania wspomnianych znaczników, a pozycja generowanych obiektów określana jest przez użytkownika. Przykładem takiej aplikacji jest program umożliwiający testowanie różnego rozmieszczenia wirtualnych mebli w rzeczywistym pomieszczeniu[(2)].

Obydwa systemy, jak również wszelkie pośrednie formy między nimi określa się jako rzeczywistość mieszaną (*ang. MR – Mixed Reality*) [1][2]. Takim przykładem może być rozszerzona wirtualność (*ang. AV – Augmented Virtuality*), której założenia są podobne do VR, ale często zawiera elementy świata rzeczywistego, z którymi można wchodzić w interakcję. Schemat przedstawiający porządek poszczególnych rzeczywistości widoczny jest na ilustracji 2.1. Dalsza część rozdziału opisuje wybrane rozwiązania dostępne na rynku związane z rzeczywistością mieszaną, ze szczególnym naciskiem na sposób interakcji i określania położenia użytkownika w wirtualnym świecie.



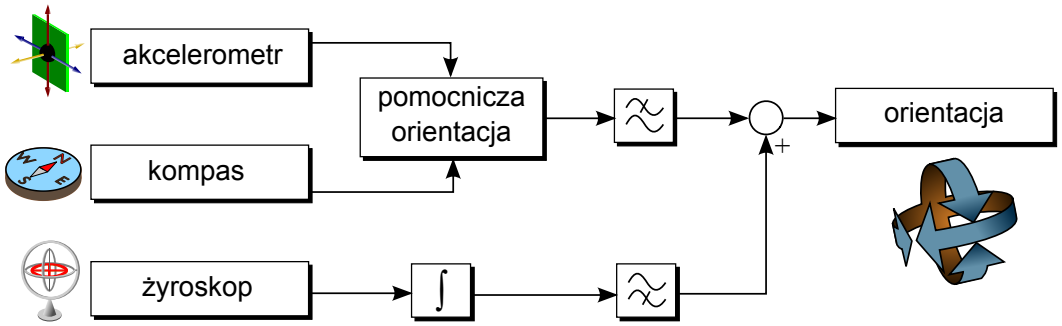
Ilustr. 2.1: Podział rozszerzonych i wirtualnych rzeczywistości

[(2)] IKEA Place: `https://www.ikea.com/au/en/apps/IKEAPlace.html` (dostęp 05.09.2018)

2.2. Sposoby śledzenia położenia

2.2.1. Bezwładnościowe

Jak wspomniano we wprowadzeniu, metody bezwładnościowe wykorzystują zestaw czujników (IMU) na który najczęściej składają się: akcelerometr, żyroskop i kompas. Odpowiednie połączenie wskazań tych sensorów umożliwia bardzo precyzyjne określenie orientacji (kąta obrotu) w przestrzeni śledzonego urządzenia (najczęściej okularów VR). Ilustracja 2.2 przedstawia reprezentację graficzną typowego łączenia danych z poszczególnych sensorów [3][4] w celu uzyskania końcowego wyniku. W pewnym uproszczeniu – pomiary z akcelerometru oraz kompasu służą do wykrywania powolnych zmian położenia. Ten pierwszy pozwala określić kierunek grawitacji, a drugi azymut, czyli kąt w płaszczyźnie poziomej, i następnie stosuje się filtr dolnoprzepustowy w celu wyeliminowania zakłóceń. Odczyty prędkości obrotowej z żyroskopu podlegają całkowaniu (kwadraturze) po czasie i po wyłuskaniu szybkich zmian, dane są łączone z tymi uzyskanymi w poprzednim etapie z pozostałych dwóch sensorów. Takie rozwiązanie minimalizuje efekty szumów wysokiej częstotliwości akcelerometru oraz powolny dryf żyroskopu, przejawiający się wskazywaniem niezerowej prędkości obrotu w spoczynku.



Źródło: *P. Lawitzki* [4]

Ilustr. 2.2: Diagram łączenia pomiarów z sensorów IMU

Bezwładnościowe metody określania orientacji są najczęściej wykorzystywaną metodą w aplikacjach mieszanej rzeczywistości (MR) dostępnych na urządzeniach mobilnych, na przykład smartfonach. Większość z nich posiada wspomniane czujniki, które pozwalają na uzyskanie modułu IMU w sposób programowy lub – co-

raz częściej – wbudowany gotowy sprzętowy przetwornik wyznaczający orientację znacznie szybciej, a przez to płynniej. Pozwoliło to na dynamiczny rozwój systemów MR opartych o urządzenia mobilne, dostępne praktycznie dla każdego. Jednym z pierwszych i zarazem najpopularniejszych jest **Google VR SDK**, początkowo znane jako **Google Cardboard**[3]. Głównym założeniem było umożliwienie w łatwy sposób zanurzenie się w świat wirtualnej rzeczywistości każdego użytkownika ze smartfonem oraz zwiększenie zainteresowania tą technologią wśród deweloperów. W 2014 roku na rynek zostały wypuszczone niedrogie okulary wykonane z tektury, dwóch soczewek oraz magnesu, z instrukcją do własnoręcznego złożenia – widoczne na ilustracji 2.3. Pozwalały one na umieszczenie w nich telefonu z uruchomioną

Źródło: *Wikipedia* [4]

Ilustr. 2.3: Kartonowe okulary VR Google Cardboard

aplikacją Cardboard i trzymając gotowe gogle przy oczach – uzyskanie wielu interaktywnych trójwymiarowych symulacji i filmów przestrzennych. Sterowanie odbywało się za pomocą magnetycznego przycisku wbudowanego w okulary, którego ruch był wykrywany przez magnetometr znajdujący się w smartfonie. Instrukcje

[3] Google VR for everyone: `https://developers.google.com/vr/` (dostęp 05.09.2018)
[4] Wikipedia: *Google-Cardboard.jpg*, `https://en.wikipedia.org/wiki/File:Google-Cardboard.jpg` (dostęp 05.09.2018)

potrzebne do budowy okularów zostały udostępnione na otwartej licencji, dzięki czemu każdy mógł wydrukować i złożyć własne, kosztem jedynie niezbędnych materiałów. Na rynku szybko pojawiły się modyfikacje gogli w szerokim przedziale cenowym, a rozwój technologii mieszanej rzeczywistości opartej na urządzeniach mobilnych nabrał szybkiego tempa. W połowie 2015 roku pojawiła się druga wersja Cardboard wspierająca telefony o większych ekranach oraz działające pod systemem operacyjnych **iOS**, a magnetyczny przycisk został zastąpiony przewodzącą pianką imitującą dotyk palca. Umożliwiło to wykorzystanie kompasu w urządzeniu mobilnym do zwiększenia dokładności określania orientacji w przestrzeni.

Kolejnym etapem ewolucji tej platformy jest system **Google Daydream** przeznaczony dla wybranych smartfonów z najwyższej półki[(5)]. Okulary zostały zaprezentowane w listopadzie 2016 roku i są wykonane z plastiku otoczonego miękkim materiałem, który może zostać zdjęty i wyczyszczony. Do sterowania służy niewielki bezprzewodowy pilot posiadający własny moduł IMU do wykrywania orientacji, 2 przyciski, regulację poziomu głośności oraz touchpad umożliwiający przeniesienie ruchu palca do aplikacji i interakcję z jej elementami.

W międzyczasie Samsung we współpracy z Oculusem zaprezentował własne rozwiązanie pozwalające użytkownikom smartfonów tej firmy na wykorzystanie ich w aplikacjach mieszanej rzeczywistości.[(6)] Pod koniec 2015 roku, w listopadzie na rynek wprowadzono okulary **Samsung Gear VR**. W porównaniu do innych systemów tego typu, posiadają one wbudowany moduł IMU umożliwiający śledzenie orientacji ze znacznie lepszą dokładnością niż sam telefon. Gear VR charakteryzuje się wysoką jakością wykonania, w szczególności zastosowanymi soczewkami[(7)]. Interakcja z aplikacjami odbywa się za pomocą bezprzewodowego kontrolera widocznego na ilustracji 2.4 razem z okularami. Zasilany jest dwoma bateriami rozmiaru AAA i łączy się z telefonem za pomocą Bluetooth. Funkcjonalność jest podobna do kontrolera dołączanego do okularów Google Daydream, jednak posiada

[(5)] Google: *Daydream*, `https://vr.google.com/daydream/` (dostęp 05.09.2018)

[(6)] Samsung Gear VR with Controller: *Samsung Gear VR with Controller*, `https://www.samsung.com/global/galaxy/gear-vr/` (dostęp 05.09.2018)

[(7)] VR & AR Wiki: *Samsung Gear VR (2015/2016)*, `https://xinreality.com/wiki/Samsung_Gear_VR_(2015/2016)` (dostęp 05.09.2018)

4 przyciski oraz trigger (przycisk imitujący spust broni) znajdujący się pod palcem wskazującym. Ponadto design jest przez wielu uznawany za bardziej ergonomiczny i intuicyjny, na co wpływ miała współpraca z Oculusem posiadającym doświadczenie w projektowaniu urządzeń MR[8].

Źródło: *Samsung* [6]

Ilustr. 2.4: Okulary Samsung Gear VR wraz z kontrolerem

Bardziej zaawansowane systemy IMU umożliwiają również określanie pozycji (przesunięcia) w pomieszczeniu. Niestety, są one podatne na dryf żyroskopu oraz szumy i niepewności pomiarowe zastosowanych czujników. Do określenia pozycji konieczne jest wykonywanie kwadratury (całkowania) odczytów, co na przykład w przypadku akcelerometru przekłada się na kumulację szumów w tempie kwadratowym. To z kolei powoduje wraz z upływem czasu coraz większą rozbieżność pozycji od rzeczywistości. Niemniej jednak, w przypadku częstego korygowania błędów za pomocą innego systemu odniesienia, dokładność pozycjonowania zwykle jest więcej niż wystarczająca. Z tego względu, bezwładnościowe metody śledzenia stosuje się jako uzupełnienie innych rozwiązań, które są bardziej dokładne niż IMU w określaniu położenia w dłuższym odcinku czasu i na większej powierzchni. Warto zauważyć, że żadne z opisanych powyżej rozwiązań mieszanej rzeczywisto-

[8] Road to VR: *Gear VR Controller Review*, `https://www.roadtovr.com/samsung-gear-vr-with-controller-review/` (dostęp 05.09.2018)

ści na urządzenia mobilne nie umożliwia śledzenia jednocześnie pozycji i orientacji metodą bezwładnościową.

2.2.2. Optyczne

Kolejnymi, po bezwładnościowych, najczęściej stosowanymi rozwiązaniami do śledzenia położenia użytkownika w mieszanej rzeczywistości są metody optyczne. Opierają się one na wykorzystaniu kamer wideo zamontowanych w urządzeniu i rejestrujących punkty statyczne (*ang. inside-out* – ze środka na zewnątrz) lub rozmieszczonych w pomieszczeniu i śledzących ruchy użytkownika (*ang. outside-in* – z zewnątrz do środka).

Te pierwsze są naturalnym rozwiązaniem w przypadku znacznej części mobilnych systemów rozszerzonej rzeczywistości. Kamera zamontowana w smartfonie lub tablecie śledzi znaczniki znajdujące się w kadrze i na tej podstawie określa swoje położenie oraz generuje wirtualne elementy nakładane na obraz. Takie rozwiązanie można znaleźć w wielu aplikacjach znajdujących się w sklepach. Alternatywnie, urządzenie może automatycznie określić punkty szczególne i na ich podstawie śledzić ruchy sceny. Przykładem takiego rozwiązania jest dodatek do aplikacji kamery w telefonach Sony Xperia „AR effect”[(9)] pozwalającego na dodanie do sceny trójwymiarowych animowanych postaci. Przykładowy efekt działania widoczny jest na ilustracji 2.5.

Jednym z urządzeń wykorzystujących tę metodę śledzenia położenia jest **Microsoft HoloLens** ogłoszony na początku 2015 roku i dostępny w wersji deweloperskiej rok później [5]. Są to okulary mieszanej rzeczywistości (MR), umożliwiające nakładanie komputerowo generowanych grafik na rzeczywisty obraz. Nie wymaga on do działania zewnętrznego urządzenia, ponieważ posiada wbudowany komputer działający pod systemem operacyjnym Windows 10. Do wyświetlania obrazu zastosowana została para specjalnych przezroczystych soczewek holograficznych, które przepuszczają światło z zewnątrz i jednocześnie pozwalają na nakładanie grafik za

[(9)] Google Play: *AR effect*, `https://play.google.com/store/apps/details?id=com.sonymobile.androidapp.cameraaddon.areffect` (dostęp 05.09.2018)

Ilustr. 2.5: Zdjęcie wykonane w aplikacji Sony AR effect

pomocą miniaturowego projektora[10]. Każde oko otrzymuje lekko przesunięty obraz, co pozwala na uzyskanie efektu głębi. Zastosowana technologia wyświetlacza ma swoje wady pod względem ograniczonego pola widzenia: 30° w poziomie i 17,5° w pionie[11]. Określanie położenia odbywa się przy wykorzystaniu szerokokątnej kamery głębokości, kamery wideo wysokiej rozdzielczości, 4 kamer śledzących, 4 mikrofonów, czujnika jasności otoczenia oraz modułu IMU. Sterowanie zrealizowano poprzez rozpoznawanie gestów wykonywanych dłońmi za pomocą kamer oraz komend głosowych wbudowanymi mikrofonami i asystentem Microsoft Cortana. W urządzeniu znajdują się dwa głośniki imitujące dźwięk przestrzenny oraz gniazdo jack 3,5 mm umożliwiające podłączenie zewnętrznych słuchawek. Wygląd okularów HoloLens został pokazany na ilustracji 2.6. Alternatywnie do interakcji może zostać wykorzystany pilot **HoloLens Clicker**. Posiada on wbudowane śledzenie orientacji oraz przycisk, co pozwala na bardziej precyzyjne sterowanie i zastępuje konieczność stosowania gestów ręką.

[10] The Imaginative Universal: *How Hololens Displays Work*, `http://www.imaginativeuniversal.com/blog/2015/10/18/how-hololens-displays-work/` (dostęp 05.09.2018)
[11] VR & AR Wiki: *Microsoft HoloLens*, `https://xinreality.com/wiki/Microsoft_HoloLens` (dostęp 05.09.2018)

Źródło: *Microsoft* [5]

Ilustr. 2.6: Urządzenie Microsoft HoloLens

Innym popularnym urządzeniem wykorzystującym optyczne śledzenie pozycji typu *inside-out* jest **HTC Vive** wprowadzony na rynek w kwietniu 2016 roku[(12)]. Są to okulary mieszanej rzeczywistości podłączane do komputera za pomocą kabla HDMI do karty graficznej, USB do komunikacji oraz kabla zasilającego. HTC Vive to część systemu **SteamVR**, który jest rozszerzeniem platformy **Steam**[(13)] opracowanej przez firmę **Valve** i oferuje gry oraz aplikacje mieszanej rzeczywistości współpracujące nie tylko z tymi okularami, ale także podobnymi rozwiązaniami jak Oculus Rift czy urządzenia Windows Mixed Reality. Najnowsza wersja **HTC Vive Pro** widoczna na ilustracji 2.7 oferuje wyświetlacze o większej rozdzielczości 1440 × 1600 pikseli każdy. Ponadto dostępne są dwie kamery, które pozwalają na mieszanie realnych elementów (RR) z symulacją. Wbudowany moduł IMU umożliwia śledzenie zarówno orientacji oraz pozycji i jest wspierany systemem **Lighthouse**.[(15)] Wykorzystuje on stacje bazowe umiejscowione w pomieszczeniu, mające postać prostopadłościanów również widocznych na ilustracji 2.7. Każda z nich wysyła dwie wiązki podczerwonego lasera obracające się w prostopadłych do siebie

[(12)] VIVE: *Discover Virtual Reality Beyond Imagination*, `https://www.vive.com/eu/` (dostęp 05.09.2018)
[(13)] *Virtual Reality on Steam*: `https://store.steampowered.com/` (dostęp 05.09.2018)
[(14)] Sector: *HTC predstavilo profesionálne balenie Vive Pro headsetu s novými senzormi* `https://www.sector.sk/novinka/149449/htc-predstavilo-profesionalne-balenie-vive-pro-headsetu-s-novymi-senzormi.htm`, 24.04.2018 (dostęp 05.09.2018)
[(15)] VR & AR Wiki: *Lighthouse*, `https://xinreality.com/wiki/Lighthouse` (dostęp 05.09.2018)

Źródło: *Sector.sk* (14)

Ilustr. 2.7: HTC Vive Pro z kontrolerami i stacjami bazowymi Lighthouse

płaszczyznach z częstotliwością około 60 Hz. Sygnał jest odbierany przez czujniki optyczne znajdujące się na okularach (zlokalizowane wewnątrz charakterystycznych wgłębień), które mierzą czas od błysku synchronizującego i na tej podstawie określają kąt odchylenia od stacji. Następnie wykorzystując znane odległości między czujnikami obliczana jest pozycja okularów w pomieszczeniu.(16) Teoretycznie do prawidłowego działania wystarczająca jest jedna stacja bazowa oraz 5 czujników na śledzonym obiekcie. Jednak ze względów praktycznych stosuje się dwie stacje wraz z kilkudziesięcioma czujnikami, co zwiększa dokładność i pozwala na pozycjonowanie urządzeń w obrębie całego pomieszczenia na powierzchni do 5 × 5 m. Producent zadbał także o bezpieczeństwo korzystania z systemu pozycjonowania oferując technologię **Chaperone**. Umożliwia ona zaznaczenie powierzchni bezpiecznej do poruszania się oraz pozycji przeszkód znajdujących się w pokoju. W przypadku zbliżania się do określonej przez użytkownika granicy pomieszczenia, zostanie wyświetlone wyraźne ostrzeżenie z możliwością podglądu otoczenia przez wbudowane w okulary kamery. Ponadto dodatkową funkcjonalnością systemu jest możliwość dostosowania symulacji lub rozgrywki do dostępnej przestrzeni. Firma Valve postanowiła udostępnić szczegóły techniczne oraz czujniki systemu Lighthouse, aby zwiększyć dostępność urządzeń wspierających tę metodę lokalizacji opracowywanych przez zewnętrzne firmy. Dzięki temu ta technologia może zostać

(16) YouTube: Scott Rumschlag: *Lighthouse Tracking: Basics*, 04.04.2017, `https://www.youtube.com/watch?v=Ikv8o0u6w9o` (dostęp 05.09.2018)

wykorzystana także poza mieszaną rzeczywistością, na przykład w pozycjonowaniu dronów, robotów i precyzyjnym wymiarowaniu pomieszczeń.[17][18]Interakcja w systemie HTC Vive odbywa się za pomocą dwóch bezprzewodowych kontrolerów **SteamVR Controller** przedstawionych na ilustracji 2.7 obok okularów. Posiadają one kilka przycisków, trigger oraz touchpad. Moduł IMU wraz z czujnikami optycznymi współpracując z systemem Lighthouse umożliwia śledzenie orientacji i położenia w przestrzeni. Nietypowy kształt zakończony owalnym przedłużeniem kontrolerów wynika z konieczności umiejscowienia sensorów światła w miejscu, które nie zostanie przysłonięte dłonią, uniemożliwiając tym samym prawidłowe wyznaczanie pozycji. W czerwcu 2018 roku firma HTC ogłosiła bezprzewodowy adapter **VIVE Wireless** pozwalający na przesyłanie danych pomiędzy komputerem i okularami bez konieczności podłączania kabli.[19] Zapewnia on również zasilanie od 2 do 3 godzin. Komunikacja ma się odbywać za pomocą technologii **Wireless Gigabit Alliance** (**WiGig**) działającej w paśmie 60 GHz z prędkością przesyłu danych do 7 Gb/s. Jednocześnie będą mogły być obsługiwane 3 adaptery w tym samym pomieszczeniu, umożliwiając tym samym pracę w trybie wieloosobowym.

Przykładem systemu wykorzystującego metodę śledzenia optycznego z zewnątrz (*outside-in*) jest **Oculus Rift**. Jego historia sięga roku 2012, kiedy założyciel firmy – Palmer Luckey – uruchomił kampanię na Kickstarterze[20] (portalu crowdfundingowym pozwalającym na zebranie środków finansowych na zrealizowanie projektów). Jej założeniem było zebranie funduszy na opracowanie i wyprodukowanie około stu okularów VR, które będą bardziej immersyjne oraz intuicyjne dla przeciętnego użytkownika od istniejących w tamtym czasie na rynku. Projekt cieszył się dużym zainteresowaniem i ostatecznie zebrano około 2,5 miliona dolarów na

[17] *SteamVR Tracking*: `https://partner.steamgames.com/vrlicensing` (dostęp 05.09.2018)
[18] Road to VR: *Valve Opens Lighthouse VR Tracking API to Third-parties*, 04.08.2016, `https://www.roadtovr.com/valve-third-party-lighthouse-api-steamvr-tracking-htc-vive-dev-kit-license-royalty/` (dostęp 05.09.2018)
[19] *VIVE Wireless Adapter*: `https://www.vive.com/us/wireless-adapter/` (dostęp 05.09.2018)
[20] Kickstarter: *Oculus Rift: Step Into the Game by Oculus*, `https://www.kickstarter.com/projects/1523379957/oculus-rift-step-into-the-game/` (dostęp 05.09.2018)

produkcję ponad 7500 okularów w wersji deweloperskiej (DK1).[21] W marcu 2014 roku firma została wykupiona przez Facebooka, a cztery miesiące później zaczęto produkcję ulepszonej edycji deweloperskiej DK2. Pierwsza wersja konsumencka okularów (oznaczana CV1) została wprowadzona na rynek pod koniec marca 2016 roku. Ich wygląd jest widoczny na ilustracji 2.8.

Źródło: *Heavy Joy Stick* [22]

Ilustr. 2.8: Oculus Rift z kontrolerami Touch i kamerami Constellation

Podobnie jak HTC Vive, okulary wyświetlają obraz pochodzący z podłączonego do nich komputera. Do określania orientacji wykorzystują one moduł IMU, natomiast pozycja w przestrzeni jest wyznaczana za pomocą technologii **Constellation**. Składają się na nią diody LED świecące w podczerwieni (IR) znajdujące się na obiektach do śledzenia oraz kamery IR monitorującej pozycję każdej z nich. Źródła światła nie są stale zapalone, lecz migają określonym wzorem, wysyłając swój unikalny identyfikator. Dzięki temu, znając rozkład poszczególnych diod na obiek-

[21] Oculus: *Update on Developer Kit Technology, Shipping Details*, `https://www.oculus.com/blog/update-on-developer-kit-technology-shipping-details/` (dostęp 05.09.2018)

[22] Heavy Joy Stick: *VirtualLink USB–C Alternate Mode Will Connect VR To PC*, 18.07.2018, `https://heavyjoystick.com/2018/07/18/virtuallink-usb-c-alternate-mode-will-connect-vr-to-pc/` (dostęp 05.09.2018)

cie, system jest w stanie wyznaczyć jego orientację oraz pozycję w pomieszczeniu. Częstotliwość odświeżania wynosi około 60 Hz, natomiast zwiększenie płynności i precyzji jest możliwe dzięki wykorzystaniu modułu IMU także do określania pozycji. Okulary Rift posiadają kilkadziesiąt LEDów rozmieszczonych z każdej strony, co pozwala na śledzenie pod dowolnym kątem obrotu. Interakcja z symulacją może być dokonywana za pomocą dołączonych do zestawu dwóch kontrolerów **Oculus Touch** widocznych na ilustracji 2.8. Posiadają one po dwa przyciski, dwa triggery – jeden pod palcem wskazującym, drugi pod środkowym oraz drążek analogowy. Ponadto każdy przycisk i drążek jest w stanie za pomocą czujników pojemnościowych określić czy znajduje się na nim palec. Określanie orientacji i pozycji jest zrealizowane podobnie jak w samych okularach – połączeniem danych z modułu IMU oraz systemu **Constellation**. Diody niezbędne do wykorzystania tego drugiego znajdują się na półksiężycowych przedłużeniach kontrolerów w przedniej części. Stąd czasami można się spotkać z ich nazwą kodową *Half Moon*. Do śledzenia położenia samych okularów wystarcza pojedyncza kamera systemu. Jednak w przypadku kontrolerów niezbędne są co najmniej dwie lub trzy, aby zapobiec błędom wynikającym z przysłaniania się urządzeń nawzajem. System jest w stanie śledzić obiekty na powierzchni całego pomieszczenia do kilku metrów w każdą stronę, podobnie jak HTC Vive.

2.2.3. Inne metody

Inne metody śledzenia położenia obejmują rozwiązania akustyczne. Ich głównym założeniem jest wykorzystanie ultradźwięków generowanych przez nadajniki znajdujące się w pomieszczeniu oraz mikrofony odbierające sygnały na monitorowanych urządzeniach. Najprostszą odmianą jest pomiar mocy odbieranego sygnału. Im dalej śledzony obiekt znajduje się od nadajników, tym odbierany dźwięk jest słabszy. Ten sposób jest mocno podatny na przysłanianie linii bezpośredniej widoczności urządzeń oraz szumy i zakłócenia pochodzące z innych źródeł dźwięku. Ponadto odbiorniki muszą być bardzo czułe i precyzyjnie przez co ta metoda jest rzadko stosowana. Znacznie lepsze rezultaty można osiągnąć przez pomiar czasu od wysłania do odebrania sygnału.[(1)] Przykładem systemu wykorzystującego taki sposób

pozycjonowania jest **InterSense IS-900**.[23] Producent zapewnia dokładność określania położenia na poziomie kilku milimetrów z częstotliwością pomiarów 180 Hz. System może działać zarówno w trybie przewodowym jak i bezprzewodowo (jednak kosztem zmniejszenia dokładności). Bezprzewodowy odbiornik systemu wraz z przykładowymi okularami (CrystalEyes) jest widoczny na ilustracji 2.9. Jedną z wad systemu jest konieczność przeprowadzania czasochłonnej kalibracji przed możliwością przystąpienia do właściwego śledzenia oraz zastosowania dodatkowego mechanizmu zwiększającego dokładność – najczęściej będzie to moduł IMU, tak jak i w przypadku wspomnianej technologii **InterSense**.

Źródło: *Oliver Kreylos* [24]

Ilustr. 2.9: Urządzenie InterSense IS-900 z okularami CrystalEyes

Podsumowując, warto zauważyć, że opisane w tym rozdziale metody śledzenia pozycji nie wspierają mieszanej rzeczywistości przeznaczonej na urządzenia mobilne takie jak na przykład smartfony. W szczególności, dedykowane frameworki typu Google VR SDK lub Samsung Gear VR nie oferują oficjalnej technologii monitorowania pozycji, a rozwiązania zewnętrzne tego typu są w fazie prototypów. Niniejsza praca magisterska jest propozycją takiego systemu przy wykorzystaniu łączności radiowej ultraszerokiego pasma UWB. Następny rozdział omawia szczegóły techniczne tego rozwiązania.

[23] InterSense: *IS-900 MicroTrax Devices*, `https://est-kl.com/images/PDF/InterSense/IS-900_MicroTrax_Datasheet.pdf` (dostęp 05.09.2018)

[24] Oliver Kreylos' Research and Development Homepage: *KeckCAVES*, `http://idav.ucdavis.edu/~okreylos/ResDev/KeckCAVES/LinkPictures.html` (dostęp 05.09.2018)

Rozdział 3

Zastosowane rozwiązania

System zrealizowany w niniejszej pracy składa się z kilku części. Każda z nich ma konkretnie określone zadanie, dzięki czemu współpraca całości pozwala na osiągnięcie celu postawionego we wprowadzeniu. U samej podstawy architektury systemu znajduje się zestaw **DecaWave TREK1000**, którego zadaniem jest określenie pozycji użytkownika za pomocą czterech nadajniko-odbiorników. Dodatkowo każdy z nich posiada odpowiednio zmodyfikowane oprogramowanie, pozwalające na osiągnięcie lepszych rezultatów w zastosowaniach do wirtualnej rzeczywistości. Następnie uzyskane pomiary są przesyłane do komputera, na którym działa aplikacja serwerowa przetwarzająca te dane i umożliwiająca zmianę konfiguracji systemu oraz dostosowanie do konkretnego środowiska, w którym znajduje się użytkownik. W końcowej fazie, pozycja jest przekazywana bezprzewodowo do urządzenia mobilnego, najczęściej smartfona, generującego wirtualny świat, który po zamontowaniu do okularów wirtualnej rzeczywistości jest przedstawiony użytkownikowi. Każda z powyższych części została dokładnie opisana w następnych rozdziałach.

3.1. System DecaWave

Firma DecaWave Ltd umiejscowiona jest w Irlandii i zajmuje się opracowywaniem układów scalonych pozwalających na lokalizację wewnątrz pomieszczeń[(1)].

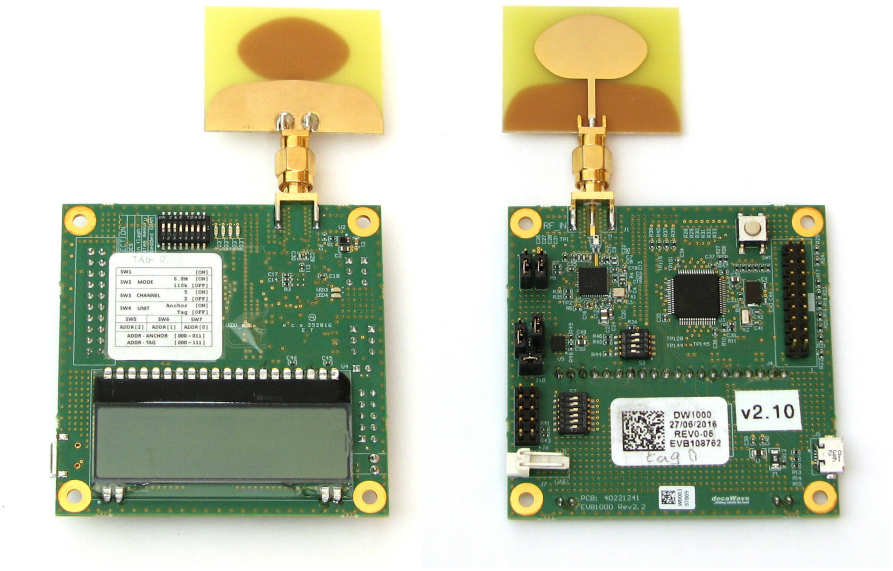
[(1)] About DecaWave: `https://www.decawave.com/about` (dostęp 05.09.2018)

Jednym z wiodących produktów jest **DW1000** – niewielki nadajnikoodbiornik (*ang. transceiver*) działający w standardzie IEEE802.15.4-2011 [6]. Zastosowana technologia pozwala na precyzyjny pomiar czasu z jednoczesnym przesyłaniem danych. Pozwala to na wykorzystanie jej w systemach przeznaczonych do lokalizacji, które jednocześnie wymagają obustronnej komunikacji nadajników. Zapewniana dokładność wyznaczania pozycji to około 10 cm, przy zasięgu do 300 m w sytuacji bezpośredniej widoczności nadajników (*ang. LOS – Line-of-sight*) i do 40 m w przeciwnym przypadku (*ang. NLOS – Non-line-of-sight*). Układ działa w zakresie ultraszerokiego pasma (*ang. UWB – Ultra Wide Band*) na częstotliwości od 3 do 6 GHz i szerokości około 500 MHz. Pozwala to na komunikację za pomocą bardzo krótkich impulsów (rzędu pikosekund), przez co wpływ interferencji jest zmniejszony w porównaniu do sygnałów o węższym zakresie. Maksymalna prędkość przesyłu danych wynosi 6,8 Mb/s.

Wykorzystany w pracy zestaw **TREK1000** składa się z czterech płytek rozwojowych **EVB1000** (*ang. evaluation board*) działających jako nadajnikoodbiorniki i pozwalających na komunikację ze sobą. Szczególnie ważna jest możliwość precyzyjnego pomiaru czasu od wysłania sygnału do otrzymania odpowiedzi. Taka funkcjonalność pozwala na określenie odległości pomiędzy poszczególnymi nadajnikami. Wygląd pojedynczej płytki został przedstawiony na ilustracji 3.1. Znajduje się na niej mikrokontroler **STM32F105RCT6** firmy **STMicroelectronics**, który za pomocą łącza SPI komunikuje się z wcześniej opisanym układem **DW1000** i przetwarza treść odebranych danych. Szczegółowy schemat połączeń elektrycznych zawiera dokumentacja [11]. Ponadto do każdej z płytek jest dołączona antena, której parametry zostały skalibrowane podczas procesu produkcji i zapisane w pamięci trwałej mikrokontrolera [9]. Są to m.in. opóźnienie powodujące wydłużenie mierzonego czasu przelotu sygnału, jak i tłumienie wpływające na maksymalną moc nadawania. Na płytce znajduje się również zestaw przełączników pozwalających na wybór parametrów pracy. Pierwszy z nich, oznaczony `S1`, umożliwia zmianę m.in.:

- prędkości połączenia: 110 kbps lub 6,8 Mbps,
- kanału radiowego: 2 (częstotliwość 4 GHz) lub 5 (częstotliwość 6,5 GHz),
- trybu działania: kotwica lub znacznik,
- adresu (identyfikatora) urządzenia.

Zamontowany na płytce wyświetlacz LCD również pokazuje zmierzone odległości oraz niektóre parametry pracy, takie jak kanał radiowy czy adres urządzenia. Urządzenie może być zasilane zarówno z portu USB, jaki i z zewnętrznego źródła o napięciu od 3,6 V do 5,5 V. Więcej informacji na temat konfiguracji można znaleźć w instrukcji zestawu [7].



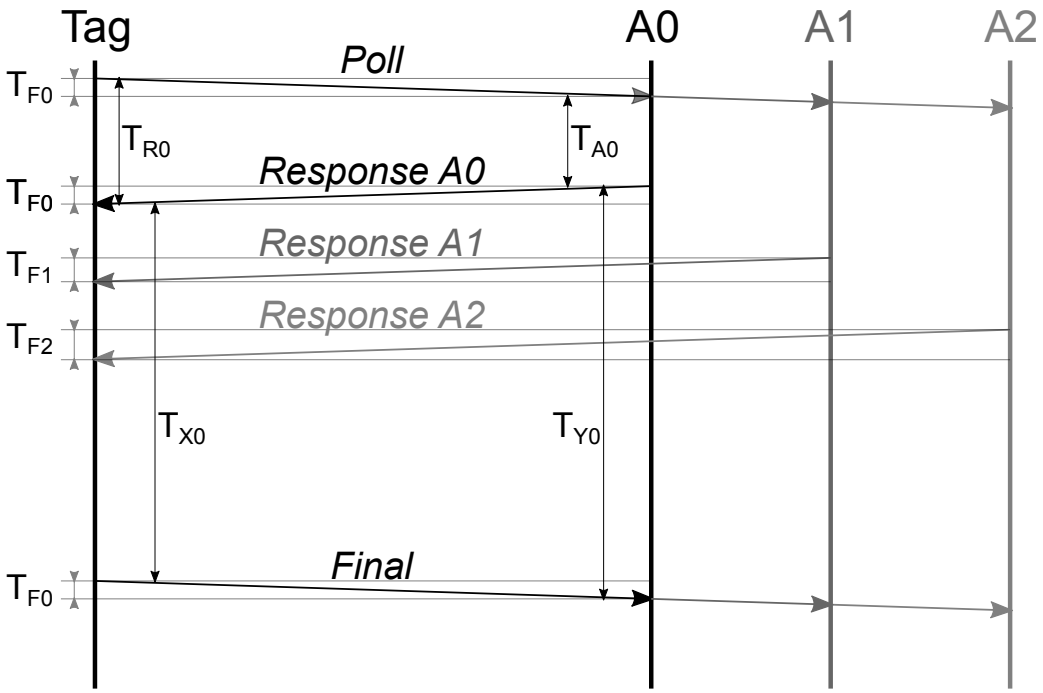
Ilustr. 3.1: Płytka rozwojowa DecaWave EVB1000

Domyślną metodą wyznaczania pozycji w zestawie **TREK1000** jest tzw. *Two Way Ranging* (TWR) [7]. Polega ona na dwukrotnej wymianie danych pomiędzy kotwicą (*ang. anchor*) przymocowaną na stałe i znacznikiem (*ang. tag*) umieszczonym na śledzonym obiekcie. Obydwa urządzenia w precyzyjnie określonym czasie muszą wysłać pakiet danych z odpowiedzią. Umożliwia to obliczenie średniego czasu przelotu sygnału (*ang. ToF – Time of Flight*), a następnie wyznaczenia odległości na podstawie znanej prędkości przesyłu sygnału w powietrzu (w przybliżeniu równej prędkości światła). Takie rozwiązanie pozwala zminimalizować błędy pomiaru wynikające z różnicy częstotliwości zegarów w układach oraz eliminuje konieczność ich ciągłej synchronizacji za pomocą zewnętrznego źródła. W celu wyznaczenia pozycji w przestrzeni trójwymiarowej potrzebne są co najmniej trzy kotwice. Oprogramowanie domyślnie znajdujące się na mikrokontrolerach płytek **EVB1000** wspiera komunikację pomiędzy maksymalnie czterema kotwicami oraz ośmioma

znacznikami. Pojedynczy zestaw zawiera tylko cztery płytki, dlatego w opracowanym systemie trzy z nich działają jako kotwice, a pozostała jest znacznikiem, zamontowanym przy użytkowniku i śledzącym jego położenie.

System wspiera do ośmiu znaczników działających na tej samej częstotliwości radiowej. Sposób komunikacji jest oparty na multipleksowaniu z podziałem czasu (*ang. TDM – Time Division Multiplexing*). Cały cykl wymiany danych między wszystkimi urządzeniami określany jest jako superramka (*ang. superframe*). Została ona podzielona na 10 slotów czasowych – po jednym dla każdego znacznika i dwa dodatkowe na pomiary odległości pomiędzy kotwicami w celu autopozycjonowania (*ang. autopositioning*). Firma DecaWave zoptymalizowała metodę TWR w celu zmniejszenia liczby przesyłanych danych oraz zużycia prądu. Standardowo wymiana komunikatów odbywałaby się pomiędzy każdym z urządzeń osobno. Biorąc pod uwagę, że wszystkie kotwice i znaczniki znajdują się w zasięgu komunikacji radiowej przez cały czas, tj. nie występuje zjawisko ukrytej stacji (*ang. hidden terminal*), można przeprowadzić redukcję przesyłanych żądań. Proces wymiany danych w pojedynczym slocie przedstawia schemat na ilustracji 3.2. Komunikacja rozpoczyna się od wysłania przez znacznik ramki rozgłoszeniowej (*ang. broadcast*) do wszystkich kotwic znajdujących się w zasięgu (*Poll*). Następnie każda z nich odpowiada w ściśle określonym momencie, w kolejności według adresu urządzenia. Ramki z odpowiedzią (*Response*) zawierają odległość od danej kotwicy, przeliczoną po poprzedniej sesji TWR. Oznacza to, że otrzymane pomiary będą zawsze opóźnione o czas trwania superramki, co negatywnie wpływa na efektywność systemu. W końcowej fazie slotu tag wysyła odpowiedź (*Final*), w której zamieszcza znaczniki czasowe otrzymania ramek *Response* od każdej z kotwic.

Wyznaczenie czasu przelotu sygnału (ToF) odbywa się poprzez uwzględnienie opóźnienia podczas wymiany trzech komunikatów: *Poll*, *Response* i *Final*. Na schemacie został on określony jako $T_{\mathrm{F}i}$, gdzie i oznacza adres kotwicy, której dotyczy mierzony czas. Dla zwiększenia czytelności pominięto większość oznaczeń dotyczących innych kotwic niż A_0. Obliczenia zakładają, że prędkość oscylatorów (oznaczonych jako α dla znacznika i β dla kotwicy) znajdujących się w poszczególnych odbiornikach może różnić się między sobą, ale nie zmienia się w trakcie



Ilustr. 3.2: Schemat wymiany danych w pojedynczym slocie

pojedynczego slotu przesyłu danych. Układ równań[(2)] pozwalający na wyznaczenie czasu przelotu do kotwicy A_0 został przedstawiony na (3.1).

$$\begin{cases} \alpha \cdot T_{\mathrm{R0}} = 2 \cdot T_{\mathrm{F0}} + \beta \cdot T_{\mathrm{A0}} \\ \beta \cdot T_{\mathrm{Y0}} = 2 \cdot T_{\mathrm{F0}} + \alpha \cdot T_{\mathrm{X0}} \\ \dfrac{\alpha + \beta}{2} = 1 \end{cases} \tag{3.1}$$

Trzecie równanie wynika z założenia, że średnia arytmetyczna α i β pozwala zniwelować różnicę prędkości oscylatorów poszczególnych urządzeń. Po przekształceniach uzyskiwany jest wzór na czas przelotu sygnału (3.2) (opisany również w dokumentacji [8] i [9]).

$$T_{\mathrm{F0}} = \frac{T_{\mathrm{R0}} \cdot T_{\mathrm{Y0}} - T_{\mathrm{A0}} \cdot T_{\mathrm{X0}}}{T_{\mathrm{R0}} + T_{\mathrm{A0}} + T_{\mathrm{X0}} + T_{\mathrm{Y0}}} \tag{3.2}$$

[(2)] Bitcraze Forums: *DWM1000 DS-TWR method calculate distance method?*, `https://forum.bitcraze.io/viewtopic.php?t=1944#p9959` (dostęp 05.09.2018)

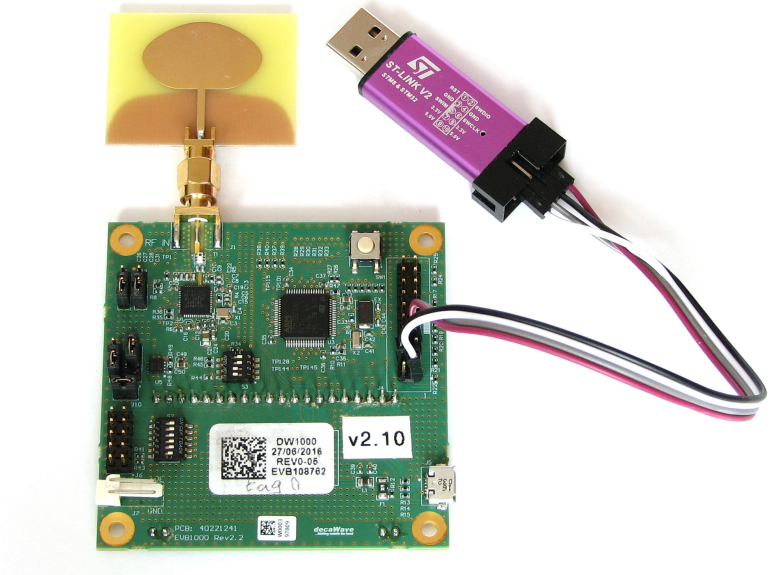
Następnie w celu uzyskania odległości, czas jest mnożony przez prędkość rozchodzenia się światła w powietrzu $c = 299\,702\,547$ m/s oraz uwzględniane są korekty opóźnień związanych z charakterystyką anteny dla wybranej częstotliwości radiowej. Przy prędkości komunikacji ustawionej na poziomie 6,8 Mbps, czas trwania jednego slotu jest równy 10 ms. Uwzględniając 10 slotów w superramce, standardowa częstotliwość odświeżania pozycji wynosi zaledwie 10 Hz. Testowany w tej pracy system posiada tylko jeden znacznik oraz trzy kotwice. Z tego powodu liczba slotów została zredukowana do trzech – jeden dla znacznika i dwa do autopozycjonowania. Zabieg ten umożliwił zwiększenie częstotliwości do około 33 Hz, a opóźnienie zredukowano z poprzednich 100 ms do 30 ms (nowy czas trwania superramki). Przekłada się to na lepszą płynność przy zastosowaniach w mieszanej rzeczywistości. Niezbędny do przeprowadzenia modyfikacji kod źródłowy w języku **C** oprogramowania układowego znajdującego się na płytkach został dostarczony przez producenta. Do edycji oraz kompilacji kodu wykorzystano środowisko **Ac6 System Workbench for STM32** dostępne na platformie **OpenSTM32 Community**[(3)]. Kompilacja odbywa się za pomocą narzędzi **GNU ARM Toolchain** na architekturę `arm-none-eabi`. Płytki **EVB1000** posiadają złącze JTAG oraz wyjścia SWD (*ang. Serial Wire Debug*) kompatybilne z programatorami **ST-Link** i oznaczone symbolem `J4`. W pracy magisterskiej został wykorzystany drugi sposób, przy zastosowaniu programatora widocznego na ilustracji 3.3. Rozkład zastosowanych połączeń przedstawia tabela 3.1.

Pin	Oznaczenie	Opis
1	VDD3V3	Zasilanie płytki 3,3 V
4	GND	Masa zasilania
7	JTMS-SWDAT	Linia danych SWD (SWDIO)
9	JTCK-SWCLK	Linia zegara SWD (SWCLK)

Tabela 3.1: Rozkład połączeń do programowania płytki EVB1000

Każda z płytek **EVB1000** posiada port **USB**, który umożliwia komunikację z systemem. Oprogramowanie układowe korzysta z wirtualnego portu zgodnego

(3) OpenSTM32 Community: *System Workbench for STM32*, `http://www.openstm32.org/System+Workbench+for+STM32` (dostęp 05.09.2018)



Ilustr. 3.3: Programator ST-Link/V2 do programowania układów EVB1000

z UART, którego sterownik jest dostarczony przez producenta mikrokontrolera – firmę **STMicroelectronics**. Ze względu na sposób działania zestawu, połączenie się ze znacznikiem umożliwi uzyskanie pomiarów odległości tylko między nim a kotwicami. W celu otrzymania danych ze wszystkich urządzeń konieczne jest przyłączenie się do głównej kotwicy o adresie `0x00`.

Po nawiązaniu połączenia, układ zaczyna wysyłać wyniki pomiarów odległości, dokonywanych pomiędzy urządzeniami systemu **TREK1000**, skonfigurowanych z tymi samymi parametrami radiowymi. Format pojedynczego pakietu danych wygląda następująco [9][10]:

```
MID MASK RANGE0 RANGE1 RANGE2 RANGE3 NRANGES RSEQ DEBUG aT:A
```

Każdy z nich jest zakończony znakiem nowej linii `CRLF`. Przykładowy pakiet (dla pomiaru odległości pomiędzy znacznikiem a kotwicami) został przedstawiony poniżej:

```
mc 0f 00000663 000005a3 00000512 000004cb 095f c1 00024c24 a0:0
```

Składa się on głównie z wartości zapisanych w systemie szesnastkowym w formie tekstowej i zawiera następujące informacje:

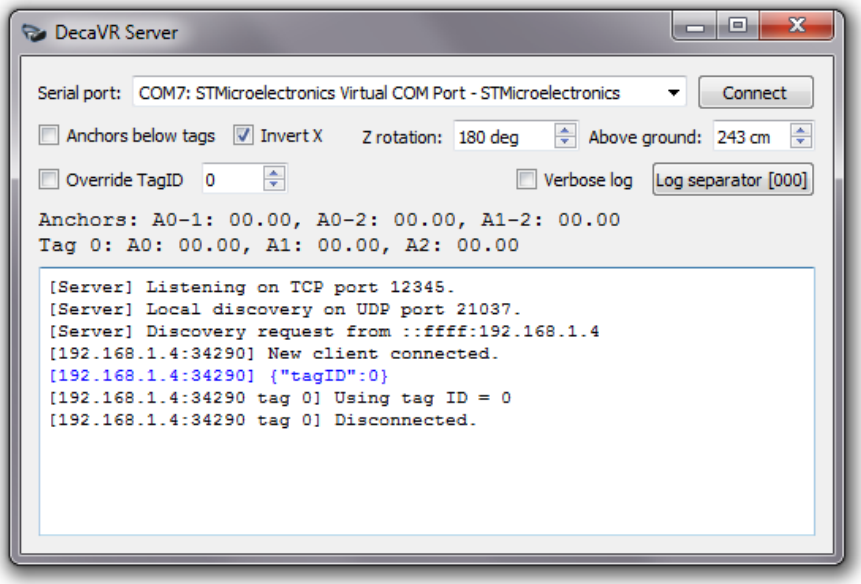
1) MID – identyfikator pakietu, może przyjąć jedną z poniższych wartości:
    - mr – nieskorygowane pomiary pomiędzy znacznikiem a kotwicami;
    - mc – jak „mr”, z uwzględnieniem korekt i opóźnień anten;
    - ma – pomiar pomiędzy kotwicami, tzw. autopozycjonowanie (po uwzględnieniu korekt).
2) MASK – maska bitowa określająca, które pomiary są wiarygodne spośród RANGE0 do RANGE3;
3) RANGE0..3 – dla pomiaru znacznik-kotwica (mr i mc) jest to odległość (w mm) od znacznika do kolejnych kotwic z adresami od A0 do A3; dla autopozycjonowania (ma):
    - RANGE0 – wartość ignorowana;
    - RANGE1 – odległość kotwicy A0 do kotwicy A1;
    - RANGE2 – odległość kotwicy A0 do kotwicy A2;
    - RANGE3 – odległość kotwicy A1 do kotwicy A2.
4) NRANGES – liczba wiarygodnych pomiarów dokonanych do tej pory;
5) RSEQ – kolejny numer sekwencyjny pakietu dla danego MID;
6) DEBUG – dla pakietu „ma” jest to zmierzone opóźnienie anteny; w pozostałych przypadkach – czas otrzymania ostatniego pomiaru;
7) aT:A – adres (identyfikator) znacznika (T) oraz adres kotwicy (A), których dotyczy pomiar

3.2. Aplikacja serwerowa

W celu odebrania pomiarów z systemu DecaWave TREK1000, a następnie przesłania ich do aplikacji mobilnej, została przygotowana aplikacja serwerowa. Do jej zbudowania wykorzystany został język C++ oraz wieloplatformowy framework Qt. Dzięki temu istnieje możliwość uruchomienia jej na najpopularniejszych systemach operacyjnych, a zastosowanie interfejsu graficznego sprawia, że jest intuicyjna. Pozwala to na jej obsługę przez użytkownika nieposiadającego szczegółowej wiedzy

na temat działania systemu. Do wykonywania przekształceń w układzie współrzędnych zastosowano bibliotekę Eigen[4].

Wygląd aplikacji przedstawia ilustracja 3.4. Na górze okna znajduje się lista dostępnych portów szeregowych, z których należy wybrać podłączoną płytkę **EVB1000**. Nazwa urządzenia powinna zawierać frazę „`STMicroelectronics Virtual COM Port`". W przypadku systemu Windows, konieczna jest wcześniejsza instalacja sterownika[5]. Po wciśnięciu przycisku „Connect" otwierany jest port szeregowy i rozpoczyna się komunikacja.



Ilustr. 3.4: Wygląd aplikacji serwerowej

Łączność z urządzeniem jest realizowana przez klasę dostępną w środowisku Qt – `QSerialPort`. Format otrzymywanych danych ma postać przedstawioną w rozdziale 3.1 na stronie 29. Na początku zbierane są pomiary autopozycjonowania między kotwicami, otrzymywane z częstotliwością 33 Hz. Dzięki temu użytkownik systemu nie musi ręcznie wprowadzać pozycji modułów. Następnie dane są wygładzane filtrem średniej kroczącej o długości okna 50. W ten sposób, niwelowane są niedokładności oraz zakłócenia spowodowane przemieszczaniem się osób w zasięgu systemu, a odczyt położenia jest bardziej stabilny. Pozycja kotwic w układzie

[4] Eigen: `http://eigen.tuxfamily.org/index.php?title=Main_Page` (dostęp 05.09.2018)
[5] STMicroelectronics: *STSW-STM32102 - STM32 Virtual COM Port Driver*, `http://www.st.com/en/development-tools/stsw-stm32102.html` (dostęp 05.09.2018)

współrzędnych XY zostaje wyznaczana za pomocą trzech równań okręgów. Celem uproszczenia obliczeń, wstępnie założono, że wszystkie kotwice znajdują się na tej samej wysokości, wprowadzanej przez użytkownika aplikacji w polu „Above ground". Ponadto przyjęto, że kotwica A_0 znajduje się w środku układu współrzędnych, natomiast A_1 na dodatniej osi X. Jeśli umiejscowienie odbiorników wymagałoby odwrócenia osi X, należy zaznaczyć w programie opcję „Invert X" lub zamienić ze sobą A_0 z A_1. Z równań okręgów można wyznaczyć pozycję kotwicy $A_2(x, y)$ – (wzór 3.3). Odległość od kotwicy A_i do A_j została oznaczona jako R_{ij}.

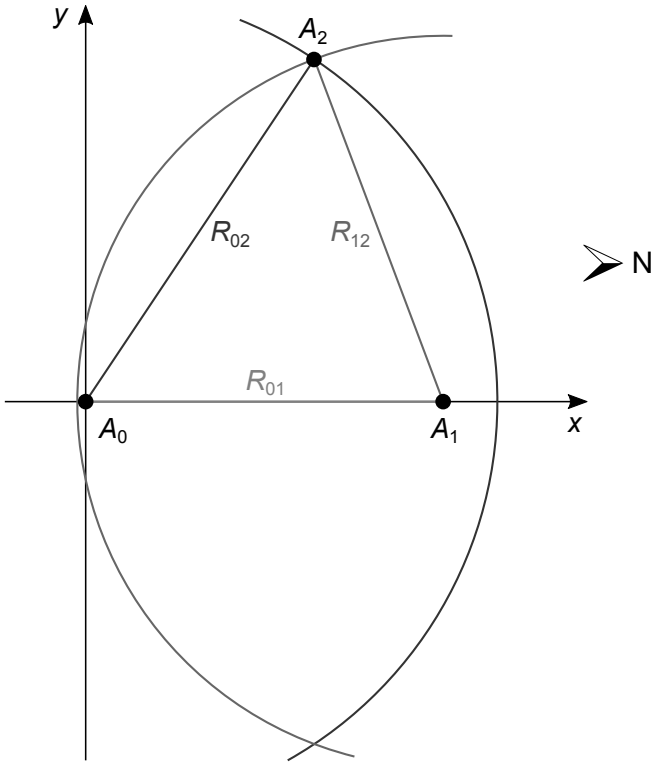
$$\begin{cases} A_0 : x^2 + y^2 = R_{02}^2 \\ A_1 : (x - R_{01})^2 + y^2 = R_{12}^2 \end{cases} \tag{3.3}$$

Po przekształceniach otrzymuje się wzory na wyznaczenie współrzędnych trzeciej kotwicy (3.4). Układ z naniesionymi odległościami i przecięciami okręgów przedstawia ilustracja 3.5.

$$\begin{cases} x = \dfrac{R_{01}^2 + R_{02}^2 - R_{12}^2}{2R_{01}} \\ y = \pm\sqrt{R_{02}^2 - x^2} \end{cases} \tag{3.4}$$

Z możliwych dwóch rozwiązań wybierane jest to o dodatniej wartości współrzędnej y. Skutkuje to tym, że punkt znajduje się w pierwszej lub drugiej ćwiartce układu współrzędnych. To pozwala na prawidłowe przełożenie pozycji do świata wirtualnego. W dalszej części programu, przetwarzane są pomiary między znacznikiem a kotwicami. Odległości zostają przeliczone na współrzędne w wirtualnym świecie za pomocą trilateracji. Przyjmuje się, że kotwice znajdują się na tej samej wysokości, co upraszcza początkowe równania bez utraty dokładności (3.5). Ewentualną korektę wyniku w przypadku różnych wysokości, można dokonać poprzez obrót układu współrzędnych.

$$\begin{cases} A_0 : x^2 + y^2 + z^2 = R_0^2 \\ A_1 : (x - A_{1x})^2 + y^2 + z^2 = R_1^2 \\ A_2 : (x - A_{2x})^2 + (y - A_{2y})^2 + z^2 = R_2^2 \end{cases} \tag{3.5}$$

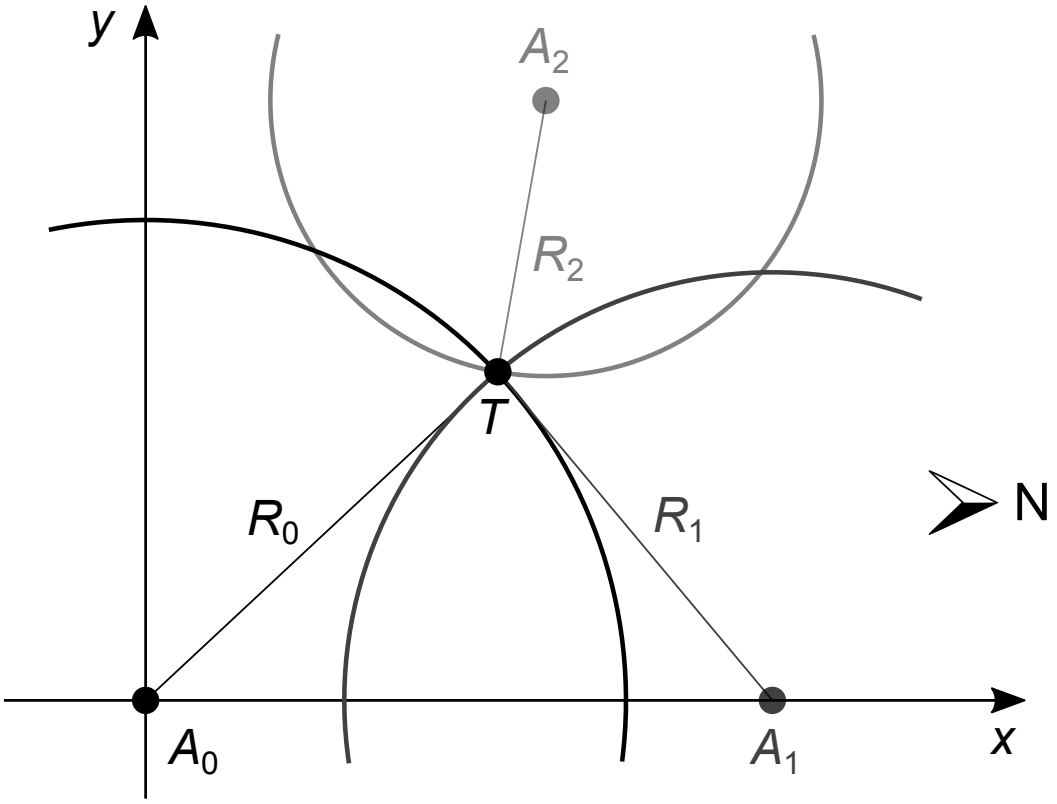


Ilustr. 3.5: Geometria w autopozycjonowaniu kotwic

Dokonując przekształceń układu równań (3.5), uzyskiwane są poszukiwane współrzędne znacznika $T(x, y, z)$ – (wzór 3.6). Graficzna reprezentacja rozwiązania w płaszczyźnie wyznaczonej przez kotwice została przedstawiona na ilustracji 3.6.

$$\begin{cases} x = \dfrac{A_{1x}^2 + R_0^2 - R_1^2}{2A_{1x}} \\ y = \dfrac{A_{2x}^2 + A_{2y}^2 + R_0^2 - R_2^2}{2A_{2y}} - \dfrac{A_{2x}}{A_{2y}}x \\ z = \pm\sqrt{R_0^2 - x^2 - y^2} \end{cases} \tag{3.6}$$

Wybór rozwiązania współrzędnej z zależy od ustawienia „Anchors below tags” w aplikacji serwerowej. Po zaznaczeniu tej opcji wybierane jest dodatnie rozwiązanie reprezentujące pozycję znacznika powyżej kotwic, a ujemne w przeciwnym wypadku. Następnie układ współrzędnych jest przekształcany w taki sposób, że środek ciężkości trójkąta wyznaczonego przez kotwice, jest umieszczony w początku układu. Dzięki temu, centralne części pomieszczenia wirtualnego i rzeczywistego w przybliżeniu znajdują się w tym samym miejscu. Odległość kotwic od ziemi po-



Ilustr. 3.6: Graficzne przedstawienie trilateracji znacznika w płaszczyźnie kotwic

winna zostać wprowadzona w polu „Above ground” i na jej podstawie dokonywane jest przesunięcie obliczonej pozycji znacznika, a tym samym wysokości w wirtualnym świecie. Następnie wykonywany jest obrót układu, aby zapewnić zgodność ze wskazaniami kompasu w telefonie. Przyjęto, że wektor $\overrightarrow{A_0A_1}$, leżący na osi X powinien być skierowany na północ. W przypadku występowania różnicy, wartość kąta w stopniach należy wprowadzić w polu „Z rotation”. Przykładowo podanie wartości „90” oznacza, że wektor $\overrightarrow{A_0A_1}$ kieruje się na wschód.

Aktualne pomiary są wyświetlane w środkowym panelu aplikacji. Są to zarówno odległości autopozycjonowania między kotwicami wraz z ich przeliczonymi współrzędnymi, jak i pozycja znacznika o identyfikatorze `0x00` w układzie. Istnieje możliwość zapisu wszystkich zbieranych danych do logu w postaci pliku tekstowego o nazwie „`log.txt`”, poprzez zaznaczenie „Verbose log”. Przycisk znajdujący się obok „Log separator [000]” pozwala na rozdzielanie zbieranych danych, co ułatwia ich późniejszą analizę. Przykładowy log z widocznym separatorem przedstawia listing 3.1.

```
17:05:59.743:[Status] Anchor coords: A1:2.76,0.00, A2:1.29,-4.05
17:05:59.744:[Status] Anchors: 01: 2.80, 02: 4.28, 12: 4.30 [70]
17:05:59.758:[Status] Tag 0: A0: 2.79, A1: 2.90, A2: 2.86 [157]
17:05:59.758:[Status] Location:  0.08,  0.43,  0.70
17:05:59.776:[Status] Anchor coords: A1:2.76,0.00, A2:1.29,-4.05
17:05:59.776:[Status] Anchors: 01: 2.77, 02: 4.25, 12: 4.34 [71]
17:05:59.786:[Status] Tag 0: A0: 2.78, A1: 2.89, A2: 2.88 [158]
17:05:59.786:[Status] Location:  0.08,  0.41,  0.69
17:06:00.955:[MAIN] -------- LOG SEPARATOR 001 --------
17:06:58.122:[Status] Anchor coords: A1:2.76,0.00, A2:1.29,-4.05
17:06:58.122:[Status] Anchors: 01: 2.75, 02: 4.23, 12: 4.31 [224]
17:06:58.135:[Status] Tag 0: A0: 2.85, A1: 2.95, A2: 2.89 [55]
17:06:58.136:[Status] Location:  0.08,  0.44,  0.62
17:06:58.151:[Status] Anchor coords: A1:2.76,0.00, A2:1.29,-4.05
17:06:58.152:[Status] Anchors: 01: 2.74, 02: 4.28, 12: 4.31 [225]
17:06:58.166:[Status] Tag 0: A0: 2.82, A1: 2.94, A2: 2.87 [56]
17:06:58.166:[Status] Location:  0.10,  0.45,  0.65
```

Listing 3.1: Szczegółowy log z separatorem

Do przesyłania lokalizacji z aplikacji serwerowej do mobilnej zastosowano komunikację bezprzewodową przez lokalną sieć Wi-Fi. Łączność ta może zostać zrealizowana dzięki przyłączeniu komputera i smartfonu do wspólnego punktu dostępowego w tej samej podsieci. Zaraz po uruchomieniu serwera następuje włączenie nasłuchiwania na ustalonych portach poprzez protokoły UDP oraz TCP. Pierwszy z nich pozwala na wykrycie adresu serwera poprzez wysłanie datagramu rozgłoszeniowego (*ang. broadcast*) do wszystkich urządzeń w podsieci na porcie 21037 o treści „`DecaVR Discovery`”. Serwer po otrzymaniu takiej wiadomości odsyła tym samym sposobem komunikat „`DecaVR Server:12345`”, gdzie `12345` jest numerem portu TCP, na którym nasłuchuje połączeń przychodzących. To pozwala klientowi na uzyskanie parametrów serwera i nawiązanie z nim komunikacji. Numer portu może się różnić, w zależności od dostępności oraz sposobu numeracji dokonywanej przez system operacyjny. Sieć powinna być skonfigurowana w taki sposób, żeby umożliwiać przesyłanie ramek rozgłoszeniowych, co umożliwi prawidłowe wykrycie serwera.

Właściwa komunikacja odbywa się przy wykorzystaniu bardziej niezawodnego protokołu TCP. Do obsługi połączenia w bibliotece Qt, zastosowano klasę `QTcpSocket`. W celu zminimalizowania opóźnień ustawiono opcję `TCP_NODELAY` wyłączającą algorytm Nagle'a poprzez ustawienie trybu gniazda `QAbstractSocket::`

`LowDelayOption`. Dzięki temu małe porcje danych nie są buforowane, tylko wysyłane natychmiast bez względu na rozmiar. Komunikaty pomiędzy urządzeniami są przesyłane w formie tekstowej, jako obiekty JSON zakończone znakiem nowej linii `CRLF`. To pozwala na łatwy odczyt przesyłanych danych zarówno przez człowieka, jak i programowo. Większość technologii posiada narzędzia umożliwiające analizę oraz kodowanie w tym formacie. Do nich należy również framework Qt, posiadający klasy `QJsonDocument` i `QJsonObject`. Po nawiązaniu połączenia przez klienta, serwer oczekuje na identyfikator znacznika, którego pozycja ma być obliczana i przesyłana. Jest to ustalane w parametrze „`tagID`" typu całkowitoliczbowego (*ang. integer*). Na przykład po otrzymaniu poniższego żądania, serwer zacznie odsyłać lokalizację znacznika o adresie `0x00`:

```
{"tagID": 0}
```

W zależności od wskazanego identyfikatora, serwer może połączyć się z fizycznym urządzeniem w sposób opisany na początku tego rozdziału (od strony 31) lub przesłać dane wygenerowane wcześniej. Na potrzeby tej pracy, w celu ułatwienia testowania systemu zaprogramowano symulator pozycji, przemieszczający się między ustalonymi punktami (klasa `SimulatedLocationProvider`). Jest on dostępny pod `tagID` o wartości `9`. Dodatkowo istnieje możliwość odtworzenia wcześniej zebranych pozycji, korzystając z logów zapisanych w pliku tekstowym. Aplikacja będzie szukać w bieżącym katalogu, logu z podanym identyfikatorem w nazwie. Na przykład dla pliku „`log42.txt`" należy podać `tagID` wynoszący `42`. Ta funkcjonalność jest realizowana przez klasę `LoggedLocationProvider` i działa dla identyfikatorów większych lub równych `10`. Wszystkie pozostałe wartości od `0` do `7` są bezpośrednio związane z fizycznym adresem znacznika i umożliwiają odczyt ich pomiarów przez port szeregowy. Obsługą tych identyfikatorów zajmuje się klasa `SerialLocationProvider`.

W przypadku nieprawidłowego formatu pakietu otrzymanego przez aplikację serwerową, odsyłane są informacje o błędzie, również w formacie JSON. Schemat wiadomości wygląda następująco:

```
{"error": "kod błędu", "message": "szczegółowy opis błędu"}
```

Wartość „message” opisuje błąd w sposób zrozumiały dla użytkownika. Natomiast parametr „error” zawiera jedną z wartości tekstowych przedstawionych w liście poniżej i jest przeznaczony do interpretacji maszynowej:

- MALFORMED_MESSAGE – wiadomość w nieprawidłowym formacie JSON lub parametr „tagID” nie jest typu *integer*;
- MISSING_PARAMETER – brak parametru „tagID” w wiadomości;
- UNKNOWN_TAG – żądany identyfikator znacznika nie istnieje.

Jeżeli odebrany pakiet zostanie prawidłowo przetworzony, serwer rozpoczyna wysyłanie pozycji dla wybranego znacznika (fizycznego bądź wirtualnego). Lista parametrów znajdujących się w każdym pakiecie została zebrana w tabeli 3.2.

Parametr	Typ	Opis
timestamp	*integer*	znacznik czasowy wyznaczenia pozycji
x	*integer*	współrzędna x w cm
y	*integer*	współrzędna y w cm
z	*integer*	współrzędna z w cm

Tabela 3.2: Parametry pakietu z pozycjami od serwera

Wartość „timestamp” służy do rozróżnienia kolejnych pomiarów i może być czasem w milisekundach od rozpoczęcia pomiarów. Po każdym zapisie danych do gniazda sieciowego, wymuszane jest ich wysłanie poprzez wykonanie metody flush(). Dzięki temu niwelowane jest opóźnienie związane z wewnętrznym buforowaniem danych przez bibliotekę Qt. Przesyłane dane w sieci pomiędzy urządzeniami są również zapisywane w logu tekstowym, aby umożliwić znajdowanie ewentualnych błędów komunikacji. Odebrany identyfikator znacznika tagID może zostać zastąpiony po stronie serwera przez zaznaczenie pola „Override TagID” i ustawieniu żądanej wartości w polu liczbowym obok. Ta funkcjonalność pozwala na testowanie wcześniej zebranych pomiarów bez konieczności wprowadzania zmian w aplikacji po stronie klienta.

3.3. Aplikacja mobilna

Aplikacja mobilna została przygotowana na system operacyjny Android, przy wykorzystaniu środowiska Unity w wersji 2017. Dodatkowo zastosowany został framework Google VR SDK (znany wcześniej pod nazwą Google Cardboard)[6], w celu generowania obrazu stereoskopowego. Pozwala to na oglądanie wyświetlanej sceny w trzech wymiarach – z możliwością postrzegania głębi. Ekran jest podzielony na dwie połowy, co powoduje, że po umieszczeniu telefonu w okularach wirtualnej rzeczywistości (VR), każde oko otrzymuje osobny obraz. Ponadto nakładany jest specjalny filtr niwelujący zniekształcenia soczewek na obrzeżach pola widzenia.

Na potrzeby tej pracy magisterskiej zostały wykorzystane okulary wirtualnej rzeczywistości **VR BOX 2.0**, widoczne na ilustracji 3.7. W zestawie z nimi znajduje się również bezprzewodowy pilot Bluetooth, przedstawiony na ilustracji 3.8. Jego funkcjonalność została użyta do interakcji z przygotowaną aplikacją mobilną.

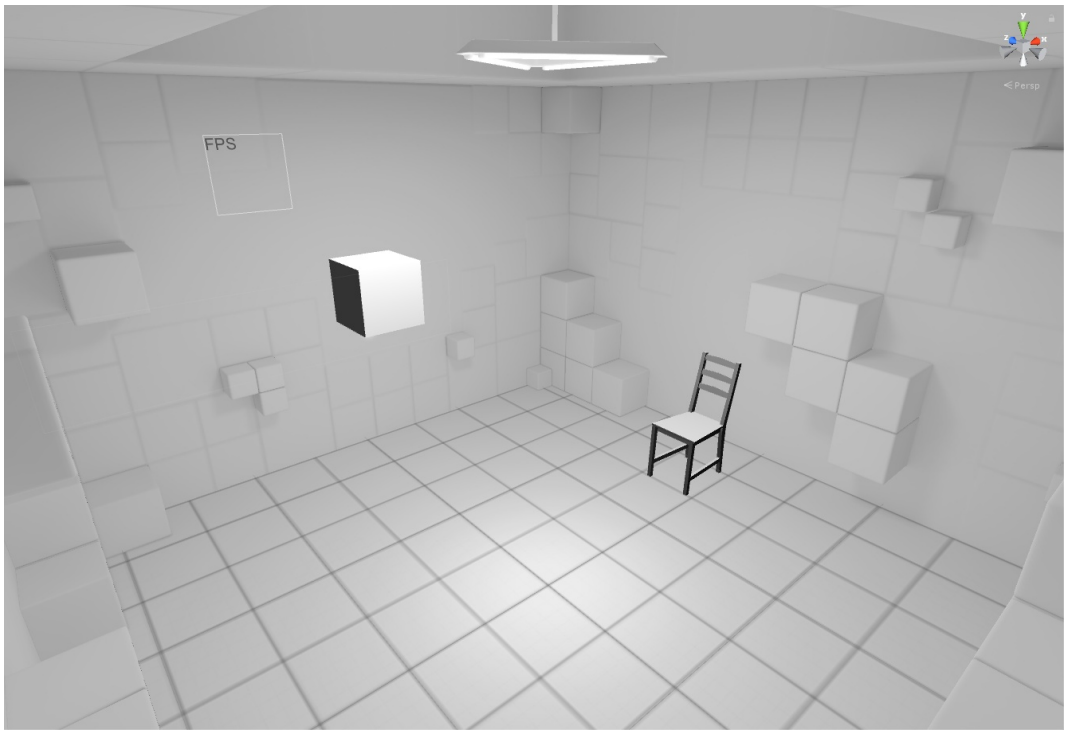
Ilustr. 3.7: Okulary VR BOX 2.0 wykorzystane w pracy

[6] Google VR for everyone: `https://developers.google.com/vr/` (dostęp 05.09.2018)

Ilustr. 3.8: Pilot Bluetooth z zestawu VR BOX 2.0

Scena w aplikacji bazuje na przykładach dostarczonych wraz z Google VR SDK. Składa się z wirtualnego pomieszczenia o wymiarach 5 × 5 × 2,5 m. Na podłodze została umieszczona kwadratowa siatka o boku 1 m, która ułatwia orientację w przestrzeni podczas przemieszczania się. Dodatkowym elementem interakcyjnym jest sześcian, który emituje dźwięk. Jego pozycja może zostać zmieniona poprzez nakierowanie wskaźnika i wciśnięcie bocznego przycisku na pilocie zdalnego sterowania. Na potrzeby projektowania i testowania systemu umieszczono pole tekstowe wyświetlające płynność symulacji w postaci liczby klatek na sekundę (FPS), orientację z czujnika telefonu oraz pozycję uzyskaną z systemu TREK1000. Wygląd sceny z lotu ptaka w edytorze przedstawia ilustracja 3.9.

Framework Google VR SDK domyślnie nie wspiera urządzeń, które nie posiadają wbudowanego żyroskopu. Z tego powodu został przygotowany dodatek (wtyczka) odczytujący orientację za pomocą sensora obrotu *ang. rotation vector sensor*, dostępnego w urządzeniach działających na systemie Android. Najczęściej jego wskazania opierają się na połączeniu odczytu akcelerometru, kompasu i – jeśli jest sprzętowa możliwość – również żyroskopu [14]. Dodatek ma postać biblioteki kompilowanej do archiwum systemu Android (*ang. AAR – Android Archive*) [16][17]. Składa się z klasy `OrientationProvider` implementującej interfejs `SensorEvent-`

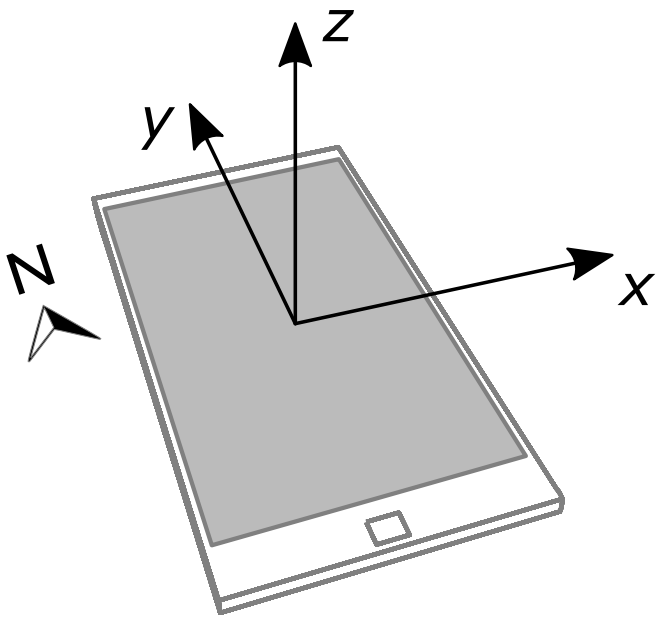


Ilustr. 3.9: Wygląd sceny w edytorze Unity

`Listener`, pozwalający na odczyt wskazań czujników. Wybrany rodzaj sensora to `TYPE_ROTATION_VECTOR` z okresem odświeżania `SENSOR_DELAY_GAME`, która wynosi co najmniej 20 ms, co przekłada się na częstotliwość 50 Hz i powinno być wystarczające do zastosowań w wirtualnej rzeczywistości. Pomiary z czujników są uzyskiwane w układzie współrzędnych zależnym od domyślnej orientacji urządzenia [15]. Dla smartfonów jest to najczęściej pozycja pionowa, zatem rozkład osi wygląda jak na ilustracji 3.10. Wskazania sensora obrotu są uzyskiwane w postaci kwaternionu jednostkowego reprezentującego obrót od położenia pierwotnego, które jest zdefiniowane następująco [14]:

- oś x styczna do płaszczyzny ziemi i kieruje się na wschód (E);
- oś y styczna do płaszczyzny ziemi i kieruje się na północ (N);
- oś z prostopadła do ziemi i kieruje się w stronę nieba.

Pierwsze cztery parametry otrzymywane z czujnika S zostały ukazane na równaniach (3.7), gdzie φ jest kątem obrotu, a x, y i z wyznaczają wektor, wokół którego jest wykonywany.



Ilustr. 3.10: Układ współrzędnych sensorów smartfona w systemie Android

$$\begin{cases} S_0 = S_x = x \cdot \sin\left(\frac{\varphi}{2}\right) \\ S_1 = S_y = y \cdot \sin\left(\frac{\varphi}{2}\right) \\ S_2 = S_z = z \cdot \sin\left(\frac{\varphi}{2}\right) \\ S_3 = S_w = \cos\left(\frac{\varphi}{2}\right) \end{cases} \tag{3.7}$$

W niektórych urządzeniach dostępny jest dodatkowy parametr określający błąd wyznaczenia położenia wynikający z rozkalibrowania wskazań kompasu. Zwiększenie dokładności można uzyskać poprzez obracanie telefonem w różnych osiach lub wykonanie kilku ruchów na kształt cyfry 8.[7] Odczyt pomiarów z wtyczki oraz obliczenia odbywają się w skrypcie `CameraRotation`. Tworzenie instancji klasy oraz jej obsługa odbywa się za pomocą mechanizmu refleksji dostępnego w środowisku Unity. Układ współrzędnych sensorów w systemie Android jest prawoskrętny, natomiast w Unity lewoskrętny. W związku z tym, konieczne jest przekształcenie otrzymanego kwaternionu S, zamieniając miejscami jego współrzędne w sposób

[7] Google Maps Help: *Find and improve your location's accuracy – Android*, `https://support.google.com/maps/answer/2839911?co=GENIE.Platform%3DAndroid&` (dostęp 05.09.2018)

przedstawiony w układzie równań (3.8), otrzymując kwaternion $Q(x, y, z, w)$.

$$\begin{cases} Q_x = -S_y \\ Q_y = S_x \\ Q_z = -S_z \\ Q_w = -S_w \end{cases} \tag{3.8}$$

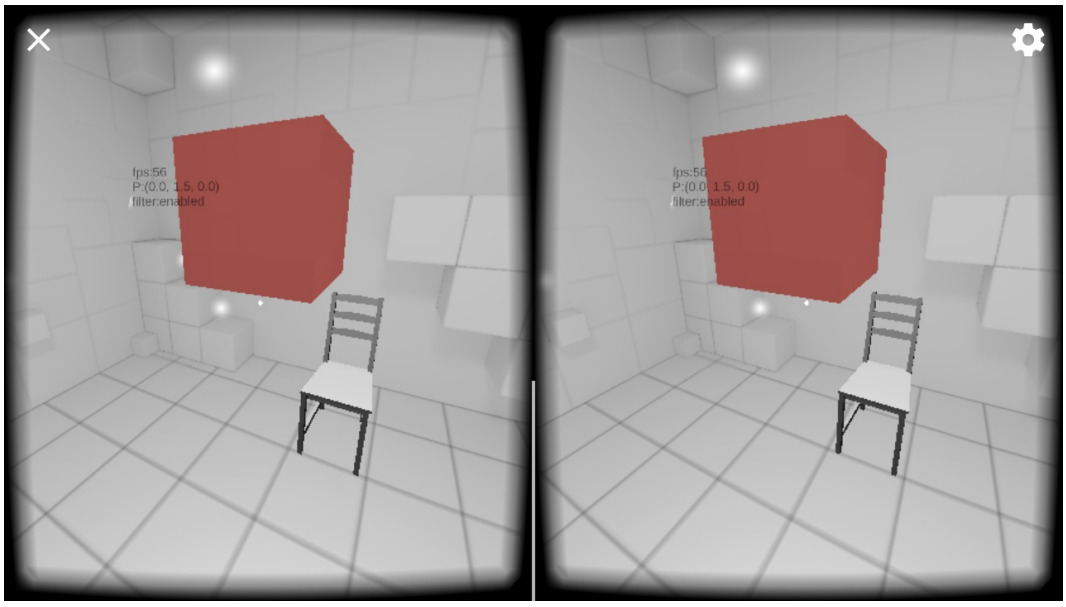
W celu uzyskania jak najlepszych wrażeń podczas użytkowania opracowanej aplikacji, została wykonana kalibracja parametrów okularów **VR BOX 2.0**. Wstępne wartości uzyskano z gotowych konfiguracji dostępnych w sieci.[8] Następnie korzystając z narzędzia online Google Cardboard Viewer Profile Generator[9] dostosowano rozpiętość soczewek, współczynniki zniekształceń oraz kąty widzenia do posiadanego modelu. W ten sposób uzyskuje się kod QR widoczny na ilustracji 3.11, który można zeskanować w aplikacji konfiguracyjnej Google Cardboard lub wygenerować plik `current_device_params`, a następnie umieścić w pamięci telefonu w katalogu `/sdcard/Cardboard/`. Wygląd aplikacji na telefonie przedstawia ilustracja 3.12.

Ilustr. 3.11: Kod QR z parametrami konfiguracyjnymi okularów VR BOX 2.0

Łączność z aplikacją serwerową opisaną w rozdziale 3.2 została zrealizowana w skrypcie `NetworkClient`. Do komunikacji wykorzystano klasy `UdpClient` oraz `TcpClient` dostępne w środowisku .NET i pozwalające na obsługę gniazd siecio-

[8] Hypergrid Business: *VR Headset QR Codes*, `https://www.hypergridbusiness.com/faq/vr-headset-qr-codes/` (dostęp 05.09.2018)

[9] Google: *Cardboard viewer profile generator*, `https://wwgc.firebaseapp.com/` (dostęp 05.09.2018)



Ilustr. 3.12: Zrzut ekranu z aplikacji mobilnej na telefonie

wych w wygodny sposób. Operacje na obiektach w formacie JSON odbywają się przy użyciu modułu `JsonUtility` oferowanego przez Unity. Po odebraniu pozycji, jej współrzędne są wygładzane za pomocą filtra alfa-beta (α-β). Jest to uproszczona pochodna filtra Kalmana, znacznie mniej wymagająca obliczeniowo i dająca zadowalające efekty podczas filtrowania obiektów poruszających się z niewielkimi przyspieszeniami [18]. Jego nazwa nawiązuje do parametrów α oraz β określających w jakim stopniu kolejny pomiar wpływa na odfiltrowaną wartość. Wzory wykorzystane w filtrze przedstawia układ równań (3.9). Funkcja $m(t)$ reprezentuje pomiar wykonany w danym momencie czasu; ΔT określa czas, który upłynął od ostatniego pomiaru; $x(t)$ jest przewidywaną wartością – lokalizacją śledzonego obiektu; a $v(t)$ jej pochodną – w tym przypadku prędkością.

$$\begin{cases} p(t) = x(t-1) + v(t-1) \cdot \Delta T \\ r = m(t) - p(t) \\ x(t) = p(t) + \alpha \cdot r \\ v(t) = v(t-1) + \dfrac{\beta \cdot r}{\Delta T} \end{cases} \tag{3.9}$$

Współczynniki filtra zostały dobrane w sposób eksperymentalny i wyniosły od-

powiednio $\alpha = 0{,}065$ oraz $\beta = 0{,}002$. Zwiększanie pierwszego z nich poprawia reakcję na szybkie zmiany pozycji, ale jednocześnie osłabia odporność filtra na szumy. Zbyt duża wartość parametru β wprowadza niepożądane oscylacje wokół wartości średniej.

Znacznik w postaci płytki EVB1000 został przymocowany do okularów VR w taki sposób, żeby przez cały czas znajdował się w bezpośredniej widoczności trzech kotwic przymocowanych do sufitu. W praktyce oznacza to, że jest umieszczony na głowie użytkownika. Przyjęto, że różnica wysokości od anteny do oczu wynosi około 20 cm i taką korektę pozycji kamery zastosowano w aplikacji mobilnej. Zasilanie urządzenia odbywa się poprzez kabel USB podłączony do baterii umieszczonej w dolnej części gogli. Zdjęcie przedstawiające użytkownika wraz z okularami i znacznikiem przedstawia ilustracja 3.13.

Ilustr. 3.13: Użytkownik z okularami oraz znacznikiem EVB1000

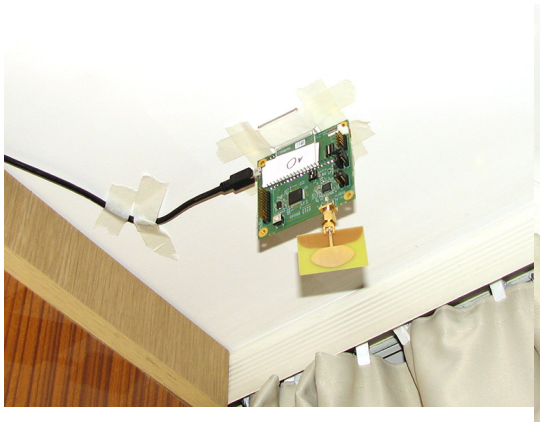
Rozdział 4

Testy działania

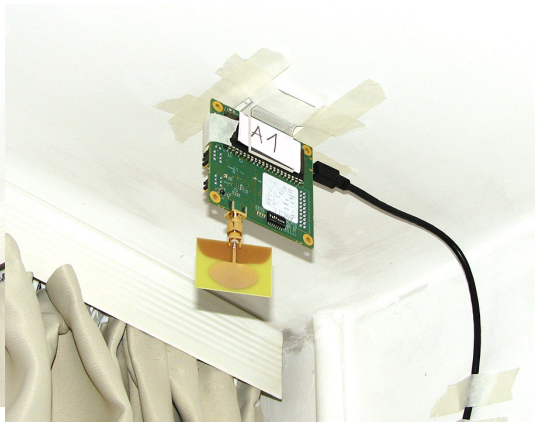
W celu określenia dokładności zrealizowanego systemu, wykonano liczne pomiary, które pozwalają stwierdzić czy nadaje się on do założonego na wstępie zastosowania. Testy podzielono na statyczne, dające pojęcie o odchyleniach od dokładnej wartości gdy układ znajduje się w jednym miejscu; dynamiczne – mówiące o tym jak system radzi sobie ze śledzeniem przemieszczającego się obiektu oraz subiektywne, będące oceną jakości działania określoną przez użytkownika. Kotwice systemu przymocowano do sufitu na wysokości $(253{,}0 \pm 0{,}5)$ cm oraz w odległości co najmniej 15 cm od najbliższej ściany lub przeszkody, zgodnie z zaleceniami producenta [7]. Ustawienie zostało zaprojektowane w taki sposób, żeby uzyskać rozkład w postaci trójkąta równoramiennego. Anteny kotwic skierowano w stronę środka pomieszczenia. Ich pozycje są widoczne na ilustracji 4.2 w postaci kwadratowych symboli, natomiast zdjęcie każdej z nich po zamocowaniu przedstawia ilustracja 4.1. Do wykonywania obliczeń i rysowania wykresów wykorzystano język programowania Python z bibliotekami NumPy oraz Matplotlib. Zasilanie kotwicy głównej A_0 zostało zrealizowane poprzez podłączenie jej do komputera Lenovo ThinkPad X201, który jednocześnie komunikował się poprzez złącze szeregowe z całym systemem i odczytywał pomiary. Kotwica A_1 została podłączona do zasilacza/ładowarki USB firmy EnerGenie EG-UC2A-01[(1)] o maksymalnym prądzie wyjściowym 2,1 A.

[(1)] EnerGenie: *Universal USB charger, 2.1 A*, `https://energenie.com/item.aspx?id=8618` (dostęp 05.09.2018)

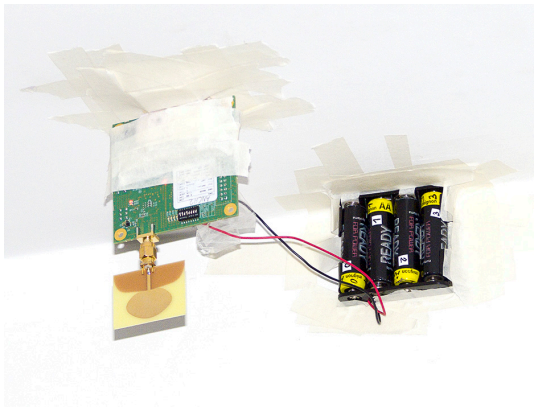
Natomiast kotwica A_2 była zasilana z czterech akumulatorków Ni-MH Hama Ready For Power, o rozmiarze AA (HR03), napięciu znamionowym 1,2 V i pojemności 2100 mAh każda, połączonych szeregowo.



(a) Kotwica A_0



(b) Kotwica A_1

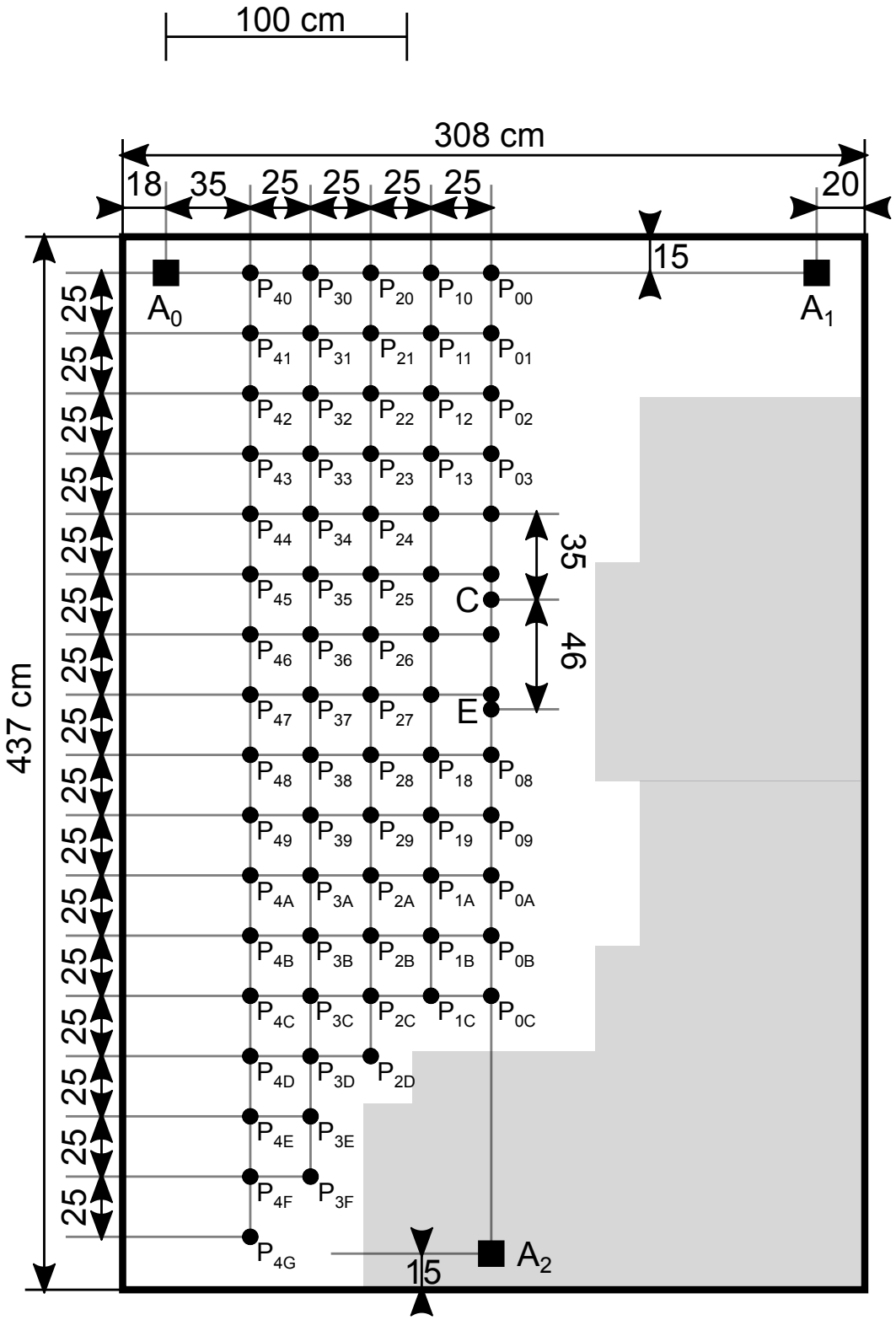


(c) Kotwica A_2

Ilustr. 4.1: Kotwice zamontowane do testów

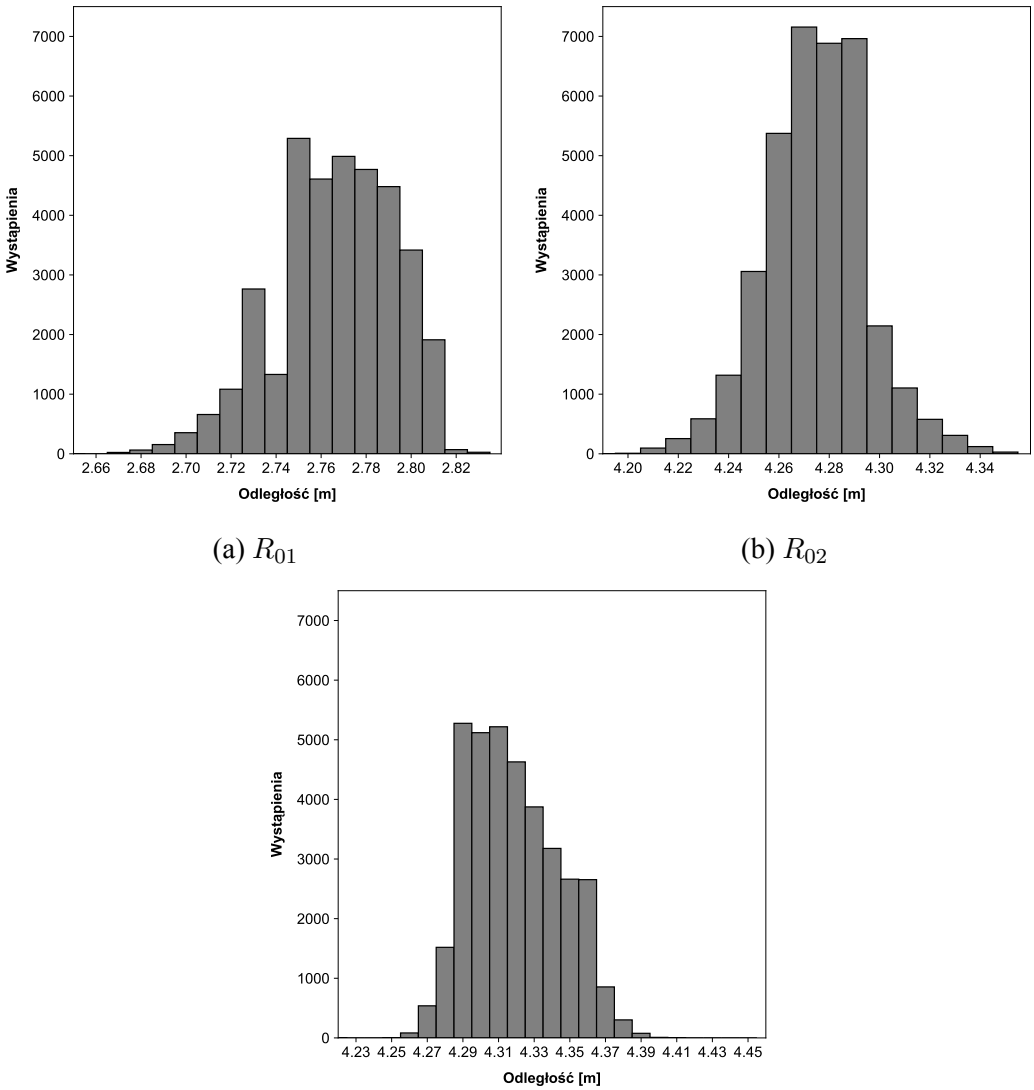
4.1. Autopozycjonowanie

W pierwszym etapie testów sprawdzono poprawność automatycznego wyznaczania współrzędnych kotwic. Wykorzystano w tym celu funkcjonalność polegającą na komunikacji między stacjonarnymi modułami systemu DecaWave, opisaną w rozdziale 3.2. Zbadano stabilność wyznaczanych współrzędnych oraz różnice od wartości oczekiwanych. Wykresy widoczne na ilustracji 4.3 zawierają histogramy 36 tys. pomiarów odległości wykonanych pomiędzy kotwicami w trakcie testów. Wartości średnie oraz odchylenie standardowe σ z uzyskanych próbek przedstawio-



Ilustr. 4.2: Rozkład kotwic i miejsc pomiarów statycznych w pomieszczeniu (wymiary w cm)

no w tabeli 4.1. Zamieszczono również wartości oczekiwane, które zostały zmierzone ręcznie.



(a) R_{01} (b) R_{02}

(c) R_{12}

Ilustr. 4.3: Histogramy odległości w autopozycjonowaniu

Następnie korzystając ze wzorów (3.3) i (3.4) wyznaczono podstawowe współrzędne kotwic (przed przekształceniami) i porównano z oczekiwaniami. Wartości te zebrano w tabeli 4.2.

Odległość [m]	Oczekiwana	Średnia	Różnica	σ
R_{01}	2,700	2,767	0,067	0,026
R_{02}	4,288	4,275	-0,013	0,020
R_{12}	4,288	4,319	0,031	0,025

Tabela 4.1: Statystyka odległości autopozycjonowania

Współrzędne [m]	Oczekiwane	Średnie	Różnica
A_0	(0,000; 0,000)	(0,000; 0,000)	(0,000; 0,000)
A_1	(0,000; 2,700)	(0,000; 2,767)	(0,000; 0,067)
A_2	(1,350; 4,070)	(1,315; 4,068)	(-0,035; -0,002)

Tabela 4.2: Wyznaczone współrzędne autopozycjonowania

4.2. Pomiary statyczne

Do wykonania testów statycznych rozmieszczono kilkadziesiąt punktów w pokoju testowym, odległych od siebie o 25 cm. Ich rozkład, pozycje nadajników systemu wraz z wymiarami pomieszczenia są widoczne na ilustracji 4.2. Nadajnikoodbiornik systemu TREK1000 pracujący w trybie znacznika przymocowano do statywu wraz z powerbankiem o pojemości 2200 mAh, zasilającego poprzez złącze USB, co jest widoczne na ilustracji 4.4. Dodatkowo zostały zaznaczone dwa punkty szczególne C i E. Pierwszy z nich jest środkiem ciężkości trójkąta $A_0A_1A_2$ wyznaczonego przez kotwice, który jednocześnie służy za początek układu współrzędnych wirtualnej sceny (jest to opisane w rozdziale 3.2). Wybrano go do testów z tego względu, że jest to miejsce, w którego okolicy użytkownik będzie prawdopodobnie przebywał najczęściej. Natomiast punkt E znajduje się w takiej samej odległości od wszystkich kotwic: (226 ± 1) cm w płaszczyźnie poziomej, zatem jest środkiem okręgu opisanego na tym trójkącie.

Przeprowadzone pomiary pozwolą na porównanie dokładności każdej z kotwic oraz określenia zależności ustawienia anteny znacznika od wskazań urządzeń. Jego położenie wyznaczono za pomocą wzoru (4.1) przy założeniu, że kotwica A_0 umieszczona jest w początku układu współrzędnych. Szarym kolorem zacieniowano przeszkody uniemożliwiające stabilne umiejscowienie statywu w tych miejscach, a tym samym wykonanie pomiarów z wymaganą dokładnością. Punkty od P_{00} do P_{0C} zostały umieszczone w linii o równej odległości od kotwic A_0 i A_1, natomiast

wszystkie kolejne od P_{10} do P_{4G} bliżej miejsca zawieszenia A_0.

$$\begin{cases} D = 2(A_{1x}A_{2y} - A_{1y}A_{2x}) \\ E_x = \dfrac{1}{D}\left[A_{2y}(A_{1x}^2 + A_{1y}^2) - A_{1y}(A_{2x}^2 + A_{2y}^2)\right] \\ E_y = \dfrac{1}{D}\left[A_{1x}(A_{2x}^2 + A_{2y}^2) - A_{2x}(A_{1x}^2 + A_{1y}^2)\right] \end{cases} \tag{4.1}$$

Ilustr. 4.4: Statyw ze znacznikiem podczas pomiarów statycznych

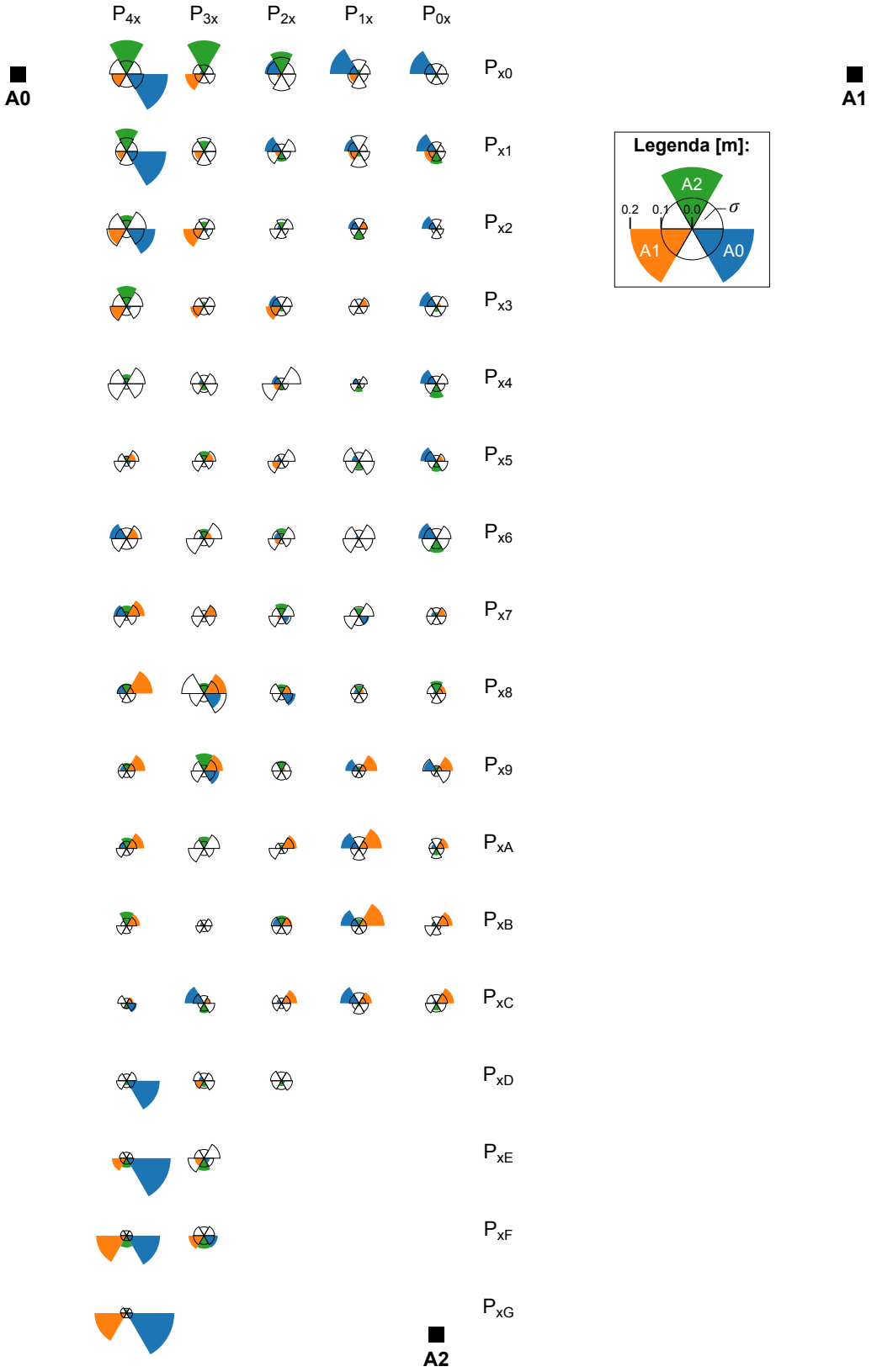
Za punkt odniesienia pomiarów pozycji przyjęto miejsce połączenia anteny z układem EVB1000. Wysokość tego punktu dla znacznika znajdującego się na statywie wyniosła (155,0 ± 0,5) cm, co przekłada się na (88,0 ± 0,5) cm od płaszczyzny wyznaczonej przez anteny kotwic przy suficie. Dokładną pozycję ustawienia statywu wskazywał zawieszony na nim ciężarek za pomocą nierozciągliwej nici.

Ustawienie w jednym z punktów jest widoczne na ilustracji 4.5.

Ilustr. 4.5: Ciężarek wskazujący dokładną pozycję umieszczenia statywu

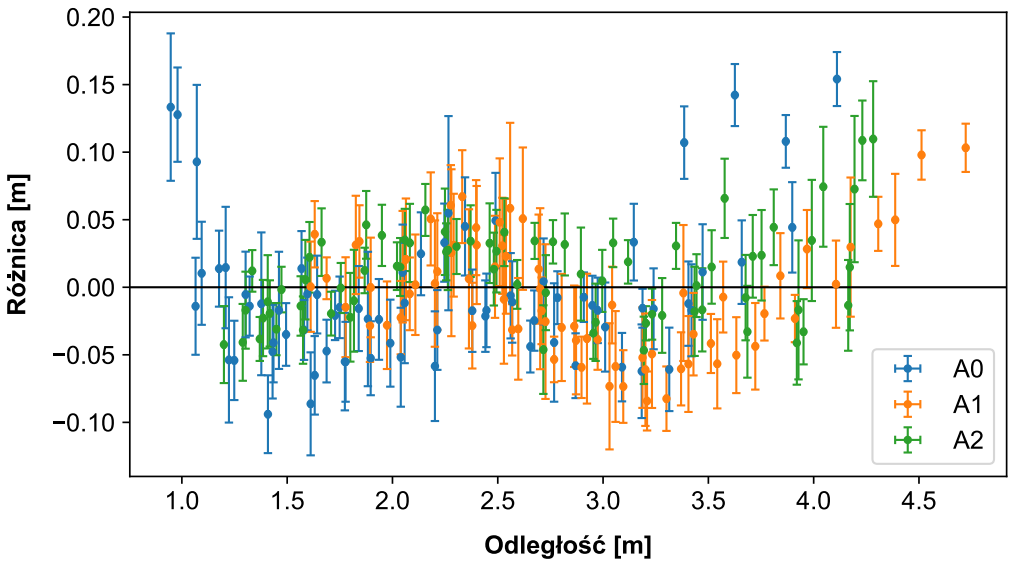
W każdym z punktów zebrano po 200 pomiarów odległości od kotwic do znacznika dla dwóch ustawień anteny – równolegle do dłuższej (|) i krótszej (–) ściany pomieszczenia. Było to wymagane przez różnice w odczycie systemu w zależności od orientacji płaszczyzn anten. Następnie wyznaczono wartości średnie $\overline{x}$ oraz odchylenie standardowe σ i porównano wyniki z wielkościami oczekiwanymi m. Dane zostały zebrane w postaci mapy widocznej na ilustracji 4.6. W każdym z punktów został naniesiony wykres radialny, podzielony na 6 części. Część wypełniona kolorem przedstawia przesunięcie odczytu odległości pomiędzy znacznikiem i poszczególną kotwicą, od wartości oczekiwanej w danym punkcie. Natomiast czarnym obramowaniem określono odchylenia standardowe σ pomiarów.

W punktach znacznie odbiegających od wartości oczekiwanych dla pewności powtórzono pomiary. Były to głównie miejsca przy ścianach oraz w okolicy kotwic. Uzyskano podobne wyniki, redukując tym samym błędy mogące wynikać z nieprawidłowo przeprowadzonego badania, na przykład przez niedokładne ustawienie statywu. Sprawdzono również czy zauważalna jest zmiana dokładności określania pozycji w funkcji odległości znacznika od kotwicy. Wykres widoczny na ilustracji 4.7 przedstawia różnice od odległości oczekiwanych, wraz z odchyleniem standar-



Ilustr. 4.6: Mapa przedstawiająca dokładność odczytów w punktach statycznych

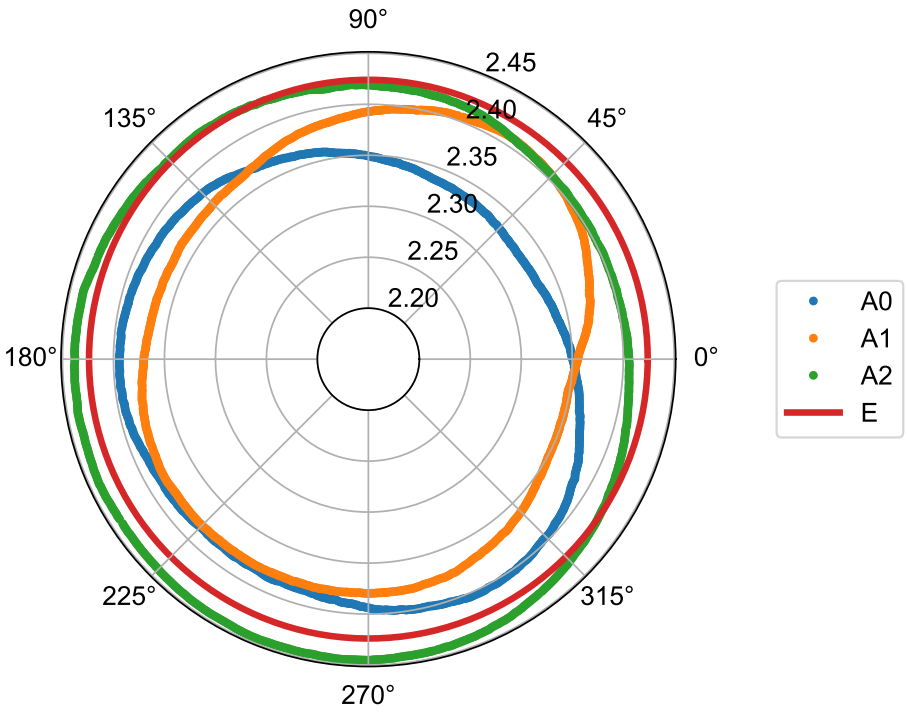
dowym, zebrane podczas wykonywania pomiarów w punktach statycznych.



Ilustr. 4.7: Dokładność wyznaczania pozycji w funkcji odległości od kotwicy

Orientacja anteny wokół osi pionowej miała zauważalny wpływ na pomiar odległości między urządzeniami. W celu dokładniejszego zbadania tej zależności, w punkcie E zostały zebrane próbki podczas wykonywania ręcznego obrotu znacznika na statywie. Następnie, po wstępnym wygładzeniu pomiarów, umieszczono je na wykresie radialnym przedstawiającym zależność od kąta, na ilustracji 4.8. Dane zostały znormalizowane w taki sposób, żeby równoległe ustawienie anten znacznika i kotwicy znajdowało się w punkcie o kącie 0°. Okrąg określający oczekiwaną odległość od punktu E zaznaczono na czerwono.

W tym samym punkcie przeprowadzony został test sprawdzający jaki wpływ na uzyskane pomiary ma przysłonięcie linii bezpośredniej widoczności nadajników (LOS) przez człowieka. Osoba, w odległości około jednego metra, krążyła powoli wokół statywu z zamontowanym znacznikiem. Na ilustracji 4.9 widoczny jest wykres przedstawiający uzyskiwane pomiary odległości od wszystkich trzech kotwic, w trakcie wykonywania tego testu. Dla lepszej czytelności wyniki zostały przefiltrowane średnią ruchomą o szerokości 30 próbek, co przekłada się na jedną sekundę zbierania danych. Zauważalne są odchylenia $\boldsymbol{d}_{\mathbf{avg}}$ od wartości średniej (wynoszącej 2,390 m) w momentach zbliżania się oraz zasłaniania kolejnych nadajników.

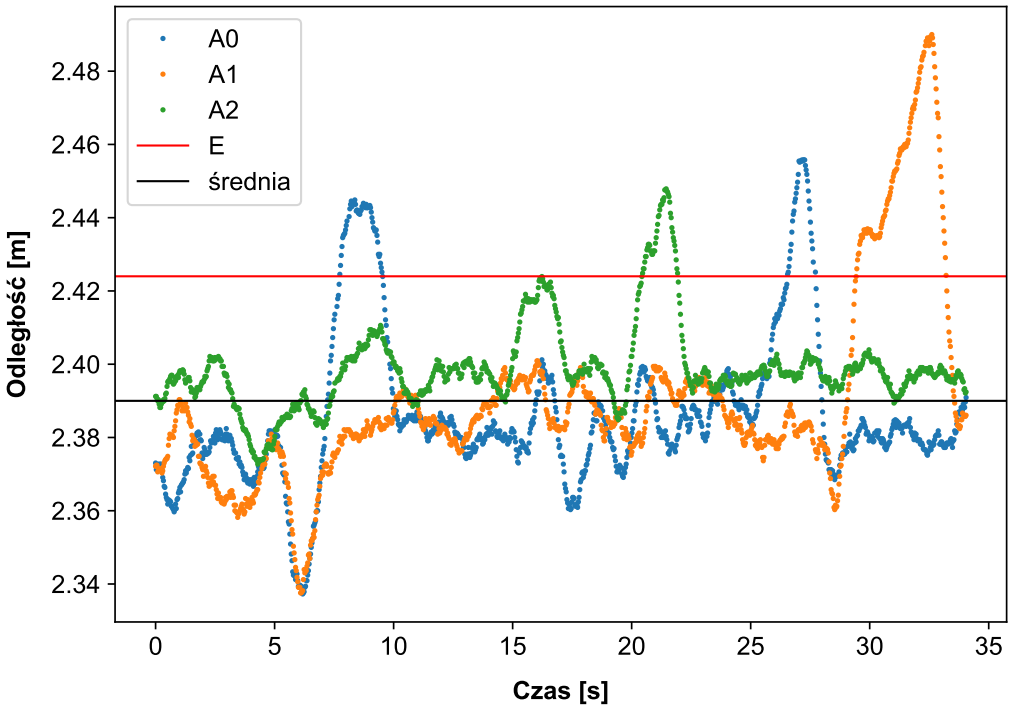


Ilustr. 4.8: Wskazania odległości w punkcie E w zależności od orientacji anteny

Ekstremalne wskazania x zostały zebrane w tabeli 4.3. Uwzględniono także różnice od teoretycznej odległości między punktem E i kotwicami wynoszącej 2,424 m. Jak widać średnie wskazania systemu były mniejsze od tej wartości, jednak należy wspomnieć, że ważniejsze jest uzyskiwanie powtarzalnych odczytów w tych samych punktach, niż ich dokładna wielkość.

t [s]	A_i	x [m]	d_{avg} [m]	d_{E} [m]
6,2	A_0, A_1	2,337	-0,053	-0,087
8,4	A_0	2,445	0,055	0,021
16,2	A_2	2,423	0,033	-0,001
17,5	A_0	2,360	-0,030	-0,064
21,5	A_2	2,448	0,058	0,024
27,1	A_0	2,455	0,065	0,031
28,6	A_1	2,360	-0,030	-0,064
32,6	A_1	2,490	0,100	0,066

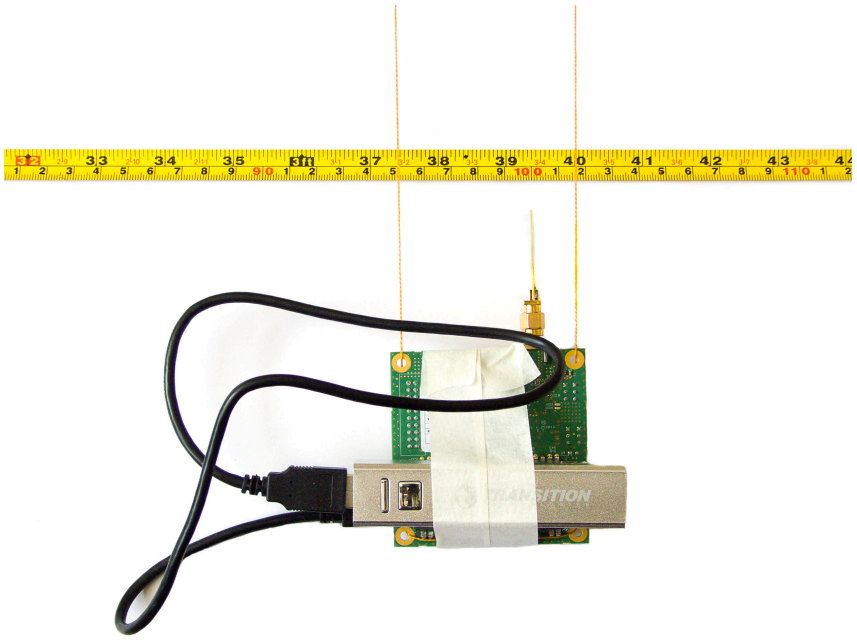
Tabela 4.3: Odstępstwa odczytu odległości po zasłonięciu nadajników



Ilustr. 4.9: Wpływ zasłonięcia nadajników przez człowieka na odczyt odległości

4.3. Pomiary dynamiczne

W celu wyznaczenia dokładności badanego systemu podczas ruchu, wykonano test na wahadle. Znacznik wraz z baterią zasilającą został przymocowany za pomocą długiej nierozciągliwej nici do sufitu pomieszczenia testowego. Pomiar wychylenia przeprowadzono wykorzystując miarę rozciągniętą równolegle do kierunku ruchów wahadła. Na ilustracji 4.10 widoczny jest moment spoczynku, w którym antena znajduje się w odległości 100 cm od początku miarki. W celu ułatwienia wykonywania odczytów, wykorzystano kamerę skierowaną na poruszający się znacznik. Wahadło poruszało się wzdłuż osi Y, natomiast ruch w osi X uznano za pomijalny. Długość nici wynosiła $l = (137{,}0 \pm 0{,}5)$ cm, a współrzędne punktu zawieszenia wahadła określono na $Q(1,26; 1,80; 0,4)$. Wzory opisujące ruch oscylacyjny przedstawiono na (4.2). Przyjęto, że wahadło poruszało się drganiami harmonicznymi o amplitudzie A, częstotliwości oscylacji f, ze współczynnikiem tłumienia wykładniczego α oraz przesunięciem względem rozpoczęcia drgań t_0.

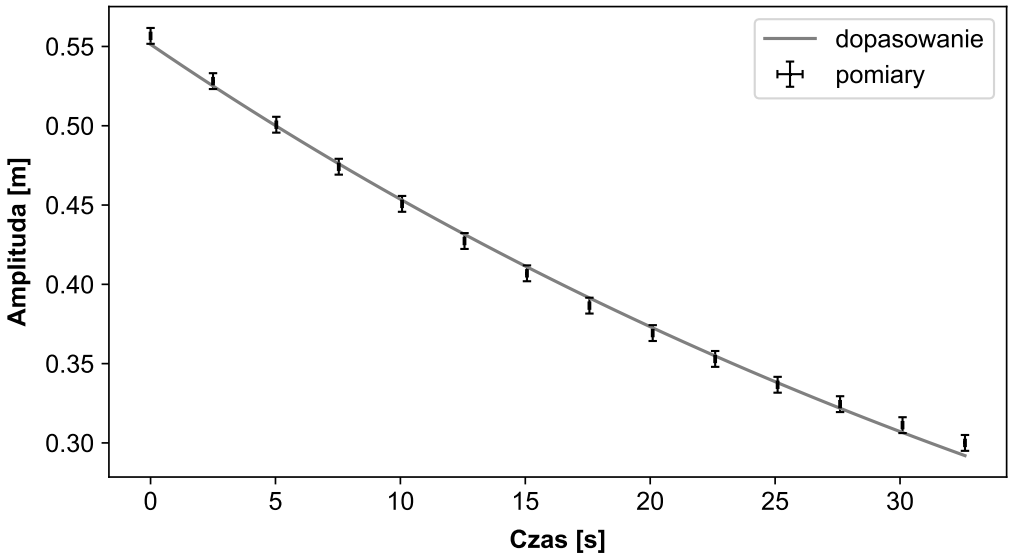


Ilustr. 4.10: Punkt spoczynku wahadła z widoczną miarką

$$\begin{cases} w(t) = A\mathrm{e}^{-\alpha t}\cos[2\pi f(t+t_0)] \\ x(t) = Q_x \\ y(t) = Q_y + w(t) \\ z(t) = Q_z + \sqrt{l^2 - w(t)^2} \end{cases} \tag{4.2}$$

Wykonano 8 pomiarów, w tym 5 z anteną obróconą prostopadle ($\perp$) do kierunku ruchu i 3 równolegle ($\parallel$). W pierwszym etapie wyznaczono współczynnik tłumienia α poprzez ustalenie maksymalnych wychyleń w kolejnych oscylacjach. Odczyt położenia odbywał się wykorzystując nagranie z kamery rejestrującej oscylacje. Za pomocą metod dostępnych w bibliotece NumPy dostosowano parametry funkcji wykładniczej do zebranych wcześniej punktów. Optymalizacja została przeprowadzona przy użyciu domyślnego algorytmu Levenberga-Marquardta. Wykres przedstawiający dopasowanie tłumienia w jednym z pomiarów przedstawia ilustracja 4.11.

Następnie wykorzystując podobny sposób optymalizacji, z funkcją drgań określoną według wzoru (4.2), wyznaczono opóźnienie pomiarów systemu t_0 względem

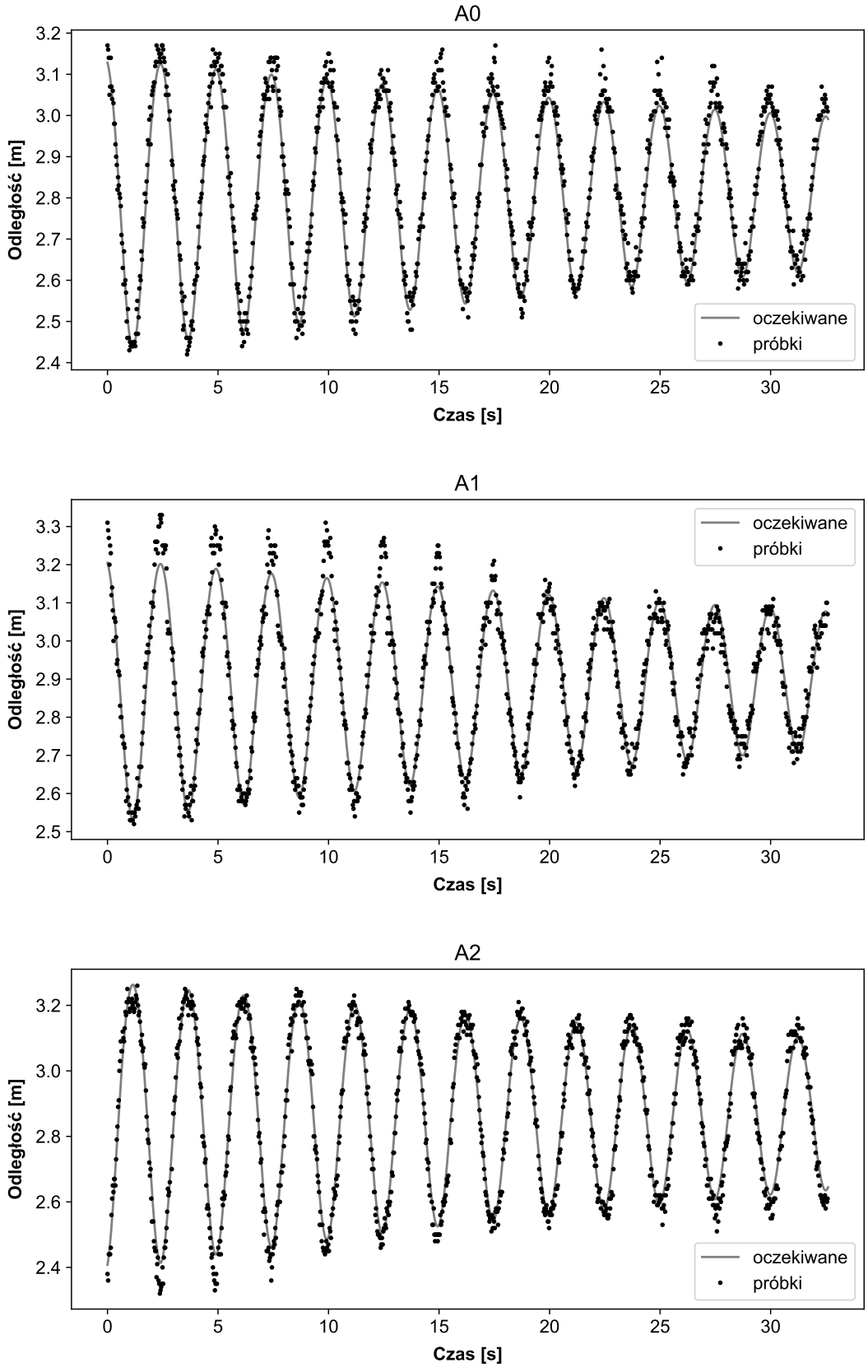


Ilustr. 4.11: Dopasowanie tłumienia wykładniczego w jednym z pomiarów

oczekiwanego ruchu i dopasowano punkty na wykresie. Na ilustracji 4.12 zamieszczone zostały trzy wykresy pomiarów odległości od każdej z kotwic, w trakcie wykonywania jednego z testów, z widoczną krzywą reprezentującą wartości oczekiwane. Tabela 4.4 zawiera oszacowane opóźnienie pomiarów t_0 oraz dane statystyczne takie jak odchylenie standardowe σ od oczekiwanej krzywej oraz liczbę próbek w kolejnych testach **T** przy orientacji anteny **O** względem kierunku ruchu. Niepewność wartości opóźnienia t_0 określono długością trwania pojedynczej klatki kamery, tj. 0,033 s (przy 30 klatkach na sekundę), która miała wpływ na wyznaczenie momentów osiągnięcia ekstremów przez wahadło.

T	**O**	t_0 [s]	**Próbek**	σ [m]		
				A_0	A_1	A_2
1	∥	(0,140 ± 0,033)	1072	0,032	0,042	0,041
2	∥	(0,090 ± 0,033)	902	0,036	0,041	0,034
3	∥	(0,106 ± 0,033)	1073	0,037	0,040	0,039
4	∥	(0,114 ± 0,033)	500	0,035	0,044	0,037
5	∥	(0,096 ± 0,033)	1069	0,043	0,047	0,040
6	⊥	(0,106 ± 0,033)	1469	0,034	0,050	0,068
7	⊥	(0,122 ± 0,033)	744	0,036	0,050	0,066
8	⊥	(0,179 ± 0,033)	1476	0,037	0,051	0,072

Tabela 4.4: Wyniki pomiarów dynamicznych



Ilustr. 4.12: Uzyskane odległości w jednym z testów wahadła

4.4. Testy subiektywne

W końcowym etapie badań przeprowadzono test całego systemu w rzeczywistym środowisku. Do komunikacji wykorzystany został router TP-Link WR-340G działający jako punkt dostępowy bezprzewodowej sieci Wi-Fi, do której były podłączone tylko urządzenia niezbędne do przeprowadzenia testów. Utworzono sieć lokalną bez dostępu do internetu. W ten sposób wyeliminowano opóźnienia i błędy związane z komunikacją innych urządzeń oraz przesyłaniem nadmiernej liczby pakietów w sieci. Aplikacja serwerowa (opisana w rozdziale 3.2) została uruchomiona na laptopie Lenovo ThinkPad X201 działającym pod systemem operacyjnym Linux Mint 17.1 Rebecca. Urządzeniem mobilnym umieszczonym w okularach wirtualnej rzeczywistości był Sony Xperia M4 z systemem operacyjnym Android 5.0. Przebieg testu został zarejestrowany na kamerze, wraz z zapisem zawartości ekranów obydwu urządzeń. Osiągana średnia płynność symulacji wynosiła 50 klatek na sekundę, jednak włączenie przechwytywania ekranu powodowało spadek tej wartości do około 30 FPS. Zmontowany film przedstawiający fragment przebiegu testu jest dostępny pod adresem `https://youtu.be/IRgKCKVc36g` (dostęp 05.09.2018). Pojedyncza klatka została przedstawiona na ilustracji 4.13.

Ilustr. 4.13: Klatka z filmu przedstawiającego przebieg testu subiektywnego

W trakcie tego testu sprawdzono również działanie filtra alfa-beta opisanego w rozdziale 3.3. Jego wyłączenie powoduje, że niepewności pomiarowe w postaci szumów pozycji są widoczne dla użytkownika jako znaczne drgania. Powoduje to duży dyskomfort, do stopnia uniemożliwiającego korzystanie z systemu. Odchylenie standardowe tych błędów może sięgać w niektórych miejscach pomieszczenia nawet ponad 10 cm. Z tego względu, bez wsparcia dodatkowego systemu pozycjonowania (na przykład bezwładnościowego) zastosowanie filtrowania jest niezbędne do komfortowego użytkowania systemu. Podczas testu sprawdzono, że wprowadza to opóźnienia wielkości kilkuset milisekund i ogranicza maksymalną prędkość poruszania się do tempa marszu. Korekta współczynników filtra pozwala na zmniejszenie tych uniedogodnień, kosztem pojawienia się wspomnianych drgań w mniejszym stopniu.

4.5. Test zużycia energii elektrycznej

Podczas badań dokonano próby oszacowania zużycia energii przez urządzenia systemu. Jedna z kotwic była zasilana czterema akumulatorkami Ni-MH. Przed testem zostały one całkowicie naładowane ładowarką everActive NC-3000[(2)] prądem 600 mA każdy, a następnie ich pojemność $C_{\mathbf{full}}$ zmierzono prądem rozładowania 300 mA. Po ponownym pełnym naładowaniu wykonano 5-godzinną sesję opisanych wcześniej badań, sprawdzono pozostały ładunek $C_{\mathbf{left}}$ w akumulatorkach rozładowując je ładowarką do napięcia 0,9 V prądem 300 mA i obliczono zużyty ładunek $C_{\mathbf{used}}$. Dodatkowo zmierzono też orientacyjną rezystancję wewnętrzną $R_{\mathbf{int}}$. Wyniki pomiaru zebrano w tabeli 4.5.

Producent ładowarki nie podaje szczegółowej specyfikacji dokładności wyznaczania ładunków akumulatorów, ani rezystancji wewnętrznej. Ponadto nie została zbadana dokładna zależność wartości napięcia baterii od czasu. Z tego powodu wynik jest tylko orientacyjny i niepewność pomiarową przyjęto na poziomie ±10 %, a średnie napięcie uzyskiwane z czterech ogniw na 5 V. Ostateczny wynik wy-

(2) *Ładowarka everActive NC-3000*: `http://everactive.pl/ladowarka-everactive-nc-3000` (dostęp 05.09.2018)

ID	C_{full} [mAh]	C_{left} [mAh]	C_{used} [mAh]	R_{int} [mΩ]
0	2107	501	1606	225
1	2088	466	1622	240
2	2093	391	1702	246
3	2120	460	1660	248

Tabela 4.5: Wyniki pomiaru zużycia energii elektrycznej przez urządzenie systemu

niósł około (330 ± 33) mAh, co przekłada się na zapotrzebowanie na moc rzędu $(1{,}6 \pm 0{,}2)$ W.

Rozdział 5

Wnioski

System zrealizowany w ramach niniejszej pracy magisterskiej jest przykładem aplikacji mieszanej rzeczywistości (MR), który składa się z kilku warstw działających na różnych urządzeniach i współpracujących ze sobą. Pozwolił on na przetestowanie w jakim stopniu, mechanizm lokalizacji w czasie rzeczywistym (RTLS) wykorzystujący fale radiowe ultraszerokiego zakresu (UWB), jest w stanie przełożyć pozycję użytkownika do wirtualnego świata generowanego przez urządzenie mobilne. W trakcie opracowywania rozwiązania zostały przeprowadzone liczne testy mające na celu sprawdzenie w obiektywny sposób, czy otrzymany system nadaje się do zastosowań w mieszanej rzeczywistości. Pierwszym etapem było zbadanie zdolności kotwic do wzajemnej komunikacji wraz z pomiarem odległości między nimi. Dzięki temu istnieje możliwość automatycznego wyznaczania ich współrzędnych w pomieszczeniu. Uzyskane wyniki pozwalają stwierdzić, że system jest w stanie z wystarczającą dokładnością (do kilku cm), określić rozmieszczenie urządzeń względem siebie. Dzięki temu zostaje wyeliminowana konieczność ręcznego ustawiania ich pozycji w aplikacji. Użytkownikowi pozostaje tylko określenie orientacji kotwic względem kierunków świata, w celu prawidłowej współpracy z kompasem oraz ich wysokości mocowania nad ziemią. W docelowym rozwiązaniu ręczne wprowadzanie wartości obrotu do aplikacji mogłoby zostać zastąpione wizualną kalibracją przy pomocy kamery znajdującej się w telefonie. Użytkownik, znajdując się wewnątrz trójkąta wyznaczonego przez kotwice musiałby nakierować środek pola widzenia na główną kotwicę. System znając aktualne orientację użytkownika oraz

wskazanie z kompasu urządzenia mobilnego znajdującego się w okularach VR byłby w stanie wyliczyć poszukiwany kąt obrotu.

Testy statyczne (opisane w rozdziale 4.2) pozwoliły na określenie odchyłów odczytów systemu w zależności od pozycji użytkownika w pomieszczeniu. Stwierdzono, że najdokładniejsze wyniki uzyskuje się w centralnej części trójkąta wyznaczonego przez kotwice (ilustracja 4.6). Wraz ze zbliżaniem się do ścian lub samych urządzeń niepewność wyznaczanych współrzędnych rośnie. Z tego powodu warto mocować kotwice obejmując jak największą powierzchnię pomieszczenia, w taki sposób, aby użytkownik nie znalazł się bliżej niż 1-2 m od urządzeń, a w szczególności bezpośrednio pod nimi. Należy jednak pamiętać o zaleceniach producenta o umiejscowieniu kotwic co najmniej 15 cm od najbliższej ściany lub obiektów metalowych. Podczas testów zauważono, że orientacja anteny ma wpływ na wartość otrzymywanych odległości (ilustracja 4.8). Różnice przeważnie nie przekraczały kilku cm, co oznacza, że błąd jest praktycznie niezauważalny w aplikacji. Możliwe, że zastosowanie bardziej dookolnej charakterystyki pomogłoby zniwelować ten problem. Alternatywnym rozwiązaniem byłaby korekta lokalizacji na podstawie orientacji użytkownika uzyskanej z kompasu.

W testach wykazano również, że brak bezpośredniej widoczności nadajników (NLOS) powoduje zakłócenia pomiarów. Zasłonięcie najkrótszej linii łączącej znacznik z kotwicą przez człowieka prowadzi do zmian (najczęściej wzrostu) wskazań odległości rzędu kilkunastu cm (ilustracja 4.9). Wynika z tego, że urządzenia muszą być zamontowane w taki sposób, aby na drodze komunikacji nie znajdowały się żadne przeszkody. Sprawdzonym rozwiązaniem są kotwice przymocowane do sufitu lub ścian bocznych, ponad maksymalną wysokością osiąganą przez ruchome urządzenia systemu, natomiast znaczniki powinny znajdować się nad głowami użytkowników. Ponadto takie zakłócenia sugerują, że metoda lokalizacji przy użyciu fal radiowych słabo nadaje się do śledzenia pozycji manipulatorów lub poszczególnych kończyn osoby. Najczęściej będą one znajdować się poniżej głowy powodując tym samym częste przysłanianie komunikacji z kotwicami i tym samym nieprawidłowy odczyt położenia. Częściowym rozwiązaniem tego problemu mogłoby być zastosowanie algorytmów korygujących wskazania urządzeń, które z wyliczeń geome-

trycznych prawdopodobnie są przysłonięte.

Próby dynamiczne na wahadle (rozdział 4.3) miały na celu symulowanie szybkiego poruszania się użytkownika i określenie z jaką dokładnością zostanie wyznaczona pozycja w takich warunkach. Uzyskane opóźnienie t_0 liczone od momentu znalezienia się znacznika w znanej pozycji do uzyskania współrzędnych (widoczne w tabeli 4.4 dla kolejnych testów) średnio wyniosło $(0{,}109 \pm 0{,}033)$ s. Jest to wartość trochę większa niż oczekiwane 0,030 s wynikające ze sposobu przetwarzania pomiarów i komunikacji między urządzeniami systemu DecaWave[(1)]. Różnica może wynikać z konieczności przetworzenia pomiarów przez urządzenia systemu, w szczególności kotwicy podłączonej do komputera i zbierającej dane z pozostałych modułów. Dodatkowe opóźnienie mogą wprowadzać bufory UART znajdujące się zarówno po stronie urządzenia DecaWave jak i komputera oraz biblioteki Qt obsługującej port szeregowy w aplikacji serwerowej. Na dokładność wyznaczenia wartości t_0 wpłynęła także użyta metoda, polegająca na określaniu czasów maksymalnych wychyleń wahadła za pomocą nagrania z kamery. Wykresy odległości widoczne na ilustracji 4.12 zawierają wyraźne zakłócenia w momentach maksymalnych wychyleń. Ich źródłem mogą być dodatkowe drgania, w które wpadało wahadło podczas zmiany kierunku ruchu, w tym częściowy obrót wokół własnej osi, powodujący zmianę odczytywanej odległości, co zostało wykazane w poprzednich testach. Pomijając te zakłócenia, dokładność wyznaczania pozycji w trakcie przemieszczania nie różni się zauważalnie od punktów statycznych.

Testy subiektywne (rozdział 4.4) umożliwiły przetestowanie całego systemu w faktycznym pomieszczeniu. Dzięki temu określono czy opracowane rozwiązanie nadaje się do zastosowania w końcowym produkcie oraz czy jego obsługa jest wystarczająco łatwa i intuicyjna dla użytkownika końcowego. W celu zwiększenia responsywności systemu można zastosować dodatkowy mechanizm lokalizacji, na przykład bezwładnościowy (IMU). Polepszyłoby to reakcję na ruchy o niewielkiej amplitudzie, a dryf żyroskopu i szumy akcelerometru byłyby niwelowane za pomocą systemu DecaWave. Podobne rozwiązania stosuje się w istniejących na ryn-

(1) Opóźnienie powstaje przez uzyskanie zmierzonej odległości dopiero w kolejnej superramce. Szczegóły zostały opisane w rozdziale 3.1.

ku systemach lokalizacji w rzeczywistości mieszanej, które łączą mało precyzyjne mechanizmy wielkoskalowe z dokładniejszymi bezwładnościowymi, lecz podatnymi na dryf. Dodatkowo wykorzystując urządzenie mobilne, takie jak na przykład smartfon, można w tym celu odczytywać pomiary z jego czujników i zastosować je w celu usprawnienia śledzenia położenia użytkownika. Oczywiście dedykowane moduły IMU, jak na przykład znajdujące się w okularach Samsung Gear VR, oferowałyby znacznie większą dokładność i płynność pozycjonowania.

Należy zauważyć, że opracowany system jest w fazie prototypu i część elementów zostałaby zmodyfikowana przed uzyskaniem produktu gotowego do sprzedaży. Jednym z nich jest design okularów VR noszonych przez użytkownika. Dobrym rozwiązaniem byłoby scalenie znacznika wraz z zasilaniem do gogli, a w szczególności miniaturyzacja układów scalonych i anteny. Firma DecaWave zaleca stosowanie sprawdzonych przez siebie anten UWB niewielkich rozmiarów do montażu powierzchniowego [20]: Taiyo Yuden AH 086M555003 lub Partron ACS5200HFAUWB. Ponadto biorąc pod uwagę, że kotwice zwykle znajdują się ponad poziomem znaczników, można dostosować charakterystyki promieniowania ich anten, aby najmocniejszy sygnał był nadawany w górę od głowy użytkownika. Pozwoliłoby to zarówno ograniczyć zużycie energii znaczników, jak i zmniejszyć promieniowanie w kierunku osoby korzystającej z systemu. Kolejnym elementem wymagającym opracowania przed uzyskaniem gotowego produktu jest sposób montażu i zasilania kotwic. W przypadku sufitów podwieszanych istnieje możliwość przymocowania urządzeń do stelaża. Innym sposobem montażu byłoby zawieszenie ich na ścianie na wcześniej przytwierdzonych uchwytach. Bardziej przenośnym rozwiązaniem byłaby możliwość ustawienia kotwic na wysokich półkach regałów lub na statywach, w miejscach gdzie nie ma takiej możliwości. Natomiast zasilanie urządzeń mogłoby zostać zrealizowane poprzez doprowadzenie przewodów i podłączenie ich do zasilacza lub za pomocą wbudowanego akumulatora, który wymagałby ładowania raz na kilka-kilkanaście godzin działania, w zależności od jego pojemności.

Opracowany system może również zostać przeskalowany na większą powierzchnię poprzez dodanie kolejnych kotwic w pomieszczeniu. Konieczne byłoby

w tym celu przeprojektowanie sposobu komunikacji urządzeń ze sobą oraz podłączenie ich do centralnego komputera zarządzającego [8]. Pozwoliłby on na określenie, które kotwice znajdują się najbliżej danego użytkownika i są w stanie określać jego pozycję w przestrzeni z najlepszą dokładnością. Zwiększenie liczby nieruchomych urządzeń umożliwiłoby także zniwelowanie w pewnym stopniu problemu przysłaniania ścieżki komunikacji ze znacznikami. Ponadto z systemu mogłoby korzystać więcej użytkowników w tym samym czasie.

Bibliografia

[1] P. Milgram, F. Kishino: *A Taxonomy of Mixed Reality Visual Displays.* [w:] *IEICE Trans. Information Systems*, vol. E77-D, no. 12, 12.1994, https://www.researchgate.net/publication/231514051_A_Taxonomy_of_Mixed_Reality_Visual_Displays (dostęp: 05.09.2018)

[2] J. Jerald: *The VR Book: Human-Centered Design for Virtual Reality*, ACM and Morgan & Claypool, Nowy Jork, USA, 16.10.2015.

[3] P. Lawitzki: *Application of Dynamic Binaural Signals in Acoustic Games*, Stuttgart Media University, 02.03.2012, http://plaw.info/wp-content/uploads/MasterThesis_Lawitzki_2012.pdf (dostęp: 05.09.2018)

[4] P. Lawitzki: *Android Sensor Fusion Tutorial*, 2014, http://plaw.info/articles/sensorfusion/ (dostęp: 05.09.2018)

[5] Microsoft HoloLens: *The World's First Holographic Head-Mounted Display*, https://www.microsoft.com/en-us/hololens/ (dostęp: 05.09.2018)

[6] IEEE: *Std 802.15.4-2011: Standard for Local and metropolitan area networks – Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*, 05.09.2011, https://standards.ieee.org/getieee802/download/802.15.4-2011.pdf (dostęp: 05.09.2018)

[7] DecaWave: *TREK1000 User Manual*, Version 1.06, 2016, https://www.decawave.com/system/files/resources/trek1000_user_manual_1.06.pdf (dostęp: 05.09.2018)

[8] DecaWave: *APS016 Application Note: Moving from TREK1000 to a product*, Version 2.01, 2015, https://www.decawave.com/sites/default/files/aps016_moving_from_trek1000_to_a_product.pdf (dostęp: 05.09.2018)

[9] DecaWave: *DecaRangeRTLS ARM Source Code Guide*, Version 2.1, 30.10.2015, (niedostępne online)

[10] DecaWave: *DecaRangeRTLS PC Source Code Guide*, Version 2.1, 30.10.2015, (niedostępne online)

[11] DecaWave: *DW1000 EVB*, Revision 0-03, 19.07.2013, (niedostępne online)

[12] STMicroelectronics: *STM32F105xx STM32F107xx Datasheet*, (DocID15274 Rev 10), 03.2017, http://www.st.com/resource/en/datasheet/stm32f105rc.pdf (dostęp: 05.09.2018)

[13] STMicroelectronics: *STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx Reference Manual*, RM0008, (DocID13902 Rev 17), 08.2017, http://www.st.com/resource/en/reference_manual/cd00171190.pdf (dostęp: 05.09.2018)

[14] Android Developers: *Motion sensors*, 24.04.2018, https://developer.android.com/guide/topics/sensors/sensors_motion (dostęp: 05.09.2018)

[15] Android Developers: *Sensors Overview*, 24.04.2018, https://developer.android.com/guide/topics/sensors/sensors_overview (dostęp: 05.09.2018)

[16] Android Developers: *Create an Android Library*, 08.05.2018, https://developer.android.com/studio/projects/android-library (dostęp: 05.09.2018)

[17] Unity - Manual: *Building and using plug-ins for Android*, 18.05.2017, https://docs.unity3d.com/Manual/PluginsForAndroid.html (dostęp: 05.09.2018)

[18] R. Penoyer: *The Alpha-beta Filter*, 07.1993, http://collaboration.cmc.ec.gc.ca/science/rpn/biblio/ddj/Website/articles/CUJ/1993/9307/penoyer/penoyer.htm (dostęp: 05.09.2018)

[19] J. H. Painter, D. Kerstetter, S. Jowers: *Reconciling steady-state Kalman and alpha-beta filter design*, IEEE Transactions on Aerospace and Electronic Systems, 11.1990, https://ieeexplore.ieee.org/document/62250/ (dostęp: 05.09.2018)

[20] DecaWave: *APH0007 Application Note: Antenna Selection / Design Guide for DW1000*, Version 1.0, 2014, https://www.decawave.com/content/antenna-selectiondesign-guide-dw1000 (dostęp: 05.09.2018)

Spis ilustracji

2.1. Podział rozszerzonych i wirtualnych rzeczywistości 10
2.2. Diagram łączenia pomiarów z sensorów IMU 11
2.3. Kartonowe okulary VR Google Cardboard 12
2.4. Okulary Samsung Gear VR wraz z kontrolerem 14
2.5. Zdjęcie wykonane w aplikacji Sony AR effect 16
2.6. Urządzenie Microsoft HoloLens 17
2.7. HTC Vive Pro z kontrolerami i stacjami bazowymi Lighthouse 18
2.8. Oculus Rift z kontrolerami Touch i kamerami Constellation 20
2.9. Urządzenie InterSense IS-900 z okularami CrystalEyes 22

3.1. Płytka rozwojowa DecaWave EVB1000 25
3.2. Schemat wymiany danych w pojedynczym slocie 27
3.3. Programator ST-Link/V2 do programowania układów EVB1000 29
3.4. Wygląd aplikacji serwerowej 31
3.5. Geometria w autopozycjonowaniu kotwic 33
3.6. Graficzne przedstawienie trilateracji znacznika 34
3.7. Okulary VR BOX 2.0 wykorzystane w pracy 38
3.8. Pilot Bluetooth z zestawu VR BOX 2.0 39
3.9. Wygląd sceny w edytorze Unity 40
3.10. Układ współrzędnych sensorów smartfona w systemie Android 41
3.11. Kod QR z parametrami konfiguracyjnymi okularów VR BOX 2.0 42
3.12. Zrzut ekranu z aplikacji mobilnej na telefonie 43
3.13. Użytkownik z okularami oraz znacznikiem EVB1000 44

4.1. Kotwice zamontowane do testów 46
4.2. Rozkład kotwic i miejsc pomiarów statycznych w pomieszczeniu 47
4.3. Histogramy odległości w autopozycjonowaniu 48
4.4. Statyw ze znacznikiem podczas pomiarów statycznych 50
4.5. Ciężarek wskazujący dokładną pozycję umieszczenia statywu 51
4.6. Mapa przedstawiająca dokładność odczytów w punktach statycznych 52
4.7. Dokładność wyznaczania pozycji w funkcji odległości od kotwicy 53
4.8. Wskazania odległości w punkcie E w zależności od orientacji anteny 54
4.9. Wpływ zasłonięcia nadajników na odczyt odległości 55
4.10. Punkt spoczynku wahadła z widoczną miarką 56
4.11. Dopasowanie tłumienia wykładniczego w jednym z pomiarów 57
4.12. Uzyskane odległości w jednym z testów wahadła 58
4.13. Klatka z filmu przedstawiającego przebieg testu subiektywnego 59