



AGH

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

KATEDRA INFORMATYKI STOSOWANEJ

Praca dyplomowa inżynierska

*Sterowanie funkcjami inteligentnego budynku w oparciu o mapy
przepływu użytkowników*

Autor:

Szymon Nowak

Kierunek studiów:

Informatyka

Opiekun pracy:

dr Dariusz Pałka

Kraków, 2017

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystycznego wykonania albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.”, a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.): „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej «sądem koleżeńskim».”, oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Składam serdeczne podziękowania dla promotora, dra Dariusza Pałki, za pomocne uwagi i cenne wskazówki, udzielane podczas pisania pracy, narzeczonej za cierpliwość i wyrozumiałość oraz rodzinie za wsparcie

Spis treści

1. Wprowadzenie	7
1.1. Cele pracy	7
2. Określanie lokalizacji	9
2.1. Lokalizacja użytkownika	9
2.2. Trilateracja	9
2.3. Lokalizacja jako funkcja probabilistyczna Gaussa	10
2.3.1. Funkcja probabilistyczna Gaussa	10
2.3.2. Model routerów jako funkcja Gaussa	11
2.3.3. Lokalizacja użytkownika opisana funkcją Gaussa	12
3. Propagacja sygnału radiowego	15
3.1. Zanik sygnału radiowego	15
3.2. Zysk energetyczny anteny	15
3.3. Margines zaniku	16
3.4. Received Signal Strength Indication	16
3.5. Free-space path loss	16
3.6. Podsumowanie	17
4. Testy sposobów komunikacji radiowej	19
4.1. Wykorzystane urządzenia	19
4.2. Warunki	20
4.3. Pobranie danych i analiza wyników	20
4.4. Wyznaczenie wartości strat	21
4.4.1. Pomiary	21
4.4.2. Wnioski	21
4.5. Przeszkody na drodze sygnału	21
4.5.1. Pomiary	22
4.5.2. Wnioski	23
5. Implementacja	25

5.1. Architektura	25
5.2. Wykorzystana technologia.....	26
5.3. Konfiguracja	27
5.4. Serwer.....	27
5.4.1. Lokalizacja użytkowników mobilnych	28
5.4.2. Sterowanie i zarządzanie urządzeniami	30
5.4.3. Konfiguracja systemu.....	33
5.4.4. Zarządzanie pozostałymi funkcjami aplikacji administracyjnej	35
5.5. Aplikacja administracyjna	36
5.5.1. Opis okien aplikacji	36
5.5.2. Wyświetlanie lokalizacji użytkowników.....	42
5.6. Aplikacja mobilna	43
6. Testy systemu.....	45
6.1. Lokalizowanie użytkownika	45
6.2. Monitorowanie i konfiguracja systemu	47
6.3. Podsumowanie.....	47

1. Wprowadzenie

Wraz z rozwojem technologii, ludzie dążą do pełnej automatyzacji w różnych sektorach swojego życia, m.in. przemyśle, bankowości, motoryzacji. Przykładem prężnie rozwijających się rozwiązań pozwalających na pełną lub częściową automatykę są systemy zarządzania budynkami. Są wykorzystywane nie tylko w fabrykach i miejscach pracy, ale zaczynają również pojawiać się w gospodarstwach domowych. Takie systemy, zwane systemami zarządzania budynkiem (ang. *Building Management System*) [1], opierają się na układzie czujników i pozwalają na monitorowanie i zarządzanie wszystkimi urządzeniami w obrębie oraz w bliskim otoczeniu budynku. Pobierają dane na temat wilgotności powietrza, temperatury, a następnie, na podstawie zebranych informacji, sterują podłączonym do systemu sprzętem. Wykorzystanie BMS pozwala na zoptymalizowanie zużycia energii, mediów, poprawienie funkcjonalności, bezpieczeństwa oraz komfortu. [1]. Ważne w tym zakresie jest również analizowanie przepływu użytkowników. Ma duży wpływ na ogrzewanie pomieszczeń, potrzebę chłodzenia i wentylacji, energię pochłanianą przez oświetlenie oraz wykorzystanie przestrzeni. Uwzględnianie położenia użytkowników podczas sterowania budynkiem pozwala na zaoszczędzenie jednej-trzeciej zużywanej energii [2].

1.1. Cele pracy

Celem niniejszej pracy jest stworzenie systemu zarządzania budynkiem, który będzie sterował podłączonymi do serwera urządzeniami na podstawie analizy lokalizacji użytkowników. Wspomniany pozwoli na kontrolę nie tylko na podstawie danych odbieranych w danym momencie, ale również zebranych przez pewien okres czasu. Do lokalizacji użytkowników zostaną użyte routery WiFi i Beacons, co zmniejszy koszty instalacji systemu, ponieważ nie trzeba będzie kupować dodatkowych, często drogich, czujników. Lokalizacja użytkowników powinna być określana również na podstawie położenia, względem siebie, aplikacji mobilnych, co rozszerzy pulę urządzeń lokalizujących. Jest to szczególnie przydatne w sytuacji, gdy obszar działania systemu będzie posiadał pola, w których sygnały ze stacjonarnych nadajników są zbyt rozproszone, aby w sposób jednoznaczny określić położenie osób korzystających z aplikacji.

W swojej pracy będę się starał także zanalizować i wybrać algorytm lokalizacji użytkowników, który będzie najlepiej pasował do wybranego przeze mnie problemu. Podczas wyboru ważną rolę będzie odgrywać odporność algorytmu na błędy pomiarowe oraz odbicia i interferencje sygnału radiowego, na podstawie którego dokonywana będzie lokalizacja.

Dodatkowo, będę starał się dokonać analizy dwóch wybranych technologii komunikacji radiowej - WiFi i Bluetooth. Obie technologie będą rozpatrywane pod kątem stabilności siły sygnału w zależności od przeszkód oraz środowiska.

Ważne dla całego systemu będzie również stworzenie aplikacji administracyjnej, która pozwoli na konfigurację, monitorowanie położenia oraz analizę przepływu użytkowników.

2. Określanie lokalizacji

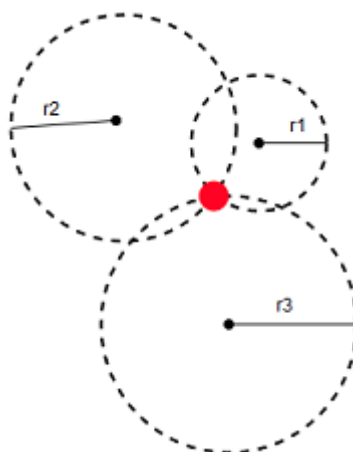
2.1. Lokalizacja użytkownika

Lokalizację użytkownika w systemie określa się na podstawie odległości od punktów, których współrzędne w przestrzeni trójwymiarowej są znane. Są to głównie routery i Beacons, jednak może się zdarzyć, że takim punktem staje się również inny użytkownik. Współrzędne użytkownika można obliczyć, dokonując pomiarów wskaźnika mocy *RSSI*, podając siłę sygnałów odebranych przez otaczające go urządzenia i obliczając na ich podstawie dystans [3].

2.2. Trilateracja

Powszechnie wykorzystywanym sposobem obliczenia lokalizacji użytkownika na podstawie dystansu do znanych punktów, jest trilateracja [4]. Metoda ta polega na przedstawieniu dystansu dzielącego punkt od nadajników w postaci okręgi (lub sfery w przypadku, gdy określamy lokalizację w trzech wymiarach). Następnie, należy wyznaczyć współrzędne punktu przecięcia okręgów [4]. Wyznaczony punkt jest lokalizacją odbiornika, dla którego dokonywaliśmy obliczenia.

Rys. 2.1. Wizualizacja modelu trilateracji



Współrzedną punktu można obliczyć na podstawie wzoru:

$$\begin{cases} (x_p - x_1)^2 + (y_p - y_1)^2 = r_1^2 \\ (x_p - x_2)^2 + (y_p - y_2)^2 = r_2^2 \\ (x_p - x_3)^2 + (y_p - y_3)^2 = r_3^2 \end{cases} \quad (2.1)$$

gdzie:

- x_p, y_p to współrzędne obliczanego odbiornika
- $x_1, y_1, x_2, y_2, x_3, y_3$ to współrzędne znanych punktów
- r_1, r_2, r_3 to odległości między punktem obliczanym, a punktami o znanych współrzędnych

Ta metoda lokalizowania użytkownika w przestrzeni ma jedną, bardzo ważną wadę, która wyklucza jej wykorzystanie w tworzonej systemie - nie potrafi się dostosować do błędów pomiarowych, których w przypadku określania lokalizacji przy użyciu sygnałów radiowych, jest dużo. Aby wyznaczone lokalizacje użytkowników w sposób zbliżony odwzorowywały rzeczywiste położenie, algorytm obliczający musiał być bardziej odporny na błędy pomiarowe [5].

2.3. Lokalizacja jako funkcja probabilistyczna Gaussa

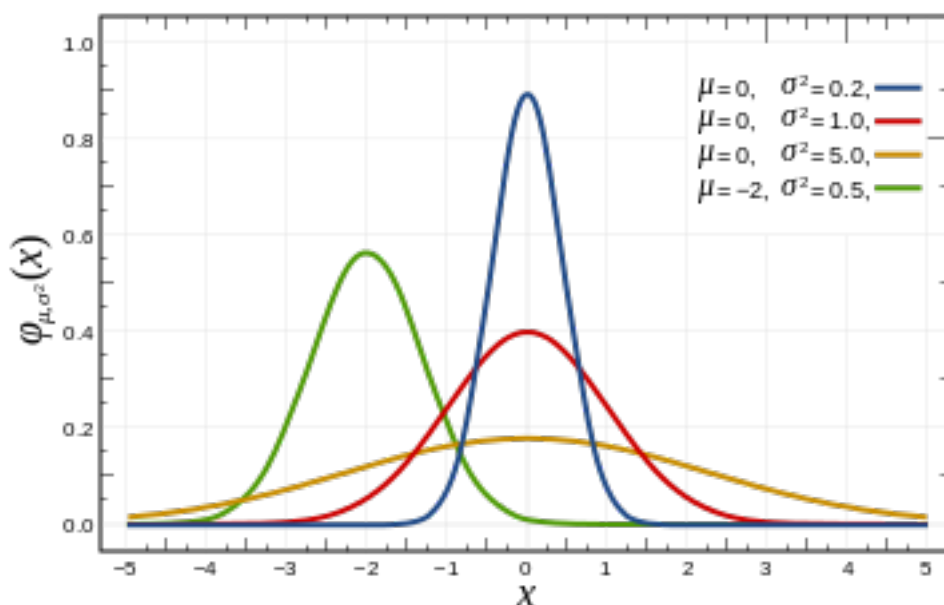
Z racji tego, iż mierzony sygnał, odbierany przez odbiornik, ulega zniekształceniom i odbiciom, określenie dystansu między transponderem, a odbiorcą w sposób liniowy jest niezgodne z fizycznymi zachowaniami sygnału. Siła sygnału oraz straty wywołane przez zniekształcenia i odbicia, określone są w jednostce dB. Wynika z tego, że najlepszym przybliżeniem zmian siły sygnału może być funkcja Gaussa [6]. Zamiast określać konkretne położenie użytkownika, w pracy zastosowane będzie podejście, które pozwoli na przedstawienie jego lokalizacji jako prawdopodobieństwo położenia.

2.3.1. Funkcja probabilistyczna Gaussa

Funkcja probabilistyczna Gaussa jest to krzywa w kształcie dzwonu, symetryczna względem średniej μ oraz uzyskująca wartość maksymalną w punkcie $\frac{1}{\sqrt{2\pi}\sigma}$. Określa się ją za pomocą wzoru:

$$F(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)} \quad (2.2)$$

Zmienne μ będąca średnią, oraz σ będąca odchyleniem standardowym, w pełni opisują tę funkcję [7].

Rys. 2.2. Wykres funkcji Gaussa dla różnych wartości parametrów

Funkcja Gaussa jest szeroko wykorzystywana w statystyce. W elektronice, korzysta się z niej podczas charakteryzowania pomiarów sensorów czy siły sygnałów radiowych.

2.3.2. Model routerów jako funkcja Gaussa

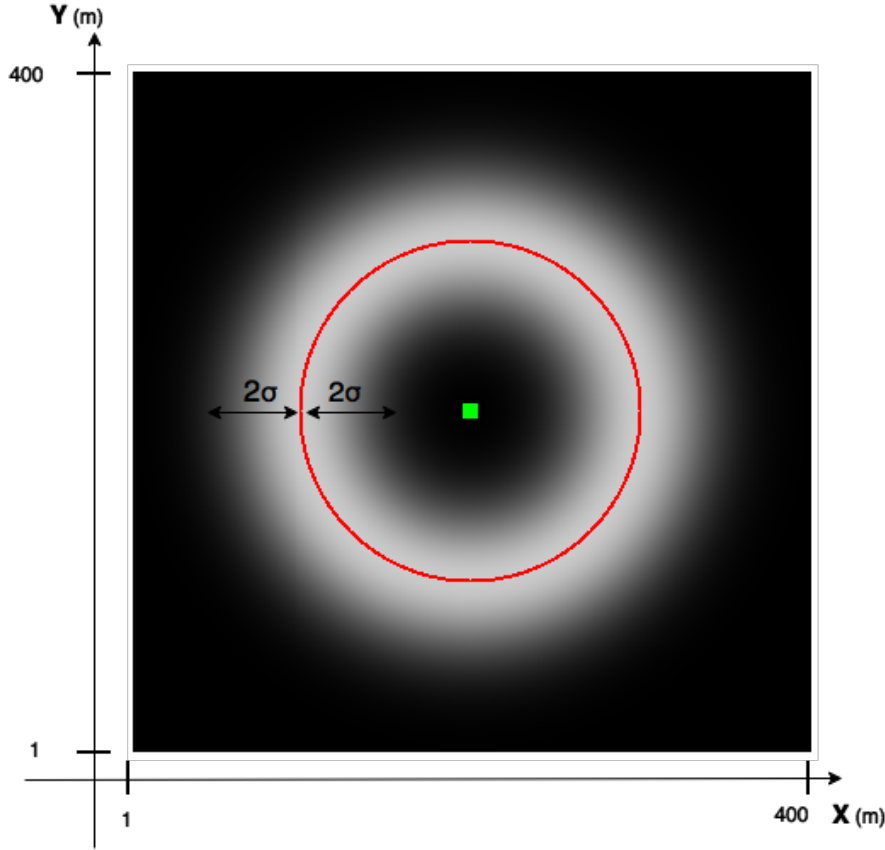
Jeżeli określimy lokalizację odbiornika względem transmitera jako funkcję prawdopodobieństwa Gaussa:

$$F(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\left(\frac{-(x-d)^2}{2\sigma^2}\right)} \quad (2.3)$$

to wartość funkcji określa, jakie jest prawdopodobieństwo, że odbiornik znajduje się w odległości x od transmitera. Największe prawdopodobieństwo przypisywane jest dystansowi, który obliczyliśmy z siły sygnału, dlatego to właśnie ta wartość podstawiana jest pod zmienną d . [8]

Jeżeli transponder opisze się jako funkcję prawdopodobieństwa Gaussa [9], wyznaczoną na podstawie odebranej siły sygnału, zwizualizowany model transmitera w przestrzeni dwuwymiarowej przypomina pierścień, którego gęstość maleje wraz z oddalaniem się od obliczonej z mocy sygnału odległości.

Rys. 2.3. Wizualizacja modelu routera opisanego funkcją Gaussa. Jasność punktów oznacza wielkość prawdopodobieństwa $F(x)$. Na czerwono zaznaczono odległość obliczoną na podstawie siły sygnału transmitera d , a zielony prostokąt oznacza lokalizację transmitera.



2.3.3. Lokalizacja użytkownika opisana funkcją Gaussa

Wyżej opisane podejście, poza realniejszym oddaniem natury rozchodzenia się sygnału w środowisku [10], ma również dodatkową zaletę - w łatwy sposób można wyliczyć prawdopodobieństwo położenia odbiornika wtedy, gdy w modelu znajduje się więcej transmiterów.

Jednym ze sposobów określenia położenia użytkownika jest zsumowanie prawdopodobieństw obliczonych w każdym punkcie w przestrzeni z funkcji Gaussa przypisanych do transmiterów. Prawdopodobieństwo dla punktu (X, Y) może być określone na podstawie wzoru:

$$F(X, Y) = \sum_{r=1}^R \frac{1}{\sigma_r \sqrt{2\pi}} e^{\left(\frac{-(D(X, Y, r) - d_r)^2}{2\sigma_r^2} \right)} \quad (2.4)$$

w którym kolejne symbol oznaczają:

- R - zbiór transmiterów. Z racji tego, iż funkcja Gaussa przyjmuje wartości większe od zera dla całego zakresu $(-\infty, \infty)$, wartość prawdopodobieństwa jest liczona dla każdego transmitera, niezależnie od tego, jak daleko znajduje się on od punktu (X, Y)

- $D(X, Y, r)$ - euklidesowa odległość punktu (X, Y) od transmitera r
- d_r - odległość obliczona na podstawie siły sygnału transmitera r
- σ_r - odchylenie standardowe - wartość wyznaczana na podstawie odległości transmiterów od odbiornika. Dodatkowo, z racji tego, iż sygnał Wi-Fi jest bardziej odporny na nagłe zaniki wynikające ze wzrostu odległości, niż sygnał Bluetooth [11], wartość odchylenia skalowana jest w zależności od tego, jaki typ sygnału wysyła konkretny transponder.

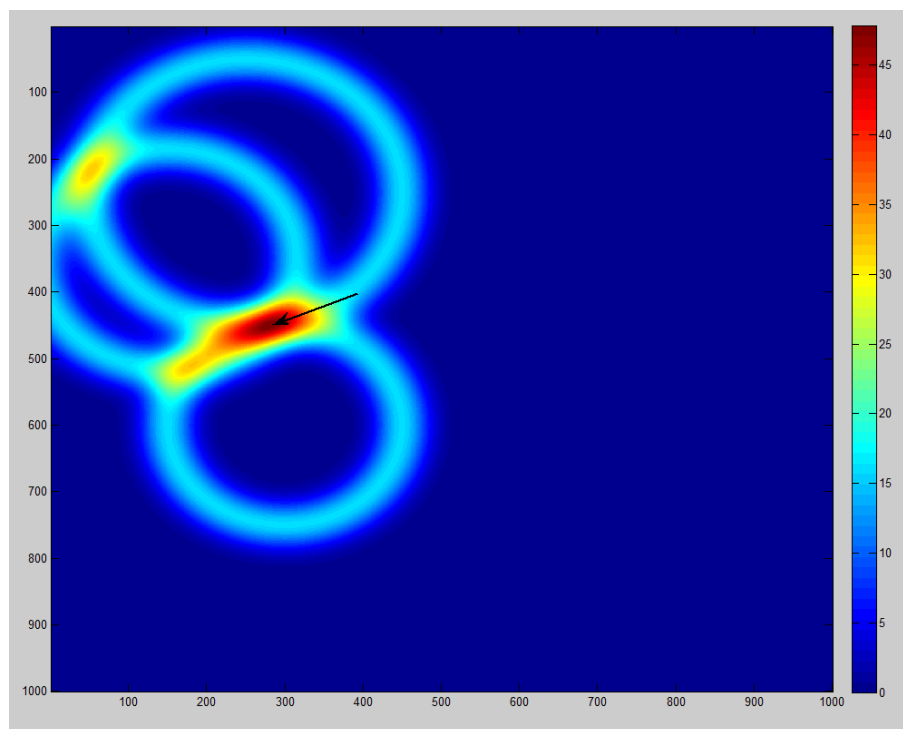
Wartość σ_r można obliczyć za pomocą wzoru:

$$\sigma_r = \frac{D_m * w_r}{30} \quad (2.5)$$

gdzie D_m to długość najdłuższego boku modelu, tworzonego przez sygnały ze wszystkich transponderów, a w_r to waga sygnału, który wysyła transponder r

Po ustaleniu wartości dla wszystkich punktów w przestrzeni, jako lokalizacja odbiornika przyjęty zostaje taki punkt (X_o, Y_o) , którego suma prawdopodobieństw jest najwyższa.

Rys. 2.4. Model lokalizacji odbiornika wykonany w programie MatLab. Punkt o największym prawdopodobieństwie oznaczony jest za pomocą czarnej strzałki.



Zaprezentowany sposób wyznaczania lokalizacji dotyczył obliczeń w przestrzeni dwuwymiarowej. Przejście na przestrzeń trójwymiarową nie stanowi problemu - jedynymi zmianami, jakie należy wprowadzić, są:

- wykonywanie obliczeń dla punktów (X, Y, Z)

- zmiana sposobu obliczania odległości euklidesowej między odbiornikiem, a transponderem - należy wyznaczać odległość w przestrzeni trójwymiarowej.

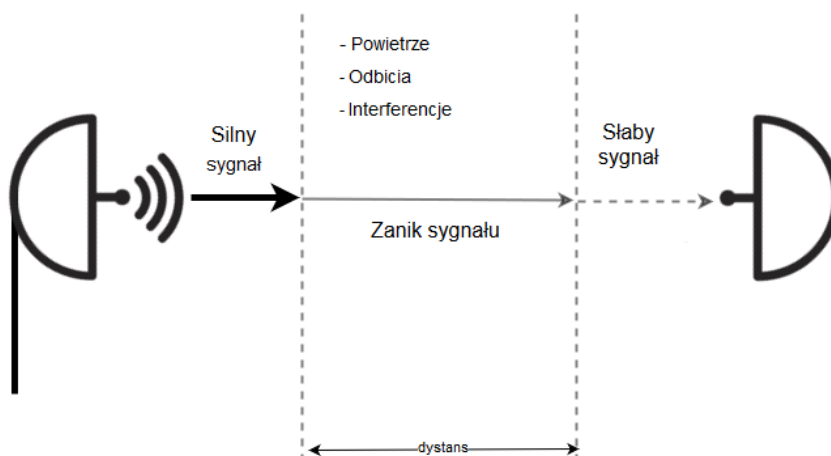
3. Propagacja sygnału radiowego

3.1. Zanik sygnału radiowego

Sygnał radiowy, wysłany przez urządzenie, w wyniku przechodzenia przez ośrodek (najczęściej jest nim powietrze) ulega zanikowi. [12] Różnica w mocy sygnału odczytanego przez odbiornik, a siłą sygnału wysłanego przez nadawcę, pozwala na obliczenie odległości pomiędzy dwoma urządzeniami. Dodatkowymi współczynnikami, które mają wpływ na obliczenia, są:

- zysk energetyczny anteny nadawcy oraz odbiorcy
- margines zaniku, który zależy od czułości odbiornika
- straty w sile sygnału wynikające ze środowiska: przeszkody, odbicia, inne urządzenia nadające

Rys. 3.1. Schemat zaniku sygnału



3.2. Zysk energetyczny anteny

Zysk energetyczny anteny jest to stosunek mocy anteny wypromieniowanej w danym kierunku do mocy wypromieniowanej przez antenę wzorcową [13]. Anteną wzorcową może być m.in. antena izotropowa, czyli antena bez fizycznych rozmiarów, która cały sygnał zasilany wysyła we wszystkich kierunkach. W takim wypadku, zysk energetyczny anteny wyrażany jest w dBi [14].

Na zysk energetyczny mają również wpływ kierunkowość oraz materiał, z którego wykonana jest antena.

3.3. Margines zaniku

Margines zaniku (ang. *fade margin*) jest to wartość określająca różnicę pomiędzy wartością siły odebranego sygnału, a czułością odbiornika [15]. Określa się ją wzorem:

$$FM = P_{rs} - P_{rx} \quad (3.1)$$

gdzie P_{rs} to siła odebranego sygnału.

Margines zaniku jest wyznacznikiem, jak "dobre" jest połączenie między transmiterem i odbiorcą [16]. Czym wyższa wartość marginesu zaniku, tym bardziej niezawodne jest połączenie. Wartość marginesu zaniku określa się w jednostce dB .

3.4. Received Signal Strength Indication

Received Signal Strength Indication (skrótom RSSI) jest to miara określająca moc sygnału odbieranego. Przyjmuje ona wartości niedodatnie (gdzie 0 oznacza sygnał najsilniejszy). Jednostką, w jakiej określa się siłę sygnału, jest dBm , która jest logarytmiczną jednostką miary mocy odniesioną do mocy $1mW$ [17].

System Android pozwala na odczytanie siły odbieranego sygnału. Można do tego wykorzystać API *WifiManager* (w przypadku odczytu sygnału WiFi) oraz *BroadcastReceiver* (w przypadku odczytu sygnału Bluetooth) [18].

3.5. Free-space path loss

Free-space path loss (FSPL) jest to utrata siły sygnału, spowodowana przejściem fali elektromagnetycznej przez ośrodek (najczęściej powietrze) [19].

Wzór na obliczanie FSPL [20]:

$$FSPL = P_{tx} + AG_{tx} + AG_{rx} - P_{rx} - FM - L \quad (3.2)$$

Gdzie symbole oznaczają:

- P_{tx} - siła transmitera, wyrażona w dBm
- AG_{tx} - zysk energetyczny anteny transmitera, wyrażony w dB
- AG_{rx} - zysk energetyczny anteny odbiorcy, wyrażony w dB
- P_{rx} - czułość odbiornika, wyrażona w dBm
- FM - margines zaniku sygnału (fade margin) - określa jak duży jest margines różnicy pomiędzy uzyskaną siłą sygnału, a czułością odbiornika

- L - straty wynikające np z oddziaływania innych transponderów, przeszkód itp.

Dodatkowo, FSPL można obliczyć, używając następujący wzór [21]:

$$FSPL = 20\log_{10}\left(\frac{d}{d_0}\right) + 20\log_{10}(f) + K \quad (3.3)$$

Gdzie symbole oznaczają:

- d - dystans dzielący transponder od odbiorcy, wyrażony w metrach
- d_0 - dystans referencyjny - w tym wypadku 1 metr
- f - częstotliwość transpondera - wyrażona w MHz
- K - stała, którą można określić wzorem:

$$K = 20\log_{10}\left(\frac{4\pi d_0}{C}\right) \quad (3.4)$$

gdzie d_0 to dystans referencyjny (taki sam jak we wzorze wyżej), a C to długość fali emitowanej przez transponder

Po przekształceniu wzoru, uzyskujemy:

$$d = 10^{\left(\frac{FSPL - K - 20\log_{10}(f)}{20}\right)} \quad (3.5)$$

A po połączeniu obu wzorów dostajemy:

$$d = 10^{\left(\frac{P_{tx} + AG_{tx} + AG_{rx} - P_{rx} - FM - L - K - 20\log_{10}(f)}{20}\right)} \quad (3.6)$$

Po podstawieniu wzoru na margines zaniku do wzoru (3.6), uzyskujemy

$$d = 10^{\left(\frac{P_{tx} + AG_{tx} + AG_{rx} - P_{rs} - L - K - 20\log_{10}(f)}{20}\right)} \quad (3.7)$$

3.6. Podsumowanie

Przedstawione powyżej współczynniki mają duży wpływ na dokładność i jakość obliczonej pozycji użytkownika. Najważniejszym współczynnikiem są straty, które należy uwzględnić podczas wyznaczania odległości. Czym więcej przeszkód i zakłóceń spotka na swojej drodze sygnał, tym jego siła będzie mniejsza. Wiąże się z tym duży błąd podczas wyznaczania lokalizacji odbiornika [3].

Dodatkowy wpływ na obliczenia ma siła obu anten - transpondera i odbiornika. Im większy jest ich zysk energetyczny, tym sygnał jest bardziej odporny na straty, a co za tym idzie, dystans pomiędzy dwoma urządzeniami, obliczony na podstawie $FSPL$, jest dokładniejszy i lepiej odzwierciedla rzeczywiste położenie [22].

Również margines zaniku ma duży wpływ na jakość odbieranego sygnału - im jest wyższy, tym rzadziej występuje minimum poziomu wydajności, które negatywnie wpływa na stabilność siły sygnału [5].

4. Testy sposobów komunikacji radiowej

Testy mają na celu zbadać skuteczność wyznaczania odległości między transmittersem, a odbiornikiem dla sygnałów Wifi i Bluetooth.

Testy odbywać się będą w dwóch etapach:

- w pierwszym, odbiornik i transmittersem będą oddalone od siebie o około 1m. Mierzona będzie siła odbieranego sygnału. Celem tego etapu jest określenie, jak duże straty siły sygnału związane są z komunikacją Wifi i Bluetooth. Straty mogą wynikać z izolacji obudowy, odbić, interferencji kilku fal lub z braku kierunkowości anteny (antena wbudowana).
- w drugim etapie, obliczone wcześniej wartości strat zostaną wykorzystane, aby zmierzyć, jak zmienia się siła sygnału, kiedy na drodze pojawi się przeszkoda. Do testów wykorzystana została książka o formacie A4 grubości 7 centymetrów, drewniane drzwi o grubości 5 centymetrów oraz ściana o grubości 26 centymetrów.

Uzyskane informacje pozwolą obliczyć współczynnik strat, jaki należy uwzględnić podczas późniejszego obliczania lokalizacji użytkowników oraz pozwoli przydzielić każdemu ze sposobów komunikacji radiowej odpowiednią wagę, w zależności od jego odporności na zakłócenia.

4.1. Wykorzystane urządzenia

1. Smartphone Sony Xperia Z1 Compact (D5503) - odbiornik

Dane techniczne:

- Częstotliwość - 2,4GHz
- Przyrost siły sygnału z anteny WiFi - 2dBi
- Przyrost siły sygnału z anteny Bluetooth - 0dBi

2. Router TP-Link TD-W8970 - nadajnik

Dane techniczne:

- Częstotliwość - 2,4GHz
- Dwie zewnętrzne anteny kierunkowe

- Przyrost siły sygnału z anteny - 4dBi
- Siła transmitera - 16.5dBm

3. Smartphone Samsung Grand 2 (G7102) - nadajnik

Dane techniczne:

- Jedna antena wbudowana
- Przyrost siły sygnału z anteny - 0dBi
- Siła transmitera Bluetooth - 6dBm

4.2. Warunki

Wszystkie pomiary wykonywane były w pomieszczeniu zamkniętym, bez przeszkód na drodze sygnału. Wszystkie urządzenia znajdowały się na tej samej wysokości, skierowane do siebie górną częścią obudowy (w przypadku routera, skierowany był on do odbiornika swoimi antenami kierunkowymi).

Rys. 4.1. Zdjęcie urządzeń pomiarowych oraz środowiska testowego



4.3. Pobranie danych i analiza wyników

Dla każdego eksperymentu została wykonana odpowiednio duża ilość powtórzeń (od 70 do 120), aby wynik był dokładniejszy. W tym celu, została stworzona lekka aplikacja, która pobierała zgłoszenia telefonu i zapisywała je do pliku tekstowego. Następnie, po skończonych testach, parsowała dane i wyliczała średnią dla sił sygnałów pochodzących z interesującego nas źródła (sygnały pochodzące z innych źródeł były odrzucane). Tak obliczone dane zostały zawarte w tabelach w dalszej części rozdziału.

4.4. Wyznaczenie wartości strat

Eksperyment polegał na ustawieniu transmitera w odległości 1 metra od odbiornika, na jednym poziomie, antenami do siebie. Na podstawie siły sygnałów obliczana została wartość strat, jakie musiałyby być uwzględnione, aby odległość obliczona równała się odległości fizycznej.

Rys. 4.2. Szkic eksperymentu nr 1



4.4.1. Pomiary

Wartości obliczone dla eksperymentu nr 1, w którym Router TP-Link TD-W8970 jest transmiterem sygnału Wifi, a Samsung Grand 2 transmiterem Bluetooth:

Sygnał	Ilość prób	Średnia siła sygnału (w dBm)	Obliczona średnia wartość strat (dB)
Wi-Fi	163	-38.88	21.18
Bluetooth	113	-57.5	23.3

4.4.2. Wnioski

Z danych uzyskanych podczas eksperymentu wynika, że straty spowodowane zakłóceniami sygnału Bluetooth są wyższe niż spowodowane zakłóceniami sygnału Wi-Fi. Może się to wiązać z tym, że transponder Wi-Fi ma większą moc, zaś kierunkowe anteny zmniejszają ilość zakłóceń wynikających z odbicia się sygnału. Wartości strat obliczone dla obu sposobów komunikacji zostaną wykorzystane podczas eksperymentu nr 2 oraz będą miały wpływ na wybór wagi dla siły sygnałów podczas określania lokalizacji użytkowników.

4.5. Przeszkody na drodze sygnału

Drugi eksperyment ma na celu zbadanie odporności każdego z rodzajów sygnałów na zakłócenia związane ze stojącą na drodze przeszkodą. Wnioski z tego eksperymentu będą miały duży wpływ na

działanie systemu, ponieważ w rzeczywistym środowisku, w którym ma działać tworzony system, transponder od użytkownika będzie dzielić czasami nawet kilka ścian i ważne jest, aby odpowiednio dobrać wagi dla sygnałów.

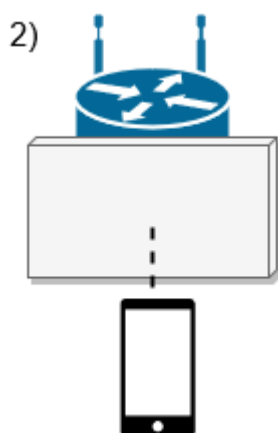
W pierwszej części eksperymentu, transponder został oddalony od odbiornika o 1 metr. Na drodze sygnału została postawiona gruba książka w formacie A4. W drugiej części eksperymentu, transponder od odbiornika oddalony został o 2 metry, w taki sposób, aby na drodze sygnału znalazły się najpierw drewniane drzwi, a w kolejnej części eksperymentu ściana.

Do dokonania pomiarów zostały wykorzystane te same urządzenia jak w przypadku eksperymentu nr 1. Wartość błędu bezwzględnego obliczana była na podstawie wzoru:

$$\delta = |D_r - D_o| \quad (4.1)$$

gdzie D_r to rzeczywista odległość między transponderem, a odbiornikiem, zaś D_o to odległość obliczona na podstawie siły sygnału.

Rys. 4.3. Szkic eksperymentu nr 2



4.5.1. Pomiary

Wartość siły sygnałów, obliczony dystans oraz wielkość błędu dla pomiaru sygnału Wifi i Bluetooth, dla którego przeszkodą była książka przy dystansie 1 metra:

Sygnał	Ilość pomiarów	Średnia siła sygnału (w dBm)	Średnia obliczona odległość (m)	Błąd bezwzględny (m)	Błąd względny (%)
Wifi	182	-40,81	1,24	0,24	24
Bluetooth	70	-61.46	1,57	0,57	57

Wartość siły sygnałów, obliczony dystans oraz wielkość błędu dla pomiaru sygnału Wifi i Bluetooth, dla którego przeszkodą były drzwi przy dystansie 2 metrów:

Sygnał	Ilość pomiarów	Średnia siła sygnału (w dBm)	Średnia obliczona odległość (m)	Błąd bez-względny (m)	Błąd względny (%)
Wifi	112	-48,1	2,89	0,89	44,5
Bluetooth	107	-67,05	3,01	1,01	50,5

Wartość siły sygnałów, obliczony dystans oraz wielkość błędu dla pomiaru sygnału Wifi i Bluetooth, dla którego przeszkodą była ściana przy dystansie 2 metrów:

Sygnał	Ilość pomiarów	Średnia siła sygnału (w dBm)	Średnia obliczona odległość (m)	Błąd bez-względny (m)	Błąd względny (%)
Wifi	103	-48,3	2,96	0,96	48,0
Bluetooth	110	-66,95	2,97	0,97	48,5

4.5.2. Wnioski

Wyniki eksperymentu nr 2 jasno wskazują, że sygnał Wifi jest bardziej odporny na zmiany spowodowane pojawiającą się na drodze przeszkodą. Ma to duży wpływ na obliczanie lokalizacji użytkownika na podstawie sygnału i dlatego technologia Wifi będzie miała wyższą wagę w tworzonym systemie. Dodatkowo, warto zwrócić uwagę, że błędy wynikające z obecności na drodze sygnału drzwi oraz ściany są mniejsze, niż można się było spodziewać. Może to wynikać z faktu, że sygnał nie rozchodzi się tylko w jednym kierunku i dzięki temu, podczas przechodzenia przez drzwi, część sygnału napotykała również na ścianę, co spowodowało większe zakłócenia.

5. Implementacja

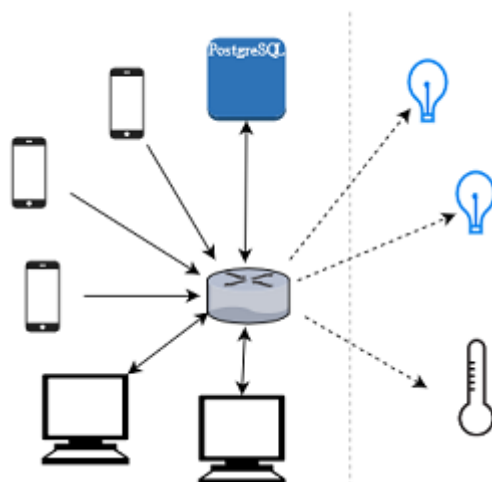
5.1. Architektura

Architektura systemu oparta jest na topologii gwiazdy. Centralny punkt stanowi serwer, z którym łączą się aplikacje mobilne, odpowiedzialne za lokalizację użytkowników, którzy z nich korzystają. Telefony komunikują się z serwerem jednostronnie, a jedyną zwrotną informacją, jaką uzyskują, jest to, czy żądanie zostało pozytywnie przyjęte.

Do serwera mogą podłączać się również aplikacje klienckie, pozwalające administratorom na konfigurację systemu oraz podgląd zebranych przez serwer lokalizacji. Podstawowym warunkiem komunikacji między serwerem, a użytkownikiem aplikacji klienckiej jest wcześniejsza autoryzacja loginem i hasłem. Aplikacja serwerowa połączona jest z bazą danych, w której przechowywane są lokalizacje użytkowników oraz dane konfiguracyjne systemu. W architekturze wykorzystana jest *open-sourcowa* baza danych *PostgreSQL*. Dostęp do danych zapewniony jest przy użyciu mapowania obiektowo-relacyjnego, zaś wykorzystanie wzorca projektowego 'Repozytorium' pozwala na odseparowanie warstwy logiki biznesowej aplikacji od warstwy danych. Dostęp aplikacji klienckich do danych może się odbywać tylko i wyłącznie przy pośrednictwie aplikacji serwerowej.

Ostatnim elementem, z którym komunikuje się serwer, są urządzenia sterowane. Fizycznie nie należą do systemu, ponieważ nie są dostarczane oraz z góry zdefiniowane podczas wdrażania systemu. Są jednak podstawowym elementem, dzięki któremu system ma sens działania. Urządzenia sterujące to wszystkie urządzenia, które są zdefiniowane w aplikacji serwerowej, a którymi steruje serwer. Serwer zarządza urządzeniami na podstawie lokalizacji użytkowników. Medium komunikacji nie jest z góry narzucone - w zaimplementowanym domyślnie module komunikacyjnym, wysyłanie informacji pomiędzy serwerem, a urządzeniami odbywa się przy użyciu protokołu HTTP.

Rys. 5.1. Schemat architektury systemu



5.2. Wykorzystana technologia

Podstawową i główną technologią wykorzystaną podczas tworzenia systemu jest *.NET Framework*. Jest to stworzona przez firmę *Microsoft* platforma programistyczna, która pozwala na tworzenie efektywnych, nowoczesnych aplikacji. Podstawowym językiem, w którym napisane są wszystkie aplikacje należące do systemu, jest *C#* [23]. Jest to obiektowy język programowania, jeden z najważniejszych języków wykorzystywanych podczas tworzenia aplikacji na platformę *.NET*.

Aplikacja serwerowa wykorzystuje serwisy udostępniane przez *ASP.NET*. Jest to webowa platforma, wchodząca w skład *.NET Framework*, która pozwala na tworzenie aplikacji opartych na architekturze *Model-View-Controller*. Komunikacja serwera z aplikacjami odbywa się przy użyciu protokołu *HTTP*. Platforma *ASP.NET* wspiera wykorzystanie czasowników *HTTP*, a w połączeniu z zastosowaniem stylu architektonicznego *REST*, pozwala na wydajną i efektywną współpracę wszystkich elementów systemu - aplikacji mobilnych i klienckich [24].

Algorytm określania lokalizacji użytkownika oraz wzory obliczające dystans dzielący odbiornik od transponderów, zostały napisane w *F#*. Jest wieloparadygmatowy, głównie funkcyjny, język programowania, wspierany przez platformę *.NET* [25]. Wybrałem ten język do wypisanych powyżej zadań, ponieważ uważam, że języki funkcyjne najlepiej nadają się do algorytmów opartych na wzorach matematycznych - w przeciwieństwie do innych języków, zachowują czystość i czytelność kodu, nie tracąc nic na szybkości wykonywania. Dodatkowo, zależało mi na poznaniu i nauczaniu się podstaw pisania w tym języku. Algorytmy lokalizacji napisane są w osobnym projekcie, a dzięki skompilowaniu kodu do przenośnej biblioteki *.NET*, mogą być bezproblemowo podłączone do aplikacji serwerowej.

Serwer, podczas dostępu do danych, wykorzystuje bibliotekę *NHibernate*. Jest to narzędzie do mapowania obiektowo-relacyjnego, które pozwala na szybki dostęp oraz operowanie na danych, zapisanych w bazie danych. Połączenie serwera z bazą zarządzane jest przez narzędzie *Npgsql* - dostawcę danych, korzystającego z bibliotek *ADO.NET*.

Systemem, na który stworzona jest aplikacja mobilna, jest *Android*. Tworzona jest również w języku *C#*, na co pozwala wykorzystanie frameworka *Xamarin*. *Xamarin* pozwala na pisanie aplikacji w języku *C#* na wszystkie najpopularniejsze platformy [26]. Jest to najważniejszy powód, dla którego wybrałem ten framework jako główne narzędzie do tworzenia swojej aplikacji mobilnej. Takie podejście pozwoli mi w przyszłości, w ramach potrzeby, na szybsze przejście na inne platformy mobilne. Kod napisany w *Xamarinie* jest tłumaczony w momencie kompilacji na kod zarządzany natywnego języka platformy *Android*, *Java*, przez co nie powoduje on spadku wydajności aplikacji. Dodatkowym powodem, dla którego sięgnąłem po to narzędzie, była chęć nauczenia się korzystania z niego.

Aplikacje klienckie korzystają z *WinForms* - biblioteki klas GUI, wchodzące w skład *.NET Framework*. Aplikacja wymaga, aby na komputerze osoby korzystającej z niej zainstalowany był *.NET Framework* w wersji 4.5 albo wyższej.

5.3. Konfiguracja

Aplikacja serwerowa jest dostarczona jako *Web Deployment Package*, który należy wdrożyć do stworzonej wcześniej witryny w menedżerze internetowych usług informacyjnych (*IIS*). Kiedy aplikacja zostanie poprawnie wdrożona, należy skonfigurować dane dostępowe do bazy danych poprzez edycję *'connection-string'* w pliku *Web.config* w folderze serwera. Po tej operacji należy zrestartować serwer. Aby utworzyć wymagane przez serwer struktury bazodanowe, należy wywołać w utworzonej dla systemu bazy danych skrypt *SQL*. Znajduje się on w pliku *'script.sql'* i dostarczony jest on wraz plikiem wdrożeniowym serwera.

Przed uruchomieniem aplikacji administracyjnej, należy skonfigurować dane znajdujące się w pliku *'config.xml'*. W danej *'PictureLocation'* należy podać względną ścieżkę do mapy, odnosząc się do folderu, w którym znajduje się wykonywalna aplikacja. Do danej *'ServerIp'* należy przypisać adres ip serwera wraz z portem, zaś w *ClientIp* należy wstawić adres ip klienta, bez portu. Dane *'Login'* i *'Password'* są opcjonalne i zawierają zapamiętane dane logowania, którymi wypełniane jest pierwsze okno aplikacji. Po włączeniu aplikacji i zalogowaniu się (domyślne dane logowania to *'admin'* i *'admin'*), okno konfiguracji pozwala na bardziej rozbudowaną konfigurację systemu.

5.4. Serwer

Serwer jest aplikacją, która ma analizować zabrane dane, sterować urządzeniami zewnętrznymi oraz stanowić most pomiędzy klientem administracyjnym, a aplikacjami mobilnymi. Na zadania aplikacji serwerowej składają się:

- Zbieranie danych z aplikacji mobilnych oraz wyznaczanie lokalizacji użytkowników
- Zezwalanie użytkownikowi administracyjnemu na konfigurację systemu poprzez żądania wysyłane z aplikacji klienckiej

- Sterowanie i zarządzanie urządzeniami zewnętrznymi poprzez analizowanie danych odebranych z aplikacji mobilnych
- Dostarczanie, w formie skumulowanej lub w czasie rzeczywistym, danych na temat położenia użytkowników do analizy i monitoringu

5.4.1. Lokalizacja użytkowników mobilnych

5.4.1.1. Pobranie i analiza danych

Serwer pobiera dane od użytkowników w formie żądań HTTP. Każde żądanie wysłane na serwer musi posiadać adres MAC urządzenia Bluetooth wysyłającego oraz listę zarejestrowanych sygnałów. Każda pozycja na liście sygnałów musi posiadać nazwę urządzenia, którego sygnał został odebrany (w przypadku sygnału wysłanego przez router jest to *SSID* sieci, zaś w przypadku urządzeń Bluetooth jest to adres MAC), typ sygnału (WIFI albo Bluetooth) oraz zarejestrowaną siłę sygnału, określoną w dBm. Następnie, dla każdego elementu z listy dociągane są stałe dane zarejestrowane w systemie - lokalizacja transmitera oraz waga sygnału. Dane na temat routerów oraz stałych urządzeń Bluetooth (np. *Beconów*) pobierane są z bazy danych. Jeżeli jakiś sygnał Bluetooth nie widnieje w bazie danych, sprawdzane są ostatnie żądania od urządzeń mobilnych, dla których udało się określić lokalizację, a czas od ich ostatniej aktualizacji nie jest większy niż 4 sekundy. Jeżeli sygnał pochodzi od jednego z tych urządzeń, potrzebne informacje pobierane są z dynamicznie budowanej, lokalnej bazy wiedzy. Jeżeli sygnał nie figuruje ani w bazie danych, ani w bazie dynamicznej, zostaje uznany jako sygnał przypadkowy i odrzucony. Waga sygnału przyjmuje wartości w skali od 1 do 4. Domyślnie, sygnałowi pochodzącemu od routera WiFi nadawana jest waga 3, sygnałowi ze stałego urządzenia Bluetooth waga 2, zaś sygnały pochodzące od innych użytkowników mobilnych mają wagę 1. Wagę stałych urządzeń WiFi i Bluetooth można edytować przy użyciu panelu konfiguracyjnego w aplikacji klienckiej. W ostatnim kroku, dla każdego zarejestrowanego sygnału obliczana jest odległość urządzenia od użytkownika. Wykorzystywana do tego jest lokalizacja użytkownika, wzór na *Free-space path loss* oraz dane statyczne (jak siła anten, siła transmitera itp.).

5.4.1.2. Algorytmiczne wyznaczenie lokalizacji

Celem algorytmu jest wyznaczenie punktu, dla którego suma prawdopodobieństw wynikających z odległości użytkownika od urządzenia, jest największa.

Dane pobrane od użytkownika, uzupełnione o statyczne dane przechowywane w systemie, przekazane są do sekcji napisanej w języku *F#*. Na wstępie, do każdego zarejestrowanego sygnału zostaje przypisana funkcja probabilistyczna Guassa, określająca prawdopodobieństwo znalezienia się użytkownika w danym punkcie w przestrzeni, gdzie stała μ przyjmuje wartość równą dystansowi obliczonemu na podstawie siły sygnału urządzenia. Dzięki takiemu podejściu, każdy sygnał można zwizualizować jako sferę, której powierzchnia zbliżona jest do chmury. Największe zagęszczenie prawdopodobieństwa występuje

dla średnicy równej odległości obliczonej z siły sygnału, rzadnie zbliżając się i oddalając od środka sfery. Pierwszym krokiem algorytmu jest wyznaczenie prostopadłościanu, dla którego wykonywane będą obliczenia. Wielkość bryły dobrana jest tak, aby wewnątrz niej znalazły się wszystkie sfery sygnałów (przy uwzględnieniu zapasu równego 2σ). Następnie prostopadłościan oraz sfery są normalizowane w taki sposób, aby początek układu zaczynał się w punkcie (0,0,0), zaś wszystkie wartości współrzędnych przyjmowały tylko wartości nieujemne. Celem takiej operacji jest uproszczenie algorytmu oraz usunięcie potrzeby skalowania iteratorów oraz odnośników do elementów w tablicach.

Kolejnym krokiem algorytmu jest podział prostopadłościanu na części, dla których liczona będzie suma prawdopodobieństw. Celem tego kroku jest podział pola działania na jednakowe sześciany w taki sposób, aby w żadnym wymiarze ilość sześciąt nie przekroczyła 100. Aby to uzyskać, najdłuższy bok prostopadłościanu zostaje podzielony na 100 części. Następnie, boki w pozostałych 2 wymiarach zostają podzielone na sześciany o krawędziach o równej długości. Podział prostopadłościanu na sześciany pozwala na wyeliminowanie błędów obliczeniowych wynikających z nierealistycznego podziału pola obliczeniowego.

Kolejnym krokiem algorytmu jest wyliczenie sumy prawdopodobieństw ze wszystkich sfer sygnałów dla każdego sześcianu w prostopadłościanie. Każde prawdopodobieństwo, będące składową sumy, przemnażane jest przez wagę danego sygnału.

$$P(x, y, z) = \sum_{r=1}^R w_r * \frac{1}{\sigma_r \sqrt{2\pi}} e^{\left(\frac{-(d-\mu_r)^2}{2\sigma_r^2}\right)} \quad (5.1)$$

gdzie:

- x, y, z - współrzędne konkretnego sześcianu
- R - ilość routerów w modelu
- w_r - waga konkretnego rutera
- σ_r - odchylenie standardowe, w naszym modelu wynosi: długość najdłuższego boku prostopadłościanu pomnożona przez wagę rutera, podzielona następnie przez 30
- μ_r - średnia, w naszym modelu jest to wartość równa, obliczonej na podstawie siły sygnału, odległości użytkownika od źródła sygnału
- d - odległość Euklidesowa pomiędzy sześcianem, a routerem

Następnie wybierany jest sześcian, dla którego suma prawdopodobieństw jest najwyższa. Jeżeli długość boku sześcianu jest równa lub mniejsza od ustalonego wcześniej przybliżenia, współrzędna sześcianu staje się lokalizacją naszego użytkownika. Jeżeli długość boku sześcianu jest większa, dla wybranego sześcianu dobierane są jego sześciany sąsiednie. Następnie wybrane 27 sześciąt staje się nowym modelem obliczeniowym. Każdy wymiar nowego pola dzielony jest na 9 równych części, dzięki czemu uzyskuje się 729 sześciąt. Algorytm zostaje powtórzony, aż lokalizacja nie zostanie określona z interesującym nas przybliżeniem. Sześcian o największej sumie prawdopodobieństw staje się lokalizacją

użytkownika.

Ostatnim krokiem algorytmu jest przeliczenie obliczonej lokalizacji przy użyciu danych uzyskanych podczas normalizacji, aby obliczone współrzędne odpowiadały lokalizacji w początkowym obszarze.

5.4.1.3. Zarządzanie lokalizacją użytkownika

Po pozytywnym obliczeniu lokalizacji, zostaje ona zapisana w bazie danych, aby potem mogła być użyta do wyświetlenia skumulowanej mapy przepływu użytkowników albo do sterowania urządzeniami. Następnie, lokalizacja zostaje asynchronicznie wysłana do wszystkich klientów administracyjnych, którzy zarejestrowali swoją chęć pobierania danych w trybie real-time (w czasie rzeczywistym). Dodatkowo, lokalizacja wraz z adresem MAC użytkownika, zostaje przekazana do wątków urządzeń sterowanych, które wykorzystują te dane do podjęcia decyzji o wywołaniu przypisanego urządzeniowi eventu. W ostatnim kroku, lokalizacja zostaje dodana (lub podmieniona, jeżeli wpis o danym użytkowniku już istnieje) w bazie dynamicznej, aby ta informacja mogła posłużyć przy wyznaczaniu lokalizacji innych użytkowników.

5.4.2. Sterowanie i zarządzanie urządzeniami

Serwer, poza pobieraniem i analizą danych, zajmuje się również sterowaniem przydzielonymi mu urządzeniami. Sterowanie urządzeniami odbywa się na dwa sposoby:

- Przy użyciu eventów - urządzenia mogą mieć przypisaną klasę obsługującą zdarzenia specjalne. Mechanizm analizuje w czasie rzeczywistym ostatnio zarejestrowane lokalizacje użytkowników, a następnie na ich podstawie oraz na podstawie wcześniej przypisanych sobie reguł, decyduje, czy mają zostać podjęte jakieś działania. Jeżeli tak, komunikuje się z urządzeniem, aby ustawić mu wyznaczone parametry. Wywołana metoda zwraca zmienną typu *Boolean*, określającą, czy została podjęta decyzja o komunikacji z urządzeniem.
- Co określony interwał czasowy - mechanizm, który co jakiś określony czas oblicza parametry, jakie należy przekazać do urządzenia. Robi to na podstawie zarejestrowanych przez ten czas pozycji użytkowników. Do swoich obliczeń wykorzystuje wagi użytkowników, przypisane im w panelu administracyjnym. Mechanizm czasowy ma niższy priorytet niż mechanizm eventów, dlatego jeżeli klasa obsługująca zdarzenia przekazała urządzeniu parametry, system czasowy zostaje zawieszony aż do kolejnego przejścia pętli. Takie rozwiązanie zapobiega nadpisywaniu parametrów wysłanych do urządzenia, zanim wcześniejsze zostaną wykorzystane.

Informacje na temat urządzeń przechowywane są w bazie danych. Danymi, które są potrzebne do sterowania urządzeniem, niezależnie od jego typu, są:

- jego lokalizacja (określona przez 3 współrzędne)
- adres ip urządzenia oraz port, na którym nasłuchuje
- nazwa rozpoznawalna przez użytkownika (np 'żarówka na korytarzu')

- sterownik określający sposób komunikacji, implementujący interfejs przypisany do konkretnego typu urządzenia
- informacja o przypisanym do urządzenia module sterowania eventami (np natychmiastowa zmiana siły oświetlenia, spowodowana zbliżeniem się określonego użytkownika)
- flaga określająca, czy dane urządzenie jest aktywne
- zasięg działania urządzenia

5.4.2.1. Dobór parametrów sterujących

Dla każdego aktywnego urządzenia zapisanego w systemie, uruchomiony jest na serwerze osobny wątek sterujący. Taki sposób pracy został przyjęty, aby sterowanie i obsługa eventów odbywała się płynnie i w równy sposób dla wszystkich urządzeń, a błąd czy problemy komunikacyjne jednego z urządzeń nie miały wpływu na inne. Do każdego wątku sterującego przypisany jest obiekt klasy zawierającej wszystkie potrzebne informacje oraz metody, aby kontrolować dany typ urządzenia (dla oświetlenia jest to klasa *LightDeviceControllingThread*).

Praca wątku polega na wywołaniu metody *StartControll()*, na której ciało składa się nieskończona pętla *while*. Niezależnie od typu urządzenia, którym zarządza wątek, każda klasa posiada obiekt kolejki, do której kontroler przyjmujący dane od urządzeń mobilnych wstawia obliczone lokalizacje (pod warunkiem, że dla urządzenia sterowanego przypisano klasę obsługującą zdarzenia).

Kolejnym krokiem, jaki wykonywany jest w metodzie, jest pobranie z kolejki wszystkich oczekujących lokalizacji, a następnie przekazanie ich do klasy sterującej eventami. W systemie zaimplementowana jest przykładowa klasa sterująca światłem - *ImportantUserFirstContr*. Wybiera z listy tych użytkowników, którzy są w zasięgu źródła światła, a następnie wyszukuje wśród nich użytkowników o najwyższej wadze. Jeżeli taki użytkownik zostanie znaleziony, system wysyła do urządzenia rozkaz ustawienia światła o największej mocy. Taki poziom zostanie utrzymany aż ostatni użytkownik uprzywilejowany nie oddali się od źródła światła. System poczeka wtedy dodatkowe 4 sekundy, a następnie wyśle do urządzenia informację o ustawieniu mocy światła na poprzednią wartość. System pozwala na rozszerzanie klas sterujących oraz tworzenie swoich i przypisywanie ich do urządzeń (o ile odpowiedni wpis zostanie dodany do bazy danych).

Jeżeli moduł zarządzania eventami zdecyduje o nie wysłaniu do urządzenia parametrów sterujących, wykonywany jest moduł zarządzania czasowego. Przed wykonaniem obliczeń, algorytm sprawdza, czy od ostatniej czasowej aktualizacji minął odpowiedni okres czasu (domyślnie algorytm ma się wykonywać co godzinę). Data poprzedniej aktualizacji przechowywana jest w polu *LastUpdate*, które jest aktualizowane obecną datą za każdym razem, gdy algorytm pozytywnie wyliczy parametry dla urządzenia. Jeżeli odpowiedni czas minął, program pobiera z bazy wszystkie pozycje użytkowników, które zostały zarejestrowane w okresie od ostatniej aktualizacji. Następnie, algorytm wybiera z pobranej listy użytkowników, którzy zostali zarejestrowani w zasięgu urządzenia, a następnie sumuje ich wagi. Parametr,

jaki ma zostać wysłany do urządzenia jest liczony na podstawie wzoru:

$$P_{ustalona} = (P_{max} - P_{min}) * \frac{N_{blisk}}{N_{og}} + P_{min} \quad (5.2)$$

gdzie zmienne w równaniu oznaczają:

- $P_{ustalona}$ - moc światła jaka ma być wysłana do urządzenia
- P_{min} - minimalna moc światła ustawiona dla urządzenia
- P_{max} - maksymalna moc światła ustawiona dla urządzenia
- N_{blisk} - suma wag użytkowników, którzy zostali zarejestrowani w zasięgu urządzenia
- N_{og} - suma wag wszystkich użytkowników w danym okresie czasowym

Wysyłany parametr jest ustawiany dodatkowo w zmiennej *PreviousStaticPowerLevel*, aby mógł być potem, w ramach potrzeby, wykorzystany przez moduł sterujący eventami. Na końcu, do zmiennej przechowującej datę ostatniej aktualizacji zostaje przypisana obecna data i godzina.

Jeżeli parametry urządzenia zostały zaktualizowane w obecnym przebiegu algorytmu, ponowne wykonywanie algorytmu rozpoczyna się od razu. Jeżeli nie, wątek zostaje uśpiony na 0,1 sekundy, aby nie zużywać niepotrzebnie zasobów serwera.

5.4.2.2. Komunikacja z urządzeniami

Sposób komunikacji serwera z urządzeniem nie jest zapisany bezpośrednio w kodzie serwera, ale jest przechowywany w bazie danych i przypisywany indywidualnie do konkretnego urządzenia. Może zostać zmieniony przy użyciu panelu administracyjnego. Dla każdego typu urządzenia może być zdefiniowana pewna pula modułów komunikacji z urządzeniami. Każdy typ ma określony interfejs, po którym ma dziedziczyć klasa komunikująca się z tego typu urządzeniami. Wewnątrz modułu komunikacyjnego określony jest format wysyłanej wiadomości, kodowanie oraz kolejność parametrów. Ilość metod, które ma udostępniać klasa implementująca interfejs jest zależna od ilości parametrów, jakie mogą być ustawiane w danym typie urządzeń.

Rys. 5.2. Model komunikacji z urządzeniami oświetleniowymi



W systemie został stworzony interfejs *ILightDeviceInterface*. Powinien być implementowany przez moduły do komunikacji z urządzeniami odpowiedzialnymi za oświetlenie. Z racji tego, iż światło posiada tylko jeden sterowalny parametr - moc światła, interfejs narzuca na klasie zaimplementowanie metody *NotifyInformationToDevice*, która jako parametry przyjmuje ip urządzenia, jego port oraz wartość mocy światła, jaka ma być wysłana do urządzenia. Zadaniem implementacji tej metody jest sformatowanie wiadomości, zawierającej ustawianą moc światła, według standardu urządzenia oraz wysłanie jej protokołem obsługiwanym przez urządzenie. Metoda ma zwrócić zmienną typu *Boolean*, informującą o tym, czy komunikacja i wysłanie wiadomości się powiodły.

W ramach przykładu, stworzony został moduł do komunikacji z urządzeniami oświetleniowymi, *LoggerLightDeviceInterface*. Wysyła on protokołem HTTP na podany adres ip wiadomość w formacie '*Należy ustawić wartość na: P_{swiatla}* ', gdzie P_{swiatla} to moc światła, którą chcemy ustawić.

5.4.3. Konfiguracja systemu

Aplikacja serwerowa pozwala na konfigurację elementów systemu bez potrzeby jego restartowania. Serwer wykorzystuje do tego kontroler *ConfController*, przyjmujący żądania HTTP. Podstawowymi elementami, które mogą być konfigurowane w taki sposób są:

- wielkość obszaru operacyjnego - wartość określająca, jak duży jest obszar, na którym zainstalowany jest system. Wewnątrz tego pola muszą znajdować się wszystkie obsługiwane routery i urządzenia Bluetooth, które nie zmieniają swojej pozycji (np *Beacony*). Jednostką, w jakiej zapisana jest wielkość mapy, są metry. Wartość ta wykorzystywana jest podczas wyświetlania lokalizacji użytkowników w aplikacji administracyjnej - współrzędna lokalizacji zostaje przeskalowana przy użyciu wielkości pola w taki sposób, aby realistycznie odwzorowywała położenie na wyświetlanej mapie.
- dane o routerach WiFi i nieporuszających się urządzeniach Bluetooth - administrator systemu ma możliwość zmiany kluczowych danych na temat urządzeń, na podstawie których lokalizowani są potem użytkownicy. Poza danymi technicznymi, dotyczącymi specyfikacji urządzenia (siła nadajnika, siła anten), zdefiniowane są również:
 - typ sygnału wysyłanego przez urządzenie - WiFi albo Bluetooth
 - nazwa, po której urządzenie jest rozpoznawalne przez aplikację mobilną - SSID w przypadku routera WiFi, MAC w przypadku urządzenia Bluetooth
 - lokalizacja urządzenia - wymagane jest podanie lokalizacji w trzech wymiarach. Współrzędne urządzenia, jak wszystkie współrzędne w systemie, są podawane stosunku do punktu określonego jako początek pola działania systemu.
 - waga urządzenia, która uwzględniana jest podczas lokalizowania użytkownika. Wartość ta podczas dodawania nowego urządzenia, inicjalizowana jest wartościami domyślnymi, które

potem można zmienić. Dopuszczalna wartość wagi jest liczbą całkowitą z zakresu $< 1 : 4 >$. Domyślne wartości to:

- * dla urządzeń Bluetooth, dynamicznie zmieniających swoją pozycję (użytkownicy aplikacji mobilnej) - 1
- * dla urządzeń Bluetooth, których lokalizacja jest określona - 2
- * dla urządzeń WiFi - 3

Dane na temat routerów przechowywane są w bazie danych, skąd są pobierane podczas startu aplikacji serwerowej. Informacje te są również przechowywane lokalnie, w pamięci serwera. Ma to na celu zmniejszenie czasu wykonywania operacji, ponieważ pobieranie danych z bazy danych jest dużo wolniejsze niż korzystanie ze zmiennych zapisanych w pamięci. Dlatego, każdorazowa zmiana informacji o routerach powoduje zmianę danych przechowywanych lokalnie przez serwer. System zakłada, że zmiany w danych o routerach dokonywane są tylko przy użyciu aplikacji serwerowej - zmiany dokonane na bazie przez zewnętrzne aplikacje (np GUI bazy danych) nie zostaną uwzględnione przez serwer aż do jego restartu.

- Dane na temat urządzeń sterowanych - podobnie jak w przypadku urządzeń do lokalizacji użytkowników, sterowane urządzenia mogą być dodawane, edytowane oraz usuwane przez administratorów systemu. Informacje różnią się w zależności od kategorii, do której należy dane urządzenie, jednak część z nich jest wspólna:
 - nazwa - nazwa urządzenia, która ma pozwolić na jego identyfikację przez użytkowników systemu
 - ip i port - podstawowe dane wykorzystywane do komunikacji z urządzeniem
 - lokalizacja - położenie urządzenia określone przez trzy współrzędne
 - moduł eventów - nazwa modułu, który ma obsługiwać wydarzenia związane z danym urządzeniem. Do urządzenia może nie być przypisany żaden moduł eventów
 - moduł komunikacyjny - nazwa modułu, który ma pozwolić na komunikację serwera z urządzeniem
 - czy moduł jest aktywny - informacja ta jest wykorzystywana podczas sterowania urządzeniem

Informacje na temat sterowanych urządzeń, poza wpisami w bazie danych, przechowywane są również w pamięci serwera. W przeciwieństwie do routerów (których dane przechowywane są statycznie w klasie *SystemDataKnowledge*), dane na temat urządzeń znajdują się w przypisanych im obiektach sterujących. Jest to związane z tym, że wątki sterujące są jedynymi elementami aplikacji serwerowej, które często korzystają z tej informacji.

Z racji tego, iż z każdym sterowanym urządzeniem, zarządzanym przez system, związany jest

obiekt sterujący, kontroler konfiguracyjny odpowiedzialny jest również za edycję wątków sterujących. W przypadku dodawania do systemu nowego sterowanego urządzenia, kontroler inicjalizuje nowy obiekt sterujący oraz startuje związany z nim wątek. W przypadku usuwania urządzenia, kontroler zatrzymuje wątek sterujący i usuwa jego obiekt z listy obiektów sterujących.

- Dane na temat użytkowników systemu - kontroler konfiguracyjny pozwala na edycję wagi użytkowników oraz ich nazwy. System rozpoznaje użytkowników po adresie MAC - zmiana nazwy użytkownika ma na celu jedynie pomóc osobie korzystającej z systemu łatwiejsze rozpoznawanie użytkowników. Waga używana jest przy sterowaniu urządzeniami - użytkownik o wadze 4 (przykładowo szef) jest ważniejszy z punktu widzenia modułu określającego parametry urządzeń, niż osoba o wadze 1 (zwykły pracownik). Dodatkowo, waga może mieć wpływ na moduły obsługi zdarzeń, przypisane do urządzenia - natężenie światła może uzyskiwać wartość maksymalną w przypadku pojawienia się w okolicy użytkownika o wadze 4.

Dane o użytkownikach przechowywane są w bazie - nie ma potrzeby tworzenia lokalnej kopii.

5.4.4. Zarządzanie pozostałymi funkcjami aplikacji administracyjnej

Do obsługi pozostałych, poza konfiguracją, żądań aplikacji klienckiej, służy kontroler *ClientController*. Zawiera on metody pozwalające na:

- logowanie i wylogowywanie się użytkownika - użytkownik w momencie włączania aplikacji zmuszony jest do zalogowania się. Na serwer zostaje wysłane żądanie zawierające login i hasło. Następnie wysłany login zostaje powiązany z adresem ip, z którego przyszło żądanie. Jest to robione w celu umożliwienia późniejszego wysyłania informacji do klienta - wykorzystywane w momencie, w którym użytkownik zadeklaruje chęć monitorowania użytkowników mobilnych.
- pobranie informacji o sterowanych urządzeniach zarejestrowanych w systemie - lista urządzeń jest pobierana w celu wyświetlenia na mapie. Dodatkowo, pobierane są wszystkie przyrządy bez względu na ich kategorię. Zwracanymi informacjami są lokalizacja urządzenia (w celu wyświetlenia w odpowiednim miejscu) oraz jego kategoria (aby aplikacja kliencka mogła dobrać odpowiednią ikonę).
- zgłoszenie żądania zapisania użytkownika do subskrypcji lokalizacji użytkowników - system zostaje poinformowany, że dany administrator chce monitorować wszystkich pojawiających się użytkowników aplikacji mobilnej. W momencie określenia lokalizacji użytkownika, do wszystkich użytkowników, na adres ip pobrany podczas logowania, zostaje wysłana notyfikacja zawierająca współrzędne użytkownika oraz jego adres MAC.
- pobranie lokalizacji użytkowników z danego okresu czasowego - administrator zgłasza chęć pobrania wszystkich lokalizacji, które zostały zarejestrowane w podanym przez niego okresie czasu. W odpowiedzi, do osoby żądającej wysłana zostaje lista zawierająca współrzędne lokalizacji użytkowników oraz ich adresy MAC.

5.5. Aplikacja administracyjna

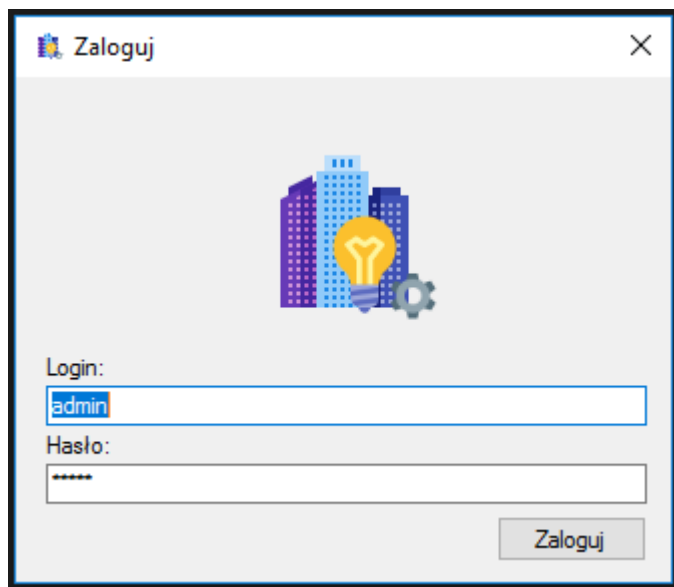
Głównym celem aplikacji klienckiej jest konfiguracja systemu i wyświetlanie oraz monitorowanie lokalizacji użytkowników. Dostęp do niej powinien być ograniczony, ze względów bezpieczeństwa, wyłącznie do uprzywilejowanych użytkowników.

5.5.1. Opis okien aplikacji

5.5.1.1. Okno logowania

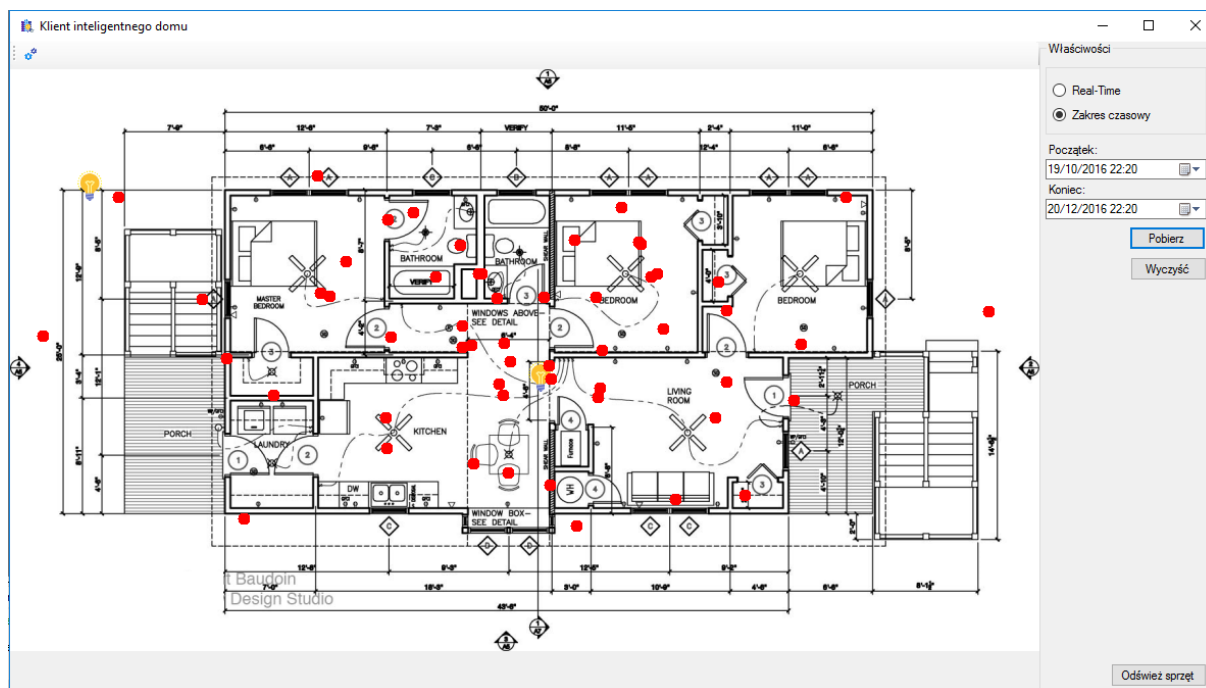
Pierwszym oknem, które pojawia się po włączeniu aplikacji, jest okno logowania. Osoba upoważniona wprowadza swój login oraz hasło. Naciśnięcie przycisku 'Zaloguj' powoduje wysłanie zapytania do aplikacji serwerowej, która tworzy lokalny wpis o logowaniu się użytkownika oraz przechowuje adres IP, z którego przyszło żądanie. Zapisany adres wykorzystywany jest m.in. do wysyłania powiadomień o pojawieniu się lokalizacji mobilnej w sytuacji, gdy użytkownik zapisze się na subskrypcję notyfikacji.

Rys. 5.3. Okno logowania użytkownika administracyjnego



Po pozytywnym przejściu procesu logowania, użytkownik zostaje przeniesiony do głównego okna aplikacji. Składa się ono z panelu z mapą, panelu wyboru trybu wyświetlania lokalizacji użytkowników oraz przycisku przejścia do konfiguracji.

Rys. 5.4. Główne okno aplikacji administracyjnej



5.5.1.2. Główne okno

W głównym panelu wyświetlana jest mapa systemu, do której ścieżka określona jest w pliku konfiguracyjnym 'conf.xml'. Jedynym wymaganiem co do mapy jest to, aby była ona plikiem graficznym. Mapa jest skalowalna - każda zmiana wielkości okna powoduje dopasowanie się obrazka do nowego panelu. To samo dotyczy wyświetlanych wewnątrz elementów. Na mapę nanoszone są urządzenia sterowane oraz lokalizacje użytkowników mobilnych. Urządzenia oznaczane są symbolem ich kategorii, zaś użytkownicy reprezentowani są przez czerwone kropki.

Lokalizacje urządzeń pobierane są podczas załadowania aplikacji oraz automatycznie wyświetlane. Z racji tego, iż zmiany w konfiguracji urządzeń przez innych użytkowników administracyjnych nie odświeżają ikon na mapie, w prawym dolnym rogu aplikacji znajduje się przycisk, którego naciśnięcie spowoduje ponowne pobranie z serwera lokalizacji sprzętów sterowanych oraz przerysowanie symboli na mapie.

Lokalizacje użytkowników mobilnych mogą być wyświetlane w dwóch trybach:

- w czasie rzeczywistym - aplikacja administracyjna zgłasza na serwerze chęć zapisania się na subskrypcję lokalizacji. Dzięki temu, za każdym razem, jak serwer ustali lokalizację użytkownika, zostaną o tym powiadomione wszystkie zalogowane aplikacje administracyjne. Punkty oznaczające użytkownika wyświetlane są na okres 4 sekund.

- w formie skumulowanej - administrator wysyła na serwer zakres czasowy, dla którego chce wyświetlić pozycje użytkowników. Aplikacja serwerowa pobiera lokalizacje z bazy, a następnie wysyła je z powrotem do klienta, na którym zostają wyświetlone. Punkty oznaczające użytkownika wyświetlane są bez określonego okresu ważności.

Przejsie pomiędzy trybami powoduje usunięcie zachowanych w pamięci lokalizacji.

Punkty reprezentujące użytkowników oraz ikony sprzętów sterowanych są skalowane, aby realistycznie odwzorowywać swoją rzeczywistą pozycję. Wykorzystuje się do tego podane podczas konfiguracji wielkości rzeczywistego systemu.

Jeżeli stosunek wysokości mapy do jej szerokości jest większy niż stosunek wysokości panelu wyświetlającego mapę do jego szerokości, pozycję ikony w panelu oblicza się na podstawie wzoru:

$$\begin{cases} x = \frac{rzecz_lok_urz_x}{rzecz_szer_mapy} * szer_pan_mapy \\ y = \frac{(rzecz_lok_urz_y}{rzecz_wys_mapy} * wys_przel_mapy + wys_marg \end{cases} \quad (5.3)$$

gdzie:

- x i y oznaczają współrzędne w panelu mapy
- $rzecz_lok_urze_x$ i $rzecz_lok_urze_y$ to fizyczne współrzędne użytkownika mobilnego lub urządzenia sterowanego
- $rzecz_szer_mapy$ i $rzecz_wys_mapy$ to rzeczywiste wymiary prostokąta, wewnątrz którego znajduje się system
- $szer_pan_mapy$ jest to szerokość panelu wyświetlającego mapę (w tym wypadku szerokość mapy i szerokość panelu jest taka sama)
- wys_przel_mapy to wysokość mapy. W tym wypadku wysokość mapy nie jest równa wysokości panelu - aby zachować realistyczne proporcje, powyżej i poniżej obrazka mapy dodany jest margines. Wysokość mapy obliczana jest ze wzoru:

$$wys_przel_mapy = \frac{szer_pan_mapy * wys_obraz_mapy}{szer_obraz_mapy} \quad (5.4)$$

w którym wys_obraz_mapy i $szer_obraz_mapy$ to wymiary obrazka mapy, określone w pikselach.

- wys_marg to wysokość marginesu, jaki musi zostać dodany, aby zachować proporcję mapy. Obliczony jest ze wzoru:

$$wys_marg = \frac{wys_pan_mapy - wys_przel_mapy}{2} \quad (5.5)$$

W przypadku, jeżeli stosunek wysokości mapy do jej szerokości jest niższy niż stosunek wysokości panelu wyświetlającego mapę do jego szerokości, obliczenia wykonuje się analogicznie, ale z tą różnicą, że w tym wypadku to wysokość mapy będzie równa wysokości panelu i dlatego to faktyczna szerokość mapy w oknie oraz szerokość bocznych marginesów muszą zostać wyliczone.

5.5.1.3. Okno konfiguracji

Wciśnięcie przycisku 'Konfiguracja' (reprezentowane w oknie jako ikona dwóch trybów), otwiera osobne okno, pozwalające na konfigurację kluczowych elementów systemu.

Pierwszy panel w oknie konfiguracji pozwala na edycję fizycznej wielkości systemu oraz danych na temat routerów (umożliwia również ich dodawanie oraz usuwanie). W dolnej części okna wyświetlone są wszystkie zarejestrowane routery. Wciśnięcie wybranego z nich powoduje przejście do trybu edycji - pola edycji routerów zostaną wypełnione danymi wybranego urządzenia.

Rys. 5.5. Panel edycji danych routerów

	Id	SSID	TrasmitterPowe	AntennaGain	LocationX	LocationY	LocationZ	RouterCategory	Weight
▶	2	TP-LINK_ACP	16	2	80	20	0	1	3
	3	AndroidAP	10	2	50	80	0	1	3
	1	TP-LINK_7C...	16,5	5	10	50	0	1	4

Poza polami wymaganymi do lokalizacji użytkownika, którymi są:

- lokalizacja routera - współrzędne X, Y, Z oraz SSID
- zysk anteny
- siła transmitera

aplikacja pozwala dodatkowo na zadeklarowanie typu sygnału wysyłanego przez urządzenie. Ma to wpływ na sposób obliczania odległości do użytkowników, ale także na wartość początkową wagi przydzielonej temu transmitterowi. Wagę można zmienić ręcznie, ale trzeba pamiętać, że musi się znaleźć w zakresie <1:4>.

Drugi panel pozwala na definiowanie i edycję urządzeń sterowanych.

Rys. 5.6. Panel edycji urządzeń sterowanych

Wszystkie urządzenia zarejestrowane w systemie podzielone są na kategorie. Dostępne kategorie przechowywane są w bazie danych i stamtąd są pobierane, kiedy aplikacja kliencka wyśle żądanie na serwer. Wybranie kategorii z lewego panelu powoduje pobranie z serwera wszystkich urządzeń, które należą do wybranej kategorii. Do każdej kategorii przydzielona jest inna akcja w kontrolerze *ConfController* na serwerze - przykładowo, dla urządzeń oświetleniowych, jest to akcja *GetLightDevices*.

Aby przejść do widoku wyświetlania i edycji danych urządzenia, należy wybrać interesującą nas pozycję ze środkowego panelu. Powoduje to pojawienie się w prawym panelu formularza z danymi danego urządzenia. Formularze są tworzone dynamicznie, a rodzaj i ilość ich pól zależne są od kategorii urządzenia. Dla urządzenia z kategorii 'Światło', dostępne są następujące pola:

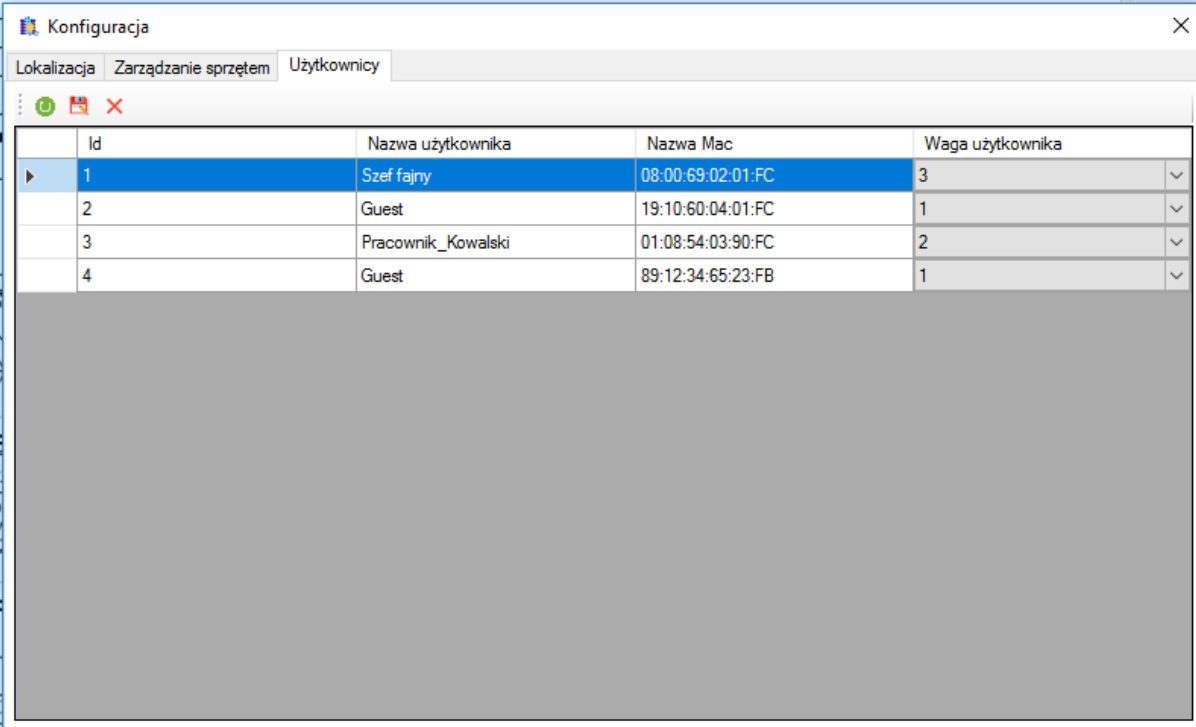
- lokalizacji - współrzędne XYZ urządzenia, pozwalające umiejscowić je w rzeczywistym środowisku
- adres ip i port - pozwalają na komunikację z urządzeniem
- nazwa - pozwala użytkownikom odróżniać urządzenia między sobą
- minimalne i maksymalne oświetlenie - zakres, określony w procentach, z którego może korzystać system podczas sterowania urządzeniem. W przypadku sterowania w interwałach czasowych, minimalne oświetlenie przydzielane jest w sytuacji, gdy, w podanym okresie czasu, w zasięgu urządzenia nie pojawił się żaden użytkownik. Analogicznie, maksymalne oświetlenie przydzielane jest, gdy wszystkie obliczone lokalizacje użytkowników znajdowały się w w zasięgu sterowanego urządzenia

- moduł komunikacyjny - określa sterownik, sposób komunikacji serwera z urządzeniem. Moduły komunikacyjne definiowane są w bazie danych, a związane z nimi klasy obsługujące muszą zostać dodane do kodu serwera
- obsługa zdarzeń - nazwa modułu, który na bieżąco steruje urządzeniem, w zależności od położenia użytkowników zlokalizowanych w zasięgu. Z modułem związana jest klasa, która obsługuje przypisane do urządzenia zdarzenia
- aktywny - informacja o tym, czy urządzenie jest aktywne
- zasięg - maksymalna odległość, w jakiej musi znajdować się użytkownik, aby był uwzględniany podczas ustalania parametrów

Panel udostępnia przyciski, które pozwalają na zapis wprowadzonych zmian, jak również dodanie lub usunięcie urządzenia.

Ostatni panel to panel użytkowników. Są to osoby, które korzystały z aplikacji mobilnej, a ich pozycja została chociaż raz obliczona.

Rys. 5.7. Panel edycji urządzeń sterowanych



	Id	Nazwa użytkownika	Nazwa Mac	Waga użytkownika
▶	1	Szef fajny	08:00:69:02:01:FC	3
	2	Guest	19:10:60:04:01:FC	1
	3	Pracownik_Kowalski	01:08:54:03:90:FC	2
	4	Guest	89:12:34:65:23:FB	1

Okno składa się z tabeli, w której znajdują się zapisani użytkownicy oraz przycisków:

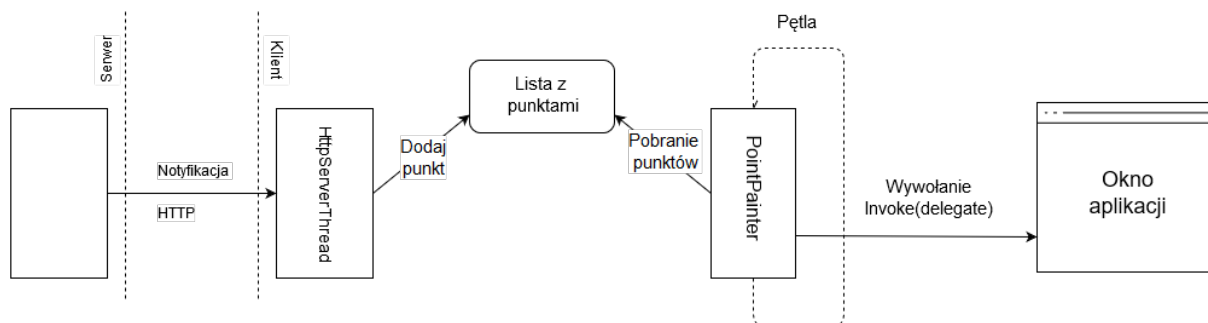
- odświeżania danych
- zapisania wszystkich użytkowników
- usunięcia wybranego w tabeli użytkownika

Widok pozwala na podglądnięcie użytkowników oraz na zmianę ich nazwy oraz wagi. Nazwa pozwala na rozróżnianie użytkowników w sposób łatwiejszy niż adres MAC. Waga odgrywa kluczową rolę podczas obliczania parametrów do sterowania urządzeniami - czym wyższa waga użytkownika, tym większe ma on znaczenie dla urządzenia, w którego zasięgu znajduje się obliczona lokalizacja. Dodatkowo, pojawienie się użytkownika o określonej wadze może mieć wpływ na zajście zdarzenia, zdefiniowanego dla urządzenia. Wagi przyjmują wartość od 0 (użytkownik o najniższym priorytecie, nie brany pod uwagę podczas określania parametrów) do 4 (użytkownik o najwyższym priorytecie).

5.5.2. Wyświetlanie lokalizacji użytkowników

W trybie wyświetlania lokalizacji w czasie rzeczywistym, aplikacja musi wyświetlać uzyskane dane bez odświeżania panelu czy ingerencji użytkownika. W związku z tym potrzebne było wprowadzenie mechanizmu opartego na osobnym wątku. Klasą zarządzającą rysowanie punktów jest *PointPainter*. W wątku wykonuje się metoda tej klasy, *DrawPoint()*, zawierająca nieskończoną pętlę, usypianą na 0,5 sekundy przy każdym przejściu. Wątek ma dostęp do listy punktów, które powinien wyświetlić. Lista uzupełniania jest przez obiekt klasy *HttpServerThread*, na podstawie notyfikacji przychodzących z serwera. Każdy wpis na liście ma informacje o lokalizacji (współrzędne XY) oraz dacie ważności danego punktu. Punkty stworzone na podstawie notyfikacji z serwera mają okres ważności równy 4 sec, za to punkty wyświetlane w sposób skumulowany mają datę ważności ustawioną na wartość null (co oznacza, że są wyświetlane dopóki nie zostanie wywołana komenda wyczyszczenia mapy).

Rys. 5.8. Schemat rysowania elementów systemu na mapie



Metoda wyświetlająca punkty nie rysuje ich bezpośrednio na wyświetlonej mapie, ponieważ odświeżanie wymagałoby ponownego przerysowania nie tylko punktów, ale i całej mapy. Mogłoby to powodować opóźnienia w płynności wyświetlania, a wiązałoby się z tym nieprzyjemne dla oka 'mrukanie' mapy. Dlatego, punkty nanoszone są na przeźroczysty obrazek, który później wyświetlany jest na warstwie wyświetlającej punkty. Zanim jednak na obrazku zostaną naniesione lokalizacje użytkowników, klasa *PointPainter* zaznacza na przeźroczystej warstwie urządzenia sterowane, rysując odpowiednią ikonę kategorii danego sprzętu. Dodatkowo, ponieważ metoda rysująca punkty działa na innym wątku niż panel zawierający mapę, wyświetlenie punktów nie może być wywołane bezpośrednio z poziomu

klasy *PointPainter*. Z tego powodu, funkcja rysująca obraz z punktami i ikonami urządzeń na mapie zostaje opakowana w delegatę (typ danych przechowujący funkcję), a następnie referencja do niej zostaje wstawiona jako parametr funkcji *Control.Invoke(delegate)*, dzięki czemu panel sam wywoła przekazaną mu jako parametr funkcję podczas kolejnego wykonywania się jego wątku.

5.6. Aplikacja mobilna

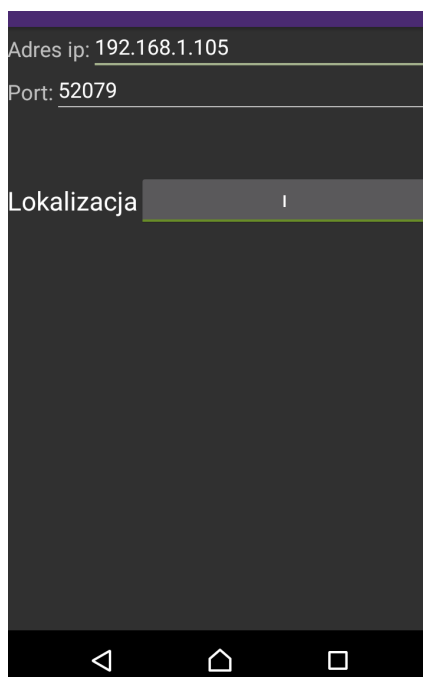
Celem aplikacji mobilnej jest pobieranie siły sygnałów Bluetooth i Wi-Fi. Wykorzystuje do tego klasę *BroadcastReceiver*. Za każdym razem, gdy *BroadcastReceiver* wykryje siły sygnałów Bluetooth i Wi-Fi, wywoływana jest metoda *OnReceive*.

W przypadku wykrywania sygnałów Wi-Fi, rezultat skanowania pobierany jest jako lista obiektów klasy *ScanResult*. Lista ta znajduje się we właściwości *ScanResults* obiektu klasy *WifiManager*, pobranego przy użyciu metody *GetSystemService()*.

W przypadku Bluetooth, siłę sygnału uzyskuje się poprzez wywołanie *GetShortExtra()* na obiekcie klasy *Intent*, przekazanego jako parametr metody wywoływanej po wykryciu urządzenia Bluetooth. Nazwa MAC urządzenia jest dostępna w polu *name* obiektu klasy *BluetoothDevice*, dostępnemu również dzięki obiektowi klasy *Intent*, przekazanemu jako parametr.

Uzyskane dane, niezależnie od rodzaju sygnału, zostają zebrane w obiekcie klasy *DeviceNotificationModel*, który później zostaje wysłany na serwer w ciele żądania HTTP. Akcja, która obsługuje przyjmowanie danych od aplikacji mobilnych, nazywa się *ReportDevices* i należy do kontrolera *DataCollector*.

Rys. 5.9. Ekran konfiguracyjny aplikacji mobilnej



Aplikacja mobilna składa się z jednego obiektu klasy dziedziczącej po *Activity*, *MainActivity*, oraz jednego obiektu klasy dziedziczącej po *Service*, *LocalizationService*. Na ekranie startowym znajdują się dwa pola tekstowe, które pozwalają na konfigurowanie aplikacji: ip serwera i port. Dodatkowo, okno posiada przycisk, który pozwala na przełączanie stanu obiektu *Service*, odpowiedzialnego na wykrywanie sygnałów Bluetooth i Wi-Fi oraz wysyłanie ich do serwera. Aby wprowadzone w polu konfiguracyjnym zmiany zostały przekazane do obiektu *Service*, należy zrestartować serwis przy użyciu opisanego przycisku.

Oba obiekty *BroadcastReceiver*, które odbierają informację o sygnałach Bluetooth i Wi-Fi, zapisują dane do jednej, wspólnej listy. Informacje te są pobierane przez wystartowany w serwisie wątek, który co 1 sekundę wysyła znajdujące się tam informację na serwer. Wątek przesyła dane tylko wtedy, kiedy lista nie jest pusta (nie musi zawierać informacji o obu typach sygnału - ma to zapobiec blokadzie aplikacji w sytuacji, gdy nie zostaną odebrane żadne sygnały jednego z typów komunikacji). Obiekty odbierające dane o sygnałach muszą dbać o to, aby na liście z danymi nie dublowały się wpisy o tym samym typie sygnału. W sytuacji, gdy tak się stanie, obiekt musi poczekać, aż wątek wysyłający zostanie wybudzony i wyczyści listę.

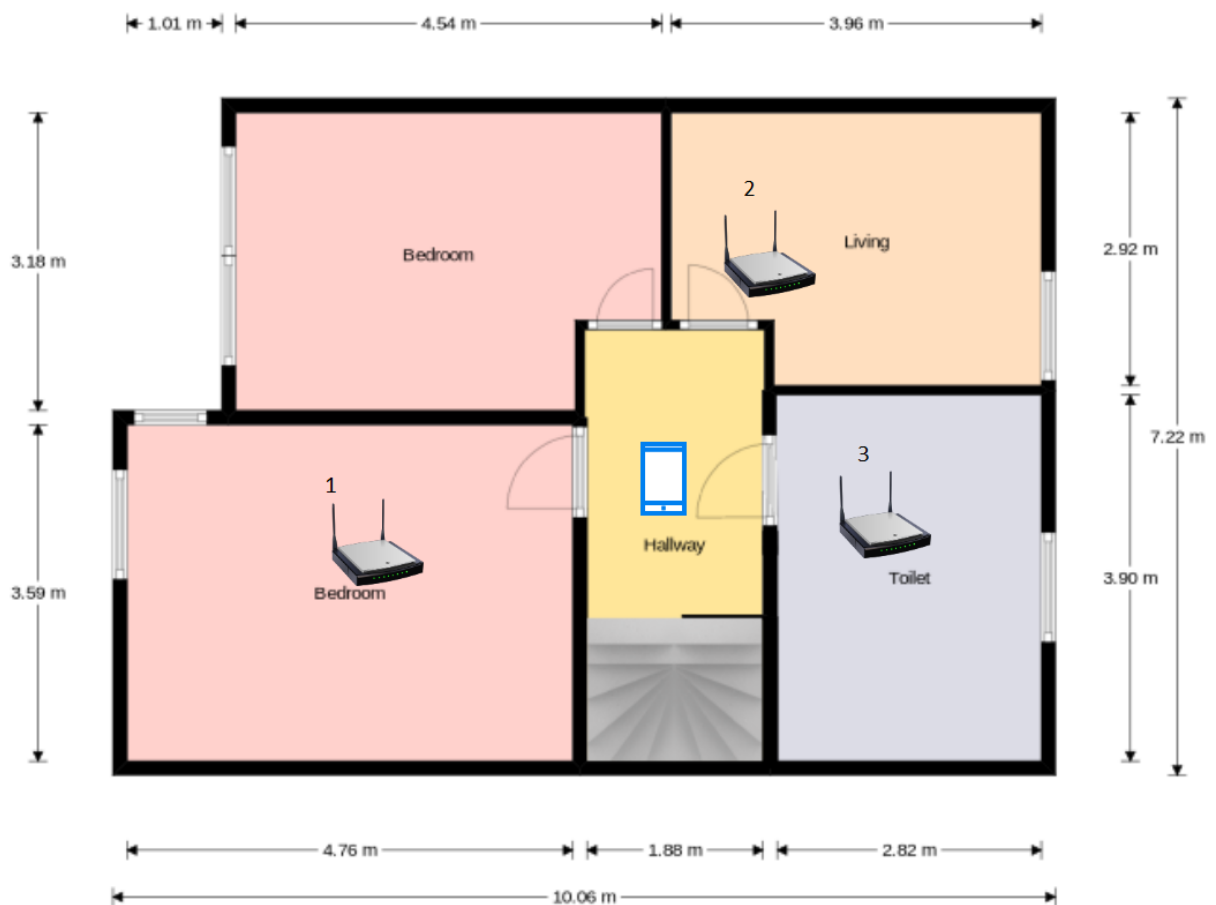
6. Testy systemu

6.1. Lokalizowanie użytkownika

Celem testów jest zbadanie jakości wyznaczonej przez system lokalizacji użytkownika w rzeczywistym środowisku.

Przygotowany został jeden telefon z aplikacją mobilną, który pełnił rolę odbiornika oraz 3 transmitery Wi-Fi (1 router i 2 access-pointy). Obszar działania miał wielkość 10 metrów na 7,2 metra. Jako punkt 'zerowy' (0,0,0) został przyjęty lewy górny róg.

Rys. 6.1. Schemat modelu do testów lokalizacji



Dla tak przyjętych założeń, kolejne urządzenia mają współrzędne:

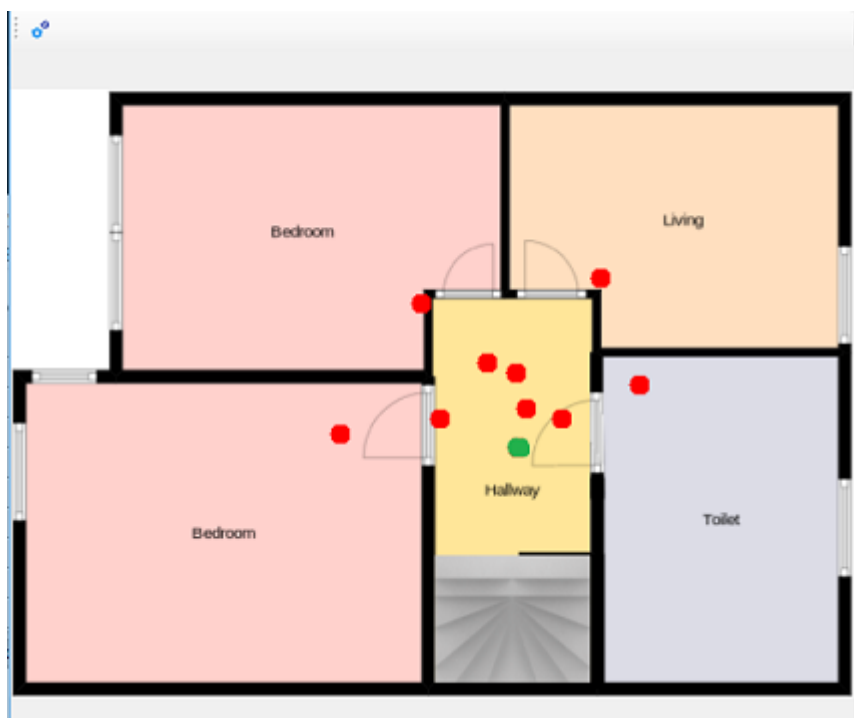
- transponder nr 1 - współrzędne (2.5, 4.75, 0.1)
- transponder nr 2 - współrzędne (7.5, 1.25, 0.0)
- transponder nr 3 - współrzędne (8.25, 4.5, 0.0)
- odbiornik - współrzędne (6.0, 4.25, 0.0)

Wykonana została seria 7 powtórzeń, podczas których aplikacja wysłała informacje o odbieranych przez nią siłach sygnałów, a serwer, po ich odebraniu, wyznaczył lokalizację. Błąd względny był obliczany w odniesieniu do założonego, akceptowalnego błędu podczas lokalizacji - 0.5 metra.

Pomiar	X	Y	Z	Dystans między rzeczywistą pozycją, a obliczoną (m)	Błąd względny (%)
1	6.11	3.78	-0.12	0.5	100
2	5.65	3.24	0.06	1.07	214
3	6.53	3.90	0.51	0.81	162
4	4.87	2.53	-0.1	2.06	412
5	6.98	2.23	-0.04	2.25	450
6	5.09	3.90	0.01	0.98	196
7	7.44	3.50	-0.22	1.64	328

Lokalizacja została również zarejestrowana przez aplikację administracyjną (na zielono zaznaczono faktyczną pozycję użytkownika).

Rys. 6.2. Widok mapy w aplikacji klienckiej



Z eksperymentu wynika, że określanie lokalizacji na podstawie siły sygnału wysyłanego przez transponder ma niższą skuteczność niż oczekiwałem na początku. W ramach rozwijania aplikacji w przyszłości, warto znaleźć lepszy sposób na dokładniejsze określanie odległości odbiornika od transponderów.

6.2. Monitorowanie i konfiguracja systemu

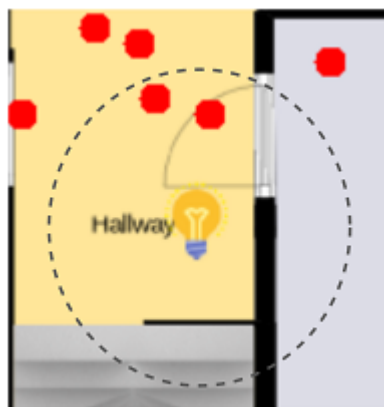
Dodatkowo, przed zarejestrowaniem lokalizacji użytkownika w poprzednim eksperymencie, do systemu zostało dodane jedno urządzenie oświetleniowe. Jego zasięg został ustawiony na 0.75 metra, a lokalizacja na (6.1, 4.4, 0). Urządzeniu został również przypisany moduł obsługi eventów - *ImportantUserFirstContr* - tak, aby światło zostało ustawione na wartość maksymalną w momencie, jak w okolicy zostanie zarejestrowana obecność telefonu z aplikacją mobilną. Użytkownikowi, który brał udział w teście, została nadana najwyższa waga.

W momencie zarejestrowania lokalizacji o współrzędnych (6.11, 3.78, -0.12), odległość od urządzenia wynosiła 0.63 metra, co oznacza, że znalazła się w zasięgu urządzenia. Spowodowało to wysłanie przez serwer do urządzenia parametru o wartości równej maksymalnej sile światła (w tym wypadku wynosiło to 80%).

Dodatkowo, w momencie wywołania się modułu czasowego, jako lokalizację przynależące do urządzenia zostały zakwalifikowane dwie współrzędne, co spowodowało, że serwer wysłał parametr równy 33%, ponieważ:

$$P_{ustalona} = (P_{max} - P_{min}) * \frac{N_{blisk}}{N_{og}} + P_{min} = (80 - 20) * \frac{2}{9} + 20 = 33,3 \quad (6.1)$$

Rys. 6.3. Wizualne przedstawienie zasięgu urządzenia



6.3. Podsumowanie

Celem pracy było stworzenie systemu, który określa położenie użytkownika w trzech wymiarach na podstawie siły sygnału znanych routerów, stałych urządzeń Bluetooth (np. *Beaconów*) oraz dynamicznie zmieniających się sił sygnałów Bluetooth innych użytkowników systemu. Obliczone lokalizacje miały

być przechowywane oraz przedstawiane na mapie budynku w sposób łatwy do analizy. Dodatkowo, celem było opracowanie algorytmu, który na podstawie określonych warunków oraz mapy przepływu zarządza wybranymi urządzeniami podłączonymi do systemu.

W ramach projektu został zaimplementowany pełny moduł do lokalizacji użytkowników. Dane zbierane z aplikacji mobilnych są przedstawiane jako trójwymiarowy model, w której odległość między transponderem, a odbiornikiem została opisana jako probabilistyczna funkcja Gaussa. Mierzenie dystansu pomiędzy urządzeniami opiera się na obliczaniu mocy zaniku siły sygnałów Bluetooth oraz Wifi, odbieranego przez urządzenia mobilne.

Aby system mógł lepiej określać lokalizację aplikacji mobilnych, przeprowadzone zostały testy technologii Bluetooth oraz Wi-Fi. Ich celem było określenie, który z wyżej wymienionych sposobów komunikacji jest bardziej niezawodny oraz odporny na zakłócenia i odbicia. Niestety, ogólne testy całej aplikacji wykazały, że określanie odległości na podstawie *RSSI* nie jest wystarczająco dokładne, jednakże dobór lepszej metody może być uwzględniony podczas dalszego rozwoju aplikacji.

Dodatkowo, stworzona została aplikacja administracyjna, mająca możliwość zarządzania całym systemem oraz monitorowania lokalizacji użytkowników. Prezentowane przez nią dane pozwalają na analizę przepływu ludzi, a dzięki odpowiedniemu skonfigurowaniu, aplikacja serwerowa ma możliwość efektywnego sterowania zarejestrowanymi urządzeniami na podstawie odnotowanych lokalizacji użytkowników.

Aplikacja może być w przyszłości rozwijana w wielu różnych kierunkach. Po pierwsze, testy wykazały, że aby aplikacja mogła dokładnie wyznaczać lokalizację, potrzebne jest znalezienie innego sposobu określania odległości między transponderem, a odbiornikiem. Dodatkowo, do aplikacji można dodawać nowe kategorie urządzeń oraz sposoby komunikacji z nimi. Aplikacja mobilna może zostać rozszerzona o moduł administracyjny, który pozwoli na sterowanie systemem z poziomu smartphona. Wreszcie, serwer może być rozwijany poprzez stworzenie większej ilości modułów obsługi eventów.

Bibliografia

- [1] *Zarządzanie budynkiem BMS, SMS*. Automatyka budynkowa. 2016. URL: <http://www.automatyka-budynkowa.com/produkty/kat/12/> (term. wiz. 2016-12-30).
- [2] Tuan Anh Nguyen i Marco Aiello. „Energy intelligent buildings based on user activity”. W: *Energy and Buildings* 56 (sty. 2013). Wyd. Alan Meier. ISSN: 0378-7788.
- [3] Ian Robertson, Nutapong Somjit i Mitchai Chongcheawchamnan. *Microwave and Millimetre-Wave Design for Wireless Communications*. 1 wyd. Wiley (Verlag), 2016, s. 27. ISBN: 978-1-118-91721-3.
- [4] Maxim Shchekotov. „Indoor Localization Method Based on Wi-Fi Trilateration Technique”. W: 16th FRUCT Conference in Oulu, Finland. ITMO University. 2014.
- [5] Murat Torlak, red. *Path Loss*. EE4367 Telecom. Switching & Transmission. The University of Texas at Dallas, 2008.
- [6] Guoqiang Mao i Barış Fidan. *Localization Algorithms and Strategies for Wireless Sensor Networks*. 1 wyd. Information Science Reference, 2009, s. 137–138. ISBN: 1605663964.
- [7] Maria Isabel Ribeiro. „Gaussian Probability Density Functions: Properties and Error Characterization”. Institute for Systems i Robotics Instituto Superior Tcnico, 2004.
- [8] Jan Kraaier. *Modeling User Mobility for the Simulation of Wireless Ad Hoc Access Networks*. Cuviller Verlag Göttinger, 2008, s. 35–40. ISBN: 3867276900.
- [9] Yaqian Xu. *Autonomous Indoor Localization Using Unsupervised Wi-Fi*. University Press Kassel GmbH, 2016, s. 30–32. ISBN: 3737600708.
- [10] Trung-Kien Dao, Eric Castelli i Hung-Long Nguyen. „Home Appliance Control System Based on Robust Indoor User Localization using Wifi Signals”. W: *MobileHCI '13 15th International Conference on Human-Computer Interaction with Mobile Devices and Services*. ACM New York, 2013. ISBN: 978-1-4503-2273-7.
- [11] Ramsey Faragher i Robert Harle. „An Analysis of the Accuracy of Bluetooth Low Energy for Indoor Positioning Applications”. W: *Proceedings of the 27th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2014)* (wrz. 2014). University of Cambridge, s. 201–210. ISSN: 0028-1522.

- [12] Simon Takens. „Indoor location tracking using Signal Strength Pinpoints”. Master of Science Thesis. Faculty of Mathematics i Natural Science Department of Computer Science, University of Groningen, 2010.
- [13] Małgorzata Langer. *WPROWADZENIE DO TELEKOMUNIKACJI Część II*. Instytut Elektroniki, Politechnika Łódzka. 2016. URL: <http://eletel.p.lodz.pl/mlanger/telek/file-No.2.pdf> (term. wiz. 2016-12-30).
- [14] Peter Joseph Bevelacqua. *Antenna Gain*. 2009. URL: <http://www.antenna-theory.com/basics/gain.php> (term. wiz. 2016-12-30).
- [15] Netcontrol Oy. *Radio Fade Margin*. 2016. URL: <http://www.radius.net/eng/services/radio-networking-tools/radio-fade-margin/> (term. wiz. 2016-12-30).
- [16] Invictus Networks. *Fade Margin / System Operating Margin Calculator*. 2016. URL: <http://www.invictusnetworks.com/faq/RF%20Technical%20Info%20and%20FCC%20Regs/Fade%20Margin%20Calculator%20-%20Basic.htm> (term. wiz. 2016-12-30).
- [17] Anupam Mahajan i Madhur Chanana. „Wi-Fi Localization using RSSI in Indoor Environment via a smartphone”. W: *International Journal Of Engineering And Computer Science* 1 (2 2012). ISSN: 2319-7242.
- [18] *Android Classes: Broadcast Receiver*. 2016. URL: <https://developer.android.com/reference/android/content/BroadcastReceiver.html> (term. wiz. 2016-12-30).
- [19] Andrea Goldsmith. *Signal Propagation and Path Loss Models*. EE 359: Wireless Communications. 2016. URL: <http://web.stanford.edu/class/ee359/pdfs/lecture2.pdf> (term. wiz. 2016-12-30).
- [20] *Distance calculation - related formula*. 2016. URL: <http://www.tplink.com/ie/support/calculator/> (term. wiz. 2016-12-30).
- [21] *Free Space Radio Propagation and Path Loss*. 2016. URL: <https://www.electronics-notes.com/articles/antennas-propagation/propagation-overview/free-space-path-loss.php> (term. wiz. 2016-12-30).
- [22] IDC Technologies. *Free Space Loss*. URL: http://www.idc-online.com/technical_references/pdfs/electronic_engineering/Free_Space_Loss.pdf.
- [23] Ian Griffiths. *Programming C# 5.0*. 1 wyd. O'Reilly Media, 2013. ISBN: 978-1-4493-2041-6.
- [24] Jess Chadwick, Todd Snyder i Hrusikesh Panda. *Programming ASP.NET MVC 4: Developing Real-World Web Application with ASP.NET MVC*. O'Reilly Media, 2012. ISBN: 978-83-246-6644-7.
- [25] Chris Smith. *Programming F#*. Wyed. Laurel Ruma. 2 wyd. O'Reilly Media, 2010. ISBN: 978-0-596-15364-9.

- [26] *Xamarin Platform - C# on Android, iOS, Mac & Windows*. 2016. URL: <https://www.xamarin.com/platform> (term. wiz. 2016-12-30).