Inteligentny system wspomagający przejazd pojazdów ratownictwa medycznego

Zespół Zero

Dąbrowski Michał,

Groen Hubert,

Laskowski Michał,

Zembroń Mateusz

**Repozytorium kodu: https://gitlab-stud.elka.pw.edu.pl/mzembron/aasd-zespol-zero**

ETAP A

**Identyfikacja i opis problemu**

Celem projekt jest zaprojektowanie systemu agentowego służącego do zarządzania sygnalizacją świetlną na skrzyżowaniach, co ma spowodować minimalizację czasu przejazdu karetek pogotowia. Jak wynika z danych Wojewódzkiej Stacji Pogotowia Ratunkowego i Transportu Sanitarnego Meditrans, średnia liczba wyjazdów karetek do pacjentów regularnie rośnie. W 2021 roku było to średnio 629 wyjazdów na dobę, a w roku 2022 już 708. Dodatkowo, według danych firmy TomTom, średni czas przejazdu 10 km w Warszawie wzrósł w roku 2022 względem 2021 z 19 do 20 min, co również wskazuje na wzrost zatłoczenia dróg. Dane te mogą wskazywać, że czas dojazdów karetek do pacjentów będzie wzrastać, co stwarza ryzyko dla zdrowia i życia osób w potrzebie. Utrudnia to również wypełnienie ustawowych wymagań dotyczących czasu dojazdu pojazdów pogotowia ratunkowego, którego mediana w miastach powyżej 10 tys. mieszkańców powinna nie przekraczać 8 min. Warto zaznaczyć, że już w tej chwili, według danych GUS, normy te nie są spełniane, szczególnie w województwie mazowieckim.

**Beneficjenci rozwiązania**

Głównymi beneficjentami rozwiązania są mieszkańcy dużego miasta,, którzy mieliby zapewniony krótszy czas przyjazdu służb ratunkowych, co może rzutować na poziom ich zdrowia i życie. Dodatkowo, beneficjentami są również pracownicy owych służb, ponieważ zmniejszenie czasu dojazdu oraz ograniczenie ich blokady w korkach może mieć pozytywny wpływ na poziom ich stresu i zmęczenia związanych z wykonywaniem obowiązków służbowych.

**Propozycję i sprecyzowanie rozwiązania**

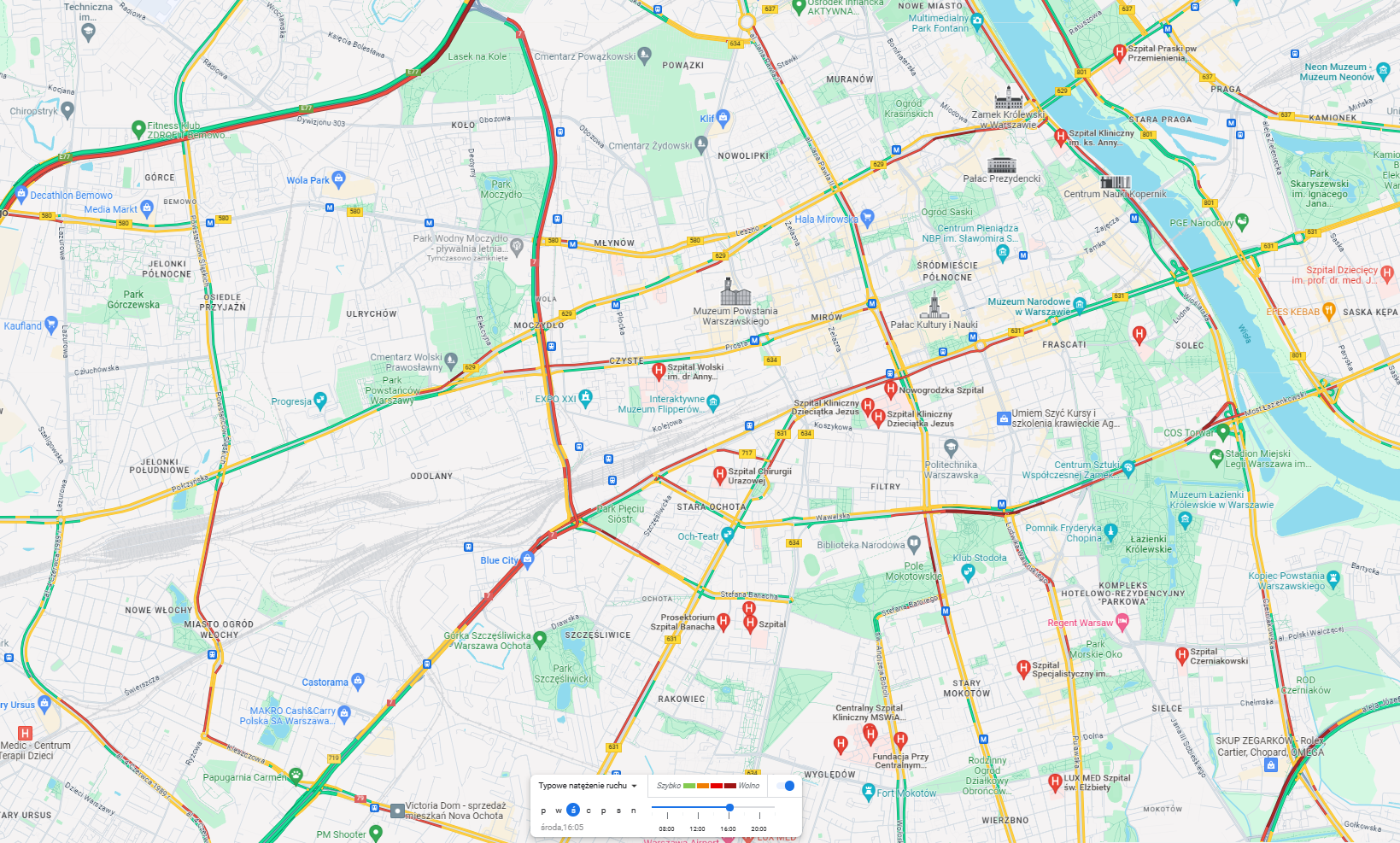
Celem projektu jest stworzenie systemu zarządzania ruchem ulicznym, który zminimalizuje czas dotarcia pojazdów ratownictwa medycznego do celu. Głównym elementem proponowanego rozwiązania jest zarządzanie infrastrukturą uliczną w taki sposób, aby zminimalizować przeszkody na trasie karetki. System powinien być w stanie ,,rozładować” zatłoczone ulice na trasie karetki manipulując sygnalizacją świetlną. Efektem będzie przeniesienie podwyższonego zagęszczenia ruchu z trasy pojazdu uprzywilejowanego na okoliczne ulice, którymi nie będzie się poruszać. Dodatkowo system może utrzymywać pewne korytarze (mniej zatłoczone ulice), które będą zapewniać łatwiejszy dostęp do newralgicznych dzielnic miasta. Taki efekt można osiągnąć ograniczając dostępność pewnych ulic np. rzadkie, krótkotrwałe zielone światło pozwalające dotrzeć do danej ulicy.

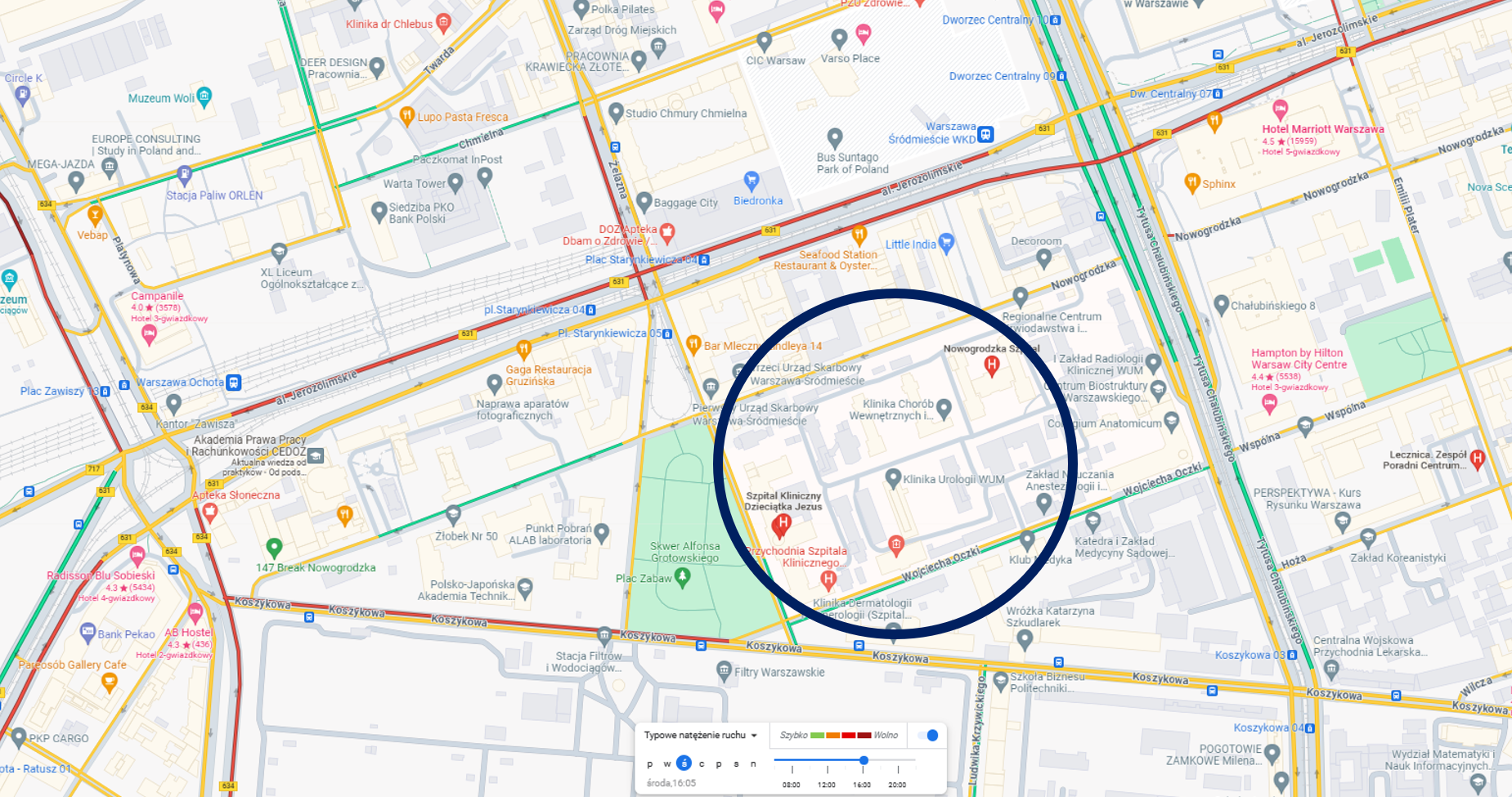
**Analiza źródeł**

Koncepcja systemu zostanie oparta o dane na temat:

* zagęszczenia ruchu w mieście (najczęstsze miejsca korków, średni czas przejazdu),
* najczęstszych miejsc wypadków lub zgłoszeń do służb ratowniczych,
* identyfikacji topografii miasta (drogi szybkiego ruchu, szpitale).

Poniższa mapa Warszawy pokazuje typowe natężenie ruchu w godzinach szczytu oraz punkty ratownictwa medycznego. Takie informacje mogą wskazać potencjalnie korzystne trasy do najszybszego przejazdu karetki oraz wykluczyć skrzyżowania/trasy z największą ilością uczestników ruchu. Dodatkowa ingerencja w sygnalizację świetlną na wybranej trasie przejazdu poprawi przepustowość zgodnie z kierunkiem poruszania się pojazdu uprzywilejowanego.





Rozmieszczenie i zagęszczenie poszczególnych punktów inteligentnego nadzoru sygnalizacji świetlnej może być oparte o raporty Zarządu Dróg miejskich, podających mapę oraz statystyki występowania wypadków w Warszawie:

[Raport o stanie bezpieczeństwa na drogach 2022 - Zarząd Dróg Miejskich w Warszawie](https://zdm.waw.pl/dzialania/badania-i-analizy/raport-brd/raport-o-stanie-bezpieczenstwa-na-drogach-2022/)

W literaturze istnieją już rozwiązania, które adresują problem czasu dojazdu do pacjenta za pośrednictwem sterowania sygnalizacją świetlną. Takie podejście zostało opisane w artykule:

[Adaptive traffic management for secure and efficient emergency services in smart cities | IEEE Conference Publication | IEEE Xplore](https://ieeexplore.ieee.org/abstract/document/6529511)

**Koncepcja systemu**

System będzie składał się z 4 rodzajów elementów:

1. Nadajnik GPS - wysyłają w czasie rzeczywistym informację o bieżącym położeniu karetki do **Systemu Głównego**.
2. Kontroler ruchu - Jest zintegrowany ze sterownikiem świateł na skrzyżowaniu. Potrafi przełączyć aktualne ustawienie świateł na skrzyżowaniu zgodnie z kierunkiem odebranym od **Systemu Głównego**
3. System monitoringu ruchu - jest to system monitorujący aktualne natężenie ruchu na skrzyżowaniach za pomocą kamer. Dane te są wysyłane do **Systemu Głównego.**
4. System Główny - jest to element inteligentny, który na podstawie odczytów z **Nadajników GPS**, **Systemu monitoringu ruchu** oraz **Kontrolerów ruchu** manipuluje światłami na skrzyżowaniach wykorzystując **Kontroler ruchu.** System główny jest jeden na całe miasto.

**Źródła**

* [Warsaw traffic report | TomTom Traffic Index](https://www.tomtom.com/traffic-index/warsaw-traffic/)
* [Raport o stanie bezpieczeństwa na drogach 2022 - Zarząd Dróg Miejskich w Warszawie (zdm.waw.pl)](https://zdm.waw.pl/dzialania/badania-i-analizy/raport-brd/raport-o-stanie-bezpieczenstwa-na-drogach-2022/)
* [Analiza ruchu na drogach 2022 - Zarząd Dróg Miejskich w Warszawie (zdm.waw.pl)](https://zdm.waw.pl/dzialania/badania-i-analizy/analiza-ruchu-na-drogach/analiza-ruchu-na-drogach-2022/)
* <https://warszawa.stat.gov.pl/en/publications/others/via-warsaw-statistics-2022,23,5.html>
* <https://injuryprevention.bmj.com/content/13/1/65.short>
* <https://essuir.sumdu.edu.ua/handle/123456789/77238>
* <https://www.cowzdrowiu.pl/aktualnosci/post/ile-czasu-w-2020-r-czekalismy-na-karetke-dane-gus>
* <https://www.sciencedirect.com/topics/computer-science/emergency-response-time>
* <https://tvn24.pl/tvnwarszawa/najnowsze/warszawa-2022-rok-w-pogotowiu-ratunkowym-nieuzasadnione-wezwania-puste-wyjazdy-liczby-6878240>
* <https://www.ijrar.org/papers/IJRAR1944721.pdf>
* [161889822.pdf (core.ac.uk)](https://core.ac.uk/download/pdf/161889822.pdf)
* <https://ieeexplore.ieee.org/abstract/document/6529511>
* [Demographic statisticsMunicipality of ŚRÓDMIEŚCIE, population density, population, average age, families, foreigners (urbistat.com)](https://ugeo.urbistat.com/AdminStat/en/pl/demografia/dati-sintesi/srodmiescie/1465108/4)

ETAP B

**Wymagania:**

* minimalizacja czasu dotarcia do pacjenta/dojazdu do szpitala  
  (do poziomu wspomnianego w ustawie)
* obsłużenie każdego wezwania
* zdefiniowanie “korytarzy” szybkiego dotarcia do celu  
  (ulic na których *system zarządzania przejazdem* będzie zmieniał światła)
* system powinien brać pod uwagę aktualnie informacje o ruchu drogowym

**Metoda projektowania:** BPMN

**Role w systemie:**

Centrala zgłoszeń:

* odbieranie zgłoszeń
* inicjacja procesu przydziału karetki

Koordynator karetek:

* zbiera dane o aktualnym położeniu GPS karetek
* wybiera najlepszą karetkę do realizacji zgłoszenia

Karetka:

* przyjmuje realizację zgłoszenia
* stosuje się do wyznaczonej trasy przejazdu

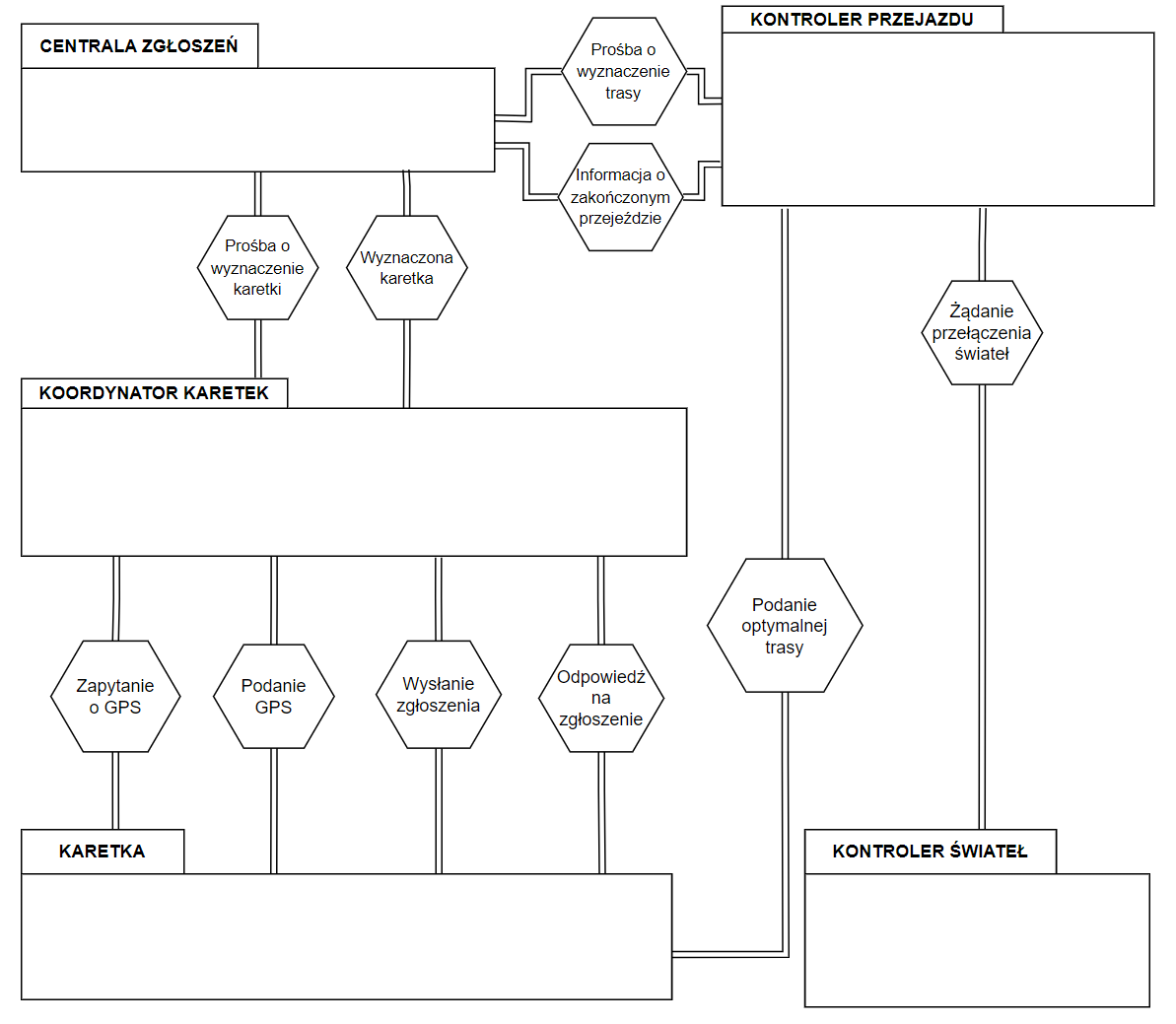
Kontroler przejazdu:

* wyznacza optymalną trasę karetki i przekazuje ją karetce oraz kontrolerowi świateł

Kontroler świateł:

* przełącza światła na skrzyżowaniu

**Diagram konwersacji:**



**Opis interakcji:**

CENTRALA ZGŁOSZEŃ - KONTROLER PRZEJAZDU:

Centrala zgłoszeń komunikuje się z kontrolerem przejazdu, aby określić optymalną trasę przejazdu karetki między szpitalem a miejscem zdarzenia.

Interakcja 1:

|  |  |
| --- | --- |
| Cel | Prośba wyznaczenia trasy |
| Inicjator | Centrala Zgłoszeń |
| Odbiorca | Kontroler Przejazdu |
| Dane | GPS zdarzenia, GPS wybranej karetki |
| Opis przetwarzania | Określenie optymalnej trasy karetki na odcinkach:  punkt startowy karetki - miejsce zdarzenia - szpital |

Interakcja 2:

|  |  |
| --- | --- |
| Cel | Informacja o zakończonym przejeździe |
| Inicjator | Kontroler Przejazdu |
| Odbiorca | Centrala Zgłoszeń |
| Dane | ID karetki, czas zakończenia przejazdu |
| Opis przetwarzania | Oznaczenie zgłoszenia jako zrealizowane oraz zmiana statusu karetki na dostępną |

CENTRALA ZGŁOSZEŃ - KOORDYNATOR KARETEK:

Centrala zgłoszeń komunikuje się z koordynatorem karetek w celu wyznaczenia karetki, która potencjalnie najszybciej będzie w stanie dotrzeć do miejsca zdarzenia.

Interakcja 1:

|  |  |
| --- | --- |
| Cel | Prośba o wyznaczenie karetki |
| Inicjator | Centrala Zgłoszeń |
| Odbiorca | Koordynator karetek |
| Dane | GPS zdarzenia |
| Opis przetwarzania | Prośba o wyznaczenie karetki, która będzie w stanie najszybciej dotrzeć na miejsce zdarzenia. |

Interakcja 2:

|  |  |
| --- | --- |
| Cel | Zwrócenie ID wyznaczonej karetki |
| Inicjator | Kontroler Przejazdu |
| Odbiorca | Centrala Zgłoszeń |
| Dane | ID karetki |
| Opis przetwarzania | Podanie ID karetki która jest najbliżej zdarzenia i zaakceptowała zgłoszenie |

KOORDYNATOR KARETEK - KARETKA:

Koordynator karetek komunikuje się z karetką w celu pobrania aktualnego położenia oraz

wysyłania danych o zgłoszeniach.

Interakcja 1:

|  |  |
| --- | --- |
| Cel | Zapytanie o GPS |
| Inicjator | Koordynator karetek |
| Odbiorca | Karetka |
| Dane | ID karetki |
| Opis przetwarzania | Koordynator karetek co określony interwał czasu aktualizuje koordynaty w bazie danych, z tego powodu wysyła zapytanie do Karetki |

Interakcja 2:

|  |  |
| --- | --- |
| Cel | Podanie GPS |
| Inicjator | Karetka |
| Odbiorca | Koordynator karetek |
| Dane | ID karetki, koordynaty karetki |
| Opis przetwarzania | Odpowiedź na Interakcję 1 i podanie koordynatów GPS karetki |

Interakcja 3:

|  |  |
| --- | --- |
| Cel | Wysłanie zgłoszenia do akceptacji |
| Inicjator | Koordynator karetek |
| Odbiorca | Karetka |
| Dane | GPS zdarzenia oraz informacje dodatkowe |
| Opis przetwarzania | Żądanie realizacji zgłoszenia dla karetki znajdującej się najbliżej zdarzenia |

Interakcja 4:

|  |  |
| --- | --- |
| Cel | Decyzja o podjęciu zgłoszenia |
| Inicjator | Karetka |
| Odbiorca | Koordynator karetek |
| Dane | Odpowiedź: TAK / NIE |
| Opis przetwarzania | Przyjęcie lub odrzucenie proponowanego zdarzenia do realizacji |

KONTROLER PRZEJAZDU - KARETKA:

Kontroler przejazdu komunikuje się z karetką w celu przekazania jej wyznaczonej przez siebie optymalnej trasy przejazdu do pacjenta i do szpitala.

Interakcja 1:

|  |  |
| --- | --- |
| Cel | Podanie optymalnej trasy |
| Inicjator | Kontroler przejazdu |
| Odbiorca | Karetka |
| Dane | Dokładna trasa wyznaczonego przejazdu |
| Opis przetwarzania | Kontroler przejazdu wysyła do karetki wyznaczoną przez siebie optymalną trasę przejazdu w relacji karetka-pacjent-szpital. |

KONTROLER PRZEJAZDU - KONTROLER ŚWIATEŁ:

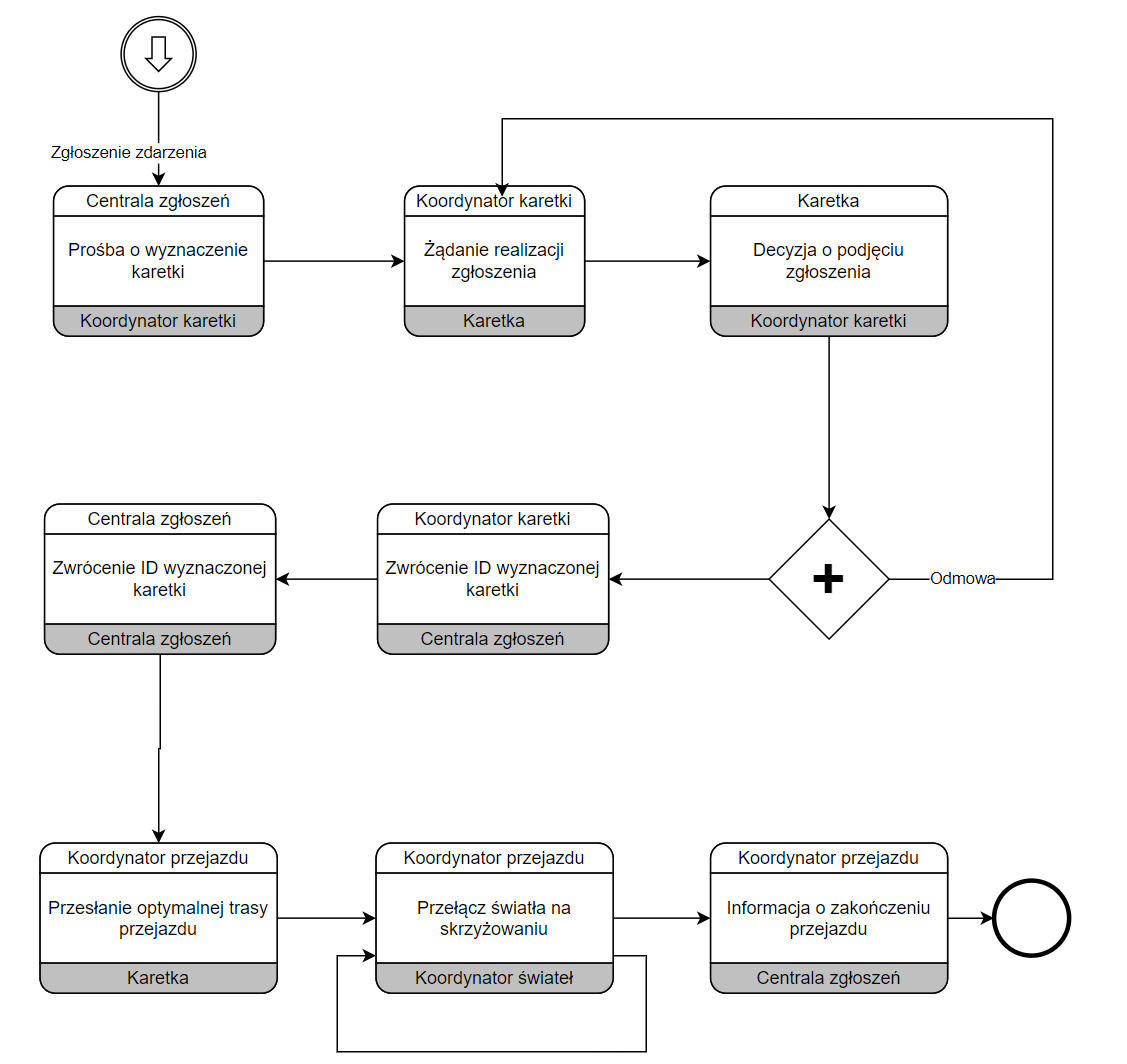
Kontroler przejazdu komunikuje się z kontrolerem świateł, aby przekazywać mu żądania przełączenia świateł na skrzyżowaniu w odpowiednim momencie, by ułatwić przejazd zbliżającej się karetce.

Interakcja 1:

|  |  |
| --- | --- |
| Cel | Żądanie przełączenia świateł |
| Inicjator | Kontroler przejazdu |
| Odbiorca | Kontroler świateł |
| Dane | ID skrzyżowania, pożądane ustawienie świateł |
| Opis przetwarzania | W momencie zbliżania się karetki do skrzyżowania, kontroler przejazdu wysyła do kontrolera świateł żądanie o ustawienie świateł na skrzyżowaniu w odpowiedni sposób. |

**Diagram choreografii:**

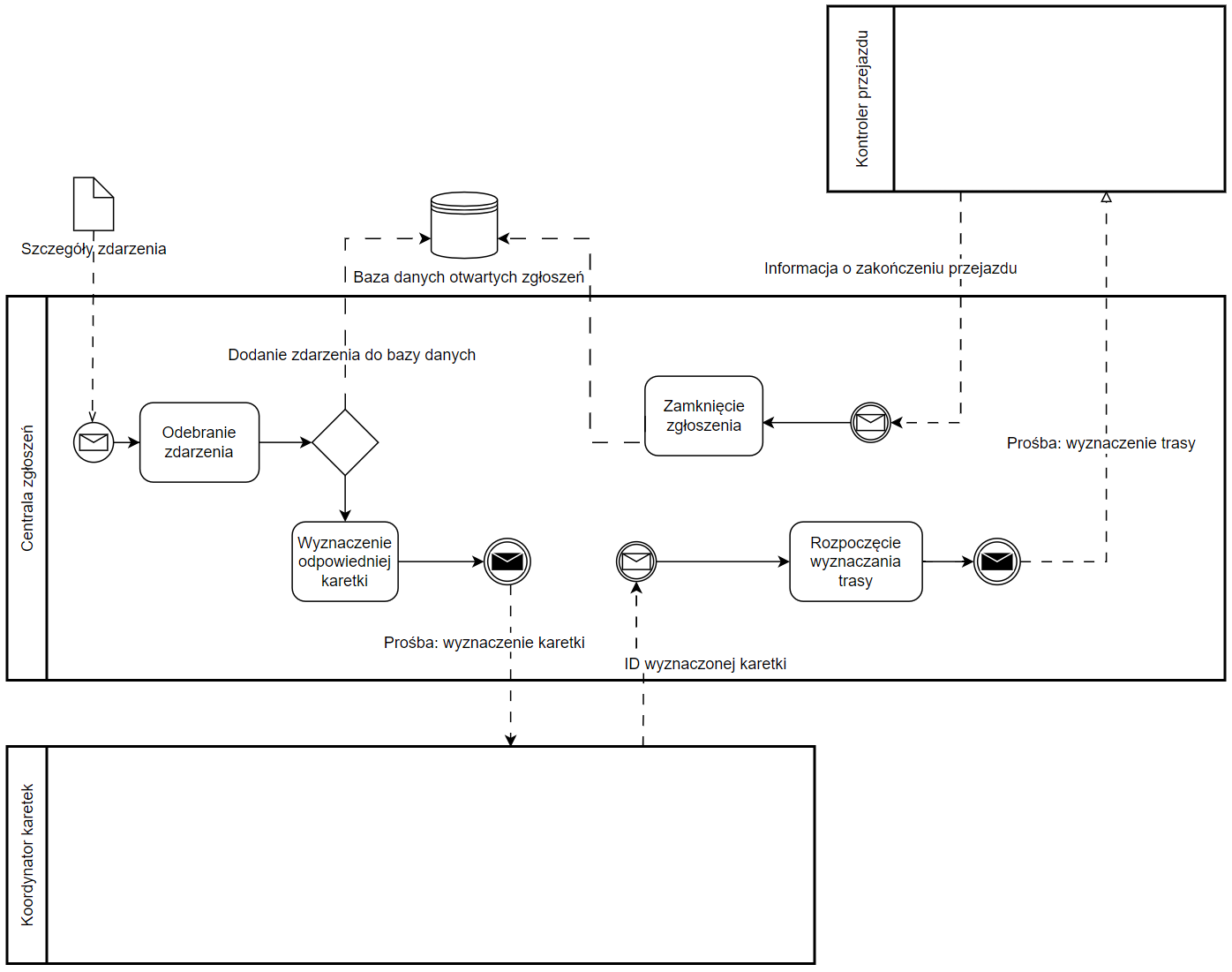
Diagram choreografii przedstawia przedstawione wyżej interakcje w sposób uporządkowany czasowo.

****

**Diagramy kolaboracji:**

**Centrum Zgłoszeń:**

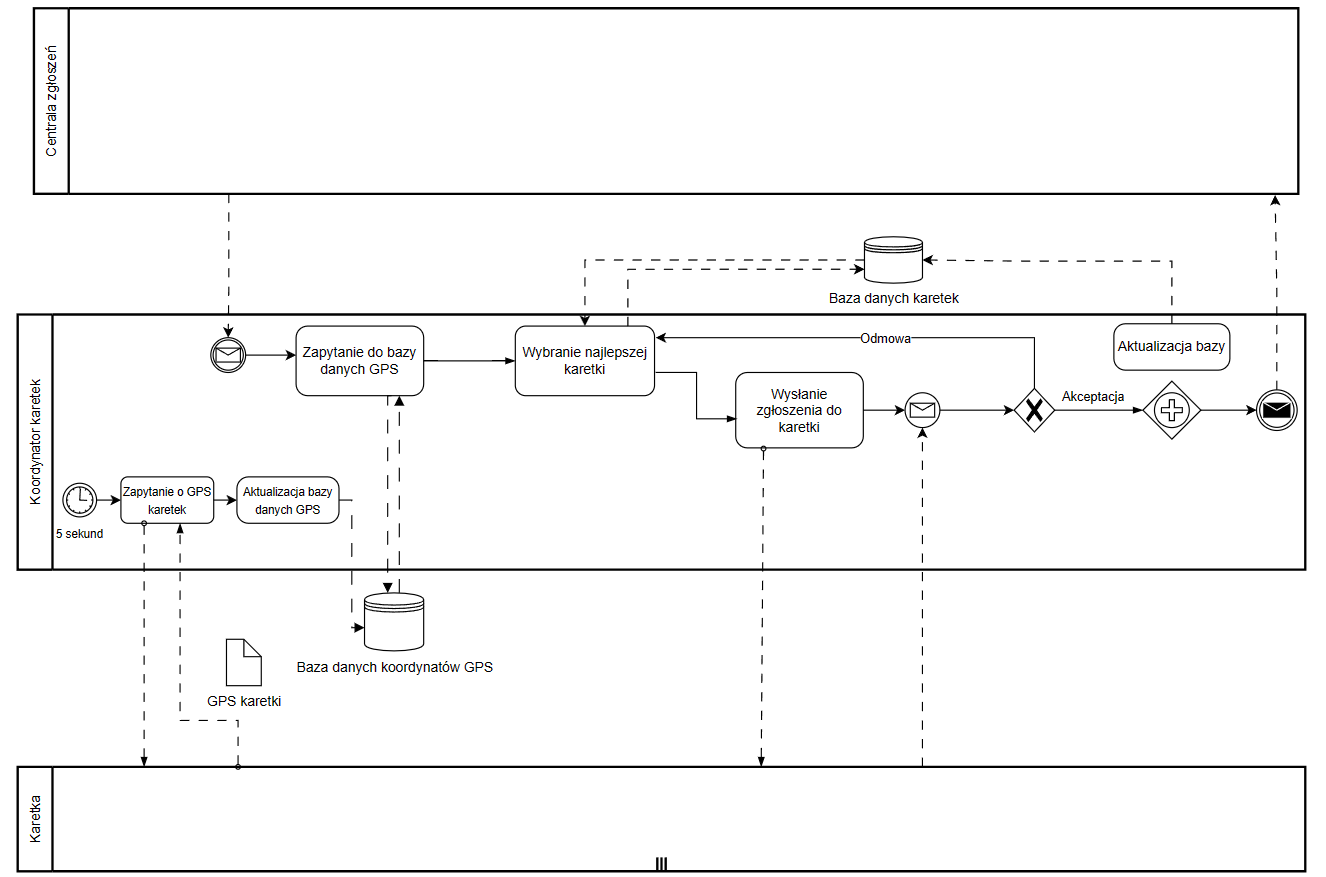
Głównym zadaniem tego agenta jest zarządzanie zgłoszeniami. Po odebraniu zgłoszenia dodaje je do bazy otwartych zgłoszeń oraz wysyła prośbę do koordynatora karetek o wyznaczenie odpowiedniej karetki. Po otrzymaniu odpowiedzi wysyła prośbę o wyznaczenie najlepszej trasy dojazdu do szpitala. Po otrzymaniu informacji o dotarciu karetki do celu (od kontrolera przejazdu) usuwa zgłoszenie z bazy otwartych zgłoszeń.



**Koordynator karetek:**

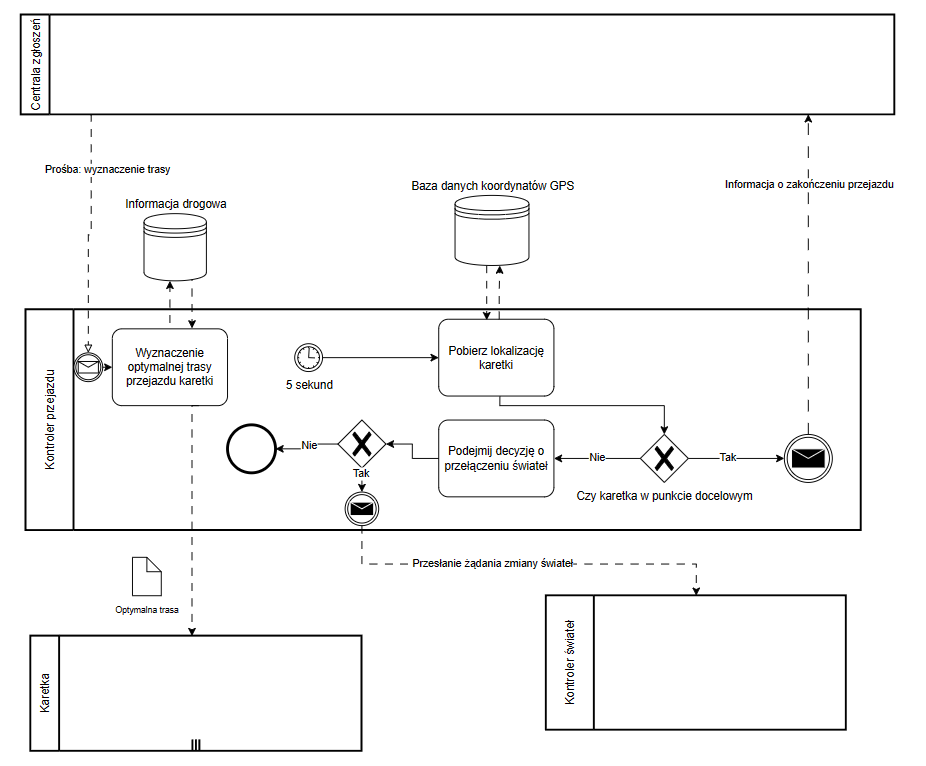
Jest odpowiedzialny za aktualizację bazy danych aktualnych współrzędnych karetek. Zapytania są wysyłane co 5 sekund.

Kolejną funkcją jest wysyłanie żądań do karetek o podjęcie wezwań. Przydział jest podyktowany najlepszym położeniem względem miejsca zdarzenia. W przypadku odmowy, wybierana jest kolejna karetka z listy. Natomiast przy akceptacji, ID karetki jest przesyłane do Centrali Zgłoszeń.

****

**Kontroler przejazdu:**

Kontroler przejazdu jest agentem odpowiedzialnym za nawigowanie karetki. Komunikuje się on z wieloma agentami. Od centrum zgłoszeń dostaje informacje o konkretnym zgłoszeniu w celu jego obsługi. Po otrzymaniu takiego zgłoszenia wyznacza optymalną trasę na podstawie informacji drogowej i wysyła tą trasę do agenta karetki. W czasie przejazdu karetki, agent kontrolera przejazdu co 5 sekund decyduje o przełączaniu świateł na skrzyżowaniu przy pomocy agenta kontrolera świateł. W przypadku gdy karetka dojedzie do celu agent ten wysyła o tym informację do agenta centrum zgłoszeń.

****

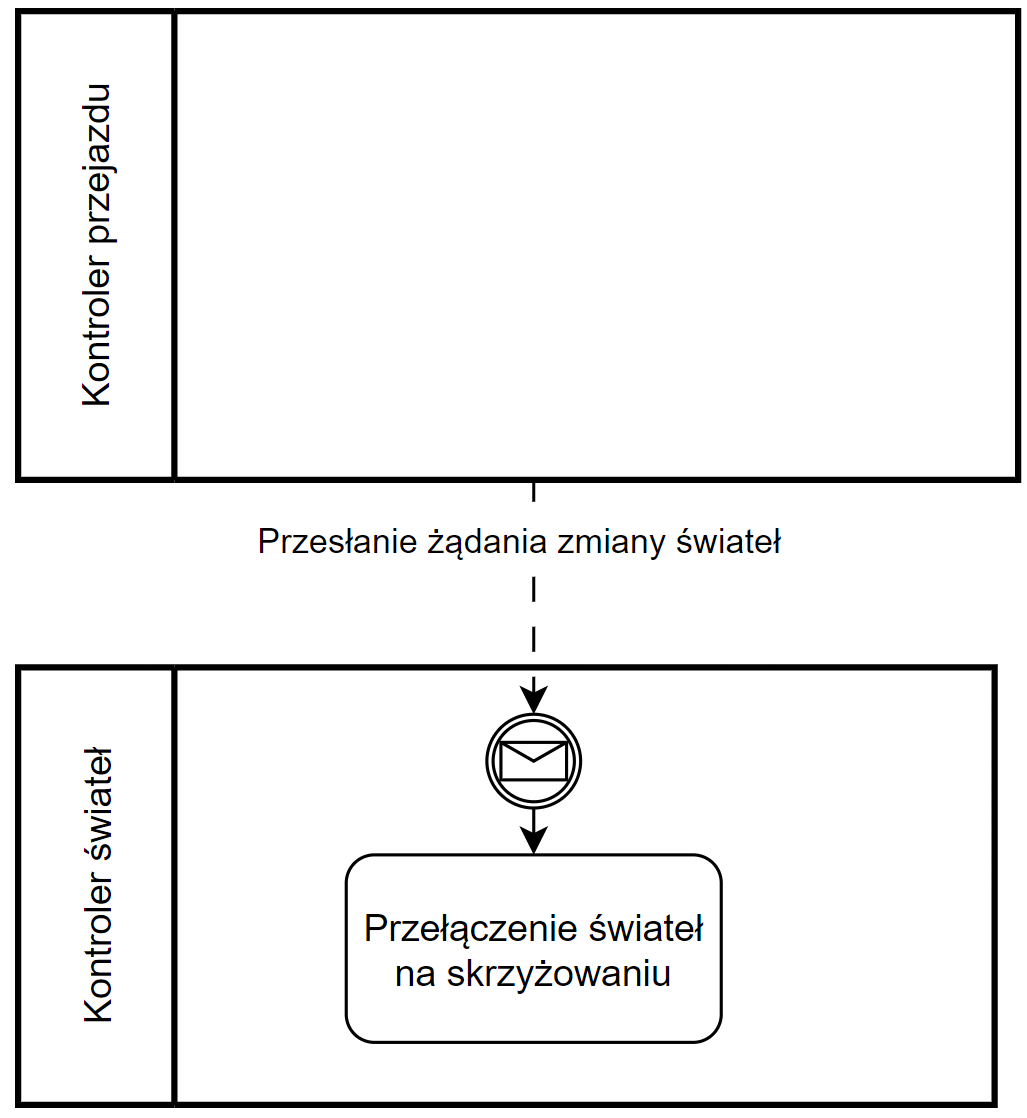
**Karetka:**

Karetka komunikuje się z Kontrolerem karetek oraz Kontrolerem przejazdu. W regularnych odstępach czasu otrzymuje od Kontrolera karetek zapytania o swoją lokalizację, w odpowiedzi na nie zwracając swoje współrzędne geograficzne. Karetka oczekuje też na zgłoszenia, które napływają od Kontrolera karetek. W zależności od tego, czy Karetka jest w stanie podjąć się w danym momencie zgłoszenia, zwraca odpowiednią informację. Jeżeli przyjmie zgłoszenie, oczekuje na otrzymanie z Kontrolera przejazdu optymalnej trasy, a po jej otrzymaniu rozpoczyna realizację zgłoszenia.

****

**Kontroler świateł:**

Kontroler świateł jest odpowiedzialny za zarządzanie światłami drogowymi na skrzyżowaniach. Komunikuje się on jedynie z Kontrolerem przejazdu i w przypadku otrzymania od ww. żądania zmiany świateł, ustawia je w sposób zawarty w żądaniu.

****

ETAP C

W ramach implementacji zaprojektowanego systemu wieloagentowego wykorzystano język ***Python*** oraz framework ***Spade***.

Zgodnie ze standardami implementacji Spade, utworzono klasy odpowiadające wyznaczonym agentom:

* Ambulance
* AmbulanceCoordinator
* EmergencyCenter
* RouteCoordinator
* TraficLightsCoordinator

W ramach poszczególnych klas utworzono podklasy implementujące zachowania konkretnych agentów, ściśle związane z protokołami komunikacji między agentami.

**Ambulance** - klasa implementuje agenta karetki. Jej głównym zadaniem jest komunikacja   
z centrum zgłoszeń (AmbulanceCoordinator): zachowanie odebrania i potwierdzania / odrzucania zgłoszenia oraz aktualizacji położenia karetki. Klasa ambulance otrzymuje wyznaczoną trasę dotarcia do celu od koordynatora przejazdu (RouteCoordinator). Trasa została zaimplementowana jako lista koordynat (x,y) na mapie 2D, przez które powinna przejechać symulowana karetka. Dodatkowo klasa implementuje symulację poruszania się karetki po mapie.

Schemat wiadomości, których nadawcą jest opisywany agent:

1. Odpowiedź potwierdzająca/odrzucająca przyjęcie zgłoszenia
   1. Do: AmbulanceCoordinator
   2. Performative: confirm
   3. Language: request-answer
   4. Body: {„response”: yes/no}
2. Przesyłanie aktualnych kordynatów GPS karetki
   1. Do: AmbulqnceCordinator
   2. Performative: inform
   3. Language: GPS
   4. Dodatkowe metadane
      1. Identyfikator ambulansu: „ambulance\_id”
   5. Body: {[x,y]}

**AmbulanceCoordinator** - klasa implementuje agenta koordynatora karetek. Jej głównym zadaniem jest komunikacja z karetkami: wyznaczenie karetki najbliżej zdarzenia, na podstawie koordynat deklarowanych przez karetki; wysłanie zapytania o obsługę zdarzenia do najbliższej karetki. Ponadto klasa AmbulanceCoordinator odpowiada za odbieranie zgłoszeń z centrali oraz przekazywanie odebranego położenia wybranej karetki do koordynatora przejazdu. Dodatkowo klasa odpowiada za prezentację graficzną położenia karetek.

Schemat wiadomości, których nadawcą jest opisywany agent:

1. Wysłanie danych zgłoszenia do konkretnej karetki
   1. Do: Ambulance
   2. Performative: request
   3. Language: Event-request
   4. Body: {[x,y]}
2. Zapytanie koordynatora przejzdu o wyznaczenie najlepszej trasy
   1. Do: RouteCoordinator
   2. Performative: request
   3. Language: path-request
   4. Body: {event\_id, ambulance\_id, ambulance\_location, event\_location}
3. Przesłanie aktualnych koordynatorów GPS karetki
   1. Do: RouteCoordinator
   2. Performative: inform
   3. Language: gps-progress
   4. Dodatkowe metadane
      1. Identyfikator ambulansu: „ambulance\_id”
   5. Body: {[x,y]}
4. Przesłanie powiadomienia o zakończeniu przejazdu do koordynatora trasy
   1. Do: RouteCoordinator
   2. Performative: inform
   3. Language: gps-progress-finished
   4. Dodatkowe metadane
      1. Identyfikator ambulansu: „ambulance\_id”
5. Przesłanie powidomiemia o zakończeniu przejazdu do centrum zgłoszeń
   1. Do: EmergencyCenter
   2. Performative: inform
   3. Language: event-finish
   4. Dodatkowe metadane
      1. Identyfikator zdarzenia : „event\_id”

**EmergencyCenter** - klasa implementuje agenta centrum zgłoszeń. Jej głównym zadaniem jest inicjacja zdarzeń, które powinny zostać obsłużone przez system - zachowanie wysłania komunikatu informujacym o zdarzeniu, do AmbulanceCoordinator.

Schemat wiadomości, których nadawcą jest opisywany agent:

1. Przesłanie danych o zgłoszeniu do koordynatora ambulansów
   1. Do: AmbulanceCoordinator
   2. Performative: inform
   3. Language: event-report
   4. Dodatkowe metadane
      1. Identyfikator zgłoszenia : „event\_id”
   5. Body: {[x,y]}

**RouteCoordinator** - klasa implementuje agenta koordynatora przejazdu. Odpowiada za wyznaczenie optymalnej trasy dojazdu karetki do celu. W ramach klasy zaimplementowano zachowanie odebrania zapytania o wyznaczenie trasy. Inicjuje ono kolejne zachowanie - wysłania optymalnej trasy, w ramach którego wyznaczana jest najlepsza droga przejazdu karetki. Klasa RouteCoordinator jest w stanie wyznaczyć trasę przejazdu dzięki cyklicznie odbieranym położeniom karetek (kolejne zachowanie: odbioru położenia GPS). Ponadto zostały zaimplementowane dwa dodatkowe zachowania: wysłania informacji o zakończonym przejeździe (skierowane do EmergencyCenter) oraz wysłania zapytania o zmianę świateł (skierowane do TrafficLightsCoordinator).

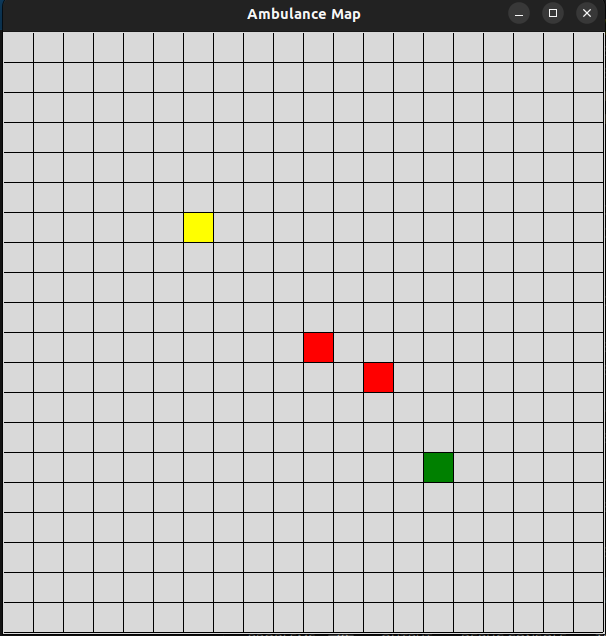
Schemat wiadomości, których nadawcą jest opisywany agent:

1. Przesłanie optymalnej trasy dojazdu do karetki
   1. Do: Ambulance
   2. Performative: inform
   3. Language: optimal-route
   4. Body: {[[x,y], … [x,y]]}
2. Przesłanie zapytania zmiany świateł
   1. Do: TrafficLightsCoordinator
   2. Performative: inform
   3. Language: change\_lights
   4. Dodatkowe metadane
      1. Identyfikator ambulansu: „ambulance\_id”
   5. Body: {[x,y]}

**TrafficLightsCoordinator** - klasa implementuje agenta koordynatora świateł. Odpowiada za zmianę świateł na skrzyżowaniu w sposób optymalny dla czasu przejazdu karetki. Polega to na zmianie świateł na zielone w kierunku poruszania się karetki. W ramach klasy zaimplementowano zachowanie odebrania sygnału zmiany świateł. Klasa ta imituje integrację z zewnętrznym systemem sygnalizacji świetlnej, który realizowałby zmianę świateł na konkretnym skrzyżowaniu.

**Zaimplementowany scenariusz** - w ramach implementacji systemu, na obecnym etapie, obsługiwany jest podstawowy scenariusz, w którym centrala otrzymuje dwa zgłoszenia. Następnie są one przydzielane do odpowiednich karetek. Scenariusz kończy się po dojechaniu karetek do miejsca zdarzenia. W kolejnym etapie realizacji projektu obsłużone zostaną sytuację wyjątkowe, które mogą wystąpić podczas scenariusza - np. odmowa realizacji zgłoszenia przez karetkę.

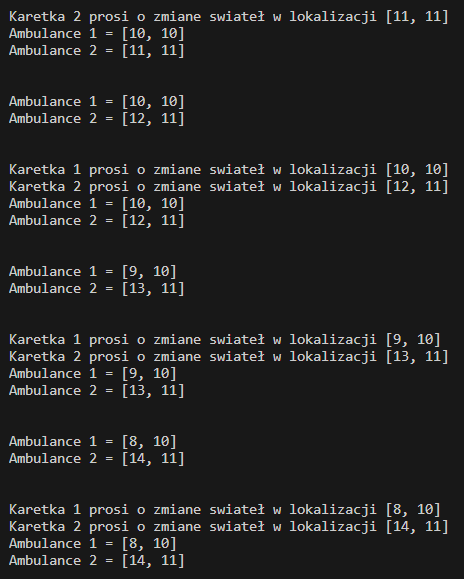
**Interfejs graficzny** - do wizualizacji środowiska, w którym operuje system wykorzystano bibliotekę **tkinter**. Bardzo uproszczoną imitację miasta przedstawiono za pomocą szachownicy, na której różnokolorowe pola przedstawiają miejsca zgłoszeń, a czerwone, przemieszczające się kwadraty, przedstawiają pozycje karetek.

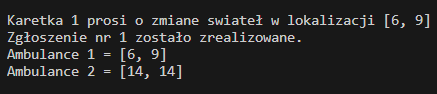


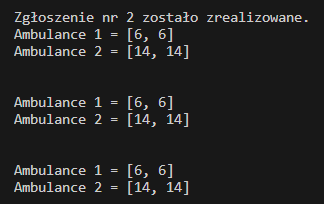
Poniżej zostały przedstawione informacje wyświetlane w konsoli, po uruchomieniu programu, pokazujące kolejność wszystkich zdarzeń (zachowań agentów):

* Początkowo *Karetki* stoją w miejscu w arbitralnych punktach na mapie
* *Centrala Zgłoszeń* przesyła zgłoszenie nr 1 do *Koordynatora Karetek*
* *Koordynator Karetek* znajduje najbliższą *Karetkę* w swojej bazie GPS, pyta ją o dostępność i dostaje potwierdzenie gotowości
* Następnie (na prośbę *Koordynatora Karetek*) *Koordynator Przejazdu* wyznacza najlepszą trasę (na podstawie lokalizacji karetki, lokalizacji zdarzenia oraz swojej wiedzy o ruchu w mieście - stosując prosty algorytm wyszukiwania najkrótszej ścieżki
* *Karetka* otrzymuje trasę i rozpoczyna drogę do miejsca zdarzenia.



* *Koordynator Przejazdu* otrzymuje na bieżąco lokalizację karetki i wysyła żądania do *Koordynatora Świateł* w celu zmiany świateł w konkretnych miejscach.  
    
  
* Gdy *Karetka* dotrze na miejsce, *Koordynator Karetek* przekazuje ten fakt *Centrali Zgłoszeń*, a ta oznacza to zgłoszenie jako zrealizowane - zakończone.



* Po zakończeniu przejazdu, *Karetki* stoją w miejscu oczekując na kolejne polecenia.  
  

ETAP D

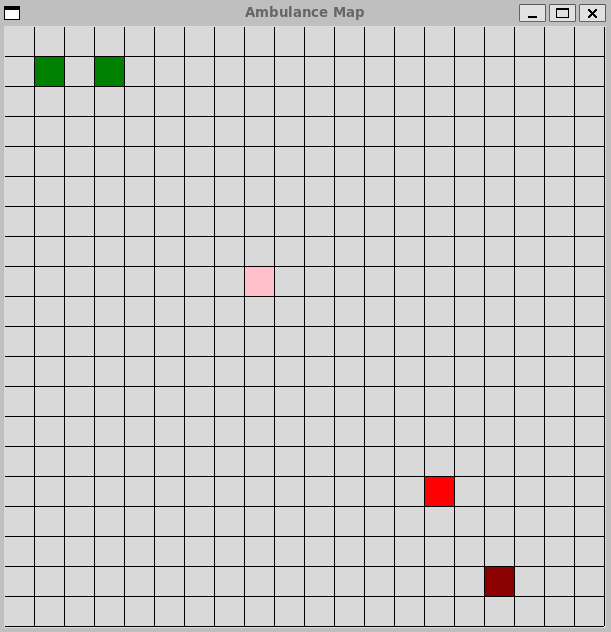
**Rozwój głównego scenariusza**

W kolejnym etapie projektu rozwinęliśmy nasz podstawowy schemat dodając większą elastyczność w komunikacji agentów. Przede wszystkim karetki nie zawsze mogą odpowiadać na wezwanie do realizacji zgłoszenia, np. ze względu na awarię pojazdu. Wówczas koordynator karetek czeka określony czas na odpowiedź, a następnie wybiera kolejną najbliższą karetkę. Ważne jest to, że czas oczekiwania jest nieodwołalny i gdy karetka nie odpowie w odpowiednim czasie, to jest zupełnie pomijana w realizacji dane zgłoszenia - by uniknąć skomplikowanych mechanizmów “odwoływania karetki”.

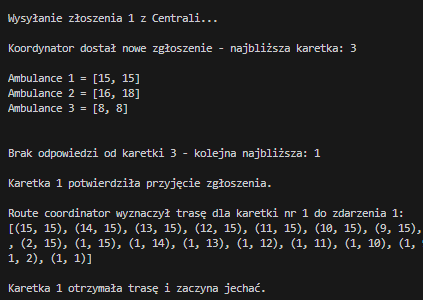
Oto przykład takiej sytuacji:

Karetki są oznaczone kolorami: 1 - czerwona, 2 - bordowa, 3 - różowa, oraz zostały zgłoszone dwa zdarzenie zaznaczone na zielono.

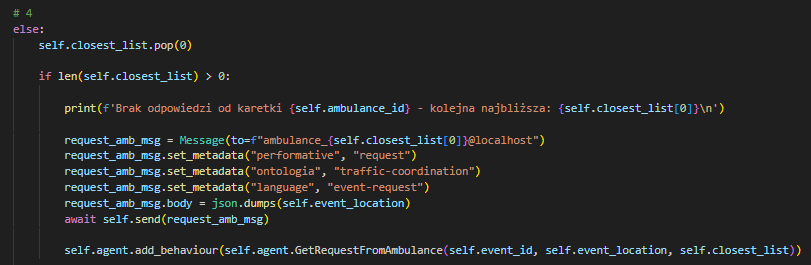
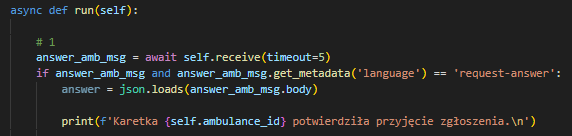
Karetka nr 3 (różowa) została celowo pozbawiona mechanizmu komunikacji tak by zasymulować brak odpowiedzi, gdyż jak widać znajduje się ona najbliżej zdarzeń.



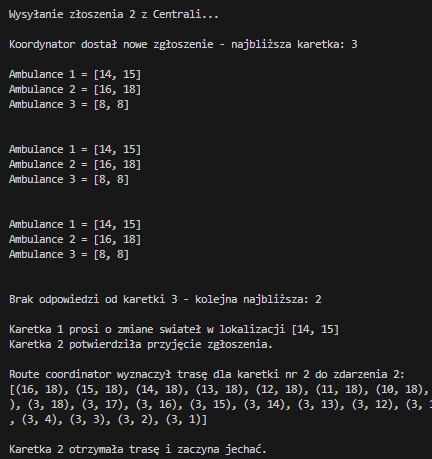
Karetka nr 3 nie odpowiada, więc zostaje wezwana karetka nr 2 (czerwona), akceptuje zgłoszenie, otrzymuje trasę oraz rozpoczyna przejazd:



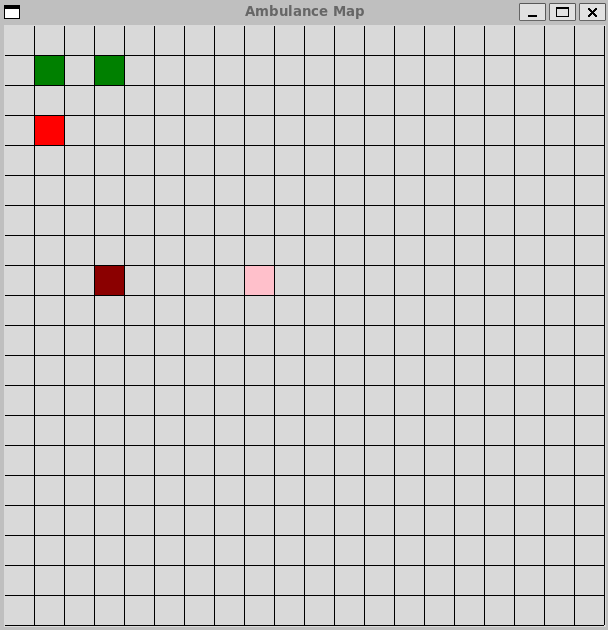
Funkcja wysyłania wezwania ma określony timeout, jeśli warunek nie zostanie spełniony drugą opcją jest wybranie kolejnej karetki z posortowanej listy i wysłanie do niej wezwania:



Centrala zawiadomiła koordynatora karetek o kolejnym zgłoszeniu. Karetka nr 3 znów nie odpowiedziała, więc została wezwana kolejna karetka nr 2 (bordowa). Warto zauważyć, że koordynator karetek wie, które karetki są obecnie zajęte, więc w tym przypadku karetka nr 1 będąca najbliżej zgłoszenia została pominięta (ponieważ już jedzie do innego zdarzenia).



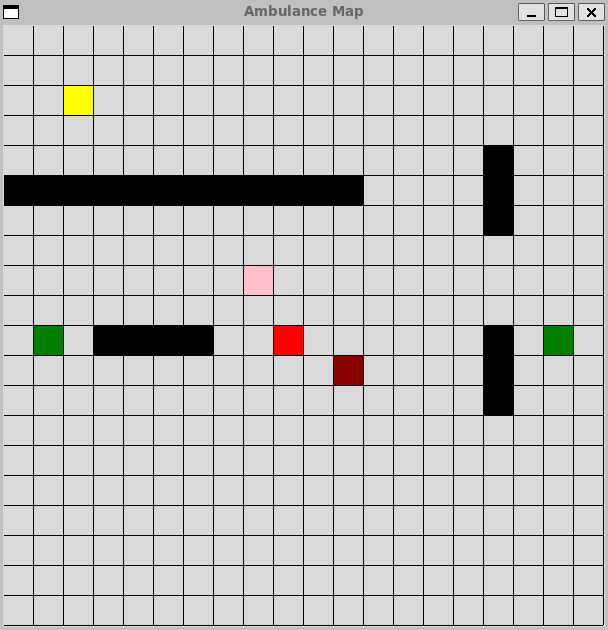
Jak można zaobserwować, karetki nr 2 i 3 po jakimś czasie dotarły do wyznaczonych zdarzeń.



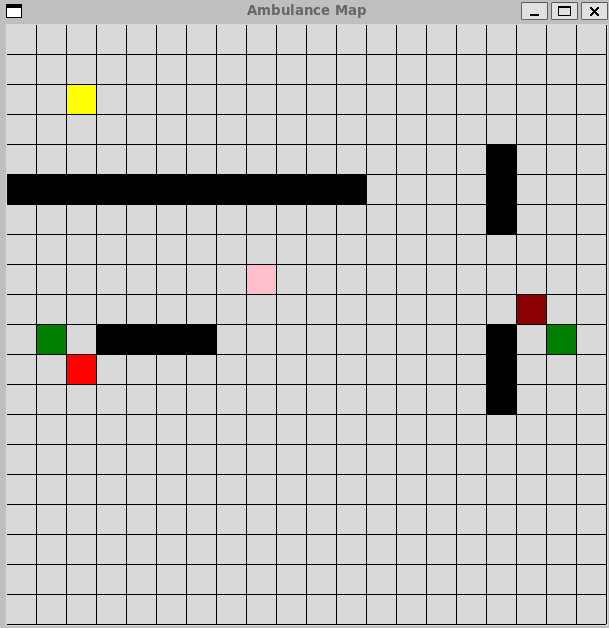
**Rozwój mapy miasta**

Kolejną zmianą jest dodanie nieprzejezdnych odcinków na mapie, symbolizujące budynki. Jest to uwzględniane przez koordynatora trasy, tak by dobierał najkrótszą trasę, ale po przejezdnych odcinkach miasta. Dodatkowo umożliwione zostało interaktywne zaznaczanie zdarzeń, poprzez kliknięcie na wybrany kwadrat na mapie. Wówczas miejsce zostaje zaznaczone na zielono, a proces wyboru karetki oraz jej dojazdu jest realizowany niezmiennie jak w poprzedniej wersji.

Interaktywne zgłaszania wypadków przy nieprzejezdnych odcinkach miasta:



Po chwili odpowiednie karetki dojeżdżają do miejsca zdarzenia:



**Uwaga:** poniżej znajdują się pliki .gif i będą one widoczne tylko sprawozdaniu w wersji *.doc*.

Zgłoszenie 3 wypadków, przydział karetek i przejazd do wyznaczonych miejsc. Następnie można zgłaszać kolejne zdarzenia:

Żółta karetka jest “nieaktywnym agentem” symulującym brak odpowiedzi karetki.

Obraz zawierający zrzut ekranu, kwadrat, linia, diagram

Opis wygenerowany automatycznie

W kolejnym przypadku jest więcej zgłoszeń niż dostępnym karetek. Nadmiarowe zgłoszenia oczekują w kolejce i natychmiast po zwolnieniu karetki z poprzedniego zgłoszenia są one obsługiwane:

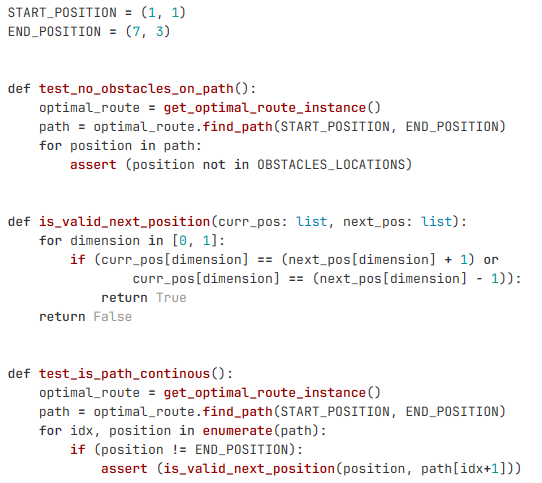
Obraz zawierający zrzut ekranu, kwadrat, linia, diagram

Opis wygenerowany automatycznie

**Testy**

Podczas implementacji napisano testy jednostkowe (znajdują się w pliku /src/city\_emergency\_m\_s/tests/test\_agents.py). Wykorzystano do tego celu bibliotekę **pytest.** Zaimplementowane testy weryfikują poprawność podstawowych metod agentów, jak np. wyznaczanie najkrótszej trasy przejazdu karetki.

Większość funkcji systemu przetestowano przeprowadzając test integracyjne. Przede wszystkim zweryfikowano działanie komunikacji między agentami. Ułatwiły to logi aplikacji pojawiające się w konsoli podczas odbierania/nadawania wiadomości. Sprawdzono poprawność działania interfejsu graficznego, również weryfikując spójność tego co widoczne jest w oknie aplikacji, z tym co wyświetlane jest w konsoli. Poniższy zrzut ekranu przedstawia kilka przykładowych testów.



**Brakujące elementy oraz możliwości rozwoju**

1. **Brak symulacji sygnalizacji świetlnej oraz analizy ruchu na drodze**

W teorii cały projekt opierał się na zarządzaniu przejazdem karetki oraz sygnalizacją świetlną majacą czas tego przejazdu skrócić. W samej implementacji jednak zabrakło elementów świateł drogowych oraz ruchu drogowego ze względu na bardzo wysoką złożoność tego problemu. Aby wdrożyć opisane funkcjonalności należałoby opracować elementy takie jak:

* Symulację ruchu drogowego na ulicach miasta,
* Funkcjonalność świateł drogowych na skrzyżowaniach oraz ich wpływ na symulowany ruch drogowy,
* Rozwój funkcjonalności wyznaczania optymalnej trasy przejazdu karetki tak, aby brała ona pod uwagę natężenie ruchu drogowego,
* Opracowanie algorytmu podejmującego decyzję o przełączeniu świateł drogowych. Algorytm taki musiałby brać pod uwagę wpływ takich zmian na generowanie korków w mieście oraz potencjalny wpływ na przejazd innych karetek.

1. **Brak “powrotu do szpitala”**

Projekt mógłby zostać rozwinięty o funkcjonalność podejmowania decyzji, czy dane zdarzenie wymaga transportu do szpitala. Jeżeli taki transport byłby konieczny następowałoby wyznaczenie optymalnej trasy przejazdu do najbliższego/najodpowiedniejszego szpitala oraz zarządzanie tym przejazdem w sposób podobny jak aktualnie zaimplementowany dojazd do zdarzenia.

1. **Rozwój mapy**

W implementacji naszego projektu posługujemy się jedynie mapą w formie szachownicy z nielicznymi polami wykluczonymi z możliwości przejazdu. Dalszy rozwój projektu mógłby być związany z rozwojem aktualnej mapy poprzez jej zwiększenie i dodanie większej ilości wykluczonych obszarów, lub nawet z kompletną zmianą koncepcji samej mapy i z przebudowaniem jej w sposób odpowiadający rzeczywistej topologii jakiegoś miasta np. Warszawy.

**Napotkane problem**

Podczas rozpoczęcia procesu implementacji cały zespół napotkał problem konfiguracji serwera XMPP na swoich komputerach. Każdy z zespołu posiadał inny system operacyjny lub środowisko uruchomieniowe (praca na obrazie Docker). Przysposobiło to problemów z odpaleniem tzw *Hello worlda.* Problemy były różne czy to ustawienie danych uwierzytelniających po certyfikaty TLS na lokalniej maszynie. Używanie jednej wersji z pewnością przyspieszyłoby proces rozpoczęcia komunikacji.

Dodatkowym problemem było zaimplementowanie i zaprojektowanie mapy miasta na tyle zaawansowanej która obsługiwałaby faktyczny proces zmiany świateł. W tej wersji systemu zostało to zaślepione.

**Wnioski**

W trakcie implementacji udało się skutecznie zrealizować kluczowe funkcjonalności systemu agentowego, takie jak komunikacja między agentami, zarządzanie zasobami, podejmowanie decyzji na podstawie obecnej sytuacji środowiska oraz adaptacyjność systemu do zmian. Projekt został wdrożony zgodnie z założeniami (choć z pewnymi uproszczeniami), potwierdzając, że paradygmat agentowy może być skutecznym narzędziem do rozwiązywania złożonych problemów. Mimo osiągnięcia satysfakcjonujących wyników, istnieją obszary, które mogłyby zostać rozwinięte. Zostały one przedstawione w rozdziale **Brakujące elementy oraz możliwości rozwoju**.

Podsumowując, projekt implementujący Inteligentny system wspomagający przejazd pojazdów ratownictwa medycznego osiągnął wyznaczone cele, dostarczając efektywne rozwiązanie oparte na paradygmacie agentowym.