Akademia Górniczo-Hutnicza
Wydział Elektrotechniki, Automatyki, Informatyki i Inżynieri Biomedycznej
Kierunek Informatyka

Modelowanie i Symulacja Systemów

Projekt symulacji aktualnych opóźnień krakowskiej komunikacji miejskiej uwzględniający aktualne warunki drogowe i pogodowe

...

Patryk Gałczyński, Szymon Duda

Kraków, 15 stycznia 2017

1. Spis treści

2. Motywacja

Symulacja to proces przybliżonego odtwarzania zachowania jakiegoś obiektu za pomocą jego modelu. Tak formalnie można określić meritum naszej pracy nad projektem. Codziennie rano gdy zmierzamy na uczelnie czy do do pracy dotyka nas ten sam problem, kiedy przyjedzie nasz tramwaj czy autobus?
Rozkład jazdy mpk nie zawsze pokrywa się z rzeczywistością, stąd w naszych głowach zrodził się pomysł, aby spróbować wyjść naprzeciw problemowi i spróbować przewidzieć z większą dokładnością kiedy nasz tramwaj przyjedzie na przystanek.
Obecnie na rynku funkcjonuje już aplikacja JakDojadę, lecz jak niejednokrotnie każdy z nas się przekonał i ona nas zawodzi. Dlaczego więc nie spróbować zrobić tego lepiej?

Symulacja ruchu w komunikacji miejskiej to bardzo złożony proces. Na wynik symulacji wpływa wiele czynników, niejednokrotnie losowych, których nie jesteśmy w stanie przewidzieć, jak wypadek, który spowoduje zwężenie ruchu czy całkowity zator a także to, że autobusy czy tramwaje w naszym mieście są mocno wysłużone więc istnieje prawdopodobieństwo awarii co niesie za sobą opóźnienia.

Warunki pogodowe również znacznie wpływają na prędkość pojazdów i jest to czynnik determinujący czas pokonania trasy. Nie można także zapomnieć, że to najczęściej my czyli użytkownicy komunikacji miejskiej generujemy opóźnienia. Rano gdy większość społeczeństwa podróżuje komunikacją w stronę miejsca pracy czy nauki w autobusach i tramwajach jest bardzo tłoczno. Mamy do czynienia z dużą liczbą osób wsiadających i wysiadających, niejednokrotnie w drzwiach tramwajów czy autobusów tworzą się zatory które uniemożliwiają sprawną rotację pasażerów co znacząco wpływa na czas postoju na przystanku. Zważyć należy również na to, że w godzinach szczytu mamy styczność z wzmożonym ruchem samochodowym, niejednokrotnie dochodzi do sytuacji w których tworzą się korki uniemożliwiające swobodny przejazd pojazdom komunikacji miejskiej.

Wszystkie opisane powyżej czynniki są ważne przy próbie symulacji ruchu komunikacji miejskiej, tak więc w naszym projekcie uwzględniamy jaką mamy pogodę którą pobieramy przez API serwisu pogodowego, statystyczną ilość osób podróżujących (statystyczne dane udostępnia mpk, nie są one bardzo precyzyjne lecz by przeprowadzić własne obserwację potrzeba by nam bardzo dużo wolnego czasu i zasobów ludzkich na co nie jesteśmy w stanie sobie pozwolić), czy jakie mamy obecnie natężenie w ruchu drogowym, które udostępniają nam mapy HereWeGo od Nokii. Wszelkie awarie pojazdów, będziemy w przyszłości w stanie odebrać za pomocą kanału RSS mpk.

Proces symulacji postanowiliśmy przedstawiać jako ciągłą animacje nałożoną na Google Maps. Ruch tramwajów został zamodelowany za pomocą grafu gdzie wierzchołki grafu to przystanki bądź potencjalne skrzyżowania, gdzie można oczekiwać zatorów, a krawędzie grafu to odległość między nimi. Nie jest to model idealny, lecz daje nam

możliwość kolejkowania tramwajów w przypadku np całkowitego zatoru na trasie gdzie biegną tory. Wszelkie szczegóły techniczne znajdują się w dalszej części artykułu.

3. Przegląd literatury

Poniżej zamieszczamy wcześniej opracowany przegląd literatury:

“Modelowanie wyboru trasy w gęstych sieciach miejskich”

[yadda.icm.edu.pl/baztech/element/bwmeta1.element.baztech-article-BSL3-0026-0080/c/Zochowska.pdf]

Publikacja “Modelowanie wyboru trasy w gęstych sieciach miejskich” autorstwa Renaty Żochowskiej rozważa temat modelowania zachowań uczestników ruchu oraz wybieranych przez nich tras w zależności od wielu czynników. Najważniejszym z nich jest tak zwany “koszt przejazdu” który zależy między innymi od długości trasy, przewidywanego czasu podróży, opłat drogowych i, co dla naszego projektu wydaje się być najistotniejsze, **opór trasy**.

Pierwsze próby zdefiniowania oporu trasy, powstawały w już w latach 50-tych ubiegłego wieku. Od tego czasu w związku z silnym rozwojem technologii powstało kilka silnie rozwiniętych modeli matematycznych definiujących zależność czasu potrzebnego na pokonanie danej trasy w zależności od między innymi:

- czasu przejazdu odcinka o jednakowej długości przy ruchu swobodnym,
- potoku na odcinku,
- przepustowości odcinka,
- natężeniu nasycenia,
- parametrach modelu

Autorka w swojej publikacji przytacza kilka różnych przykładowych modeli matematycznych oporu (dokładnie 7, przykład w załączniku nr 4.), jesteśmy pewni, że któryś z nich będzie odpowiedni dla naszego projektu. W celu wybrania odpowiedniego modelu posłużymy się tabelą określającą “warunki przydatności funkcji do opisu oporu trasy” zdefiniowaną w wyżej wymienionej publikacji jako tabela 1 (załącznik nr 4.). Tabela ta określa zarówno warunki matematyczne jak i behawioralne według których wybierzemy najbardziej przydatną funkcję dla naszych potrzeb. Jedyna obawa jaką widzimy to to, że publikacja ukazała się 5 lat temu i od tej pory prawdopodobnie ruch miejski zdążył wyewoluować.

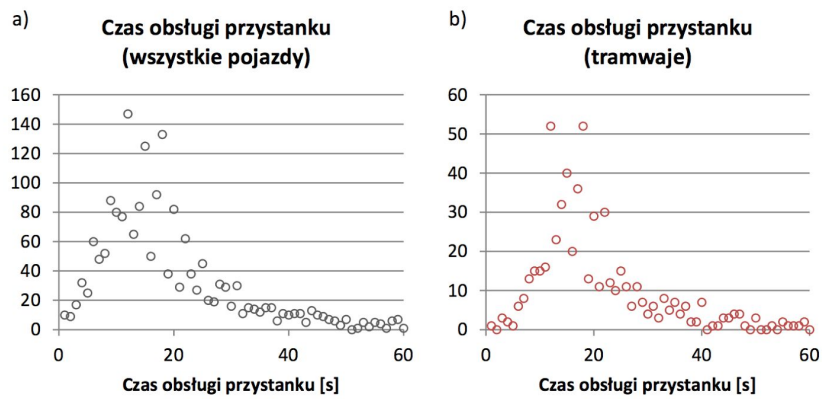
“Analiza czynników wpływających na prędkość pojazdów transportu zbiorowego na przykładzie Gdańska.”

[https://www.researchgate.net/profile/Kazimierz_Jamroz/publication/259043038_Analiza_czynnikow_wplywajacych_na_predkosc_pojazdow_transportu_zbiorowego_na_przykladzie_Gdanska_Analysis_of_factors_affecting_public_transport_vehicle_speed_on_case_of_Gdansk/links/0c960529cf07e67ccc000000.pdf]

Publikacja “ANALIZA CZYNNIKÓW WPŁYWAJĄCYCH NA PRĘDKOŚĆ POJAZDÓW TRANSPORTU ZBIOROWEGO NA PRZYKŁADZIE GDAŃSKA.” autorstwa Krystian Birr, Kazimierz Jamroz, Wojciech Kustra to rzetelna analiza czynników jakie mają wpływ na opóźnienia autobusów oraz tramwajów na przykładzie komunikacji miejskiej Gdańska.

Jak wynika z raportu, około połowa zatrzymań w sumarycznej ich liczbie wynika z konieczności obsłużenia przystanku (odpowiednio ~51% dla autobusów oraz 66% dla tramwajów). Sygnalizacja świetlna wymusza natomiast około 30% zatrzymań (~32% dla autobusów i ~28,5% dla tramwajów). Ostatnim istotnym zdarzeniem jest zator który oddziałuje na autobusy w ilości ~11% oraz 2,5% dla tramwajów. Pozostałe zdarzenia jak ustąpienia pierwszeństwa, zablokowany przejazd, piesi i inne nie przekraczają 3,2% (ta najwyższa wartość występuje w przypadku ustępowania pierwszeństwa przez autobusy).

Rys. 3. Średni czas zatrzymania z podziałem na powód zatrzymania.



Rys.3 pokazuje, że średni czas zatrzymania w przypadku przystanków wynosi ok 20 sekund dla autobusu i tramwaju, sygnalizacja wymusza natomiast zatrzymania rzędu 25-35 sekund, zatory natomiast dla tramwajów blisko minutę a dla autobusu blisko 40 sekund, ustąpienie pierwszeństwa 15 a 30 sekund, zablokowany przejazd lub piesi nieco poniżej 20 sekund oraz inne powody w granicy 40 sekund.

Do oszacowania czasu przejazdu między skrzyżowaniami autorzy sugerują wzór nr 4 w publikacji. Do wyliczenia średniego czasu (podanego w sekundach) przejazdu między skrzyżowaniami potrzebne nam sa :

- odległość między skrzyżowaniami
- średni sumaryczny czas obsługi przystanków
- liczba przystanków między skrzyżowaniami

Spośród wszystkich opracowanych w tym raporcie publikacji, ta wydaje się dostarczać najwięcej konkretnych oraz bardzo przydatnych w naszej sytuacji danych. Dane statystyczne pozyskane z tej publikacji planujemy uwzględnić z dużą wagą przy późniejszej *“fuzji”* czynników.

“Modeling Bus Delays due to Passenger Boardings and Alightings.”

[http://onlinepubs.trb.org/Onlinepubs/trr/1983/915/915-002.pdf]

Publikacja “Modeling Bus Delays due to Passenger Boardings and Alightings.” autorstwa RICHARD P. GUENTHNER AND KUMARES C. SINHA mówi o szacowaniu opóźnień jakie generuje procedura zatrzymania się środka komunikacji miejskiej na przystanku. Co za tym idzie, głównym powodem powstawania tych opóźnień jest moment wysiadania/wsiadania pasażerów.

Czas opóźnienia jaki generują podróżujący komunikacją miejską nie zależy tylko i wyłącznie od ich liczby, jakby mogło się wydawać. Ważne jest także zwrócenie uwagi na takie czynniki jak :

- wiele osób wsiada/wysiada tylko na kilku przystankach, tj niektóre przystanki są pomijane podczas podróży np. przystanki “na żądanie”
- ilość osób podróżujących które są w podeszłym wieku
- ilość osób niepełnosprawnych (problemy z poruszaniem się, konieczność rozkładania specjalnej platformy umożliwiającej osobom na wózkach inwalidzkich dostanie się do autobusu/ tramwaju)

W publikacji znaleźć można wzór na oszacowanie czasu jaki generują pasażerowie:

$$\text{TIME}(z) = z\,[5.0 - 1.2\ln(z)] \qquad z \leqq 23 \qquad (13a)$$

$$\text{TIME}(z) = 1.2z \qquad z \geqq 24 \qquad (13b)$$

gdzie z to ilość pasażerów.

W publikacji znajdziemy także wzór na łączny czas oczekiwania w związku z zatrzymywaniem się na przystankach w przeliczeniu na mile (oryg. *“total dwell time per mile”*) dany numerem (19) który jest sumą czasów wyliczanych w równaniach (18) oraz (17). Równania (17) oraz (18) wyliczają łączny czas oczekiwania na przystankach w zależności od ilości osób zaangażowanych we wsiadanie/wysiadanie z pojazdu.

“Estimation of the bus delay at the stopping point on the base of traffic parameters”

[http://yadda.icm.edu.pl/baztech/element/bwmeta1.element.baztech-36408a18-a18d-4ef7-a430-984ad88c88d0/c/horbachov_naumov_kolii_estimation_aot-2015-3-02.pdf]

Ta publikacja przybliża temat czasu jaki jest potrzebny aby włączyć się do ruchu z miejsc postojowych komunikacji miejskiej, mówiąc bardziej po polsku, tak zwanych przystanków.

Według autorów, głównym czynnikiem wpływającym na ograniczenie możliwości włączenia się do ruchu jest aktualny potok pojazdów, konkretniej lider kolumny oraz pojazdy utrzymujące optymalną odległość od lidera. Do zamodelowania tego zjawiska autorzy wybrali mikroskopowy model *“Intelligent Driver Model”,* oraz rozkład Poissona (gęstość występowania pojazdów na drodze). Wychodząc z założenia istnienia 3 głównych przypadków:

1. A_0 - brak pojazdów mijających autobus przed czasem t oraz brak pojazdów mijających autobus w czasie (t; t + τ(v))
2. A_n - przed czasem t było dokładnie n pojazdów mijających autobus, w czasie (t; t + τ(v)) nie było już mijających pojazdów
3. B - $\sum_{n=0}^{inf} A_n$

Oraz długiej liście przekształceń, autorzy dochodzą do postaci określającej średni czas potrzebny na *“wydostanie się”* z przystanku:

$$\bar{T}_e = \bar{T}_l(v) - \tau(v) = \frac{1}{\lambda} \cdot \left(e^{\tau(v) \cdot \lambda} - 1 \right) - \tau(v) \qquad (46)$$

gdzie:

- **λ -** basic distribution parameter (traffic flow intensity), veh./sec.
- τ(v) = v/a - v - prędkość jadącej kolumny, a przyspieszenie autobusu

W przypadku sprawdzonym przez autorów, zgodność modelu z pomiarami wyniosła 83.2% co jest wystarczającym przybliżeniem dla potrzeb naszego projektu.

“An Approach to Urban Traffic State Estimation by Fusing Multisource Information”

[http://ieeexplore.ieee.org/document/5169913/]

Publikacja o powyższym tytule ukazuje próbę podejścia do scalania danych zebranych z różnych źródeł w jedną informację na podstawie przykładowych danych zbieranych na temat ruchliwości Szanghajskich ulic. Autorzy publikacji w dyskusji nad wynikami ustalili, że wynik działania ich algorytmu który omówimy poniżej, był dokładniejszy, niż dane estymowane na tak zwanych “GPS-based methods”, co brzmi bardzo obiecująco biorąc pod uwagę założenia naszego projektu.

Sam algorytm opracowany przez autorów publikacji jest podzielony na cztery główne moduły:

1. Preprocessing - wygładzanie danych (filtr Kalmana) oraz oceny wariancji (?) (oryg. “*variances evaluating”*)
2. Kalkulacja wag dla danych z różnych źródeł (*“evidence theory”*)
3. Faktyczne łączenie danych
4. Postprocessing - decyzja o poprawności oraz przydatności wygenerowanych danych

Cały schemat blokowy algorytmu dołączamy jako załącznik 1 na końcu tego raportu.

Metody wybrane przez twórców powyższego algorytmu do jego wykonania wydają się być mocno zasobochłonne (filtr Kalmana do wygładzania danych, ekstrapolacja na podstawie *“evidence theory”*) i mamy wątpliwości czy będziemy w stanie tak doskalować nasz backend aby był w stanie efektywnie w czasie rzeczywistym przeliczać taką ilość danych (zwłaszcza biorąc pod uwagę, że nas backend jest postawiony na małym klastrze raspberry pi). Jesteśmy pewni natomiast, że chcemy przynajmniej w “okrojonym” stopniu skorzystać z metody przedstawionej w tej publikacji.

“Wpływ warunków atmosferycznych na prędkość pojazdów.”

[http://yadda.icm.edu.pl/baztech/element/bwmeta1.element.baztech-article-BAT1-0036-0028/c/nr_2_45-53.pdf]

Publikacja “Wpływ warunków atmosferycznych na prędkość pojazdów.” autorstwa Tomasza Szczuraszka przedstawia badania, które ukazują, że pogoda ma bardzo duży wpływ na szybkość poruszających się pojazdów.

Najistotniejsze dla naszego projektu cechy atmosferyczne i ich wpływ na prędkość pojazdu to:

- ciśnienie atmosferyczne (tabela 1)
- temperatura powietrza (tabela 3)
- opady deszczu i śniegu (tabela 5)
- mgła (tabela 7)

Wyniki badań jasno pokazują, że warunki atmosferyczne mają znaczący wpływ na samopoczucie kierowców czy pogorszenie się stanu drogi. Za wzorcowe warunki przyjmujemy :

- ustabilizowane ciśnienie wyżowe
- słonecznie, bez zachmurzenia nieba(słońce wysoko nad horyzontem)
- temperatura powietrza od 20 do 23 C
- niska wilgotność powietrza <50%
- bezwietrznie lub wiatr poniżej 3m/s.

Wszystkie dane statystyczne odnoszące się do zmniejszenia się prędkości pojazdów, zawarte są w tabelach dołączonych poniżej w załączniku (załącznik 2).

Pomimo faktu iż są to dane uzyskane przez pomiary na odcinkach dróg, które różnią się od rzeczywistego stanu dróg w aglomeracji, gdzie chcemy dokonać symulacji, pomimo to, na nasze potrzeby mogą okazać się wystarczające. Nie będą w pełni oddawać stanu rzeczywistego, natomiast aby dokonać bardzo dokładną symulację potrzebna by nam była szeroka wiedza z wielu dziedzin nauki, której nie uda nam się posiąść na potrzeby projektu.

Reasumując choć dane zawarte w artykule nie są idealnie dopasowane do naszej sytuacji, wydają się być wystarczająco dobrym przybliżeniem aby z powodzeniem użyć ich w naszym projekcie.

“Empirical Studies on Traffic Flow in Inclement Weather”

[http://ops.fhwa.dot.gov/publications/weatherempirical/weatherempirical.pdf]

Publikacja bada wpływ aktualnych warunków atmosferycznych na ruch drogowy na terenie trzech metropolii: Minneapolis, Baltimore i Seattle. Obszary, na których skupia się niniejsze opracowanie są następujące:

- wpływu opadów atmosferycznych na makroskopowe parametry ruchu drogowego
- badanie regionalnych różnic w reakcji na opady
- wpływ ograniczonej widoczności na ruchu

Samo badanie polegało na wykorzystaniu modelu Van Aerde, dobraniu odpowiednich parametrów, predykcji ruchu, a następnie jego weryfikacji na podstawie odpowiednich danych empirycznych, których źródłem były stacje pogodowe oraz badanie rzeczywistego ruchu. Szczegółowy opis modelu oraz dobranych parametrów znajduje się w publikacji. Okazuje się, że o ile wpływ deszczu na ruch we wszystkich badanych miastach był mniej więcej podobny, to opady śniegu miały o wiele większy wpływ na ruch w tych miastach gdzie zazwyczaj spada go mniej.

Wydaje się, że wyniki uzyskane dla Minneapolis mogą być dla nas przydatne z racji podobnego klimatu. Warto również wykorzystać model Van Aerde, który pozwala na estymację ruchu drogowego w zależności od pogody, a artykuł ten stanowi dobry przykład jego wykorzystania.

Wpływ pogody oraz widoczności na przepustowość dróg oraz prędkość pojazdów zamieszczone są w załączniku (załącznik 3).

“Handbook of Computer Models for Traffic Operation Analysis”

[https://babel.hathitrust.org/cgi/pt?id=mdp.39015040706627;view=1up;seq=1]

Książka ta opisuje 10 przykładowych modeli komputerowych służących do analizy oraz symulacji ruchu. Dla każdego modelu omówiono sposoby jego użycia, wstępne wymagania, opis algorytmu, zasadę działania algorytmu oraz przykładowe zastosowania. Wymienione w publikacji modele to:

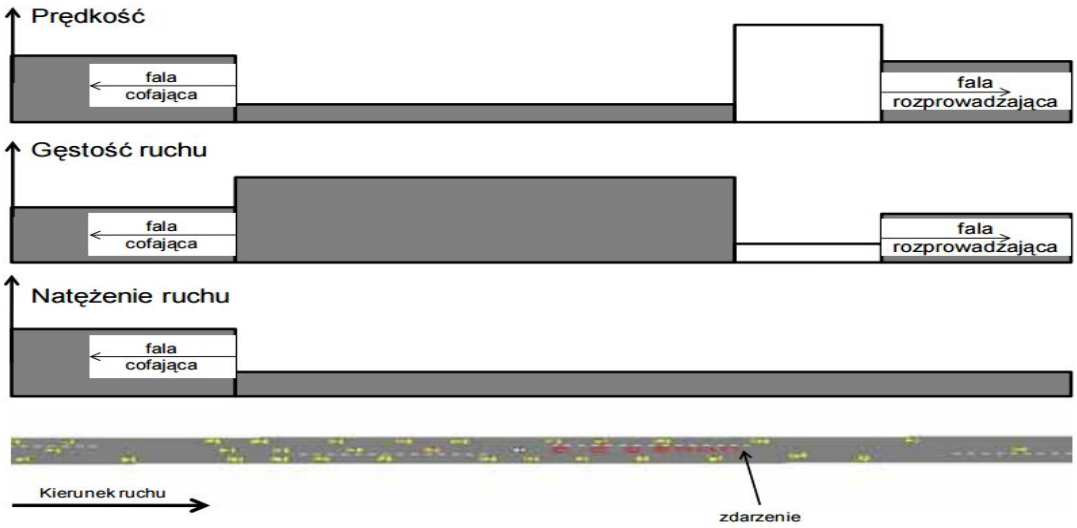
- soap (intersection optimization model) - optymalizacja ruchu ze skrzyżowaniami
- texas (intersection simulation model) - symulacja ruchu przez skrzyżowania
- passer II(80) (arterial optimization model) - optymalizacja poprzez upłynnienie ruchu pojazdów
- passer III (diamond interchange optimization model) - optymalizacja ruchu na skrzyżowaniach w kształcie diamentu (węzeł typu WB)
- sub (arterial bus simulation model) - symulacja ruchu autobusów
- transyt-7f (network optimization model) - optymalizacja ruchu pojazdów
- sigop III (network optimization model) - optymalizacja ruchu pojazdów
- netsim (network simulation model) - symulacja sieci drogowej
- rprifre (freeway simulation model) - optymalizacja ruchu autostradowego
- freq3cp (freeway simulation model) - symulacja ruchu autostradowego

Z racji tematu naszego projektu najbardziej będą interesowały nas modele, które będą potrafiły symulować ruch pojazdów komunikacji miejskiej oraz ruch drogowy. Do nich na pewno zaliczają się: sub (simulation of urban buses) oraz netsim. Wadą tej publikacji jest użycie przestarzałych języków programowania (fortran IV), niemniej jednak opisy algorytmów wciąż są aktualne.

“Wpływ lokalnych ograniczeń przepustowości na rozkład ruchu”

[http://www.sitk.org.pl/wp-content/uploads/2015/07/148.pdf]

Publikacja Tomasza Dybicza zawiera podsumowanie i opis metod wykrywania oraz wyznaczania lokalnych ograniczeń przepustowości (LOP). Aktywne LOP można scharakteryzować jako przekrój drogi, od którego rozchodzą się fale zaburzeń ruchu, których odzwierciedleniem są zmiany warunków ruchu w zależności od zmieniających się odległości w stosunku do LOP.



Rys. 1. Schemat powstawania fal zaburzeń ruchu

Fale cofające mają bardzo negatywne oddziaływanie na warunki ruchu w sieci drogowej. W wyniku ich oddziaływania zmniejsza się prędkość pojazdów, a jednocześnie wzrasta gęstość ruchu. Powstawanie fal cofających powoduje spadek przepustowości ciągu drogowego, na którym dochodzi do aktywacji LOP. Przepustowość ta jest często mniejsza, niż teoretyczna przepustowość przekroju drogi w miejscu LOP.

W publikacji autor opisuje metodykę wyznaczania przepustowości zaproponowana przez Banksa opartej na założeniu, że natężenie ruchu jest odwrotnością średniego odstępu pomiędzy poprzedzającymi pojazdami. Banks rozkłada czas odstępu między pojazdami na dwie składowe: czas przejazdu - jest to czas w którym pojazd mija wyznaczony punkt oraz czas pomiędzy pojazdami - jest to luka czasowa mierzona od momentu minięcia przekroju przez tylny zderzak pierwszego pojazdu do momentu minięcia tego przekroju przez przedni zderzak drugiego pojazdu. Stąd bierze się wniosek, że maksymalne natężenie na krytycznym pasie ruchu przed LOP jest funkcją średnich wartości: czasu przejazdu „i” pomiędzy pojazdami oraz udziału ruchu na tym pasie.

Przydatne mogą okazać się wzory na obliczanie maksymalnego natężenia ruchu w trakcie zwężeń czyli tzw. lokalnych ograniczeń przepustowości, które mogą być spowodowane przez wypadki, remonty

“Minor Road Traffic Delays at Priority Junctions on Low Speed Roads in Suburban Areas”

[https://www.dropbox.com/s/pelrq2wdtqml4ud/P9%20Minor%20Road%20Traffic%20Delays%20at%20Priority%20Junctions%20on%20Low%20Speed%20Roads%20in%20Suburban%20Areas%20(JT).pdf?dl=0]

Powyższa publikacja porusza temat badania opóźnień w ruchu miejskim spowodowanych skrzyżowaniami. Autorzy przyjmują założenie, że każde skrzyżowanie można przedstawić jako jedno z dwóch poniżej wymienionych:

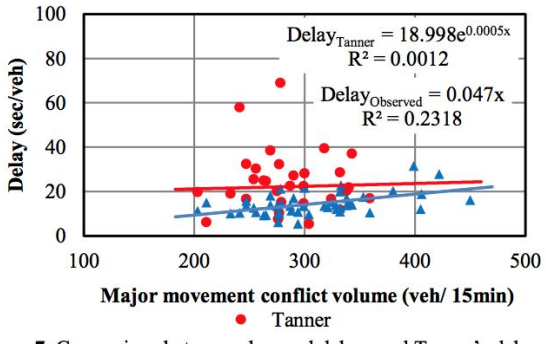
1. *“Two Way Stop Controlled”* (np skrzyżowanie ul. Reymonta i ul. Piastowskiej)
2. *“All Ways Stop–Controlled” (np skrzyżowanie ul. Armii Krajowej oraz ul. Piastowskiej)*

W publikacji do oszacowania opóźnienia, autorzy posługują się dwoma modelami - Tannera oraz “Highway Capacity Manual”.

Badania były prowadzone w oparciu o obserwację, tj. nagranie zachowania ruchu na kilku skrzyżowaniach w różnych porach dnia a następnie kilkukrotna analiza tych nagrań pod kątem konkretnych czynników:

- Vehicle arrival times for major road traffic;
- Vehicle arrival and departure times for vehicles on the minor approaches; and
- Traffic composition

Z przeprowadzonych analiz porównawczych, nas interesują wykresy porównujące obliczone wartości z faktycznymi zaobserwowanymi, oraz tabela średnich czasów oczekiwania:



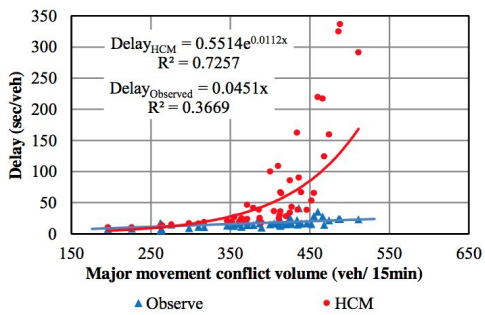


Figure 10 Comparison between observed delays and HCM’s delay model for Kebudayaan/ Kebudayaan3 junction

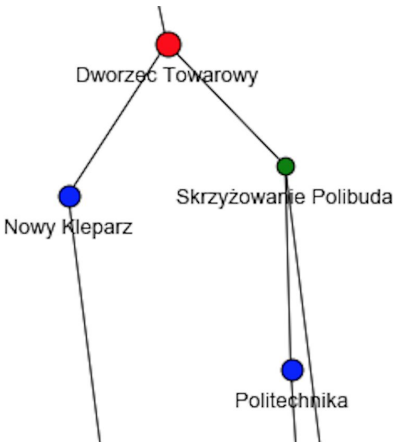
Table 1 t–Test results for Tanner’s model versus observed delay

	Delay based on Tanner’s Model	Observed Delay
Mean	3.76	5.91
Variance	3.23	11.23
Observations	4.00	4.00
t-Test	0.31	

Dla nas, istotną informacją informacją jest stopień dopasowania R^2 który w naszej ocenie jest stosunkowo niski, co powinno wpłynąć na nasze ograniczone zaufanie do modeli matematycznych, oraz przekonać nas bardziej do danych statystycznych.

4. Opis stworzonego modelu

Ponieważ ruch tramwajów odbywa się między stałymi punktami na znanych z góry określonych trasach, naturalnym pomysłem na reprezentację takiego modelu jest graf (nieskierowany, ponieważ tramwaje jeżdżą w dwie strony pomiędzy przystankami) gdzie wierzchołkami są kolejne przystanki posiadające swoje współrzędne geograficzne oraz nazwę. Nasz model zakłada następujące uproszczenie: w wypadku gdy dany przystanek pod jedną nazwą występuje w kilku miejscach, patrz obrazek po prawej, uśredniliśmy go do jednego przystanku mniej więcej na środku ciężkości wszystkich tych przystanków. Natomiast, rozpatrzenie samych przystanków byłoby zbyt dużym uogólnieniem, ponieważ istotą rzeczą są również skrzyżowania linii tramwajowych, które również postanowiliśmy dodać jako wierzchołki naszego grafu.



Na reprezentacji grafu generowanej przez naszą aplikację (fragment po lewej) widać:

- na czerwono pętle tramwajowe,
- na zielono skrzyżowania tramwajowe nie będą ce przystankiem,
- na niebiesko zwykły przystanek

Następną istotną rzeczą są krawędzie po których będą się poruszać tramwaje w naszym grafie. Każda taka krawędź będzie tworzona na podstawie trasy tramwajów, początek takiej krawędzi to współrzędne przystanku z którego dany tramwaj wyjeżdża, koniec takiej krawędzi to współrzędne przystanku do którego tramwaj zmierza. Taka krawędź trzyma kilka istotnych informacji:

1. Jakie linie tramwajowe po niej jeżdżą
2. Jaka jest długość takiej krawędzi (odległość w linii prostej na mapie, nie posiadamy informacji o faktycznej długości, wydaje się nam, że będzie to dobre przybliżenie)
3. Dwie kolejki w których będą trzymane tramwaje (każda kolejka w jedną stronę), aby mogły one oddziaływać na siebie nawzajem (tramwaje nie mogą się wyprzedzać, pierwszy tramwaj ogranicza prędkość całej kolumny itd.)

Każdy tramwaj jeżdżący po tym grafie ma zdefiniowaną swoją prędkość, uzależnioną od wielu czynników ograniczających:

1. Prędkość tramwaju który jest wcześniej w kolumnie, jeżeli ich odległość od siebie jest odpowiednio mała
2. Warunki pogodowe
3. Aktualne warunki drogowe (zagęszczenie ruchu na danym obszarze)
4. Statystyczna ilość osób wsiadających i wysiadających o danej porze

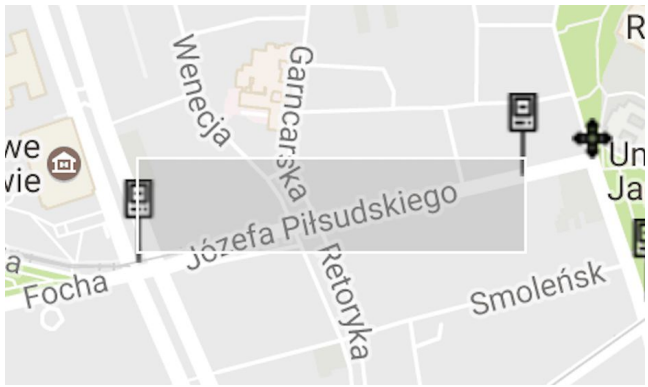
Uwzględnione przez nas czynniki ograniczające można podzielić na “globalne” oraz “lokalne”.

Czynniki globalne, to takie, które dotyczą wszystkich tramwajów w takim samym stopniu, w ten sposób można traktować:

- pogodę (raczej taka sama w obrębie jednego miasta)
- statystyczną ilość podróżnych (nie dysponujemy środkami aby zmierzyć faktyczną ilość osób)

Czynniki lokalne, to takie, które dotyczą bezpośrednich jednostek, takimi czynnikami są:

- zagęszczenie pojazdów na danej trasie (obliczane jako średnia z zagęszczenia w prostokącie wyznaczonym przez przystanek poprzedni oraz następny)
- oddziaływanie między tramwajami koło siebie w jednej kolejce



Podsumowując:

Na wyjściu z naszego modelu chcemy dostać aktualne opóźnienia tramwajów Krakowskiej Komunikacji Miejskiej w celu ustalenia najbardziej prawdopodobnego czasu odjazdu tramwaju z danego przystanku. W takiej postaci aplikacja była by gotowa do integracji np. z JakDojade.pl które by informowało o aktualnych opóźnieniach.

Na wejściu, bierzemy tramwaj wyjeżdżający z pętli o zadanej porze, którego prędkość jest modyfikowana przez wpływ czynników zewnętrznych, takich jak pogoda, ilość ludzi, zagęszczenie ruchu drogowego ale także innych tramwajów.

5. Opis użytych algorytmów z przeglądu literatury

Wszystkie niżej wymienione algorytmy zostały omówione w przeglądzie literatury, dlatego podarujemy ponowne ich omawianie w tej sekcji. Poniżej prezentujemy spis algorytmów na które się zdecydowaliśmy.

W modelowaniu czasów potrzebnych na postój na przystanku skorzystamy z danych statystycznych dostarczonych z publikacji “ANALIZA CZYNNIKÓW WPŁYWAJĄCYCH NA PRĘDKOŚĆ POJAZDÓW TRANSPORTU ZBIOROWEGO NA PRZYKŁADZIE GDAŃSKA” oraz wzorach znalezionych w publikacji “Modeling Bus Delays due to Passenger Boardings and Alightings”.

Wpływ oddziaływania tramwajów na siebie, na jednej krawędzi, opracujemy na podstawie opracowania “fali cofającej” opisanej w “Wpływ lokalnych ograniczeń przepustowości na rozkład ruchu”.

Uwzględniając dane pogodowe, posłużymy się danymi statystycznymi wynikającymi z tabeli w załączniku nr 3 która odnosi się do publikacji “Empirical Studies on Traffic Flow in Inclement Weather”

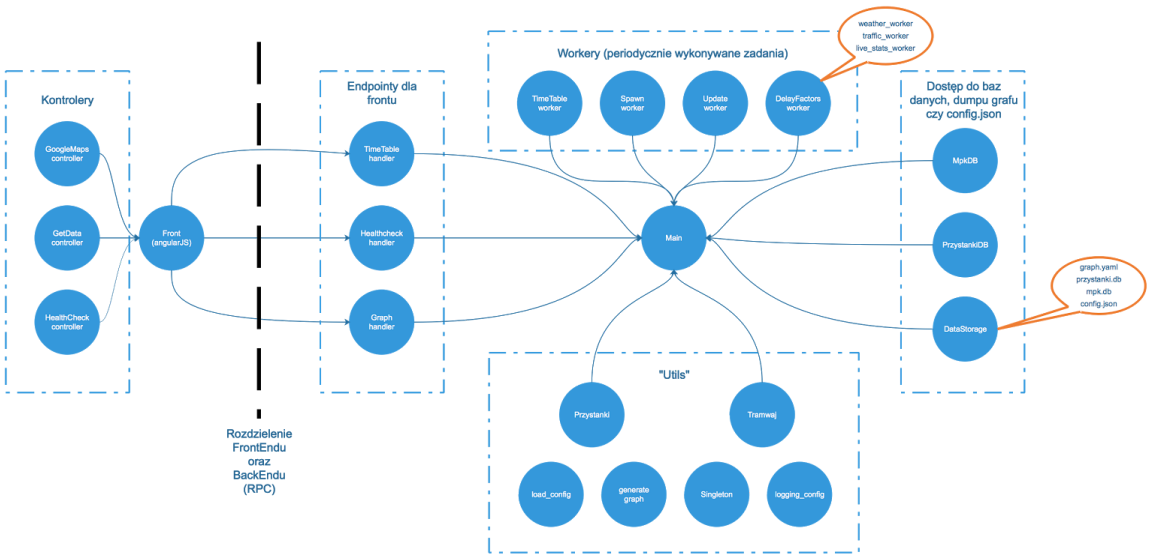
Do złożenia wszystkich powyższych czynników postaramy się użyć algorytmu znalezionego w publikacji “An Approach to Urban Traffic State Estimation by Fusing Multisource Information”.

W związku z niewystarczającą ilością czasu, implementacje poszczególnych części mogą odbiegać od tych opisanych w publikacjach na rzecz ich prostszych odpowiedników.

6. Opis zastosowanych technologii

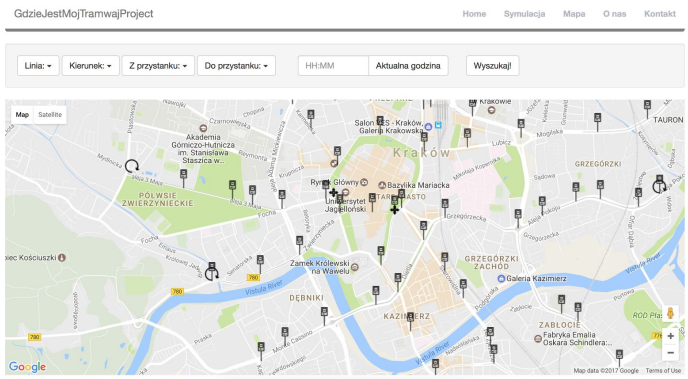
W naszym projekcie zdecydowaliśmy się zbudowanie aplikacji webowej z całkowitym rozdzieleniem części frontowej od programu realizującego wyżej omówiony model.

Aby prościej było omówić strukturę naszej aplikacji, poniżej prezentujemy uproszczony diagram użytych modułów. (Większą wersję tego obrazka załączamy na końcu tego dokumentu jako załącznik 5.)



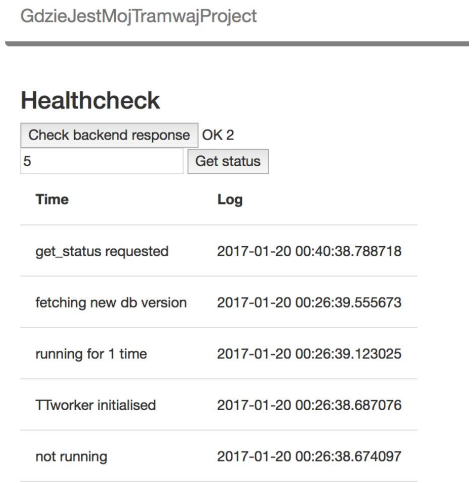
Frontend

Ponieważ jednym z kluczowych aspektów naszej symulacji była wizualizacja danych, oraz naturalnym pomysłem było przedstawianie obecnej sytuacji tramwajów na mapie, do tego celu wybraliśmy Google Maps. Co za tym idzie, idealnym kandydatem na użyty framework frontowy był AngularJS, tego samego wydawcy co mapy, co miało gwarantować łatwą integrację z mapami. Jak się okazało, AngularJS nie miał natywnego wsparcia dla map, co bardzo nas zdziwiło, niemniej jednak użyliśmy zewnętrznej biblioteki która sprawdziła się całkiem dobrze.



Ponieważ AngularJS implementuje wzorzec projektowy MVC, stworzyliśmy 3 kontrolery.

1. Google Maps controler - odpowiedzialny za wyświetlanie objektów na mapie
2. Get Data controller - wysyłający polecenia RPC do backendu po dane
3. HealthCheck controller - sprawdzający czy backend wciąż odpowiada i podaje dane, potencjalnie użyty do wykrycia awarii, czy mierzenia uptime'u usługi.



Przy komunikacji frontendu oraz backendu naszej aplikacji, zdecydowaliśmy się na imlpementacje okrojonej wersji JSON-RPC. Dla naszych potrzeb, backend przyjmuje calle tylko metodą POST, GET jest wykorzystywany tylko do pobierana plików statycznych, takich jak html, css czy js. Umożliwia to nam testowanie aplikacji za pomocą prostych narzędzi, takich jak cURL, bez konieczności czekania na działający front. Dzięki temu, mogliśmy efektywnie zrównoleglić pracę.

```
[~] curl -X POST -H "Accept:application/json" -H "Content-Type:application/json"
 http://0.0.0.0:8888/healthcheck
{"status": "OK", "number": 5}%
```

Backend

Część backendowa naszej aplikacji jest oparta o Pythona 3.5 oraz framework Tornado zapewniający serwer HTTP oraz wspierający podejście asynchroniczne. Asynchroniczność w naszym wypadku jest o tyle ważna, iż wszystkie workery wykonujące periodycznie swoje operacje nie blokują całej aplikacji, dalej jest ona responsywna i odpowiada na żądania HTTP.

W głównym module naszej aplikacji tworzone są wszystkie handlery zapytań HTTP oraz obiekty workerów które zostaną omówione później. Ponieważ port na którym nasłuchuje aplikacja jest ustawiany z przekazywanych argumentów, jesteśmy w stanie puścić naszą aplikację równocześnie w różnych wariantach, oraz porównywać w czasie rzeczywistym różnice wynikające z przekazanych ustawień (config.json)

Udostępnione endpointy

Nasza aplikacja na tyle się rozrosła, że zdecydowaliśmy się zdefiniować osobne end-pointy pod które może "pukać" klient, są to:

1. /healthcheck - handler obsługujący healhcheck aplikacji
2. /mpk_db/(.*) - handler do pobierania informacji na temat przystanków
3. /graph_api/(.*) - handler dostarczający informacje o stanie grafu

O wszystkich metodach implementowanych przez te handlery można przeczytać w dokumentacji API która znajduje się w naszym repozytorium (link w załącznikach jako załącznik 6.)

Workery

Biorą pod uwagę to, że nasza symulacja musi “żyć własnym życiem”, potrzebne było znalezienie modułu który by pozwalał na cykliczne wykonywanie pewnych zadań oraz aby był nieblokujący. Idealnym w tym wypadku okazał się YieldPeriodicCallback zgodny z Tornado. Za jego pomocą stworzyliśmy kilku robotników (*ang worker, tłumaczenie autorskie*), których najważniejsze cechy przedstawiamy w tabeli poniżej:

Nazwa workera	Częstotliwość wykonywania	Wykonywane zadania
TimeTableWorker	24 h	TimeTable worker odpowiedzialny jest za ściągnięcie najbardziej aktualnych rozkładów z bazy danych mpk
SpawnWorker	1 min	SpawnWorker wybudza się co minutę i sprawdza czy z którejś pętli powinien w tym momencie wyjechać tramwaj, zleca tworzenie obiektów typu tramwaj
UpdateWorker	3 sek	Update worker co 3 sekundy przechodzi po liście wszystkich tramwajów i update’uje ich stan do najbardziej aktualnego. W tym samym czasie odświerza sytuacje na kolejkach w grafie.
DelayFactorWorker	1 min	Worker odpowiedzialny za obliczanie aktualnej prędkości każdego tramwaju na podstawie waunktów “globalnych” jak pogoda, ale też “lokalnych” takich jak zagęszczenie ruchu w konkretnym obszarze

Warto bliżej się przyjżeć DelayFactorWorkerowi, gdyż implementuje on przeliczanie wszystkich czynników ograniczających.

- Pogoda - jest pobierana cyklicznie co minutę z openweathermap.org przez publiczne api dla miasta Krakowa a następnie trzymana w obiekcie. Na żądanie obliczany jest znormalizowany współczynnik dla temperatury, deszczu, śniegu, gotowy do wymnożenia z prędkością tramwaju. Dodatkową opcją którą bierzemy pod uwagę, gdybyśmy kiedyś planowali symulować większy okres w przód, jest ściąganie prognozy pogody na najbliższe 5 dni.
- Ruch uliczny - pobierany dla konkretnego tramwaju, korzysta z api map HereWeGo. Ponieważ to api udostępnia ściąganie danych o trafficu tylko dla prostokąta, uśredniamy wszystkie “jam factors” które dostaniemy w tym prostokącie.
- Statystyczna ilość osób - jest pobierana z configa, w którym znajdują się dane statystyczne of MPK. Na podstawie tych danych obliczana jest średnia ilość ludzi w danym tramwaju o danej porze.
- Kolejki na krawędziach grafu - ich obsługą zajmuje się UpdateWorker który zapewnia, odpowiednie przeskalowanie prędkości tramwajów w zależności od sytuacji w kolejce, zgodnie z falą cofającą.

Storage

Jak każdy projekt, potrzebujemy przechowywać pewne dane. Baza danych ściągana przez TimeTableWorkera z proxowanego API aplikacji mobilnej MPK (nie udostępniają publicznego endpointu) jest w formacie SQLite (Zapewnia ona tylko dostęp do aktywnych wariantów rozkładów jazdy na dany dzień, brak szczegółowych informacji o odjazdach z przystanków). W związku z tym nie chcieliśmy komplikować dostępu do danych i kolejną bazę danych “przystanki” (Baza danych trzymająca dane na temat aktualnych odjazdów z przystanków) też utrzymaliśmy w formacie SQLite. Dostęp do tych danych jest realizowany przez moduły mpk_db.py oraz przystanki_db.py w których są zawarte odpowiednie metody do wybierania odpowiednich danych za pomocą biblioteki sqlite3.

Konfigurację każdej instancji naszego projektu trzymamy w pliku conf/config.json zaczytywanego w głównym module jako Singleton.

Ostatnim ważnym elementem z przechowywanych danych jest plik graph.yaml. Jest to dump kompletnego grafu, wygenerowanego przy zapytaniu o metodę “generate_graph” w graph_api handlerze. Format danych, tj. yaml jest dla nas kompletnie przypadkowy, biblioteka której używamy do implementacji grafu (NetworkX) ma możliwość exportu oraz zaczytania w tym formacie. Nie mniej jednak jest on wygodny ze względu na czytelność. Po otwarciu tego pliku, od razu widać obiekty które są trzymane przez każdy wierzchołek czy krawędź.

Utils

W module utils znajdują się moduły których nie mogliśmy przyporządkować nigdzie indziej w strukturze organizacyjnej naszego kodu. Znajdują się w nim takie rzeczy jak:

- implementacja Singletona
- implementacja singletonu configa dostępnego jako Pythonowy dict dla innych obiektów
- metoda konfigurująca wygląd logów aplikacji
- implementacja logiki generowania grafu z dostępnych danych
- implementacja klasy Przystanki trzymającą informacje o przystankach jako singleton dostępnego jako Pythonowy dict dla innych obiektów
- implementacja klasy Tramwaj reprezentującej tramwaj, a także fabryki tramwajów

Pozwolimy sobie pominąć pierwsze trzy punkty powyższego zestawienia, gdyż nie wnoszą one nic ciekawego do opisu projektu.

Generowanie grafu

generate_graph.py to moduł mający jedno zadanie, wygenerowanie grafu na podstawie posiadanych danych, zaczytywane są numery linii które mają zostać uwzględnione z configu, ich trasy na podstawie aktywnych w danym dniu wariantów oraz dane o położeniach wszystkich przystanków tramwajowych (przygotowane pół-ręcznie, po dokładniejszy opis zapraszam do tego: `6cb82c3` commitu w naszym repozytorium) z pliku przystanki_0_159.json. Wszystkie przystanki są dodawane do grafu jako wierzchołki z rozróżnieniem na pętle, przystanki oraz skrzyżowania. Następnie dodawane są krawędzie do grafu. Wyciągane one są z tras tramwajów, tworzone są dla każdej pary wierzchołków w danej trasie. Jeżeli istnieje już taka krawędź, to jest ona pomijana. Każda krawędź posiada kilka właściwości, takich jak:

1. lista linii które mogą się poruszać po danej krawędzi
2. dwie kolejki (Python deque) w których będą trzymane tramwaje
3. długość danej krawędzi w metrach

Na końcu całość jest zapisywana do pliku graph.yaml oraz graph.png aby można było zweryfikować poprawność wygenerowanego grafu. Plik PNG z wygenerowanym grafem można znaleźć w naszym repozytorium oraz na końcu tego dokumentu jako załącznik 7.

Przystanki

przystanki.py jest singletonem zapewniającym wszystkim obiektom dostęp do danych z grafu ale też surowych danych o przystankach w postaci Pythonowego dicta. Implementuje też różne funkcje pomocnicze, takie jak “pobierz wszystkie krawędzie dla danej linii” itp.

Tramwaje

W module tramwaj.py znajdują się dwie kluczowe rzeczy dla tego projektu. Klasa której obiekty reprezentują dany tramwaj oraz fabryka tych tramwajów. Wzorzec projektowy fabryka znalazł się tam tylko dlatego, iż nie chcieliśmy aby każda instancja klasy Tramwaj otwierała połączenie do baz danych w momencie tworzenia obiektu.

Projektując klasę tramwaj staraliśmy się jak najlepiej odwzorować obiekt jakim jest faktyczny tramwaj. Każdy obiekt tej klasy posiada pola:

1. Numer linii
2. Swoją trasę

3. Swoją prędkość
4. Swoją aktualną pozycję geograficzną
5. Swój stan (jedzie/stoi na przystanki)
6. Informacje o następnym przystanku
7. Informacje o poprzednim przystanku
8. Czas jaki powinien stać na przystanku

A także metody:

1. calculate_position - oblicza aktualne położenie tramwaju na podstawie posiadanych właściwości prędkość, następny przystanek itd.
2. calculate_next_stop - znalezenie następnego przystanku na który ma się udać
3. calculate_distance - oblicza jaki dystans przebył tramwaj od ostatniego odświeżenia przez update workera
4. stop - wywoływana w momencie zatrzymania się tramwaju na przystanku, powoduje ona zdjęcie referencji do tramwaju na kolejce z poprzedniej krawędzi i dodanie jej do nowej oraz ustawienie prędkości na 0
5. start - wystartowanie tramwaju z przystanku

Cykl życia tramwaju

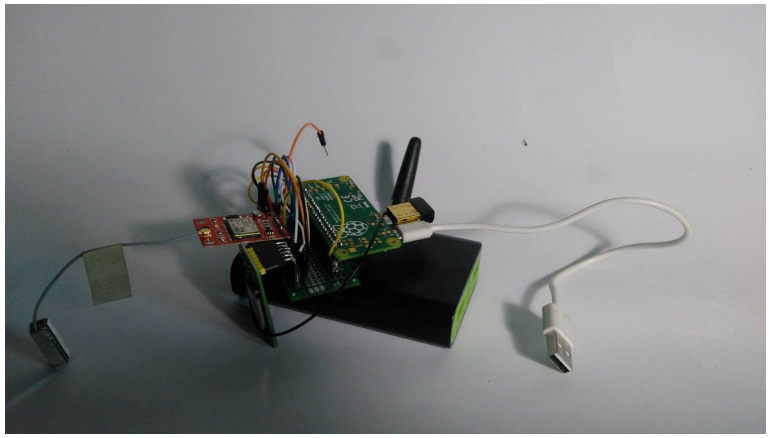
Życie każdego obiektu klasy Tramwaj zaczyna się w momencie gdy SpawnWorker po wybudzeniu znajdzie, że tramwaj X powinien wyjechać z pętli Y. W tym momencie SpawnWorker zleca stworzenie obiektu klasy Tramwaj do fabryki TramwajFactory. Wytworzony obiekt jest dodawany do listy istniejących tramwajów oraz wpychany na odpowiednią kolejkę w grafie.

Następnie, taki tramwaj czeka aż UpdateWorker wykona na nim metodę calculate_distance co około 3 sekundy z dokładnością do opóźnień spowodowanych ilością tramwajów na mapie itd. W tym momencie tramwaj przelicza jaką prędkość powinien mieć w zależności od aktualnych warunków podanych przez DelayFactorWorkera. Następnie obliczy odległość jaką przebył od ostatniego odświeżenia przez UpdateWorkera. Gdy taki tramwaj dotrze do przystanku (jest od niego bliżej niż 10m) staje na zadany czas, w zależności od statystycznego obłożenia w danym czasie oraz jest usuwany z kolejki na danej krawędzi. Po skończeniu postoju na danym przystanku wyrusza on w drogę do następnego, jest on wpychany do następnej kolejki na następnej krawędzi.

Po pokonaniu całego dystansu, obiekt ten woła metodę self_destruct która odznacza go jako usuniętego i przy następnym przebiegu SpawnWorker usunie go z listy aktywnych tramwajów.

W związku z wymaganiami zaliczenia, cały kod dołączamy na płycie CD, natomiast ponieważ cały nasz kod jest trzymany w repozytorium na GitHub oraz projekt dalej się rozwija, zachęcamy do klonowania kodu z repozytorium w celu posiadania najbardziej aktualnej wersji. Link do repozytorium to załącznik numer 8.

7. Kalibracja oraz walidacja modelu



W celu kalibracji modelu chcemy się posłużyć efektem powstałym z projektu realizowanego w ramach przedmiotu “Wzorce i problemy projektowe”. Wynikiem tamtego projektu jest urządzenie mobilne, oparte o mikrokomputer RaspberryPi Zero, które określa swoje położenie za pomocą modułu GPS oraz wysyła tą pozycję za pomocą modułu GSM.

Dla potrzeb kalibracji wystawimy specjalny endpoint w którym to urządzenie będzie raportować swoją pozycję w danym czasie oraz numer linii tramwajowej w której aktualnie się znajduje. Przy odpowiednio licznej ilości iteracji tego procesu, dostaniemy rzetelne dane statystyczne na temat średniej prędkości danego tramwaju co pozwoli nam na dobre dopasowanie reszty składników liczących tą prędkość.

8. Wyniki testów środowiskowych

W celu zweryfikowania dokładności naszej symulacji postanowiliśmy dokonać testów na kilku liniach. Test polegał na przejechaniu się tramwajem, z odpaloną symulacją w tle, a następnie porównywaniu rzeczywistej godziny przyjazdu z naszymi wynikami. Poniższa tabela zawiera godziny w jakich dany tramwaj przybywał na przystanek oraz godzinę o której tramwaj był na przystanku wg naszej symulacji.

linia i przystanek	Godzina rzeczywista	Godzina wg symulacji
14 urzędnicza	21:26	21:27
14 plac inwalidów	21:28	21:28
14 batorego	21:29	21:30
14 bagatela	21:30	21:32
14 basztowa	21:34	21:34
14 dworzec główny	21:36	21:36
14 lubicz	21:38	21:37:41

14 rondo mogilskie	21:40	21:39:36
8 solvay	21:49	21:49
8 bozego milosierdzia	21:51	21:51:31
8 lagiewniki	21:54	21:54:24
8 rzemieślnicza	21:55	21:56:26
8 rondo mateczne	21:57	21:58
24 bronowice	22:07	22:07
24 głowackiego	22:09	22:09
14 plac inwalidów	10:28	10:41

Jak przedstawiają powyższe wyniki nasza kalibracja modelu wręcz doskonale sprawdza się dla godzin wieczornych. Niestety jak pokazuje ostatni wynik w dzień symulacja odbiega od rzeczywistości co naszym zdaniem jest spowodowane niepoprawnym obliczaniem czasu postoju na przystanku tramwaju. Bazujemy tylko na uśrednionych danych które udostępnia mpk więc albo błąd leży też po ich stronie albo jest to kwestia błędu implementacyjnego. W przyszłości wierzymy, że uda się rozwiązać ten problem.

9. Wnioski

Okazuje się, że zaprojektowanie modelu komunikacji miejskiej jest sprawą dość skomplikowaną, głównie przez dużą liczbę czynników, które mają wpływ na szybkość ruch pojazdów. Niewątpliwie największy wpływ na prędkość zarówno autobusów jak i tramwajów mają warunki pogodowe, natężenie ruchu oraz liczba osób korzystających z pojazdów. Dlatego właśnie te czynniki zostały zaimplementowane. Dostępne dane miały charakter statystyczny i często były na tyle ogólne, że uniemożliwiały podanie indywidualnego parametru dla danego pojazdu np. przyjęty wpływ warunków atmosferycznych na ruch każdego pojazdu był taki sam w całym Krakowie, gdyż serwisy pogodowe nie udostępniają bardziej szczegółowych danych. Kolejnym problemem był z przewidywanie sytuacji nagłych, gdyż na podstawie dostępnych informacji nie da się ich w żaden sposób przewidzieć.. Informację o tego typu zdarzeniach można uzyskać dopiero wtedy gdy się już wydarzą np. za pomocą mpk RSS da się uzyskać dane na temat awarii pojazdów komunikacji miejskiej. Mimo wszystko wyniki symulacji okazały się zaskakująco zgodne z rzeczywistością, co pokazuje, że w standardowych warunkach pewne uproszczenia mają sens i spisują się zaskakująco dobrze.

Zdecydowana większość z obecnie dostępnych usług tematycznie związanych z naszym tematem takich jak serwis JakDojadę nie próbuje w żaden sposób przewidzieć czy oszacować rzeczywistej godziny przyjazdu autobusu czy tramwaju. Czas przyjazdu wyznaczany jest na podstawie rozkładu jazdy, do którego ułożenia wykorzystywane są symulacje i modele ruchu. Coraz popularniejsze stają się systemy tzw. dynamicznej

informacji pasażerskiej, które bazując na danych na temat aktualnego położenia pojazdów z GPSów przewidują czas przyjazdu na przystanek. Istnieje też wiele serwisów, które pokazują aktualną pozycję pojazdów komunikacji miejskiej na mapie. Żadne z nich jednak nie jest tak kompleksową symulacją jak niniejsza. Jej przydatność mogłaby być szczególnie cenna w krajach, w których komunikacja miejska jest źle zorganizowana i nie występuje pojęcie rozkładu jazdy.

Samo opracowanie zagadnienia okazało się dość czasochłonne i dość wymagające. Wymagało to nie tylko poznania problemu od strony teoretycznej, ale także, a może przede wszystkim od praktycznej. Musieliśmy poznać technologie i języki, którymi dotychczas mieliśmy do czynienia w sposób pobieżny. Do nich zaliczają się Python z frameworkiem Tornado i AngularJs. Sporo trudności sprawiła implementacja w formie grafu. Niewątpliwie największym sukcesem projektu jest działająca symulacja ruchu tramwajowego.

W przyszłości planujemy rozszerzyć symulację na inne środki komunikacji miejskiej to znaczy autobusy i pociągi. Należałoby także popracować nad udoskonaleniem algorytmu w taki sposób by położyć większy nacisk na indywidualne czynniki wpływające na dany pojazd, jednak do tego potrzebne są bardziej szczegółowe dane. Warto także dane uwzględnić wypadkach czy awariach pojazdów np.dane o awarii pojazdów z kanału RSS mpk. Kolejną istotną kwestią jest również optymalizacja działania algorytmu.

załącznik 1. Schemat blokowy algorytmu scalania danych

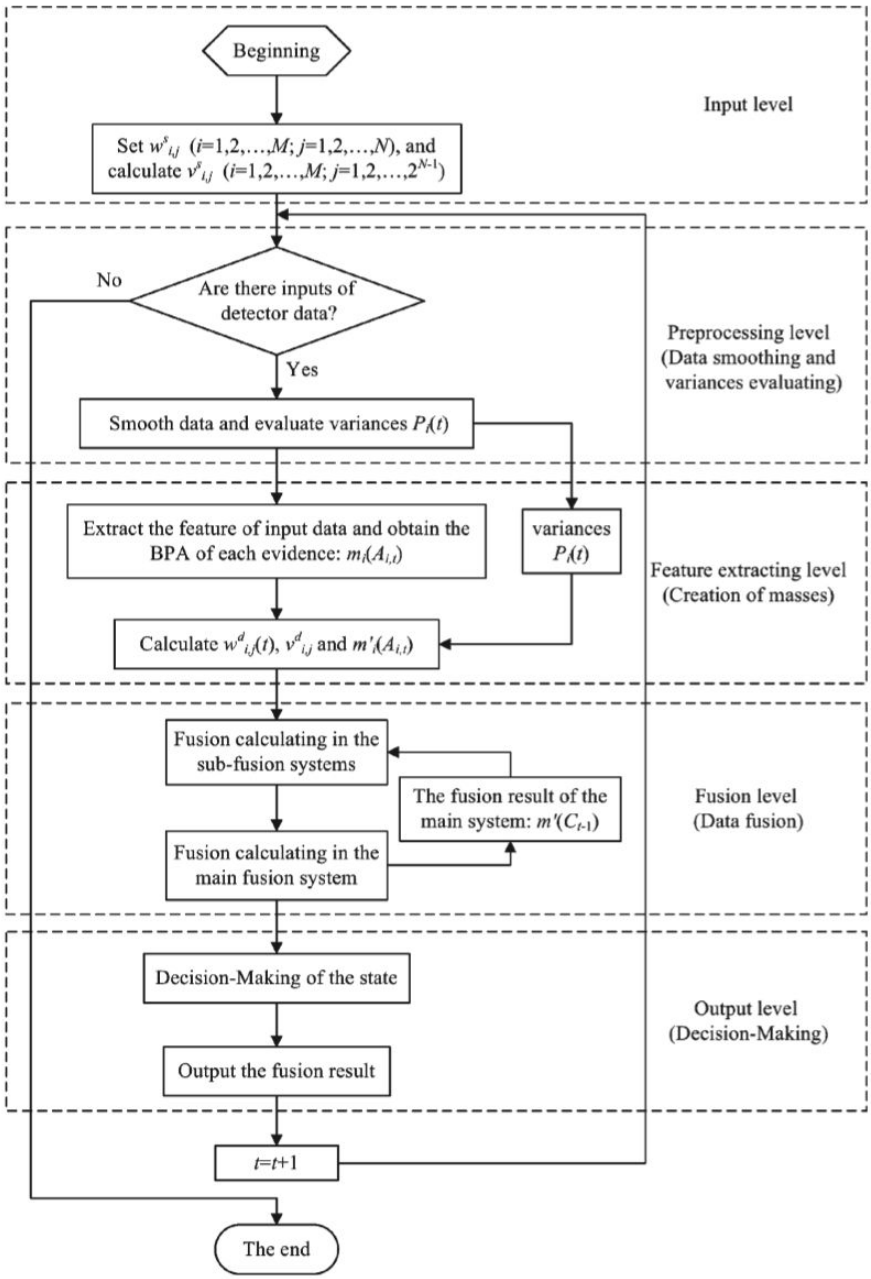


Fig. 1. Flowchart of the proposed evidential fusion algorithm.

załącznik 2.

Tabela 1. **Wyniki badań prędkości samochodów w zależności od ciśnienia atmosferycznego na ulicy dwujezdniowej [km/h]**

Lp.	Pas ruchu	Miara prędkości	Rodzaj ciśnienia atmosferycznego			
			wyżowe	normalne	obniżające się	niżowe
Samochodowy osobowe						
1	Lewy	$\bar{V}$	89,3	86,2	85,5	77,5
		V_{85}	103,0	101,0	100,0	90,0
2	Prawy	$\bar{V}$	80,5	79,4	77,1	73,3
		V_{85}	93,0	92,0	91,0	85,0
Samochodowy dostawcze						
3	Lewy	$\bar{V}$	80,2	76,4	75,4	71,2
		V_{85}	92,0	91,0	89,5	82,5
4	Prawy	$\bar{V}$	70,7	69,0	66,3	62,7
		V_{85}	84,0	81,0	80,0	74,0
Samochodowy ciężarowe						
5	Prawy	$\bar{V}$	62,9	60,8	59,5	58,5
		V_{85}	74,0	73,0	70,0	69,0

Tabela 3. **Wyniki badań prędkości samochodów osobowych w zależności od temperatury powietrza na ulicy dwujezdniowej [km/h]**

Lp.	Pas ruchu	Miara prędkości	Temperatura powietrza [°C]			
			28–32	20–23	11–15	<0
Samochodowy osobowe						
1	Lewy	$\bar{V}$	83,9	86,9	86,0	84,3
		V_{85}	94,0	101,5	100,0	95,5
2	Prawy	$\bar{V}$	76,6	79,8	78,8	77,1
		V_{85}	87,0	92,0	91,0	88,0

Tabela 5. **Wyniki badań prędkości samochodów w zależności od intensywności opadów atmosferycznych na ulicy dwujezdniowej [km/h]**

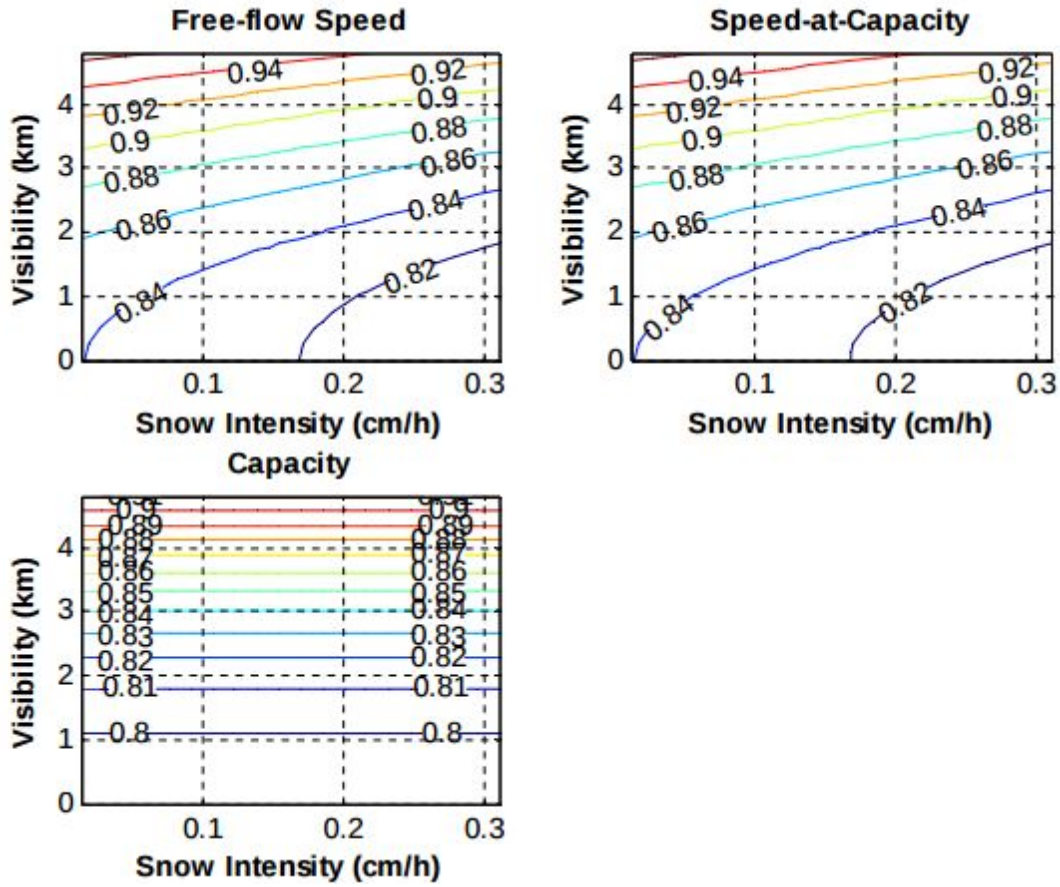
Lp.	Pas ruchu	Miara prędkości	Intensywność opadów [mm/h]				Warunki wzorcowe
			Brak opadów	deszcz 2,5-4,0	deszcz 4,0-6,0	śnieg 3,0-5,0	
Samochodowy osobowe							
1	Lewy	$\bar{V}$	77,5	72,6	71,4	60,4	89,3
		V_{85}	90,0	83,0	82,0	74,5	103,0
2	Prawy	$\bar{V}$	73,3	68,1	65,6	59,6	80,5
		V_{85}	85,0	78,0	77,0	74,0	93,0
Samochodowy dostawcze							
3	Lewy	$\bar{V}$	71,2	66,4	63,0	49,8	80,2
		V_{85}	82,5	76,0	71,5	58,0	92,0
4	Prawy	$\bar{V}$	62,7	61,5	59,1	48,5	70,7
		V_{85}	74,0	71,0	68,8	56,0	84,0
Samochodowy ciężarowe							
5	Prawy	$\bar{V}$	58,5	57,6	55,3	50,0	62,9
		V_{85}	69,0	66,0	65,0	61,4	74,0

Tabela 7. **Wyniki badań prędkości samochodów osobowych w wybranych stanach zamglenia powietrza na ulicy dwujezdniowej [km/h]**

Lp.	Pas ruchu	Miara prędkości	Warunki atmosferyczne	
			słonecznie	pełne zamglenie
Samochodowy osobowe				
1	Lewy	$\bar{V}$	86,2	69,0
		V_{85}	101,0	80,0
2	Prawy	$\bar{V}$	79,4	66,6
		V_{85}	92,0	78,0

załącznik 3. Podsumowanie badań

Traffic Parameter	Weather Condition	Range of Impact
Free-flow speed	Light Rain (<0.01 cm/h)	-2% to -3.6%
	Rain (~1.6 cm/h)	-6% to -9%
	Light snow (<0.01 cm/h)	-5% to -16%
	Snow (~0.3 cm/h)	-5% to -19%
Speed at Capacity	Light Rain (<0.01 cm/h)	-8% to -10%
	Rain (~1.6 cm/h)	-8% to -14%
	Light snow (<0.01 cm/h)	-5% to -16%
	Snow (~0.3 cm/h)	-5% to -19%
Capacity	Light Rain (<0.01 cm/h) and Rain (~1.6 cm/h)	-10% to -11%
	Light snow (<0.01 cm/h)	-12% to -20%



Załącznik 4. Tabela wyboru funkcji oraz przykładowe funkcje

Tablica 1

Warunki przydatności funkcji do opisu oporu trasy

WARUNKI MATEMATYCZNE	WARUNKI BEHAWIORALNE
– funkcja nieliniowa; – funkcja ściśle rosnąca dla przedziału natężenia ruchu w zakresie liczb dodatnich; – funkcja nieujemna; – w przypadku braku natężenia ruchu wartość funkcji odpowiada czasowi przejazdu odcinka przy prędkości swobodnej; – funkcja jest ciągła i różniczkowalna; – pochodna funkcji w przypadku braku natężenia ruchu jest dodatnia.	– czas spędzony w sieci silnie obciążonej ruchem - na poziomie swobody F - jest dla użytkownika znacznie bardziej uciążliwy niż czas jazdy; – po przekroczeniu przepustowości natężenie krytyczne na odcinku spada, a czas przejazdu rośnie; – funkcja powinna spełniać warunek „ostrego blokowania odcinka" w przypadku osiągnięcia granicy przepustowości (dodatkowa „kara" dla potoków powyżej przepustowości); – przyjęcie pojęcia „prędkości swobodnej" jako prędkości w warunkach włączonej sygnalizacji świetlnej; – naturalnym pierwszym wyborem ścieżki podróży jest prowadzenie jej po ulicach szerokich i wygodnych z punktu widzenia użytkownika (wyższych klas z punktu widzenia klasyfikacji funkcjonalnej); – funkcja powinna umożliwiać występowanie chwilowych przeciążeń w sieci (potoki o wartościach wyższych niż przepustowość).

Źródło: [7], [13].

– funkcję wykładniczą Overgaarda:

$$t = t_0 \cdot \alpha^{\left(\frac{q}{q_C}\right)^{\beta}}, \qquad (1)$$

– funkcję logarytmiczną Moshera:

$$t = t_0 + \beta \cdot \ln(\alpha) - \beta \cdot \ln(\alpha - q) \quad dla \quad q \le q_{max}, \qquad (2)$$

$$t = t_C + s \cdot (q - q_{max}) \quad dla \quad q > q_{max}, \qquad (3)$$

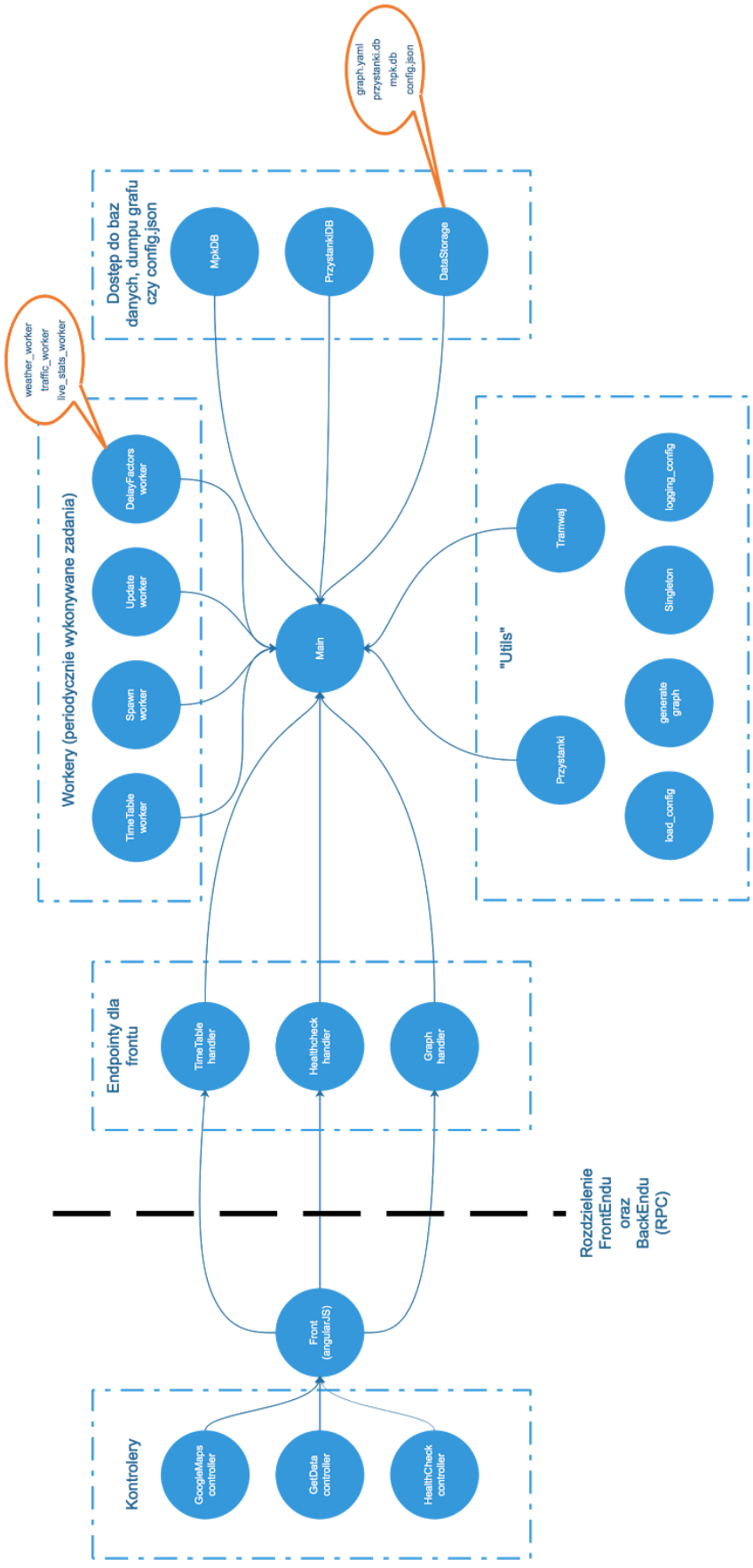
gdzie:

– $t_C = t_0 + \beta \cdot \ln(\alpha) - \beta \cdot \ln(\alpha - q_{max})$,

– $s = \frac{\beta}{\alpha - q_{max}}$,

– $\alpha > q_{max}$,

Załącznik 5. Schemat modułów



Załącznik 6. Link do dokumentacji API
https://github.com/evemorgen/GdzieJestTenCholernyTramwajProject/blob/master/backend/api_docs.md

Załącznik 8. Link do repozytorium
https://github.com/evemorgen/GdzieJestTenCholernyTramwajProject

Załącznik 7. graph.png