Κωνσταντίνος Τσιάρας ΑΜ: 1115201500165

Αθηνά Χατζοπούλου ΑΜ: 1115201500175

**Ανάπτυξη Λογισμικού για Δίκτυα και Τηλεπικοινωνίες - Project**

**MySQL Database:**

Περιλαμβάνει 1 μοναδικό table (*datapoints*) το οποίο περιέχει όλες τις ζητούμενες από την εκφώνηση στήλες. Κάθε επικοινωνία (select, insert, κλπ) του Edge Server με τη βάση γίνεται μέσω του DBBridge.java . Οδηγίες για σύνδεση σε αυτή περιλαμβάνονται στο *./mysql/DB\_Readme.txt* .

**Edge Server (ES):**

Το κομμάτι του Java project επιτελεί 4 διαφορετικές λειτουργίες, οι οποίες ενεργοποιούνται μέσω κατάλληλων flags από τη γραμμή εντολών:

* XML to CSV conversion ( *-c* )

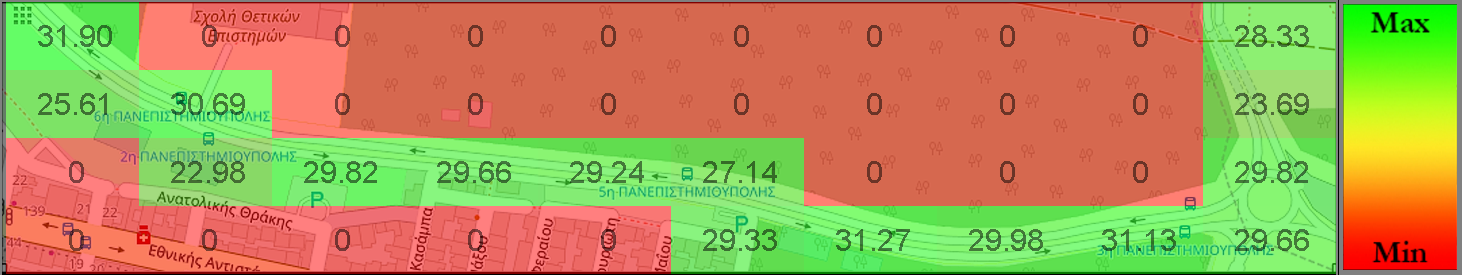
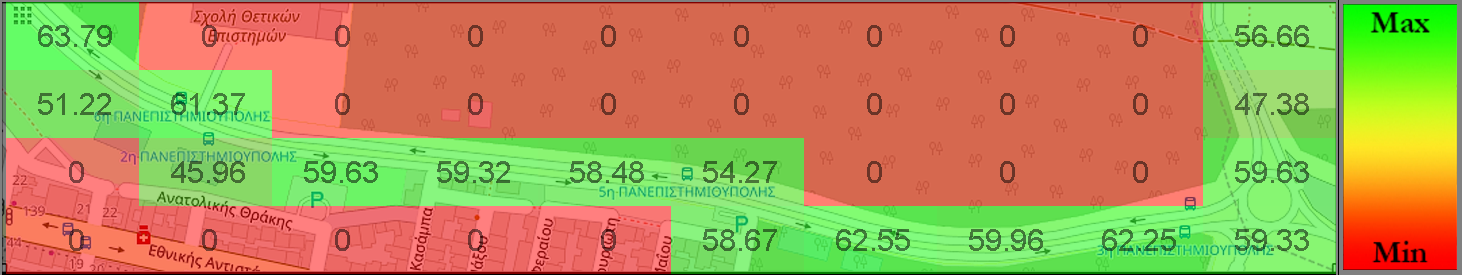
Η λειτουργία αυτή, όπως περιγράφεται στην εκφώνηση, διαβάζει από τα SUMO xml αρχεία για κάθε datapoint τα “timestep, id, lat, long, angle, speed”, δημιουργεί “RSSI, throughput” με τον τρόπο που περιγράφεται στην εκφώνηση και τα γράφει σε καινούργια αρχεία csv, αντίστοιχα για κάθε xml.

* Heatmap generation ( *-g* )

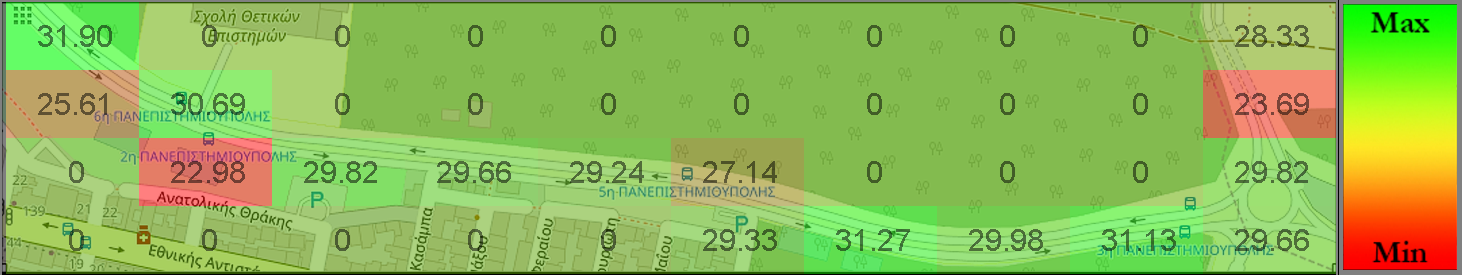
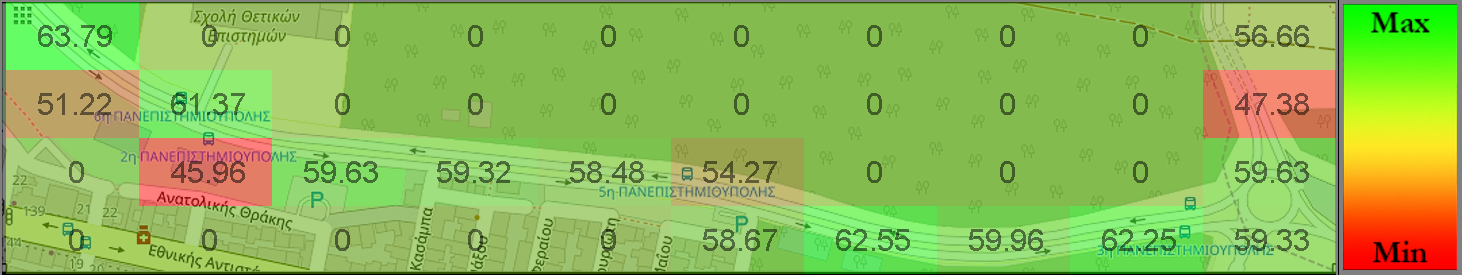
Εδώ αρχικά διαβάζονται τα αρχεία csv που δημιουργήθηκαν παραπάνω προκειμένου να δημιουργηθούν 2 δισδιάστατοι πίνακες (*rssiCellMap* και *throughputCellMap*) με διαστάσεις όσες και του ζητούμενου grid επί του δοθέντος χάρτη *Map.jpg* (default 4x10). Το κάθε κελί αυτών περιέχει το μέσο όρο των τιμών – RSSI και throughput αντίστοιχα – που βρίσκονται στις συντεταγμένες που αναλογούν στο εκάστοτε κελί.

Στη συνέχεια, για καθένα από τους 2 αυτούς πίνακες κατασκευάζεται από 1 heatmap με χρήση της βιβλιοθήκης [JHeatChart](http://www.javaheatmap.com/). Το κάθε heatmap τοποθετείται πάνω από την εικόνα του χάρτη (overlay), με τις τιμές του κάθε κελιού να αναγράφονται στο εσωτερικό τους.

Ένα πρόβλημα που συναντήσαμε στο κομμάτι των heatmaps ήταν το ότι για πολλά κελία, συγκεκριμένα όσα δεν αντιστοιχούσαν σε δρόμους, όπως ήταν λογικό δεν υπήρχαν διαθέσιμες τιμές στα δεδομένα μας και συνεπώς ο μέσος όρος των RSSI/throughput για αυτά ήταν 0. Αυτό όμως είχε σαν αποτέλεσμα η JHeatChart να χρωματίζει αυτά κόκκινα και όλα τα υπόλοιπα – οι διακυμάνσεις μεταξύ των οποίων είναι προφανώς μικρές – να τα χρωματίζει όλα πράσινα. Προκειμένου να το αντιμετωπίσουμε αυτό σκεφτήκαμε να υπολογίσουμε το μέσο όρο όλων των τιμών του κάθε πίνακα και να περάσουμε αυτή την τιμή αντί για 0 στη JHeatChart στα κελιά των οποίων η τιμή είναι 0 με σκοπό αυτά να χρωματιστούν με το πιο «ουδέτερο» χρώμα, μεταξύ κόκκινου και πράσινου. Η λειτουργικότητα αυτή ενεργοποιείται μέσω του πεδίου *Heatmap.normalizeFlag* και τα διαφορετικά αποτελέσματα για ενδεικτικά csv φαίνονται παρακάτω:



Heatmaps RSSI (πάνω) και throughput (κάτω) με *normalizeFlag=false*



Heatmaps RSSI (πάνω) και throughput (κάτω) με *normalizeFlag=true*

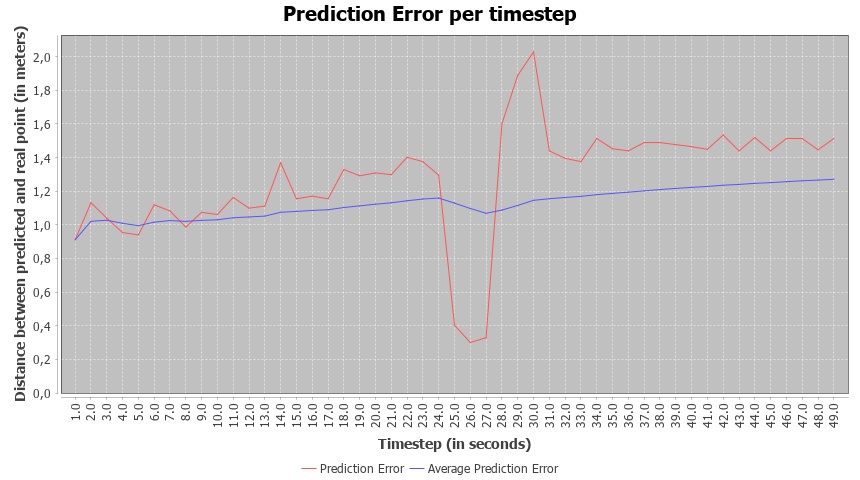
* Vehicle MQTT topics subscription ( *-s* )

Εδώ απλά γίνεται subscribe στα MQTT topics στα οποία τα android terminals στέλνουν τα datapoints τους. Κάθε datapoint που φτάνει στον Edge Server εκτυπώνεται στο stdout.

Για καθένα από αυτά ο ES κάνει μια πρόβλεψη για το επόμενο timestep χρησιμοποιώντας τα “lat, long, angle, speed” σύμφωνα με τους τύπους που δίνονται στην εκφώνηση και την στέλνει στο terminal από το οποίο προήλθε το εκάστοτε datapoint. Τόσο τα πραγματικά όσο και τα προβλεπόμενα δεδομένα αποθηκεύονται σε αυτό το σημείο στη βάση. Πατώντας “Enter” ο ES αποσυνδέεται από τα topics αυτά και τερματίζει.

* Prediction errors’ calculation and visualization ( *-e* )

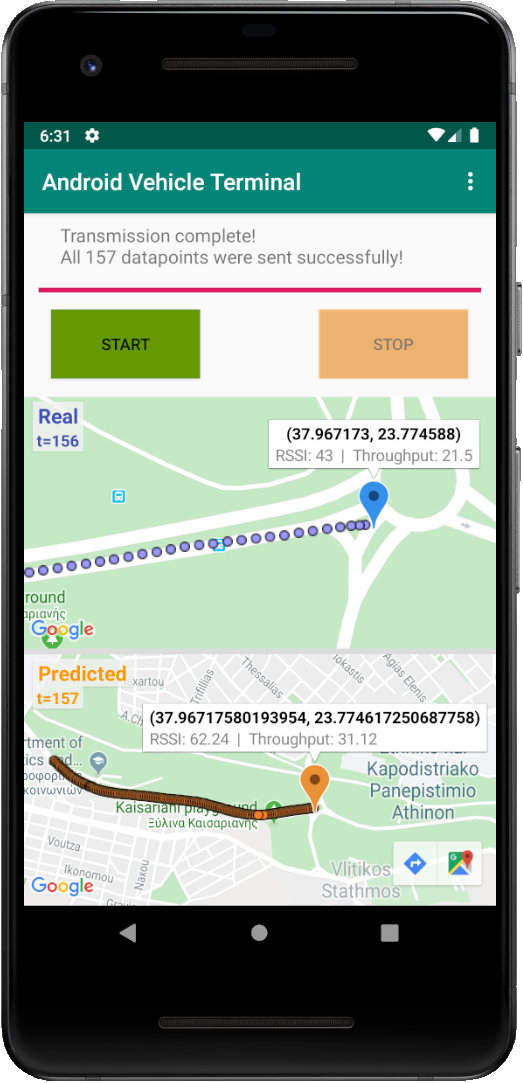
Ανακτώντας όλα τα δεδομένα (real & predicted) από τη βάση, γίνεται υπολογισμός της απόστασης μεταξύ του πραγματικού και του προβλεπόμενου σημείου για κάθε διαθέσιμο timestep ανά terminal. Για το καθένα δημιουργείται ένα line graph με τις αποκλίσεις αυτές και το μέσο όρο τους κάθε στιγμή, ενώ ο συνολικός μέσος όρος σφάλματος τους εκτυπώνεται και στην οθόνη. Τα πλήρη γραφήματα που προκύπτουν από το σύνολο των διαθέσιμων datapoints – 157 για το vehicle\_26 και 175 για το vehicle\_27 – περιλαμβάνονται στον παραδοτέο φάκελο *./img\_out/ .* Εδώ παρατίθεται ενδεικτικά ένα γράφημα για τα πρώτα 50 του vehicle\_26:



**Android Terminals:**

Η ταυτότητα του terminal, αν δηλαδή διαβάζει το *vehicle\_26.csv* ή το *vehicle\_27.csv* και, αντίστοιχα, σε ποιο MQTT topic θα επικοινωνεί με τον ES ρυθμίζεται μέσω της μεταβλητής *terminalId*.

Τα terminals κάνουν περιοδικούς ελέγχους κάθε *onlineCheckFrequency* (default 5’’) για σύνδεση στο internet. Σε περίπτωση που αυτή δεν υπάρχει παραπέμπουν μέσω Alert στην ενεργοποίησή του.

* Main Activity

Το κεντρικό activity αποτελείται από κουμπιά “Start” και “Stop” για εκκίνηση και τερματισμό της αποστολής των datapoints (γραμμές του csv) του android terminal στον ES, μια μπάρα προόδου καθώς και 2 χάρτες (real και predicted) με χρήση του Google Maps API.

Η αποστολή γίνεται με σταθερό ρυθμό ένα datapoint ανά δευτερόλεπτο και εάν σταματήσει (μέσω του κουμπιού “Stop”) η επανέναρξή της (με το κουμπί “Start”) γίνεται από την αρχή. Το πλήθος των datapoints που αποστέλλεται εξαρτάται από την τιμή Runtime των ρυθμίσεων (default το σύνολο των γραμμών του csv).

Τα πραγματικά datapoints σημειώνονται σε πραγματικό χρόνο στον πάνω (real) χάρτη σε πραγματικό χρόνο, συνοδευόμενο από ένα info window όπου περιέχονται οι συντεταγμένες, το RSSI και το throughput του καθενός.

Παράλληλα, για κάθε datapoint που στέλνεται στον Edge Server το terminal λαμβάνει μια απάντηση-πρόβλεψη για τη θέση του στο επόμενο timestep η οποία σημειώνεται στον κάτω (predicted) χάρτη, με τρόπο αντίστοιχο του real – info window με τις προβλεπόμενες συντεταγμένες και τα προβλεπόμενα RSSI και throughput.

* Settings Activity

Εδώ υπάρχουν επιλογές για είσοδο IP Address, Port και Runtime, όπως και κουμπί “Exit” για έξοδο από την εφαρμογή κατόπιν σχετικού Alert για επιβεβαίωση. Για όλες τις εισόδους πραγματοποιούνται έλεγχοι εγκυρότητας. Σημειώνεται ότι οι ρυθμίσεις αυτές αποθηκεύονται ακόμα και μετά από κλείσιμο τη εφαρμογής.