ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών

Ανάπτυξη Λογισμικού για Δίκτυα & Τηλεπικοινωνίες

Δεύτερο Παραδοτέο

Ονοματεπώνυμο	**ΑΜ**
Σεραφείμ Λέντας	1115201700070
Αθανασία Νούλα	1115201700108
Κυριάκος Χαραλαμπους	1115201700204

Edge Server

xmltocsv

Η συναρτηση **void XMLtoCSV(String)** (XMLtoCSV.java) παίρνει ως όρισμα το όνομα ενός αρχείου XML που βρίσκεται στο φάκελο InputData και δημιουργεί ένα αρχείο CSV με το ίδιο όνομα. Αρχικά δημιουργεί μία λίστα με τα timestep του XML και ορίζει τη διασπορα και τη μέση τιμή για την κανονική κατανομή στο [20,100]. Έπειτα για κάθε κόμβο timestep δημιουργεί μια ακόμα λίστα με τα δείγματα ανα δευτερόλεπτο, για κάθε ένα από αυτά υπολογίζει το RSSI, με τη βοήθεια της nextGaussian() και του τύπου της κανονικής κατανομής ελέγχοντας, ελέγχοντας ότι οι τιμές οτι ειναι στο [20,100]. Στη συνέχεια υπολογίζουμε και το Throughput και γράφουμε το διάνυσμα στο CSV που δημιουργήσαμε.

rssi_throughput_values

1. ### Αρχικοποίηση πινάκων (num_of_samples.java)
   Για την αρχικοποίηση των πινάκων με 0, οι οποίοι θα χρησιμοποιηθούν για τον υπολογισμό των τιμών rssi & throughput. Στην ουσία αρχικοποιείται ένας πίνακας με διαστάσεις [4][10], όπου κάθε θέση του πίνακα αντιπροσωπεύει ένα “κουτάκι” του πίνακα στο grid.

2. ### Υπολογισμός RSSI(average_rssi.java)
   Η συνάρτηση **double[][] average_rssi** η οποία επιστρέφει ένα δισδιάστατο πίνακα με διαστάσεις (4,10) που περιέχει το μέσο όρο των τιμών rssi για το grid διαστάσεων (4,10). Ο υπολογισμός του μέσου όρου γίνεται με τη αφαίρεση των τιμών max και min, που δίνονται από την εκφώνηση και η διαίρεση τους με το 4 και το 10, για το lat και το long αντίστοιχα, και έτσι προκύπτουν οι τιμές που θα πρέπει να ληφθούν υπόψη για κάθε “κελί”. Έπειτα, σαρώνουμε όλα τα δεδομένα γραμμή-γραμμή από το αρχείο CSV και ανάλογα με τις τιμές lat και long που έχει βρίσκουμε σε ποιο “κελί” βρίσκεται, ώστε να τα προσθέτουμε ανάλογα και στο τέλος του αρχείου να τα διαιρούμε με το πλήθος των στοιχείων που προστέθηκαν για να προκύψει ο μέσος όρος, που μετατρέπεται σε ποσοστιαία μονάδα στη κλίμακα του 100. Σε περίπτωση που το στοιχείο δεν ανήκει στο διάστημα min-max για κάποια από τις δύο διαστάσεις, αγνοείται.

3. ### Υπολογισμός THROUGHPUT(average_throughput.java)
   Η συνάρτηση **double[][] average_throughput** η οποία έχει την ίδια λειτουργία με την avarege_rssi, αλλά αυτή επιστρέφει με τη διαδικασία που έχει περιγραφεί παραπάνω, έναν πίνακα (4,10) με το μέσο όρο των τιμών throughput για κάθε “κελί”.

Heatmaps

1. ### Δημιουργία πίνακα με χρώματα(create_gradient.java)
   Η συνάρτηση **Color[] createGradient** που κατά τη κλήση της δίνονται οι παράμετροι: *Color one, Color two, int numSteps*. Η συνάρτηση εξάγει ένα πίνακα χρωμάτων

διαστάσεων *numsteps* με το χρώμα *one* να χαρακτηρίζει τη πιο χαμηλή τιμή και το *two* την πιό υψηλή. Σε αυτή τη εργασία, χρησιμοποιείται το κόκκινο ως *one* και το πράσινο ως *two*. Η δημιουργία των χρωμάτων γίνεται υπολογίζοντας νέες τιμές rgb και για κάθε θέση του πίνακα μεταβάλλοντας τις προκαθορισμένες τιμές των rgb ποσοστιαία, που το ποσοστό βασίζεται στον αριθμό *numsteps*.

2. Δημιουργία grid με κατάλληλα χρώματα(draw_and_fill_grid.java)

   Η συνάρτηση **Draw_and_Fill_Grid** η οποία πέρνει ως παραμέτρους μια μεταβλητή g τύπου Graphics, το ύψος της εικόνα, το πλάτος και ένας πίνακας (4,10) που προκύπτει είτε από τη **average_throughput** είτε από τη **average_rssi.** Κατά τη κλήση της πάνω στη μεταβλητή g, που στη περιπτωσή μας χρησιμοποιεί τη Buffered Image, δημιουργεί το grid που υπολογίζεται με τις διαίρεση των 2 διαστάσεων της εικόνας με το 4 και το 10 ανάλογα, και γεμίζει το κάθε “κελί” με το κατάλληλο χρώμα. Για να γίνει αυτό, καλείται η συνάρτηση **createGradient** και χρησιμοποιείται ο πίνακας που δίνεται ως παράμετρος, δηλαδή των μέσων όρων των rssi ή throughput, και σε συνδυασμό με τα 10 χρώματα που επιστρέφει με τη παράμετρο numstep=10 μπορούμε και γεμίζουμε κατάλληλα τα κελιά του grid, με τη προσαρμογή των χρωμάτων με transparency value=200, για να φαίνεται και ο χάρτης.

3. Ορισμός της κλάσης rssi_heatmap για τη δημιουργία του(rssi_heatmap.java)

   Η κλάση **RSSI_HeatMap** extends Jpanel που είναι βασισμένη στη έτοιμη κλάση Jpanel. Ο constructor της κλάσης στην ουσία διαβάζει και εισάγει τη BufferedImage που είναι η εικόνα που θα είναι η βάση του heatmap, δηλαδή ο χάρτης και γεμίζει το πίνακα με τις μέσες τιμές για το RSSI σύμφωνα με τη παράμετρο που δόθηκε από τη main . Σε αυτή τη κλάση γίνεται override των δύο συναρτήσεων **paintComponent** και **getPreferredSize**. Η πρώτη δίνει το πίνακα με τις μέσες τιμές του RSSI μαζί με την εικόνα και τις διαστάσεις της, στη συνάρτηση **Draw_and_Fill_Grid** που σχηματίζει το RSSI Heatmap, όπως έχει αναλυθεί παραπάνω. Η δεύτερη συνάρτηση που έχει γίνει @Override, η **getPreferredSize**, απλά επιστρέφει τις διαστάσεις της εικόνας που δίνουμε διαφορετικά αν δεν δοθεί σωστά η εικόνα, γυρίζει τις τυχαίες διαστάσεις (300,300) για να δημιουργηθεί ένα window 300x300 και βγάζει αντίστοιχο μήνυμα λάθους.

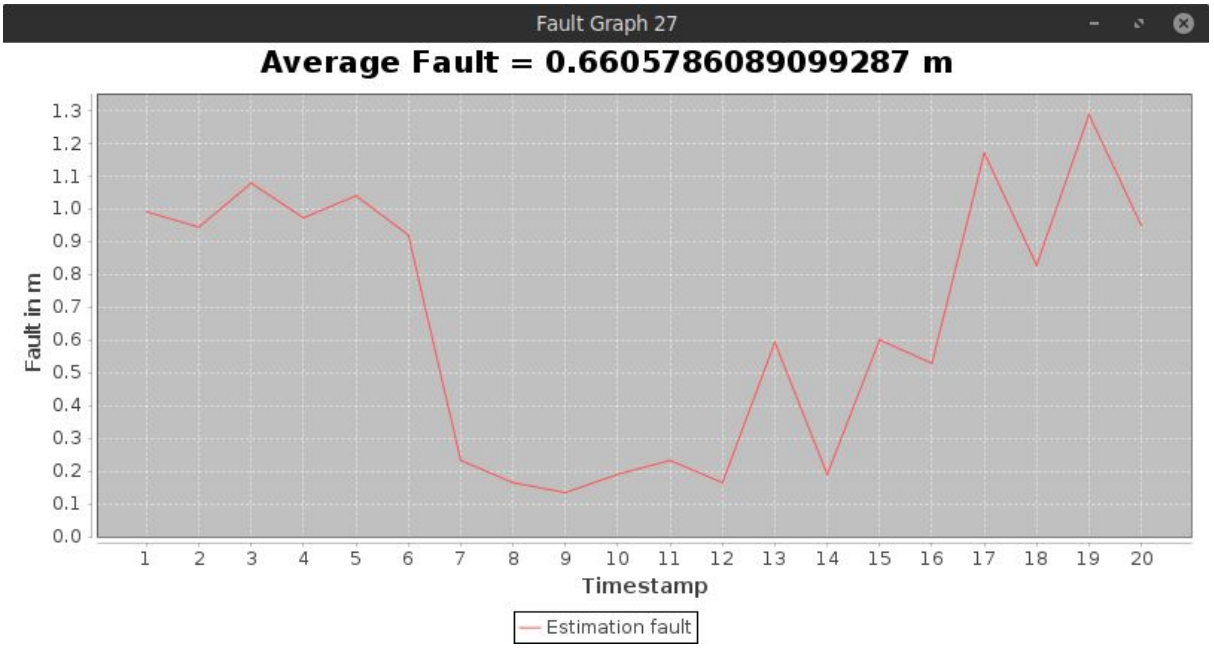
4. Ορισμός της κλάσης throughput_heatmap για τη δημιουργία του(throughput_heatmap.java)

   Η κλάση **THROUGHPUT_HeatMap extends Jpanel** όπου έχει την ίδια λειτουργία με την RSSI_HeatMap extends Jpanel, αλλά αυτή επικεντρώνεται στις τιμές του throughput. Πιο συγκεκριμένα, σε αυτή χρησιμοποιεί τις τιμές του **average_throughput** όπως έχουν δοθεί από τη main που επιστρέφει τις τιμές μέσων όρων των throughput, για τη διαχείρισή τους.

Fault_graph

1. ### Υπολογισμός απόκλισης της απόστασης(distance.java)
   Στη συνάρτηση distance γίνεται υπολογισμός της απόκλισης της απόστασης μεταξύ της προβλεπόμενης γεωγραφικής θέσης με την πραγματική. Η συνάρτηση δέχεται τις 4 τιμές[2 προβλεπόμενες(lat & long) και δύο πραγματικές] και υπολογίζεται η απόκλιση σύμφωνα με το παράρτημα της σελίδας https://www.geodatasource.com/developers/java σε μέτρα, ενώ άν είναι ίδιες οι τιμές ορίζεται ως 0.

2. ### Οπτικοποίηση μέσου όρου αποκλίσεων(fault_graph.java)
   Για τη οπτικοποίηση την σφαλμάτων η συνάρτηση, δέχεται το όνομα της απεικόνισης, τη λίστα με τις αποκλίσεις καθώς και το τελευταίο timestamp που καταγράφηκε στο τελευταία απόκλιση. Σύμφωνα με τη λίστα υπολογίζεται ο μέσος όρος των αποκλίσεων και γίνεται η απεικόνισή του ανα timestamp σε JFreeChart σε νέο Panel.



Custom_types

Ορισμός κλάσης για τις προβλεπόμενες τιμές(estimations.java). Γίνεται ο ορισμός για τις προβλεπόμενες τιμές, που περιλαμβάνουν lat, long,rssi και throughput, ο constructor που συμφωνα με τις δοθείσες παραμέτρους καταχωρεί τις τιμές, και γίνεται ο ορισμός των σχετικών συναρτήσεων για την άντληση των τιμών.

Main συνάρτηση

- ## Heatmaps

  Αρχικά δημιουργούνται τα csv αρχεία και υπολογίζονται οι μέσες τιμές των rssi & throughput για τη κλίμακα του grid[4][10] και για τις απεικονίσεις των heatmaps καλούμε τις κλάσεις Jframe για τη δημιουργία του frame και GridBagLayout στο Jpanel για το grid στο frame και το αποτέλεσμα του δίνεται για επεξεργασία στις συναρτήσεις που αναφέρθηκαν παραπάνω, στη κλάση RSSI_HeatMap και THOUGHPUT_HeatMap αντίστοιχα. Έπειτα πραγματοποιείται η σύνδεση με τη κενή βάση.

- ## Λήψη και αποστολή δεδομένων

  Αρχικά δημιουργούμε ένα serverID, ορίζουμε το url του broker, δημιουργούμε έναν Mqtt Client και συνδεόμαστε στο broker. Στη συνέχεια κάνουμε subscribe σε όλα τα subtopics του **vehicles/#**. Όταν λάβει κάποιο μήνυμα το κάνουμε split ώστε να πάρουμε το id του αποστολέα, ελέγχουμε αν έχει σταλεί από τον ιδιο το Server και τότε εμφανίζεται σχετικό μήνυμα, αλλιώς ελέγχει αν το μήνυμα που έχει ληφθεί είναι το μήνυμα “done” για να γίνει ο τερματισμός της διαδικασιας.Σε περίπτωση που δεν είναι το μήνυμα τερματισμού ("client id"/done/"id),εκτυπώνεται το μήνυμα που έχει ληφθεί από το ελέγχει αν οι τιμές που παίρνει είναι μέσα στη περιοχή που δίνεται (Εικόνα 1) και αν είναι τότε γίνεται και αναζήτηση στο hashmap που έχει δημιουργηθεί για τις προβλεπόμενες τιμές. Αν πρόκειται για μήνυμα ανανέωσης συνεδρίας ("client id"/fresh/"id) διαγράφουμε από τη βάση όλες τις εγγραφές για το όχημα αυτό.

  Το “κλειδί” για την αναζήτηση στο hashmap ορίζεται η συμβολοσειρά που θα περιέχει το id του αποστολέα και το ανάλογο timestamp, ώστε να είναι μοναδικό. Αν το δεδομένο με το συγκεκριμένο “κλειδί” υπάρχει στο hashmap τότε εισάγονται τα δεδομένα της καταγραφής στη βάση (Εικόνα 2) και έπειτα διαγράφονται από το hashmap. Μετά, βασιζόμενοι στα δεδομένα του αποστολέα, γίνεται ο υπολογισμός της προβλεπόμενης θέσης και η εύρεση των προβλεπόμενων RSSI και Throughput απο τον πίνακα των μέσων τιμών που δημιουργήσαμε. Σε επόμενο βήμα, συλλέγονται τα παραπάνω δεδομένα πρόβλεψης σε ένα αντικείμενο-δεδομένο estimations και τοποθετείται στο hashmap, καθώς και αποστέλλονται στο ίδιο topic με τη χρήση νέου thread(Εικόνα 3). Αν το μήνυμα είναι μήνυμα τερματισμού("client

id"/done/"id), τότε με βάση το id του οχήματος συλλέγονται τα δεδομένα του από τη βάση και υπολογίζονται οι αποκλίσεις στην απόσταση μεταξύ της προβλεπόμενης γεωγραφικής θέσης με την πραγματική με τη βοήθεια της distance συνάρτηση και να συλλέγονται σε ανάλογη λίστα.

Τέλος γίνεται και η απεικόνιση της απόκλισης(fault_graph).Για τον τερματισμό του Edge Server κάνουμε handle το σήμα ctrl+C, ώστε να εξασφαλίσουμε ότι θα γίνει ελεγχόμενη αποσύνδεση από τον MQTT Broker και τη βάση δεδομένων.

Εικόνες

‘Ελεγχος των στοιχείων για το αν υπάρχουν στη περιοχή που ελέγχεται

```
From: v26
New session for v26

From: v26
REAL DATA 0.0 0.00,26,37.971738,23.758638,38.291786,0.000000,39.41683942095901,19.708419710479504
Vehicle is outside of the selected area

From: v26
REAL DATA 1.0 1.00,26,37.971745,23.758645,38.291786,1.000000,39.2462769947987,19.62313849739935
Vehicle is outside of the selected area

From: v26
REAL DATA 2.0 2.00,26,37.971759,23.758660,38.291786,2.000000,29.45851910720151,14.729259553600754
Vehicle is outside of the selected area

From: v26
REAL DATA 3.0 3.00,26,37.971780,23.758682,38.291786,3.000000,51.69408555686546,25.847042778432726
Vehicle is outside of the selected area

From: v26
Time to calculate fault for 26
Average Fault = NaN m
_
```

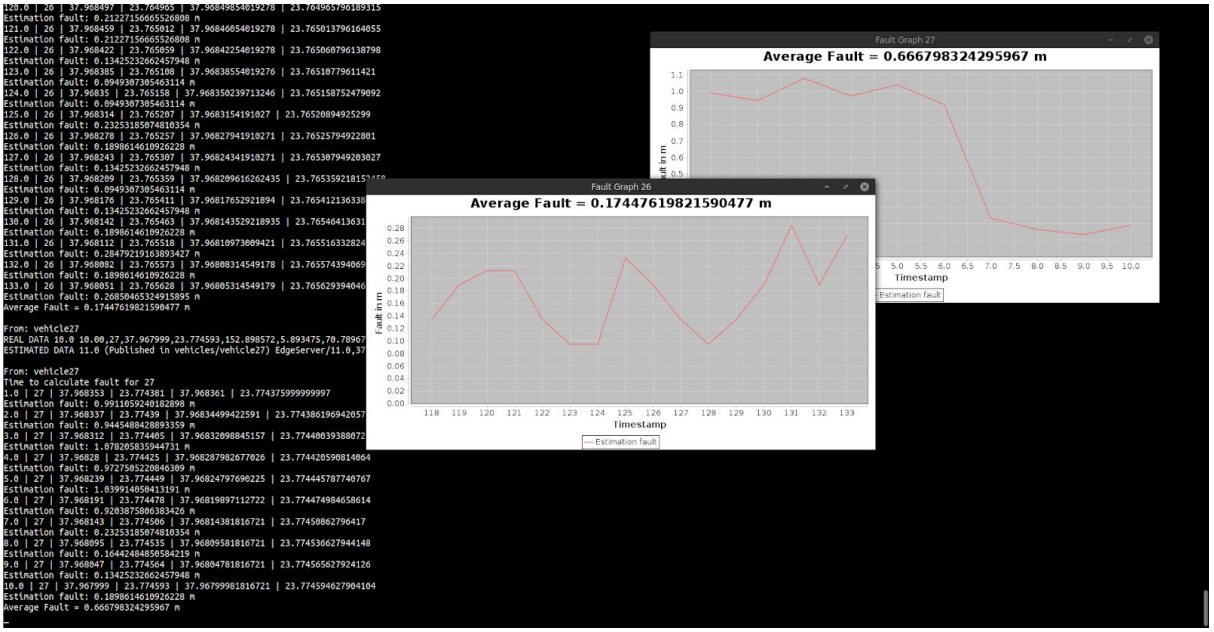
Στιγμιότυπο βάσης

```
From: v26
Time to calculate fault for 27
1.0 | 27 | 37.968353 | 23.774381 | 37.968361 | 23.774375999999997
Estimation fault: 0.9911059240182898 m
2.0 | 27 | 37.968337 | 23.77439 | 37.96834499422591 | 23.774386196942057
Estimation fault: 0.9445488428893359 m
3.0 | 27 | 37.968312 | 23.774405 | 37.96832098845157 | 23.77440039388072
Estimation fault: 1.078205835944731 m
4.0 | 27 | 37.96828 | 23.774425 | 37.968287982677026 | 23.774420590814064
Estimation fault: 0.9727505220846309 m
5.0 | 27 | 37.968239 | 23.774449 | 37.96824797690225 | 23.774445787740767
Estimation fault: 1.039914050413191 m
6.0 | 27 | 37.968191 | 23.774478 | 37.96819897112722 | 23.774474984658614
Estimation fault: 0.9203875806383426 m
7.0 | 27 | 37.968143 | 23.774506 | 37.96814381816721 | 23.77450862796417
Estimation fault: 0.23253185074810354 m
Average Fault = 0.8827778009623749 m
_
```

Απεικόνιση των μηνυμάτων που στέλνει ο Edge Server στο Android τερματικό

```
From: v26
REAL DATA 0.0 0.00,27,37.968361,23.774376,152.898572,0.000000,43.894469025078166,21.947234512539083
ESTIMATED DATA 1.0 (Published in vehicles/v26) EdgeServer/1.0,37.968361,23.774375999999997,67.19282654757245,33.596413273786226

From: v26
REAL DATA 1.0 1.00,27,37.968353,23.774381,152.898572,1.000000,45.98357540402146,22.99178770201073
ESTIMATED DATA 2.0 (Published in vehicles/v26) EdgeServer/2.0,37.96834499422591,23.774386196942057,67.19282654757245,33.596413273786226

From: v26
REAL DATA 2.0 2.00,27,37.968337,23.774390,152.898572,2.000000,85.59311025376643,42.79655512688321
ESTIMATED DATA 3.0 (Published in vehicles/v26) EdgeServer/3.0,37.96832098845157,23.77440039388072,67.19282654757245,33.596413273786226

From: v26
REAL DATA 3.0 3.00,27,37.968312,23.774405,152.898572,3.000000,77.53426303365677,38.767131516828385
ESTIMATED DATA 4.0 (Published in vehicles/v26) EdgeServer/4.0,37.968287982677026,23.774420590814064,67.19282654757245,33.596413273786226

From: v26
REAL DATA 4.0 4.00,27,37.968280,23.774425,152.898572,4.000000,88.59443990493256,44.29721995246628
ESTIMATED DATA 5.0 (Published in vehicles/v26) EdgeServer/5.0,37.96824797690225,23.774445787740767,67.19282654757245,33.596413273786226

From: v26
REAL DATA 5.0 5.00,27,37.968239,23.774449,152.898572,5.000000,58.12349491220296,29.06174745610148
ESTIMATED DATA 6.0 (Published in vehicles/v26) EdgeServer/6.0,37.96819897112722,23.774474984658614,67.19282654757245,33.596413273786226
```

Απεικόνιση μέσης απόκλισης για δύο τερματικά



Εφαρμογή Android

MainActivity

onCreate()

Αρχικά με το που ανοίγουμε την εφαρμογή, μας εμφανίζεται η διεπαφή χρήστη του MainActivity. Αποτελείται από 4 EditText, 1 TextView, και 3 Buttons. Σε περίπτωση που δεν έχουμε αποδεχτεί τις απαιτήσεις της εφαρμογής μας εμφανίζεται και ένας διάλογος για να τις αποδεχτούμε.

Στο πρώτο EditText ο χρήστης καλείται να ορίσει μία διεύθυνση ip, στο δεύτερο να ορίσει port, στο τρίτο να ορίσει χρόνο προσομοίωσης(προεραιτικά) και στο τέταρτο να ορίσει id. Το πρώτο Button ανοίγει ένα file explorer ώστε να επισυνάψουμε το μονοπάτι για csv αρχείο μας. Σε περίπτωση που δεν έχουμε αποδεχτεί τις απαιτήσεις της εφαρμογής μας εμφανίζεται διάλογος για να τις αποδεχτούμε και μετά μας δίνεται η δυνατότητα να επισυνάψουμε το μονοπάτι για csv αρχείο. Όταν το επισυνάψουμε τότε το μονοπάτι εμφανίζεται στο TextView μας. Το δεύτερο Button ελέγχει αν έχουν συμπληρωθεί όλα τα EditText και το TextView κατάλληλα και ξεκινάει το RealActivity. Αν υπάρχει κάποιο λάθος επιστρέφει το κατάλληλο μήνυμα. Το τρίτο Button είναι για την έξοδο μας από την εφαρμογή, αλλά πρώτα μας ανοίγει ένα διάλογο για να επιβεβαιώσουμε την επιλογή μας.

onResume()

Γίνεται έλεγχος για σύνδεση σε δίκτυο, αλλά και για πρόσβαση στο διαδίκτυο ανά 5 δευτερόλεπτα.

onPause()

Σταματάει τους ελέγχους όταν η δραστηριότητα δεν είναι στο προσκήνιο.

openNewActivity()

Συνάρτηση για να το εκκίνηση του RealActivity στέλνωντας παραμέτρους που ορίστηκαν από το χρήστη.

onBackPressed()

Όταν ο χρήστης πατήσει το κουμπί “πίσω” για να βγει από την εφαρμογή, εμφανίζεται ένας διάλογος για να επιβεβαιώσει την επιλογή του.

isNetworkConnected()

Συνάρτηση για έλεγχο αν το android τερματικό είναι συνδεδεμένο σε κάποιο δίκτυο.

isConnected()

Συνάρτηση για έλεγχο αν το android τερματικό έχει πρόσβαση στο διαδίκτυο.

showListItemDialog()

Εμφανίζει ένα διάλογο με μια λίστα από αρχεία του android τερματικού για να βρούμε το αρχείο .csv

onResult()

Όταν ο χρήστης βρει αρχείο ο διάλογος κλείνει και συμπληρώνει το path στο textView.

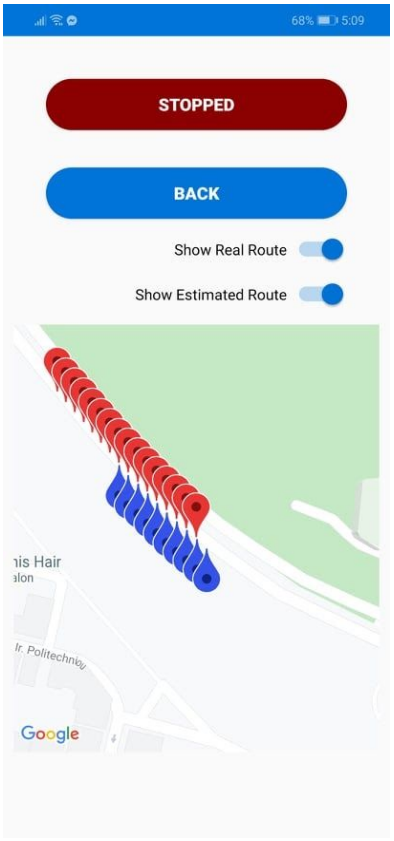
check()

Ελέγχει αν ο χρήστης έχει δώσει άδεια πρόσβασης για την δοθέν άδεια.

RealActivity

onCreate()

Στο activity υπάρχουν 2 buttons, ένα για σταμάτημα της αποστολής δεδομένων από τον χρήστη και ένα για επιστροφή στο MainActivity, αφού πρώτα σταματήσει η αποστολή των δεδομένων. Επίσης υπάρχει ένας χάρτης που δείχνει παράλληλα την πραγματική πορεία του τερματικού και την προβλεπόμενη πορεία του τερματικού. Ακόμη υπάρχουν 2 διακόπτες για εμφάνιση/απόκρυψη των δύο πορείων.



onResume()

Γίνεται έλεγχος για σύνδεση σε δίκτυο, αλλά και για πρόσβαση στο διαδίκτυο ανά 5 δευτερόλεπτα.

onBackPressed()

Επιστρέφει το χρήστη στο MainActivity, αφού πρώτα σταματήσει η αποστολή των δεδομένων, όταν πατηθεί το back.

onPause()

Σταματάει τους ελέγχους όταν η δραστηριότητα δεν είναι στο προσκήνιο.

isNetworkConnected()

Συνάρτηση για έλεγχο αν το android τερματικό είναι συνδεδεμένο σε κάποιο δίκτυο.

isConnected()

Συνάρτηση για έλεγχο αν το android τερματικό έχει πρόσβαση στο διαδίκτυο.

onMapReady()

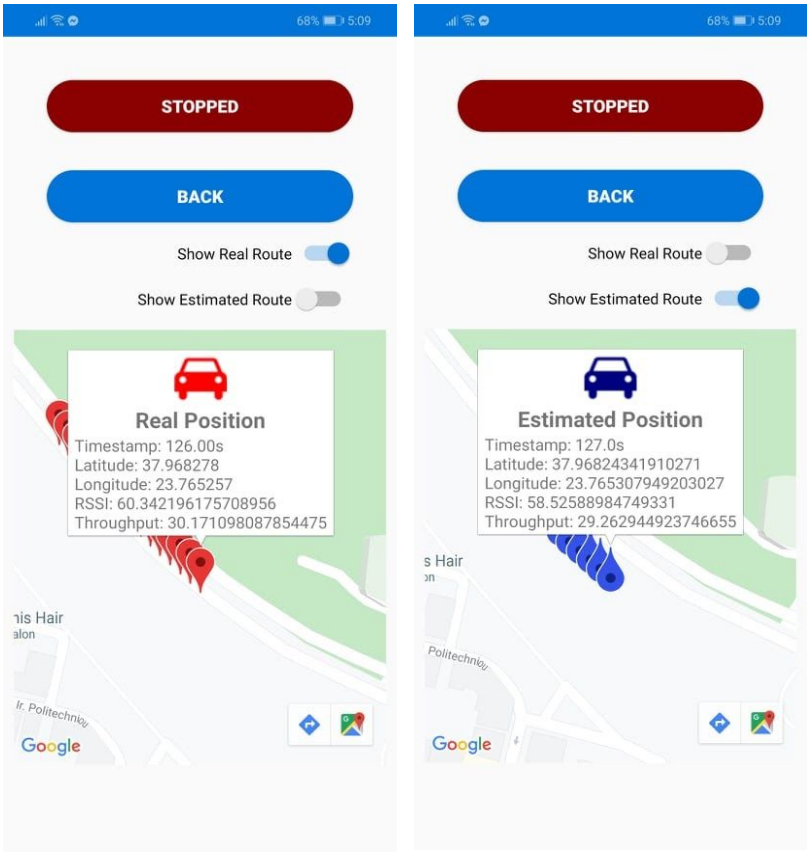
Έπειτα από τη δημιουργία του Google Map το τερματικό συνδέεται στο broker και κάνει subscribe στο κατάλληλο topic. Σε περίπτωση σφάλματος τερματίζει το Activity και επιστρέφει τον χρήστη στο MainActivity. Αν γίνει σύνδεση δημιουργεί ένα νέο thread και στέλνει μέσα από αυτό τα δεδομένα στο Edge Server μέχρι να σταλούν όλα τα δεδομένα ή διακοπεί η αποστολή με άλλο τρόπο (stop, back, λήξη simulation time), ανάμεσα στις αποστολές υπάρχει ένα wait ώστε να εξασφαλίζει την αποστολή των vectors ανα δευτερόλεπτο.

messageArrived()

Η συνάρτηση messageArrived(String topic, MqttMessage message) καλείται όταν ληφθεί ένα μήνυμα μέσω του broker είτε από τον Edge Server, είτε από το ίδιο το κινητό. Έπειτα δημιουργεί τον κατάλληλο marker ανάλογα με το αν είναι μέρος της πραγματικής ή της προβλεπόμενης διαδρομής.

PopupAdapter

Εμφανίζει ένα Information window για κάθε marker σε περίπτωση που το επιλέγει ο χρήστης με τις πληροφορίες του μηνύματος(Timestamp, Latitude, Longitude, RSSI, Throughput).



Οδηγίες χρήσης

Mosquitto Broker

Για την εκτέλεση του Local Mosquitto Broker πρέπει αρχικά να επιτρέψουμε τις συνδέσεις tcp στο port 1883 με την εντολή `sudo ufw allow 1883`. Στη συνέχεια ανοίγουμε τον Broker με τη `mosquitto -v` .

Edge Server

Για την εκτέλεση του Edge Server εκτελούμε την παρακάτω εντολή μέσα στο φάκελο του Edge Server, όπου ως “broker’s IP” ορίζετε την διεύθυνση IP του υπολογιστή που εκτελείται ο mosquitto broker.

```
java -classpath
out/production/project:lib/dom-2.3.0-jaxb-1.0.6.jar:lib/mysql-conn
ector-java-8.0.21.jar:lib/org.eclipse.paho.client.mqttv3-1.2.5.jar
:lib/jcommon-1.0.23.jar:lib/jfreechart-1.0.19.jar project.Main
"broker's IP" 1883
```

Εφαρμογή Android

Απαιτείται απλώς εγκατάσταση της εφαρμογής σε ένα android τερματικό.