ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη εφαρμογής σε κινητά τερματικά για Πλοήγηση σε Εσωτερικούς χώρους

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΚΩΝΣΤΑΝΤΙΝΟΥ Δ. ΠΑΠΑΣΠΥΡΟΥ

Επιβλέπων : Ευστάθιος Δ. Συκάς
Καθηγητής

Αθήνα, Ιούνιος 2012

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη εφαρμογής σε κινητά τερματικά για Πλοήγηση σε Εσωτερικούς χώρους

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΚΩΝΣΤΑΝΤΙΝΟΥ Δ. ΠΑΠΑΣΠΥΡΟΥ

Επιβλέπων : Ευστάθιος Δ. Συκάς
Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 19η Ιουνίου 2012.

.....	.....	.....
Ευστάθιος Συκάς	Μιχαήλ Θεολόγου	Μιλτιάδης Αναγνώστου
Καθηγητής Ε.Μ.Π	Καθηγητής Ε.Μ.Π	Καθηγητής Ε.Μ.Π

Αθήνα, Ιούνιος 2012

.....

Κωνσταντίνος Δ. Παπασπύρου

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Αφιερώνεται
στην οικογένειά μου
και στη Μαρία

Παπασπύρου Κωνσταντίνος

Περιεχόμενα

Πρόλογος

Η τεχνολογική πρόοδος μέσα στην προηγούμενη δεκαετία συνέβαλε στη διάδοση χρήσης των τεχνολογιών εντοπισμού θέσης. Δεδομένου ότι οι τεχνολογίες υπολογισμού και επικοινωνιών έχουν βελτιωθεί και προωθηθεί, εταιρείες όπως οι Garmin Ltd., Tom Tom και Magellan Navigation Inc. έχουν προσφέρει συστήματα εντοπισμού θέσης με αυξημένη χρησιμότητα και λειτουργία. Αυτή την περίοδο αναπτύσσουν εφαρμογές εντοπισμού θέσης και πλοήγησης σε εσωτερικό χώρο, όπως σε αεροδρόμια και σε μεγάλα καταστήματα.

Στην κατεύθυνση αυτή αναπτύξαμε ένα σύστημα πλοήγησης εσωτερικού χώρου. Ο κύριος σκοπός αυτής της διπλωματικής εργασίας είναι η ανάπτυξη μιας εφαρμογής που να παρέχει στο χρήστη τη δυνατότητα να πλοηγηθεί στο εσωτερικό ενός κτηρίου και στη συγκεκριμένη περίπτωση στα Νέα Κτήρια των Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του ΕΜΠ, και να του παρέχει χρήσιμες πληροφορίες. Η παρούσα διπλωματική εργασία θεωρεί δεδομένο τον εντοπισμό του χρήστη στο χώρο και ασχολείται μόνο με τους αλγορίθμους πλοήγησης μέσα σε αυτόν. Για το λόγο αυτό η παρούσα θέση του χρήστη καθορίζεται από τον ίδιο. Η εφαρμογή ECE Indoor Navigation που αναπτύξαμε είναι για κινητά με λειτουργικό σύστημα Android, λόγω της δυναμικής που έχει, αλλά και των πολλών δυνατοτήτων που προσφέρει.

Μέσα από τις γραμμές αυτές θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου κ. Ευστάθιο Συκά για τη διαρκή του καθοδήγηση και τη δυνατότητα που μου έδωσε να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα.

Ιδιαίτερες ευχαριστίες θα ήθελα επίσης να δώσω στον Μιχάλη Μασίκο, υποψήφιο διδάκτορα της σχολής ΗΜΜΥ, στη Δρ. Ευγενία Αδαμοπούλου της σχολής ΗΜΜΥ και στα υπόλοιπα μέλη του Εργαστηρίου Κινητών και Προσωπικών Επικοινωνιών, για τις συμβουλές και την πολύτιμη βοήθεια που μου παρείχαν καθόλη τη διάρκεια της παρούσας διπλωματικής εργασίας καθώς και για την εξαίρετη συνεργασία που είχα μαζί τους.

Τέλος, θα ήθελα να ευχαριστήσω τα υπόλοιπα δύο μέλη της τριμελούς εξεταστικής επιτροπής, τους κ. Μιχαήλ Θεολόγου και κ. Μιλτιάδη Αναγνώστου καθηγητές του Εθνικού Μετσόβιου Πολυτεχνείου.

Ευελπιστώ το παρόν κείμενο να φανεί χρήσιμο στους μελλοντικούς αναγνώστες.

Περίληψη

Ο ακριβής εντοπισμός θέσης σε εσωτερικούς χώρους και η πλοήγηση σε αυτούς αποτελεί τα τελευταία χρόνια ένα δημοφιλή και εξελισσόμενο κλάδο των ασύρματων τηλεπικοινωνιών, με μεγάλη ποικιλία εφαρμογών στους τομείς της βιομηχανίας, της υγείας, της ασφάλειας, αλλά και σε πολλούς άλλους τομείς της καθημερινής ανθρώπινης δραστηριότητας. Η περιορισμένη έκταση των εσωτερικών χώρων, αλλά και οι ιδιαιτέρως απρόβλεπτες και δυσμενείς συνθήκες διάδοσης των σημάτων σε αυτούς, οι οποίες καθιστούν τα γνωστά συστήματα εντοπισμού θέσης εξωτερικών χώρων ανεπαρκή για εντοπισμό σε εσωτερικούς χώρους, αποτελούν πρόκληση για τους σχεδιαστές. Έχει προταθεί ένας μεγάλος αριθμός συστημάτων εντοπισμού θέσης εσωτερικού χώρου και πλοήγησης, καθώς και ένα μεγάλο εύρος τεχνολογιών και μεθόδων εντοπισμού που αφορούν αυτά.

Σημαντικός παράγοντας για την ανάπτυξη τέτοιων συστημάτων ήταν η εμφάνιση έξυπνων συσκευών κινητής τηλεφωνίας, γνωστών και ως smartphones και ειδικότερα καινοτόμων λειτουργικών συστημάτων όπως το Android. Το Android είναι ένα λογισμικό ανοιχτού κώδικα, το οποίο υποστηρίζεται πλήρως από την εταιρεία Google. Στο λογισμικό αυτό μπορεί οποιοσδήποτε χρήστης να αναπτύξει τη δική του εφαρμογή σύμφωνα με τις ανάγκες αλλά και τις επιθυμίες του.

Έτσι, στα πλαίσια της συγκεκριμένης διπλωματικής εργασίας αναπτύχθηκε σε λειτουργικό Android OS ένα σύστημα πλοήγησης σε εσωτερικό χώρο. Στο σύστημα αυτό, θεωρείται δεδομένη η θέση του χρήστη, ο οποίος με τη χρήση κατάλληλων αλγορίθμων δρομολόγησης μπορεί να οδηγηθεί στον προορισμό του. Επίσης, παρέχονται χρήσιμες πληροφορίες στον χρήστη είτε κατά την διάρκεια της πλοήγησης είτε κατά την επεξεργασία της εφαρμογής.

Η διπλωματική εργασία χωρίζεται σε 6 κεφάλαια. Στο πρώτο κεφάλαιο πραγματοποιείται μία βιβλιογραφική επισκόπηση των γνωστών μεθόδων και συστημάτων εντοπισμού θέσης σε εξωτερικούς και εσωτερικούς χώρους. Στο δεύτερο κεφάλαιο παρουσιάζονται οι αλγόριθμοι εύρεσης ελάχιστης απόστασης. Στο τρίτο κεφάλαιο, περιγράφονται τα χαρακτηριστικά, οι δυνατότητες, οι εκδόσεις και άλλες χρήσιμες πληροφορίες του λειτουργικού Android OS. Το τέταρτο κεφάλαιο ασχολείται με την περιγραφή των λειτουργιών της εφαρμογής και δίνονται κάποια παραδείγματα τρεξίματος του αλγορίθμου δρομολόγησης. Στο πέμπτο κεφάλαιο δίνεται μια αναλυτική περιγραφή του συστήματος που αναπτύχθηκε. Τέλος, στο έκτο κεφάλαιο, παρουσιάζονται πιθανές μελλοντικές βελτιστοποιήσεις και επεκτάσεις της εφαρμογής. Στο παράρτημα μπορεί κανείς τα βασικά τμήματα του κώδικα της εφαρμογής.

Λέξεις κλειδιά
Εσωτερική Πλοήγηση, Μέθοδοι εντοπισμού θέσης σε εσωτερικό και εξωτερικό χώρο, Λειτουργικό Σύστημα Android, Αλγόριθμοι δρομολόγησης

Abstract

In the past few years, accurate indoor positioning and navigation constitute one of the most popular and evolving applications of wireless telecommunications, with a great variety of commercial applications in the sectors of industry, health, safety, but also in a lot of other sectors of human activities. The limited extent of indoor spaces, but also especially the unpredicted and unfavorable propagation conditions of signals make the well-known outdoor positioning systems insufficient for indoor positioning. This is a technical challenge for the system designers. A large number of indoor positioning and navigation systems have been proposed, as well as a great range of technologies and methods of localization that concern these.

An important factor for the development of such systems was the emergence of intelligent mobile devices known as smartphones and especially innovative operating systems such as Android. The Android is an open source software, which is fully supported by the company Google. In this software any user is able to develop his own application in accordance with his needs and desires.

The scope of this diploma thesis is to develop an indoor navigation application for Android terminals. This application exploits the existing positioning systems and navigates the user to his destination. Also, vocal messages are generated while using the application.

This diploma thesis consists of 6 chapters. The first chapter contains a literature review of outdoor and indoor positioning methods and systems. The second chapter presents a review of the existing routing algorithms. Chapter 3 includes the characteristics, the capabilities, the versions and other useful information about Android OS. The fourth chapter describes in detail the developed application and presents some generated examples of the routing algorithm. The fifth chapter contains a thorough description of the exploited routing algorithm. Future expansions and optimizations of the application are presented in chapter 6. Finally, in the appendix one can find the basic parts of the programming code.

Key words
Indoor Navigation, Indoor and Outdoor Location Based Techniques, Android OS, Routing algorithms

Κεφάλαιο 1

Υπηρεσίες, τεχνολογίες και τεχνικές εντοπισμού θέσης του κινητού τερματικού

1. Εισαγωγή

Σήμερα, η αυξανόμενη απαίτηση για προηγμένες, εξατομικευμένες, context-aware, ευφυείς και "πάντα διαθέσιμες" εφαρμογές έχει οδηγήσει στη σύγκλιση των πληροφοριών τεχνολογίας και τηλεπικοινωνιακών περιοχών. Η έννοια context-awareness άρχισε να διαδραματίζει έναν ουσιαστικό ρόλο στην ανάπτυξη εφαρμογών.

Οι υπηρεσίες εντοπισμού θέσης (Location-Based Services - LBS) αποτελούν μια δημοφιλή περιοχή των context-aware εφαρμογών. Βρίσκουν εφαρμογή στην βιομηχανία, στα συστήματα υγείας και ασφάλειας, αλλά και σε πάρα πολλές δραστηριότητες της καθημερινής ζωής.

- Ένα σύστημα εντοπισμού θέσης σε εσωτερικό χώρο θα μπορούσε να βοηθήσει στον εντοπισμό ενός ηλικιωμένου ατόμου ή ενός ατόμου με ειδικές ανάγκες ή προβλήματα υγείας, σε ένα νοσοκομείο, σε έναν οίκο ευγηρίας, ή σε ένα οποιοδήποτε μεγάλο κτίριο.

- Παράλληλα θα μπορούσε να βοηθήσει τους υπαλλήλους των φυλακών και την αστυνομία να εντοπίζουν τα ίχνη επικίνδυνων κακοποιών, καθώς και να διευκολύνει στρατιωτικές αποστολές.

- Επίσης, θα ήταν χρήσιμο στον εντοπισμό ενός πυροσβέστη σε ένα φλεγόμενο κτήριο, και στον εντοπισμό εκπαιδευμένων σκύλων της αστυνομίας την ώρα που ψάχνουν εκρηκτικά ή επιζώντες.

Όμως, εκτός από τον εντοπισμό ανθρώπων για λόγους υγείας και ασφάλειας, ένα σύστημα εντοπισμού θέσης σε εσωτερικούς χώρους θα μπορούσε να είναι πολύτιμο και για τον εντοπισμό αντικειμένων.

- Θα μπορούσε να βοηθήσει στον εντοπισμό χρήσιμων εργαλείων και εξαρτημάτων μέσα σε ένα εργοστάσιο, στην εύρεση απαραίτητων αντικειμένων που είναι πιθανό να χαθούν μέσα σε μία αποθήκη, καθώς και στον εντοπισμό απαραίτητου ιατρικού εξοπλισμού μέσα σε ένα νοσοκομείο.

Όμως, πέρα από όλα τα παραπάνω παραδείγματα, υπάρχουν αναρίθμητοι λόγοι για τους οποίους κάποιο πρόσωπο, κάποια εταιρεία ή κάποιος οργανισμός, είναι δυνατό να θελήσει να χρησιμοποιήσει ένα σύστημα εντοπισμού θέσης εμπορικής εφαρμογής σε έναν εσωτερικό χώρο, για δικούς του ιδιαίτερους λόγους. Επομένως, λόγω των πολλών εφαρμογών, είναι αναγκαία η συνεχής βελτίωση των συστημάτων εντοπισμού θέσης εσωτερικών χώρων. Ο τρόπος-εύρεσης σε εσωτερικό χώρο αποτελεί μια μεγάλη πρόκληση για τις μεγαλύτερες εταιρίες ανάπτυξης εφαρμογών εσωτερικού χώρου, κυρίως λόγω της ακαταλληλότητας των ώριμων και ευρέως καθιερωμένων υπαίθριων τεχνολογιών προσδιορισμού θέσης για τη χρήση σε εσωτερικά κτήρια. Το Παγκόσμιο Σύστημα Εντοπισμού Θέσης (Global Positioning System - GPS) είναι μια άριστη τεχνολογία που μπορεί να χρησιμοποιηθεί για τον προσδιορισμό της απόλυτης θέσης στα υπαίθρια περιβάλλοντα, αλλά είναι ακατάλληλο στο εσωτερικό.

Οι υπαίθριες υπηρεσίες πλοήγησης έχουν γίνει παντού διαθέσιμες λόγω των μικρών φορητών συσκευών καθώς τα κινητά τηλέφωνα χρησιμοποιούν το GPS ή κινητά

συστήματα πλοήγησης. Δύο ήταν κύριοι λόγοι οι οποίοι οδήγησαν στην ευρεία επέκταση τον υπηρεσιών εντοπισμού κινητών:

- Απόκτηση των πληροφοριών προσδιορισμού θέσης με ένα συγκεκριμένο βαθμό ακρίβειας

- Ευρεία (κινητής) πρόσβαση των δικτύων υπολογιστών για να χρησιμοποιούν τις πληροφορίες θέσης με μία μεγάλη βάση πληροφοριών όπως το Διαδίκτυο.

Παρατηρώντας την μεγάλη επεκτασιμότητα τον συστημάτων εντοπισμού θέσης, η εσωτερική πλοήγηση και οι υπηρεσίες υπολογισμού θέσης (Location Based Services - LBS) προβλέπεται πως θα χρησιμοποιούνται όπως χρησιμοποιούνται οι υπαίθριες υπηρεσίες τώρα.

1.1. Υπηρεσίες Εντοπισμού θέσης και εσωτερική πλοήγηση

Οι υπηρεσίες εντοπισμού θέσης (LBS) ενσωματώνουν τη γεωγραφική θέση στις υπηρεσίες για να παρέχουν μια ακόμη χρήσιμη πληροφορία για το χρήστη. Οι location based services έχουν ευρεία διάδοση σε πολύ σύντομο χρονικό διάστημα και αυτό γιατί βοήθησε η εξέλιξη στις κινητές συσκευές, η διαθεσιμότητα του GPS και πρόσφατα το κινητό διαδίκτυο. Σήμερα, αυτές οι υπηρεσίες χρησιμοποιούνται εκτενώς μέσα σε υπαίθρια συστήματα πλοήγησης και σε εφαρμογές (web applications) κινητών τηλεφώνων. Η επιτυχία των υπαίθριων συστημάτων πλοήγησης παρουσιάζει την ανάγκη των χρηστών για πλοήγηση και για πληροφορίες εντοπισμού θέσης.

Η πλοήγηση σε handhelds συσκευές παρέχει στο όχημα ή στους πεζούς πλοήγηση και μπορούν να ενσωματώνουν πληροφορίες για την κυκλοφορία στους δρόμους, σημεία ενδιαφέροντος (POI) όπως τα βενζινάδικα ή τους χώρους στάθμευσης, και άλλα. Αυτές οι πρόσθετες πληροφορίες είτε παρέχονται από τη συσκευή, μέσω ενός συστήματος ενημέρωσης είτε φορτώνονται από το διαδίκτυο κατά τη διάρκεια της πλοήγησης. Πολλές ελπιδοφόρες εφαρμογές πλοήγησης για το εσωτερικό των κτηρίων γίνονται πραγματικότητα. Στα μεγάλα κτήρια όπως επιχειρήσεις, εργοστάσια, σταθμοί τραίνων, καταστήματα, αερολιμένες, πανεπιστήμια, στάδια και άλλα, διαφορετικές ομάδες χρηστών θα μπορούσαν να χρησιμοποιήσουν τα διαφορετικά είδη υπηρεσιών:

- Οι δυνάμεις διάσωσης ψάχνουν τη γρηγορότερη διαδρομή για ένα τραυματισμένο άτομο.

- Το προσωπικό συντήρησης ψάχνει πρόσβαση στις σήραγγες υπηρεσιών που είναι κρυμμένες από άλλους χρήστες.

- Οι πελάτες θα μπορούσαν να κοιτάξουν για υπαλλήλους, καταστήματα ή γραφεία και η εφαρμογή θα μπορούσε να τους πλοηγήσει εκεί χρησιμοποιώντας μόνο δημόσιες προσβάσιμες διαδρομές.

1.1.1. INLBS – Indoor Location Based Systems

Δεδομένα για την εσωτερική πλοήγηση / υπηρεσίες εντοπισμού θέσης (Indoor Location Based Systems - INLBS) χαρακτηρίζονται από ένα πιο τοπικό και χρονικό πλαίσιο. Οι εσωτερικές δομές των κτηρίων, όπως οι τοπολογίες, είναι όχι πάντα δημόσια γνωστές και είναι πιθανό να υπάγονται σε συχνές αλλαγές.

Οι υπάλληλοι κινούνται προς άλλα γραφεία ή εγκαταλείπουν τις εργασίες τους, γραφεία σύσκεψης κρατιούνται έτσι τα δωμάτια πρέπει να εμφανιστούν και η θέση των σημαντικών κοινών αλλαγών στις συσκευές, επίσης. Καθώς αυτές οι πληροφορίες είναι μόνο τοπικά διαθέσιμες και ενδεχομένως να υπάρχει πρόσβαση με περιορισμούς, δεν είναι επιθυμητό να μοιραστούν αυτές οι πληροφορίες δημόσια στο διαδίκτυο. Αντ' αυτού, το περιεχόμενο πρέπει να παρέχεται και να ρυθμιστεί από ένα χειριστή ή μια οργάνωση που θα διατηρεί τις πληροφορίες σύγχρονες. Μια τεχνολογία εντοπισμού θέσης με υψηλή ακρίβεια για τον εντοπισμό των χρηστών μέσα στα κτήρια είναι η προϋπόθεση για μια επιτυχημένη προσαρμογή των υπαίθριων LBS σε INLBS.

Γενικά, δύο προϋποθέσεις αποτελούν ουσιαστικούς παράγοντες για INLBS, αυτοί είναι:

- Μια τεχνολογία εσωτερικού προσδιορισμού θέσης που είναι ικανή να παρέχει εκτιμήσεις θέσης σε έναν ορισμένο βαθμό ακριβείας.

- Ασύρματη επικοινωνία προκειμένου να μεταφερθούν τα στοιχεία μεταξύ της υποδομής και των κινητών συσκευών.

Υπάρχουν πρόσθετοι παράγοντες, οι οποίοι ισχύουν για σχεδόν όλες τις φορητές συσκευές όπως η τιμή, οι διαστάσεις και η διάρκεια ζωής των μπαταριών. Αυτοί θεωρούνται δευτερεύων από άλλες συσκευές όπως τα κινητά τηλέφωνα ή τους ψηφιακούς προσωπικούς βοηθούς (Personal Digital Assistants - PDA) οι οποίοι υπόκεινται στους ίδιους παράγοντες.

Τέσσερις παράγοντες οι οποίοι αποτελούν ιδιαίτερης σπουδαιότητας στην εκτίμηση των τεχνικών προσδιορισμού θέσης είναι η ακρίβεια, η ακεραιότητα, η διαθεσιμότητα και η συνοχή. Αυτές οι παράμετροι εστιάζουν στην εξέταση της ποιότητας υπηρεσιών (QoS) για τον κινητό χρήστη συμπεριλαμβανομένης της πλοήγησης και της περιοχής κάλυψης.

- Η <u>ακρίβεια</u> ενός συστήματος είναι το μέτρο της πιθανότητας να υπάρξει ένα λάθος σε μια θέση και σε μία δεδομένη στιγμή.

- Η <u>ακεραιότητα</u> του συστήματος είναι το μέτρο της πιθανότητας ότι το λάθος ακρίβειας είναι μέσα σε ένα συγκεκριμένο όριο.

- Η <u>διαθεσιμότητα</u> ενός συστήματος είναι ένα μέτρο της ικανότητάς της να καλύψει τις απαιτήσεις ακρίβειας και ακεραιότητας ταυτόχρονα.

- Η <u>συνοχή</u> ενός συστήματος είναι ένα μέτρο του ελάχιστου χρονικού διαστήματος για το οποίο η υπηρεσία είναι διαθέσιμη στο χρήστη.

Αυτές οι έννοιες θα χρησιμοποιηθούν αργότερα οπότε ήταν απαραίτητη η ανάλυσή τους.

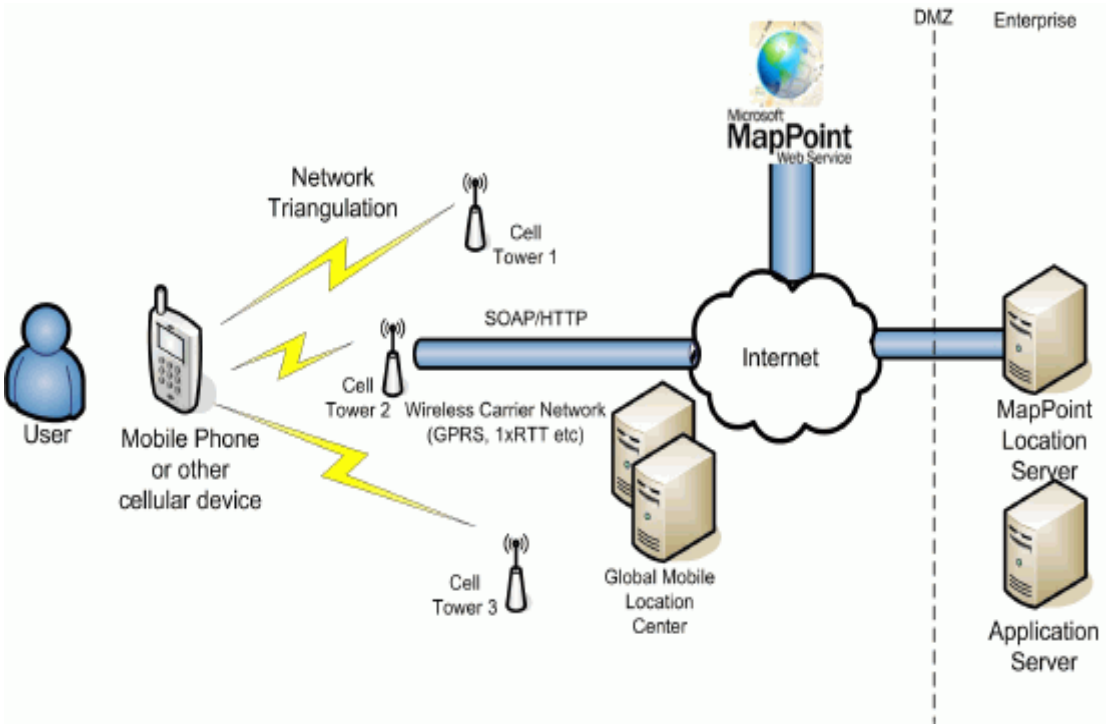
1.2. Λειτουργία συστημάτων εντοπισμού θέσης

Οι LBS είναι προσωποκεντρικές υπηρεσίες που βασίζονται στην τρέχουσα θέση του χρήστη και επομένως χρησιμοποιούν την τεχνολογία GPS και το δίκτυο κινητών επικοινωνιών για να λάβουν τις πληροφορίες τις σχετικές με την τοποθεσία και να τις μεταδώσουν στην πλευρά του server για περαιτέρω επεξεργασία.

Γενικά, όποτε και οπουδήποτε ένας κινητός χρήστης χρειάζεται πληροφορίες ή κάποια υπηρεσία βασισμένη στην τρέχουσα θέση του ή το σύστημα διαχείρισης χρειάζεται να τον εντοπίσει, τότε το GPS ή το σύστημα εντοπισμού βασισμένο στο δίκτυο (network positioning system) παρέχει πληροφορίες θέσης και τις μεταδίδει στο κέντρο. Οι απαιτήσεις του χρήστη αναλύονται από το κέντρο παροχής υπηρεσιών (service provider) και αφού διαμορφωθεί η απάντηση με το χρήστη της κεντρικής βάσης δεδομένων, οι ζητούμενες πληροφορίες στέλνονται πίσω στο κινητό τερματικό.

Συμπερασματικά, οι υπηρεσίες που βασίζονται στη θέση του χρήστη περιλαμβάνουν δύο κύριες διαδικασίες:

- «Συλλογή» της πληροφορίας της θέσης/τοποθεσίας του κινητού τερματικού
  .
- Παροχή της υπηρεσίας σύμφωνα με τις προδιαγραφές που θέτει ο ίδιος ο χρήστης



Σχήμα 1. Η λειτουργία των LBS

1.3. Μέθοδοι εντοπισμού θέσης

Οι τεχνολογίες εντοπισμού της θέσης ενός χρήστη έχουν μελετηθεί διεξοδικά τα τελευταία χρόνια καθότι αφενός επιτρέπουν την παροχή υπηρεσιών θέσης και αφετέρου είναι εξαιρετικά σημαντικές για την παροχή υπηρεσιών έκτακτης ανάγκης. Μέχρι σήμερα έχουν αναπτυχθεί και προταθεί αρκετές μέθοδοι για την εκτίμηση της θέσης των κινητών τερματικών, οι οποίες αν εξαιρεθεί η περίπτωση που ο χρήστης προσδιορίζει χειροκίνητα τη θέση του, τότε μπορούν να ταξινομηθούν σε δύο κατηγορίες, στον αυτό-εντοπισμό (self-positioning) και στον απομακρυσμένο εντοπισμό (remote-positioning):

- Self-positioning: Το κινητό τερματικό χρησιμοποιεί σήματα, μεταδιδόμενα από τις κεραίες (οι οποίες μπορεί να είναι είτε επίγειες είτε δορυφόροι) για να υπολογίσει τη θέση του. Έτσι, η φυσική θέση αυτοκαθορίζεται από τη συσκευή του χρήστη χρησιμοποιώντας τα μεταδιδόμενα σήματα από τα επίγεια ή δορυφορικά αναγνωριστικά σήματα. Ειδικότερα, ο δέκτης κάνει κατάλληλες μετρήσεις σημάτων από τους γεωγραφικά διανεμημένους πομπούς σημάτων και χρησιμοποιεί τις μετρήσεις αυτές για να καθορίσει τη θέση του. Συνεπώς, ο δέκτης "ξέρει" που είναι η θέση του και μπορεί να χρησιμοποιηθεί από τις εφαρμογές και τις υπηρεσίες που λειτουργούν στην κινητή συσκευή του χρήστη όπως για την πλοήγηση οχημάτων.

- Remote-positioning: Σε αυτή την περίπτωση, η θέση καθορίζεται στην πλευρά κεντρικών υπολογιστών (Servers) χρησιμοποιώντας τα εκπεμπόμενα σήματα από τη συσκευή του χρήστη. Η θέση κατόπιν είτε χρησιμοποιείται από τον server σε ένα σύστημα εντοπισμού θέσης, είτε μεταδίδεται πίσω στη συσκευή μέσω μιας μεθόδου μεταφοράς δεδομένων. Σε αυτήν την περίπτωση το κινητό τερματικό μπορεί να βρεθεί με τη μέτρηση των σημάτων που ταξιδεύουν σε και από ένα σύνολο δεκτών. Συγκεκριμένα, οι δέκτες που μπορούν να εγκατασταθούν σε μία ή περισσότερες θέσεις μετρούν ένα σήμα που προέρχεται από το αντικείμενο που επιθυμείται να προσδιοριστεί η θέση του. Αυτές οι μετρήσεις των σημάτων χρησιμοποιούνται για να καθορίσουν το μήκος ή/και την κατεύθυνση των ράδιο διαδρομών και έπειτα η θέση του κινητού τερματικού υπολογίζεται από τις γεωμετρικές σχέσεις.

1.3.1. Τεχνολογίες Self-positioning

Οι ενότητες που ακολουθούν περιγράφουν τις πιθανές τεχνολογίες που μπορούν να χρησιμοποιηθούν σε ένα τέτοιο σύστημα.

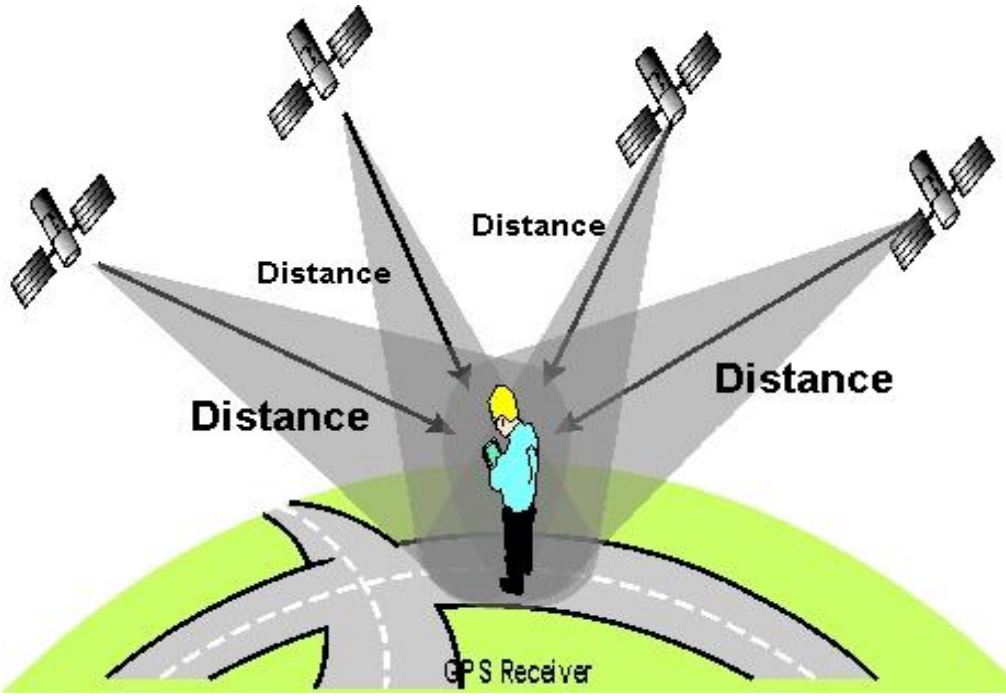
1.3.1.1. Δορυφόροι (satellites)

Τα δορυφορικά συστήματα πλοήγησης παρέχουν γεο-χωρικό προσδιορισμό θέσης με παγκόσμια κάλυψη. Αυτήν την περίοδο υπάρχουν διάφορα παγκόσμια δορυφορικά συστήματα πλοήγησης που χρησιμοποιούνται στον αστικό προσδιορισμό θέσης συμπεριλαμβανομένου του αμερικάνικου US Navstar παγκόσμιου συστήματος εντοπισμού (GPS), του ρωσικού Glonass, και του Γαλιλαίου (Galileo) της Ευρωπαϊκής Ένωσης. Το πλεονέκτημα των δορυφορικών συστημάτων είναι ότι οι δέκτες μπορούν να καθορίσουν το γεωγραφικό πλάτος, το γεωγραφικό μήκος, και το ύψος σε έναν υψηλό βαθμό ακρίβειας. Εντούτοις, η γραμμή θέας (Line Of Sight - LOS) απαιτείται για τη λειτουργία αυτών των συστημάτων. Αυτό οδηγεί σε ανικανότητα, στο να χρησιμοποιηθούν αυτά τα συστήματα για ένα εσωτερικό περιβάλλον όπου το LOS εμποδίζεται από τους τοίχους και τις στέγες.

1.3.1.1.1 Παγκόσμιος Προσδιορισμός Θέσης (GPS)

Το Global Positioning System - GPS σχεδιάστηκε αρχικά για το υπουργείο Άμυνας των ΗΠΑ και ονομάστηκε "NAVSTAR GPS" (Navigation Signal Timing and Ranging Global Positioning System), αλλά τα τελευταία χρόνια χρησιμοποιείται ευρέως σε όλο τον κόσμο. Το σύστημα έχει τα εξής χαρακτηριστικά :

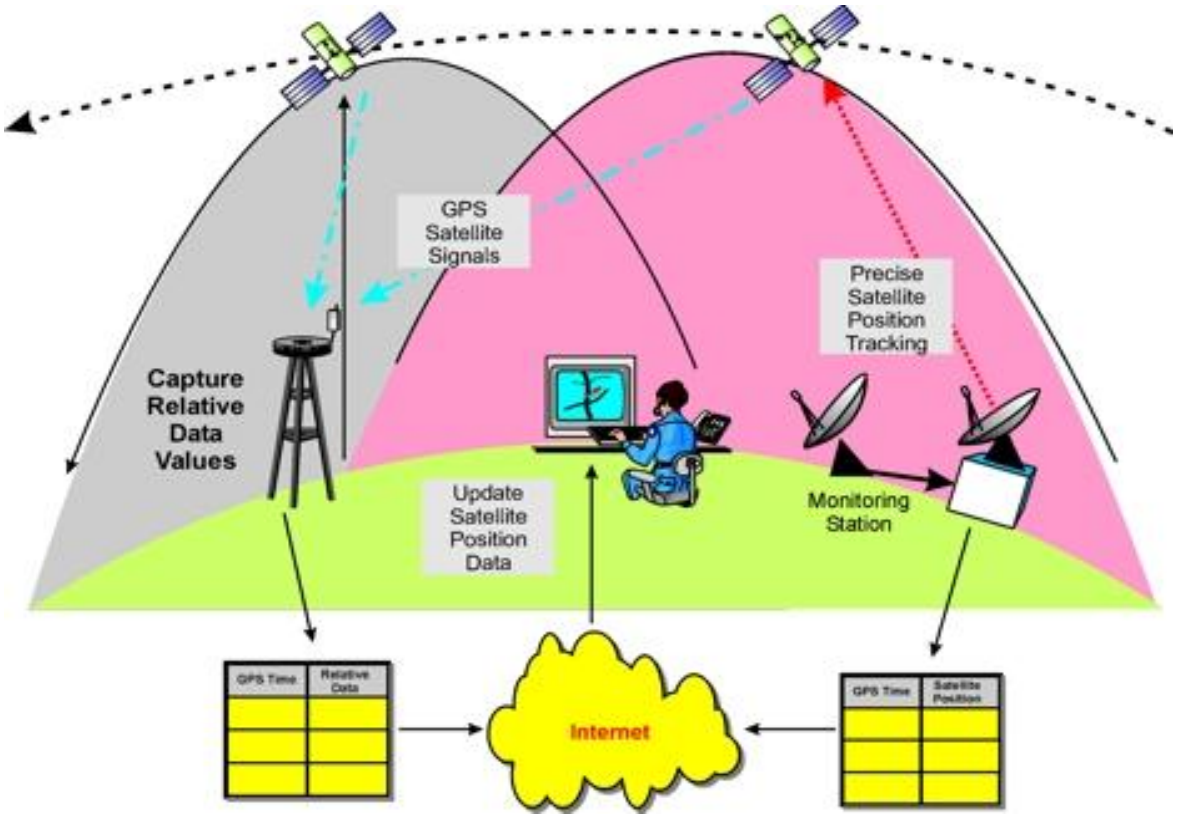
- ✓ Είναι ένα ημιακριβές παγκόσμιο σύστημα εντοπισμού και πλοήγησης για τις υπαίθριες εφαρμογές.
- ✓ Το σύστημα GPS αποτελείται από 24 δορυφόρους ίσων αποστάσεων σε έξι τροχιακά επίπεδα 20.200 χλμ επάνω από τη γη.
- ✓ Είναι σχεδιασμένο κατά τέτοιο τρόπο ώστε όλοι οι δορυφόροι να βρίσκονται σε κλίση 55^o ως προς το επίπεδο του ισημερινού, σε 60^o μεταξύ τους ο ένας από τον άλλο στο επίπεδο της κάθε τροχιάς.
- ✓ Η ακρίβεια των συσκευών GPS βελτιώνεται συνεχώς αλλά έχουν ακόμα εύρος 5-6 μέτρα στον ανοιχτό χώρο.
- ✓ Μια συσκευή GPS δεν μπορεί να χρησιμοποιηθεί για ένα εσωτερικό περιβάλλον επειδή το LOS εμποδίζεται.
- ✓ Έχει αποδειχθεί πως 4 ή περισσότεροι δορυφόροι που βρίσκονται σε οπτική επαφή με το κινητό τερματικό επαρκούν για να προσδιορίσουν τις γεωγραφικές συντεταγμένες του και μάλιστα με πολύ καλή ακρίβεια, της τάξης 3-50m για το 95% του χρόνου (Σχήμα 2).



Σχήμα 2. GPS – Οπτική επαφή τουλάχιστον τεσσάρων δορυφόρων για τον εντοπισμό του κινητού τερματικού

Εκτός από το διαστημικό του τμήμα, το σύστημα GPS απαρτίζεται και από επίγειο τμήμα ελέγχου με ένα επανδρωμένο τμήμα επίγειου ελέγχου και τέσσερα μη επανδρωμένα κέντρα σε ισάριθμες περιοχές του πλανήτη. Τα κέντρα αυτά πραγματοποιούν ελέγχους όσον αφορά την ταχύτητα, το υψόμετρο και την επάρκεια των δορυφόρων σε ηλεκτρική ενέργεια. Παράλληλα, γίνονται και οι όποιες διορθωτικές ενέργειες απαιτούνται και αφορούν το σύστημα χρονομέτρησης των δορυφόρων ώστε να αποτρέπεται η παροχή λανθασμένων πληροφοριών στους χρήστες του συστήματος (Σχήμα 3).

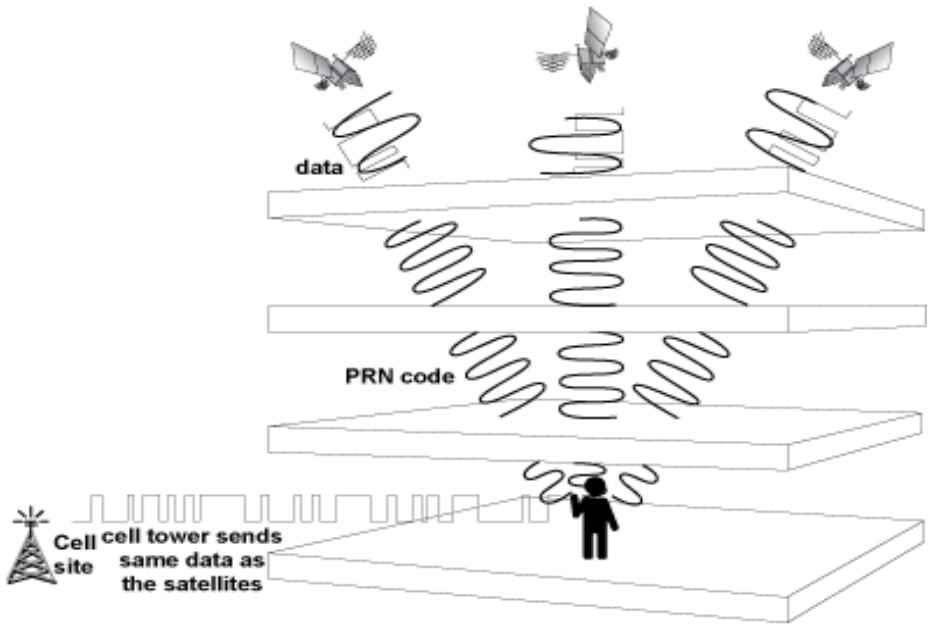
Ο κυριότερος σταθμός βάσης είναι αυτός στο Colorado Springs των ΗΠΑ, ο οποίος είναι μάλιστα και ο μοναδικός που βρίσκεται στη ξηρά. Αναλαμβάνει τον έλεγχο της σωστής λειτουργίας των εναπομεινάντων τεσσάρων σταθμών, καθώς και τον συντονισμό τους. Σημειώνοντας τη θέση των σταθμών αυτών πάνω σε έναν παγκόσμιο χάρτη, παρατηρεί κανείς ότι η διάταξη τους δεν είναι τυχαία, αλλά ακολουθούν μια γραμμή παράλληλη με τα γεωγραφικά μήκη.



Σχήμα 3. GPS

1.3.1.1.2. Assisted-GPS

Για να βελτιωθεί η αξιοπιστία και να μειωθεί ο χρόνος αποτύπωσης θέσης, ένα ασύρματο δίκτυο πληροφορίας μπορεί να συνδυαστεί με τον Παγκόσμιο προσδιορισμό θέσης GPS, γνωστό ως Assisted-GPS (A-GPS). Το A-GPS χρησιμοποιείται πρώτιστα στα κινητά τηλέφωνα. Η μέθοδος A-GPS ενισχύεται από ένα τρίτο φορέα παροχής υπηρεσιών, όπως ένα δίκτυο κινητών τηλεφώνων, για να βοηθήσει την κινητή συσκευή με την καθοδήγηση στο να αναζητήσει τον συγκεκριμένο δορυφόρο (Σχήμα 4). Επίσης, τα δεδομένα από την ίδια τη συσκευή χρησιμοποιούνται για να εκτελέσουν τους υπολογισμούς προσδιορισμού θέσης που άλλωστε μπορεί να μην οφείλονται στην περιορισμένη υπολογιστική ισχύ. Το A-GPS είναι χρήσιμο όταν μερικά δορυφορικά σήματα είναι αδύνατα ή μη διαθέσιμα. Ο πύργος κυττάρων παρέχει τις πληροφορίες οι οποίες βοηθούν το δέκτη GPS (Σχήμα 5). Κατά τη χρησιμοποίηση του A-GPS, η ακρίβεια είναι τυπικά περίπου 10-20 μέτρα αλλά υφίσταται παρόμοιους εσωτερικούς περιορισμούς με το αυτόνομο GPS.



Σχήμα 4: Η Assisted-GPS μέθοδος

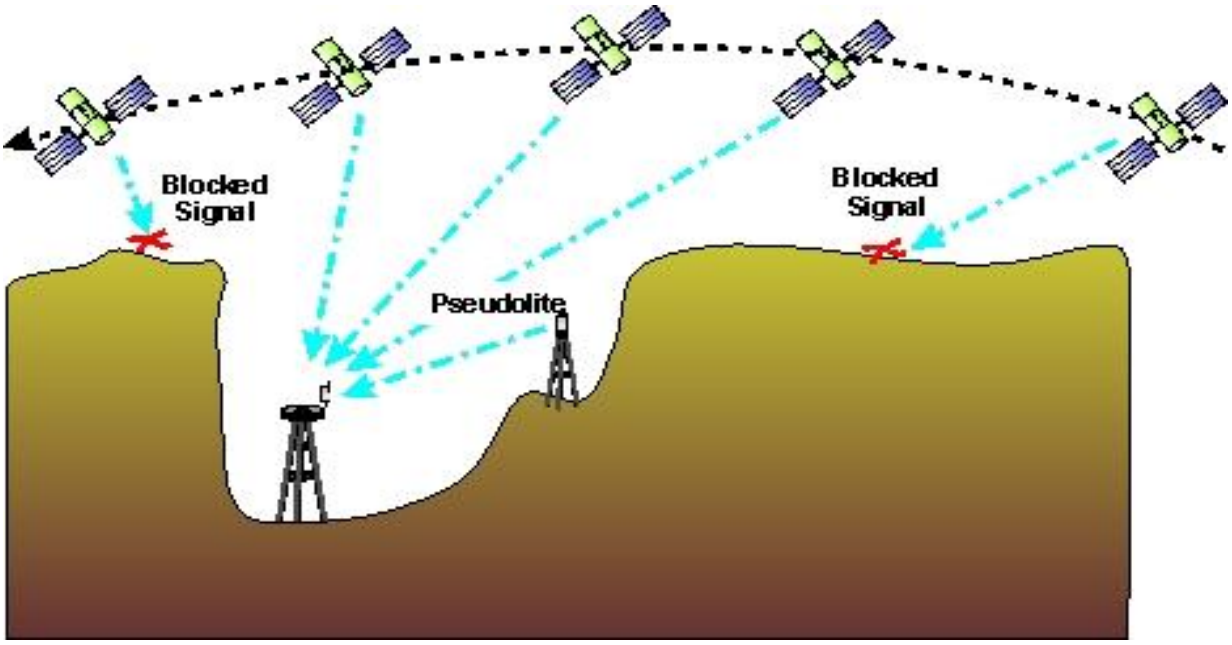
Σχήμα 5. Η δομή του A-GPS

1.3.1.1.3. Pseudolite GPS

Διάφορες μέθοδοι έχουν αναπτυχθεί για να υπερνικήσουν την απαίτηση LOS του GPS. Μία μέθοδος είναι με την οργάνωση «ψεύτικων» (pseudo-satellite) συστημάτων που μιμούνται τους δορυφόρους GPS με την αποστολή GPS - όπως τα σήματα διόρθωσης στο δέκτη μέσα στο κτήριο ως Pseudolite GPS (PL). Τα PLs είναι συχνά μικροί πομποδέκτες που χρησιμοποιούνται για να δημιουργήσουν έναν τοπικό, επίγειο σταθμό βάσης GPS (Σχήμα 6). Το εύρος του σήματος κάθε πομποδέκτη εξαρτάται από τη διαθέσιμη δύναμη κάθε μονάδας.

Μάλιστα έχει αναπτυχθεί από το εθνικό πανεπιστημιακό GPS εργαστήριο της Σεούλ (National University GPS Lab of Seoul) ένα σύστημα, με το οποίο επιτυγχάνεται ο εντοπισμός θέσης κινητού με ακρίβεια εκατοστών για σύστημα GPS εσωτερικής πλοήγησης. Αυτό το σύστημα έχει έναν χρόνο σύγκλισης κάτω από 0,1 δευτερόλεπτα, το οποίο βοηθά να αυξηθεί η ανταπόκριση για έναν κινητό τερματικό. Αυτό το σύστημα χρησιμοποιεί PL και ένα σταθμό αναφοράς για να βοηθήσει ένα κινητό όχημα GPS σε ένα εσωτερικό περιβάλλον. Τα PLs έχουν μια σταθερή θέση και χρησιμοποιούν ένα αντίστροφο φορέα διαφορικού φάσης GPS για να υπολογίσουν τη θέση του κινητού χρήστη. Ο σταθμός αναφοράς επίσης είναι καθορισμένος και μεταδίδει το φορέα διορθωμένης φάσης στο κινητό τερματικό. Το σύστημα αντιμετωπίζει διάφορες προκλήσεις συμπεριλαμβανομένου των σοβαρών λαθών των πολλαπλών διαδρομών διάδοσης και των αυστηρών απαιτήσεων συγχρονισμού PL. Η χρησιμοποίηση ενός κέντρου PL λύνει το πρόβλημα συγχρονισμού. Το πρωτότυπο έχει επιτύχει το στατικό λάθος να είναι μόλις 0,14 εκατοστά και το δυναμικό λάθος να είναι 0,79 εκατοστά.

Η γεωμετρική διαμόρφωση PLs είναι πολύ σημαντική και γίνεται απαραίτητη για τις εφαρμογές εσωτερικού χώρου. Ακόμα κι αν τα PLs βελτιώνουν γενικά την ακρίβεια προσδιορισμού θέσης, οι χρήστες πρέπει να γνωρίζουν τις αρνητικές πτυχές τους. Παραδείγματος χάριν, υπάρχει μια πιο σύντομη απόσταση μεταξύ μιας PL και του δέκτη χρηστών απ' ότι με έναν δορυφόρο GPS, κατά τέτοιο τρόπο ώστε οποιαδήποτε λάθη στη θέση της PL θα έχουν μια σημαντική επίδραση στις καθορισμένες συντεταγμένες της κεραίας του δέκτη. Ο βαθμός επιρροής εξαρτάται από τη γεωμετρία μεταξύ των PLs και του δέκτη (η αδύνατη γεωμετρία μπορεί ακόμη και να προκαλέσει μια ιδιομορφία στη λύση). Επιπλέον, αυτό το σύστημα είναι πολύ οικονομικά δαπανηρό για να εφαρμοστεί λόγω της απαίτησης για έναν μεγάλο αριθμό PLs.



Σχήμα 6. Pseudo-satellite GPS

1.3.1.2. RFID συστήματα

Ένα σύστημα αναγνώρισης ραδιοσυχνοτήτων (Radio Frequency Identification: RFID) έχει την ικανότητα να αναγνωρίζει συγκεκριμένα αντικείμενα ή πρόσωπα, χρησιμοποιώντας ραδιοκύματα. Το RFID σύστημα λειτουργεί με τον εξής τρόπο:

- Αρχικά, ένας πομποδέκτης RF συχνοτήτων στέλνει ένα RF σήμα σε έναν αναμεταδότη. Ο αναμεταδότης αυτός λέγεται RFID tag και περιλαμβάνει όλες τις χαρακτηριστικές πληροφορίες που χρειάζεται το RFID σύστημα για να τον αναγνωρίσει. Ο RFID tag εισάγει στο σήμα που έλαβε τις χαρακτηριστικές του πληροφορίες, και επανεκπέμπει τη νέα εκδοχή του RF σήματος προς τον πομποδέκτη.

- Στη συνέχεια, ο πομποδέκτης προωθεί το σήμα σε μία μονάδα, η οποία λέγεται RFID reader. Ο RFID reader επεξεργάζεται ψηφιακά το σήμα και έτσι αποκτά την χαρακτηριστική πληροφορία του RFID tag. Οι RFID readers και tags επικοινωνούν μεταξύ τους ασύρματα, ακολουθώντας ένα καθορισμένο πρωτόκολλο επικοινωνίας.

Οι μονάδες RFID tag χωρίζονται σε δύο κατηγορίες: στις παθητικές (passive) και στις ενεργητικές (active).

- Τα passive RFID tags έχουν σημαντικά πλεονεκτήματα, όπως ότι λειτουργούν χωρίς μπαταρία, όμως το μεγάλο τους μειονέκτημα είναι ότι το εύρος λειτουργίας τους είναι πολύ μικρό, και κυμαίνεται από 1 ως 2 m.

- Αντίθετα, τα active RFID tags λειτουργούν σε πολύ μεγαλύτερο εύρος, το οποίο μπορεί να είναι και δεκάδες μέτρα, όμως χρησιμοποιούν μπαταρία, η οποία έχει περιορισμένη διάρκεια ζωής.

Τα passive RFID συστήματα συνήθως χρησιμοποιούν τις εξής μπάντες συχνοτήτων: LF (125 kHz), HF (13.56MHz), UHF (433 , 868-915 MHz), και μικροκυματικές συχνότητες (2.45 GHz, 5.8 GHz). Οι μπάντες συχνοτήτων που χρησιμοποιούν τα active RFID συστήματα είναι όμοιες με αυτές των passive, εκτός από τις LF και HF μπάντες.

	Passive RFID	Active RFID
Πηγή Ενέργειας	Η ενέργεια μεταφέρεται από τον RFID reader μέσω RF	Εσωτερικά στον αναμεταδότη
Μπαταρία	Όχι	Ναι
Διαθεσιμότητα της ισχύς του αναμεταδότη	Μόνο μέσα στο πεδίο του RFID reader	Συνέχεια
Απαίτηση για ισχύ σήματος από τον reader στον αναμεταδότη	Πολύ υψηλή	Πολύ χαμηλή
Διαθεσιμότητα ισχύς σήματος από τον αναμεταδότη στον reader	Πολύ χαμηλή	Υψηλή
Εύρος επικοινωνίας	Χαμηλό ή πολύ χαμηλό (3m ή και λιγότερο)	Μεγάλο (100m ή περισσότερο)
Ικανότητα του αισθητήρα	Ικανότητα να διαβάζει και να μεταφέρει τις τιμές μόνο του αισθητήρα όταν ο αναμεταδότης λειτουργεί λόγω της ισχύς του reader	Ικανότητα να δείχνει και να αποθηκεύει συνεχώς τα δεδομένα που λαμβάνει (ημερομηνία / ώρα)
Αποθήκευση δεδομένων	Μικρός αποθηκευτικός χώρος για διάβασμα/εγγραφή (128 Bytes)	Μεγάλος αποθηκευτικός χώρος για διάβασμα/εγγραφή (128KB) με έξυπνη αναζήτηση δεδομένων και δυνατότητα πρόσβασης στα δεδομένα

Πίνακας 1. Χαρακτηριστικά των Passive και Active RFID tags

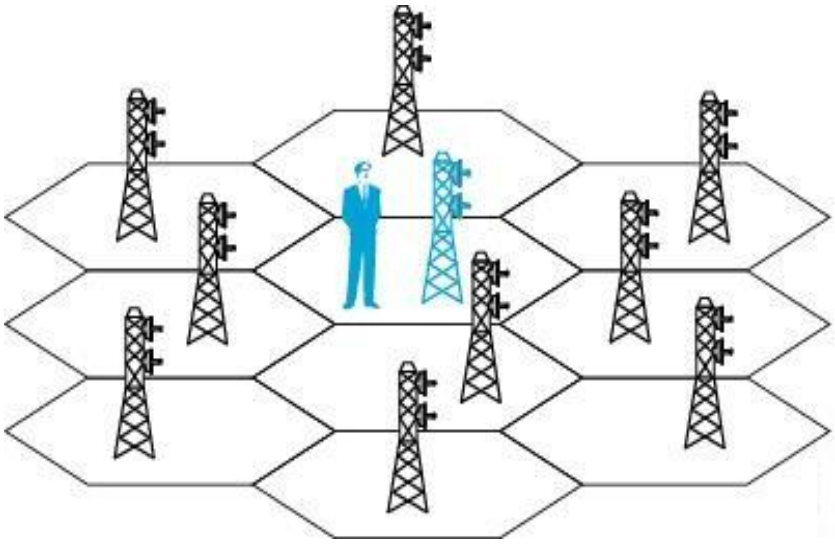
Ένα πολύ γνωστό σύστημα εντοπισμού θέσης, το οποίο χρησιμοποιεί την RFID τεχνολογία, είναι το SpotON. Το SpotON σύστημα χρησιμοποιεί έναν συναθροιστικό αλγόριθμο που βασίζεται στην ανάλυση RSS τιμών, για να εντοπίσει ένα αντικείμενο στον τρισδιάστατο (3-D) χώρο. Η τοπολογία ενός SpotON δικτύου δεν βασίζεται σε κάποια κεντρική διαχείριση, αλλά οι δομικές μονάδες του δικτύου επικοινωνούν με Ad Hoc τρόπο. Οι δομικές μονάδες λειτουργούν σαν αισθητήρες που συνεργάζονται μεταξύ τους για να εντοπίσουν τη θέση ενός αντικειμένου. Οι RSS τιμές των σημάτων που λαμβάνουν τα SpotON tags χρησιμοποιούνται με κατάλληλο τρόπο ώστε να υπολογιστούν οι αποστάσεις μεταξύ των tags. Η accuracy και η precision ενός συστήματος μπορούν να βελτιωθούν, αν εκμεταλλευτεί με σωστό τρόπο η πυκνότητα των tags και η συσχέτιση των πολλαπλών μετρήσεων.

Ένα άλλο σύστημα εντοπισμού θέσης εσωτερικού χώρου που χρησιμοποιεί ενεργητική (active) RFID τεχνολογία είναι το LANDMARC. Το σύστημα αυτό συνηθίζεται να

λειτουργεί στα 308 MHz. Μία ιδέα που εισάγει ένα LANDMARC σύστημα, είναι η αύξηση της ακρίβειας (accuracy) χωρίς την τοποθέτηση επιπλέον readers, αλλά με την τοποθέτηση επιπρόσθετων ετικετών αναφοράς (reference tags) που αφορούν καθορισμένες θέσεις στο χώρο. Τα reference tags βοηθούν στην βαθμονόμηση (calibration) του χώρου, και παίζουν το ρόλο των σημείων αναφοράς (reference points) στο σύστημα. Το LANDMARC σύστημα χρησιμοποιεί τις RSS μετρήσεις μεταξύ tag και readers. Ο αλγόριθμος που εφαρμόζει ένα LANDMARC σύστημα για να εντοπίσει μία θέση είναι ο kNN (k-Nearest-Neighbor). Έχει διαπιστωθεί ότι η απόσταση σφάλματος σε ένα LANDMARC σύστημα είναι περίπου 1 m σε ποσοστό 50%, ενώ οι μέγιστες αποστάσεις σφάλματος είναι μικρότερες από 2 m.

1.3.1.3. Κυψελοειδές δίκτυο επικοινωνίας

Κυψελοειδές δίκτυο επικοινωνίας είναι ένα σύστημα που επιτρέπει στα κινητά τηλέφωνα να επικοινωνούν το ένα με το άλλο. Αυτό το σύστημα χρησιμοποιεί τους μεγάλους κυψελωτούς πύργους για να συνδέσει ασύρματα τις κινητές συσκευές. Το εύρος των κυψελοειδών δικτύων επικοινωνίας εξαρτάται από την πυκνότητα των μεγάλων κτηρίων, των δέντρων και άλλων πιθανών παρεμποδίσεων. Το μέγιστο εύρος για έναν κυψελωτό πύργο είναι 35 χιλιόμετρα σε μια ανοικτή αγροτική περιοχή. Αυτή η μέθοδος είναι μια βασική τεχνική που χρησιμοποιεί το Cell-ID, αποκαλούμενη επίσης και ως κυψέλη προέλευσης (Cell of Origin), για να παρέχει τις υπηρεσίες θέσης για τους χρήστες κινητών τηλεφώνων. Αυτή η μέθοδος βασίζεται στην ικανότητα του δικτύου να υπολογίζει τη θέση ενός κινητού τηλεφώνου με την αναγνώριση του κυψελωτού πύργου που η συσκευή χρησιμοποιεί σε έναν συγκεκριμένο χρόνο (Σχήμα 7). Το πλεονέκτημα αυτής της τεχνικής είναι η πανταχού παρούσα διανομή, η εύκολη εφαρμογή του και το γεγονός ότι όλα τα κινητά τηλέφωνα την υποστηρίζουν. Η ακρίβεια αυτής της τεχνικής είναι πολύ μικρή εξαιτίας του γεγονότος ότι οι κυψελωτοί πύργοι μπορούν να υποστηρίξουν εύρος 35 χιλιομέτρων ή και περισσότερο. Στα αστικά περιβάλλοντα, οι κυψελωτοί πύργοι διανέμονται πιο πυκνά.



Σχήμα 7. Κυψελοειδές δίκτυο επικοινωνίας

1.3.1.4. Wi-Fi

Το Wireless Fidelity (Wi-Fi) είναι το κοινό παρωνύμιο για τα IEEE 802.11 πρότυπα. Η ασύρματη σύνδεση θεωρείται ως η πλέον διαδεδομένη και επικρατούσα στην καθημερινή ζωή. Κάθε ασύρματος δρομολογητής μεταδίδει ένα σήμα το οποίο λαμβάνεται από τις συσκευές σε αυτή τη περιοχή. Οι ασύρματες συσκευές έχουν την ικανότητα να μετρήσουν την ισχύ αυτού του σήματος. Αυτή η ισχύς μετατρέπεται σε έναν αριθμό, γνωστό ως δείκτης λαμβανόμενης ισχύος σημάτων (RSSI). Η συσκευή ενός χρήστη μπορεί να ανιχνεύσει το RSSI και τη MAC διεύθυνση πολλαπλών δρομολογητών συγχρόνως (Σχήμα 8).

Το RSSI είναι αδιάστατο μέτρο που χρησιμοποιείται από τα συστήματα για να συγκρίνει την ισχύ των σημάτων από τα πολλαπλά σημεία πρόσβασης. Δεν υπάρχει καμία τυποποιημένη μετατροπή μεταξύ RSSI και της πραγματικής λαμβανόμενης ισχύος σημάτων (RSS). Πολλοί κατασκευαστές έχουν τα σχέδια μετατροπής τους. Τα σημαντικότερα χαρακτηριστικά που προκύπτουν από τη μετατροπή του RSSI σε RSS είναι οι μέγιστες και ελάχιστες τιμές RSSI (αδιάστατοι ακέραιοι αριθμοί), οι μέγιστες και ελάχιστες τιμές RSS που μπορούν να αντιπροσωπευθούν (dBm) και η ανάλυση της μετατροπής (τιμή σε dBm που αντιπροσωπεύεται από μια μονάδα RSSI). Ο πίνακας 2 περιλαμβάνει αυτές τις ποσότητες για τους κοινούς κατασκευαστές.

Κατασκευαστής	RSSI Min	RSSI Max	RSS Min	RSS Max	Resolution
Atheros	0	60	-95dBm	-35dBm	1dBm
Symbol	0	31	-100dBm	-50dBm	10dBm
Cisco	0	100	-113dBm	-10dBm	1dBm

Πίνακας 2. Κοινές μετατροπές RSSI σε RSS

Οι Wi-Fi συσκευές όπως οι φορητοί υπολογιστές και τα έξυπνα κινητά (smartphones) χαρακτηριστικά εκτελούν αυτήν την μετατροπή προκειμένου να παρέχουν αυτόματα τις πληροφορίες ισχύος σήματος στις εφαρμογές που τρέχουν στη συσκευή. Ένα πλεονέκτημα του Wi-Fi είναι ότι τα ασύρματα δίκτυα είναι παγκόσμια. Υπάρχουν σε πολυπληθυσμικές περιοχές και διαδίδονται συνεχώς εξωτερικά. Αυτό προκαλεί τα Wi-Fi συστήματα να έχουν χαμηλότερο κόστος εφαρμογής.

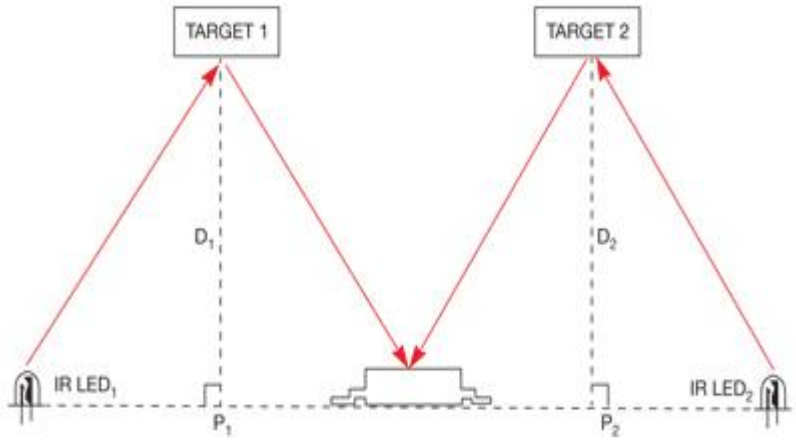
Σχήμα 8. Wi-Fi

1.3.1.5. Bluetooth

Το Bluetooth είναι ασύρματη μέθοδος επικοινωνίας που χρησιμοποιείται από δύο συσκευές για κοντινές αποστάσεις. Το Bluetooth υιοθετήθηκε από την IEEE ως πρότυπο 802.15 για WPAN και είναι παρόμοιο με το Wi-Fi. Η μέγιστη απόσταση για την επικοινωνία Bluetooth είναι μέχρι 100 μέτρα για την κατηγορία 1. Οι συσκευές μπορούν να στείλουν δεδομένα με ρυθμό 3Mb/s και η εφαρμογή τους είναι ιδιαίτερα ακριβή.

1.3.1.6. Υπέρυθρες (Infrared)

Η υπέρυθρη ασύρματη δικτύωση **(**Infrared**)** ήταν μια πρωτοπόρα τεχνολογία στον τομέα του εσωτερικού προσδιορισμού θέσης. Ωστόσο, αυτό το σύστημα έχει αντιμετωπίσει διάφορα ουσιώδη προβλήματα. Η αρχική πρόκληση είναι το περιορισμένο εύρος ενός δικτύου IR. Επίσης, η υπέρυθρη δεν έχει καμία μέθοδο για την παροχή υπηρεσιών δεδομένων δικτύωσης. Μια πρόωρη εφαρμογή μιας τεχνικής IR είναι το ενεργό σύστημα διακριτικών (Active Badge System). Αυτό είναι ένα μακρινό σύστημα εντοπισμού στο οποίο η θέση ενός ατόμου καθορίζεται από το μοναδικό σήμα IR που εκπέμπεται κάθε δέκα δευτερόλεπτα. Τα σήματα λαμβάνονται από τους αισθητήρες που είναι τοποθετημένοι σε διάφορες θέσεις μέσα σε ένα κτήριο και τα μεταδίδουν σε ένα κεντρικό σύστημα διαχείρισης θέσης. Η ακρίβεια που επιτυγχάνεται από αυτό το σύστημα είναι αρκετά υψηλή στα εσωτερικά περιβάλλοντα. Όμως, το σύστημα πάσχει από διάφορους περιορισμούς όπως το κόστος εγκαταστάσεων αισθητήρων λόγω του περιορισμένου εύρους του IR, το κόστος συντήρησης, και η ευαισθησία του δέκτη στο φως του ήλιου, το οποίο εμφανίζεται συχνά στα δωμάτια με τα παράθυρα.



Σχήμα 9. Η μέθοδος των υπερύθρων

1.3.1.7. Ultra Wide Band

Για τα εξαιρετικά-ευρείας ζώνης σήματα (UWB) που χρησιμοποιούνται για τον προσδιορισμό θέσης, το ενδιαφέρον είναι συνεχώς αυξανόμενο, γεγονός που οφείλεται στην ικανότητά τους να παρέχουν με ακρίβεια εκατοστών πληροφορίες προσδιορισμού θέσης. Τα πλεονεκτήματα του UWB περιλαμβάνουν τη χαμηλή πυκνότητα ισχύος και το μεγαλύτερο εύρος ζώνης, το οποίο αυξάνει την αξιοπιστία. Η χρήση μιας ευρείας συχνότητας στοιχείων αυξάνει την πιθανότητα ότι ένα σήμα θα πλησιάσει ένα εμπόδιο, που προσφέρει υψηλότερη ανάλυση. Επίσης, το σύστημα υπόκειται στη λιγότερη παρέμβαση από άλλες ραδιοσυχνότητες που είναι σε χρήση στην περιοχή.

Η φύση του σήματος UWB επιτρέπει στην προσέγγιση χρονικής καθυστέρησης να παρέχει υψηλότερη ακρίβεια από την ισχύ σήματος ή τις κατευθυντικές προσεγγίσεις επειδή η ακρίβεια της χρονικής καθυστέρησης του προσδιορισμού θέσης είναι αντιστρόφως ανάλογη προς το αποτελεσματικό εύρος ζώνης των σημάτων. Αυτό παρουσιάζεται στους τύπους που δίνονται κατωτέρω. Η ακρίβεια της μέτρησης της δύναμης σημάτων είναι βασισμένη σε Cramér-Rao Lower Bound (CRLB) για τις εκτιμήσεις απόστασης ως εξής:

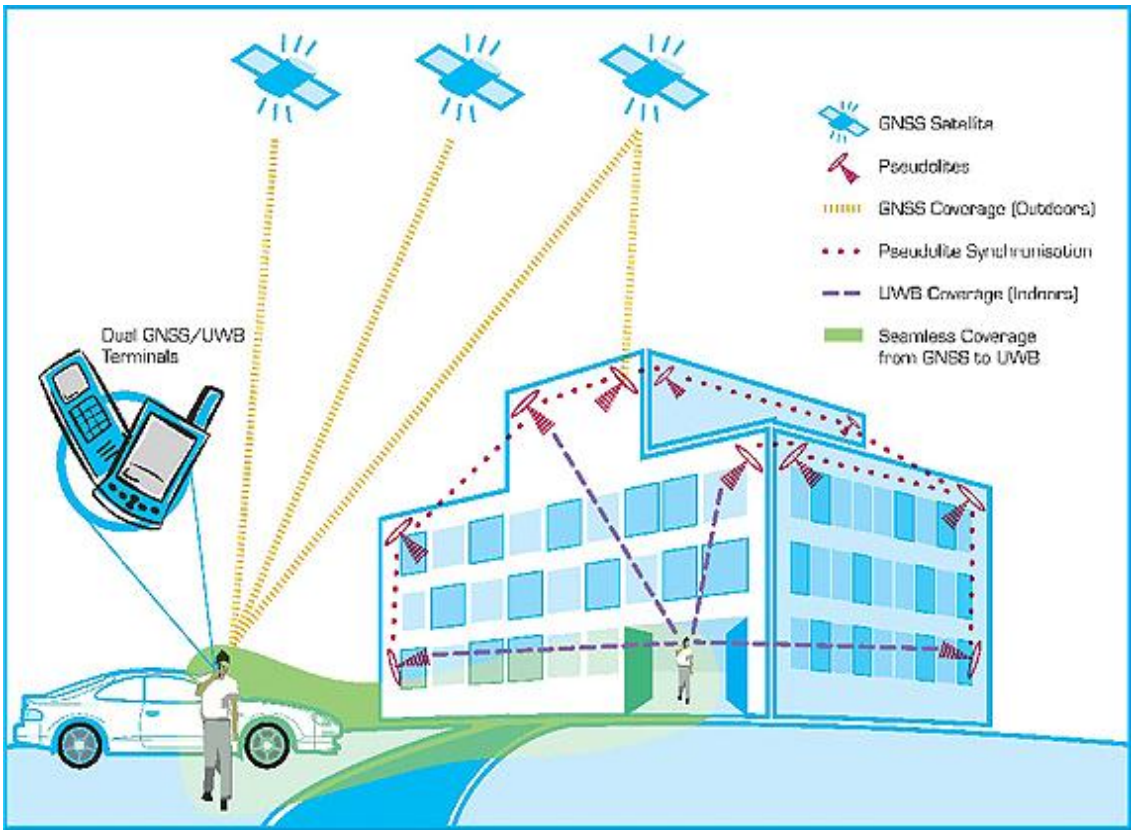
$$\sqrt{Var(\hat{d})} \geq \frac{\ln 10}{10} \frac{\sigma_{sh}}{n_p} d.$$

Στον τύπο, το $\sqrt{Var(\hat{d})}$ είναι η ακρίβεια της ισχύς σήματος, το d είναι η απόσταση μεταξύ των δύο κόμβων, το n_p είναι ο παράγοντας απώλειας διαδρομής (path loss), το σ_{sh} είναι η τυπική απόκλιση της τυχαίας μεταβλητής τύπου Gauss με μηδενική μέση τιμή που ακολουθεί λογαριθμοκανονική κατανομή και αντιπροσωπεύει την επίδραση του φαινομένου της σκίασης στο δίαυλο.

Ο τύπος για την ακρίβεια των μετρήσεων χρονικής καθυστέρησης μιας μονής πορείας-διαδρομής με πρόσθετο λευκό γκαουσσιανό θόρυβο (Additive White Gaussian Noise - AWGN) δείχνει ότι η ακρίβεια εξαρτάται άμεσα από το αποτελεσματικό εύρος ζώνης σήματος β του μεταδιδόμενου σήματος UWB, δηλαδή:

$$\sqrt{Var(\hat{d})} \geq \frac{c}{2\sqrt{2\pi}\sqrt{SNR}\beta}$$

Στον τύπο, το $\sqrt{Var(\hat{d})}$ είναι η ακρίβεια ισχύς σήματος, το c είναι η ταχύτητα του φωτός, το SNR είναι σηματοθορυβικός λόγος και το β είναι το αποτελεσματικό (ή μέση τετραγωνική ρίζα) εύρος ζώνης σήματος. Το οικονομικό κόστος της εφαρμογής περιλαμβάνει ένα ικανοποιητικό δίκτυο σταθμών UWB προκειμένου να εκτελεσθούν οι τεχνικές προσδιορισμού θέσης.



Σχήμα 10. Η Ultra Wide Band μέθοδος

1.3.1.8. Σύνοψη των Self-positioning τεχνολογιών

Στον αμέσως κατωτέρω εμφανιζόμενο πίνακα συνοψίζονται τα θετικά και αρνητικά της κάθε τεχνολογίας :

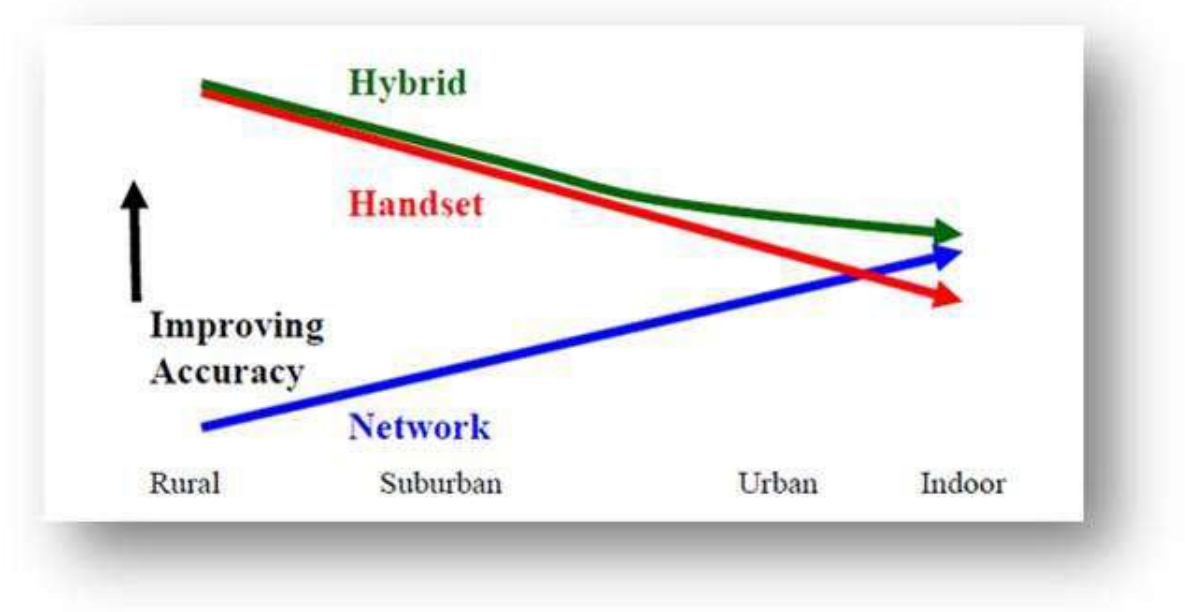
Self-positioning	Θετικά	Αρνητικά
GPS	• Μέτρια υπαίθρια ακρίβεια • Μεγάλη διαθεσιμότητα	• Ελάχιστη ακρίβεια σε εσωτερικό χώρο
A-GPS	• Μέτρια υπαίθρια ακρίβεια	• Ελάχιστη εσωτερική ακρίβεια
Pseudolite GPS	• Υψηλή εσωτερική και υπαίθρια ακρίβεια	• Πολύ ακριβός εξοπλισμός

RFID	<u>Passive</u> ♦ Δε χρειάζεται μπαταρία	<u>Active</u> ♦ Μεγάλο εύρος επικοινωνίας (100m ή περισσότερο) ♦ Μεγάλος αποθηκευτικός χώρος	<u>Passive</u> ♦ Χαμηλό ή πολύ χαμηλό εύρος επικοινωνίας (3m ή και λιγότερο) ♦ Μικρός αποθηκευτικός χώρος	<u>Active</u> ♦ Απαιτείται μπαταρία ♦ Συνεχής διαθεσιμότητα της ισχύος του αναμεταδότη
Cell Tower	• Μεγάλο εύρος		• Ιδιαίτερα ανακριβής σε εσωτερικό και υπαίθριο χώρο	
Wi-Fi	• Εύκολα διαθέσιμος στα περισσότερα κτήρια • Ελάχιστο κόστος εφαρμογής • Μέτριο εύρος		• Η ισχύς των δικτύων μπορεί αν ποικίλει λόγω των πολλαπλών διαδρομών διάδοσης	
Bluetooth	• Χαμηλή ισχύ • Μικρό οικονομικό κόστος		• Χαμηλό εύρος • Μεγάλο κόστος εφαρμογής	
Infrared	• Μεγάλη ακρίβεια		• Μεγάλο κόστος εφαρμογής • Το φως του ήλιου μπορεί να έχει επιπτώσεις • Χαμηλό εύρος	
UWB	• Μεγάλη ακρίβεια • Χαμηλή πυκνότητα ισχύος • Ευρύ εύρος ζώνης		• Μεγάλο κόστος εφαρμογής • Δε χρησιμοποιείται συχνά	

Πίνακας 3. Θετικά και αρνητικά των Self-positioning τεχνολογιών

1.3.2. Μέθοδοι Remote-positioning

Τα αποτελεσματικότερα συστήματα θέσης είναι εκείνα που βασίζονται στις ράδιο τεχνικές θέσεις (radio location techniques) τα οποία εκμεταλλεύονται τις μετρήσεις των φυσικών ποσοτήτων σχετικά με τα ράδιο σήματα που ταξιδεύουν μεταξύ του κινητού τερματικού και ενός συνόλου πομποδεκτών των οποίων η θέση είναι γνωστή, π.χ. σταθμός βάσης (BSs), δορυφόροι πλοήγησης. Οι μετρήσεις των ράδιο σημάτων χαρακτηριστικά είναι η λαμβανόμενη ισχύς σήματος (RSS), η Cell of Origin, η γωνία άφιξης (Angle of Arrival-AOA), ο χρόνος άφιξης (Time of Arrival-TOA), και η χρονική διαφορά άφιξης (Time Difference of Arrival-TDOA). Πιο πρόσφατα, οι ράδιο αλγόριθμοι θέσης βασίζονταν σε έναν συνδυασμό των ανωτέρω τεχνικών. Τέτοιες υβριδικές τεχνικές εγγυώνται μια υψηλή ακρίβεια θέσης, οι οποίες χρησιμοποιούνται όλο και περισσότερο για εφαρμογές ασφαλείας και για παιχνίδια κινητών τηλεφώνων. Μάλιστα, οι υβριδικές τεχνολογίες όπως παρουσιάζεται και στο Σχήμα 11 αποτελούν την ιδανικότερη περίπτωση για εντοπισμό θέσης σε εσωτερικό χώρο.



Σχήμα 11. Ακρίβεια μεθόδων εντοπισμού κινητού τερματικού

1.3.2.1. Μέθοδος αναγνώρισης ταυτότητας κυψέλης (COO)

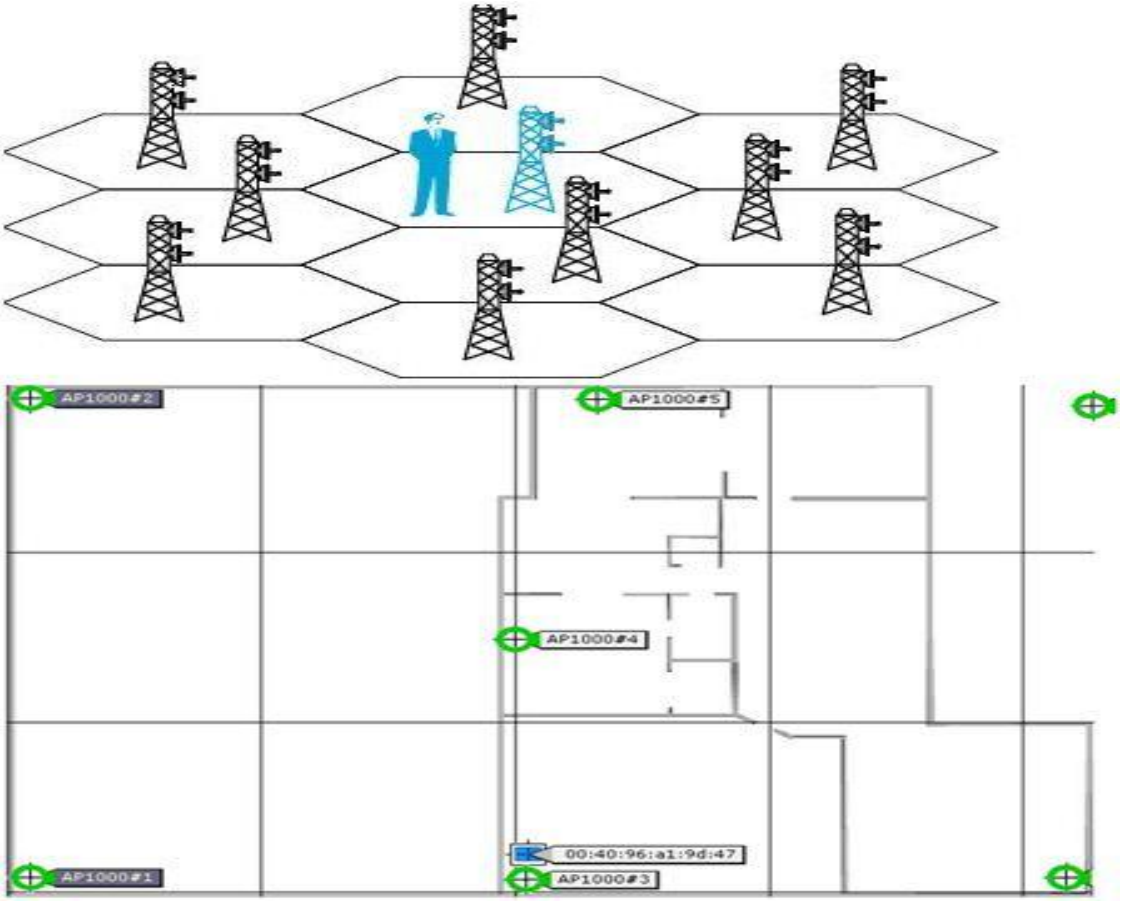
Η Cell-ID μέθοδος αποτελεί βασική τεχνική παροχής υπηρεσιών και εφαρμογών θέσης. Η μέθοδος στηρίζεται στο γεγονός ότι τα κινητά δίκτυα μπορούν να προσδιορίσουν κατά προσέγγιση θέση ενός κινητού τηλεφώνου με τη γνώση ποια κυψέλη η συσκευή χρησιμοποιεί σε μία δεδομένη στιγμή. Η πληροφορία που χρησιμοποιείται είναι η παγκόσμια ταυτότητα της κυψέλης (CGI-cell global identity), η οποία είναι μοναδική παγκοσμίως και χαρακτηρίζει την κυψέλη. Στη συνέχεια παρατίθενται περιληπτικά τα κυριότερα χαρακτηριστικά της γνωρίσματα.

- Καθορίζεται η κυψέλη που εξυπηρετεί το κινητό τερματικό και χρησιμοποιούνται οι γεωγραφικές συντεταγμένες του σταθμούς βάσης ή του κέντρου της περιοχής κάλυψης της συγκεκριμένης κυψέλης ως μια εκτίμηση της θέσης του κινητού.

- Οι πληροφορίες για την αντιστοίχιση του CGI με το γεωγραφικό πλάτος και μήκος του σταθμού βάσης βρίσκονται αποθηκευμένες στη βάση δεδομένων του δικτύου

- Το σημαντικότερο όφελος της τεχνολογίας είναι ότι είναι ήδη σε χρήση σήμερα και μπορεί να υποστηριχθεί από όλα τα κινητά τηλέφωνα.

- Η ακρίβεια της μεθόδου είναι αρκετά χαμηλή και εξαρτάται από την ακτίνα της κυψέλης, η οποία κυμαίνεται από 50m περίπου για εσωτερικούς χώρους έως 30km για αγροτικές περιοχές, οπότε ένα σφάλμα στο προσδιορισμό της θέσης μετά από μερικά χιλιόμετρα (km) δεν θα είναι αποδεκτό. Γενικά, η ακρίβεια είναι υψηλότερη στις πυκνοκατοικημένες περιοχές (αστικές) και χαμηλώνει πολύ στα αγροτικά περιβάλλοντα. Η ακρίβεια μπορεί να βελτιστοποιηθεί με τη χρήση κατευθυντικών κεραιών, με την αύξηση της πυκνότητας των σταθμών βάσης, με την αλλαγή της γεωμετρίας των κυψελών καθώς και τη μείωση του μεγέθους τους (microcells, picocells).

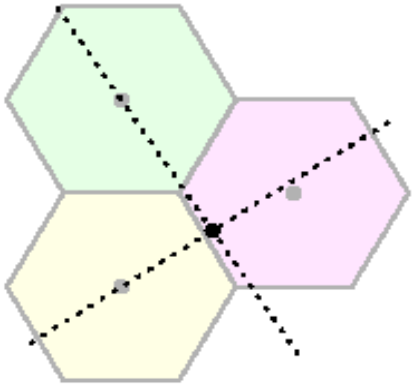


Σχήμα 12. Cell of Origin

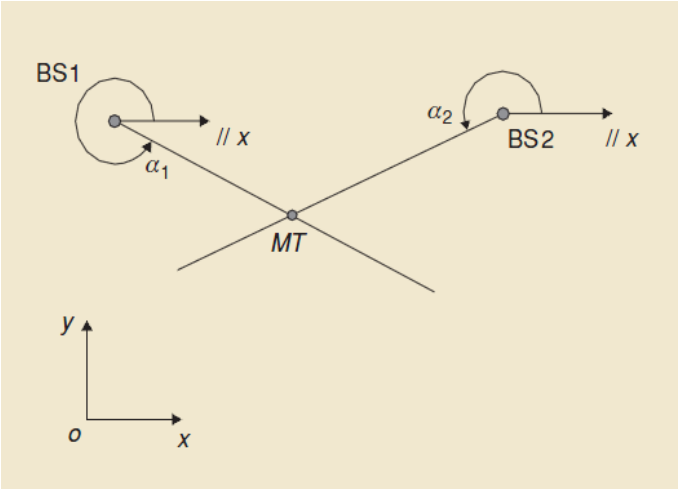
.

1.3.2.2. Μέθοδος της γωνίας άφιξης (Angle of Arrival-AOA)

Η γωνία άφιξης (AOA) είναι μέθοδος που χρησιμοποιεί τους πολλαπλούς σταθμούς βάσεων για να προσεγγίσει τη θέση του χρήστη. Σε αυτή τη τεχνική χρειάζονται τουλάχιστον δύο σταθμοί βάσης γνωστών θέσεων και προσανατολισμού οι οποίοι πρέπει να καθορίσουν τη γωνία με την οποία το σήμα έφθασε στο χρήστη. Η γωνία καθορίζεται με την οδήγηση μιας κατευθυντικής κεραίας μέχρι τη μέγιστη ισχύ σήματος ή όταν ανιχνευθεί συμφωνία φάσης. Κάθε μέτρηση παράγει μια ευθεία γραμμή έτσι αν το κινητό τηλέφωνο δεν είναι πάνω στην ευθεία γραμμή των δύο σταθμών βάσης τότε η θέση καθορίζεται από τη διατομή των δύο ευθειών στη θέση που βρίσκεται το κινητό τηλέφωνο (Σχήμα 13 και 14). Εάν ο χρήστης και οι σταθμοί βάσεων δεν είναι συνεπίπεδοι τότε απαιτούνται τρισδιάστατες κατευθυντικές κεραίες. Η χρήση περισσότερων σταθμών βάσεων από αυτό που απαιτείται, μπορεί να βελτιώσει κατά πολλή την ακρίβεια (Σχήμα 15). Η ακρίβεια του όλου συστήματος εξαρτάται από τη διάδοση σημάτων, την ακρίβεια των κατευθυντικών κεραιών που χρησιμοποιούνται και την απόσταση της συσκευής από τις κεραίες.



Σχήμα 13. Angle of Arrival



Σχήμα 14. Η AOA μέθοδος

Σχήμα 15. Η AOA μέθοδος

1.3.2.3. Χρονική καθυστέρηση

Δεδομένου ότι τα ηλεκτρομαγνητικά κύματα ταξιδεύουν με μια σταθερή ταχύτητα (ταχύτητα του φωτός) στο κενό, η απόσταση μεταξύ δύο σημείων μπορεί να υπολογιστεί εύκολα με τη μέτρηση του χρόνου καθυστέρησης ενός ράδιο κύματος που διαδόθηκε μεταξύ τους. Αυτή η μέθοδος είναι κατάλληλη για δορυφορικά συστήματα και χρησιμοποιείται παγκοσμίως από αυτά. Υπάρχουν δύο τύποι μεθόδων χρονικής καθυστέρησης: Απόλυτος συγχρονισμός (Absolute Timing) ή χρόνος άφιξης (Time of Arrival - TOA) και διαφορικός χρόνος άφιξης (Time Difference of Arrival - TDOA) ή υπερβολική τεχνική (Hyperbolic Technique).

1.3.2.4. Μέθοδος του χρόνου άφιξης (Time of Arrival-TOA)

Με τη μέθοδο αυτή χρησιμοποιείται μια μορφή triangulation προκειμένου να προσδιορισθεί η θέση του χρήστη. Περαιτέρω, για να εκτελεστεί το triangulation, θα πρέπει πρώτα να έχουν προκαθορισθεί οι θέσεις των τριών σταθμών βάσης (Σχήμα 16). Ειδικότερα:

- Σε ένα σύγχρονο σύστημα, οι πληροφορίες προσδιορισμού θέσης προέρχονται από τον απόλυτο χρόνο ενός κύματος που χρειάζεται για να ταξιδέψει μεταξύ ενός πομπού και ενός δέκτη ή και αντίστροφα. Αυτό υπονοεί ότι ο δέκτης ξέρει τον ακριβή χρόνο της μετάδοσης.

- Σε ένα ασύγχρονο σύστημα, αυτή η προσέγγιση χρειάζεται τη μέτρηση του round-trip χρόνου ενός σήματος μεταδιδόμενου από μια πηγή σε έναν προορισμό και στη συνέχεια επιστρέφει πίσω στην πηγή με αποτέλεσμα να δίνει τη μονόδρομη μέτρηση εις διπλούν.

➢ Η ακρίβεια της τεχνικής εξαρτάται από τα λάθη καθυστέρησης διάδοσης (propagation delay) και από την ακρίβεια των μετρήσεων χρόνου.

Συνεπώς, η απόσταση από τον κινητό τερματικό προς μονάδα μέτρησης είναι άμεσα ανάλογη προς το χρόνο διάδοσης. Για να υπάρξει δισδιάστατο προσδιορισμός θέσης, οι TOA μετρήσεις, όσον αφορά τα σήματα, πρέπει να γίνονται από τουλάχιστον τρία σημεία αναφοράς, όπως φαίνεται στο Σχήμα 16. Για τα συστήματα βάσης, ο μονόδρομος χρόνος διάδοσης μετριέται, όπως και η απόσταση μεταξύ της μονάδας μέτρησης και του πομπού. Γενικά, χρησιμοποιώντας τη μέθοδο TOA εμφανίζονται δύο προβλήματα στα αποτελέσματα. Κατ' αρχάς, όλοι οι πομποί και δέκτες στο σύστημα πρέπει να συγχρονιστούν ακριβώς. Δεύτερον, το timestamp πρέπει να περνάει μαζί με μεταδιδόμενο σήμα έτσι ώστε η μονάδα μέτρησης να διακρίνει την απόσταση που το σήμα έχει ταξιδέψει. Η TOA μπορεί να μετρηθεί χρησιμοποιώντας διαφορετικές τεχνικές σήματος όπως direct sequence spread-spectrum (DSSS) ή ultra wide band (UWB) μετρήσεις.

Χρησιμοποιείται μια απλή προσέγγιση της γεωμετρικής μέθοδο για τον υπολογισμό των σημείων διατομής των κύκλων TOA. Η θέση από το στόχο μπορεί επίσης να υπολογιστεί με τη μέθοδο ελαχίστων τετραγώνων μιας μη γραμμικής συνάρτησης κόστους, Αν υποτεθεί ότι το κινητό τερματικό βρίσκεται στη θέση (X_0, Y_0), μεταδίδει ένα σήμα στο χρόνο t_0, οι N σταθμοί βάσης τοποθετημένοι στις θέσεις (X_1, Y_1), (X_2, Y_2) , . . . , (X_N, Y_N) λαμβάνουν το

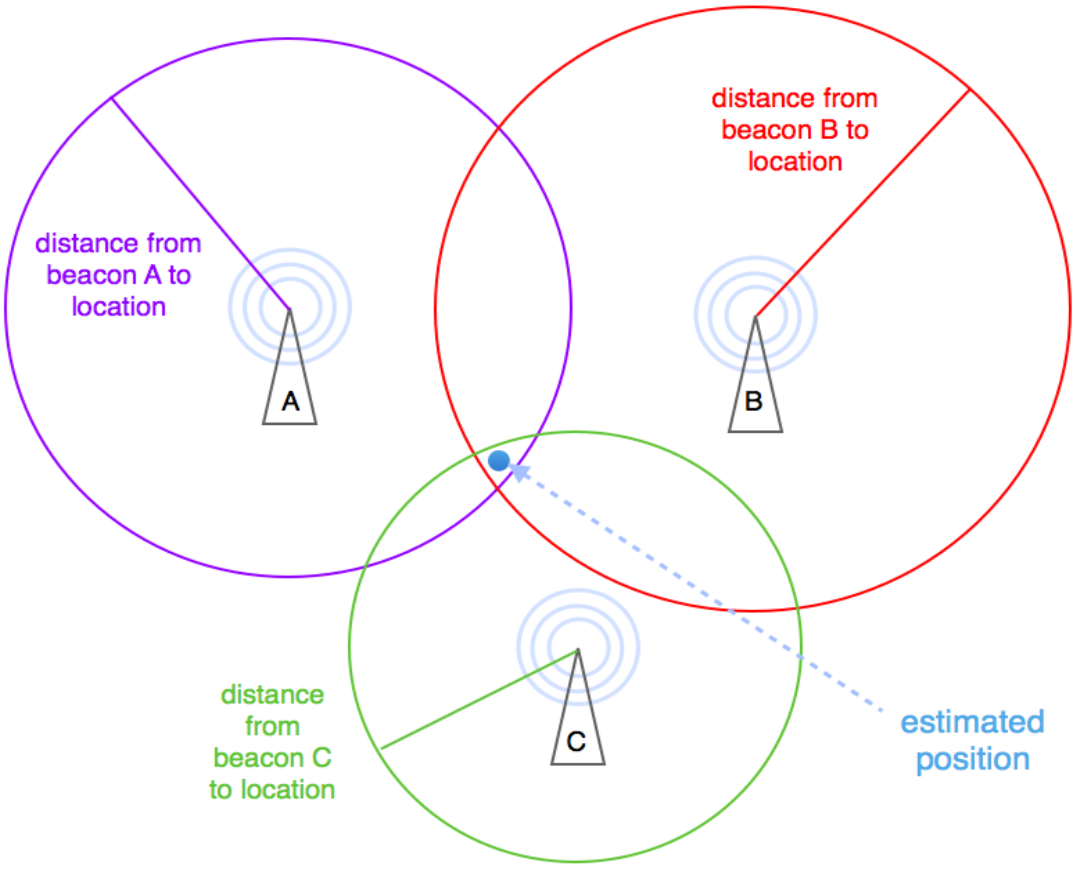
σήμα στο χρόνο t_1 , t_2 , . . . , t_N. Σαν μέτρο απόδοσης, η συνάρτηση κόστους μπορεί να διαμορφωθεί ως εξής:

$$F(x) = \sum_{i=1}^{N} a_i^2 f_i^2(x)$$

Όπου α_i μπορεί να επιλεχτεί για να απεικονίσει την αξιοπιστία του σήματος λαμβάνοντας τη μονάδα μέτρησης i, και $f_i(x)$ δίνεται ως εξής:

$$f_i(x) = C(t_i - t) - \sqrt{(X_i - X)^2 - (Y_i - Y)^2}$$

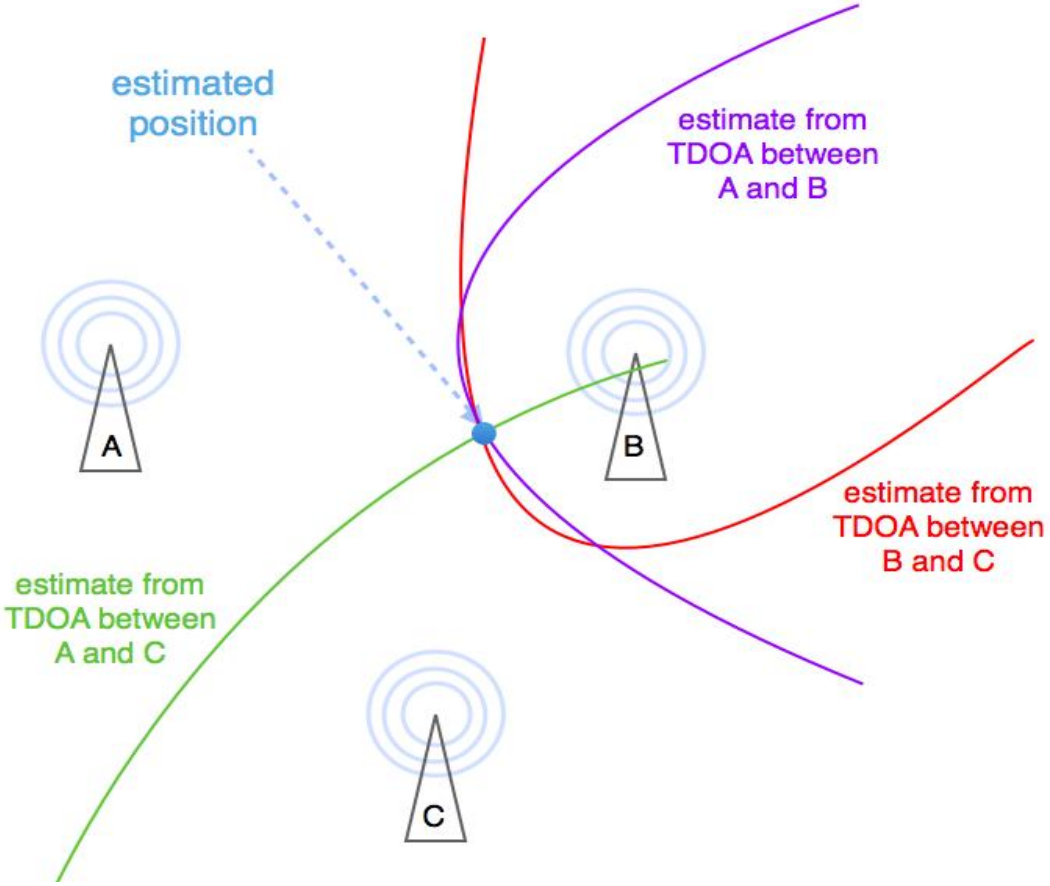
όπου c είναι η ταχύτητα του φωτός, και x = (x, y, t). Αυτή η συνάρτηση είναι διαμορφωμένη για κάθε μονάδα μέτρησης, i = 1, . . . , N, και $f_i(x)$ θα μπορούσε να γίνεται μηδέν με την κατάλληλη επιλογή των x, y, και t. Η εκτιμώμενη θέση καθορίζεται με την ελαχιστοποίηση της συνάρτησης F(x). Υπάρχουν άλλοι αλγόριθμοι βασισμένοι στην τεχνική TOA για συστήματα προσδιορισμού εσωτερικής θέσης όπως ο κοντινότερος γείτονας (Closest-Neighbor - CN) και υπόλοιπη στάθμιση (residual weighting - RWGH). Ο αλγόριθμος CN υπολογίζει τη θέση του χρήστη ως θέση σταθμού βάσης ή σημείο αναφοράς το οποίο βρίσκεται πιο κοντά σε εκείνο τον χρήστη. Ο αλγόριθμος RWGH μπορεί βασικά να παρατηρηθεί ως μορφή weighted least-squares αλγορίθμου. Είναι κατάλληλο για περιπτώσεις στις οποίες εμφανίζεται LOS, Non-LOS και μικτό LOS/NLOS.



Σχήμα 16. Time of Arrival

1.3.2.5. Μέθοδος της διαφοράς του χρόνου άφιξης (Time Difference of Arrival-TDOA)

Ένα κινητό τηλέφωνο μπορεί να «ακούσει» τα σήματα που μεταδίδονται ταυτόχρονα από διάφορους σταθμούς βάσης και να μετρήσει τη χρονική διαφορά μεταξύ κάθε ζευγαριού αφίξεων. Εάν σε ένα δισδιάστατο σύστημα μια γραμμή ζωγραφίζεται ενώνοντας όλα τα σημεία που έχουν την ίδια χρονική διαφορά, μια υπερβολή θα σχεδιαστεί (Σχήμα 17). Η διατομή των υπερβολικών γεωμετρικών τόπων θα καθορίσει τη θέση του κινητού συστήματος (Σχήμα 18).



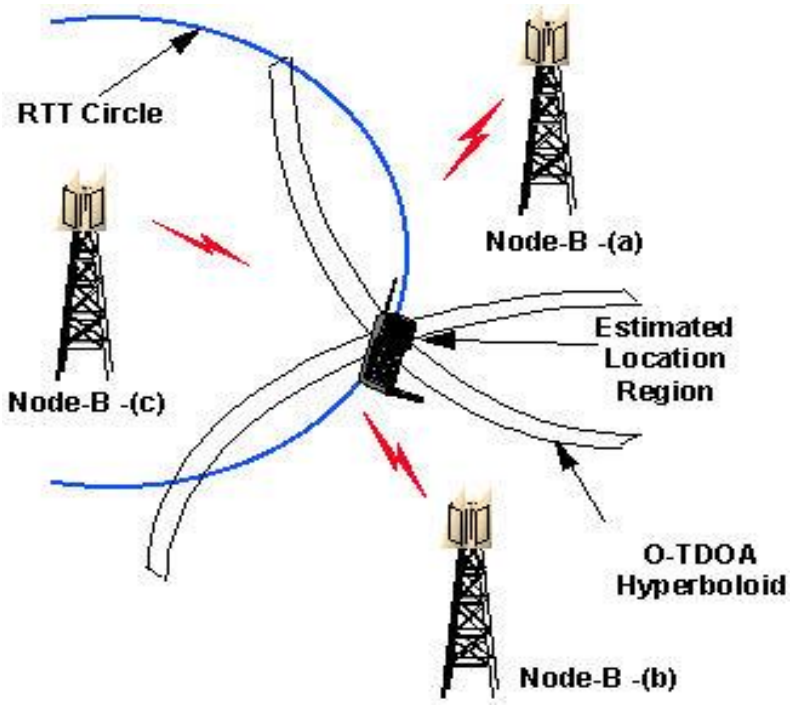
Σχήμα 17. Η μέθοδος Time Difference of Arrival

Με τη μέθοδο TDOA, κάθε μέτρηση καθορίζεται με μια καμπύλη στην οποία το κινητό τηλέφωνο θα πρέπει να βρεθεί. Έστω (X_A, Y_A) και (X_B, Y_B) οι γνωστές θέσεις των σταθμών βάσεων Α και Β και έστω Δt_{AB} το TDOA. Η θέση (X_C, Y_C) μπορεί να καθοριστεί από :

$$\sqrt{(X_A - X_C)^2 - (Y_A - Y_C)^2} - C\Delta t_{AB} = \sqrt{(X_B - X_C)^2 - (Y_B - Y_C)^2}$$

Δύο ή τρεις μετρήσεις TDOA απαιτούνται για το σαφή προσδιορισμό θέσης. Ένα σημαντικό ζήτημα για τα συστήματα TDOA είναι η ανάγκη να υπάρξουν μερικά μέσα συγχρονισμού των σταθμών βάσης. Το σήμα που μεταδίδεται από το κινητό τηλέφωνο,

παραλαμβάνεται από διάφορους σταθμούς βάσεων και πρέπει να υπάρξει μια γνωστή χρονική σχέση μεταξύ των ρολογιών των δεκτών.



Σχήμα 18. Η TDOA τεχνική

1.3.2.6. Ισχύς λαμβανόμενου σήματος (RSS)

Οι προαναφερθείσες μέθοδοι παρουσιάζουν ορισμένα μειονεκτήματα. Για τα εσωτερικά περιβάλλοντα, είναι δύσκολο να βρεθεί ένα κανάλι με οπτική επαφή (LOS) μεταξύ του πομπού και του δέκτη. Η ράδιο διάδοση σε τέτοια περιβάλλοντα θα έπασχε από την επίδραση πολλαπλών διαδρομών. Ο χρόνος και η γωνία ενός σήματος άφιξης θα επηρεαζόταν από την επίδραση των πολλαπλών διαδρομών, και κατά συνέπεια η ακρίβεια της εκτιμώμενης θέσης θα μπορούσε να μειωθεί. Μία μέθοδος που θα μπορούσε να εκτιμήσει τη θέση ενός αντικειμένου, χωρίς να εξαρτάται τόσο άμεσα από μία πιθανή έλλειψη οπτικής επαφής, είναι να χρησιμοποιηθούν μετρήσεις που δείχνουν την ισχύ του λαμβανόμενου σήματος ή την εξασθένηση του σήματος (Received Signal Strength: RSS ή Signal Attenuation) που εκπέμπει ο πομπός στους δέκτες.

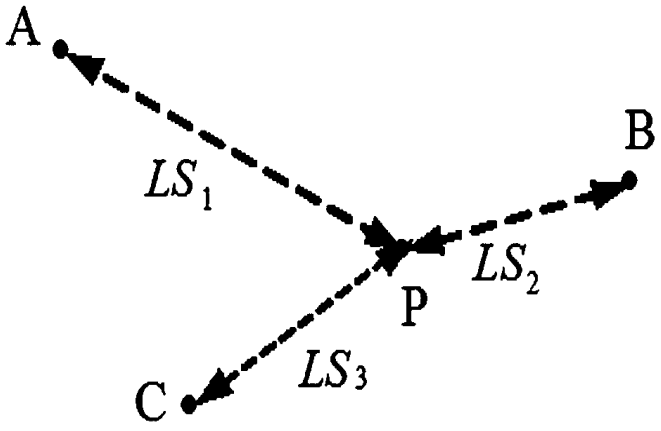
Έστω ότι ο πομπός εκπέμπει ένα σήμα, και ένας σταθμός βάσης λαμβάνει το σήμα αυτό εξασθενημένο κατά LS. Αυτό που χρειάζεται είναι να βρεθεί ένας τρόπος για να συσχετίσει την εξασθένιση του σήματος με την απόσταση ανάμεσα στον πομπό και τον σταθμό βάσης. Η συσχέτιση αυτή μεταξύ απόστασης και RSS επιτυγχάνεται αν μοντελοποιηθεί η διάδοση του σήματος στον εσωτερικό χώρο που βρίσκεται ο χρήστης. Θα μπορούσε να χρησιμοποιηθεί ένα εμπειρικό, ένα στατιστικό, ή ένα αναλυτικό μοντέλο, για τον εσωτερικό χώρο. Αν καταφερθεί και μοντελοποιηθεί η διάδοση του σήματος στο χώρο, τότε υπάρχουν διαθέσιμες συναρτήσεις, οι οποίες δείχνουν με ποιο τρόπο εξασθενεί το σήμα όταν διαδίδεται, εξαιτίας των ιδιαίτερων συνθηκών που επικρατούν στον συγκεκριμένο χώρο, και οι οποίες επηρεάζουν τη διάδοση του σήματος. Συνεπώς, αν τοποθετηθούν σε

καθορισμένες θέσεις A: (X_1, Y_1) , B: (X_2, Y_2) και C: (X_3, Y_3) στον εσωτερικό χώρο των σταθμών βάσης, όπως παρουσιάζεται στο Σχήμα 19, τότε είναι δυνατόν να ληφθούν οι μετρήσεις LS_1, LS_2 και LS_3 αντίστοιχα, οι οποίες δείχνουν την εξασθένιση του σήματος σε κάθε σταθμό βάσης. Επομένως, αν χρησιμοποιηθεί το μοντέλο διάδοσης το οποίο είναι διαθέσιμο για τον συγκεκριμένο χώρο, μπορούν να υπολογιστούν οι αποστάσεις R_1, R_2 και R_3 ανάμεσα στον πομπό και τους σταθμούς βάσης A, B και C αντίστοιχα. Με αυτό τον τρόπο, όπως και στον αλγόριθμο που βασίζεται σε TOA μετρήσεις, προκύπτει το εξής μη γραμμικό σύστημα,

$$R_1^2 = (X_1 - X)^2 + (Y_1 - Y)^2$$
$$R_2^2 = (X_1 - X)^2 + (Y_1 - Y)^2$$
$$R_3^2 = (X_1 - X)^2 + (Y_1 - Y)^2$$

η λύση του οποίου, και άρα η θέση του αντικειμένου που επιθυμείται να εντοπιστεί, είναι η τομή τριών κύκλων.

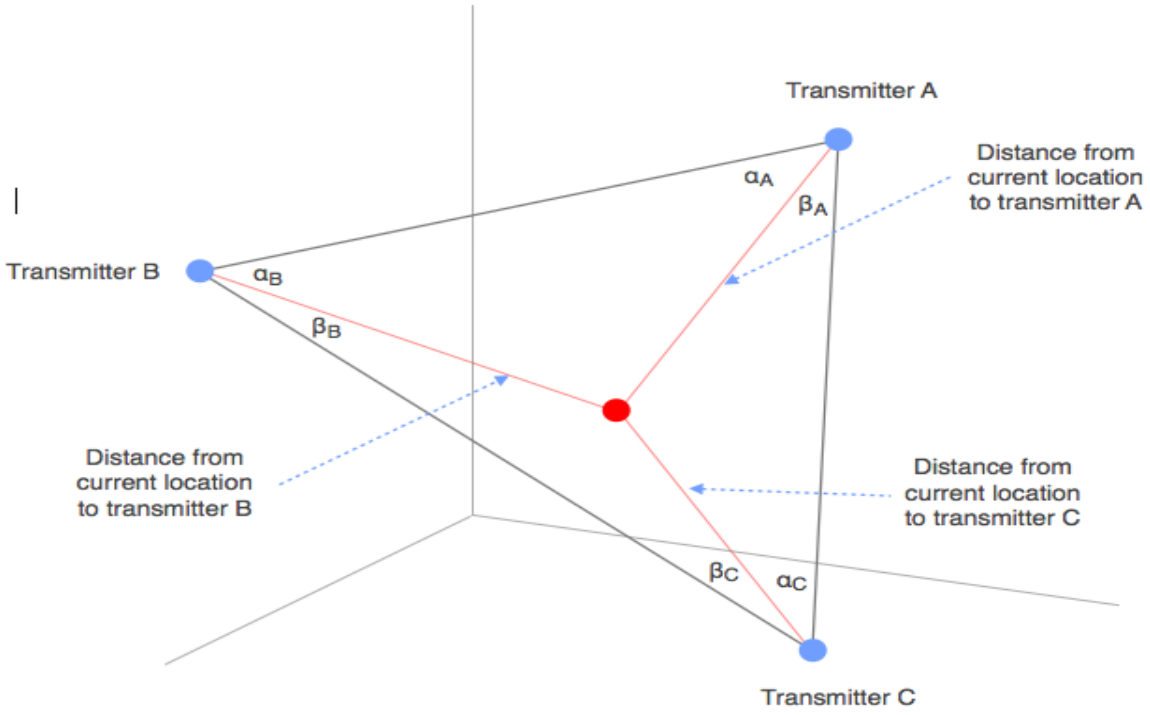
Για να έχει το σύστημα μοναδική λύση, πρέπει να υπάρχουν τουλάχιστον τρεις σταθμοί βάσης. Ομοίως, αν χρειαζόταν να εντοπιστεί ένα αντικείμενο στον τρισδιάστατο (3-D) χώρο, θα έπρεπε να υπάρχουν τουλάχιστον τέσσερις σταθμοί βάσης, για να δώσει μοναδική λύση ένα μη γραμμικό σύστημα τομής σφαιρών. Όπως έχει προαναφερθεί, είναι πολύ δύσκολο να μοντελοποιηθεί με επιτυχία η διάδοση σε έναν εσωτερικό χώρο. Αυτό, γιατί οι συνθήκες σκίασης και πολλαπλών διαδρομών προκαλούν έντονες και αρκετά απρόβλεπτες διαλείψεις, και επομένως υπάρχει μεγάλη πιθανότητα οι τιμές του μοντέλου να διαφέρουν αρκετά από τις πραγματικές τιμές. Κατόπιν τούτου, πολλές φορές, οι αλγόριθμοι που χρησιμοποιούν RSS μετρήσεις οδηγούν σε μεγάλα σφάλματα.



Σχήμα 19. Η RSS μέθοδος

1.3.2.7. Τριγωνισμός (Triangulation)

Σε ένα περιβάλλον με τις γνωστές απώλειες διάδοσης, η ισχύς σημάτων μπορεί να μετατραπεί άμεσα σε απόσταση. Στο κενό, η ισχύς σημάτων ποικίλει με το αντίστροφο του τετραγώνου της απόστασης από το πομπό στο δέκτη. Για να μετατραπεί ακριβώς η απόσταση σε μια πραγματική ρύθμιση, θα πρέπει να υπολογιστούν οι παράγοντες όπως τα κέρδη κεραιών και η παρέμβαση από αντικείμενα που μπορούν να εμφανιστούν στην πορεία των σημάτων. Η ακρίβεια αυτής της μεθόδου εξαρτάται από την ακρίβεια με την οποία οι απώλειες διάδοσης μπορούν να υπολογιστούν. Είναι επίσης απλή στην εφαρμογή της.



Σχήμα 20. Η μέθοδος του Τριγωνισμού (Triangulation)

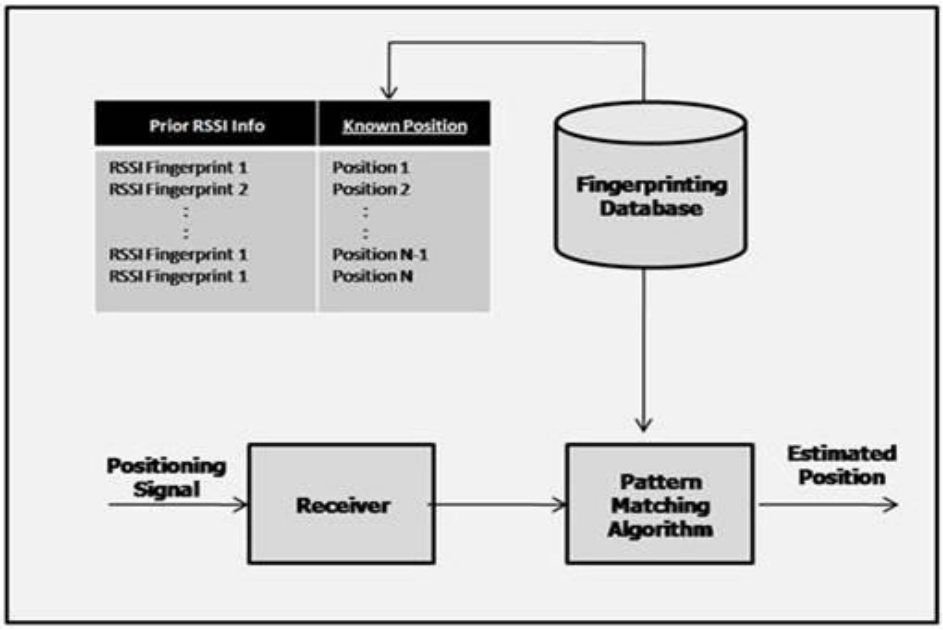
1.3.2.8. Fingerprinting

Η fingerprinting αποτελείται από δύο βήματα:

- το πρώτο βήμα αφορά τη συλλογή μετρήσεως σημάτων στα σημεία αναφοράς ελεύθερα αλλά καλά επιλεγμένα, οι οποίες μετρήσεις, κατόπιν, περνιούνται στην τοπική ή κεντρική βάση δεδομένων.

- το δεύτερο βήμα αναφέρεται στο ταίριασμα του μετρημένου σήματος του πιο πιθανού σημείου αναφοράς χρησιμοποιώντας έναν αλγόριθμο ταιριάσματος σχεδίων.

Συνεπώς, η fingerprinting είναι μια τεχνική προσδιορισμού θέσης που συγκρίνει τα μετρημένα στοιχεία RSSI με μια βάση δεδομένων των αναμενόμενων τιμών για να υπολογίσει την εκτιμώμενη θέση. Συγκεκριμένα, οι μετρήσεις λαμβάνονται σε ένα αυθαίρετο σχέδιο πλέγματος γύρω από το κτήριο. Ένας πολλαπλός αλγόριθμος συσχετισμού μητρών μπορεί να χρησιμοποιηθεί για να ψάξει αυτήν την βάση δεδομένων για την καλύτερη αντιστοιχία, δίνοντας κατά συνέπεια μια εκτίμηση θέσης. Αυτή η μέθοδος είναι ιδιαίτερα ακριβής αλλά χρειάζεται αρκετό χρόνο για να εφαρμοστεί.



Σχήμα 21. Η μέθοδος Fingerprinting

1.3.2.9. Σύνοψη των Remote-positioning μεθόδων

Στον παρακάτω πίνακα (Πίνακας 4) παρατίθενται τα θετικά και αρνητικά της κάθε τεχνικής :

Τεχνικές	Θετικά	Αρνητικά
Cell of Origin	❖ Ύπαρξη σταθμού βάσης ❖ Ο σταθμός βάσης μένει ακίνητος	❖ Μεγάλη ανακρίβεια
Angle of Arrival	❖ Σχετικά καλή ακρίβεια με τον κατάλληλο εξοπλισμό	❖ Απαιτούνται κατευθυντικές κεραίες ❖ Απαιτείται γνώση προσανατολισμού

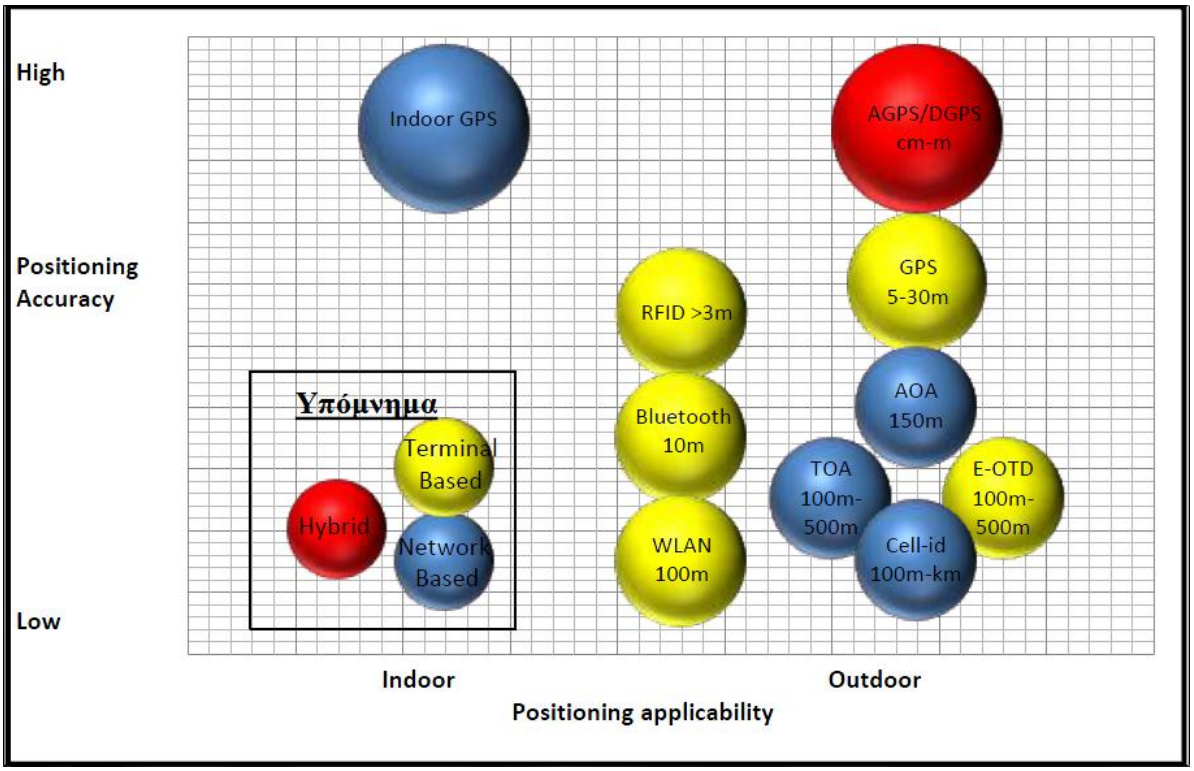
Time of Arrival	❖ Συγκριτικά καλή εσωτερική απόδοση	❖ Οι σταθμοί βάσης πρέπει να είναι συγχρονισμένοι ❖ Χαμηλή ακρίβεια
Time Difference of Arrival	❖ Συγκριτικά καλή εσωτερική απόδοση	❖ Χαμηλή ακρίβεια
RSS	❖ Καλή ακρίβεια	❖ Απαιτούνται πολλές μετρήσεις
Triangulation	❖ Πολύ εύκολη εφαρμογή	❖ Απαιτείται ο προσδιορισμός των γωνιών
Fingerprinting	❖ Μεγάλη ακρίβεια	❖ Απαιτείται μεγάλο χρονικό διάστημα βαθμονόμησης

Πίνακας 4. Θετικά και αρνητικά των Remote-positioning μεθόδων

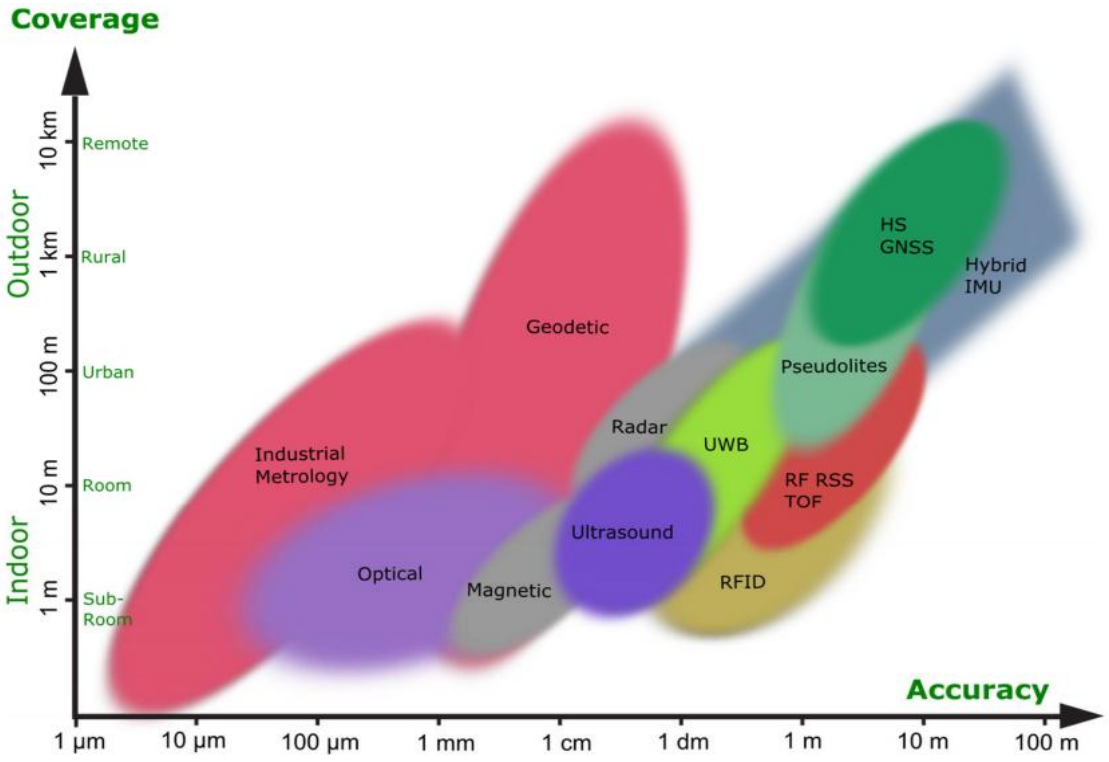
1.4. Σύνοψη όλων των μεθόδων και τεχνολογιών εντοπισμού θέσης

Μέθοδος	Indoor Accuracy	Συνθήκες LOS/NLOS	Επίδραση από το multipath	Κόστος
Proximity	Χαμηλή προς υψηλή	Και τα δύο	Όχι	Χαμηλό προς υψηλό
Direction (AOA)	Μέτρια	LOS	Ναι	Υψηλό
Time (TOA, TDOA)	Υψηλή	LOS	Ναι	Υψηλό
Propagation modeling	Μέτρια	LOS	Ναι	Μέτριο
Fingerprinting	Υψηλή	Και τα δύο	Όχι	Μέτριο

Πίνακας 5. Σύνοψη ορισμένων μεθόδων εντοπισμού



Σχήμα 22. Αντιστοίχιση χρήσης μεθόδων εντοπισμού θέσης σε Εσωτερικό και Εξωτερικό χώρο με κριτήριο την ακρίβεια.



Σχήμα 23. Αντιστοίχιση χρήσης τεχνολογιών εντοπισμού θέσης σε Εσωτερικό και Εξωτερικό χώρο με κριτήριο την ακρίβεια.

1.5. Ανάλυση Τοποθεσίας

Οι τεχνικές που χρησιμοποιούνται για την ανάλυση της τοποθεσίας (scene analysis) επιδιώκουν να ελαττώσουν την αρνητική επίδραση που έχουν οι αντιξοότητες του περιβάλλοντος στο τελικό αποτέλεσμα. Οι τεχνικές που χρησιμοποιούν τριγωνισμό (triangulation) εξαρτώνται πολύ από παράγοντες που επηρεάζουν τη διάδοση του σήματος, όπως θόρυβος, παρεμβολές, πολλαπλές ανακλάσεις, διαθλάσεις, εξασθενήσεις, σκίαση, σκεδάσεις. Επίσης, η έλλειψη οπτικής επαφής μεταξύ πομπού και δέκτη, πολλές φορές, είναι μία πολύ σημαντική αιτία σφάλματος στις triangulation τεχνικές. Και στην περίπτωση των τεχνικών τριγωνισμού που βασίζονται σε RSS μετρήσεις, η πιθανότητα σημαντικών σφαλμάτων είναι μεγάλη, λόγω της αδυναμίας να βρεθεί ένα μοντέλο που να αντιστοιχίζει την RSS μέτρηση με εκείνη την απόσταση που υπάρχει καλή ακρίβεια, σε έναν εσωτερικό χώρο. Επομένως, μία μεγάλη πρόκληση των τεχνικών που χρησιμοποιούν scene analysis, είναι να αντιμετωπίσουν με επιτυχία το πρόβλημα που δημιουργείται από την επίδραση των συνθηκών διάδοσης στις μετρήσεις, και ιδιαιτέρως στις RSS μετρήσεις, επειδή αυτές χρησιμοποιούνται συχνότερα στη scene analysis.

- Ένα πλεονέκτημα των τεχνικών scene analysis είναι ότι μελετούν το πρόβλημα εντοπισμού θέσης με μία διαφορετική οπτική γωνία από ότι οι triangulation τεχνικές. Δηλαδή, οι scene analysis τεχνικές δεν στηρίζονται σε μία ή λίγες μετρήσεις, η ακρίβεια των οποίων θα επηρέαζε σε αυτή την περίπτωση το αποτέλεσμα σε μεγάλο βαθμό. Αντιθέτως, στηρίζονται σε ένα μεγάλο σύνολο μετρήσεων, το οποίο αναλύεται με κατάλληλο τρόπο για να δώσει μία εικόνα για το περιβάλλον της παρούσας μελέτης. Έτσι, η προσοχή των scene analysis τεχνικών επικεντρώνεται περισσότερο στο να καταφέρουν να πάρουν και να εκμεταλλευτούν τις μετρήσεις με τέτοιο τρόπο, ώστε να αποτυπωθούν όσο το δυνατό καλύτερα τα ιδιαίτερα χαρακτηριστικά του περιβάλλοντος, και δεν επικεντρώνεται τόσο πολύ στο πόσο ακριβείς είναι οι μετρήσεις. Σε ένα περιβάλλον εσωτερικού χώρου, οι τιμές RSS που μετρώνται σε μία σταθερή θέση είναι πολύ πιθανό να διαφέρουν αρκετά μεταξύ τους, λόγω της πολυπλοκότητας και της συνεχούς μεταβολής του περιβάλλοντος. Αν χρησιμοποιείτο μία triangulation τεχνική, τότε το σφάλμα θέσης θα ήταν μεγάλο, γιατί οι μεγάλες αποκλίσεις των RSS τιμών από το μοντέλο μεταξύ RSS και αντίστοιχης απόστασης που θα χρησιμοποιούταν, θα έκαναν το μοντέλο αυτό να μην είναι λειτουργικό. Όμως, οι scene analysis τεχνικές δεν επηρεάζονται τόσο πολύ από τις μεταβολές των RSS τιμών που μετρώνται. Αυτό, γιατί κατά το offline στάδιο, σε κάθε μία από τις υποψήφιες θέσεις, συλλέγονται αρκετές μετρήσεις, οι οποίες προφανώς διαφέρουν μεταξύ τους, λόγω των συνθηκών εσωτερικού χώρου. Συνεπώς, το offline στάδιο βοηθά να κατανοηθεί σε κάποιο βαθμό, με έμμεσο τρόπο, το πώς μεταβάλλονται οι RSS μετρήσεις σε διάφορα γνωστά σημεία του χώρου. Επομένως, οι ασταθείς RSS μετρήσεις δεν αποτελούν μεγάλο πρόβλημα, επειδή οι μεταβολές των μετρήσεων σε διάφορα σημεία του χώρου προσπαθούν να προβλεφθούν στο offline στάδιο.

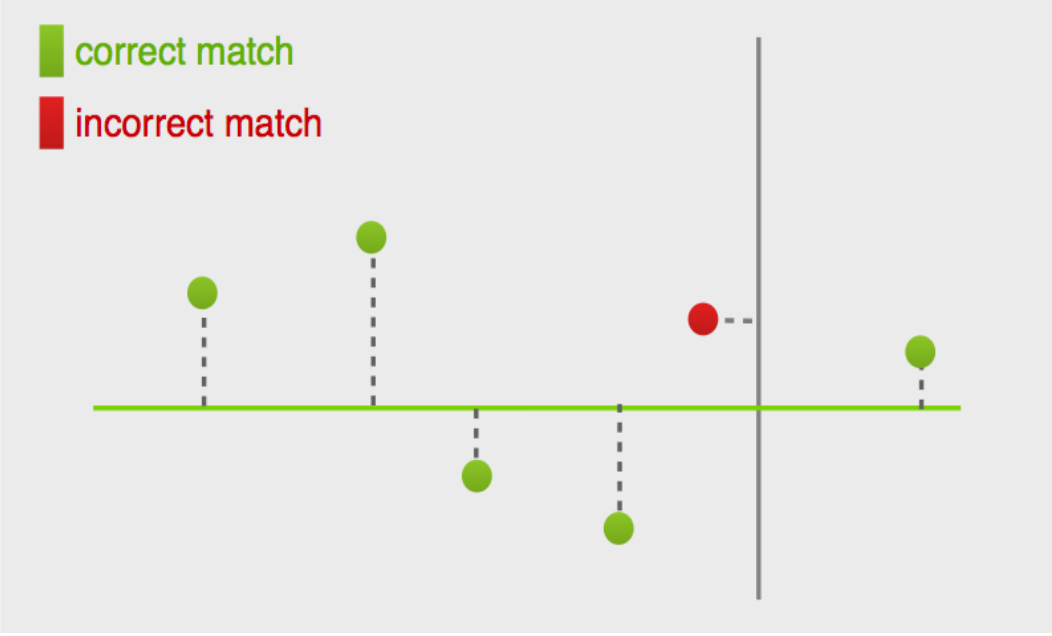
Επιπλέον, στις scene analysis τεχνικές δεν συγκρίνεται μόνο μία online μέτρηση με μία ή λίγες τιμές του radio map. Αντιθέτως, έστω και με έμμεσο τρόπο, συγκρίνονται πολλές offline τιμές με online τιμές. Αν λόγω κάποιας μεταβολής του περιβάλλοντος άλλαζαν αρκετά κάποιες RSS τιμές στο online στάδιο, σε σύγκριση με τις τιμές που θα είχαν μετρηθεί στο offline στάδιο, το σφάλμα του τελικού αποτελέσματος θα ήταν μεγαλύτερο αν έπρεπε να συγκριθούν λίγες τιμές μεταξύ τους, από ότι αν έπρεπε να συγκριθούν πολλές

τιμές μεταξύ τους, γιατί τότε οι μετρήσεις με μεγάλες αποκλίσεις θα ήταν αμελητέα ποσότητα στο σύνολο όλων των μετρήσεων που θα συγκρίνονταν.

➢ Ένα ακόμα μεγάλο πλεονέκτημα των scene analysis τεχνικών είναι ότι δεν απαιτείται η ύπαρξη οπτικών επαφών, και επομένως είναι πολύ λειτουργικές σε περιβάλλοντα εσωτερικών χώρων. Όμως, σε πολλές περιπτώσεις, κάποιες τιμές που αποθηκεύονται σε έναν radio map μπορεί να οδηγήσουν σε λάθος συμπεράσματα για τη θέση ενός αντικειμένου.

- Για παράδειγμα, έστω ότι υπάρχουν δύο υποψήφιες θέσεις Α και Β, και το αντικείμενο άγνωστης θέσης βρίσκεται πολύ κοντά στην υποψήφια θέση Α, και μακριά από την υποψήφια θέση Β. Σε αυτή την περίπτωση, θα ήταν λογικό οι online μετρήσεις του αντικειμένου να είναι πιο κοντά στις μετρήσεις της κοντινής υποψήφιας θέσης Α, παρά στις μετρήσεις της μακρινής θέσης Β. Όμως, αν το τμήμα του εσωτερικού χώρου στο οποίο βρίσκεται το αντικείμενο υποφέρει από έντονες διαλείψεις, τότε είναι πιθανό οι online μετρήσεις του αντικειμένου να διαφέρουν αρκετά από τις offline τιμές της κοντινής στο αντικείμενο θέσης Α. Και αν τύχει οι online μετρήσεις να είναι πολύ κοντά στις offline μετρήσεις της θέσης Β, τότε η μέθοδος είναι πιθανό να θεωρήσει ότι το αντικείμενο βρίσκεται κοντά στη θέση Β, αν και στην πραγματικότητα η θέση Β δεν είναι κοντινή στο αντικείμενο. Άρα, η κατασκευή του radio map, καθώς και η επιλογή των υποψήφιων θέσεων, πρέπει να γίνονται προσεχτικά ώστε να μειώνεται όσο το δυνατό περισσότερο η πιθανότητα να ληφθούν παραπλανητικά αποτελέσματα σχετικά με τη θέση ενός αντικειμένου.

- Επίσης, στην περίπτωση που οι offline RSS τιμές δύο υποψήφιων θέσεων τυχαίνει να είναι πολύ κοντά μεταξύ τους, υπάρχει η πιθανότητα η μέθοδος εντοπισμού θέσης να μην μπορέσει να διακρίνει με επιτυχία ποια από τις δύο υποψήφιες θέσεις προσεγγίζει καλύτερα τη θέση του αντικειμένου.



Σχήμα 24. Παράδειγμα πιθανού λάθους στο ταίριασμα των online και offline τιμών στον radio map

Επομένως, σε μία μέθοδο scene analysis προκειμένου να κατασκευαστεί ο radio map με κατά τον καλύτερο δυνατό τρόπο, θα πρέπει οι offline μετρήσεις να συλλέγονται και να επεξεργάζονται με τέτοιο τρόπο, ώστε το τελικό αποτέλεσμα να είναι όσο το δυνατό πιο ακριβές.

Τέλος, στις scene analysis μεθόδους, αντιστοιχίζεται η θέση του αντικειμένου άγνωστης θέσης με την πιο κατάλληλη υποψήφια θέση. Όμως, η αντιστοίχηση αυτή δεν δίνει μια ακριβή θέση για το αντικείμενο, αλλά δείχνει την περιοχή του χώρου στην οποία βρίσκεται το αντικείμενο. Δηλαδή, το περιβάλλον της παρούσας μελέτης χωρίζεται σε μικρότερες υποπεριοχές. Κάθε υποπεριοχή αντιπροσωπεύεται από μία ή περισσότερες υποψήφιες θέσεις. Έτσι, όταν θεωρείται μία υποψήφια θέση ως τη θέση του αντικειμένου, στη πραγματικότητα θεωρείται ότι το αντικείμενο βρίσκεται στην υποπεριοχή του χώρου της παρούσας μελέτης, η οποία αντιπροσωπεύεται από την υποψήφια θέση αυτή. Όμως, άλλες μέθοδοι που χρησιμοποιούν scene analysis δίνουν ως αποτέλεσμα τις συντεταγμένες στις οποίες βρίσκεται το αντικείμενο άγνωστης θέσης. Τα σημεία στα οποία διαφέρουν οι διάφορες τεχνικές που βασίζονται στην scene analysis είναι η μέθοδος κατασκευής του radio map, και ο τρόπος με τον οποίον εκμεταλλεύονται τις online μετρήσεις και τον radio map, για τον εντοπισμό μιας θέσης. Ένας αλγόριθμος εντοπισμού θέσης είναι οι k πλησιέστεροι γείτονες (k-Nearest-Neighbor: kNN).

1.5.1. Ο αλγόριθμος k Πλησιέστερων Γειτόνων (kNN)

Ο αλγόριθμος *k* Πλησιέστερων Γειτόνων (k-Nearest-Neighbor: kNN) συγκρίνει τις μετρήσεις που συλλέχθηκαν κατά το online στάδιο με τις offline τιμές που είναι αποθηκευμένες στον radio map. Στη συνέχεια, από το σύνολο όλων των υποψήφιων θέσεων, επιλέγει τις *k* υποψήφιες θέσεις με offline τιμές που βρίσκονται πιο κοντά στις online μετρήσεις, και λαμβάνει υπόψη του τις *k* υποψήφιες θέσεις που επέλεξε, με σκοπό να εκτιμήσει τη θέση του αντικείμενου άγνωστης θέσης. Πιο αναλυτικά, κατά το offline στάδιο, για κάθε μία υποψήφια θέση l_i αποθηκεύεται στον radio map ένα διάνυσμα τιμών:

$S_i = (RSS_{1i}, RSS_{2i}, \ldots, RSS_{mi})$, το οποίο αποτελείται από RSS τιμές που χαρακτηρίζουν τη συμπεριφορά των σημάτων που εκπέμπονται από *m* σταθμούς βάσης στην υποψήφια θέση l_i.

Επίσης, θεωρείται ότι κατά το online στάδιο, το αντικείμενο άγνωστης θέσης λαμβάνει ένα διάνυσμα RSS μετρήσεων $o = (o_1, o_2, \ldots, o_m)$ από *m* σταθμούς βάσης. Τότε, ο kNN αλγόριθμος θα προσπαθήσει να υπολογίσει τις αποστάσεις ανάμεσα στο διάνυσμα online μετρήσεων *o* και στο διάνυσμα offline τιμών S_i κάθε μίας υποψήφιας θέσης l_i . Ένα χρήσιμο μέτρο της απόστασης μεταξύ δύο διανυσμάτων στον χώρο είναι η ευκλείδεια διανυσματική απόσταση. Επομένως, αν χρησιμοποιηθεί η ευκλείδεια απόσταση για να μετρηθεί το πόσο κοντά βρίσκεται το διάνυσμα online μετρήσεων στο διάνυσμα offline τιμών κάθε μίας υποψήφιας θέσης, τότε μπορεί να πραγματοποιηθούν οι εξής υπολογισμοί :

$$d_i = \sqrt{(o_1 - RSS_{1i})^2 + (o_2 - RSS_{2i})^2 + \ldots + (o_m - RSS_{mi})^2}$$

i = 1, ... , n, όπου *n* ο αριθμός των υποψήφιων θέσεων, και d_i η ευκλείδεια απόσταση του διανύσματος online τιμών *o* από το διάνυσμα offline τιμών S_i της υποψήφιας θέσης l_i.

.

Όταν ο kNN αλγόριθμος έχει πλέον στη διάθεσή του τις παραπάνω αποστάσεις ανάμεσα σε online και offline διανύσματα, καλείται να επιλέξει τις *k* υποψήφιες θέσεις με τις μικρότερες αποστάσεις d_i. Δηλαδή, καλείται να επιλέξει τις *k* υποψήφιες θέσεις με offline τιμές που βρίσκονται πιο κοντά στις online μετρήσεις. Επομένως, ο kNN αλγόριθμος, από τις *n* συνολικά υποψήφιες θέσεις του χώρου, διαλέγει τις *k* υποψήφιες θέσεις στις οποίες θεωρεί ότι το αντικείμενο άγνωστης θέσης βρίσκεται πιο κοντά.

Ο τελικός στόχος ενός kNN αλγόριθμου, όπως και κάθε αλγόριθμου εντοπισμού θέσης, είναι να δώσει μία εκτίμηση της θέσης ενός αντικειμένου. Αν ο χώρος της παρούσας μελέτης χωριζόταν σε υποπεριοχές, και κάθε υποπεριοχή αντιπροσωπευόταν από μία ή περισσότερες υποψήφιες θέσεις, τότε ο kNN αλγόριθμος θα εκτιμούσε ότι το αντικείμενο άγνωστης θέσης βρίσκεται στην υποπεριοχή του χώρου, στην οποία θα ανήκαν οι περισσότερες από τις k υποψήφιες θέσεις που έχουν επιλεχθεί. Στην περίπτωση που αναζητιόταν ο εντοπισμός των συντεταγμένων θέσης ενός αντικειμένου, και όχι απλά σε ποια υποπεριοχή βρίσκεται, τότε οι συντεταγμένες (x, y) του αντικείμενου άγνωστης θέσης υπολογίζονται παίρνοντας το μέσο όρο των συντεταγμένων (x_i, y_i) των *k* υποψήφιων θέσεων που επιλέχτηκαν, δηλαδή:

$$(x, y) = ((x_1, y_1) + \ldots + (x_k, y_k)) / k$$

Όμως, από τις *k* υποψήφιες θέσεις που επιλέχτηκαν, δεν έχουν όλες την ίδια απόσταση από το αντικείμενο άγνωστης θέσης. Επίσης, αν κάποιες από τις *k* υποψήφιες θέσεις απέχουν πολύ περισσότερο από το αντικείμενο άγνωστης θέσης από ότι οι άλλες, τότε ο υπολογισμός του μέσου όρου από τον παραπάνω τύπο μπορεί να οδηγήσει σε σημαντικό σφάλμα. Επομένως, σε περιπτώσεις όπου ο μέσος όρος δίνει σημαντικό σφάλμα, θα ήταν καλύτερο να τροποποιηθεί ο υπολογισμός του. Δηλαδή, ο μέσος όρος των συντεταγμένων των *k* υποψήφιων θέσεων, μπορεί να λαμβάνει υπόψη του ως βάρη τις αποστάσεις των διανυσμάτων offline τιμών των *k* υποψήφιων θέσεων από το διάνυσμα online μετρήσεων. Έτσι, με τον υπολογισμό ενός μέσου όρου συντεταγμένων με βάρη, το τελικό σφάλμα μειώνεται, γιατί δίνεται μεγαλύτερη σημασία στις συντεταγμένες των υποψήφιων θέσεων που έχουν μικρότερες αποστάσεις d_i από το αντικείμενο άγνωστης θέσης. Όμως, αν και βελτιώνεται η αξιοπιστία του συστήματος, όταν χρησιμοποιούνται βάρη αυξάνεται η πολυπλοκότητα του αλγόριθμου.

Η απόσταση που υπολογίζει ο kNN αλγόριθμος ανάμεσα στο διάνυσμα RSS τιμών μίας υποψήφιας θέσης (x_i, y_i) και στο διάνυσμα μετρήσεων της θέσης του αντικείμενου άγνωστης θέσης, είναι δυνατό να μεταβάλλεται, ακόμα και αν το αντικείμενο άγνωστης θέσης μένει ακίνητο. Αυτό, γιατί στο σημείο όπου βρίσκεται το αντικείμενο άγνωστης θέσης, είναι δυνατό το σήμα να μεταβάλλεται έντονα, και έτσι οι online RSS μετρήσεις στο σημείο αυτό να διαφέρουν σημαντικά με την πάροδο του χρόνου. Επίσης, σε ορισμένες περιπτώσεις, αν και η θέση του αντικειμένου άγνωστης θέσης και μία υποψήφια θέση βρίσκονται πολύ κοντά μεταξύ τους, είναι δυνατό τα RSS διανύσματά τους να διαφέρουν πολύ. Έτσι, αν και ο radio map θα πρέπει να δίνει μία όσο το δυνατό καλύτερη απεικόνιση των χαρακτηριστικών διάδοσης του χώρου, και οι τιμές του radio map θα πρέπει να ενσωματώνουν τη μεταβλητότητα των RSS τιμών στο χώρο, καμιά φορά η αβεβαιότητα των μετρήσεων είναι τόσο μεγάλη, με αποτέλεσμα να μην εξασφαλίζεται ότι ο υπολογισμός μιας διανυσματικής απόστασης θα δίνει σωστή πληροφόρηση για το πόσο κοντά σε μία υποψήφια θέση βρίσκεται η θέση του αντικείμενου, στην πραγματικότητα. Επομένως, αν στον αλγόριθμο kNN ίσχυε k=1, δηλαδή αν λαμβανόταν υπόψη μόνο η μικρότερη διανυσματική απόσταση που υπολογίστηκε για την εκτίμηση της θέσης ενός αντικειμένου,

τότε είναι πιθανό η εκτίμηση της θέσης του αντικειμένου να είναι αναξιόπιστη. Όμως, αν ο kNN αλγόριθμος επέλεγε μία τιμή του *k* μεγαλύτερη του ένα, τότε το σφάλμα στο τελικό αποτέλεσμα θα μειωνόταν. Αυτό, γιατί όσο περισσότερες διανυσματικές αποστάσεις λαμβάνονται υπόψη στο τελικό αποτέλεσμα, τόσο μικρότερο είναι το πιθανό ποσοστό των αναξιόπιστων διανυσματικών αποστάσεων. Επομένως, οι kNN αλγόριθμοι είναι καλό να επιλέγουν μεγάλες τιμές του *k*, για να αντιμετωπίζουν τα προβλήματα που δημιουργούν στη διάδοση του σήματος τα ιδιαίτερα χαρακτηριστικά ενός εσωτερικού χώρου.

Οι βασικές περιπτώσεις που καθιστούν αναγκαία την επιλογή μεγαλύτερων αριθμών *k* είναι όταν υπάρχουν παραπάνω από μία υποψήφιες θέσεις με "δαχτυλικά αποτυπώματα" που απέχουν παρόμοιες διανυσματικές αποστάσεις από το δειγματικό διάνυσμα μετρήσεων του online σταδίου, ή όταν υπάρχουν υποψήφιες θέσεις που απέχουν, όχι ιδιαίτερα μικρές, αλλά παρόμοιες φυσικές αποστάσεις από την πραγματική θέση, και παράλληλα βρίσκονται ανά ζεύγη σε αντίθετους προσανατολισμούς σε σχέση με την πραγματική θέση. Στις περιπτώσεις αυτές, ο μέσος όρος των *k* υποψήφιων θέσεων θα προσέγγιζε καλύτερα την πραγματική θέση.

Όμως, αν η τιμή του *k* είναι μεγάλη, τότε ο μέσος όρος των συντεταγμένων των *k* υποψήφιων θέσεων είναι πιθανό να επηρεάζεται και από υποψήφιες θέσεις οι οποίες δίνουν αρκετά μεγάλες διανυσματικές αποστάσεις. Ένας τρόπος ελάττωσης του σφάλματος αυτού είναι ο υπολογισμός ενός μέσου όρου με βάρη, όπως περιγράφτηκε παραπάνω. Κατόπιν όλων αυτών εξάγεται το συμπέρασμα ότι η επιλογή της τιμής του *k* δεν είναι καθόλου εύκολη υπόθεση, και επομένως υπάρχει ανάγκη να επιλεγεί μια βέλτιστη τιμή του *k*, η οποία να δίνει το ελάχιστο δυνατό σφάλμα σε έναν kNN αλγόριθμο.

Έχει παρατηρηθεί ότι ο kNN αλγόριθμος παρουσιάζει πολύ ικανοποιητική ακρίβεια. Ένα σημαντικό μειονέκτημα του kNN αλγόριθμου είναι το πιθανώς μεγάλο χρονικό διάστημα που απαιτείται για να εκτιμήσει μία θέση. Όμως, το χρονικό αυτό διάστημα μπορεί να μειωθεί αισθητά αν μειωθεί και ο αριθμός των πληροφοριών που αποθηκεύονται κατά το offline στάδιο. Επίσης, έχει αναφερθεί ότι ο kNN αλγόριθμος εξασφαλίζει καλή ευρωστία (robustness) και εξελιξιμότητα (scalability).

Κεφάλαιο 2

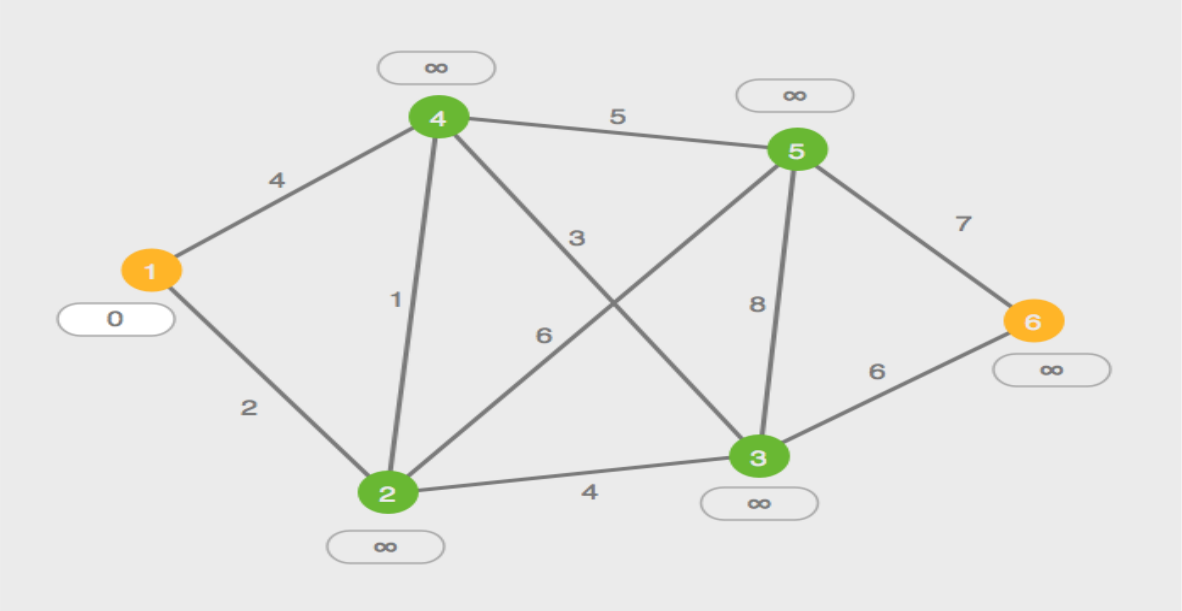
Αλγόριθμοι δρομολόγησης

2. Εισαγωγή

Οι τεχνικές δρομολόγησης χρησιμοποιούν αλγορίθμους οι οποίοι βρίσκουν τη συντομότερη πορεία μεταξύ δύο θέσεων. Οι συνδέσεις μεταξύ των κόμβων μιας γραφικής παράστασης θα αντιπροσωπευθούν από μια δισδιάστατη μήτρα. Στη σχέση σύνδεσης μεταξύ των κόμβων υπάρχει ο παράγοντας βάρος (Weight) ή κόστος (Cost), ο οποίος εξαρτάται από την απόσταση των κόμβων, το χρόνο που χρειάζεται για να διανυθούν, ή το βαθμό ευκολίας πρόσβασης στον επόμενο κόμβο. Η απόδοση του αλγορίθμου εξαρτάται από την επιλογή της δομής δεδομένων καθώς επηρεάζεται από το μέγεθος της βάσης δεδομένων. Μεταξύ των κοινών αλγορίθμων είναι ο Dijkstra, ο Bellman-Ford, και ο A*. Η περιγραφή κάθε αλγορίθμου θα αναλυθεί στις ακόλουθες υποενότητες.

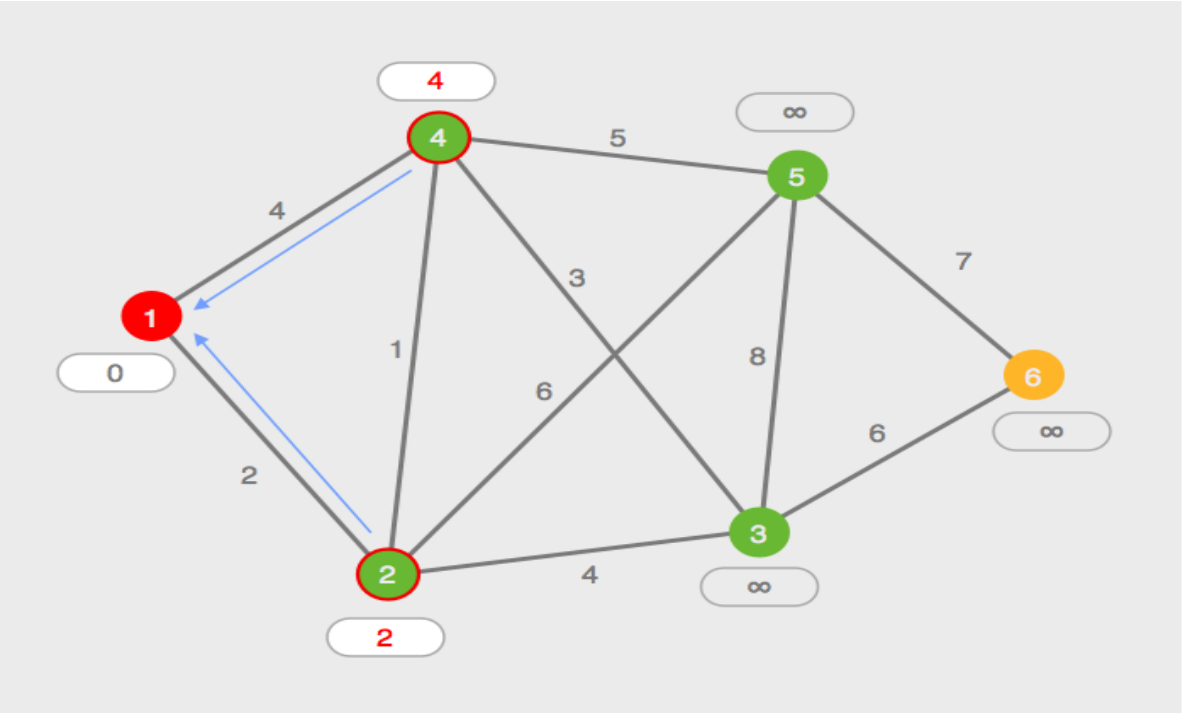
2.1. Ο αλγόριθμος του Dijkstra

Ο αλγόριθμος του Dijkstra είναι ένας από τους περισσότερο χρησιμοποιημένους αλγορίθμους δρομολόγησης. Ο ολλανδός επιστήμονας Edsger Dijkstra εφήϋρε τον αλγόριθμο το 1959. Αυτός ο αλγόριθμος είναι κατάλληλος για προβλήματα που εξετάζουν έναν μόνο κόμβο πηγής και έναν ή περισσότερους κόμβους προορισμού. Ο αλγόριθμος λειτουργεί με την προώθηση ενός μόνο κόμβου σε ένα χρόνο (διάρκεια ενός βρόχου), ο οποίος αρχίζει από τον κόμβο πηγή. Σε κάθε βήμα κατά τη διάρκεια του βρόχου, ο αλγόριθμος επιλέγει έναν κόμβο, εκείνος ο οποίος έχει στο σύνολο μαζί με τους άλλους πιθανούς ενδιάμεσους κόμβους το ελάχιστο κόστος από τον κόμβο πηγή. Ο κόμβος που επισκέπτεται από την πηγή σημειώνεται ως βελτιστοποιημένος και το κόστος σε όλους τους γειτονικούς κόμβους αξιολογείται. Ο αλγόριθμος Dijkstra αποδεικνύεται από μαθηματική άποψη ότι βρίσκει την πιο σύντομη διαδρομή. Το βελτιστοποιημένο κόστος στον προορισμό βρίσκεται μόλις φθάσει ο αλγόριθμος στον κόμβο προορισμού. Με ένα σύνολο απεικονίσεων του αλγορίθμου Dijkstra θα παρουσιαστεί πως ο αλγόριθμος βρίσκει τις συντομότερες διαδρομές σε ένα σύνολο έξι κόμβων με πηγή τον κόμβο 1 και προορισμό τον κόμβο 6.



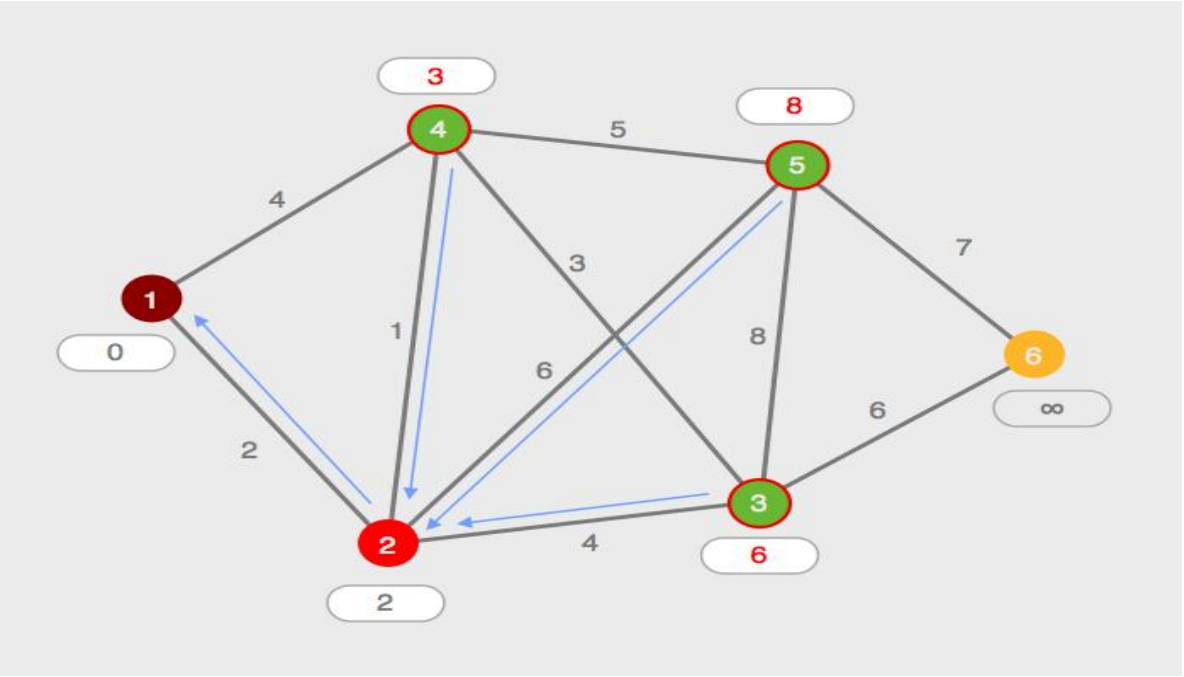
Σχήμα 25. Το διάγραμμα Dijkstra τη χρονική στιγμή 0 μετά την αρχικοποίηση

Στο Σχήμα 25, ο αριθμός στην άκρη αντιπροσωπεύει το κόστος που θα έχει για να ταξιδέψει μέσω εκείνης της άκρης. Ο αριθμός δίπλα σε κάθε κόμβο αντιπροσωπεύει το συνολικό κόστος που θα έχει για να ταξιδέψει από τον κόμβο πηγή σε αυτόν τον κόμβο. Μάλιστα, αυτά τα δεδομένα θα αποθηκευτούν σε ένα μονοδιάστατο πίνακα. Αρχικά, το κόστος σε όλους κόμβους εκτός από την πηγή θα δοθεί η αξία του απείρου. Το άπειρο κόστος αντιπροσωπεύει ότι καμία διαδρομή δεν έχει βρεθεί για να ταξιδεύει από τον κόμβο πηγή σε αυτόν τον κόμβο. Το κόστος από την πηγή στην πηγή, διαισθητικά, είναι ίσο με 0.



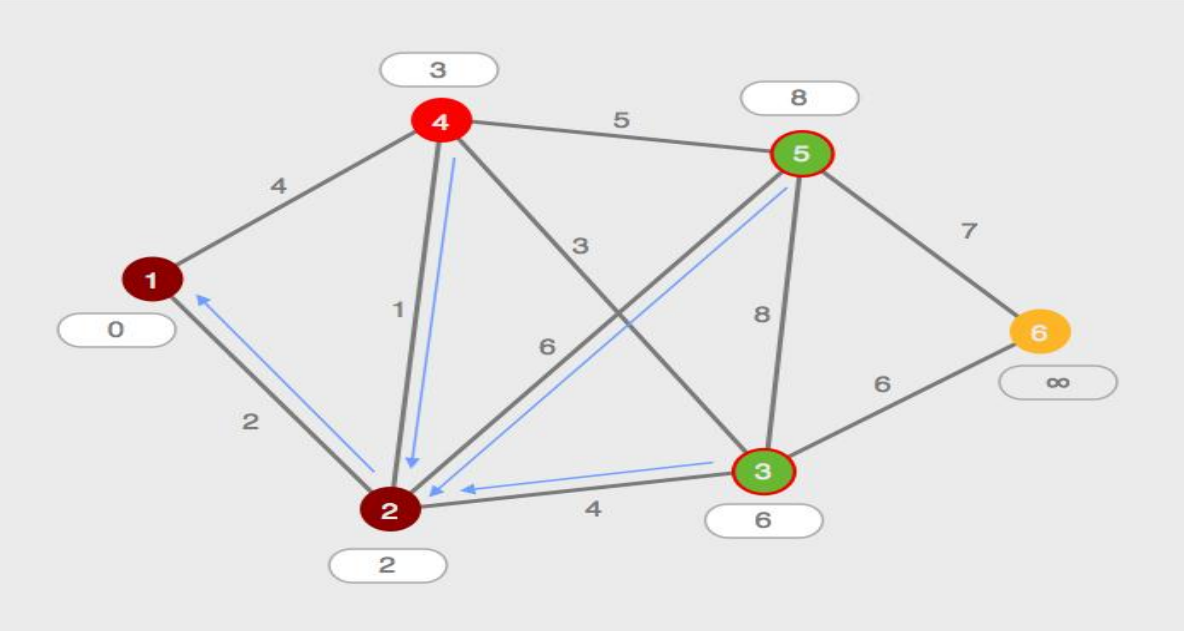
Σχήμα 26. Το διάγραμμα Dijkstra ξεκινώντας από τον κόμβο 1

Ο αλγόριθμος αρχίζει το βρόχο του με την επιλογή του κόμβου, ο οποίος έχει το μικρότερο κόστος, που σε αυτήν την περίπτωση είναι ο κόμβος 1 με κόστος 0. Ο κόμβος που αξιολογείται είναι ζωγραφισμένος με κόκκινο χρώμα. Όταν ένας κόμβος βελτιστοποιείται (optimize), σημαίνει ότι το κόστος του θα είναι το τελικό και δεν μπορεί να είναι μικρότερο. Κατόπιν, το κόστος όλων των κόμβων που είναι δίπλα στον κόμβο 1 και δεν έχουν ακόμα βελτιστοποιηθεί, θα συγκριθεί με το κόστος από τον κόμβο 1 και την άκρη μεταξύ του κόμβου 1 και εκείνου του κόμβου. Σε αυτήν την περίπτωση, δεδομένου ότι το κόστος για τη μετάβαση στον κόμβο 2 και τον κόμβο 4 είναι άπειρο, θα ενημερωθούν τα κόστη τους σε 2 και 4, αντίστοιχα. Το κόστος στον κόμβο 2 είναι 2 επειδή πρέπει να ταξιδέψει μέσω του κόμβου 1 με κόστος 0 το οποίο προστίθεται στο κόστος που έχει για να πάει από τον κόμβο 1 στον κόμβο 2, το οποίο είναι 2. Το μπλε βέλος αντιπροσωπεύει την διαδρομή ανίχνευσης και θα χρησιμοποιηθεί αφότου τελειώσει ο αλγόριθμος, προκειμένου να διαμορφωθεί η διαδρομή. Αυτό παρουσιάζεται στην Σχήμα 26.



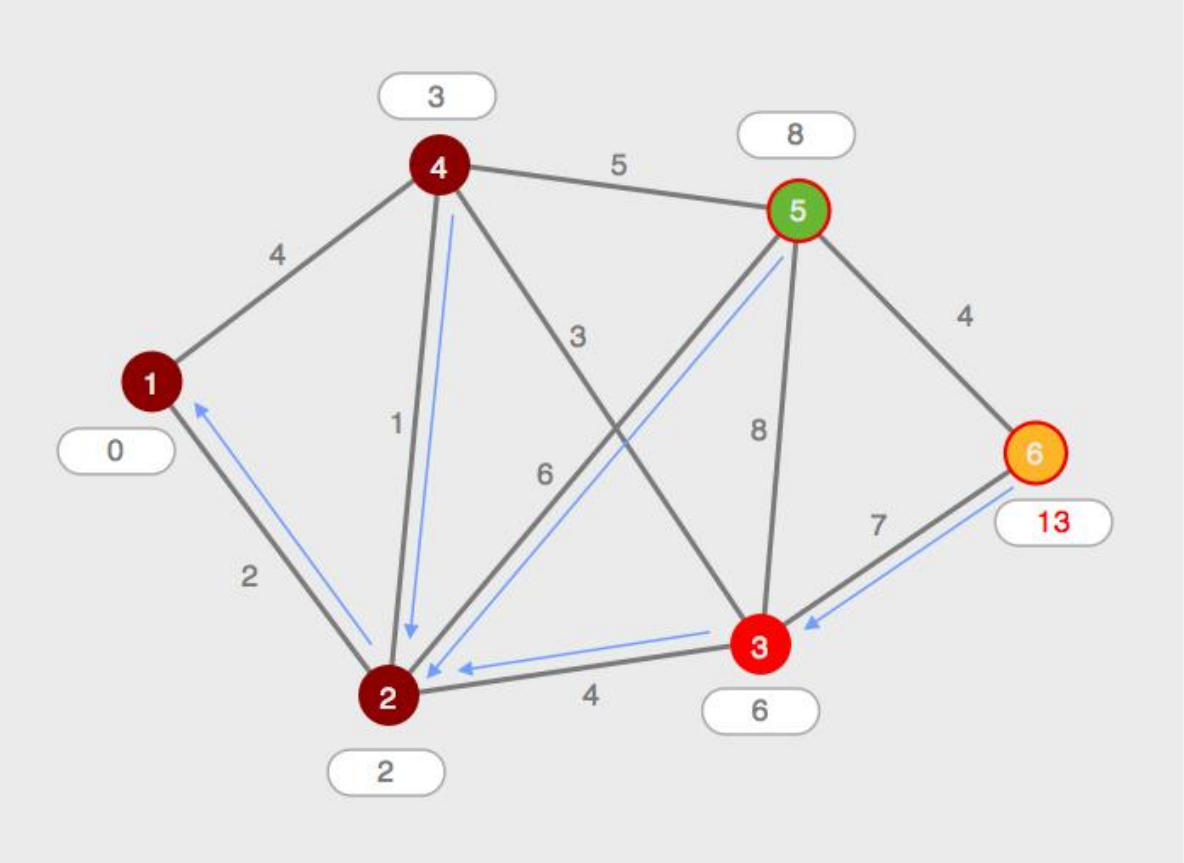
Σχήμα 27. Το διάγραμμα Dijkstra όταν ο κόμβος 2 έχει βελτιστοποιηθεί

Έπειτα ο κόμβος με το μικρότερο κόστος, που είναι κόμβος 2, αξιολογείται. Σημειώνεται δε ότι ο κόμβος 1 έχει ζωγραφιστεί με σκούρο κόκκινο χρώμα για να φαίνεται ότι έχει βελτιστοποιηθεί και δεν θα ενημερωθεί άλλο. Έπειτα, οι κόμβοι δίπλα στον κόμβο 2 αξιολογούνται, συμπεριλαμβανομένου του κόμβου 4, του κόμβου 3, και του κόμβου 5. Το κόστος στον κόμβο 4 ήταν 4 στο προηγούμενο βήμα με την διαδρομή άμεσα από την πηγή (κόμβο 1). Επισημαίνεται, όταν αξιολογείται ο κόμβος 2, το κόστος στον κόμβο 4 μέσω του κόμβου 2 είναι μικρότερο από το τρέχον κόστος του. Επομένως, το κόστος θα ενημερωθεί ως 3, με τα μπλε βέλη, αυτή τη φορά, να δείχνουν την κατεύθυνση της διαδρομής προς τον κόμβο 2, όπως παρουσιάζεται στο Σχήμα 27.



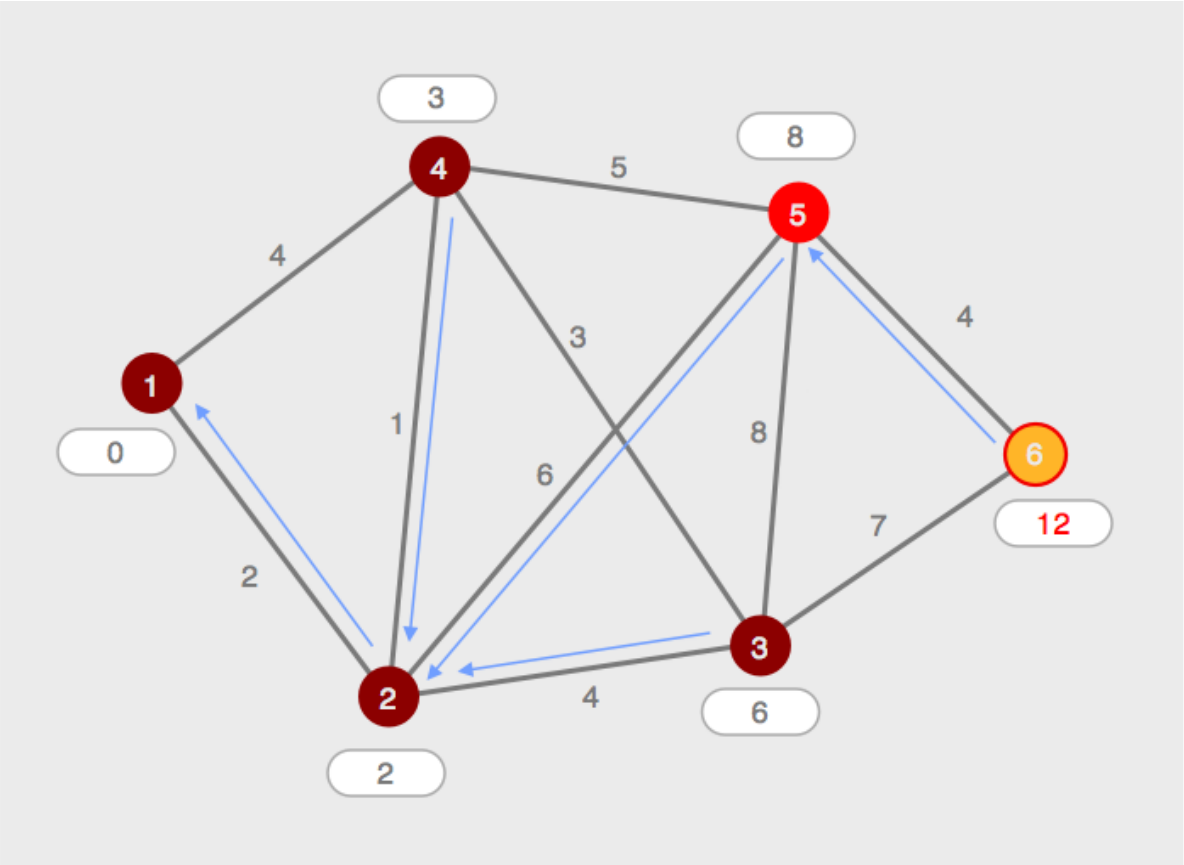
Σχήμα 28. Το διάγραμμα Dijkstra όταν ο κόμβος 4 έχει βελτιστοποιηθεί

Στο επόμενο βήμα, ο κόμβος 4 αξιολογείται και χαρακτηρίζεται βελτιστοποιημένος. Το τρέχον κόστος στον κόμβο 3 και τον κόμβο 5 είναι λιγότερο απ' ότι μέσω του κόμβου 4. Επομένως, τα κόστη των κόμβων αυτών δεν ενημερώνονται, όπως είναι εμφανές και στο Σχήμα 28.



Σχήμα 29. Το διάγραμμα Dijkstra όταν ο κόμβος 3 έχει βελτιστοποιηθεί

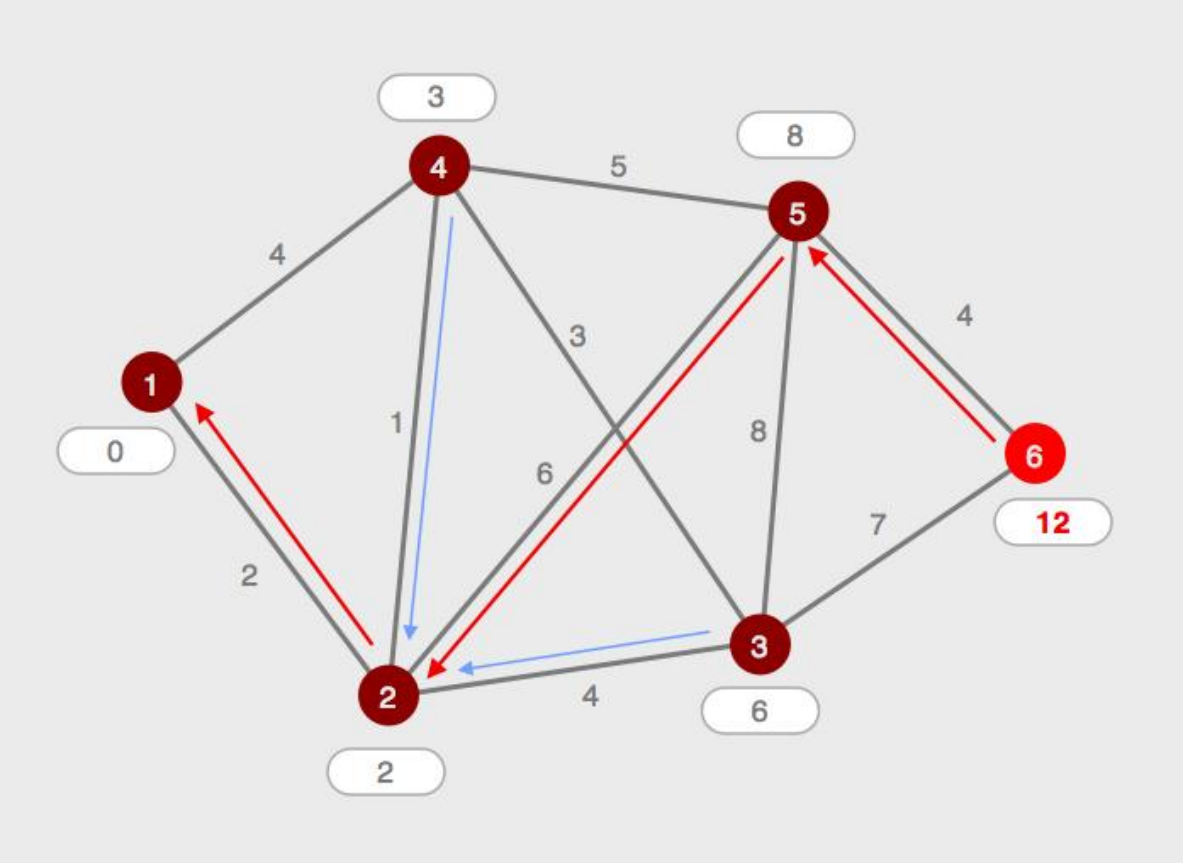
Το επόμενο βήμα είναι να αξιολογηθεί ο κόμβος 3. Στο Σχήμα 29, μια διαδρομή έχει βρεθεί στον κόμβο προορισμού, η οποία έχει κόστος 13 και περνά μέσω των κόμβων 2 και 3. Παρόλα αυτά, αυτό δεν είναι το τελικό κόστος επειδή ο κόμβος προορισμού δεν έχει βελτιστοποιηθεί ακόμα. Αυτό σημαίνει ότι μάλλον δεν υπάρχει άλλη διαδρομή, η οποία να έχει μικρότερο κόστος.



Σχήμα 30. Το διάγραμμα Dijkstra όταν ο κόμβος 5 έχει βελτιστοποιηθεί

Σε αυτό το βήμα, ο κόμβος 5 αξιολογείται και μια συντομότερη διαδρομή προς τον προορισμό βρέθηκε με κόστος 12. Στη συνέχεια, η συντομότερη διαδρομή προς τον προορισμό ενημερώνεται και περνάει μέσω του κόμβου 5. Η απεικόνιση αυτού του βήματος εμφανίζεται στο Σχήμα 30. Στο τέλος, ο προορισμός αξιολογείται και βελτιστοποιείται. Το τελικό κόστος έχει την αξία 12 και είναι το ελάχιστο κόστος που χρειάζεται για να ταξιδέψει από την πηγή στον προορισμό. Χρησιμοποιώντας τα βέλη ανίχνευσης, μπορεί να διαμορφωθεί η ελάχιστη διαδρομή προς τον προορισμό μέσω των κόμβων 2 και 5 όπως απεικονίζεται στο Σχήμα 31.

Η σημαντικότερη πτυχή του αλγορίθμου Dijkstra είναι ότι επιλέγει το μικρότερο κόστος που έχει κάποιος κόμβος στην αρχή κάθε βρόχου. Κατά συνέπεια, ο αλγόριθμος μπορεί να ελεγχθεί με την απόδειξη ότι σε κάθε βρόγχο, το ελάχιστο κόστος των κόμβων βελτιστοποιείται. Συνεπώς, έστω ότι υπάρχει μια διαφορετική διαδρομή με ένα μικρότερο κόστος από την πηγή στον κόμβο με το ελάχιστο κόστος, τότε θα πρέπει αυτή η διαδρομή να περάσει μια φορά από οποιοδήποτε υπόλοιπο κόμβο. Με τα δεδομένα αυτά, η επιλογή του κόμβου με το ελάχιστο κόστος δείχνει ότι το κόστος σε όλους τους άλλους κόμβους πρέπει να είναι μεγαλύτερο. Συνεπώς η προσθήκη της άκρης αυξάνει μόνο το συνολικό κόστος το οποίο σε καμία περίπτωση δεν μπορεί να είναι μικρότερο. Αυτό αποδεικνύει το αδύνατο αυτής της υπόθεσης, και κατά συνέπεια το κόστος στον κόμβο με το ελάχιστο κόστος είναι τελικό και βελτιστοποιημένο.

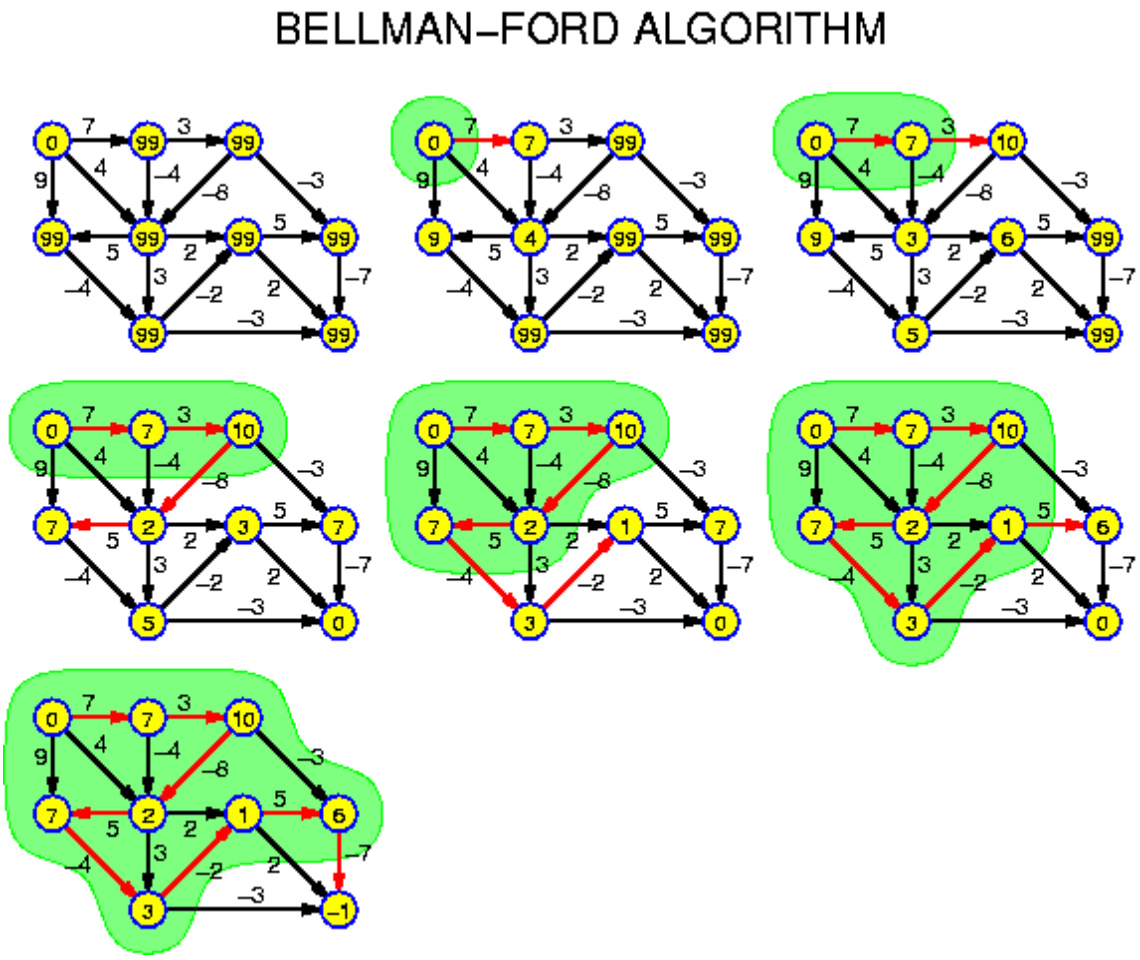


Σχήμα 31. Το διάγραμμα Dijkstra όταν ο κόμβος Προορισμού (κόμβος 6) έχει βελτιστοποιηθεί

2.2. Ο αλγόριθμος των Bellman-Ford

Ο αλγόριθμος των Bellman-Ford έχει ως σκοπό να λύσει το πρόβλημα εύρεσης συντομότερης πορείας σε μια γραφική παράσταση όπου οι άκρες έχουν αρνητικές τιμές βάρους. Δείχνει την ανικανότητα του Dijkstra να δουλέψει με αρνητικές άκρες με την προσθήκη μιας συνθήκης προστασίας που αποτρέπει την πορεία από τη διάβαση μέσω ενός αρνητικού κύκλου. Αντί της επιλογής του ελάχιστου κόστους κόμβου σε κάθε βρόχο και της ενημέρωσης όλων των γειτονικών κόμβων του όπως στον αλγόριθμο Dijkstra, ο αλγόριθμος των Bellman-Ford ενημερώνει όλες τις πιθανές άκρες που υπάρχουν στη γραφική παράσταση. Αυτό αποτρέπει στον αλγόριθμο να επιλέξει τον ίδιο ελάχιστο κόμβο άπειρες φορές (endless loop). Ο αριθμός ενημέρωσης του χρόνου στον αλγόριθμο είναι ισοδύναμος με το μέγιστο μήκος της συντομότερης πορείας που εξετάζεται. Με την ενημέρωση των χρόνων ν φορές, ο οποίος είναι ο αριθμός των vertices στη γραφική παράσταση, ο αλγόριθμος μπορεί να βρει τη βέλτιστη λύση. Παρόλο που ο αλγόριθμος των Bellman-Ford μπορεί να λειτουργήσει με γράφους που έχουν κόστη, ο τρέχων χρόνος είναι χειρότερος από τον αλγόριθμο Dijkstra επειδή πρέπει να ενημερώσει όλες τις πιθανές άκρες στη γραφική παράσταση αντί ακριβώς των διπλανών άκρων από τον κόμβο με το μικρότερο κόστος. Με αυτό τον τρόπο αυξάνεται η συνολική πολυπλοκότητα του αλγορίθμου στο O(N3). Λόγω της υψηλής πολυπλοκότητάς της, ο αλγόριθμος των Bellman-Ford πρέπει μόνο να χρησιμοποιηθεί στην περίπτωση που η γραφική παράσταση έχει άκρες με αρνητικά

βάρη. Στο Σχήμα 32 παρουσιάζεται ένα παράδειγμα εφαρμογής του αλγορίθμου των Bellman-Ford όπου ο γράφος έχει άκρες με αρνητικές τιμές βάρους.



Σχήμα 32. Τα στάδια υπολογισμού συντομότερης λύσης χρησιμοποιώντας τον αλγόριθμο των Bellman-Ford

*2.3. Ο αλγόριθμος A**

Ο αλγόριθμος A* είναι ένας διαφορετικός αλγόριθμος ο οποίος και αυτός προσπαθεί να λύσει το πρόβλημα της συντομότερης διαδρομής μεταξύ δύο κόμβων σε μια γραφική παράσταση. Έχει παρόμοια λειτουργία με τον αλγόριθμο Dijkstra, βρίσκει την συντομότερη πορεία με την προώθηση ενός κόμβου τη φορά από την πηγή. Η διαφορά μεταξύ του αλγορίθμου A* και του αλγορίθμου Dijkstra είναι στη λειτουργία εύρεσης της συντομότερης διαδρομής και στην ενημέρωση της διαδικασίας αυτής. Αντί να ενημερώνει όλους τους γειτονικούς κόμβους, ο αλγόριθμος A* ενημερώνει μόνο τους κόμβους που δεν έχουν επισκεφτεί πριν, με μια αισιόδοξη ελπίδα ότι αυτή θα είναι η συντομότερη διαδρομή επειδή προέρχεται από την συντομότερη "κατ' εκτίμηση πορεία". Η συντομότερη διαδρομή σε κάθε βήμα δεν είναι απόλυτη. Αντιθέτως, η συντομότερη διαδρομή είναι το άθροισμα της απόλυτης διαδρομής από την πηγή στον κόμβο και μια «ευρετική» εκτίμηση της απόστασης από αυτόν τον κόμβο στον προορισμό. Η βέλτιστη απόδοση του αλγορίθμου A* εξαρτάται από το «ευρετικό» πρότυπο. Η «ευρετική» εκτίμηση πρέπει να είναι λογική, και δεν πρέπει να υπερεκτιμήσει την απόσταση μεταξύ του κόμβου και του προορισμού. Ο

«ευρετικός» τύπος της απόστασης μεταξύ του κόμβου και του προορισμού πρέπει να ικανοποιήσει τον όρο ότι δεν υπάρχει καμία διαδρομή στη γραφική παράσταση που θα μπορούσε να είναι μικρότερη από την «ευρετική» εκτίμηση. Ωστόσο, δεν πρέπει να υπερεκτιμηθεί η απόσταση με την ανάθεση μιας πολύ μικρής αξίας, η οποία έχει επιπτώσεις στη διαδικασία επιλογής του ελάχιστου κόμβου. Παραδείγματος χάριν, ένα καλό «ευρετικό» πρότυπο μπορεί να αναπτυχθεί όταν η γραφική παράσταση είναι στις γήινες συντεταγμένες. Το «ευρετικό» πρότυπο μπορεί να υπολογιστεί με την ευκλείδεια απόσταση μεταξύ δύο συντεταγμένων. Αν και ο αλγόριθμος A* παρέχει μια χαμηλότερη πολυπλοκότητα από τον αλγόριθμο Dijkstra, η λύση που καθορίζεται από αυτόν τον αλγόριθμο δεν είναι η βέλτιστη από μαθηματική άποψη, το οποίο σημαίνει ότι η λύση δεν ελέγχεται αν όντως είναι η καλύτερη λύση.

2.4. Σύγκριση αλγορίθμων δρομολόγησης

Στον παρακάτω πίνακα παρουσιάζονται συνοπτικά τα πλεονεκτήματα και τα μειονεκτήματα του κάθε Αλγορίθμου δρομολόγησης:

Routing Algorithm	Θετικά	Αρνητικά
Dijkstra	Πολύ υψηλή ταχύτητα Ιδανικός	Δεν δουλεύει με αρνητικά βάρη στο γράφο
Bellman-Ford	Δουλεύει με αρνητικά βάρη Ιδανικός	Χαμηλή ταχύτητα
A*	Ταχύτατος	Δεν είναι πάντα ιδανικός

Πίνακας 6. Θετικά και αρνητικά των Αλγορίθμων δρομολόγησης

Με βάση τα παραπάνω στοιχεία για την πραγματοποίηση της πλοήγησης της εφαρμογής χρησιμοποιήθηκε ο αλγόριθμος του Dijkstra ως αλγόριθμος δρομολόγησης καθώς στη βάση δεδομένων της εφαρμογής δεν υπάρχουν αρνητικά κόστη. Τα κόστη της εφαρμογής βασίζονται στην απόσταση των κόμβων που έχουν οριστεί στα διάφορα σημεία των εσωτερικών χαρτών των κτηρίων, οπότε είναι μόνο θετικοί αριθμοί. Επίσης, όπως φαίνεται και από τον παραπάνω πίνακα ο αλγόριθμος του Dijkstra έχει την υψηλότερη ταχύτητα και αποτελεί τον ιδανικό αλγόριθμο για την εφαρμογή αυτή. Ο αλγόριθμος Dijkstra είναι ο γρηγορότερος και χρησιμοποιεί τους λιγότερους πόρους για την εύρεση της συντομότερης διαδρομής, καθώς επίσης έχει μικρή πολυπλοκότητα. Ποια είναι η βάση δεδομένων, πώς χρησιμοποιήθηκε ο αλγόριθμος του Dijkstra στην εφαρμογή, ο κώδικας καθώς και τον τρόπο εύρεσης της ελάχιστης απόστασης μεταξύ αρχικής τοποθεσίας χρήστη και προορισμού θα αναλυθεί στο κεφάλαιο 5: «Υλοποίηση Εφαρμογής ECE Indoor Navigation».

Κεφάλαιο 3

Ανάλυση του Λειτουργικού Συστήματος Android OS

3.1. Εισαγωγή

Μια κινητή πλατφόρμα είναι μια φορητή συσκευή, ένα κινητό τηλέφωνο ή ένα smartphone, στην οποία ένας χρήστης μπορεί να εκτελέσει διάφορες λειτουργίες όπως την κυψελοειδή δικτύωση και την πρόσβαση στο Διαδίκτυο. Υπάρχουν όλο και περισσότερες νέες λειτουργίες που προστίθενται στην τρέχουσα παραγωγή της κινητής συσκευής για να αυξήσουν τη λειτουργικότητα και την παραγωγικότητά τους. Ένα από τα λειτουργικά συστήματα τα οποία αναπτύχθηκαν είναι και το Android OS. Αποτελεί ένα σύστημα ιδανικό για συσκευές οι οποίες λειτουργούν με τη χρήση μπαταρίας και δομούνται κυρίως από υλικοτεχνικό εξοπλισμό. Τέτοιες συσκευές είναι τα κινητά τερματικά (smartphones, tablets) αλλά και συσκευές εντοπισμού θέσης (GPS). Όπως όλα τα λειτουργικά συστήματα, έτσι και το Android δίνει τη δυνατότητα στις διάφορες εφαρμογές που φιλοξενεί να επικοινωνούν και να χειρίζονται τα χαρακτηριστικά της εκάστοτε συσκευής.

Η εταιρεία Android Inc. ιδρύθηκε στο Palo Alto της California, τον Οκτώβριο του 2003 από τους Andy Rubin, Rich Miner, Nick Sears και Chris White. Παρά τις προφανείς προηγούμενες ολοκληρώσεις των ιδρυτών και των πρώτων υπαλλήλων, η Android Inc. λειτούργησε κρυφά, αποκαλύπτοντας μόνο ότι λειτουργούσε σε λογισμικό για τα κινητά τηλέφωνα. Εκείνο το έτος, Rubin ξέμεινε από χρήματα. Η Google απέκτησε την Android Inc τον Αύγουστο του 2005, κάνοντάς τη θυγατρική της Google Inc, με σχεδόν όλους τους ιδρυτές της να παραμένουν στην εταιρεία μετά την ένωση αυτή. Αρχικός στόχος υπήρξε η ανάπτυξη εξυπνότερων κινητών συσκευών που γνώριζαν την τοποθεσία του χρήστη τους και τις προτιμήσεις του.

Σχήμα 33. Λογότυπο πλατφόρμας Android

3.2. Χαρακτηριστικά και λειτουργίες

Το Android είναι ένα λειτουργικό σύστημα έτσι σχεδιασμένο, ώστε να προσφέρει τη μέγιστη παραμετροποίηση στα κινητά τηλέφωνα, ανάλογα με τις επιθυμίες του χρήστη. Προσφέρει στους καταναλωτές τη δυνατότητα να χρησιμοποιήσουν εξελιγμένες υπηρεσίες του Διαδικτύου, όπως ανταλλαγή μηνυμάτων ηλεκτρονικού ταχυδρομείου, πρόσβαση σε σελίδες κοινωνικής δικτύωσης και φυσικά πλοήγηση στον Παγκόσμιο Ιστό, χωρίς να αμελεί τον τομέα ψυχαγωγία. Επίσης, υπόσχεται πληθώρα εφαρμογών που θα βελτιώσουν την εμπειρία της αναπαραγωγής μουσικής, της διαχείρισης αρχείων, της λήψης φωτογραφιών και βίντεο αλλά και της επικοινωνίας μέσω SMS, MMS και φωνητικών κλήσεων. Αρχικά αναπτύχθηκε από την Google και αργότερα από την Open Handset Alliance. Επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με την χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας την συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google.

Το Android βασίζεται στον πυρήνα του λειτουργικού συστήματος Linux. Αρχικά αναπτύχθηκε από την Google και πλέον αναπτύσσεται από την Open Handset Alliance (OHA), μια κοινοπραξία 48 τηλεπικοινωνιακών εταιριών, εταιριών λογισμικού και κατασκευής hardware, οι οποίες είναι αφιερωμένες στην ανάπτυξη και εξέλιξη ανοιχτών προτύπων στις συσκευές κινητής τηλεφωνίας, όπως η Google, η Motorola, η HTC, η Samsung, η LG, η Qualcomm, η Sprint, η T-Mobile κ.α. οι οποίες το Νοέμβρη του 2007 έφεραν στο φως το νέο αυτό λειτουργικό σύστημα. Η Google δημοσίευσε το μεγαλύτερο μέρος του κώδικα του Android υπό τους όρους της Apache License, μιας ελεύθερης άδειας λογισμικού, γεγονός που επιτρέπει σε πλήθος προγραμματιστών να σχεδιάσουν εφαρμογές για το Android με τη χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας την συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google. Οι εφαρμογές αυτές μπορούν να μεταφορτωθούν στα κινητά τηλέφωνα από τον δικτυακό τόπο Android Market.

Στη Google, η ομάδα που οδηγήθηκε από τον Rubin ανέπτυξε μια κινητή πλατφόρμα συσκευών που τροφοδοτήθηκε από τον πυρήνα Linux. Η Google εμπορεύτηκε την πλατφόρμα στους κατασκευαστές και τους μεταφορείς μικροτηλεφώνων με την υπόσχεση της παροχής ενός εύκαμπτου, βελτιώσιμου συστήματος. Η Google είχε παρατάξει μια σειρά υλικών τμημάτων και συνεργατών λογισμικού και είχε επισημάνει στους μεταφορείς ότι ήταν ανοικτό στους διάφορους βαθμούς συνεργασίας.

Στον παρακάτω πίνακα φαίνονται τα γενικά στοιχεία του λειτουργικού συστήματος Android:

Programmed in	C (core), Java (UI), C++
OS family	Linux
Working state	Current
Source model	Open Source
Initial release	September 20, 2008
Latest stable release	4.0.4 (Ice Cream Sandwich)
Package manager	Google Play / APK
Supported platforms	ARM, MIPS, x86
Kernel type	Monolithic (modified Linux kernel)
Default user interface	Graphical
License	Apache License 2.0 Linux kernel patches under GNU GPL v2
Official website	www.android.com

Πίνακας 7. Πληροφορίες του Android OS

Η βασική διαφορά του, από τα άλλα λειτουργικά συστήματα που ανέπτυξαν διάφορες εταιρείες για παρόμοιους σκοπούς, είναι πως το Android βασίζεται εξ' ολοκλήρου στη χρήση Java. Έτσι, για την ανάπτυξη μιας Android εφαρμογής γίνεται, εκτός από τη χρήση Java, και η χρήση ενός εικονικού προσομοιωτή. Για το λόγο αυτό, για την ανάπτυξη του παρέχεται μια εικονική μηχανή (Dalvik) η οποία εκτελεί τον δικό της κώδικα byte, καθώς και μια βιβλιοθήκη συναρτήσεων και εργαλείων που ονομάζεται Android SDK. Τα βασικά χαρακτηριστικά και οι κυριότερες λειτουργίες του Android είναι τα εξής:

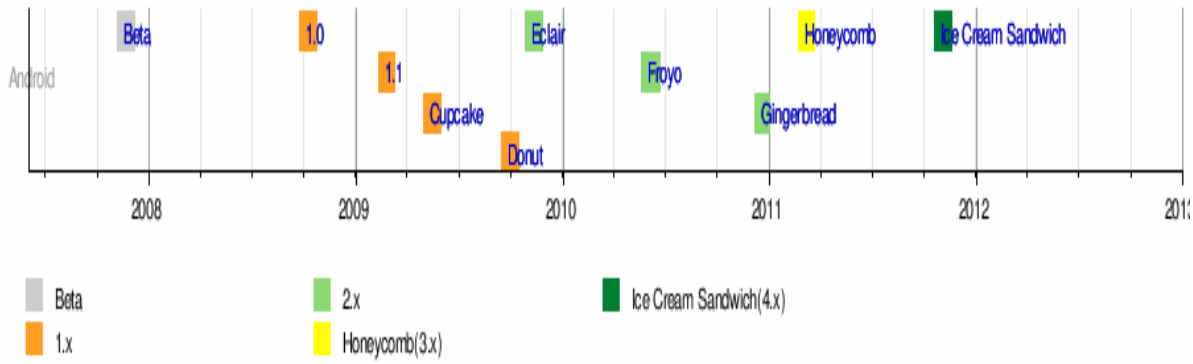
Λειτουργίες Οθόνης	Η πλατφόρμα είναι προσαρμόσιμη σε μεγαλύτερη ανάλυση (VGA), δισδιάστατες ψηφιακές γραφικές βιβλιοθήκες, τρισδιάστατα γραφικά βασισμένα στην OpenGL ES 1.0 έκδοση χαρακτηριστικών, καθώς και παραδοσιακές απεικονίσεις οθόνης "έξυπνων" συσκευών κινητής τηλεφωνίας.
Αποθήκευση Δεδομένων	Χρήση βάσης δεδομένων SQLite για τις ανάγκες αποθήκευσης
Συνδεσιμότητα	Το Android υποστηρίζει τεχνολογίες συνδεσιμότητας συμπεριλαμβανομένου GSM/EDGE, CDMA, EV-DO, UMTS,Bluetooth, και Wi-Fi.
Αποστολή μηνυμάτων	SMS και MMS είναι οι διαθέσιμοι τρόποι ανταλλαγής μηνυμάτων.
Περιήγηση στον Ιστό	Για την περιήγηση στον ιστό το Android διαθέτει ένα φυλλομετρητή βασισμένο στην ανοιχτή τεχνολογία WebKit.
Υποστήριξη Java	Λογισμικό γραμμένο στην Java είναι δυνατόν να μεταγλωττιστεί και να εκτελεστεί στην εικονική μηχανή Dalvik, η οποία είναι μια εξειδικευμένη υλοποίηση εικονική μηχανής, σχεδιασμένη για χρήση σε φορητές συσκευές, παρόλο που δεν είναι μια πρότυπη εικονική μηχανή Java.
Υποστήριξη Πολυμέσων	Το λειτουργικό Android υποστηρίζει τις ακόλουθες μορφές ήχου, στατικής και κινούμενης εικόνας: H.263, H.264 (σε 3GPή MP4 container), MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, MP3, MIDI, OGG Vorbis, WAV, JPEG, PNG, GIF,BMP.
Επιπλέον υποστήριξη hardware	Το λειτουργικό Android μπορεί να συνεργαστεί με κάμερες στατικής ή κινούμενης εικόνας, οθόνες αφής, GPS, αισθητήρες επιτάχυνσης, μαγνητόμετρα, δισδιάστατους καθώς και τρισδιάστατους επιταχυντές γραφικών.
Περιβάλλον Ανάπτυξης Λογισμικού	Περιλαμβάνει ένας προσομοιωτή συσκευής, εργαλεία για διόρθωση σφαλμάτων, μνήμη και εργαλεία ανάλυσης της απόδοσης του εκτελέσιμου λογισμικού καθώς και ένα επιπρόσθετο για το Eclipse IDE.

Αγορά και Εγκατάσταση Εφαρμογών	Παρόμοια με το App Store του iPhone OS, το Android Market είναι ένας κατάλογος εφαρμογών που μπορούν να μεταφορτωθούν και εγκατασταθούν στην συσκευή άμεσα μέσω ασύρματων καναλιών, χωρίς την χρήση υπολογιστή. Αρχικά μόνο δωρεάν εφαρμογές ήταν δυνατόν να εγκατασταθούν. Εφαρμογές επί πληρωμή ήταν μετέπειτα διαθέσιμες στο Android Market στις ΗΠΑ ύστερα από τις 19 Φεβρουαρίου 2009.
Οθόνη Αφής Πολλαπλών Σημείων	Το λειτουργικό Android είχε εξ ορισμού υποστήριξη για οθόνες πολλαπλών σημείων αλλά η δυνατότητα αυτή έχει κλειδωθεί σε επίπεδο πυρήνα (πιθανόν για αποφυγή παραβιάσεων των πατεντών λογισμικού της Apple στις τεχνολογίες οθονών αφής). Κυκλοφορεί μια ανεπίσημη τροποποίηση (mod) που έχει αναπτυχθεί για να υποστηρίζει πολλαπλή επαφή (multi-touch), αλλά απαιτεί δικαιώματα πρόσβασης υπερχρήστη (superuser) στη συσκευή για να γραφεί στη μνήμη flash ένας πυρήνας που να μην είναι υπογεγραμμένος (unsigned kernel).

Πίνακας 8. Χαρακτηριστικά και λειτουργίες του Android OS

3.3. Εκδόσεις

Η ιστορία έκδοσης του λειτουργικού συστήματος Android άρχισε με την έκδοση Android beta τον Νοέμβριο του 2007. Η πρώτη εμπορική έκδοση, Android 1.0, βγήκε τον Σεπτέμβριο του 2008. Το Android είναι ένα λειτουργικό σύστημα κινητού που αναπτύχθηκε από τη Google και την Open Handset Alliance, και έχουν υπάρξει πολλές ενημερώσεις στο λειτουργικό σύστημα βάσεών της από την αρχική της έκδοση. Αυτές οι αναπροσαρμογές διορθώνουν τα σφάλματα και προσθέτουν νέα χαρακτηριστικά γνωρίσματα. Από τον Απρίλιο του 2009, κάθε Android έκδοση έχει αναπτυχθεί κάτω από ένα όνομα κώδικα βασισμένο σε ένα επιδόρπιο ή γλυκό. Αυτές οι εκδόσεις που έχουν κυκλοφορήσει κατά χρονολογική και αλφαβητική σειρά: Cupcake, Donut, Eclair, Froyo (παγωμένο γιαούρτι), Gingerbread (μελόψωμο), Honeycomb (κηρήθρα) και Ice Cream Sandwich (σάντουιτς παγωτό). Η πιο πρόσφατη έκδοση του Android OS είναι το Ice Cream Sandwich v4.0.4, τον Μάρτιο του 2012.



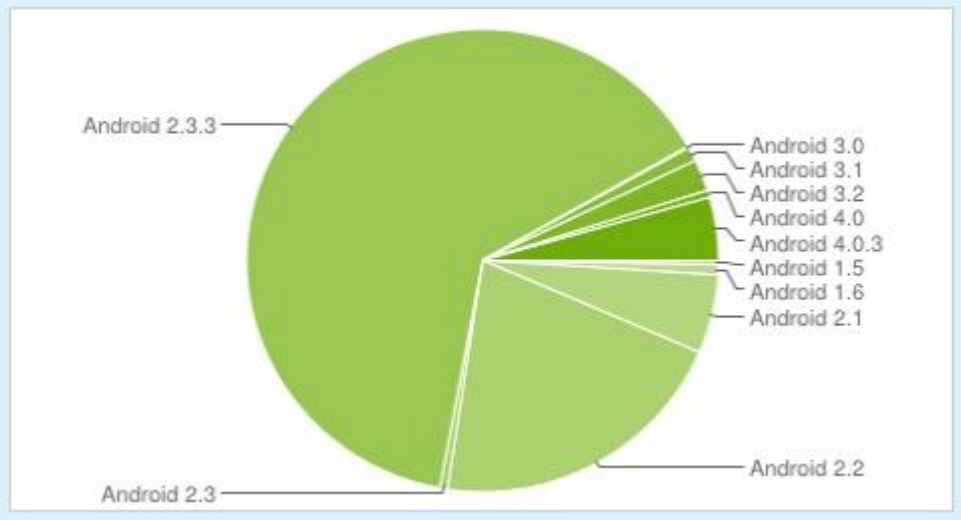
Σχήμα 34. Οι εκδόσεις του Android OS ως προς το χρόνο

Το πλαίσιο API, που παρέχει μια πλατφόρμα Android, εξειδικεύεται χρησιμοποιώντας έναν ακέραιο αριθμό αναγνώρισης, γνωστό και ως API Level. Έτσι, το επίπεδο API είναι ένας ακέραιος αριθμός που αναγνωρίζει μοναδικά το framework API που προσφέρεται από μια έκδοση της πλατφόρμας Android. Πρέπει να γίνει κατανοητό ότι η αναγνώριση του API Level παίζει σημαντικό ρόλο στο να διασφαλιστεί η συμβατότητα της εφαρμογής με τις συσκευές στις οποίες θα εγκατασταθεί. Κάθε διαδοχική έκδοση της πλατφόρμας Android υποστηρίζει ακριβώς ένα τέτοιο Level και μπορεί να περιλαμβάνει ενημερώσεις στο πλαίσιο εφαρμογών API της Android. Καθώς τα μέρη της API ενημερώνονται, τα παλαιότερα που αντικαθίστανται, αν και δε χρησιμοποιούνται πλέον, δεν απομακρύνονται τελείως. Οι ενημερώσεις αυτές είναι έτσι σχεδιασμένες ώστε το νέο επίπεδο API να παραμένει συμβατό με τις παλαιότερες εκδόσεις. Έτσι, οι περισσότερες αλλαγές είναι προσθετικές και εισάγουν μια νέα ή αντικαθιστούν μια λειτουργικότητα.

Στον πίνακα 9 παρουσιάζονται οι εκδόσεις του λειτουργικού συστήματος Android με τα αντίστοιχα API Level μαζί με την ονομασία της έκδοσης αλλά και το ποσοστό των χρηστών που χρησιμοποιούν την κάθε έκδοση.

Platform Version	**Codename**	**API Level**	**Distribution**
Android 1.0	Base	1	0%
Android 1.1	Base 1.1	2	0%
Android 1.5	Cupcake	3	0.3%
Android 1.6	Donut	4	0.7%
Android 2.1	Eclair	7	5.5%
Android 2.2	Froyo	8	20.9%
Android 2.3 - Android 2.3.2	Gingerbread	9	0.5%
Android 2.3.3 - Android 2.3.7		10	63.9%
Android 3.0	Honeycomb	11	0.1%
Android 3.1		12	1.0%
Android 3.2		13	2.2%
Android 4.0 - Android 4.0.2	Ice Cream Sandwich	14	0.5%
Android 4.0.3 - Android 4.0.4		15	4.4%

Πίνακας 9. Οι εκδόσεις του Android OS



Σχήμα 35. Ποσοστά χρησιμοποίησης κάθε έκδοσης

Μερικά στοιχεία για την κάθε έκδοση παρατίθενται παρακάτω:

- Η έκδοση Cupcake εισήγαγε κάποια καινούργια χαρακτηριστικά και ανανεώσεις στην διεπιφάνεια χρήστη (User Interface):
  - Ικανότητα για καταγραφή και παρακολούθηση βίντεο μέσα από την λειτουργία της βιντεοκάμερας, μεταφόρτωση βίντεο στο YouTube και φωτογραφιών στο Picasa απευθείας από το τηλέφωνο, καινούργιο μαλακό πληκτρολόγιο (αφής) με πρόβλεψη κειμένου
  - Υποστήριξη προτύπου Bluetooth A2DP και AVRCP
  - Ικανότητα αυτόματης σύνδεσης σε μικροσυσκευή Bluetooth από μια συγκεκριμένη απόσταση
  - Καινούργια widgets και φάκελοι που μπορούν να δημοσιευτούν στην αρχική οθόνη
  - Κινούμενες μεταβάσεις οθόνης

Σχήμα 36. Λογότυπο Android 1.5 CUPCAKE

- Η έκδοση Donut ήρθε τον Σεπτέμβριο του 2009 και περιλάμβανε:

    - Βελτιωμένο Android Market
    - Ενσωματωμένη φωτογραφική μηχανή, βιντεοκάμερα και διεπαφή (interface) γκαλερί
    - Η γκαλερί επιτρέπει πλέον στους χρήστες την επιλογή πολλαπλών φωτογραφιών για διαγραφή
    - Ανανεωμένη αναζήτηση με φωνή, με ταχύτερη απόκριση και βαθύτερη ολοκλήρωση με εγγενείς (native) εφαρμογές, συμπεριλαμβανομένης της δυνατότητας να καλούνται επαφές
    - Ανανεωμένη αναζήτηση με την δυνατότητα αναζήτησης σελιδοδεικτών, ιστορικού, επαφών και στο διαδίκτυο από την αρχική οθόνη
    - Ανανεωμένη υποστήριξη τεχνολογιών για CDMA/EVDO, 802.1x, VPNs και με μηχανή μετατροπής κειμένου σε ομιλία (text-to-speech)
    - Υποστήριξη για ανάλυση οθονών WVGA
    - Βελτιώσεις στην ταχύτητα για αναζήτηση και για εφαρμογές φωτογραφικής μηχανής

Σχήμα 37. Λογότυπο Android 1.6 DONUT

- Η έκδοση Éclair, Android 2.0 εμφανίστηκε τον Νοέμβριο 2009, με τις επανεκδόσεις του σε Android 2.0.1 τον Δεκέμβριο 2009 (Eclair 0.1) και τον Ιανουάριο 2010 με το Android 2.1 (Éclair MR1). Μερικές αλλαγές ανάμεσα από την προηγούμενη έκδοση είναι:

    - Βέλτιστη ταχύτητα υλικού
    - Υποστήριξη για περισσότερες οθόνες και αναλύσεις

- Βελτιωμένη διεπιφάνεια χρήστη
- Καινούργια διεπιφάνεια χρήσης για την μηχανή αναζήτησης και υποστήριξη του προτύπου HTML5
- Καινούργιες λίστες επαφών
- Καλύτερος λόγος άσπρου – μαύρου για φόντα
- Βελτιωμένοι χάρτες Google (google maps) 3.1.2
- Υποστήριξη Microsoft Exchange
- Ενσωματωμένη υποστήριξη flash για την Camera
- Ψηφιακή μεγέθυνση (zoom)
- Κλάση MotionEvent βελτιωμένη ώστε οι κατασκευαστές να μπορούν να παρακολουθούν αποτελεσματικότερα τα γεγονότα πολλαπλής αφής
- Ανανεωμένο εικονικό πληκτρολόγιο
- Bluetooth 2.1

Σχήμα 38. Λογότυπο Android 2.0 ECLAIR

- Η έκδοση Froyo, ήρθε τον Μάιο του 2010 και ανάμεσα στις αλλαγές περιλάμβανε:
  - Βελτιστοποιήσεις στην ταχύτητα γενικά του λειτουργικού συστήματος, στην μνήμη και στην απόδοση
  - Ενσωμάτωση στην μηχανή αναζήτησης, της μηχανής Javascript του Chrome V8
  - Αυξημένη υποστήριξη Microsoft Exchange (σε πολιτικές ασφαλείας, συγχρονισμού ημερολογίου, auto – discovery, GAL look-up, remote wipe)
  - Βελτιωμένος προωθητής εφαρμογής (application launcher), με συντομεύσεις προς τις εφαρμογές τηλεφώνου και εφαρμογές της Μηχανής Αναζήτησης

- Πρόσδεση USB και λειτουργία δυναμικής ζώνης (hotspot) WiFi
- Ανανεωμένη εφαρμογή Αγοράς (Market) με αυτόματη ανανέωση
- Επιλογή για απαγόρευση πρόσβασης δεδομένων πάνω από ένα δίκτυο κινητής τηλεφωνίας
- Γρήγορη εναλλαγή ανάμεσα σε πολλαπλές γλώσσες του πληκτρολογίου και των λεξικών τους
- Φωνητική κλήση και διαμοιρασμός επαφών με Bluetooth
- Υποστήριξη για αριθμητικούς και αλφαριθμητικούς κωδικούς
- Η μηχανή αναζήτησης μπορεί να αποτυπώσει κινούμενα GIFs
- Υποστήριξη για πεδία μεταφόρτωσης αρχείων στην μηχανή αναζήτησης
- Υποστήριξη για εγκατάσταση εφαρμογών στην επεκτάσιμη μνήμη
- Υποστήριξη Adobe Flash 10.1

Σχήμα 39. Λογότυπο Android 2.2 FROYO

- Η έκδοση Gingerbread παρουσιάστηκε τον Δεκέμβριο του 2010 και η έκδοση αυτή περιλάμβανε:
  - Βελτιωμένο UI για απλότητα και ταχύτητα
  - Πιο γρήγορη, πιο διαισθητική εισαγωγή κειμένου
  - Επιλογή λέξεων και αντιγραφή/επικόλληση με ένα άγγιγμα
  - Βελτιωμένη ενεργειακή διαχείριση
  - Υποστήριξη NFC (Near Field Communication)
  - Υποστήριξη video κλήσης
  - Υποστήριξη του πρωτόκολλου WebM για αναπαραγωγή video

Σχήμα 40. Λογότυπο Android 2.3 GINGERBREAD

- Η έκδοση Honeycomb είναι στη διάθεση των χρηστών και προγραμματιστών από τον Φεβρουάριο του 2011, λίγες μέρες μετά την επανέκδοση του Android 2.3.3, και προορίζεται αποκλειστικά για ταμπλέτες. Μερικά χαρακτηριστικά:

  - Υποστηρίζει διπύρηνους και τετραπύρηνους επεξεργαστές
  - Βελτιωμένη υποστήριξη των ταμπλετών ανάπτυξη λογισμικού (scripting) για 3D, σε γλώσσα η οποία καλείται "Renderscript"
  - Video chat μέσω Google Talk
  - Google eBooks
  - "Ιδιωτική περιήγηση"

Σχήμα 41. Λογότυπο Android 3.0 HONEYCOMB

- Η έκδοση Ice Cream Sandwich ήρθε στην αγορά στις 19 Οκτωβρίου του 2011 και περιλάμβανε:

    - Ξεκλείδωμα της συσκευής με αναγνώριση προσώπου
    - Αλλαγή μεγέθους των Widgets
    - Software πλήκτρα πλοήγησης
    - Νέα Live Wallpapers
    - Δημιουργία φακέλων που περιλαμβάνουν εφαρμογές ή επαφές
    - Εύκολη πλοήγηση στις τελευταίες ανοιχτές εφαρμογές και άμεσο κλείσιμο τους
    - Εύκολο τράβηγμα Screenshot οποιασδήποτε οθόνης
    - Ορθογράφος κατά την πληκτρολόγηση κειμένου
    - People app στα πρότυπα του People hub των Windows Phone όπου εμφανίζονται όλες οι διαθέσιμες πληροφορίες για κάθε επαφή
    - Ανανεωμένη εφαρμογή Gmail με 2 γραμμές προεπισκόπησης και offline αναζήτησης στα email των τελευταίων 30 ημερών
    - Πιο όμορφο σύστημα ειδοποιήσεων με προσθήκη φωτογραφιών και μεμονωμένο κλείσιμο τους

Σχήμα 42. Λογότυπο Android 4.0 ICE CREAM SANDWICH

Η Google αλλάζει τακτική στις εκδόσεις του λειτουργικού της συστήματος, δημιουργώντας ένα κοινό «μέτωπο» με τους κατασκευαστές smartphones και tablets. Σε μια προσπάθεια να σταματήσει τις εταιρείες κινητής τηλεφωνίας να έχουν τον έλεγχο των συσκευών, η εταιρεία σχεδιάζει να δώσει νωρίτερα πρόσβαση στους κατασκευαστές συσκευών σε νέες εκδόσεις του λειτουργικού και να πωλούνται κατευθείαν στους καταναλωτές. Θέλοντας να αλλάξει το καθεστώς που επικρατεί σήμερα, όπου πρώτα έκλεινε συμφωνία με έναν κατασκευαστή ο οποίος κυκλοφορούσε τη συσκευή αυτή με την νέα έκδοση του Android και μετά από ένα χρονικό διάστημα δίνονταν και στους υπόλοιπους κατασκευαστές.

Η κίνηση αυτή έχει σκοπό να δώσει στη Google επιπλέον έλεγχο επάνω στα βασικά χαρακτηριστικά και τις εφαρμογές που λειτουργούν στο Android, μειώνοντας τον έλεγχο που θέλουν να έχουν οι πάροχοι κινητής τηλεφωνίας στις συσκευές. Η Google θα συνεργαστεί με έως και πέντε κατασκευαστές ταυτόχρονα για την ανάπτυξη της νέας σειράς smartphone και tablet «Nexus», τα οποία και θα πωλούνται απευθείας στους καταναλωτές σε ΗΠΑ, Ασία και Ευρώπη μέσω της ιστοσελίδας της και ίσως από ορισμένους προμηθευτές λιανικής. Η Google δείχνει να μην πτοείται και συνεχίζει να βγάζει και νέες εκδόσεις με ονόματα γλυκών αυτή τη φορά. Οι νέες συσκευές θα χρησιμοποιούν την επόμενη έκδοση του Android 5.0, Jelly Bean, και η Google ευελπιστεί να έχει έτοιμη μια πλήρη γκάμα συσκευών προς διάθεση μέχρι την Ημέρα των Ευχαριστιών του 2012. Η Google σχεδιάζει μια αναβάθμιση του λειτουργικού συστήματος κάθε χρόνο οπότε ενδέχεται να παρουσιάσει την επόμενη αναβάθμιση στις αρχές του 2013, η οποία θα ονομάζεται Android 6.0, Key Lime Pie (λεμονόπιτα).

3.4. Android SDK

Η ανάπτυξη λογισμικού Android είναι η διαδικασία από την οποία οι νέες εφαρμογές δημιουργούνται για το λειτουργικό σύστημα του Android. Οι εφαρμογές αναπτύσσονται συνήθως στη γλώσσα Java αλλά και άλλα εργαλεία ανάπτυξης είναι διαθέσιμα. Από τον Απρίλιο του 2011 περισσότερες από 200.000 εφαρμογές έχουν αναπτυχθεί για το Android, με πάνω από 3 δισεκατομμύρια κατεβάσματα. Η Android πλατφόρμα αναπτύσσεται για να γίνει η αγαπημένη σε άτομα που είναι υπεύθυνα για την ανάπτυξη κινητών τηλεφώνων. Μια έρευνα του Ιουνίου 2011 έδειξε ότι πάνω από 67% των developers κινητών τηλεφώνων χρησιμοποίησε την πλατφόρμα, κατά την διάρκεια της δημοσίευσης.

Το πακέτο ανάπτυξης λογισμικού (Software Development Kit-SDK) περιλαμβάνει ένα περιεκτικό σύνολο εργαλείων ανάπτυξης. Αυτά περιλαμβάνουν έναν διορθωτή (debugger), βιβλιοθήκες, ένα προσομοιωτή κινητού τηλεφώνου βασισμένο στο QEMU, αρχεία, δείγματα κωδίκων και σεμινάρια. Το επίσημα υποστηριγμένο ενσωματωμένο περιβάλλον ανάπτυξης (Integrated Development Environment-IDE) είναι το Eclipse που χρησιμοποιεί τα Android εργαλεία ανάπτυξης (Android Development Tools-ADT) Plugin, αν και οι developers μπορούν να χρησιμοποιήσουν οποιοδήποτε επεξεργαστή κειμένων για να επεξεργαστούν την Java και τα αρχεία XML, και έπειτα χρησιμοποιούν τη γραμμή εντολής εργαλείων (το πακέτο ανάπτυξης Java (Java Development Kit-JDK) και το Apache Ant απαιτούνται) για να δημιουργήσουν, να χτίσουν και να διορθώσουν τις Android εφαρμογές καθώς επίσης και να ελέγχουν τις συσκευές Android.

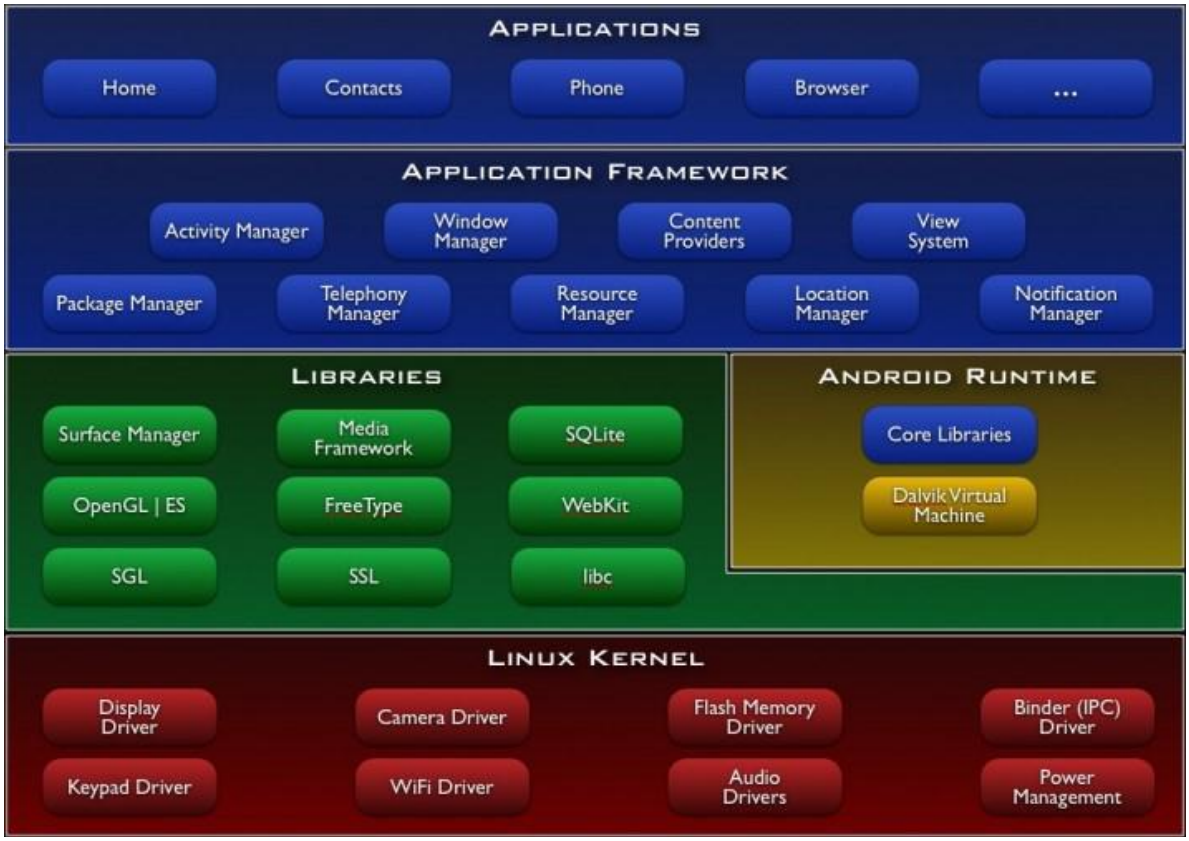
Οι βελτιώσεις στο Android SDK πηγαίνουν μαζί με την ανάπτυξη πλατφορμών Android. Το SDK υποστηρίζει επίσης τις παλαιότερες εκδόσεις της Android πλατφόρμας σε περίπτωση που οι developers επιθυμούν να στοχεύσουν σε εφαρμογές παλαιότερων κινητών. Τα

εργαλεία ανάπτυξης αποτελούν συστατικά τα οποία μπορούν να κατέβουν από το διαδίκτυο, συνεπώς αφότου έχει κατέβει η πιο πρόσφατη έκδοση και η πλατφόρμα, παλαιότερες πλατφόρμες και εργαλεία μπορούν επίσης να κατέβουν για τεστ συμβατότητας. Οι Android εφαρμογές συσκευάζονται σε .apk και αποθηκεύονται κάτω από το /data/app folder του Android OS (ο φάκελος είναι προσβάσιμος μόνο για τον χρήστη (root user) που δεν έχει πειράξει το λειτουργικό του συστήματος για λόγους ασφάλειας).

3.5. Αρχιτεκτονική

Η πλατφόρμα Android είναι μια ανοικτού συστήματος αρχιτεκτονική, με ευέλικτο περιβάλλον ανάπτυξης ενώ παράλληλα υποστηρίζει την επεκτασιμότητα της εμπειρίας χρήστη, με τα βελτιστοποιημένα γραφικά συστήματα που διαθέτει, την πλούσια υποστήριξη πολυμέσων και τον ισχυρό browser που έχει ενσωματωμένο. Επιτρέπει τη χρήση και την αντικατάσταση συστατικών του και υποστήριξη μιας αποτελεσματικής βάσης δεδομένων, ενώ ταυτόχρονα υποστηρίζει ποικίλα μέσα ασύρματων επικοινωνιών. Χρησιμοποιεί επίσης μια εικονική συσκευή, την Dalvik Virtual Machine (DVM), που είναι άριστα βελτιστοποιημένη για τις κινητές συσκευές.

Το Android είναι μια στοίβα λογισμικού. Η λογική πίσω από αυτήν την έκφραση και την όλη φιλοσοφία του Android, κρύβεται στο ακόλουθο διάγραμμα που παρουσιάζονται τα βασικά συστατικά του (Σχήμα 43).



Σχήμα 43. Τα βασικά περιεχόμενα του λειτουργικού συστήματος Android

Στην στοίβα του Android, παρατηρούνται 4 επίπεδα. Κάθε επίπεδο στην αρχιτεκτονική αυτή, χρησιμοποιεί τις υπηρεσίες που του προσφέρονται από τα πιο κάτω επίπεδα. Παρακάτω θα περιγραφούν αυτά τα επίπεδα ξεκινώντας από το πιο χαμηλό.

3.5.1. Πυρήνας Linux (Linux kernel)

Το Android είναι βασισμένο στα γερά θεμέλια του Linux. Ο πυρήνας Linux είναι δοκιμασμένος, σταθερός και πετυχημένος και μπορεί να βρεθεί παντού, από ρολόγια χειρός μέχρι υπερυπολογιστές. Το Linux παρέχει στο Android το αφαιρετικό επίπεδο υλικού, επιτρέποντάς του να μπορεί να χρησιμοποιηθεί σε μεγάλη ποικιλία πλατφόρμων στο μέλλον. Ειδικότερα, το Android χρησιμοποιεί τον πυρήνα Linux για την διαχείριση μνήμης, την διαχείριση διεργασιών, την δικτύωση και άλλες υπηρεσίες του λειτουργικού συστήματος.

3.5.2. Εγγενείς Βιβλιοθήκες – Native Libraries

Στο αμέσως ψηλότερο επίπεδο βρίσκονται οι Native Libraries – Εγγενείς Βιβλιοθήκες. Όλες αυτές είναι γραμμένες στην γλώσσα προγραμματισμού C και C++ και μεταγλωττίστηκαν για την συγκεκριμένη αρχιτεκτονική υλικού που χρησιμοποιείται από το τηλέφωνο. Οι βιβλιοθήκες αυτές δεν είναι εφαρμογές που μπορούν να σταθούν από μόνες τους. Υπάρχουν για να μπορούν να κληθούν από προγράμματα υψηλότερου επιπέδου. Από την έκδοση Donut και μετά, οι κατασκευαστές μπορούν να γράφουν τις δικές τους τέτοιες βιβλιοθήκες με την χρήση της Εργαλειοθήκης NDK (Native Development Kit).

3.5.3. Χρόνος Εκτέλεσης – Android Runtime

Στο ίδιο επίπεδο με τις εγγενείς βιβλιοθήκες, βρίσκεται και ο χρόνος εκτέλεσης Android. Εδώ ζουν βασικές βιβλιοθήκες της Java και η εικονική μηχανή Dalvik. Η Dalvik είναι μια βελτιστοποιημένη υλοποίηση μιας εικονικής μηχανής Java για φορητές συσκευές από την Google. Η Dalvik τρέχει .dex αρχεία, τα οποία είναι bytecodes που προέρχονται από αρχεία .class και .jar. Εν αντιθέσει όμως με τα .class αρχεία, τα .dex είναι πολύ πιο συμπαγή και αποδοτικά, γεγονός σημαντικό για συσκευές με περιορισμένη μνήμη και μπαταρία. Το Android περιλαμβάνει ένα σύνολο βασικών βιβλιοθηκών που παρέχουν τις περισσότερες από τις διαθέσιμες λειτουργίες των βασικών βιβλιοθηκών της Java. Κάποια πακέτα και κλάσεις υπάρχουν και στο Android κάποια άλλα δεν υποστηρίζονται καθόλου, ενώ ταυτόχρονα το Android παρέχει και επιπρόσθετα προσαρμοσμένα στις δικές του ανάγκες.

3.5.4. Πλαίσιο Υποστήριξης Ανάπτυξης Εφαρμογών – Application Framework

Πάνω από τις εγγενείς βιβλιοθήκες και το χρόνο εκτέλεσης Android, είναι το πλαίσιο υποστήριξης ανάπτυξης εφαρμογών. Στο επίπεδο αυτό παρέχονται υψηλού επιπέδου δομικές μονάδες οι οποίες μπορούν να χρησιμοποιηθούν για την κατασκευή των εφαρμογών. Το πλαίσιο υποστήριξης ανάπτυξης εφαρμογών είναι προ-εγκατεστημένο στο

Android, αλλά είναι επεκτάσιμο, αφού ο κάθε κατασκευαστής μπορεί να το συμπληρώσει με δικά του κομμάτια. Τα σημαντικότερα δομικά στοιχεία του πλαισίου αυτού είναι:

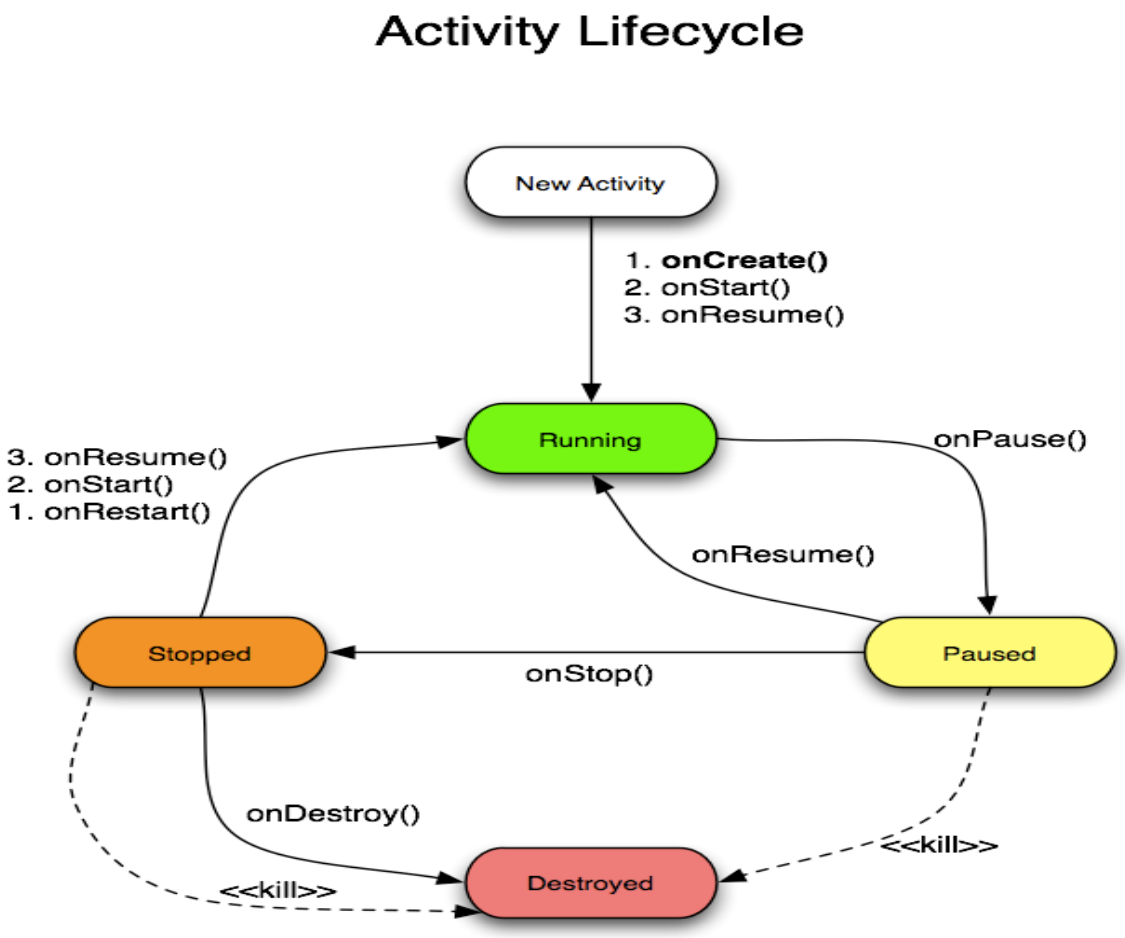
- Διαχειριστής Δραστηριοτήτων - Activity Manager: Υπεύθυνος για τον έλεγχο του χρόνου ζωής (Σχήμα 44) των εφαρμογών και για την διατήρηση μιας στοίβας που επιτρέπει την πλοήγηση του χρήστη σε προηγούμενες οθόνες.

- Παροχέας Περιεχομένου - Content Providers: Αυτά τα αντικείμενα περιέχουν δεδομένα που μπορούν να διαμοιραστούν μεταξύ εφαρμογών.

- Διαχειριστής Πόρων - Resource Manager: Οι πόροι, είναι οτιδήποτε υπάρχει σε ένα πρόγραμμα και δεν είναι κώδικας. Για παράδειγμα μπορεί να είναι κωδικοί χρωμάτων, αλφαριθμητικοί χαρακτήρες ή ακόμα και έτοιμα σχεδιαγράμματα οθονών φτιαγμένα σε XML, τα οποία μπορεί το πρόγραμμα να καλεί.

- Διαχειριστής Τοποθεσίας - Location Manager: Χρησιμοποιείται για να μπορεί να ξέρει το τηλέφωνο που βρίσκεται ανά πάσα στιγμή.

- Διαχειριστής Κοινοποιήσεων - Notification Manager: Ιδανικός τρόπος για να ενημερώνεις τον χρήστη για γεγονότα που συμβαίνουν, διακριτικά χωρίς να διακόπτεις την εργασία του.



Σχήμα 44. Ο κύκλος ζωής μιας δραστηριότητας (Activity) Android

3.5.5. Εφαρμογές– Applications

Ένα set από εφαρμογές βρίσκονται στο υψηλότερο επίπεδο της αρχιτεκτονικής του Android λογισμικού και περιλαμβάνει email client, SMS/MMS εφαρμογή, ημερολόγιο, έναν web browser, χάρτες και εφαρμογές επί αυτών, επαφές (contacts) κ. α. Όλες οι εφαρμογές είναι γραμμένες σε γλώσσα Java.

3.6. Μερίδιο Αγοράς

Πριν από 3 χρόνια κανείς δεν μπορούσε να φανταστεί ότι η αγορά των Android smartphones θα μπορούσε όχι μόνο να απειλήσει την Nokia και την Apple αλλά να κυριεύσει τον κόσμο των κινητών. Η επιτυχία της Google είναι αξιοσημείωτη. Το κόλπο για να το πετύχει αυτό είναι πως:

1) είναι δωρεάν.

2) είναι ανοιχτό σε όλους.

3) επιτρέπει στην κάθε εταιρία κινητών τηλεφώνων να προσφέρει τα δικά της γραφικά και ιδιαίτερες ρυθμίσεις.

4) ακολουθεί την ίδια λογική με τις επιπλέον εφαρμογές που μπορείς να κατεβάσεις που ακολουθεί και το AppleStore και γενικά είναι σχετικά εμφανές πως προσπαθεί να χτυπήσει την Apple στο πόσο «cool» είναι να έχεις το δικό σου smartphone.

5) ταυτόχρονα, με το να έχει ένα σωρό "σοβαρές" εφαρμογές, προσπαθεί να χτυπήσει τη RIM με το blackberry, μια εταιρία που απευθύνεται σε κοινό που βλέπει το κινητό περισσότερο σαν εργαλείο επαγγελματικού κι όχι κοινωνικού πρεστίζ.

Παρακάτω θα περιγραφεί η πορεία της στο παγκόσμιο μερίδιο αγοράς:

- Την δεύτερη περίοδο (Q2) του 2009 το Android είχε μερίδιο 2,8% στις παγκόσμιες πωλήσεις smartphone.

- Τον Φεβρουάριο του 2010, η πλατφόρμα Android κατείχε το 9,0% της Αμερικάνικης αγοράς smartphone, όπως μετριέται από τους τρέχοντες κινητούς συνδρομητές. Αυτός ο αριθμός ήταν πιο πάνω από μια προηγούμενη εκτίμηση (5,2%) τον Νοέμβριο του 2009.

- Στο Q2 του 2010, το iOS της Apple ήταν πιο πάνω κατά 11%, δείχνοντας ότι το Android παίρνει το μερίδιο αγοράς κυρίως από τη RIM, και πρέπει ακόμα να ανταγωνιστεί τους καταναλωτές με υψηλές απαιτήσεις για τις νέες προσφορές ανταγωνιστών. Επιπλέον, οι αναλυτές επικεντρώθηκαν στα πλεονεκτήματα ότι το Android έχει ένα πολυδιαυλικό, πολυ-μεταφερόμενο λειτουργικό σύστημα (Operating System - OS).

- Μέχρι το τέλος Q3 του 2010, το μερίδιο αγοράς στις ΗΠΑ είχε αυξηθεί στο 21.4%. Μάλιστα, τον Μάιο του 2010, οι πωλήσεις του Android στην Αμερική ξεπέρασαν αυτή της αντίπαλης πλατφόρμας iPhone.

- Στο Q4 του 2010, το μερίδιο είχε αυξηθεί στο 33% της αγοράς και γίνεται έτσι η κορυφαία πλατφόρμα smartphone σε πωλήσεις, έχοντας το Android το 59% της συνολικής εγκατεστημένης βάσης χρηστών iOS της Apple στις ΗΠΑ και το 46% της συνολικής εγκατεστημένης βάσης χρηστών iOS στην Ευρώπη.

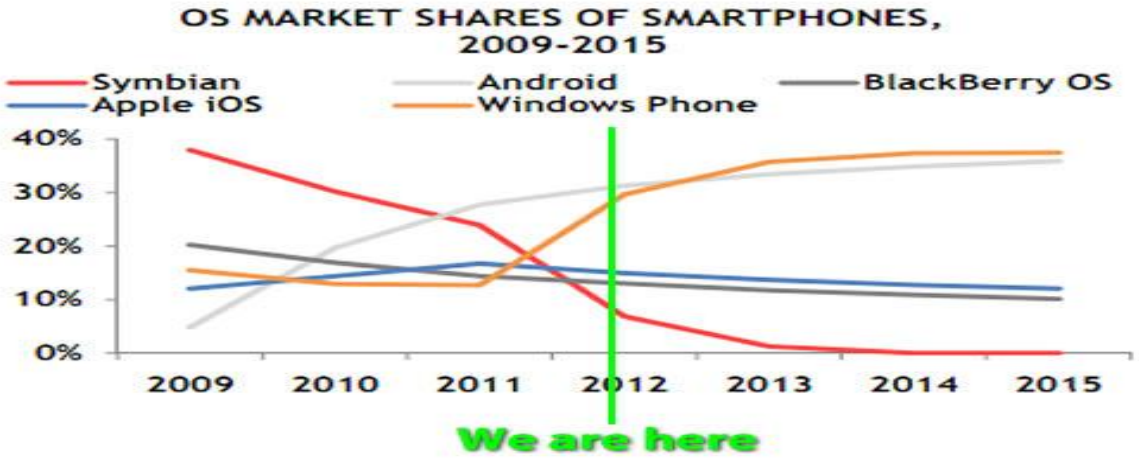
- Στο Q1 του 2011, το λειτουργικό σύστημα που χρησιμοποιούν τα smartphones φέρνει το Android στην 1η θέση με εκρηκτική άνοδο 26% σε σχέση με το περσινό τρίμηνο, το Symbian της Nokia στη δεύτερη, το iOS της Apple στην τρίτη και το λογισμικό των BlackBerry στην 4η θέση.

- Στο Q2 του 2011, το Android ήταν πρώτο σε 35 από τις 56 χώρες ενώ στο παγκόσμιο μερίδιο το Android έχει το 48% χάρη σε αγορές εκτός ΗΠΑ. Δεύτερο το Apple iOS με 19% στο Q2, ξεπέρασε τη Nokia και το λειτουργικό της (Symbian).

- Στο Q3 του 2011, το Android κατατάσσεται ως η κορυφαία smartphone πλατφόρμα, με μερίδιο αγοράς 52,5%, αυξημένο κατά 4,6% από την προηγούμενη τρίμηνη περίοδο. Το Symbian έπεσε 16,9% και το iOS της Apple είναι στην τρίτη θέση με μερίδιο 15%. Η RIM - συνεχίζοντας την πτώση - κατέχει την τέταρτη θέση με μερίδιο 11%, ακολουθούμενη από την Microsoft με 1,5%.

- Στο Q4 του 2011, το Android της Google παραμένει στην κορυφή με ποσοστό 50,9% των συνολικών πωλήσεων του τριμήνου, αν και παρουσιάζει μια μικρή κάμψη σε σύγκριση με το αμέσως προηγούμενο τρίμηνο, προφανώς λόγω της κυκλοφορίας του iPhone 4S. Το iOS βρίσκεται στη δεύτερη θέση με ποσοστό 23,8%, ενώ το Symbian κατρακύλησε στην τρίτη θέση, με ποσοστό μόλις 11,7%

- Στο Q1 του 2012, το Android κρατάει τα σκήπτρα με 56,1% παγκοσμίως έναντι 22,9% της iOS. Το Symbian με 8,6% και η RIM με πωλήσεις 9,9 εκατομμυρίων μονάδων στο ίδιο διάστημα και ποσοστό της τάξεως του 2,4% δε φαίνεται να έχουν καμιά τύχη μπροστά στους δύο μεγάλους.



Σχήμα 45. Πορεία των λειτουργικών συστημάτων (2009-2015)

Το Android ενεργοποιούσε 300.000 ανά ημέρα πριν τον Δεκέμβριο του 2010. Μέχρι τις 14 Ιουλίου του 2011, 550.000 Android συσκευές ενεργοποιούνταν από την Google κάθε ημέρα, με αύξηση 4,4% την εβδομάδα. Την 1η Αυγούστου του 2011, υπολογίστηκε ότι το android είχε περίπου 48% του μεριδίου αγοράς smartphone. Στις 13 Οκτωβρίου του 2011, η Google ανήγγειλε ότι υπήρξαν 190 εκατομμύριο Android συσκευές στην αγορά. Από τις 16 Νοεμβρίου του 2011, κατά τη διάρκεια της ανακοίνωσης της Google μουσικής "These Go to Eleven", 200 εκατομμύρια Android συσκευές είχαν ενεργοποιηθεί. Με βάση αυτόν τον αριθμό, με 1,9% των Android συσκευών να είναι ταμπλέτες, περίπου 3,8 εκατομμύριο Android Honeycomb ταμπλέτες έχουν πωληθεί. Στις 20 Δεκεμβρίου του 2011, Andy Rubin ανήγγειλε ότι Google ενεργοποιούσε 700.000 νέες Android συσκευές καθημερινά. Δύο μήνες αργότερα, στις 27 Φεβρουαρίου του 2012, Andy Rubin ανήγγειλε ότι Google ενεργοποιούσε πάνω από 850.000 Android smartphones και ταμπλέτες καθημερινά.

- ***<u>Πρόβλεψη για το έτος 2015</u>***

  - Τα πράγματα, σύμφωνα με την IDC, θα αλλάξουν ριζικά το 2015 από τη δεύτερη θέση και κάτω. Στην πρώτη θα παραμείνει το Android, το οποίο θα έχει εξελιχθεί στην πιο δημοφιλή πλατφόρμα με ένα μερίδιο της τάξεως του 45,4%.

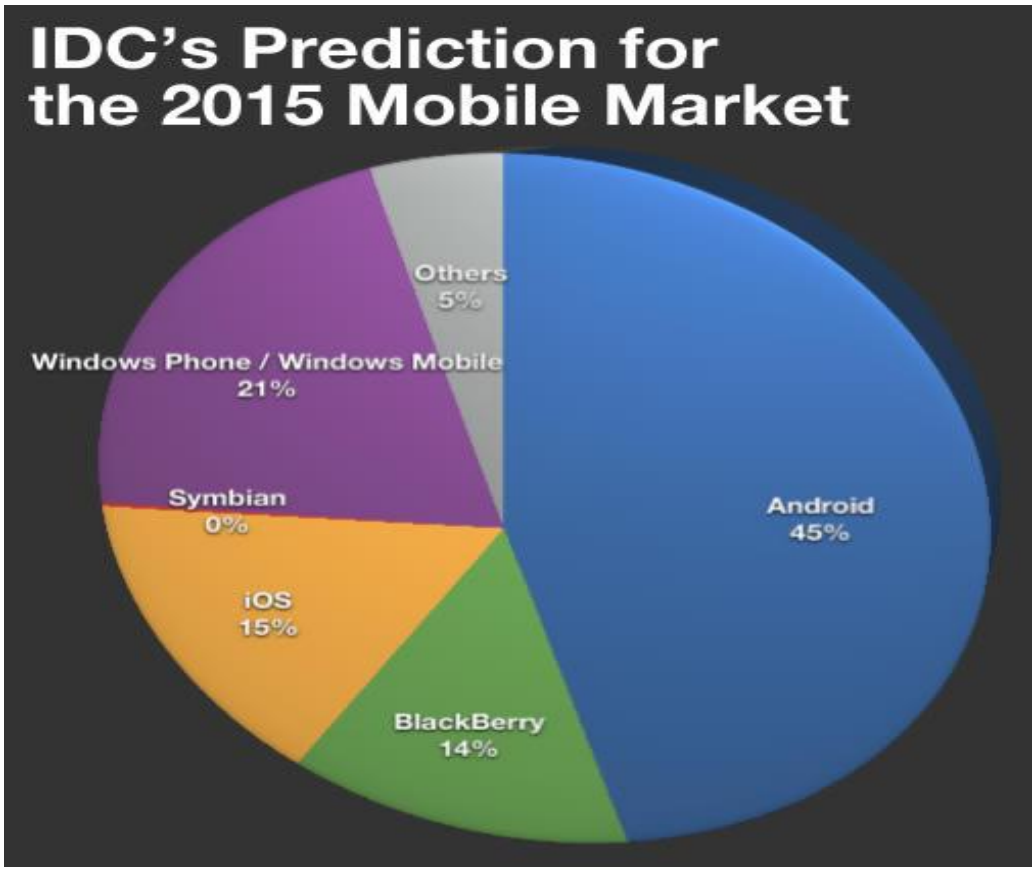
  - Όμως, στη δεύτερη θέση θα βρεθούν τα Windows Phone με 20,9%, καθώς η IDC αναμένει ότι η συνεργασία Microsoft - Nokia θα λειτουργήσει καταλυτικά στην αύξηση της δημοτικότητας της συγκεκριμένης πλατφόρμας.

  - Ενδιαφέρον είναι ακόμη το γεγονός πως δύο πλατφόρμες που έχουν τους δικούς τους φανατικούς οπαδούς θα διατηρήσουν σταθερά τα μερίδιά τους. Συγκεκριμένα, το iOS της Apple θα είναι περίπου στο 15,3%, με το Blackberry OS να κινείται στο 13,7%. Όσο για το Symbian, ουσιαστικά δεν υπάρχει δεδομένο ότι το μερίδιό του θα είναι στο 0,2%.

  - Πέραν των προαναφερθέντων λειτουργικών συστημάτων, υπάρχουν και άλλες πλατφόρμες για smartphones. Η IDC εκτιμά ότι το συνολικό μερίδιό τους θα είναι περίπου στο 4,6%.

  - Οι αναλυτές της IDC δεν δείχνουν να πιστεύουν ότι το webOS, το λειτουργικό σύστημα για φορητές συσκευές που έχει αναπτύξει η Palm, η οποία αποτελεί πλέον τμήμα της HP, να έχει την ανταπόκριση που περιμένουν τα στελέχη του αμερικανικού κολοσσού.

  - Ενδεχομένως και η IDC να υποστηρίζει ότι στην παγκόσμια αγορά δεν μπορούν να αντέξουν περισσότερες από τέσσερις πλατφόρμες, εκ των οποίων οι δύο θα απευθύνονται στο ευρύ κοινό και θα έχουν μεγάλη γκάμα συσκευών.



Σχήμα 46. Η πρόβλεψη για τις πωλήσεις των λειτουργικών συστημάτων μέχρι το 2015 από την εταιρία IDC

Κεφάλαιο 4

Η εφαρμογή Πλοήγησης σε Εσωτερικούς χώρους - ECE Indoor Navigation

4.1. Εισαγωγή

Στην παρούσα διπλωματική εργασία αναπτύχθηκε μια εφαρμογή πλοήγησης εσωτερικού χώρου και εύρεσης πληροφοριών για τα Νέα Κτήρια στης Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών (ΗΜΜΥ) του Εθνικού Μετσόβιου Πολυτεχνείου (ΕΜΠ). Το σύστημα αυτό παρέχει πολλές δυνατότητες και σημαντικές πληροφορίες για τον χρήστη:

- Μπορεί να βρει πληροφορίες καθηγητών ή γραμματειών σχετικά με τα τηλέφωνα και την τοποθεσία τους.

- Μπορεί να δει τον χάρτη του κάθε ορόφου, στον οποίο είναι καταγεγραμμένες όλες οι αίθουσες, τα αμφιθέατρα, τα γραφεία των καθηγητών και άλλα σημεία ενδιαφέροντος (Points of Interest – POI). Επίσης μπορεί να κάνει zoom in και zoom out το χάρτη για να δει πιο καλά τις πληροφορίες που τον ενδιαφέρουν όπως και να μετακινεί (scroll) το χάρτη σε όποια πλευρά θέλει.

- Το κυριότερο κομμάτι της εφαρμογής είναι η πλοήγηση. Ο χρήστης διαλέγει μέσα από τη βάση δεδομένων που βρίσκεται και έπειτα σε ποιο σημείο (αίθουσα, αμφιθέατρο, γραφείο, wc, exit κ.α.) θέλει να πάει. Η πλοήγηση ξεκινάει από το σημείο που βρίσκεται ο χρήστης και του δείχνει την ελάχιστη διαδρομή για τον προορισμό με μια μπλε γραμμή που ζωγραφίζεται σταδιακά.

- Πολύ σημαντικό μέρος της εφαρμογής είναι η φωνητική πλοήγηση αλλά και τα κατάλληλα φωνητικά και οπτικά μηνύματα που ακούγονται και εμφανίζονται, ταυτόχρονα, στα αγγλικά κατά την αναζήτηση πληροφοριών. Γενικά καθ' όλη την διάρκεια της εφαρμογής, τα οπτικά μηνύματα είναι και φωνητικές οδηγίες για να διευκολύνουν το χρήστη.

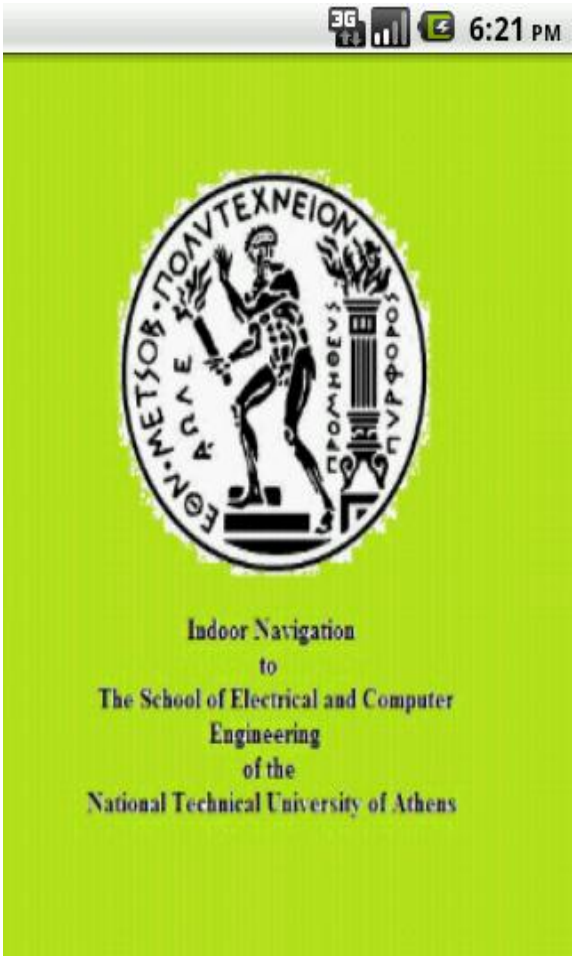
- Έχει τη δυνατότητα να αλλάξει την ύπαρξη των φωνητικών ομιλιών, την ταχύτητα της πλοήγησης, ποιον όροφο θα δείχνει η εφαρμογή όταν ξεκινάει και άλλες ακόμα διαθέσιμες επιλογές μέσα από τις ρυθμίσεις της εφαρμογής.

4.2. Η εφαρμογή ECE Indoor Navigation

Στην ενότητα αυτή θα αναλυθεί η εφαρμογή ECE Indoor Navigation αναφέροντας τις επιμέρους δυνατότητες, αλλά και επιλογές που έχει ο χρήστης χρησιμοποιώντας τη.

4.3. Έναρξη Εφαρμογής

Όταν ο χρήστης «τρέξει» την εφαρμογή τότε η αρχική οθόνη φαίνεται στις δύο ακόλουθες Εικόνες:



Εικόνα 1

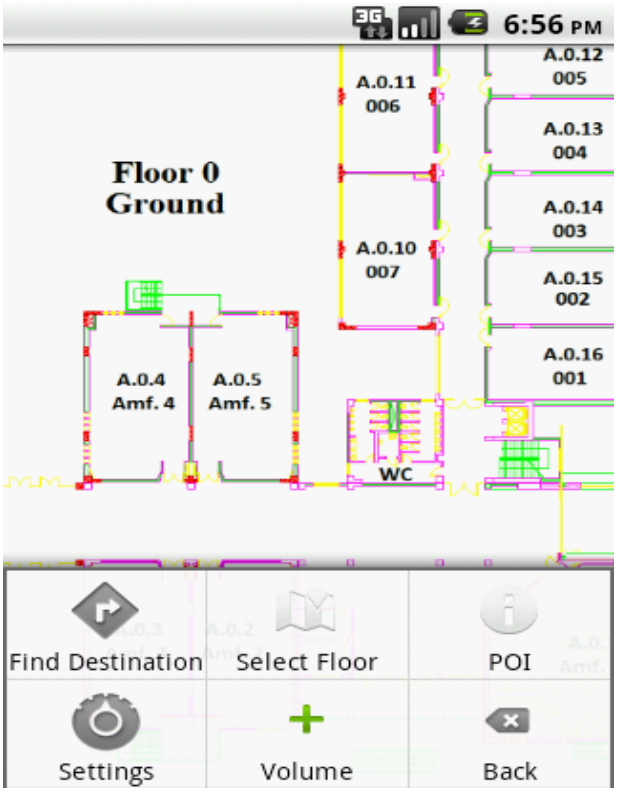
Εικόνα 2

- Στην Εικόνα 1 φαίνεται πως ξεκινάει η εφαρμογή, στην οποία παρουσιάζεται τι είδους εφαρμογή είναι και για ποια σχολή του Εθνικού Μετσόβιου Πολυτεχνείου αναπτύχθηκε («Indoor Navigation to The School of Electrical and Computer Engineering of The National Technical University of Athens»). Επίσης, κατά τη διάρκεια της εικόνας αυτής ακούγεται για τέσσερα δευτερόλεπτα μουσική.

- Στην Εικόνα 2, φαίνεται ο χάρτης του ισογείου από τα Νέα Κτήρια Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, στον οποίο όπως φαίνεται είναι καταγεγραμμένες οι ονομασίες των αιθουσών, αμφιθεάτρων, wc κ.α. Στο κάτω μέρος της εικόνας φαίνεται ένα οπτικό μήνυμα αλλά ταυτόχρονα είναι και φωνητική οδηγία το οποίο λέει «Hello, I am Ntuadroid and I will help you to find your destination. You are on the Floor 0». Το όνομα «Ntuadroid» δόθηκε στο σύστημα των φωνητικών οδηγιών και ειδοποιήσεων.

4.4. Κεντρικό Μενού

Όταν ο χρήστης πατήσει το πλήκτρο «menu» του κινητού εμφανίζεται το κεντρικό μενού της εφαρμογής, στο οποίο ο χρήστης έχει τις παρακάτω επιλογές όπως παρουσιάζονται στην Εικόνα 3 που ακολουθεί.



Εικόνα 3

Τα εργαλεία που δίνονται στο χρήστη στο κεντρικό μενού (Εικόνα 3) είναι τα παρακάτω:

1. **Find Destination**: Με την επιλογή αυτή ο χρήστης μπαίνει στο μενού για την πλοήγηση. Ο χρήστης διαλέγει την θέση του και το μέρος στο οποίο θέλει να πάει. Ύστερα ακολουθεί η πλοήγηση με τα οπτικά μηνύματα και τις φωνητικές οδηγίες προς το μέρος που έχει επιλέξει.

2. **Select Floor:** Ο χρήστης επιλέγοντας αυτό το κουμπί μπορεί να επιλέξει ποιον όροφο θέλει να δει.

3. **POI:** Πατώντας αυτό το κουμπί, ο χρήστης μπαίνει σε μια λίστα από σημεία ενδιαφέροντος (Points of Interest). Στη λίστα αυτή μπορεί να δει την τοποθεσία των σημείων αυτών πάνω στο χάρτη όπως και επίσης πληροφορίες για τους καθηγητές και τις γραμματείες.

4. **Settings:** Η επιλογή αυτή μεταφέρει το χρήστη στη κλάση προβολής των ρυθμίσεων της εφαρμογής. Ο χρήσης έχει μια σειρά επιλογών και προτιμήσεων.

5. **Volume:** Ο χρήστης με την επιλογή αυτή του εμφανίζεται μια μπάρα επιλογής της έντασης των φωνητικών ειδοποιήσεων.

6. **Back:** Η επιλογή αυτή μεταφέρει τον χρήστη στην προηγούμενη activity και αν δεν υπάρχει άλλη activity τότε τον μεταφέρει στην κεντρική οθόνη του κινητού.

Για λόγους καλύτερης συνοχής στην περιγραφή της εφαρμογής, τα παραπάνω θα αναφερθούν αντίστροφα στην ανάλυση των επιμέρους προγραμμάτων και επιλογών που έχει ο χρήστης.

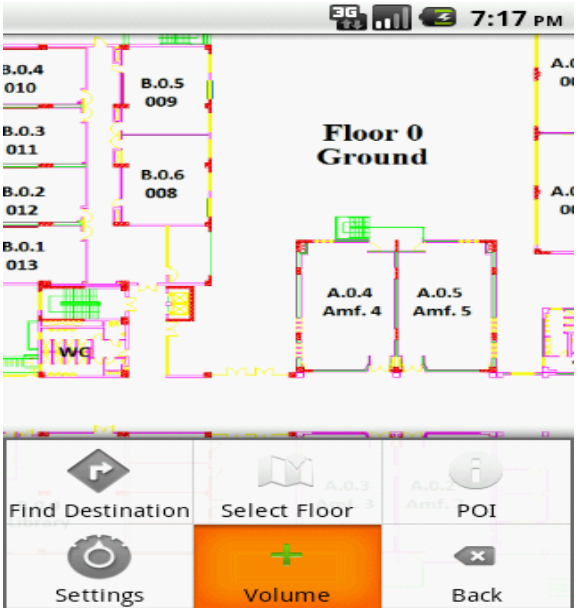
4.4.1. Επιλογή Back

- Με την επιλογή του κουμπιού «**Back**» (Εικόνα 4), όπως υποδηλώνει και το όνομα του κουμπιού επιστρέφει τον χρήστη στην προηγούμενη κλάση της εφαρμογής στην οποία ήταν. Μάλιστα, αν σε περίπτωση είναι στην πρώτη «activity» και πατήσει το κουμπί αυτό, τότε θα κλείσει η εφαρμογή και θα τον πετάξει στη κεντρική λίστα που έχει ο χρήστης τις εφαρμογές του κινητού τηλεφώνου του. Λειτουργεί σαν το «Back Button» του κινητού και συνήθως επιστρέφει τον χρήστη στην κεντρική κλάση, την MapActivity.class.

Εικόνα 4

4.4.2. Επιλογή Ρύθμισης Έντασης Ήχου (Volume)

- Πατώντας το κουμπί «**Volume**» από το κεντρικό μενού (Εικόνα 5) αναδύεται ένα παράθυρο πάνω στην υπάρχουσα οθόνη το οποίο αναγράφει «**Options**» και από κάτω «**Speech Volume**» με μια μπάρα και ο χρήστης μπορεί να επιλέξει την ένταση των φωνητικών ειδοποιήσεων της εφαρμογής με το να μετακινείστην μπάρα αριστερά (μείωση) ή δεξιά (αύξηση). Έπειτα πατώντας το «**Ok**» αποθηκευεται η επιλογή του ή «**Cancel**» ακυρώνονται οι αλλαγές που έγιναν (Εικόνα 6).



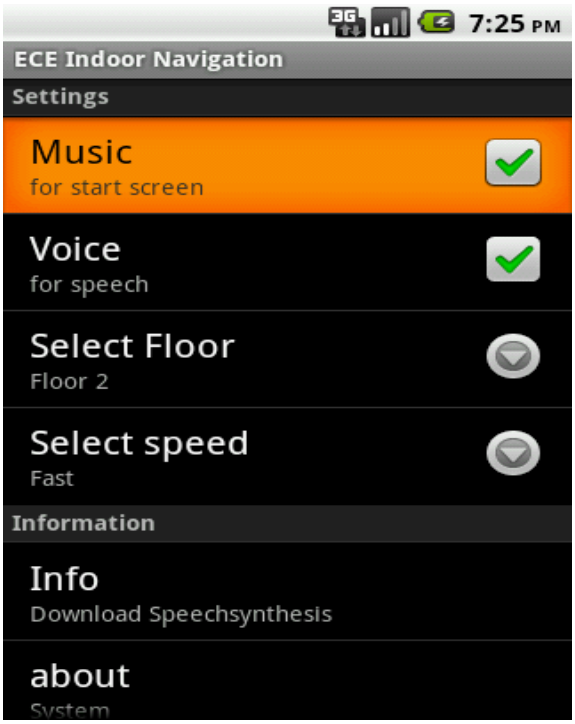
Εικόνα 5

Εικόνα 6

4.4.3. Επιλογή Ρυθμίσεις Εφαρμογής (Settings)

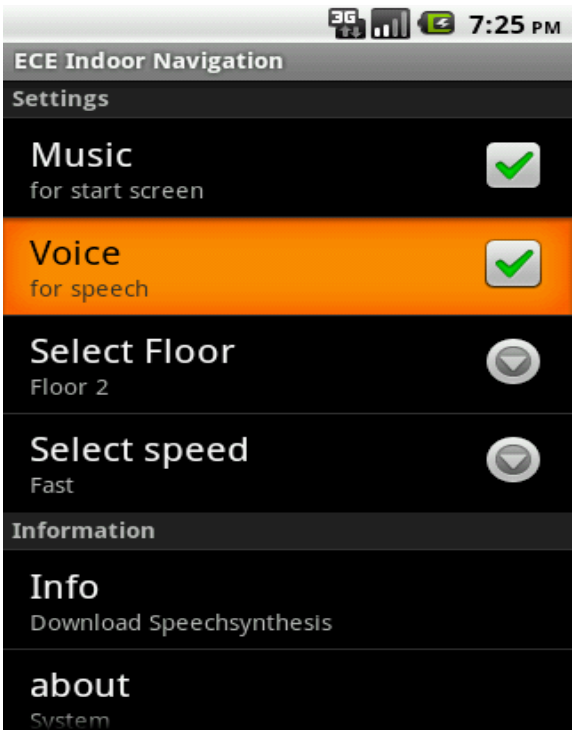
- Με το κουμπί «**Settings**» δίνεται η δυνατότητα στο χρήστη να μεταφερθεί στις ρυθμίσεις της εφαρμογής. Τις αλλαγές που μπορεί να κάνει στο σύστημα της εφαρμογής αναλύονται παρακάτω, οι οποίες παρουσιάζονται και σε εικόνες.

- **Music *for start screen*:** Όπως αναφέρθηκε και προηγουμένως, κατά την έναρξη της εφαρμογής ακούγεται για τέσσερα δευτερόλεπτα μουσική. Ο χρήστης έχει τη δυνατότητα να επιλέξει αν θέλει να ακούγεται ή όχι η μουσική με το να «τικάρει» ή να «ξετικάρει» το κουτί αντίστοιχα (Εικόνα 7). Η επιλογή αυτή αποθηκεύεται στο σύστημα, και την επόμενη φορά που θα ξεκινήσει η εφαρμογή, θα ισχύει η επιλογή που επέλεξε ο χρήστης.



Εικόνα 7

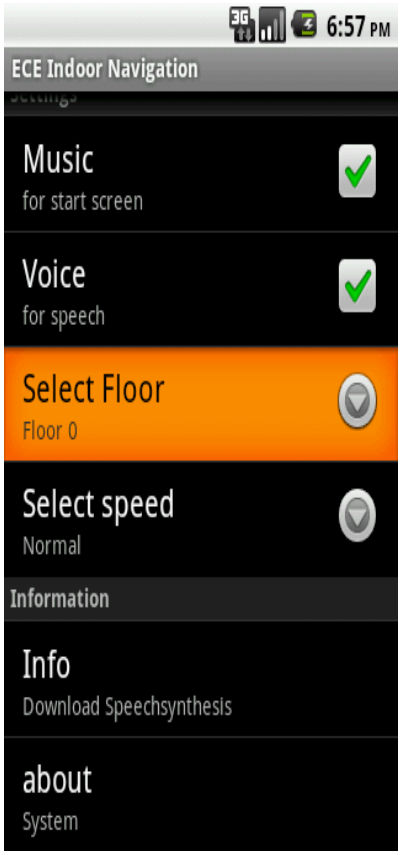
- **Voice *for speech*:** Είτε κατά την εξερεύνηση της εφαρμογής για την αναζήτηση πληροφοριών είτε κατά την πλοήγηση, ακούγονται φωνητικές οδηγίες και ειδοποιήσεις στα αγγλικά. Ο χρήστης μπορεί να διαλέξει αν θέλει να ακούγονται οι φωνητικές ομιλίες ή όχι. Η επιλογή αυτή εφαρμόζεται αμέσως στο σύστημα της εφαρμογής (Εικόνα 8).



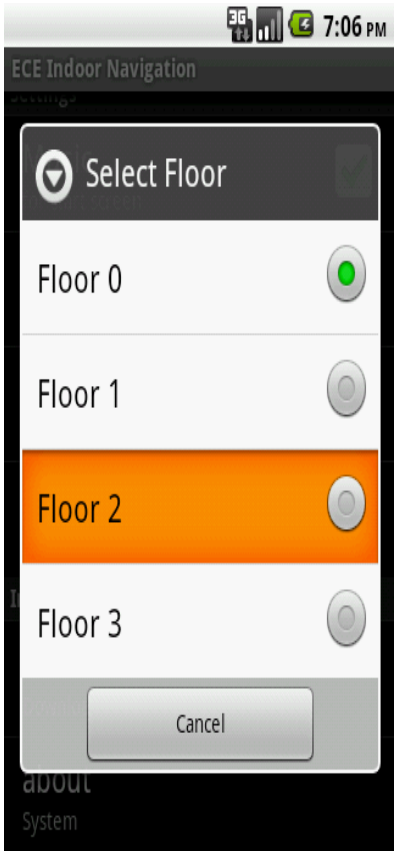
Εικόνα 8

- **Select Floor:** Ο χρήστης μπορεί να επιλέξει όταν θα ξεκινάει η εφαρμογή, από ποιον όροφο θέλει να εμφανίζεται ο χάρτης (Εικόνα 9). Οι δυνατές επιλογές όπως φαίνεται στην Εικόνα 10 είναι οι εξής:
    - **Floor 0**
    - **Floor 1**
    - **Floor 2**
    - **Floor 3**

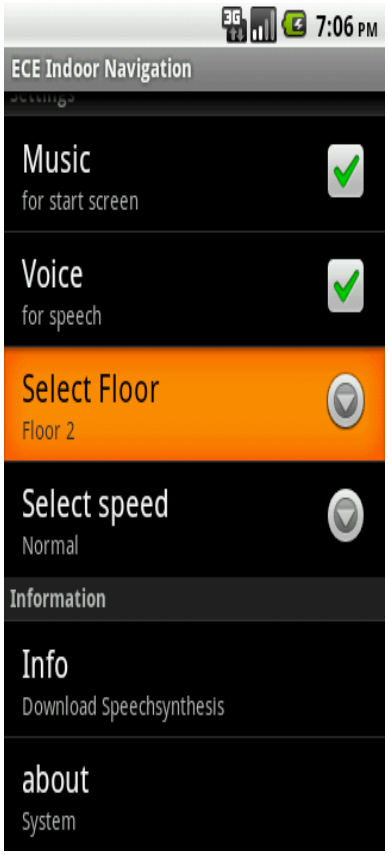
Ο χρήστης μπορεί να επιλέξει ένα από αυτά είτε να πατήσει το κουμπί «**Cancel**» για να ακυρώσει την καταχώρηση. Σε περίπτωση που επιλέξει κάποιο όροφο, η επιλογή του εμφανίζεται κάτω από το «**Select Floor**». Στην προκειμένη περίπτωση ήταν αρχικά ο όροφος 0-ισοόγειο «**Floor 0**» (Εικόνα 9) και επιλέχθηκε ο 2ος όροφος «**Floor 2**» (Εικόνα 11). Έτσι, την επόμενη φορά που ο χρήστης επιθυμήσει να χρησιμοποιήσει την εφαρμογή, ο χάρτης που θα φορτωθεί θα είναι του 2ου ορόφου.



Εικόνα 9



Εικόνα 10

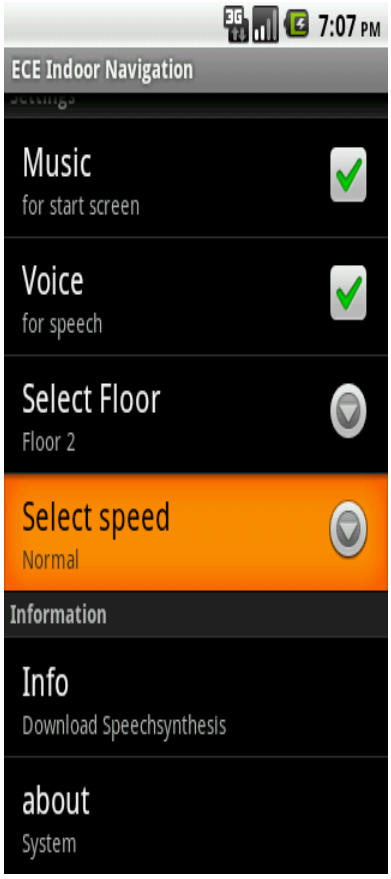


Εικόνα 11

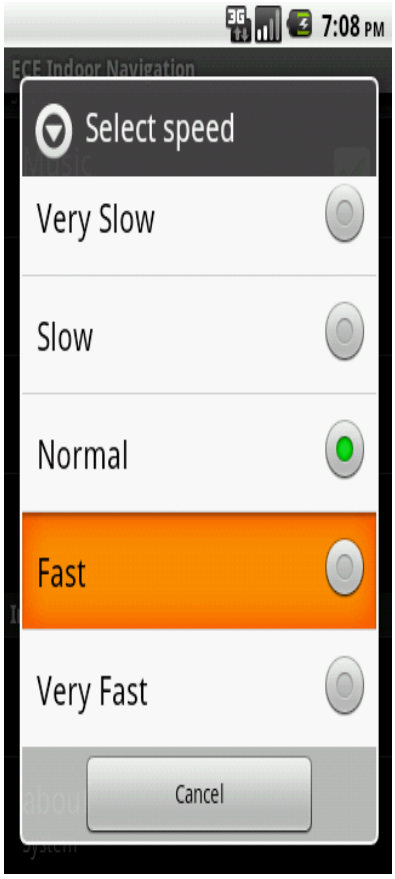
- **Select speed:** Κατά την πλοήγηση, ζωγραφίζεται με ορισμένη ταχύτητα η γραμμή, η οποία δείχνει την διαδρομή που πρέπει να ακολουθήσει ο χρήστης για να φτάσει τον προορισμό που έχει δηλώσει. Έχει οριστεί η ταχύτητα «**Normal**», η οποία είναι η πιο κατάλληλη για να την ακολουθήσει ένας χρήστης (Εικόνα 12). Αλλά υπάρχει η δυνατότητα να αυξηθεί ή και να μειωθεί ανάλογα με τις προτιμήσεις και τις ιδιαιτερότητες του κάθε χρήστη. Οι δυνατές επιλογές είναι οι εξής (Εικόνα 13):

    - **Very Slow**
    - **Slow**
    - **Normal**
    - **Fast**
    - **Very Fast**

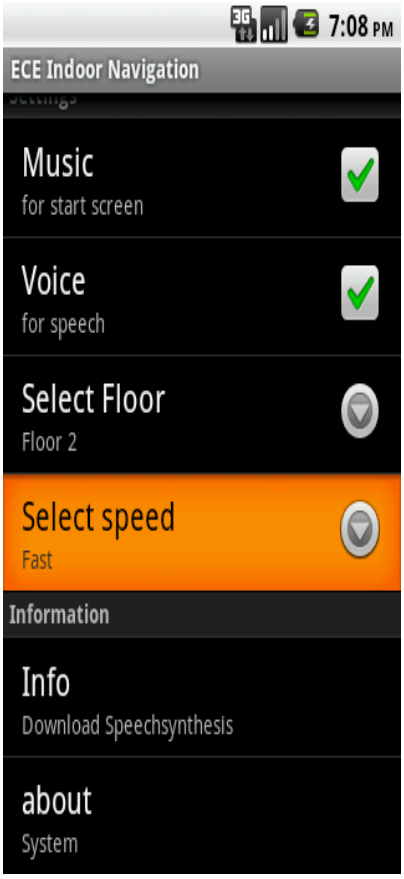
Ο χρήστης μπορεί να επιλέξει ένα από αυτά είτε να πατήσει το κουμπί «**Cancel**» για να ακυρώσει την καταχώρηση. Σε περίπτωση που επιλέξει κάποια ταχύτητα, η επιλογή του εμφανίζεται κάτω από το «**Select Speed**». Στην προκειμένη περίπτωση ήταν αρχικά η επιλογή «**Normal**» (Εικόνα 12) και επιλέχθηκε η ταχύτητα «**Fast**» (Εικόνα 14).



Εικόνα 12



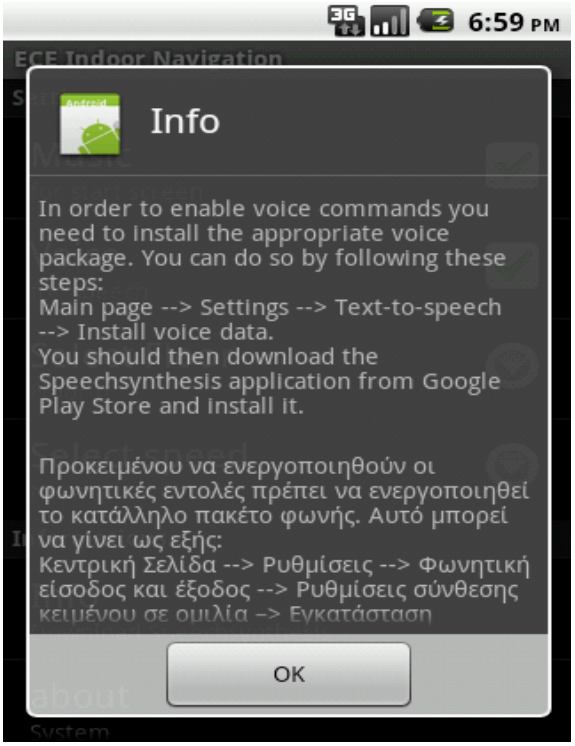
Εικόνα 13



Εικόνα 14

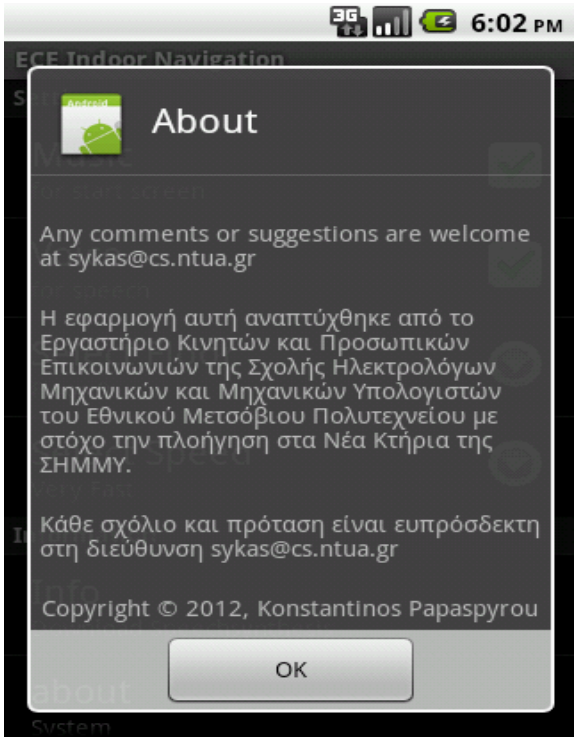
Στις ρυθμίσεις υπάρχει και μια ενότητα που ονομάζεται «**Information**». Σε αυτήν υπάρχουν δύο αντικείμενα., το «**Info**» και το «**about**».

- **Info *Download Speechsynthesis***: Πατώντας πάνω στο «**Info**», αναδύεται ένα παράθυρο στο οποίο περιγράφεται πώς να κατεβάσει ο χρήστης το Speechsynthesis (Εικόνα 15). Το Speechsynthesis είναι μια εφαρμογή του κινητού για να μπορεί το ίδιο το κινητό του να συνθέσει κείμενο σε ομιλία (text-to-speech). Αυτή η εφαρμογή χρειάζεται για να μπορούν να ακουστούν οι φωνητικές οδηγίες και ειδοποιήσεις της εφαρμογής.



Εικόνα 15

- **about *System*:** Πατώντας πάνω στο «**about**», αναδύεται ένα παράθυρο όπως φαίνεται στην Εικόνα 16, στο οποίο υπάρχουν πληροφορίες για την εφαρμογή αυτή, δηλαδή για ποια σχολή και για ποιο λόγο αναπτύχθηκε. Επιπλέον υπάρχει σύνδεσμος επικοινωνίας αν υπάρχουν προτάσεις από τους χρήστες καθώς και το όνομα του δημιουργού της εφαρμογής, Κωνσταντίνος Παπασπύρου.

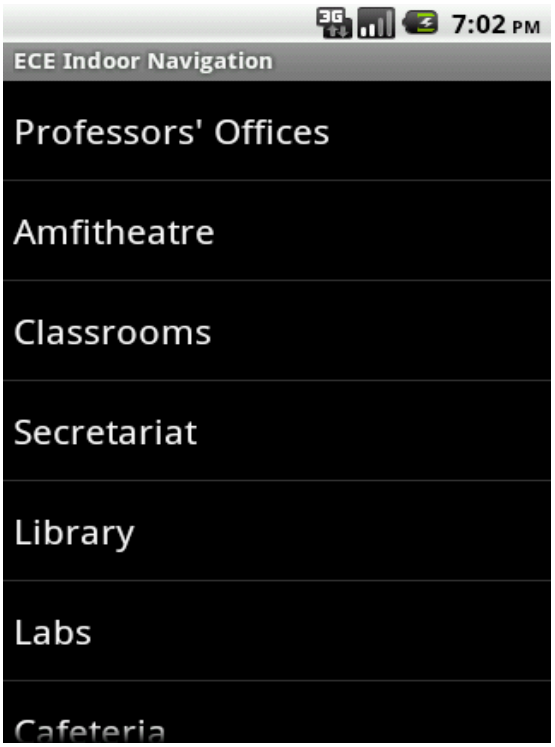


Εικόνα 16

4.4.4. Επιλογή Σημεία Ενδιαφέροντος (POI)

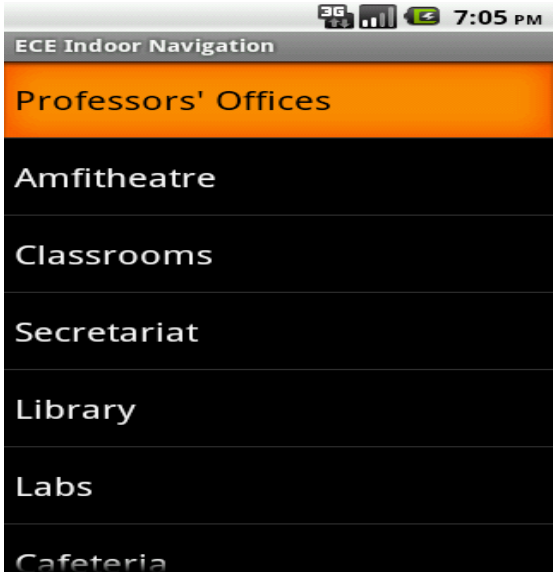
Επιλέγοντας το κουμπί «**POI**», ο χρήστης μεταφέρεται σε μια λίστα από σημεία ενδιαφέροντος όπως γραφεία καθηγητών, αμφιθέατρα, αίθουσες, γραμματείες, κυλικείο, τουαλέτες, εξόδους και άλλα πολλά (Εικόνα 17). Έπειτα, επιλέγοντας το καθένα από αυτά, ο χρήστης μεταφέρεται και στην αντίστοιχη κλάση. Είτε θα δίνει στο χρήστη τη δυνατότητα να επιλέξει σε ποιον όροφο θέλει να δει αυτό που επέλεξε και μετά να δει την θέση του πάνω στο χάρτη είτε αν υπάρχει μόνο ένα στα Νέα Κτήρια τότε θα του εμφανίσει κατευθείαν τη θέση του στο χάρτη και σε ποιον όροφο είναι.. Όλα τα POI είναι τα εξής:

- **Professors' Offices**
- **Amfitheatre**
- **Classrooms**
- **Secretariat**
- **Library**
- **Labs**
- **Cafeteria**
- **WC**
- **Elevators**
- **Stairs**
- **Exit**
- **Safety Exit**



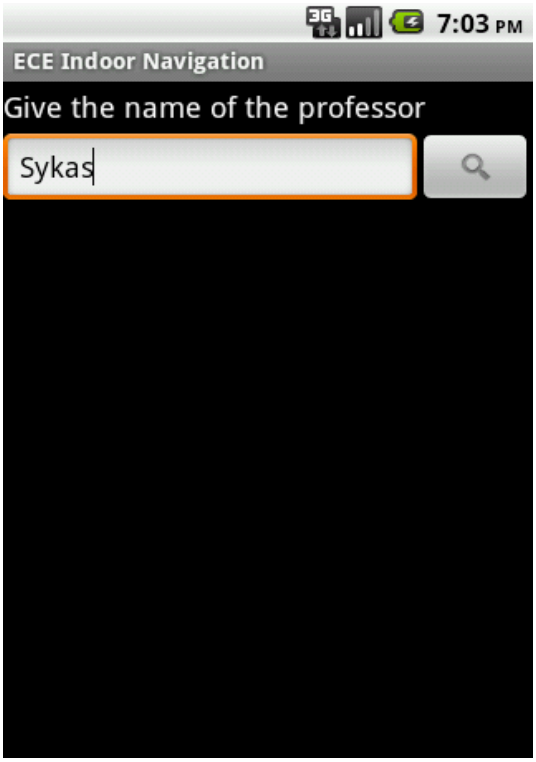
Εικόνα 17

❖ Ένα από τα πιο σημαντικά μέρη μιας Σχολής είναι τα γραφεία των καθηγητών (**Professors' Offices**). Έτσι ο χρήστης έχει τη δυνατότητα να ψάξει όποιον καθηγητή θέλει, ο οποίος να έχει γραφείο στα Νέα Κτήρια της Σχολής ΗΜΜΥ, και να δει πληροφορίες σχετικά με τον αριθμό του γραφείου του και τον όροφο που βρίσκεται το γραφείο. Επιπλέον, μπορεί να δει το τηλέφωνο, το φαξ και το email του (όσα είναι διαθέσιμα) καθώς επίσης και το που βρίσκεται το γραφείο του πάνω στο χάρτη. Παρακάτω παρουσιάζεται ένα παράδειγμα πως ακριβώς λειτουργεί.

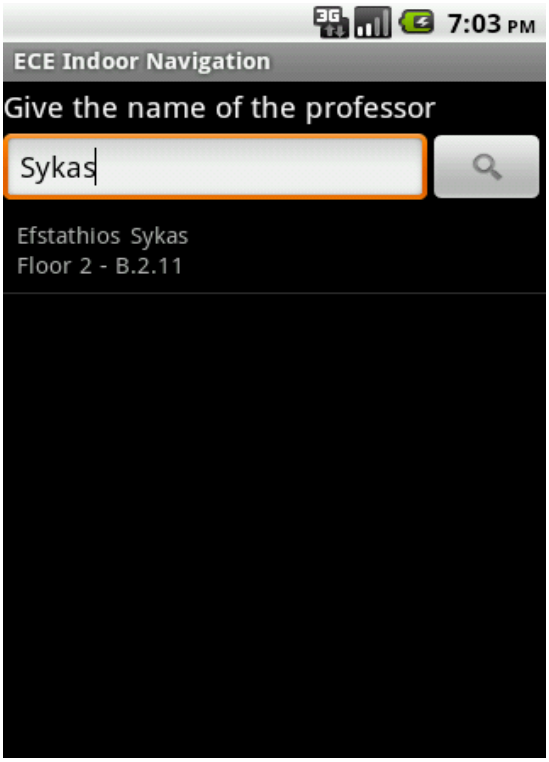


Εικόνα 18

➔ Αρχικά, επιλέγοντας ο χρήστης το «**Professors' Offices**» από τη λίστα των POI, μεταφέρεται σε μια άλλη κλάση στην οποία υπάρχει ένα κουτί αναζήτησης (Search box). Ο χρήστης μπορεί να γράψει το όνομα του καθηγητή που επιθυμεί και κατόπιν, το σύστημα αναζητεί από τη βάση δεδομένων το όνομα που έδωσε ο χρήστης. Έστω ότι ο χρήστης γράφει «Sykas» όπως φαίνεται και στην Εικόνα 19. Ύστερα, πατά το κουμπί για την αναζήτηση. Στην Εικόνα 20, παρατηρείται ότι έχει εμφανιστεί το όνομα «Efstathios Sykas» και κάτω από το όνομά του αναγράφεται ο όροφος και το νούμερο του γραφείου του. Στη συγκεκριμένη περίπτωση είναι ο 2^{ος} όροφος «Floor 2» και ο αριθμός του γραφείου του «B.2.11».

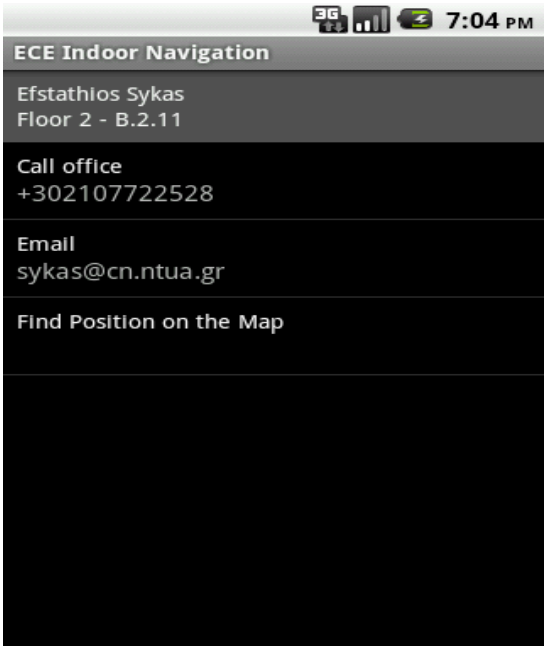


Εικόνα 19



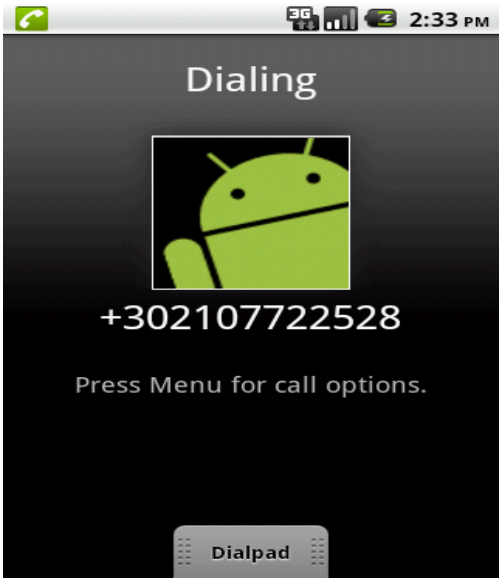
Εικόνα 20

Στη συνέχεια, πατάει πάνω σε αυτό που του έβγαλε η αναζήτηση και ο χρήστης μεταφέρεται στα στοιχεία του καθηγητή κ. Ευστάθιου Συκά. Στην Εικόνα 21 όπως είναι εμφανές έχουν εμφανιστεί το τηλέφωνο του γραφείου του, το email του και μια επιλογή που λέει «Find Position on the Map».



Εικόνα 21

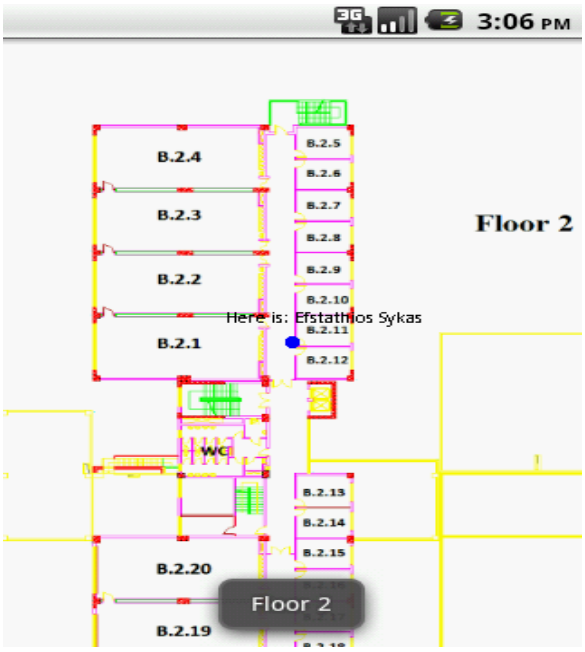
- Όταν ο χρήστης πατήσει το «**Call office**», τότε καλεί τον αριθμό που φαίνεται κάτω από αυτό και έτσι μπορεί να επικοινωνήσει με τον καθηγητή (Εικόνα 22).



Εικόνα 22

- Όταν ο χρήστης πατήσει το «**Email**», τότε μπορεί να στείλει email και να επικοινωνήσει μέσω ηλεκτρονικού ταχυδρομείου με τον καθηγητή. Μάλιστα, το email είναι διαθέσιμο για τον χρήστη, όπως φαίνεται στην Εικόνα 21.

- Τέλος, όταν πατηθεί το «Find Position on the Map», τότε θα εμφανιστεί η θέση του γραφείου του καθηγητή πάνω στο χάρτη, όπως φαίνεται και στην Εικόνα 23.



Εικόνα 23

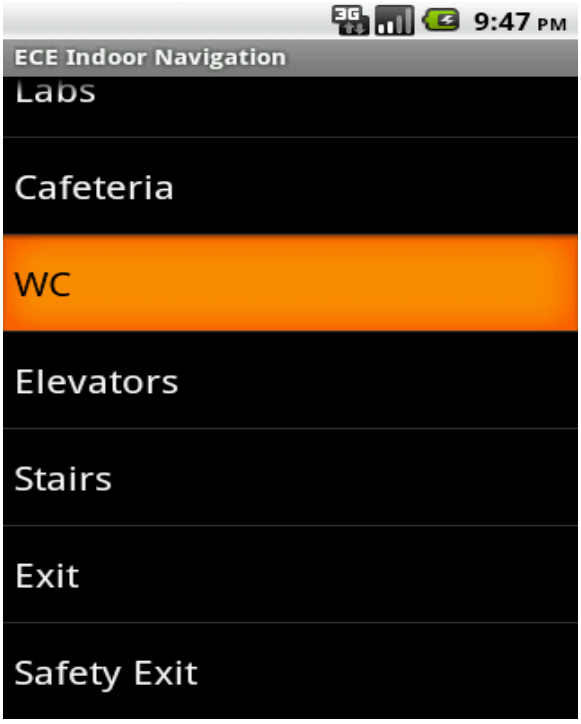
➔ Αντίστοιχος τρόπος αναζήτησης των καθηγητών είναι και για τις γραμματείες «**Secretariat**». Αλλά υπάρχουν δύο διαφορές.

- Η μία διαφορά είναι ότι η λίστα είναι δυναμική, δηλαδή φαίνεται από την αρχή η λίστα με τις γραμματείες και όταν ο χρήστης πατήσει στο κουτί αναζήτησης τη γραμματεία που ψάχνει, θα εμφανιστεί μόνο αυτή.

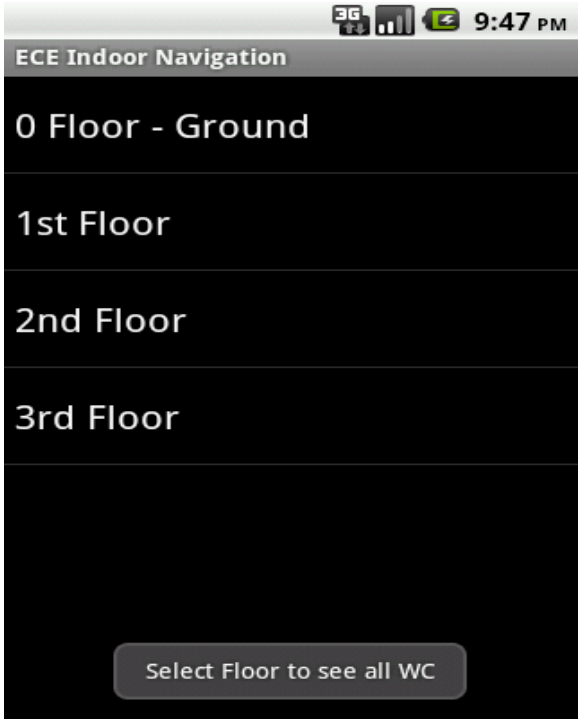
- Η δεύτερη είναι ότι έχει περισσότερα τηλέφωνα και φαξ διαθέσιμα για τον χρήστη.

➔ Τα υπόλοιπα **POI** λειτουργούν με τον ίδιο τρόπο. Ο χρήστης είτε πατάει σε μία επιλογή από τη λίστα και του δείχνει κατευθείαν στο χάρτη την τοποθεσία του (**Cafeteria**, **Library**) είτε δίνει στον χρήστη να επιλέξει σε ποιον όροφο θέλει να δει αυτό που επέλεξε (**WC**, **Elevators**, **Stairs**, **Exit**, **Safety Exit**) είτε να του εμφανίζεται νέα λίστα με νέες επιλογές για να επιλέξει πιο συγκεκριμένα (**Amfitheatre**, **Classrooms, Labs**). Για την κάθε περίπτωση θα παρουσιαστεί σε εικόνες ένα παράδειγμα από αυτά.

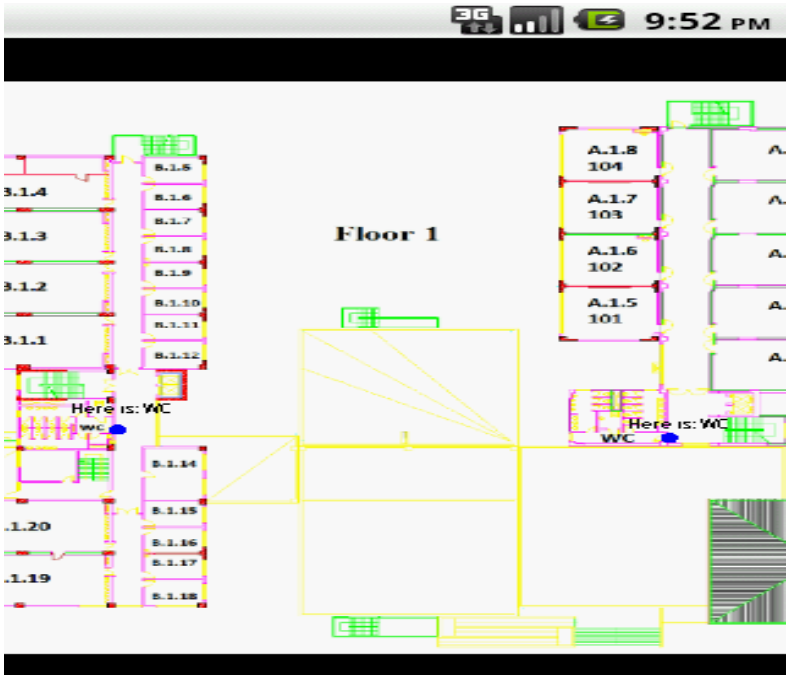
✓ **<u>WC</u>**



Εικόνα 24

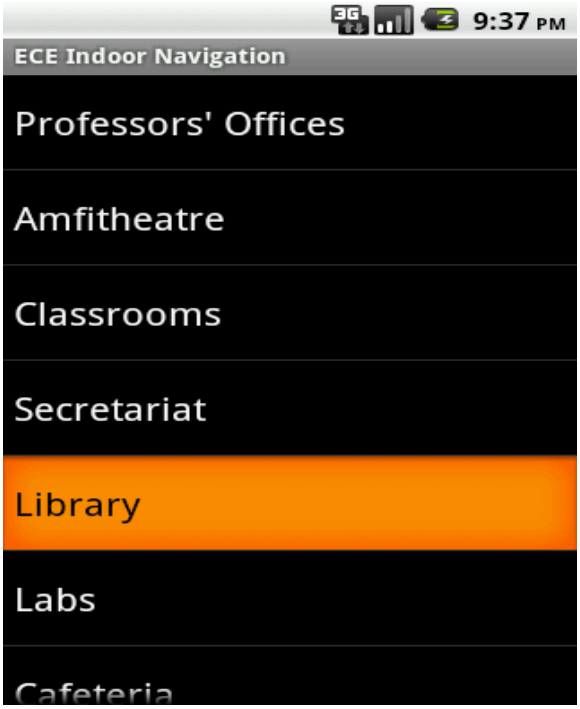


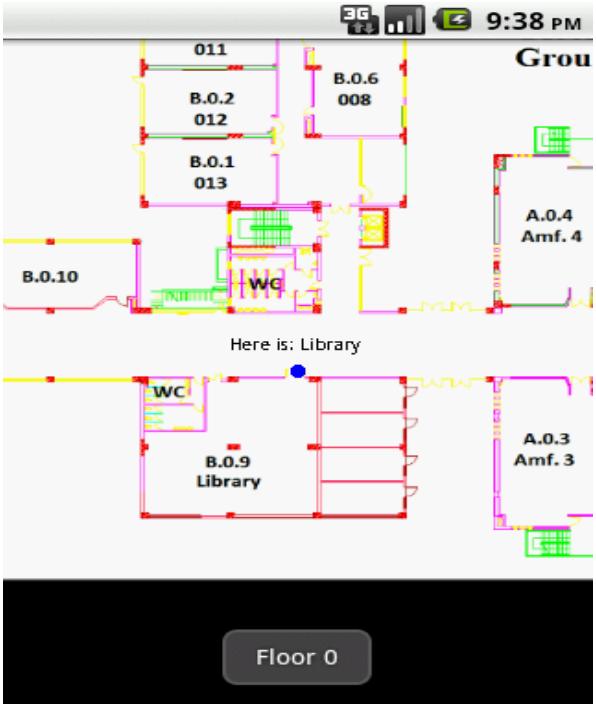
Εικόνα 25



Εικόνα 26

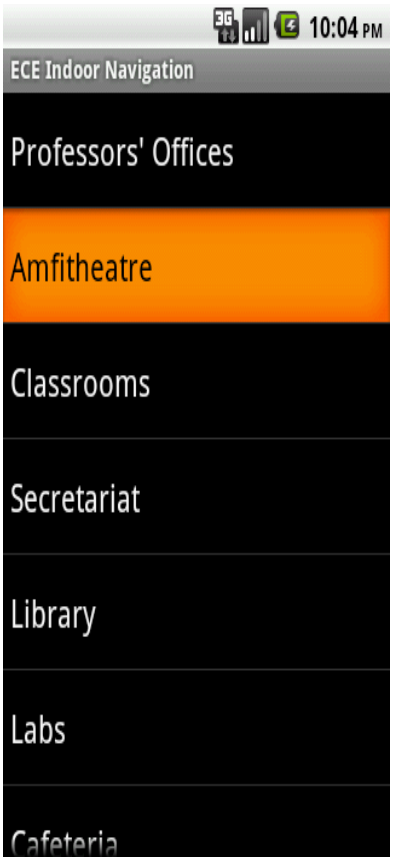
✓ **Library**

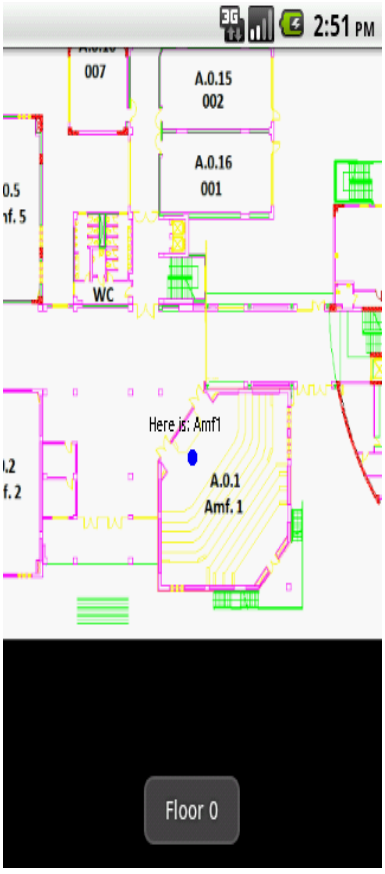




Εικόνα 27 *Εικόνα 28*

✓ **Amfitheatre**



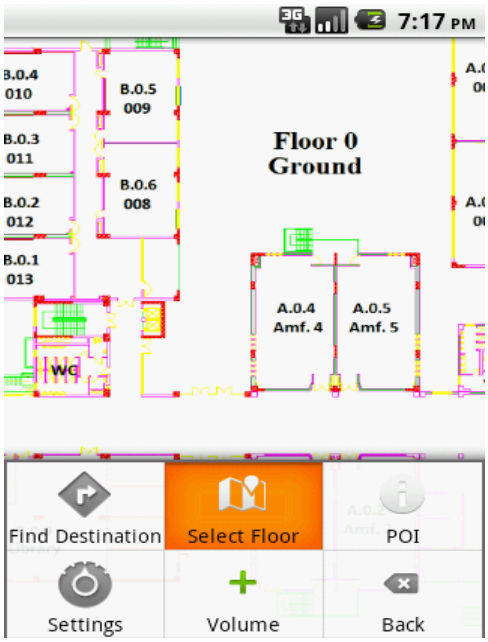


Εικόνα 29 *Εικόνα 30* *Εικόνα 31*

4.4.5.Επιλογή Αλλαγή Ορόφου (Select Floor)

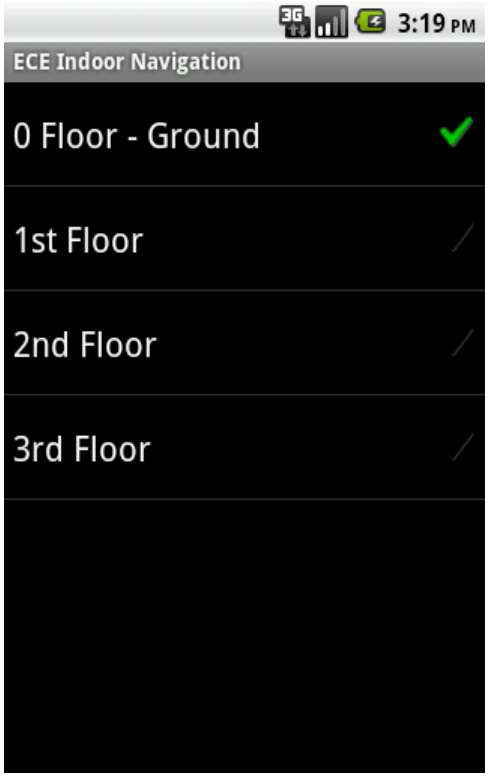
- Με το κουμπί «**Select Floor**» δίνεται η δυνατότητα στο χρήστη να μεταφερθεί σε μια λίστα με τους ορόφους του κτηρίου. Η λίστα όπως φαίνεται στην Εικόνα 32, είναι:

  - **0 Floor – Ground**
  - **1st Floor**
  - **2nd Floor**
  - **3rd Floor**

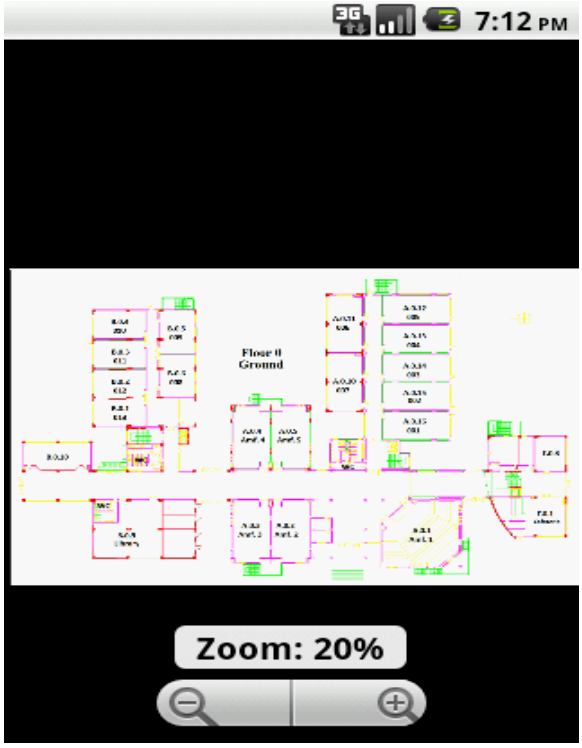


Εικόνα 32

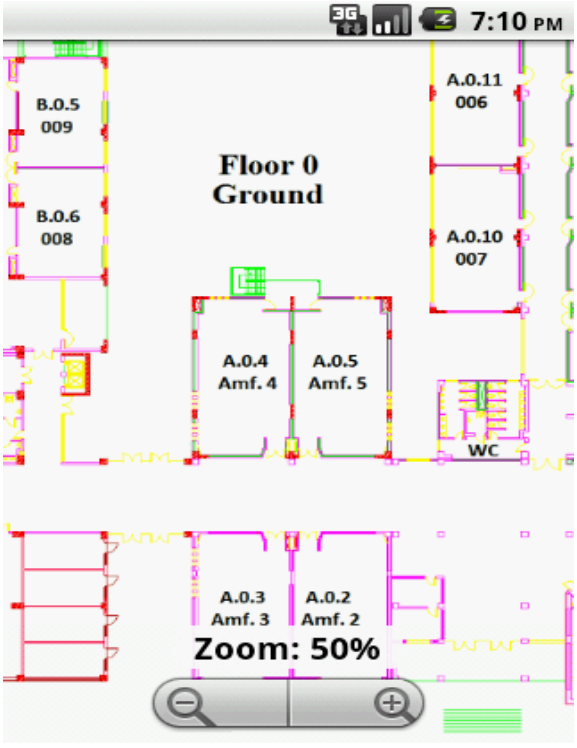
Κοιτάζοντας την Εικόνα 33, παρατηρείται ότι υπάρχει ένα «τικ» δίπλα από την επιλογή «**0 Floor – Ground**». Αυτό γίνεται γιατί πριν ο χάρτης έδειχνε τον όροφο 0 – ισόγειο (Εικόνα 32). Οπότε, σε όποιον όροφο δείχνει η εφαρμογή και πατήσει ο χρήστης στην επιλογή ορόφου θα του έχει επιλεγμένο τον όροφο στον οποίο είναι. Επίσης, όταν ο χρήστης βρίσκεται στους χάρτες έχει τη δυνατότητα να τους μετακινεί προς όποια κατεύθυνση επιθυμεί (scroll) και έχει ακόμα την επιλογή να μεγεθύνει (zoom in) ή να μικρύνει (zoom out) το χάρτη πατώντας πάνω στην οθόνη για ορισμένο χρόνο και εμφανίζεται ο zoom controller όπως φαίνεται και στις παρακάτω εικόνες. Το zoom κυμαίνεται από 10% έως 300% και η προεπιλογή είναι στο 50%. Στις ακόλουθες εικόνες (Εικόνα 34, Εικόνα 35, Εικόνα 36) εμφανίζονται τρία στιγμιότυπα με Zoom 20%, 50% και 100% αντίστοιχα.



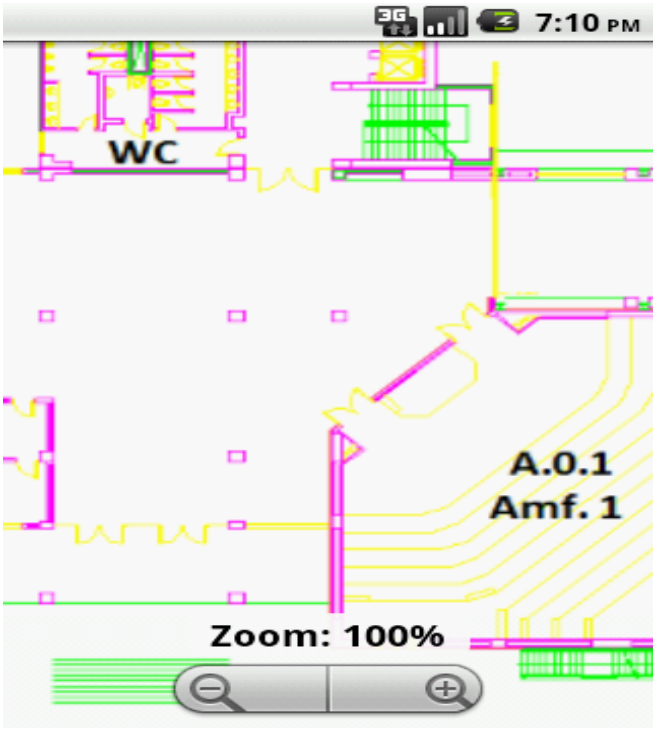
Εικόνα 33



Εικόνα 34



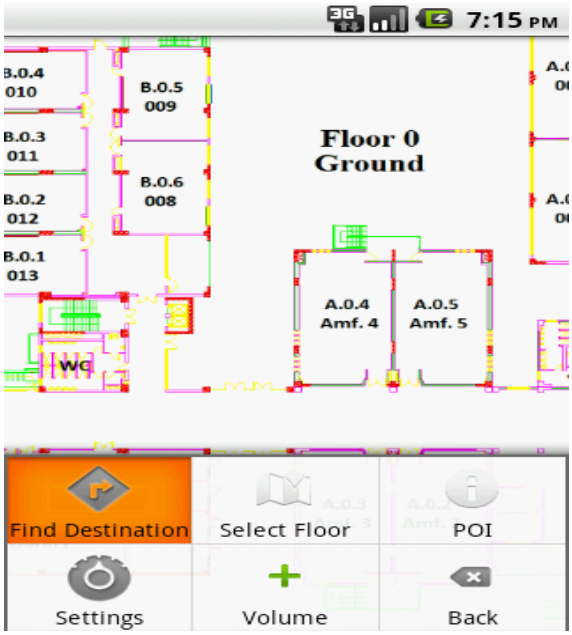
Εικόνα 35



Εικόνα 36

4.5. Επιλογή Έναρξης Διαδικασίας Πλοήγησης

Σε αυτή την ενότητα θα περιγραφεί η διαδικασία πλοήγησης της εφαρμογής. Μία από τις δυνατές επιλογές που έχει ο χρήστης όταν πατήσει το κεντρικό «μενού» είναι η «**Find Destination**» (Εικόνα 37). Με την επιλογή αυτή ξεκινάει η διαδικασία της πλοήγησης στα Νέα κτήρια της σχολής ΗΜΜΥ.

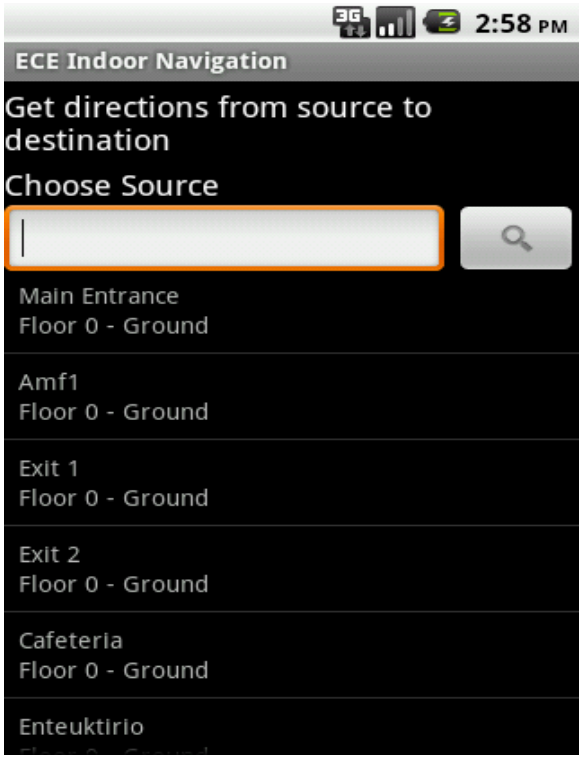


Εικόνα 37

Περιληπτικά, η διαδικασία της πλοήγησης πραγματοποιείται ως εξής:

- Αρχικά, ο χρήστης μόλις πατήσει την επιλογή «**Find Destination**», μεταφέρεται στο παράθυρο εκχώρησης της θέσης τους, στην αρχή μάλιστα ακούγονται φωνητικές οδηγίες οι οποίες είναι ταυτόχρονα και οπτικό μήνυμα «Where are you? Choose your position». Ο χρήστης γράφει στο κουτί αναζήτησης (search box) την τοποθεσία του. Σε περίπτωση που δεν ξέρει πώς να θέσει τη θέση του ή αν δε του βγάζει κάποιο αποτέλεσμα η αναζήτηση, πατώντας το κουμπί αναζήτησης με κενό το κουτί αναζήτησης, εμφανίζονται όλες οι δυνατές επιλογές που είναι καταχωρημένες στη βάση δεδομένων (Εικόνα 38).

- Επισημαίνεται πως τα μικρά ή κεφαλαία γράμματα δεν παίζουν ρόλο στην αναζήτηση. Αυτό που παίζει ρόλο είναι η σειρά με την οποία θα γράψει ο χρήστης την τοποθεσία του. Για παράδειγμα, αν έγραφε «Mobile lab» δεν θα έβγαζε τίποτα. Επίσης, ο χρήστης ότι γράφει στο search box, πραγματοποιείται αναζήτηση πάνω σε αυτό που έχει γράψει δηλαδή αν πατήσει τη λέξη - ενωμένα γράμματα «la», θα του εμφανίσει όλες τις επιλογές που περιλαμβάνουν τη λέξη αυτή μέσα στις λέξεις, όπως χαρακτηριστικά παρουσιάζει η Εικόνα 39. Δεν παίζει ρόλο αν θα ξεκινάει με αυτά η λέξη ή αν περιέχονται μέσα σε αυτές.



Εικόνα 38

Εικόνα 39

- Ύστερα, πατώντας την τωρινή του θέση από τη λίστα, η οποία του εμφανίζεται από τη βάση δεδομένων, μεταφέρεται σε μια άλλη κλάση. Στην κλάση αυτή, αρχικά, του εμφανίζεται ένα οπτικό και φωνητικό μήνυμα, στο οποίο του λέει «You are in the (την επιλογή που έκανε). Where do you want to go? ». Ο χρήστης μπορεί να διαλέξει το μέρος στο οποίο θέλει να πάει. Πάλι με την ίδια διαδικασία που επέλεξε την τοποθεσία του, επιλέγει τον προορισμό.

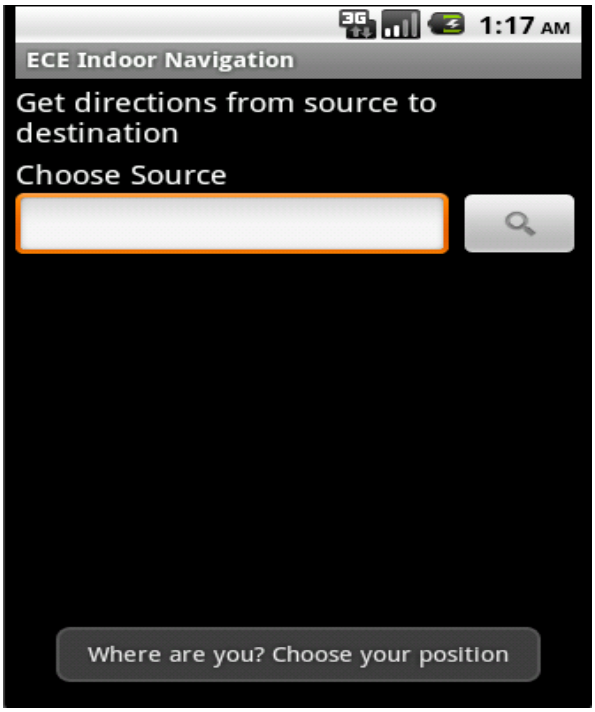
- Σε αυτό το σημείο, επιλέγοντας ο χρήστης την τοποθεσία στην οποία θέλει να βρεθεί, μεταφέρεται στο στάδιο της πλοήγησης. Εμφανίζονται τα οπτικά μηνύματα «Chose (η επιλογή προορισμού)» και τον όροφο στον οποίο είναι και ξεκινάει η πλοήγηση.

Για να γίνει πιο κατανοητό, θα παρουσιαστούν δύο παραδείγματα πλοήγησης.

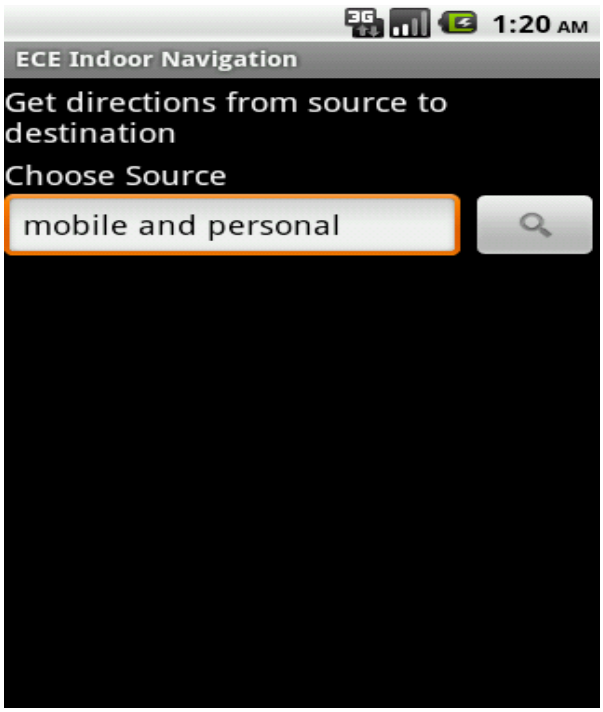
1) Τοποθεσία χρήστη: «**Mobile and Personal Communications Lab - B.2.2**»

Προορισμός : «**Undergraduate Secretariat G.1.1**»

Αρχικά λοιπόν ο χρήστης πατάει στο κεντρικό «μενού» του κινητού και βλέπει τις επιλογές τις εφαρμογής. Πατάει στο «**Find Destination**» όπως φαίνεται προηγούμενη Εικόνα 37 και μεταφέρεται στην κλάση επιλογής της τοποθεσίας του. Εμφανίζεται το οπτικό μήνυμα «Where are you? Choose your position» με την αντίστοιχη φωνητική οδηγία (Εικόνα 39). Έπειτα, ο χρήστης γράφει στο search box «**mobile and personal**» (Εικόνα 40) και πατάει το κουμπί αναζήτησης και του εμφανίζεται η επιλογή «**Mobile and Personal Communications Lab - B.2.2**».

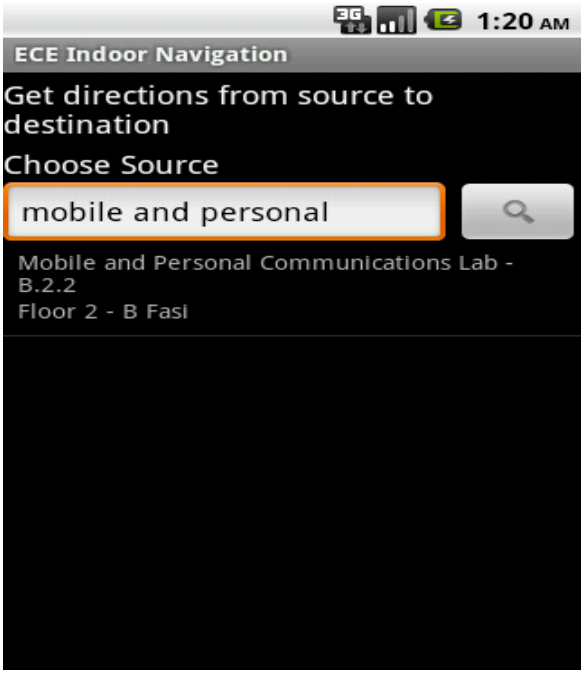


Εικόνα 41



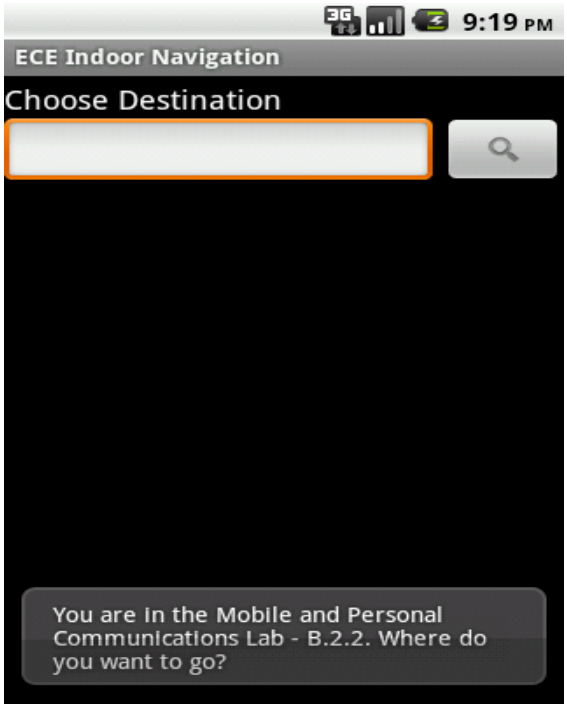
Εικόνα 40

Στην διπλανή εικόνα (Εικόνα 42) παρατηρείται ότι μαζί με το όνομα «**Mobile and Personal Communications Lab – B.2.2**» υπάρχει και το «**B.2.2**» το οποίο αναφέρεται στον αριθμό της αίθουσας του εργαστηρίου. Επιπλέον, κάτω από το όνομα, εμφανίζεται σαν υποσημείωμα (sub item) «**Floor 2 – B Fasi**» τα οποία δείχνουν τον όροφο και σε ποια φάση των κτηρίων είναι το εργαστήριο (τα Νέα κτήρια έχουν Α και Β φάση).

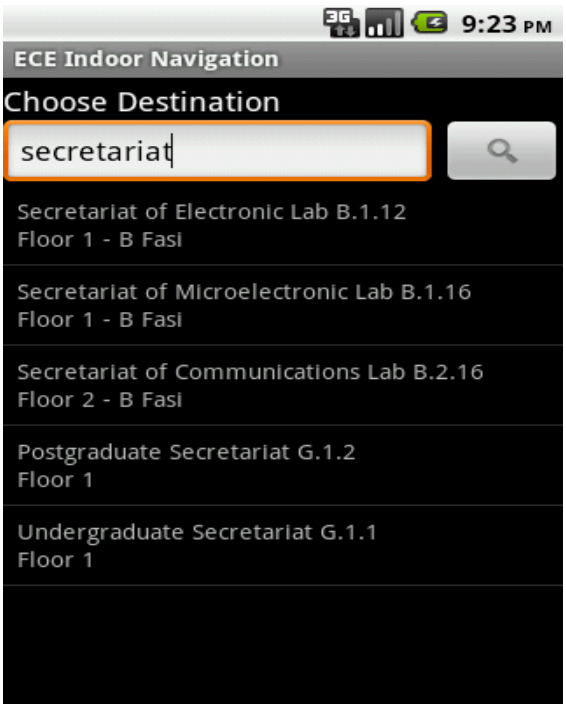


Εικόνα 42

Στη συνέχεια, μεταβαίνει στην επόμενη κλάση (Εικόνα 43), στην οποία διαλέγει το μέρος που θέλει να πάει. Μόλις μεταβεί, εμφανίζεται αλλά και ακούγεται το εξής μήνυμα «**You are in the Mobile and Personal Communications Lab - B.2.2. Where do you want to go?** ». Ο χρήστης λοιπόν διαλέγει τον προορισμό της πλοήγησης. Στην περίπτωση αυτή επιλέγει ότι θέλει να πάει στη γραμματεία των προπτυχιακών. Γράφει «**secretariat**» και του εμφανίζονται όλες οι γραμματείες (Εικόνα 44). Ο χρήστης επιλέγει τη γραμματεία η οποία γράφεται ως «**Undergraduate Secretariat G.1.1**».

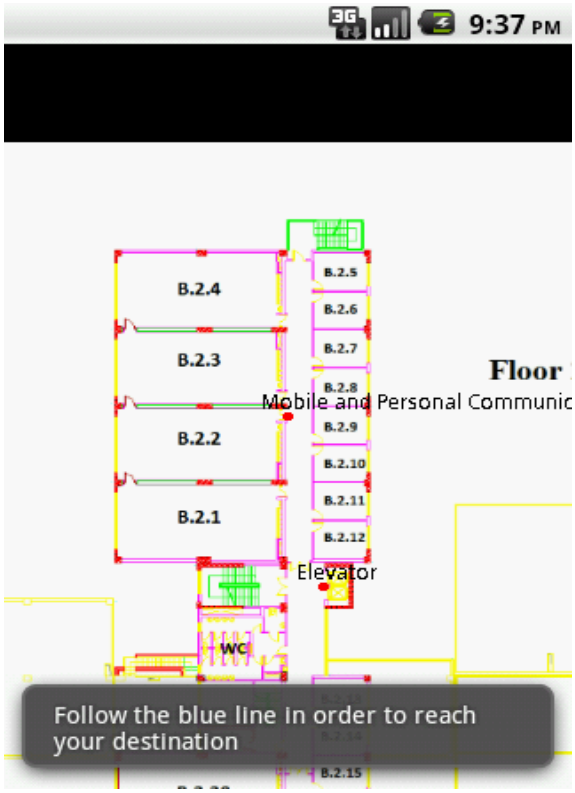


Εικόνα 43

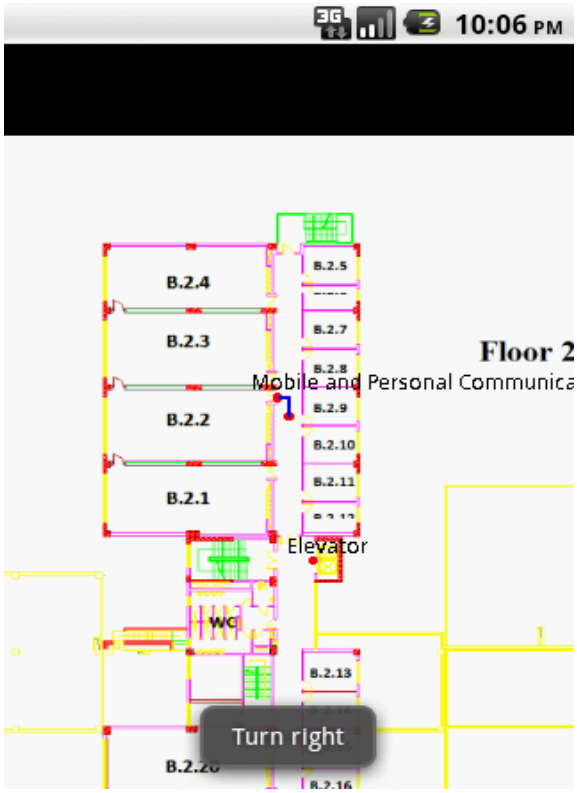


Εικόνα 44

Το επόμενο στάδιο είναι η πλοήγηση. Κατά τη μετάβαση στην άλλη κλάση εμφανίζεται η επιλογή του προορισμού που έκανε «**You chose Undergraduate Secretariat G.1.1**». και μετά σε ποιον όροφο είναι, δηλαδή στον όροφο 2 «**Floor 2**». Έπειτα, εμφανίζεται ο χάρτης του ορόφου που δήλωσε ότι βρίσκεται ο χρήστης, και πάνω στο χάρτη φαίνεται με μια κόκκινη κουκίδα η τοποθεσία του και από πάνω το όνομα του μέρους που διάλεξε. Επίσης, αν στον ίδιο όροφο είναι ο προορισμός, τότε θα εμφανίζεται και αυτό με τον ίδιο τρόπο. Μάλιστα, αν πρόκειται να χρησιμοποιηθούν σκάλες (stairs) ή ασανσέρ (elevator) τότε εμφανίζεται και αυτό στο χάρτη για να διευκολύνει το χρήστη ώστε να ξέρει από πριν πως θα μεταβεί στον προορισμό του. Στην αρχή της πλοήγησης, εμφανίζεται ένα μήνυμα το οποίο λέει «**Follow the blue line in order to reach your destination**», δηλαδή κατά τη διάρκεια της πλοήγησης ζωγραφίζεται σταδιακά μια μπλε γραμμή η οποία στην άκρη της έχει μια κόκκινη κουκίδα η οποία αντιπροσωπεύει την τοποθεσία του χρήστη, όπως φαίνεται στην Εικόνα 45.



Εικόνα 45

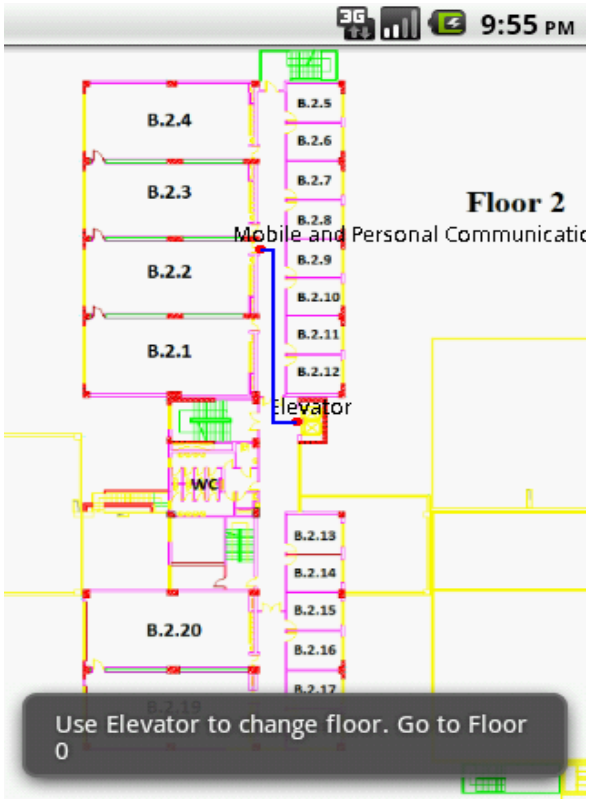


Εικόνα 46

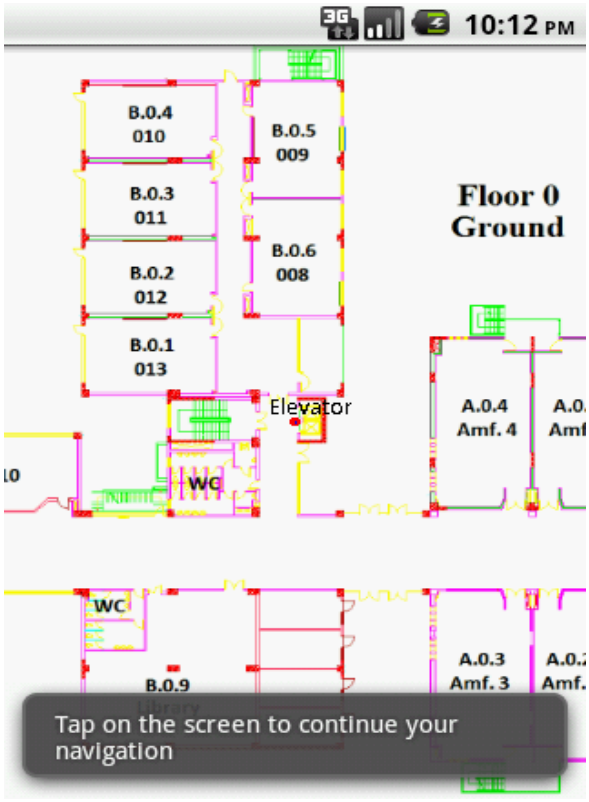
Στην Εικόνα 46 εμφανίζεται η μπλε γραμμή με την κόκκινη κουκίδα, όπως αναφέρθηκε προηγουμένως, όπως και οι κουκίδες και τα ονόματα της αρχικής τοποθεσίας του χρήστη και το μέσο που θα χρησιμοποιήσει για να φτάσει στον προορισμό του. Στη συγκεκριμένη περίπτωση είναι «**Elevator**». Επίσης, κάθε φορά που αλλάζει πορεία ο χρήστης προς τα δεξιά ή προς τα αριστερά εμφανίζονται τα μηνύματα «**Turn right**» και «**Turn left**» αντίστοιχα.

Ο χρήστης μόλις φτάσει στο ασανσέρ εμφανίζεται το εξής μήνυμα «**Use Elevator to change floor. Go to Floor 0**» (Εικόνα 47). Ο χρήστης χρησιμοποιεί ασανσέρ για να μεταβεί από τον όροφο που βρίσκεται ($2^{ο}$) στο ισόγειο. Έπειτα, αλλάζει ο χάρτης και εμφανίζεται ο χάρτης του ισογείου. Η πλοήγηση σταματάει γιατί μέχρι να μεταβεί ο χρήστης από τον $2^{ο}$ όροφο στο ισόγειο θα χρειαστεί κάποιο χρόνο. Για αυτό το λόγο και εμφανίζεται το μήνυμα που λέει «**Tap on the screen to continue your navigation**» (Εικόνα 48). Συνεπώς ο χρήστης πατάει πάνω στην οθόνη (single Tap up) και ξεκινάει η πλοήγηση. Ο χρήστης καθ' όλη τη διάρκεια της πλοήγησης μπορεί να κάνει single Tap up στην οθόνη και να σταματήσει η πλοήγηση και να ξανακάνει single Tap up και να συνεχίσει. Κάθε φορά που το κάνει αυτό, εμφανίζεται το μήνυμα «**Pause**» όταν είναι να σταματήσει (Εικόνα 49) και «**Resume**» όταν είναι να συνεχίσει (Εικόνα 50). Αυτό είναι μια πολύ χρήσιμη επιλογή που δίνει τη δυνατότητα στο χρήστη να σταματήσει την πλοήγηση και να αναζητήσει κάτι στο χάρτη είτε επειδή αλλάζει όροφο είτε επειδή κάτι έγινε και τον παρεμποδίζει να συνεχίσει προς τον προορισμό του είτε αν πηγαίνει γρήγορα η πλοήγηση. Για την τελευταία περίπτωση, είχε αναφερθεί στις ρυθμίσεις ότι ο χρήστης έχει τη δυνατότητα να πατήσει εμφάνιση του κεντρικό μενού και να πάει στα «**Settings**» και να πατήσει πάνω στην επιλογή «**Speed**» και να επιλέξει την ταχύτητα με την οποία θα γίνεται η πλοήγηση. Αυτό που θα αλλάξει είναι ότι το σύστημα θα ζωγραφίζει πιο γρήγορα ή πιο αργά αν ο χρήστης επιλέξει «**fast**», «**very fast**» ή «**slow**», «**very slow**» αντίστοιχα. Η

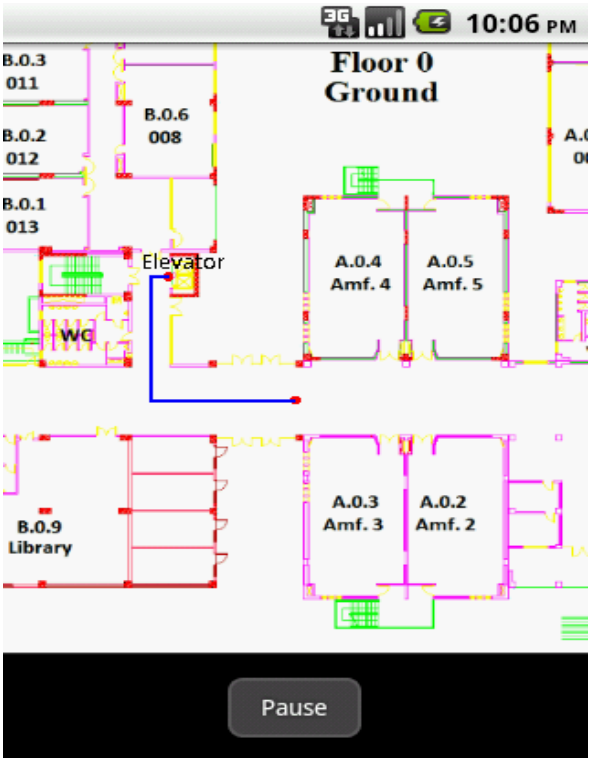
προεπιλεγμένη ταχύτητα «**normal**» θεωρείται η μέση ταχύτητα του ανθρώπου. Στις τέσσερις παρακάτω εικόνες φαίνονται όσα περιγράφτηκαν.



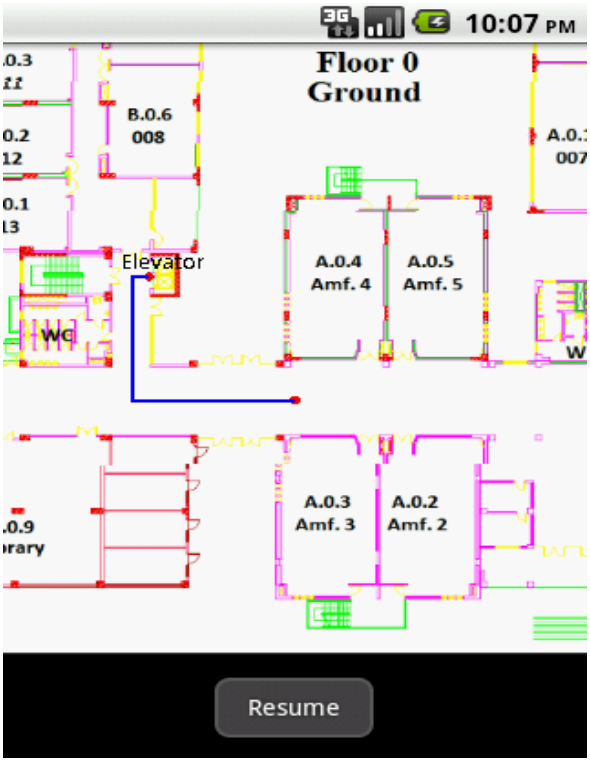
Εικόνα 47



Εικόνα 48

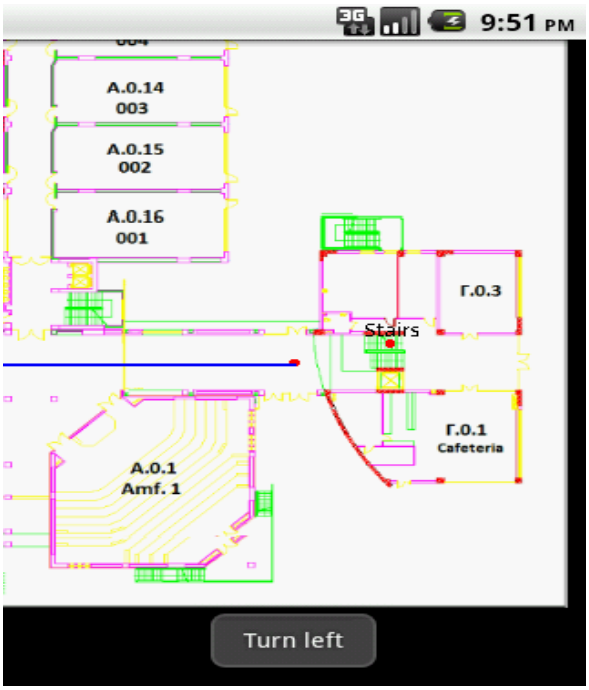


Εικόνα 49

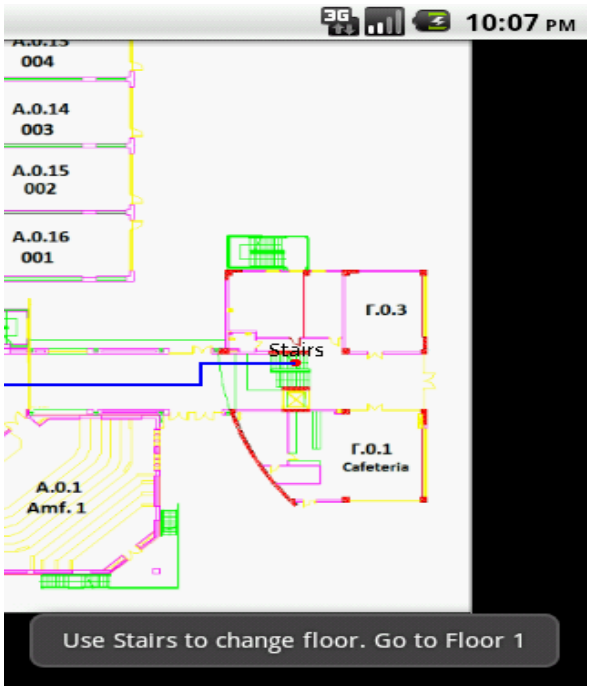


Εικόνα 50

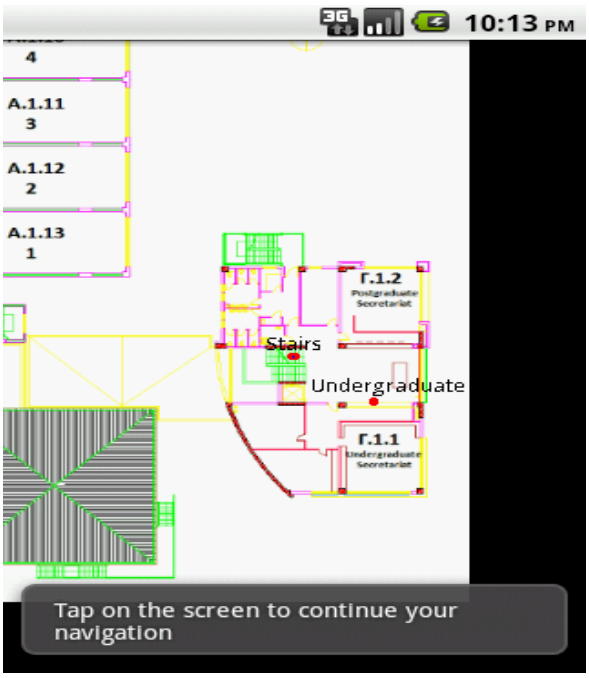
Ο χρήστης πλησιάζει προς τον προορισμό του (Εικόνα 51). Αλλά πριν φτάσει πρέπει να χρησιμοποιήσει κάποιες σκάλες. Μόλις φτάσει, το σύστημα τον ενημερώνει με οπτικό-φωνητικό μήνυμα «**Use stairs to change floor. Go to floor 1**» (Εικόνα 52). Πάλι όπως και προηγουμένως με το ασανσέρ, έτσι και τώρα με τις σκάλες αλλάζει ο χάρτης και εμφανίζεται ο χάρτης του $1^{ο}$ ορόφου και ταυτόχρονα σταματάει η πλοήγηση και εμφανίζεται το μήνυμα «**Tap on the screen to continue your navigation**» (Εικόνα 53). Στο χάρτη αυτό φαίνεται ο προορισμός, καθώς ο προορισμός είναι στον $1^{ο}$ όροφο. Ο χρήστης πατάει πάνω στην οθόνη και συνεχίζει η πλοήγηση. Μετά από λίγο ο χρήστης φτάνει στον προορισμό του και το «**Ntuadroid**» τον ενημερώνει ότι έφτασε «**Congratulations, you reached your destination**» (Εικόνα 54).



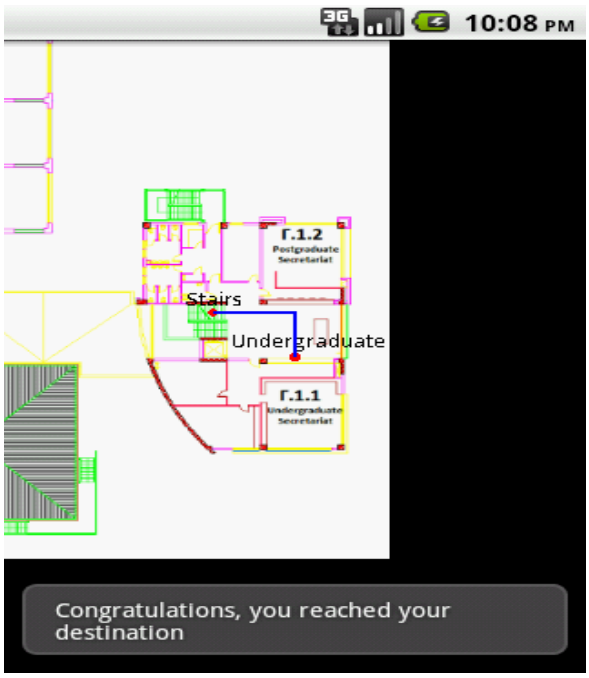
Εικόνα 51



Εικόνα 52



Εικόνα 53



Εικόνα 54

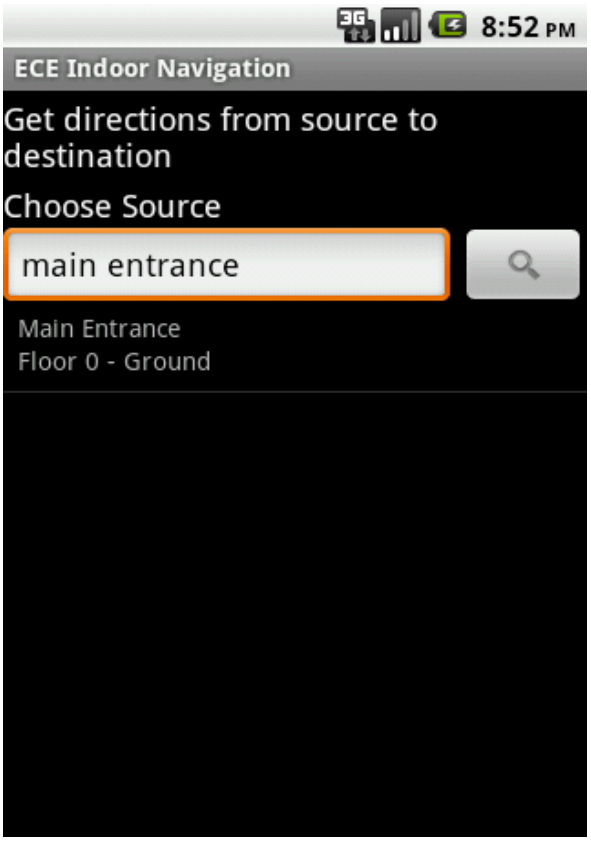
Η πλοήγηση τελείωσε και ο χρήστης έφτασε στον προορισμό του. Αν θελήσει μπορεί να πλοηγηθεί προς άλλο μέρος με την ίδια διαδικασία.

2) Θα περιγραφτεί άλλο ένα παράδειγμα για να γίνει πιο εμφανής η πραγματοποίηση της διαδικασίας της πλοήγησης αλλά πιο συνοπτικά αυτή τη φορά.

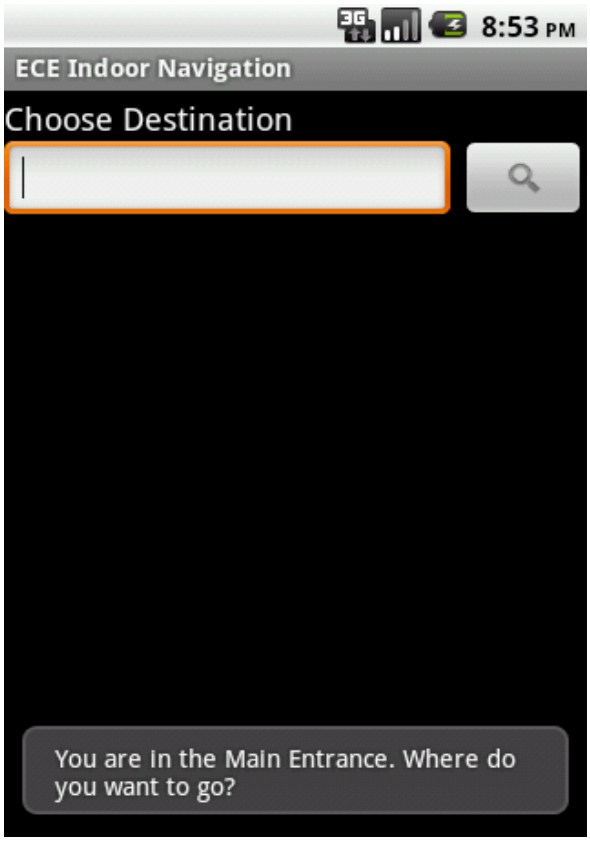
   Τοποθεσία χρήστη: «**Main Entrance**»

   Προορισμός : «**PC LAB A.1.10-4**»

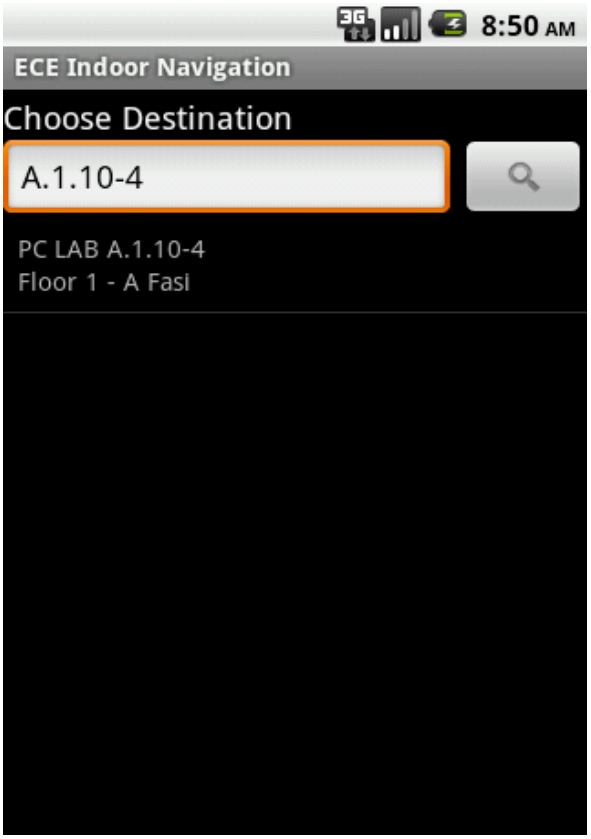
Αρχικά, ο χρήστης βρίσκεται στην κύρια είσοδο των Νέων Κτηρίων. Οπότε ανοίγει την εφαρμογή, πατάει το κουμπί του κινητού για την εμφάνιση του κεντρικού μενού και πατάει στην επιλογή «**Find Destination**». Στο search box της επόμενης κλάσης γράφει «**main entrance**» και του εμφανίζεται από τη βάση δεδομένων ότι ταιριάζει σχετικά με την εκχώρηση που έκανε. Στην Εικόνα 55, εμφανίζεται μόνο μία επιλογή καθώς μόνο μία κύρια είσοδο έχουν τα κτήρια αυτά, στην οποία επιλογή αναγράφεται από κάτω «**Floor 0 – Ground**» που υποδηλώνει ότι είναι στο ισόγειο. Έπειτα αφού το επιλέξει μεταφέρεται στην άλλη κλάση που είναι για την επιλογή του προορισμού με το αντίστοιχο μήνυμα «**You are in the Main Entrance. Where do you want to go?** » (Εικόνα 56). Ο χρήστης θέλει να πάει στο εργαστήριο υπολογιστών, στην αίθουσα Α.1.10-4. Οπότε γράφει στο search box μόνο τον αριθμό της αίθουσας «**A.1.10-4**» για να του το βγάλει κατευθείαν (Εικόνα 57). Θα μπορούσε όμως και να γράψει «**PC LAB**» και να του εμφανιστούν όλα τα PC LAB της σχολής που βρίσκονται στα Νέα Κτήρια και έπειτα να διαλέξει το συγκεκριμένο που θέλει (Εικόνα 58). Παρακάτω φαίνονται σε εικόνες τα αντίστοιχα στιγμιότυπα.



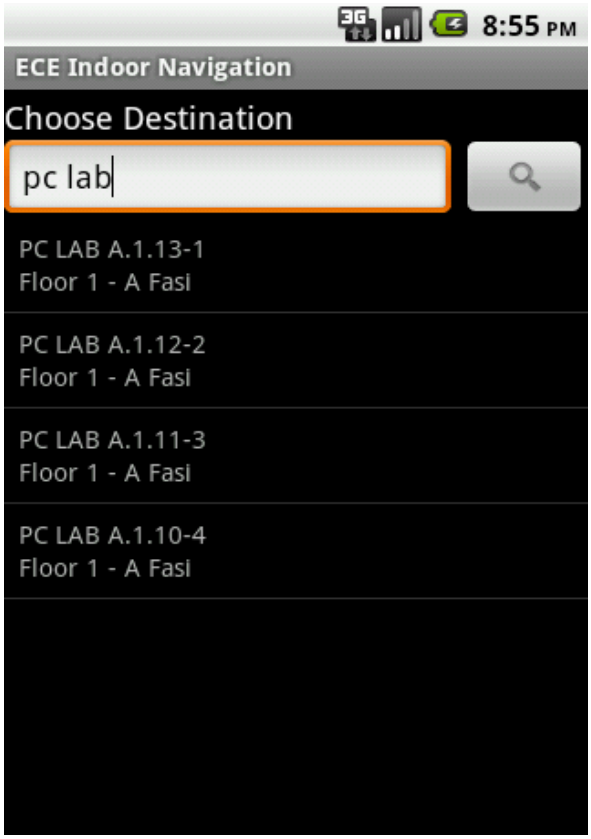
Εικόνα 55



Εικόνα 56

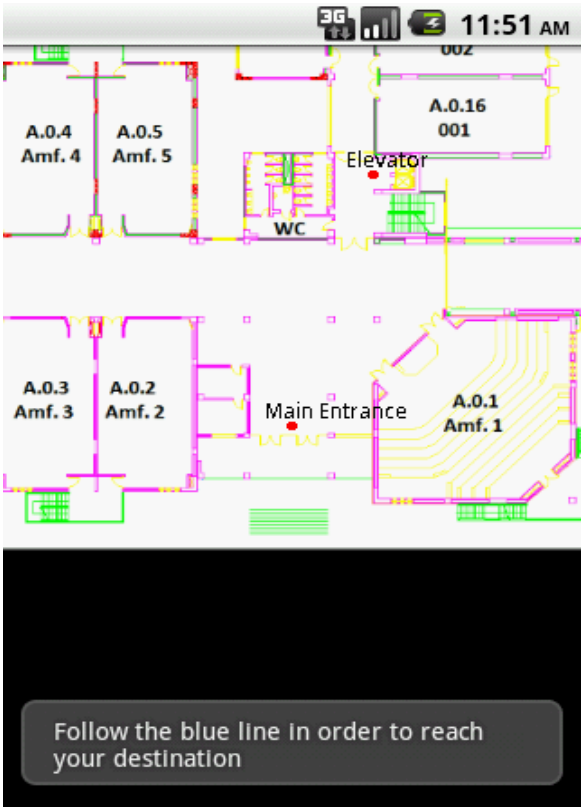


Εικόνα 57

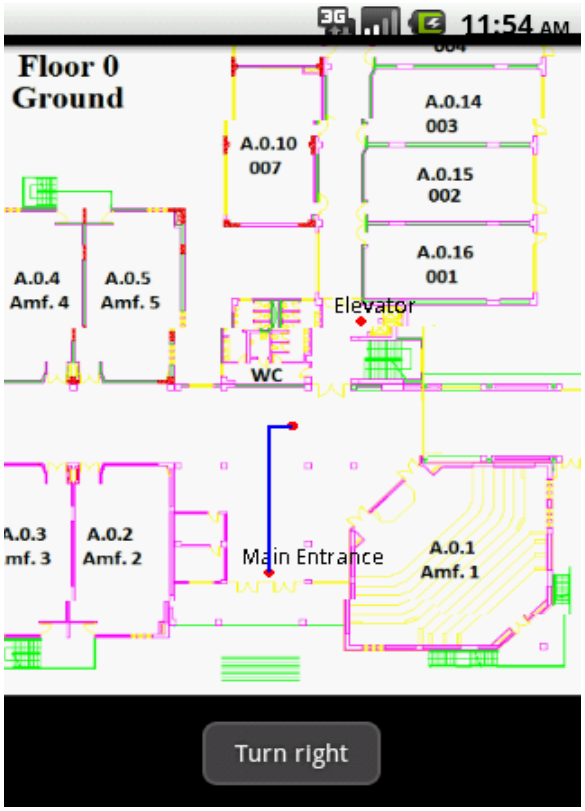


Εικόνα 58

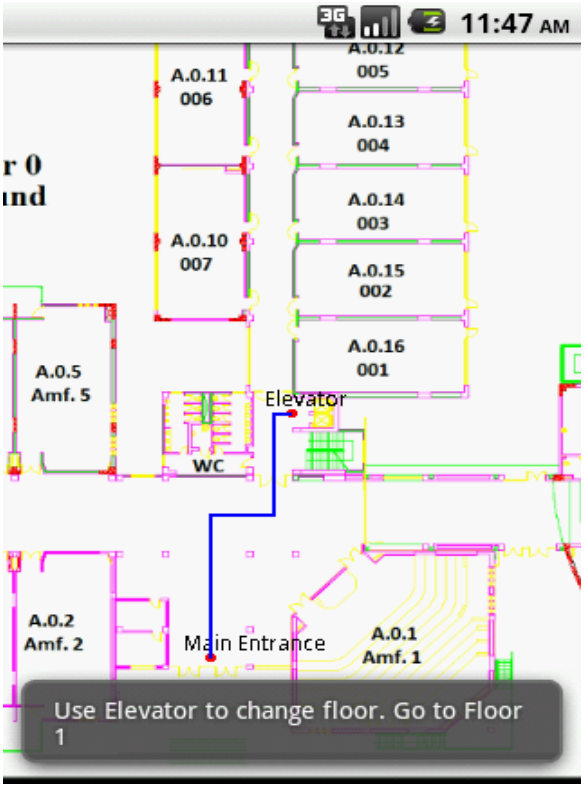
Έπειτα, ο χρήστης μεταφέρεται στον χάρτη για την έναρξη της πλοήγησης με τα κατάλληλα οπτικά μηνύματα να εμφανίζονται και τις αντίστοιχες φωνητικές οδηγίες από το «**Ntuadroid**» να ακούγονται. Στην αρχή ενημερώνει τον χρήστη ότι διάλεξε το «**PC LAB A.1.10-4**», σε ποιον όροφο είναι «**Floor 0**» και ότι πρέπει να ακολουθήσει την μπλε γραμμή για να φτάσει τον προορισμό του «**Follow the blue line in order to reach your destination**» (Εικόνα 59). Στη συνέχεια ξεκινάει η πλοήγηση και παρουσιάζεται ένα στιγμιότυπο στην Εικόνα 60 στο οποίο εμφανίζεται η οδηγία να στρίψει δεξιά ο χρήστης. Κατά τη διάρκεια της πλοήγησης ο χρήστης έχει τη δυνατότητα να αλλάξει την ταχύτητα πλοήγησης όπως και για το αν θέλει να την σταματήσει και να την ξεκινήσει οποιαδήποτε στιγμή θελήσει πατώντας πάνω στην οθόνη (Single Tap up). Επιπλέον, ο χρήστης μπορεί να κάνει zoom in και zoom out για να δει με περισσότερη λεπτομέρεια, με μεγαλύτερη ευκρίνεια το χάρτη και να τον διευκολύνει στην πλοήγηση. Καθώς ο χρήστης ακολουθεί τη μπλε γραμμή στο χάρτη πλησιάζει το ασανσέρ και τότε εμφανίζεται το μήνυμα «**Use Elevator to change floor. Go to floor 1**» (Εικόνα 61) και αυτόματα αλλάζει ο χάρτης και σταματάει η πλοήγηση έτσι ώστε να φτάσει ο χρήστης στον $1^{ο}$ όροφο και να συνεχίσει. Για να συνεχιστεί η πλοήγηση πρέπει να πατήσει πάνω στην οθόνη (Εικόνα 63), άλλωστε εμφανίζεται και το κατάλληλο μήνυμα «**Tap on the screen to continue your navigation**» (Εικόνα 62). Μάλιστα, στον χάρτη του ορόφου αυτού εμφανίζεται και ο προορισμός. Ο χρήστης ακολουθεί τη μπλε γραμμή, σταματάει για λίγο και για αυτό φαίνεται το οπτικό μήνυμα «**Pause**» (Εικόνα 64) και όταν συνεχίζει την πλοήγηση «**Resume**» (Εικόνα 65). Τέλος, μετά από λίγο ο χρήστης φτάνει στον προορισμό του και το «**Ntuadroid**» τον ενημερώνει ότι έφτασε «**Congratulations, you reached your destination**». (Εικόνα 66). Στις ακόλουθες εικόνες παρουσιάζονται στιγμιότυπα καθ' όλη τη διάρκεια της πλοήγησης.



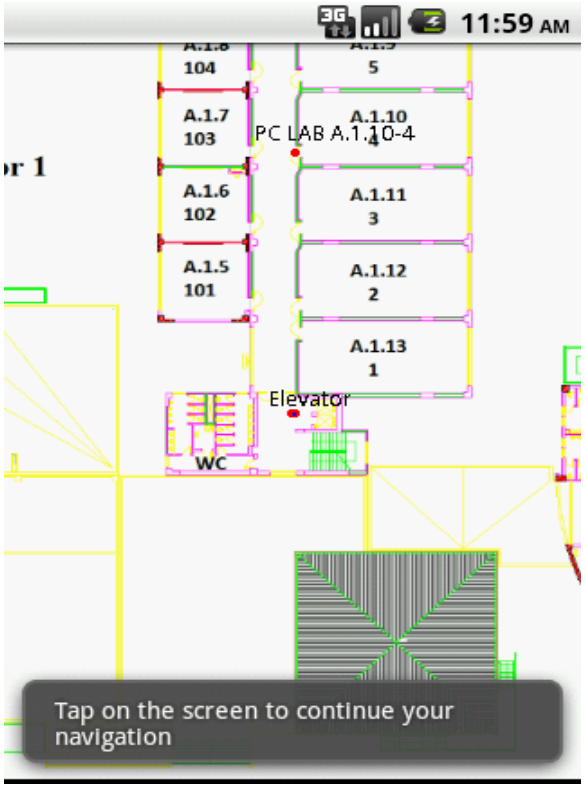
Εικόνα 59



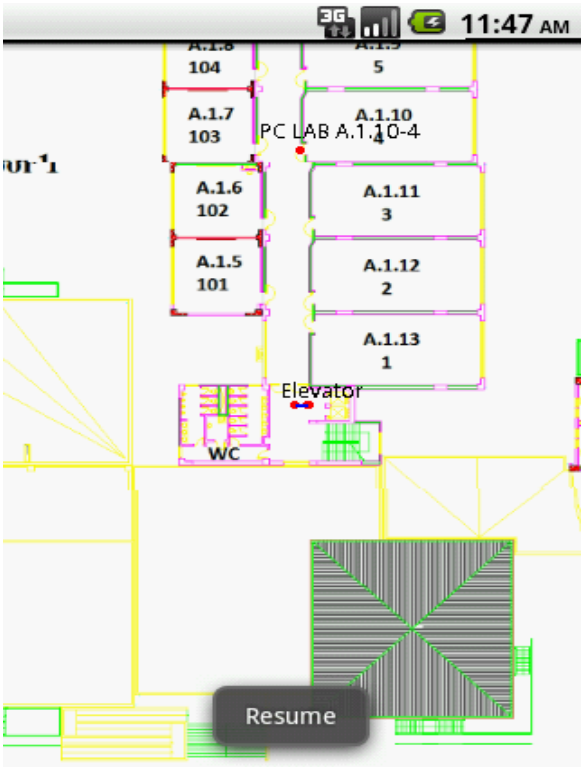
Εικόνα 60



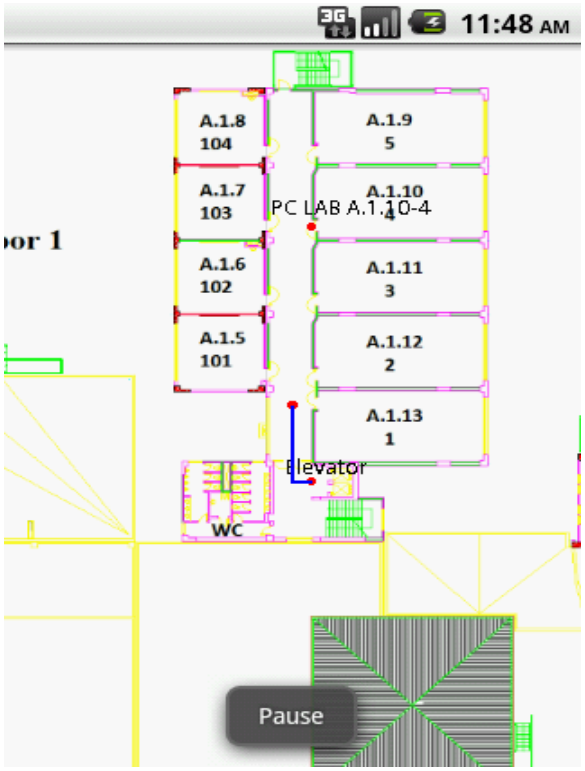
Εικόνα 61



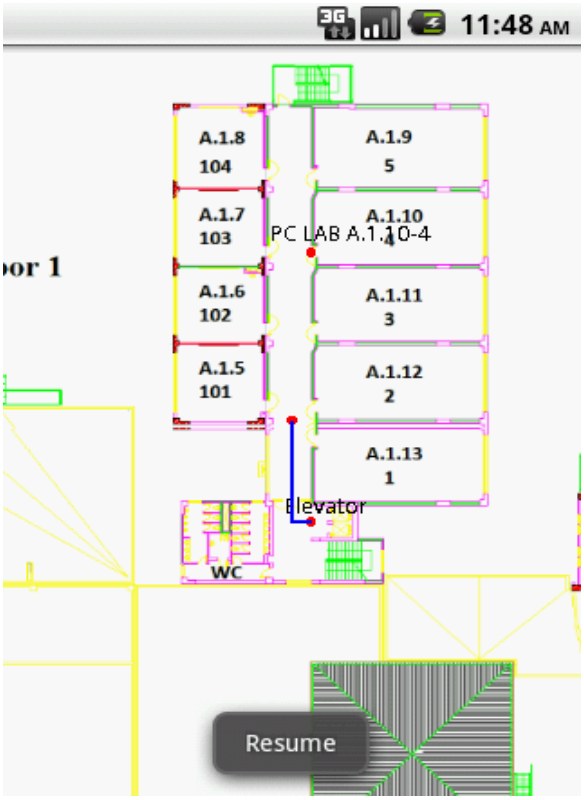
Εικόνα 62



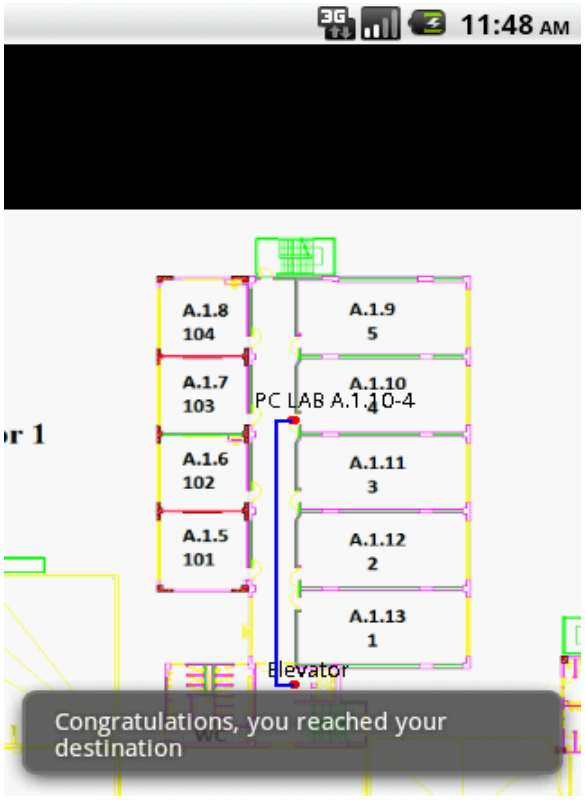
Εικόνα 63



Εικόνα 64



Εικόνα 65



Εικόνα 66

Κεφάλαιο 5

Υλοποίηση της εφαρμογής ECE Indoor Navigation

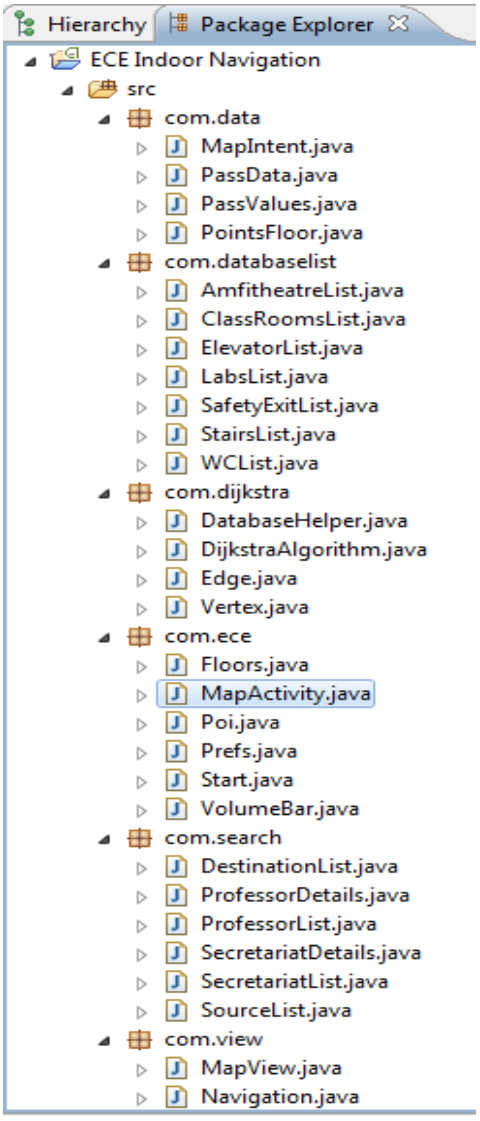
5.1. Εισαγωγή

Ο χρήστης/client έχει τη δυνατότητα χρησιμοποιώντας αυτή την εφαρμογή να πλοηγηθεί στα Νέα Κτήρια της σχολής ΗΜΜΥ. Μάλιστα, ο χρήστης μπορεί να λάβει χρήσιμες πληροφορίες για τις αίθουσες, την γραμματεία και τους καθηγητές, καθώς επίσης μπορεί να δει μέσα από τους χάρτες την τοπολογία των κτηρίων. Τέλος, δίνεται η δυνατότητα στο χρήστη να μπει στις ρυθμίσεις της εφαρμογής και να αποθηκεύσει τις προτιμήσεις του.

Τα δεδομένα ανάμεσα στις κλάσεις μεταφέρονται με την χρήση getters και setters ενώ αντίστοιχα γίνεται και χρήση intents για την μετάβαση από τη μία κλάση στην επόμενη. Το μενού επιλογών υλοποιήθηκε με την χρήση γλώσσας XML ενώ συνδέθηκαν διαδραστικά με εργαλεία Java τα οποία είναι διαθέσιμα στους Android Developers.

5.2. Διάγραμμα Κλάσεων

Στην ενότητα αυτή γίνεται ανάλυση των κλάσεων που απαρτίζουν την εφαρμογή ECE Indoor Navigation, οι οποίες παρουσιάζονται στο παρακάτω σχήμα. Η κάθε μία υλοποιεί μια διαφορετική λειτουργία.



Αρχικά, η έναρξη της εφαρμογής γίνεται μέσω της κλάσης **Start.java**. Η κλάση «τρέχει» την activity η οποία εμφανίζει στην οθόνη του κινητού εικόνα η οποία αναγράφει για ποια σχολή του ΕΜΠ πραγματοποιήθηκε η εφαρμογή και ταυτόχρονα ακούγεται μουσική.

Κατόπιν, ακολουθεί η κύρια κλάση **MapActivity.java**, η οποία αποφασίζει ανάλογα με τα δεδομένα που έχει αν θα καλέσει την **MapView.java** ή την **Navigation.java**. Επίσης, η κλάση MapActivity.java φιλοξενεί και το μενού επιλογών, από το οποίο ο χρήστης μπορεί να εκκινήσει την πλοήγηση, να αλλάξει ο χάρτης (όροφο), να επιλέξει την ένταση των φωνητικών πολυμέσων, να μπει στις ρυθμίσεις της εφαρμογής, να μπει στη λίστα των Points of Interest (POI) και να πάει στην προηγούμενη κλάση (back).

Η κλάση **MapView.java** εμφανίζει στην οθόνη του υπολογιστή τους εσωτερικούς χάρτες των κτηρίων και μέσα από αυτή παρέχονται οι δυνατότητες στον χρήστη για να κάνει zoom in και zoom out όπως και να μετακινεί τον χάρτη προς όποια κατεύθυνση επιθυμεί (scroll).

Η κλάση **Navigation.java** καλείται από την κλάση **MapView.java** όταν ο χρήστης πρόκειται να χρησιμοποιήσει την πλοήγηση της εφαρμογής. Παρέχει τις ίδιες δυνατότητες στον χρήστη σε σχέση με την κλάση MapView αλλά ο χρήστης μπορεί επίσης να σταματήσει και να συνεχίσει την πλοήγηση όσες φορές θέλει πατώντας πάνω στην οθόνη (Single Tap up).

Οι κλάσεις **Floors.java**, **VolumeBar.java**, **Prefs.java**, **Poi.java** και **SourceList.java** αποτελούν επιλογές του κεντρικού μενού. Επίσης υπάρχει και η επιλογή «Back», η οποία επιστρέφει τον χρήστη στην προηγούμενη κλάση.

Η κλάση **Floors.java** ξεκινάει όταν ο χρήστης πατήσει την επιλογή «Select Floor» του κεντρικού μενού. Στη κλάση αυτή, ο χρήστης επιλέγει ποιανού ορόφου θέλει να δει τον χάρτη και μετά την επιλογή επιστρέφει στην MapActivity, η οποία με τα δεδομένα που έχει καλεί την MapView και εμφανίζει τον χάρτη της επιλογής.

Η κλάση **VolumeBar.java** εμφανίζει ένα αναδυόμενο παράθυρο όταν ο χρήστης πατήσει «Volume» στο κεντρικό μενού. Στη κλάση αυτή, ο χρήστης μπορεί να αυξήσει ή να μειώσει την ένταση των ηχητικών πολυμέσων. Κάθε φορά που αυξομειώνει την ένταση ακούγεται η λέξη «sound» για να γνωρίζει ο χρήστης πόσο δυνατά ή χαμηλά ρύθμισε την ένταση του ήχου.

Η κλάση **Prefs.java** μεταφέρει το χρήστη στις ρυθμίσεις του συστήματος της εφαρμογής όταν ο χρήστης πατήσει στην επιλογή «Settings» του κεντρικού μενού. Στη κλάση αυτή, ο χρήστης επιλέγει αν θέλει να ακούγεται μουσική στην έναρξη της εφαρμογής, να ακούγονται οι φωνητικές οδηγίες και επίσης, με ποιον όροφο να ξεκινάει η εφαρμογή και ποια να είναι η ταχύτητα της πλοήγησης. Επιπλέον, έχει πληροφορίες όσον αφορά το σύστημα και πώς να κατεβάσει ο χρήστης την εφαρμογή speechsynthesis η οποία είναι απαραίτητη για την ύπαρξη της φωνητικής ομιλίας (Ntuadroid).

Η κλάση **Poi.java** ξεκινάει όταν ο χρήστης πατήσει την επιλογή «POI» του κεντρικού μενού. Στη κλάση αυτή, υπάρχει μια λίστα από σημεία ενδιαφέροντος (Professors' Offices, Amfitheatre, Classrooms, Secretariat, Library, Labs, Cafeteria, WC, Elevators, Stairs, Exit, Safety Exit). Επιλέγοντας ο χρήστης ένα από αυτά εκκινεί μια άλλη κλάση.

Η κλάση **ProfessorList.java** ενεργοποιείται όταν ο χρήστης επιλέξει το «Professors' Offices» από τα POI. Ο χρήστης μεταφέρεται σε ένα κουτί αναζήτησης, στο οποίο μπορεί να αναζητήσει κάποιο καθηγητή. Οπότε αφού εκχωρήσει το όνομα του καθηγητή, πατάει το κουμπί αναζήτησης και εμφανίζονται τόσα ονόματα από τη βάση δεδομένων όσα ταιριάζουν με την εκχώρηση του χρήστη. Μόλις επιλέξει το όνομα που επιθυμεί, εκκινεί η κλάση **ProfessorDetails.java** στην οποία εμφανίζονται οι πληροφορίες του καθηγητή. Αν πατήσει πάνω στον αριθμό του τηλεφώνου, τον καλεί, όταν πατήσει πάνω στο email μπορεί να επικοινωνήσει μαζί του με το ηλεκτρονικό ταχυδρομείο και μπορεί να επιλέξει να δει τη θέση του πάνω στο χάρτη. Τότε εκκινεί η MapActivity και με τη σειρά της εκκινεί την MapView μεταφέροντας τα απαραίτητα δεδομένα για την εμφάνιση του σημείου πάνω στο χάρτη.

Η κλάση **SecretariatList.java** ενεργοποιείται όταν ο χρήστης επιλέξει το «Secretariat» από τα POI. Όπως και προηγουμένως με την κλάση ProfessorList, η SecretariatList εμφανίζει ένα κουτί αναζήτησης, στο οποίο μπορεί να αναζητήσει τις γραμματείες της σχολής. Οπότε αφού εκχωρήσει το όνομα της γραμματείας, πατάει το κουμπί αναζήτησης και εμφανίζονται

τόσα ονόματα από τη βάση δεδομένων όσα ταιριάζουν με την εκχώρηση του χρήστη. Μόλις επιλέξει αυτό που επιθυμεί, εκκινεί η κλάση SecretariatDetails.java στην οποία εμφανίζονται οι πληροφορίες της γραμματείας. Αν πατήσει πάνω στον αριθμό του τηλεφώνου, την καλεί, όταν πατήσει πάνω στο email μπορεί να επικοινωνήσει μαζί με τα άτομα της γραμματείας με το ηλεκτρονικό ταχυδρομείο και τέλος, μπορεί να επιλέξει να δει τη θέση της πάνω στο χάρτη. Τότε εκκινεί η MapActivity και με τη σειρά της εκκινεί την MapView μεταφέροντας τα απαραίτητα δεδομένα για την εμφάνιση του σημείου πάνω στο χάρτη.

Η κλάση **SourceList.java** ενεργοποιείται όταν ο χρήστης επιλέξει το «Find Destination» από το κεντρικό μενού. Κατά την εκκίνηση της, καλείται η κλάση **DatabaseHelper.java**. Η DatabaseHelper διαβάζει το sqlall.xml αρχείο και το μετατρέπει στην SQLiteDatabase db. Έπειτα, στην κλάση SourceList ο χρήστης επιλέγει την θέση του μέσα από τη βάση δεδομένων και αφού την επιλέξει μεταφέρεται στην επόμενη κλάση, την **DestinationList.java**. Στην κλάση αυτή, ο χρήστης επιλέγει τον προορισμό και μετά την επιλογή, στην κλάση αυτή καλούνται εσωτερικά οι κλάσεις **DijkstraAlgorithm.java**, **Edge.java**, **Vertex.java**. Αφού ολοκληρώσει ο αλγόριθμος δρομολόγησης του υπολογισμούς και βρει τη συντομότερη διαδρομή, καλείται η MapActivity και εν συνεχεία αυτή την Navigation και ξεκινάει η πλοήγηση.

Οι κλάσεις **AmfitheatreList.java**, **ClassroomsList.java**, **ElevatorList.java**, **StairsList.java**, **LabsList.java**, **SafetyExitList.java**, **WCList.java** βρίσκονται στη λίστα των POI.

- Με την κλάση **AmfitheatreList.java** δίνεται η δυνατότητα στο χρήστη να διαλέξει ένα από τα αμφιθέατρα και να δει την θέση του στο χάρτη.

- Με την κλάση **ClassroomsList.java**, ο χρήστης μπορεί να επιλέξει από μια λίστα αιθουσών και να δει την τοποθεσία του.

- Με τις κλάσεις **ElevatorList.java** και **StairsList.java**, ο χρήστης μπορεί να δει που υπάρχουν ασανσέρ και σκάλες σε κάθε όροφο.

- Η κλάση **LabsList.java** περιλαμβάνει όλα τα εργαστήρια της σχολής στα Νέα Κτήρια και ο χρήστης μπορεί να δει ποια είναι και επιλέγοντας κάποιο από αυτά βλέπει τη θέση του στο χάρτη.

- Η κλάση **SafetyExitList.java** εμφανίζει όλες τις εξόδους ασφαλείας των κτηρίων.

- Η κλάση **WCList.java** δίνει τη δυνατότητα στο χρήστη να δει που υπάρχουν τουαλέτες σε κάθε όροφο.

Οι κλάσεις **MapIntent.java**, **PassData.java**, **PassValues.java**, **PointsFloor.java** ενεργοποιούνται όταν χρειάζεται να μεταφερθούν δεδομένα από τη μία κλάση στην άλλη. Η MapIntent χρειάζεται όταν είναι για να μεταφερθούν δεδομένα σχετικά με συντεταγμένες θέσης ενώ οι άλλες τρεις κλάσεις περιλαμβάνουν getters και setters για την μεταφορά πληροφοριών όπως στην περίπτωση των κλάσεων ProfessorDetails και SecretariatDetails.

Στις υποενότητες που ακολουθούν γίνεται περιγραφή των βασικών τμημάτων της εφαρμογής καθώς και των κλάσεων που απασχολούν.

5.3. Η διαδικασία έναρξης της εφαρμογής

Η εφαρμογή ξεκινάει με την εμφάνιση μιας εικόνας και ταυτόχρονα ακούγεται ένα τραγούδι. Όπως φαίνεται στον παρακάτω κώδικα, αρχικά φορτώνεται το τραγούδι song.mp3 από τον φάκελο /r/raw. Με τα SharedPreferences λαμβάνεται (getPrefs) η προτίμηση του χρήστη για το αν θέλει να ακουστεί το τραγούδι. Το περιεχόμενο του key «checkboxMusic» αποθηκεύεται στη μεταβλητή Boolean «music». Αν η τιμή είναι αληθής τότε το τραγούδι ξεκινάει να παίζεται (ourSong.start()) και μπαίνει σε ένα thread το οποίο ξεκινάει και τρέχει για τέσσερα δευτερόλεπτα (sleep(4000)). Αφού περάσει ο χρόνος μεταφέρεται με τη χρήση intent στην MapActivity class.

```
ourSong = MediaPlayer.create(Start.this, R.raw.song);

SharedPreferences getPrefs =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
            boolean music = getPrefs.getBoolean("checkboxMusic", true);
            if (music == true)
                  ourSong.start();

            Thread timer = new Thread() {
                  public void run() {
                        try {
                              sleep(4000);
                              System.gc();
                        } catch (InterruptedException e) {
                              e.printStackTrace();
                        } finally {
                              startActivity(new
                              Intent("com.ece.MAPACTIVITY"));
                        }
                  }
            };
            timer.start();
      }



      @Override
      protected void onPause() {
            // TODO Auto-generated method stub
            super.onPause();
            ourSong.release();
            System.gc();
            finish();
      }
```

Επίσης, η εικόνα έχει οριστεί ως background με την εντολή setContentView(R.layout.start). Στον φάκελο /r/layout είναι αποθηκευμένη η εικόνα start.img, η οποία καλείται μέσω της setContentView και η οποία είναι υπεύθυνη για το τι θα εμφανιστεί στην οθόνη.

5.4. Υλοποίηση των χαρτών

Πολύ σημαντικό τμήμα της εφαρμογής είναι οι χάρτες των κτηρίων και οι δυνατότητες που δίνονται στο χρήστη για να τους επεξεργαστούν (zoom, scroll). Στον κώδικα που ακολουθεί, παρατηρείται ο τρόπος με τον οποίο το σύστημα εμφανίζει τους χάρτες αλλά και πως οι διάφορες δυνατότητες που παρέχονται στο χρήστη κατασκευάστηκαν.

Αρχικά, από την κύρια κλάση από τα δεδομένα που παίρνει το σύστημα από το χρήστη θα καλέσει και τις αντίστοιχες κλάσεις. Υπάρχει η συνάρτηση showFloor στην οποία καθορίζεται ο όροφος που θα δείξει η εφαρμογή στην οθόνη του κινητού. Όταν πρόκειται ο χρήστης να μπει στους χάρτες του συστήματος και να τους επεξεργαστεί και να πάρει πληροφορίες για αυτόν τότε μέσω της setContentView η MapView είναι αυτή που καλείται. Καθώς έχει οριστεί ότι όταν είναι για να δει ο χρήστης τους χάρτες, η μεταβλητή Boolean isView είναι αληθής (true), αλλιώς να είναι ψευδής (false) όταν είναι για τη λειτουργία της πλοήγησης και κατόπιν καλείται η κλάση Navigation.

```
public void showFloor(int floor) {

            if (isView) {
                  if (drawpoints == null) {
                        int x = 0;
                        int y = 0;
                        nametxt = null;
                        setContentView(new MapView(this, null, floor, x,
                        y, nametxt));
                  } else {
                        setContentView(new  MapView(this,  null,  floor,
                        drawpoints.getcoX(),drawpoints.getcoY(),
                        drawpoints.getS()));
                  }
            } else {
                  view = new Navigation(this, null, x1, y1, z1, x2, y2,
                  z2, xs, ys, zs, name_source, name_destination, this);
                  setContentView(view);
            }
}
```

Παρακάτω θα περιγραφθούν μερικές δυνατότητες που έχει ο χρήστης κατά τη διάρκεια είτε της πλοήγησης είτε όταν βρίσκεται στην χρησιμοποίηση και επεξεργασία των πληροφοριών βλέποντας τους χάρτες. Οι παρακάτω κώδικες βρίσκονται στις MapView και Navigation κλάσεις.

5.4.1. Zoom in και Zoom out

Ο χρήστης έχει τη δυνατότητα να μεγεθύνει και να μικρύνει τον χάρτη. Για να εμφανιστεί ο zoom controller πρέπει να έχει πατημένη για λίγο την οθόνη (onLongPress). Τότε θα του εμφανιστεί ο zoom controller με ένα πλην (-) και ένα συν (+) για να μικραίνει και να μεγεθύνει αντίστοιχα το χάρτη που βλέπει. Με τη συνάρτηση onZoom υπολογίζεται το πόσο θα μικραίνει ή θα μεγεθύνει ο χάρτης με κάθε πάτημα του χρήστη στις αντίστοιχες επιλογές. Μέσω της συνάρτησης makeZoomLabel καθορίζεται η εμφάνιση του zoom controller. Παρακάτω φαίνεται ο κώδικας της λειτουργίας αυτής.

Ο χρήστης πατώντας πάνω στην οθόνη ενεργοποιείται η συνάρτηση onLongPress και θέτει τον mZoomController να είναι ορατός setVisible(true).

```
public void onLongPress(MotionEvent e) {
            mZoomController.setVisible(true);
            notTouched = false;
      }
```

Στη συνάρτηση onZoom ρυθμίζεται η κλίμακα. Όπως φαίνεται παρακάτω, η mScale ορίζεται να αλλάζει κατά 10% με το πάτημα του κάθε κουμπιού είτε θετικά είτε αρνητικά.

```
public void onZoom(boolean zoomIn) {
            mScale += zoomIn ? 0.1 : -0.1;
            mScale = Math.min(MAX_ZOOM, Math.max(MIN_ZOOM, mScale));
            mZoomLabel.setText("Zoom: " + mZoomFormat.format(mScale));
            System.out.println("onzoom");
            invalidate();

            mZoomController.setZoomInEnabled(mScale != MAX_ZOOM);
            mZoomController.setZoomOutEnabled(mScale != MIN_ZOOM);
      }
```

Στην παρακάτω συνάρτηση, καθορίζεται η εμφάνιση του mZoomController, δηλαδή το πλαίσιο, το μέγεθος των γραμμάτων, το background χρώμα και η θέση του στην οθόνη.

```
private void makeZoomLabel(Context context, ZoomButtonsController
                        zoomController) {
            ViewGroup container = zoomController.getContainer();
            View controls = zoomController.getZoomControls();
            LayoutParams p0 = controls.getLayoutParams();
            container.removeView(controls);
            LinearLayout layout = new LinearLayout(context);
            layout.setOrientation(LinearLayout.VERTICAL);
            mZoomLabel = new TextView(context);
            mZoomLabel.setPadding(12, 0, 12, 0);
            mZoomLabel.setTypeface(Typeface.DEFAULT_BOLD);
            mZoomLabel.setTextColor(0xff000000);
            PaintDrawable d = new PaintDrawable(0xeeffffff);
            d.setCornerRadius(6);
            mZoomLabel.setBackgroundDrawable(d);
            mZoomLabel.setTextSize(20);
            mZoomLabel.setGravity(Gravity.CENTER_HORIZONTAL);
            LinearLayout.LayoutParams p1 = new
                LinearLayout.LayoutParams(LayoutParams.WRAP_CONTENT,
                LayoutParams.WRAP_CONTENT);
            p1.gravity = Gravity.CENTER_HORIZONTAL;
            layout.addView(mZoomLabel, p1);
            layout.addView(controls);
            container.addView(layout, p0);
      }
```

5.4.2. Κίνηση χαρτών - Scroll

Έπειτα, ο χρήστης μπορεί να θελήσει να μετακινήσει το χάρτη προς κάποια κατεύθυνση. Αυτό γίνεται με τη συνάρτηση onScroll. Μάλιστα, υπάρχει και η συνάρτηση onFling με την οποία μπορεί και υπολογίζει την ταχύτητα που μετακινεί ο χρήστης την εικόνα (τον χάρτη) με σκοπό να την διατηρήσει αφού ο χρήστης έχει απομακρύνει το δάχτυλό του από την οθόνη.

```
public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX,
                        float distanceY) {
            mX -= distanceX / mScale;
            mY -= distanceY / mScale;
            mX = Math.max(getWidth() - WIDTH, Math.min(0, mX));
            mY = Math.max(getHeight() - HEIGHT, Math.min(0, mY));
            notTouched = false;
            invalidate();
            return true;
      }

public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
                       float velocityY) {
            int minX = (int) (getWidth() - WIDTH);
            int minY = (int) (getHeight() - HEIGHT);
            mScroller.fling((int) mX, (int) mY, (int) velocityX, (int)
                            velocityY, minX, 0, minY, 0);
            invalidate();
            return true;
      }
```

5.4.3. Εμφάνιση και Επεξεργασία χάρτη

Υπεύθυνη συνάρτηση για τη χρήση και για την οποιαδήποτε επεξεργασία πάνω στον χάρτη είναι η makeContent. Στη συνάρτηση αυτή, μπορεί να ξεκινήσει η φόρτωση της εικόνας του χάρτη όπως φαίνεται στη $12^{η}$ γραμμή του παρακάτω κώδικα και οποιαδήποτε επεξεργασία μπορεί να γίνει μεταξύ των εντολών picture.beginRecording() και picture.endRecording().

```
private Picture makeContent() {
            Picture picture = new Picture();
            Paint paint = new Paint();
            Paint points = new Paint();
            Paint txt = new Paint();
            points.setColor(Color.BLUE);
            points.setStyle(Paint.Style.FILL_AND_STROKE);
            points.setStrokeWidth(10);
            txt.setColor(Color.BLACK);
            txt.setTextSize(20);
            Canvas c = picture.beginRecording(WIDTH, HEIGHT);
            c.drawBitmap(image,0,0,paint);
            picture.endRecording();
            System.gc();
            return picture;

      }
```

Τέλος, υπεύθυνη συνάρτηση για την εμφάνιση όλων αυτών στην οθόνη του υπολογιστή είναι η onDraw. Αφού πάρει όλες τις πληροφορίες από τις παραπάνω συναρτήσεις, επεξεργάζεται τα δεδομένα και εμφανίζει τον canvas στην οθόνη του κινητού.

```
protected void onDraw(Canvas canvas) {
            canvas.save();
            if (mScroller.computeScrollOffset()) {
                  mX = mScroller.getCurrX();
                  mY = mScroller.getCurrY();
                  invalidate();
            }
            mMatrix.reset();
            mMatrix.preTranslate(mX * mScale, mY * mScale);

            int w = getWidth();
            int h = getHeight();
            float pivotX = Math.max(Math.min(-mX, w / 2), 2 * w - WIDTH -
                                                mX);
            float pivotY = Math.max(Math.min(-mY, h / 2), 2 * h - HEIGHT -
                                                mY);
            mMatrix.preScale(mScale, mScale, pivotX, pivotY);
            if (notTouched) {
                  mMatrix.postTranslate(w / 2f - mScale * followX, h / 2f
                                    - mScale * followY);
            }
            canvas.concat(mMatrix);

            // draw content
            mPicture.draw(canvas);
            canvas.restore();
      }
```

5.5. Δημιουργία Κεντρικού Μενού

Ο χρήστης όταν πατήσει το κουμπί του κινητού για το κεντρικό μενού τότε αναδύεται στην οθόνη ένα μενού με 6 επιλογές. Αυτό δημιουργείται στην MapActivity κλάση. Η δημιουργία του μενού και των επιλογών γίνεται χρησιμοποιώντας την συνάρτηση boolean onCreateOptionMenu, όπως φαίνεται παρακάτω.

```
public boolean onCreateOptionsMenu(Menu menu) {
            super.onCreateOptionsMenu(menu);
            menu.add(0, 0, 0, "Find
Destination").setIcon(android.R.drawable.ic_menu_directions);
            menu.add(0, 1, 0, "Select
Floor").setIcon(android.R.drawable.ic_dialog_map);
            menu.add(0, 2, 0,
"POI").setIcon(android.R.drawable.ic_dialog_info);
            menu.add(0, 3, 0,
"Settings").setIcon(android.R.drawable.ic_menu_preferences);
            menu.add(0, 4, 0,
"Volume").setIcon(android.R.drawable.ic_input_add);
            menu.add(0, 5, 0,
"Back").setIcon(android.R.drawable.ic_input_delete);
            return true;
}
```

Όπως παρατηρείται, δημιουργούνται 6 επιλογές με την εντολή

```
menu.add(),
```

στις οποίες αναγράφονται ονόματα, αντιστοιχούνται εικονίδια και θέσεις εμφάνισης στο μενού.

Πατώντας μία από αυτές τις επιλογές, ο χρήστης μεταφέρεται σε μία άλλη κλάση. Η μετάβαση αυτή γί νεται με την συνάρτηση boolean

```
onOptionsItemSelected(MenuItem item),
```

και μετά υπάρχουν οι περιπτώσεις cases, όπου ανάλογα με την επιλογή που θα κάνει (η θέση αποτελεί κριτήριο), θα πραγματοποιηθεί αυτό που υπάρχει στην αντίστοιχη case.

```
public boolean onOptionsItemSelected(MenuItem item) {
            switch (item.getItemId()) {

            case 0:
                  Intent intent = new Intent(this,
                                          com.search.SourceList.class);
                  intent.putExtra("Floor1", z2);
                  startActivity(intent);
                  if (!isView) {
                        isOK = false;
                        finish();
                        break;
                  }
                  finish();
                  return true;

            case 1:
                  Intent intent1 = new Intent(this, com.ece.Floors.class);
                  intent1.putExtra("Floor1", z2);
                  startActivity(intent1);
                  if (!isView) {
                        isOK = false;
                        finish();
                        System.exit(0);
                        break;
                  }
                  finish();
                  return true;

            case 2:
                  Intent intent2 = new Intent(this, com.ece.Poi.class);
                  intent2.putExtra("Floor1", z2);
                  startActivity(intent2);
                  if (!isView) {
                        isOK = false;
                        finish();
                        System.exit(0);
                        break;
                  }
                  finish();
                  return true;

            case 3:
                  onlySpeak("Settings");
```

```
                  Intent intent3 = new Intent(this, com.ece.Prefs.class);
                  intent3.putExtra("Floor1", z2);
                  startActivity(intent3);
                  return true;

            case 4:
                  onlySpeak("Volume");
                  final AlertDialog.Builder alert2 = new
AlertDialog.Builder(this);

                  // get the tool to change the media volume of the phone
                  final AudioManager am = (AudioManager)
getSystemService(Context.AUDIO_SERVICE);
                  double maxVol =
am.getStreamMaxVolume(AudioManager.STREAM_MUSIC);
                  final double curVol =
am.getStreamVolume(AudioManager.STREAM_MUSIC);

                  LinearLayout ll = new LinearLayout(this);
                  ll.setOrientation(LinearLayout.VERTICAL);
                  final VolumeBar b = new VolumeBar(this, tts, am);

                  b.measure(150, 30);
                  b.layout(0, 0, 150, 30);
                  b.forceLayout();
                  b.setPadding(15, 10, 15, 10);

                  // Set up the bar to start at the current volume
                  b.setProgress((int) (b.getMax() * curVol / maxVol));
                  TextView tv = new TextView(this);
                  tv.setText("Speech Volume:");
                  ll.addView(tv);
                  ll.addView(b);
                  alert2.setView(ll);

                  alert2.setPositiveButton("Ok", new
DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int
whichButton) {

                        }
                  });
                  alert2.setNegativeButton("Cancel", new
DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int
whichButton) {
                              dialog.cancel();

      am.setStreamVolume(AudioManager.STREAM_MUSIC, (int) curVol, 0);
                        }
                  });

                  Dialog alertDialog = alert2.create();
                  alertDialog.setTitle("Options");
                  alertDialog.getWindow().setLayout(15, 10);
                  alertDialog.show();

                  return true;

            case 5:
                  onlySpeak("Back");
```

```
                isOK = false;
                finish();
                System.exit(0);
                break;
            }

            return false;
        }
```

Επιλέγοντας, μία από τις παραπάνω επιλογές εμφανίζονται και οι αντίστοιχες κλάσεις. Στις παρακάτω υποενότητες, περιγράφεται αναλυτικά η δομή για την κάθε μία επιλογή.

5.5.1. Ρυθμίσεις (Settings)

Ο χρήστης/client έχει τη δυνατότητα να καθορίσει τις προτιμήσεις του σχετικά με την έναρξη της εφαρμογής π.χ. μουσική, όροφο. Επιπλέον μπορεί να καθορίσει την ταχύτητα της πλοήγησης, την ύπαρξη των φωνητικών ειδοποιήσεων και οδηγιών (Ntuadroid). Αυτό επιτυγχάνεται μέσα από το menu_item «Settings» στην MapActivity class. Η μεταφορά στην κλάση γίνεται με την χρησιμοποίηση των intents.

```
Intent intent3 = new Intent(this, com.ece.Prefs.class);
startActivity(intent3);
```

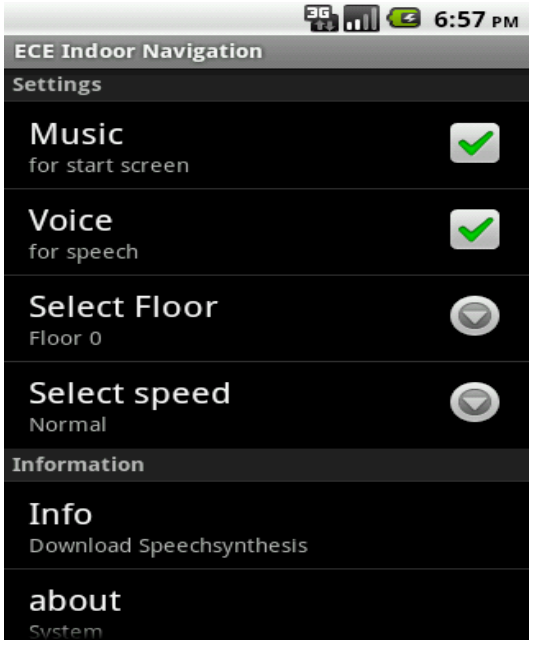
Εκεί έχει δημιουργηθεί ένα Preference Screen Layout, όπου οι επιλογές του χρήστη αποθηκεύονται στη μορφή Shared Preferences. Οι Shared Preferences είναι ένας απλός και ελαφρύς, από θέμα κατανάλωσης πόρων συστήματος, μηχανισμός αποθήκευσης δεδομένων της εφαρμογής. Επιτρέπουν στον χρήστη να αποθηκεύει ζεύγη key/value δεδομένων σαν named preferences και να χρησιμοποιεί σε κάθε κλάση της εφαρμογής του. Το preference key είναι απλά ένα string που ταυτοποιεί μοναδικά τη συγκεκριμένη preference και η preference value είναι η τιμή της preference αυτής.

Οι τύποι δεδομένων που υποστηρίζονται από την SharedPreferences class, μέσω της οποίας μπορούν να δημιουργηθούν τα named preferences, είναι: boolean, float, integer, long, string. Λόγω της δυνατότητας τους αυτής, χρησιμοποιούνται για να αποθηκεύουν γρήγορα default τιμές, class instance variables, την τρέχουσα κατάσταση του user interface και φυσικά τις επιλογές του χρήστη. Επίσης, χρησιμοποιούνται συχνά για να κρατούν μόνιμα δεδομένα κατά τη διάρκεια των user sessions και για να κάνουν διαθέσιμες τις επιλογές αυτές σε κάθε component της εφαρμογής.

Μέσω, λοιπόν, του PreferenceScreen Layout σε .xml αρχείο που καλείται από τη κλάση Prefs.java, καθορίστηκαν τέσσερα ζεύγη Shared Preferences.

1) Ένα με key = “checkboxMusic”, με default value = “true”, καθώς είναι της μορφής «CheckBoxPreference», οπότε έχει ως τιμές «true» ή «false» αν είναι «τικαρισμένο» το κουτί ή όχι αντίστοιχα (Εικόνα 67).

2) Ένα με key = “checkboxVoice”, με default value = “true”, καθώς είναι της μορφής «CheckBoxPreference», οπότε έχει ως τιμές «true» ή «false» αν είναι «τικαρισμένο» το κουτί ή όχι αντίστοιχα (Εικόνα 67).



Εικόνα 67

3) Ένα με key = "floorlist", με default value = "0", και είναι της μορφής «ListPreference» οπότε έχει τα entries="@array/chooseFloor" entryValues="@array/updateFloor". Όπου ο πίνακας «chooseFloor» έχει τις εξής τιμές:

   - Floor 0
   - Floor 1
   - Floor 2
   - Floor 3

Ενώ ο πίνακας «updateFloor» έχει τις τιμές:

   - 0
   - 1
   - 2
   - 3

4) Ένα με key = “speed”, με default value = “30”, και είναι της μορφής «ListPreference» οπότε έχει τα entries=“@array/selectSpeed” entryValues=“@array/updateSpeed”. Όπου ο πίνακας «selectSpeed» έχει τις εξής τιμές:

   - Very Slow
   - Slow
   - Normal
   - Fast
   - Very Fast

Ενώ ο πίνακας «updateSpeed» έχει τις τιμές:

   - 18
   - 24
   - 30
   - 40

Στις περιπτώσεις 3 και 4 που υπάρχει το «ListPreference», κάτω από το «Select Floor» και «Select Speed» υπάρχει ένα sub item «Floor 0» και «Normal» αντίστοιχα, στο οποίο εμφανίζεται η επιλογή του χρήστη όπως φαίνεται στην Εικόνα 67. Ο κώδικας αυτού εμφανίζεται παρακάτω.

```
protected void onResume() {
            super.onResume();
            // Set up a listener whenever a key changes
            getPreferenceScreen().getSharedPreferences().registerOnSharedP
            referenceChangeListener(this);
      }

protected void onPause() {
            super.onPause();
            // Unregister the listener whenever a key changes
            getPreferenceScreen().getSharedPreferences().unregisterOnShare
            dPreferenceChangeListener(this);
      }

public void onSharedPreferenceChanged(SharedPreferences sharedPreferences,
                                      String key) {
            updatePrefSummary(findPreference(key));
      }
```

```
private void initSummary(Preference p) {
            if (p instanceof PreferenceCategory) {
                  PreferenceCategory pCat = (PreferenceCategory) p;
                  for (int i = 0; i < pCat.getPreferenceCount(); i++) {
                        initSummary(pCat.getPreference(i));
                  }
            } else {
                  updatePrefSummary(p);
            }

      }

      private void updatePrefSummary(Preference p) {
            if (p instanceof ListPreference) {
                  ListPreference listPref = (ListPreference) p;
                  p.setSummary(listPref.getEntry());
            }
            if (p instanceof EditTextPreference) {
                  EditTextPreference editTextPref = (EditTextPreference)p;
                  p.setSummary(editTextPref.getText());
            }
      }
```

Τέλος, υπάρχουν στην ενότητα «Information» οι επιλογές «Info» και «about» τα οποία εμφανίζονται με τη συνάρτηση InstructionsDialog(). Παρακάτω παρατίθεται ο κώδικας:

```
Preference info = (Preference) findPreference("info");
Preference about = (Preference) findPreference("about");

            info.setOnPreferenceClickListener(new
                                    OnPreferenceClickListener() {

                  public boolean onPreferenceClick(Preference preference)
                    {
                        i = 1;
                        InstructionsDialog();
                        return true;
                    }
            });

            about.setOnPreferenceClickListener(new
                                    OnPreferenceClickListener() {

                  public boolean onPreferenceClick(Preference preference)
                    {
                        i = 2;
                        InstructionsDialog();
                        return true;
                    }
            });

            PreferenceManager.setDefaultValues(Prefs.this, R.xml.prefs,
                                          false);

            for (int i = 0; i <
               getPreferenceScreen().getPreferenceCount(); i++) {
                  initSummary(getPreferenceScreen().getPreference(i));
            }
      }
```

```
public void InstructionsDialog() {
            AlertDialog.Builder ad = new AlertDialog.Builder(this);
            ad.setIcon(R.drawable.icon);
            if (i == 1) {
                  ad.setTitle("Info");
            ad.setView(LayoutInflater.from(this).inflate(R.layout.info,
                                          null));
            } else {
                  ad.setTitle("About");
            ad.setView(LayoutInflater.from(this).inflate(R.layout.about,
                                          null));
            }

            ad.setPositiveButton("OK", new
            android.content.DialogInterface.OnClickListener() {
                  public void onClick(DialogInterface dialog, int arg1) {
                        // OK, go back to Main menu
                  }
            });

            ad.setOnCancelListener(new DialogInterface.OnCancelListener()
                  {
                  public void onCancel(DialogInterface dialog) {
                        // OK, go back to Main menu
                  }
            });

            ad.show();
      }
```

5.5.2. Ρύθμιση Έντασης Ήχου Πολυμέσων (Volume)

Από το menu_item «Volume» αναδύεται στην κεντρική οθόνη παράθυρο της μορφής «AlertDialog» με μια μπάρα για την ρύθμιση της έντασης του ήχου της εφαρμογής, όπως το τραγούδι στην έναρξη της εφαρμογής και η φωνητική ομιλία. Αρχικά, ο σχεδιασμός του παραθύρου αυτού, όπως ο τίτλος, το background χρώμα και η μπάρα, γίνεται μέσα από τη MapActivity class γιατί δεν μεταφέρεται σε άλλη κλάση. Οπότε:

```
final AlertDialog.Builder alert2 = new AlertDialog.Builder(this);

                  // get the tool to change the media volume of the phone

                  final AudioManager am = (AudioManager)
                        getSystemService(Context.AUDIO_SERVICE);
                  double maxVol =
                        am.getStreamMaxVolume(AudioManager.STREAM_MUSIC);
                  final double curVol =
                        am.getStreamVolume(AudioManager.STREAM_MUSIC);

                  LinearLayout ll = new LinearLayout(this);
                  ll.setOrientation(LinearLayout.VERTICAL);
                  final VolumeBar b = new VolumeBar(this, tts, am);

                  b.measure(150, 30);
                  b.layout(0, 0, 150, 30);
                  b.forceLayout();
                  b.setPadding(15, 10, 15, 10);
```

```
                // Set up the bar to start at the current volume

                b.setProgress((int) (b.getMax() * curVol / maxVol));
                TextView tv = new TextView(this);
                tv.setText("Speech Volume:");
                ll.addView(tv);
                ll.addView(b);
                alert2.setView(ll);

                alert2.setPositiveButton("Ok", new
                                DialogInterface.OnClickListener() {
                      public void onClick(DialogInterface dialog, int
                                        whichButton) {

                      }
                });
                alert2.setNegativeButton("Cancel", new
                                DialogInterface.OnClickListener() {
                      public void onClick(DialogInterface dialog, int
                                        whichButton) {
                            dialog.cancel();

                am.setStreamVolume(AudioManager.STREAM_MUSIC, (int)
                                curVol, 0);
                      }
                });

                Dialog alertDialog = alert2.create();
                alertDialog.setTitle("Options");
                alertDialog.getWindow().setLayout(15, 10);
                alertDialog.show();
}
```

Στην 12η γραμμή του κώδικα καλείται η VolumeBar class. Όταν ο χρήστης μετακινεί την μπάρα, ακούγεται η λέξη «sound». Αυτό γίνεται για να μπορεί ο χρήστης να επιλέξει την επιθυμητή ένταση των ήχων του συστήματος. Δηλαδή, όταν ο χρήστης μετακινεί προς τα αριστερά (δεξιά) τη μπάρα, θα ακούγεται η λέξη «sound» όλο και πιο χαμηλά (δυνατά). Η VolumeBar κλάση είναι υπεύθυνη για αυτό.

```
public class VolumeBar extends SeekBar {

      private TextToSpeech tts;
      AudioManager am;

      public VolumeBar(MapActivity mapActivity, TextToSpeech tts,
                        AudioManager am) {
            super(mapActivity);
            this.tts = tts;
            this.am = am;

      }

      public boolean onTouchEvent(MotionEvent event) {
            if (event.getAction() == MotionEvent.ACTION_UP) {
                  double curVol = getProgress();
                  double maxVol = getMax();
                  am.setStreamVolume(AudioManager.STREAM_MUSIC, (int)
                  (curVol / maxVol *
                  am.getStreamMaxVolume(AudioManager.STREAM_MUSIC)), 0);
```

```
                    tts.speak("", TextToSpeech.QUEUE_FLUSH, null);
                    tts.speak("sound", TextToSpeech.QUEUE_FLUSH, null);
                }
                return super.onTouchEvent(event);
        }
}
```

Επίσης, χρησιμοποιείται η μηχανή TTS (TextToSpeech), το οποίο μετατρέπει το κείμενο σε ομιλία. Έτσι, λοιπόν, η λέξη «sound» ακούγεται

```
tts.speak("sound", TextToSpeech.QUEUE_FLUSH, null);
```

5.5.3. Αλλαγή Ορόφου (Select Floor)

Η εναλλαγή των ορόφων γίνεται από το menu_item με την επιλογή «Select floor». Η μεταφορά στην κλάση γίνεται με την χρησιμοποίηση των intents. Κατά τη μεταφορά αποθηκεύεται στο intent μια πληροφορία που αφορά τον όροφο-χάρτη στον οποίο είναι ο χρήστης.

```
Intent intent1 = new Intent(this, com.ece.Floors.class);
                    intent1.putExtra("Floor1", z2);
                    startActivity(intent1);
```

Έπειτα, ξεκινάει η Floors class, η οποία ουσιαστικά είναι μια λίστα ArrayAdapter<String>, η οποία χρησιμοποιεί τον παρακάτω String πίνακα.

```
String classes[] = { "0 Floor - Ground", "1st Floor", "2nd Floor", "3rd
Floor" }
```

Στη λίστα αυτή, φαίνεται δίπλα από τον όροφο στον οποίο είναι ο χρήστης ένα «τικ», αυτό γίνεται λαμβάνοντας την τιμή από το intent, η οποία ύστερα αποθηκεύεται στην μεταβλητή integer selected.

```
Bundle extras = this.getIntent().getExtras();
selected = extras.getInt("Floor1");
```

Η μεταβλητή selected χρησιμοποιείται για να γίνει check η θέση του ορόφου στον οποίο ο χρήστης είναι.

```
public View getView(int position, View convertView, ViewGroup parent) {
                        CheckedTextView v = (CheckedTextView)
                             super.getView(position, convertView, parent);
                        v.setChecked(position == selected);
                        return v;
                    }
```

Τέλος, ανάλογα με την επιλογή του χρήστη, μεταφέρεται και στον αντίστοιχο όροφο. Αυτό γίνεται πάλι με την χρησιμοποίηση των intents. Για την ακρίβεια, ο όροφος στη λίστα έχει μια θέση και μόλις πατήσει ο χρήστης τον όροφο που θέλει μεταφέρεται στην MapIntent class και κατά τη μεταφορά μεταφέρεται και η πληροφορία της επιλογής του χρήστη.

```
protected void onListItemClick(ListView l, View v, int position, long id)
{
	// TODO Auto-generated method stub
	super.onListItemClick(l, v, position, id);

	if (position == 0) {

	startActivity(MapIntent.openFloor(this.getApplicationContext(), 0));

	} else if (position == 1) {

	startActivity(MapIntent.openFloor(this.getApplicationContext(), 1));

	} else if (position == 2) {

	startActivity(MapIntent.openFloor(this.getApplicationContext(), 2));

	} else {

	startActivity(MapIntent.openFloor(this.getApplicationContext(), 3));
		}

	}
```

Μπαίνοντας στην MapIntent, καλείται η συνάρτηση openFloor και στην οποία αποθηκεύονται ο όροφος «floor» και άλλες δύο μεταβλητές «action» και «view» οι οποίες χρειάζονται για λογικές συνθήκες.

```
public static MapIntent openFloor(Context context, int floor) {
		MapIntent mapIntent = new MapIntent(context);
		mapIntent.putExtra("action", "open floor");
		mapIntent.putExtra("floor", floor);
		mapIntent.putExtra("view", true);
		return mapIntent;
	}
```

Και ύστερα καλείται η MapActivity class, η οποία είναι υπεύθυνη για την εμφάνιση του χάρτη.

```
private static Class<?> clazz;
	static {
		try {
			clazz = Class.forName("com.ece.MapActivity");
		} catch (ClassNotFoundException e) {
			e.printStackTrace();
		}
	}
```

5.5.4. Σημεία Ενδιαφέροντος (POI)

Από το menu_item η επιλογή POI συμβάλει στη μεταφορά της κλάσης MapActivity στην Poi. Η μεταφορά στην κλάση γίνεται με την χρησιμοποίηση των intents.

```
Intent intent2 = new Intent(this, com.ece.Poi.class);
startActivity(intent2);
```

Έπειτα, ξεκινάει η Poi class, η οποία ουσιαστικά είναι μια λίστα ArrayAdapter<String>, η οποία χρησιμοποιεί τον παρακάτω String πίνακα.

```
String classes[] = { "Professors' Offices", "Amfitheatre", "Classrooms",
"Secretariat", "Library", "Labs", "Cafeteria", "WC", "Elevators",
"Stairs", "Exit", "Safety Exit" };
```

Τα παραπάνω αποτελούν σημεία ενδιαφέροντος για τον χρήστη. Κάθε επιλογή στη λίστα έχει και μία θέση (position). Πατώντας ο χρήστης σε μία επιλογή της λίστας, λαμβάνεται η θέση της επιλογής για να μεταφερθεί στην επόμενη αντίστοιχη κλάση της επιλογής του χρήστη. Η μετάβαση στην επόμενη κλάση γίνεται για ακόμα μια φορά με τη χρήση των intents.

```
protected void onListItemClick(ListView l, View v, int position, long id)
{
            super.onListItemClick(l, v, position, id);

            if (position == 0) {

                  startActivity(new Intent("com.search.PROFESSORLIST"));

            } else if (position == 1) {

                  startActivity(new
                  Intent("com.databaselist.AMFITHEATRELIST"));

            } else if (position == 2) {

                  startActivity(new
                  Intent("com.databaselist.CLASSROOMSLIST"));

            } else if (position == 3) {

                  startActivity(new Intent("com.search.SECRETARIATLIST"));

            } else if (position == 4) {

                  startActivity(MapIntent.DrawPoints(this.getApplicationCo
                  ntext(), 400, 737, 0, "Library"));

            } else if (position == 5) {

                  startActivity(new Intent("com.databaselist.LABSLIST"));

            } else if (position == 6) {

                  startActivity(MapIntent.DrawPoints(this.getApplicationCo
                  ntext(), 1495, 735, 0, "Cafeteria"));

            } else if (position == 7) {
```

```
                startActivity(new Intent("com.databaselist.WCLIST"));

            } else if (position == 8) {

                startActivity(new
                Intent("com.databaselist.ELEVATORLIST"));

            } else if (position == 9) {

                startActivity(new
                Intent("com.databaselist.STAIRSLIST"));

            } else if (position == 10) {

                Toast toast = Toast.makeText(this, "All Exits are on the
                Floor 0 - Ground", 5000);
                toast.setGravity(Gravity.BOTTOM, 0, 0);
                toast.show();

                startActivity(MapIntent.DrawPoints(this.getApplicationCo
                ntext(), 563, 745, 0, "Exit"));

            } else {

                startActivity(new
                Intent("com.databaselist.SAFETYEXITLIST"));
            }
        }
```

Κατά την επιλογή των «Professors' Offices» και «Secretariat» μεταφέρεται ο χρήστης στις κλάσεις ProfessorList και SecretariatList, οι οποίες έχουν ένα search box. Στο κουτί αναζήτησης ο χρήστης έχει τη δυνατότητα να αναζητήσει τον καθηγητή ή τη γραμματεία για τα οποία θέλει να μάθει πληροφορίες είτε αυτές είναι η τοποθεσία του είτε είναι κάποιο τηλέφωνο, φαξ και email. Πατώντας πάνω στο search button καλείται η βάση δεδομένων και τίθεται ένα query για το αν ταιριάζει η εκχώρηση του χρήστη με κάποια δεδομένα από τη βάση δεδομένων. Η βάση δεδομένων διαβάζεται με την εντολή:

```
db = (new DatabaseHelper(this)).getWritableDatabase();
```

Έπειτα, η αναζήτηση που γίνεται με το db.rawQuery το οποίο φαίνεται παρακάτω:

```
public void search(View view) {
            // || is the concatenation operation in SQLite
            cursor = db.rawQuery("SELECT _id, firstName, lastName, floor
            FROM info WHERE firstName || ' ' || lastName  || ' ' || floor
            LIKE ?", new String[] { "%" + searchText.getText().toString()
            + "%" });
            startManagingCursor(cursor);
            adapter = new SimpleCursorAdapter(this, R.layout.prof_list,
            cursor, new String[] { "firstName", "lastName", "floor" }, new
            int[] { R.id.firstName, R.id.lastName, R.id.floorN });
            setListAdapter(adapter);
        }
```

Αφού εμφανιστεί αυτό που αναζητούσε ο χρήστης και το επιλέξει τότε μεταφέρεται στην επόμενη κλάση η οποία περιλαμβάνει τις χρήσιμες πληροφορίες σχετικά με την επιλογή του

χρήστη. Η κλάση όσον αφορά τους καθηγητές είναι η ProfessorDetails και όσον αφορά τις γραμματείες είναι η SecretariatDetails. Και στις δύο περιπτώσεις περιλαμβάνονται τηλέφωνα, email και η δυνατότητα να δει ο χρήστης την τοποθεσία της επιλογής του πάνω στο χάρτη. Μάλιστα, ένα πολύ σημαντικό χαρακτηριστικό αυτών των δεδομένων είναι ότι ο χρήστης μπορεί αν τα χρησιμοποιήσει άμεσα. Δηλαδή, αν πατήσει πάνω στον αριθμό τηλέφωνου τότε το κινητό θα τον καλέσει, αντίστοιχα και για τα email που μπορεί αν γράψει και να το στείλει. Όλα αυτά γίνονται μέσω του παρακάτω κώδικα, ο οποίος είναι μέρος της ProfessorDetails κλάσης και είναι παρόμοιος με αυτόν της SecretariatDetails :

Αρχικά, διαβάζεται η βάση db και μετά με την επιλογή που έκανε ο χρήστης προηγουμένως μεταφέρεται μέσω intent σε αυτή την κλάση και θέτεται ερώτημα db.rawQuery για να βρεθούν οι αντίστοιχες πληροφορίες της επιλογής το χρήστη. Παρακάτω φαίνονται πως εμφανίζεται το officephone και το email. Αφού γίνουν όλα αυτά, τα δεδομένα παρουσιάζονται σε μια λίστα ListAdapter.

```
db = (new DatabaseHelper(this)).getWritableDatabase();
            Cursor cursor = db.rawQuery("SELECT _id, firstName, lastName,
               floor, z, x, y, phone, email FROM info  WHERE _id = ?", new
               String[] { "" + profId });
            startManagingCursor(cursor);
            if (cursor.getCount() == 1) {
                  cursor.moveToFirst();
                  profNameText = (TextView) findViewById(R.id.profName);
 profNameText.setText(cursor.getString(cursor.getColumnIndex("firstName"))
             + " " + cursor.getString(cursor.getColumnIndex("lastName")));

                  titleText = (TextView) findViewById(R.id.floorNum);

                  titleText.setText(cursor.getString(cursor.getColumnIndex
                                                ("floor")));

                  actions = new ArrayList<PassValues>();
                  positions = new ArrayList<PointsFloor>();

                  String officePhone =
                      cursor.getString(cursor.getColumnIndex("phone"));
                  if (officePhone != null) {
                        actions.add(new PassValues("Call office",
                        officePhone, PassValues.ACTION_CALL));
                  }

                  String email =
cursor.getString(cursor.getColumnIndex("email"));
                  if (email != null) {
                        actions.add(new PassValues("Email", email,
PassValues.ACTION_EMAIL));
                  }

                  positionX = cursor.getInt(cursor.getColumnIndex("x"));
                  positionY = cursor.getInt(cursor.getColumnIndex("y"));
                  positionZ = cursor.getInt(cursor.getColumnIndex("z"));
                  actions.add(new PassValues("Find Position on the Map",
null, PassValues.ACTION_MAP));

                  name =
cursor.getString(cursor.getColumnIndex("firstName")) + " " +
cursor.getString(cursor.getColumnIndex("lastName"));
            }
```

```
            adapter = new PassValuesAdapter();
            setListAdapter(adapter);
      }
```

Και για τις τρεις περιπτώσεις (officephone, email, position on the map) καλείται η κλάση PassValues, η οποία περιλαμβάνει getters και setters για τη μεταφορά των δεδομένων.

```
public static final int ACTION_CALL = 1;
public static final int ACTION_EMAIL = 2;
public static final int ACTION_MAP = 3;

      public PassValues(String label, String data, int type) {
            super();
            this.label = label;
            this.data = data;
            this.type = type;
      }

      public String getLabel() {
            return label;
      }

      public void setLabel(String label) {
            this.label = label;
      }

      public String getData() {
            return data;
      }

      public void setData(String data) {
            this.data = data;
      }

      public int getType() {
            return type;
      }

      public void setType(int type) {
            this.type = type;
      }

}
```

Για το officephone αντιστοιχεί η ενέργεια ACTION_CALL, για το email αντιστοιχεί η ενέργεια ACTION_EMAIL και για την εύρεση της θέσης στο χάρτη η ACTION_MAP. Για την κάθε ενέργεια έχει οριστεί ένα αριθμός.

```
ACTION_CALL = 1
ACTION_EMAIL = 2
ACTION_MAP = 3
```

Ανάλογα με την επιλογή που θα κάνει ο χρήστης, θα πραγματοποιηθεί και η αντίστοιχη ενέργεια. Παρακάτω φαίνονται οι αντίστοιχες ενέργειες:

```
➢ case PassValues.ACTION_CALL:
            Uri callUri = Uri.parse("tel:" + action.getData());
            intent = new Intent(Intent.ACTION_CALL, callUri);
            startActivity(intent);
            break;
```

Στην περίπτωση αυτή, ο χρήστης πάτησε να κληθεί ο αριθμός. Γίνεται μεταφορά στο μενού κλήσης του κινητού χρησιμοποιώντας intent και μέσα στο intent μεταφέρεται ο τηλεφωνικός αριθμός.

```
➢ case PassValues.ACTION_EMAIL:
                intent = new Intent(android.content.Intent.ACTION_SEND);
                intent.setType("plain/text");
                intent.putExtra(android.content.Intent.EXTRA_EMAIL, new
String[] { action.getData() });
                startActivity(intent);
                break;
```

Αντίστοιχα, ο χρήστης μπορεί να στείλει email στο άτομο που επέλεξε.

```
➢ case PassValues.ACTION_MAP:
            startActivity(MapIntent.DrawPoints(this.getApplicationContext(
            ), positionX, positionY, positionZ, name));
                  break;
```

Τέλος, για να εμφανιστεί η θέση της επιλογής του χρήστη στο χάρτη, λαμβάνονται οι συντεταγμένες της επιλογής και καλείται η συνάρτηση DrawPoints της κλάσης Map Intent. Αφού περαστούν στα intent οι απαραίτητες πληροφορίες για την εμφάνιση στο χάρτη, καλείται η κλάση MapView για την εμφάνιση του κατάλληλου χάρτη (ανάλογα τον όροφο της επιλογής) μαζί με το σημείο της τοποθεσίας της επιλογής.

Με τον παρακάτω κώδικα, εμφανίζονται στη λίστα τα τηλέφωνα, email και position και πραγματοποιείται η εκχώρηση των ενεργειών για την κάθε επιλογή.

```
public View getView(int position, View convertView, ViewGroup parent) {
                PassValues action = actions.get(position);
                LayoutInflater inflater = getLayoutInflater();
                View view = inflater.inflate(R.layout.passvalues_list,
                                             parent, false);
                TextView label = (TextView)
                                view.findViewById(R.id.label);
                label.setText(action.getLabel());
                TextView data = (TextView) view.findViewById(R.id.data);
                data.setText(action.getData());
                return view;
          }
```

5.5.5. Διαδικασία της πλοήγησης

Η πλοήγηση ξεκινάει πατώντας ο χρήστης στο menu_item την επιλογή «Find Destination» και μεταφέρεται μέσω intents στην επόμενη κλάση στην οποία ο χρήστης ορίζει την τοποθεσία του. Αρχικά καλείται η βάση δεδομένων.

```
db = (new DatabaseHelper(this)).getWritableDatabase();
```

Έπειτα ο χρήστης έχει τη δυνατότητα στο search box να καταχωρήσει τη θέση του και να πατήσει το κουμπί αναζήτησης. Η αναζήτηση γίνεται με την εντολή db.rawQuery στην οποία τίθεται ένα ερώτημα, αν η καταχώρηση του χρήστη ταιριάζει με κάποιο από τα δεδομένα της στήλης «node_name» του πίνακα «nodes». Στη συνέχεια, τα δεδομένα αυτά καταχωρούνται σε ένα cursor και αυτά εμφανίζονται σε μια ListAdapter.

```
public void search(View view) {
            // || is the concatenation operation in SQLite
            cursor1 = db.rawQuery("SELECT _id, name, node_name, numFloor,
pox, poy, poz FROM nodes WHERE node_name LIKE ?", new String[] { "%"
                        + searchText1.getText().toString() + "%" });
            startManagingCursor(cursor1);
            adapter1 = new SimpleCursorAdapter(this, R.layout.nodeslist,
cursor1, new String[] { "node_name", "numFloor" }, new int[] { R.id.name,
R.id.floorZ });
            setListAdapter(adapter1);
```

Για να μεταβεί ο χρήστης στην επόμενη κλάση, πρέπει να πατήσει σε μία από τις επιλογές που θα του έχει η λίστα. Μόλις πατήσει μεταφέρεται στην επόμενη κλάση στην οποία θα δηλώσει τον προορισμό του. Κατά τη μετάβαση, μεταφέρονται και κάποια δεδομένα μέσω intent.putextra όπως οι συντεταγμένες της θέσης του, η ονομασία της θέσης του αλλά και η ονομασία του κόμβου.

```
public void onListItemClick(ListView parent, View view, int position, long
id) {
            Intent intent = new Intent(this, DestinationList.class);
            Cursor cursor1 = (Cursor) adapter1.getItem(position);
            intent.putExtra("SOURCE_ID",
cursor1.getInt(cursor1.getColumnIndex("_id")));
            intent.putExtra("source",
cursor1.getString(cursor1.getColumnIndex("name")));
            intent.putExtra("name_source",
cursor1.getString(cursor1.getColumnIndex("node_name")));
            intent.putExtra("x1",
cursor1.getInt(cursor1.getColumnIndex("pox")));
            intent.putExtra("y1",
cursor1.getInt(cursor1.getColumnIndex("poy")));
            intent.putExtra("z1",
cursor1.getInt(cursor1.getColumnIndex("poz")));
            name_source =
cursor1.getString(cursor1.getColumnIndex("node_name"));
            startActivity(intent);

      }
```

Στην κλάση στην οποία θα δηλώσει ο χρήστης τον προορισμό του, έχει πάλι ένα search box και με την ίδια διαδικασία επιλέγει που θέλει να πάει.

```
public void search(View view) {
            // || is the concatenation operation in SQLite
            cursor2 = nodes_db.rawQuery("SELECT _id, name, node_name,
numFloor, pox, poy, poz FROM nodes WHERE node_name LIKE ?", new String[] {
"%"
                        + searchText2.getText().toString() + "%" });
            startManagingCursor(cursor2);
            adapter2 = new SimpleCursorAdapter(this, R.layout.nodeslist,
cursor2, new String[] { "node_name", "numFloor" }, new int[] { R.id.name,
R.id.floorZ });
            setListAdapter(adapter2);
      }
```

Μόλις επιλέξει ο χρήστης, καλείται ο αλγόριθμος του Dijkstra. Στην κλάση αυτή μεταφέρθηκε, όπως αναφέρθηκε, μέσω intents ο κόμβος της τοποθεσίας του χρήστη και τώρα είναι γνωστός ο προορισμός. Συνεπώς μπορεί να βρεθεί η ελάχιστη διαδρομή μεταξύ source και destination.

Στο πρώτο κομμάτι του κώδικα, τίθεται ένα ερώτημα (nodes_db.rawQuery) στη βάση δεδομένων για τον πίνακα «nodes», να επιλέξει όλα όσα είναι στον πίνακα αυτό ("SELECT * FROM nodes") . Στη συνέχεια, πηγαίνει ο cursor στην πρώτη θέση και ξεκινάει η καταχώρηση των συντεταγμένων όλων των κορυφών, μέχρι να φτάσει στον τελευταίο κόμβο.

```
      // Dijkstra
            DijkstraAlgorithm algo = new DijkstraAlgorithm();

            Cursor nodes = nodes_db.rawQuery("SELECT * FROM nodes", null);
            startManagingCursor(nodes);
            nodes.moveToFirst();
            while (!nodes.isAfterLast()) {
                  Vertex vertex = new
Vertex(nodes.getString(nodes.getColumnIndex("name")));
                  vertex.x = nodes.getInt(nodes.getColumnIndex("pox"));
                  vertex.y = nodes.getInt(nodes.getColumnIndex("poy"));
                  vertex.z = nodes.getInt(nodes.getColumnIndex("poz"));

      algo.vertices.put(nodes.getString(nodes.getColumnIndex("name")),
vertex);
                  nodes.moveToNext();
            }
```

Μόλις τελειώσει, θέτει ένα καινούργιο ερώτημα στον πίνακα «path» για να βρει τους κόμβους που ενώνονται μεταξύ τους και ποιο θα είναι το κόστος σε κάθε περίπτωση. Βρίσκει, λοιπόν, την ελάχιστη διαδρομή και καταχωρεί τις συντεταγμένες των κόμβων της ελάχιστης διαδρομής σε πίνακες καθώς θα χρειαστούν για την πλοήγηση.

```
            Cursor edges = edges_db.rawQuery("SELECT * FROM path", null);
            startManagingCursor(edges);
            edges.moveToFirst();
            while (!edges.isAfterLast()) {
```

```
                String fromStr =
edges.getString(edges.getColumnIndex("fromNode"));
                String toStr =
edges.getString(edges.getColumnIndex("toNode"));
                int cost = edges.getInt(edges.getColumnIndex("cost"));
                Vertex from = algo.vertices.get(fromStr);
                Vertex to = algo.vertices.get(toStr);
                from.adjacencies.add(new Edge(to, cost));
                edges.moveToNext();
          }

          algo.computePaths(algo.vertices.get(source));
          List<Vertex> sortestPath =
algo.getShortestPathTo(algo.vertices.get(destination));
          int[] xs = new int[sortestPath.size()];
          int[] ys = new int[sortestPath.size()];
          int[] zs = new int[sortestPath.size()];
          int i = 0;
          for (Vertex v : sortestPath) {
                xs[i] = v.x;
                ys[i] = v.y;
                zs[i] = v.z;
                i++;
          }
```

Αφού έχει βρεθεί και η ελάχιστη διαδρομή, καλείται μέσω intents η MapIntent.

```
startActivity(MapIntent.SrcDest(this.getApplicationContext(), source, x1,
y1, z1, destination, x2, y2, z2, xs, ys, zs, name_source,
                              name_destination));
```

Χρησιμοποιείται η SrcDest στην οποία περνιούνται οι ονομασίες των κόμβων της θέσης του χρήστη και του προορισμού καθώς επίσης και οι πίνακες με τις συντεταγμένες όλων των κόμβων της ελάχιστης διαδρομής.

```
public static MapIntent SrcDest(Context context, String source, int x1,
int y1, int z1, String destination, int x2, int y2, int z2, int[] xs,
int[] ys,
                int[] zs, String name_source, String name_destination) {
          MapIntent mapIntent = new MapIntent(context);
          mapIntent.putExtra("points", "show position");
          mapIntent.putExtra("source", source);
          mapIntent.putExtra("x1", x1);
          mapIntent.putExtra("y1", y1);
          mapIntent.putExtra("z1", z1);
          mapIntent.putExtra("destination", destination);
          mapIntent.putExtra("x2", x2);
          mapIntent.putExtra("y2", y2);
          mapIntent.putExtra("z2", z2);
          mapIntent.putExtra("xs", xs);
          mapIntent.putExtra("ys", ys);
          mapIntent.putExtra("zs", zs);
          mapIntent.putExtra("name_source", name_source);
          mapIntent.putExtra("name_destination", name_destination);
          mapIntent.putExtra("view", false);
          return mapIntent;
     }
```

Κατόπιν, η MapIntent καλεί την MapView η οποία είναι υπεύθυνη για το τι θα εμφανιστεί στην οθόνη.

```
private static Class<?> clazz;
	static {
		try {
			clazz = Class.forName("com.ece.MapActivity");
		} catch (ClassNotFoundException e) {
			e.printStackTrace();
		}
	}
```

Η MapView, επειδή πρόκειται για την περίπτωση της πλοήγησης θα καλέσει την Navigation στην οποία ζωγραφίζονται οι διαδρομές και «τρέχει» η πλοήγηση.

```
private float getLengthUpToPoint(int index) {
		float l = 0f;
		for (int i = 1; i <= index; i++) {
			x1 = xs[i - 1];
			y1 = ys[i - 1];
			x2 = xs[i];
			y2 = ys[i];
			l += Math.sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 -
y2));
		}
		return l;
	}
```

Η συνάρτηση getLengthUpToPoint υπολογίζει για κάθε χρονική στιγμή πόση απόσταση από την αρχική θέση έχει καλύψει ο χρήστης.

```
	private int getIndexCoveredByLength(float length) {
		for (int i = 1; i < xs.length; i++) {
			x1 = xs[i - 1];
			y1 = ys[i - 1];
			x2 = xs[i];
			y2 = ys[i];
			length -= Math.sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) *
(y1 - y2));
			if (length <= 0)
				return i - 1;
		}
		return xs.length - 1;
	}
```

Η συνάρτηση getIndexCoveredByLength υπολογίζει μέχρι ποιο κόμβο έχει φτάσει ο χρήστης για κάθε χρονική στιγμή.

Η κάθε χρονική στιγμή έχει οριστεί στα 100ms.

Υπάρχει μια συνάρτηση makeContent στην οποία πραγματοποιείται η αποτύπωση της διαδρομής στο χάρτη. Ο κώδικας που ακολουθεί αποτελεί ένα μέρος της makeContent για να παρουσιαστεί πως λειτουργεί. Διατηρείται σταθερό το αρχικό σημείο, και καλώντας τις

παραπάνω δύο συναρτήσεις λαμβάνεται συνεχώς η θέση του χρήστη και ζωγραφίζεται μέχρι εκείνο το σημείο.

```
mPath.moveTo(xs[0], ys[0]);
                  for (int i = 1; i <= indexCovered; i++) {
                        mPath.lineTo(xs[i], ys[i]);
                  }
                  if (indexCovered < xs.length - 0.1) {
                        float l = getLengthUpToPoint(indexCovered);
                        float dl = coveredLength - l;
                        float x1 = xs[indexCovered];
                        float y1 = ys[indexCovered];
                        float x2 = xs[indexCovered + 1];
                        float y2 = ys[indexCovered + 1];
                        float d = (float) Math.sqrt((x1 - x2) * (x1 - x2)
+ (y1 - y2) * (y1 - y2));
                        float t = dl / d;
                        mPath.lineTo(x1 + t * (x2 - x1), y1 + t * (y2 -
y1));
                        c.drawCircle(x1 + t * (x2 - x1), y1 + t * (y2 -
y1), 3, points);
                        followX = x1 + t * (x2 - x1);
                        followY = y1 + t * (y2 - y1);
                  }
```

Με την αλλαγή του ορόφου, ορίζεται ως νέο αρχικό σημείο η θέση του κόμβου που είναι το ασανσέρ ή οι σκάλες. Η πλοήγηση τρέχει μέσα από ένα thread run, το οποίο σταματάει μόλις ο χρήστης καλύψει την απόσταση.

```
private void makepicture() {

            mPicture = makeContent();
            timer = new Thread() {
                  public void run() {
                        try {
                              sleep(8000);
                              while (true) {
                                    sleep(interval);
                                    if (changeFloorBefore) {
                                          isPaused = !isPaused;
                                          activity.change(isPaused);
                                          if ((updownY[u] == 728) &&
(xs[indexCovered - 1] == 1300)) {
                                                activity.message("Go
downstairs");
                                                changeFloorBefore = false;
                                                sleep(2000);
                                          } else if ((updownY[u] == 728)
&& (xs[indexCovered - 1] == 1495)) {
                                                activity.message("Go
upstairs");
                                                changeFloorBefore = false;
                                                sleep(2000);
                                          } else {
                                                activity.message("Use " +
updown[u] + " to change floor. Go to Floor " + zs[indexCovered + 1]);
                                                sleep(5000);
                                          }
```

```
                                        activity.message("Tap on the
screen to continue your navigation");
                                        u = 2;
                                  }

                                  speed = activity.changeSpeed();
                                  if (!isPaused)
                                        coveredLength += speed *
(interval / 1000f);
                                  if (coveredLength > totalLength) {
                                        i = 0;

      activity.message("Congratulations, you reached your destination");
                                        return;
                                  }
                                  mPicture = makeContent();
                                  postInvalidate();
                            }
                      } catch (InterruptedException e) {
                            e.printStackTrace();
                      }
                }
          };
          timer.start();
          timer = null;
      }
```

5.6. Δημιουργία Αλγόριθμου δρομολόγησης

Για αλγόριθμο δρομολόγησης επιλέχτηκε ο αλγόριθμος του Dijkstra. Αρχικά, υπάρχει μια βάση δεδομένων «Allinfo», η οποία έχει ποιοι κόμβοι ενώνονται μεταξύ τους και με ποιο κόστος (Cost).

Οι κορυφές (Vertices) εκπροσωπούνται από την Vertex class.

```
public class Vertex implements Comparable<Vertex> {
      public final String name;
      // public Edge[] adjacencies;
      public List<Edge> adjacencies = new ArrayList<Edge>();
      public double minDistance = Double.POSITIVE_INFINITY;
      public Vertex previous;

      public Vertex(String argName) {
            name = argName;
      }

      public String toString() {
            return name;
      }

      public int compareTo(Vertex other) {
            return Double.compare(minDistance, other.minDistance);
      }

}
```

Είναι απαραίτητο να διατηρηθεί μια λίστα άκρων για κάθε κορυφή. Πριν τη χρήση του αλγορίθμου θα πρέπει να αναφερθούν δύο άλλα πεδία:

- minDistance: Η μικρότερη απόσταση από την πηγή προς την κορυφή στο γράφημα. Αρχικοποιείται στο θετικό άπειρο (όσο το δυνατόν μεγαλύτερο).

- previous: Η αναφορά στην προηγούμενη κορυφή για να υπολογιστεί η συντομότερη διαδρομή από την πηγή κορυφή σε αυτή την κορυφή.

Έπειτα, πρέπει για τον αλγόριθμο να μπουν σε σειρά οι κορυφές για αυτό υπάρχει η συνάρτηση compareTo.

Υπάρχει, επίσης, μια κλάση η οποία αντιπροσωπεύει τις άκρες και μεταφέρει την αξία του κόστους του και τον στόχο κορυφή:

```
public class Edge {
	private Vertex target;;
	private double weight;

	public Edge(Vertex target, double weight) {
		this.setTarget(target);
		this.setWeight(weight);
	}

	public void setTarget(Vertex target) {
		this.target = target;
	}

	public Vertex getTarget() {
		return target;
	}

	public void setWeight(double weight) {
		this.weight = weight;
	}

	public double getWeight() {
		return weight;
	}
}
```

Στην παραπάνω κλάση, για την μεταφορά των δεδομένων χρησιμοποιήθηκαν getters και setters.

Στη συνέχεια, ακολουθεί το κύριο μέρος του αλγορίθμου δρομολόγησης. Θα χωριστεί σε δύο στάδια ο τρόπος υπολογισμού:

- Υπολογισμός της ελάχιστη απόστασης από την πηγή σε κάθε κορυφή του γραφήματος. Ταυτόχρονα, καταγράφει τις previous αναφορές για κάθε κορυφή *v*, η οποία δίνει την προηγούμενη κορυφή της συντομότερης διαδρομής από την πηγή κορυφή στο *v* . Αυτό είναι το ακριβό βήμα.

- Αργότερα, κάθε φορά που επιθυμείται να βρεθεί μια συγκεκριμένη συντομότερη διαδρομή μεταξύ της πηγής κορυφής και του δοθέντος κόμβου, ακολουθούνται οι προηγούμενες αναφορές για την γρήγορή του κατασκευή.

Για το πρώτο μέρος, υπάρχει η συνάρτηση computePaths , η οποία παίρνει ως είσοδο την πηγή κορυφή από την οποία βρίσκονται όλα τα συντομότερα μονοπάτια.

```
      public void computePaths(Vertex source) {
            source.minDistance = 0.;
            PriorityQueue<Vertex> vertexQueue = new
PriorityQueue<Vertex>();
            vertexQueue.add(source);

            while (!vertexQueue.isEmpty()) {
                  Vertex u = vertexQueue.poll();

                  // Visit each edge exiting u
                  for (Edge e : u.adjacencies) {
                        Vertex v = e.getTarget();
                        double weight = e.getWeight();
                        double distanceThroughU = u.minDistance + weight;
                        if (distanceThroughU < v.minDistance) {
                              vertexQueue.remove(v);
                              v.minDistance = distanceThroughU;
                              v.previous = u;
                              vertexQueue.add(v);
                        }
                  }
            }
```

Το περίγραμμα του πώς λειτουργεί η συνάρτηση εμφανίζεται παρακάτω:

- Πραγματοποιείται επίσκεψη για κάθε κορυφή, όταν ξεπεράσει τις άκρες και προσαρμογή της minDistance ανάλογα με τις ανάγκες.

Εάν (u, v) είναι μια ακμή και u είναι το συντομότερο μονοπάτι προς v, τότε:

$d(u) + w(u, v) = d(v)$.

Με άλλα λόγια, η εύρεση του *ν* γίνεται πηγαίνοντας από την πηγή στο u, ακολουθώντας την ακμή (u , v). Τελικά, θα επισκεφθεί κάθε προκάτοχο του *ν*, ο οποίος είναι προσβάσιμος από την πηγή. Η συντομότερη διαδρομή περνάει μέσα από ένα από αυτά. Παρακολουθείται η μικρότερη απόσταση η οποία έχει υπάρξει μέχρι στιγμής από τη minDistance και την κορυφή που πέρασε από τη previous.

```
       double distanceThroughU = u.minDistance + weight;
                        if (distanceThroughU < v.minDistance) {
                              vertexQueue.remove(v);
                              v.minDistance = distanceThroughU;
                              v.previous = u;
                              vertexQueue.add(v);
                        }
```

Τέλος, χρειάζεται ένας τρόπος για να επισκεφθούν τις κορυφές με την ελάχιστη απόσταση τους. Χρησιμοποιείται η PriorityQueue class της Java με την minDistance ως προτεραιότητα. Η ουρά προτεραιότητας δεν ταιριάζει όταν η σειρά των στοιχείων είναι αλλαγμένη, οπότε όταν αλλάζει η ελάχιστη απόσταση του κάθε κόμβου, θα πρέπει να

αφαιρεθεί και να ξανατοποθετηθεί στο σύνολο. Η ουρά θα αποτελείται μόνο από τις κορυφές που έχουν πεπερασμένη απόσταση.

```
while (!vertexQueue.isEmpty()) {
                  Vertex u = vertexQueue.poll();
```

Υπάρχει πρόσβαση και μπορούν να απομακρυνθούν τα μικρότερα στοιχεία χρησιμοποιώντας την poll(). Αν πραγματοποιηθεί αλλαγή στην ελάχιστη απόσταση της κορυφής, πρέπει να γίνει ενημέρωση.

```
vertexQueue.remove(v);
vertexQueue.add(v);
```

Παρακάτω με τη συνάρτηση getShortestPathTo, αλλάζει η σειρά των vertices και από την τελευταία μεταφέρεται στην πηγή κορυφή.

```
public ArrayList<Vertex> getShortestPathTo(Vertex target) {
            ArrayList<Vertex> path = new ArrayList<Vertex>();
            for (Vertex vertex = target; vertex != null; vertex =
vertex.previous)
                  path.add(vertex);

            Collections.reverse(path);
            return path;
      }
```

5.7. Φωνητικές οδηγίες

Κατά τη διάρκεια της πλοήγησης αλλά και όταν χρησιμοποιείται η εφαρμογή ακούγονται φωνητικές οδηγίες και ειδοποιήσεις. Το σύστημα αναπαραγωγής ομιλίας ονομάστηκε «Ntuadroid». Η αναπαραγωγή της ομιλίας πραγματοποιείται με τη χρησιμοποίηση του Text-To-Speech engine. Αυτό που κάνει είναι να μετατρέπει ένα κείμενο σε ομιλία. Για να γίνει αυτό, αρχικά, πρέπει να κατεβάσει ο χρήστης από το Android Market το speechsynthesis και να το κάνει εγκατάσταση.

Όσον αφορά τον κώδικα, η class που θα χρησιμοποιεί το text-to-speech (tts) πρέπει να έχει την διεπαφή onInitListener «implements OnInitListener». Συνεπώς, όσες κλάσεις χρησιμοποίησαν το tts είχαν και την διεπαφή αυτή. Επίσης, χρησιμοποιώντας την εντολή

```
tts = new TextToSpeech(this, this);
```

καλείται η παρακάτω συνάρτηση, η οποία ορίζει ως γλώσσα ομιλίας τα αμερικάνικα αγγλικά και στο message το κείμενο το οποίο θα μετατραπεί σε ομιλία.

```
public void onInit(int status) {
            tts.setLanguage(Locale.US);
            tts.speak(message, TextToSpeech.QUEUE_FLUSH, null);
            }
      }
```

Έπειτα, αν χρειαστεί κατά τη διάρκεια της activity, χρησιμοποιείται το

```
tts.speak(message, TextToSpeech.QUEUE_FLUSH, null);
```

Ένα παράδειγμα από μια κλάση της εφαρμογής:

```
public void onInit(int status) {
            tts.setLanguage(Locale.US);
            SharedPreferences                    getPrefs                    =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
            boolean voice = getPrefs.getBoolean("checkboxVoice", true);
            if (voice == true) {
                  if (isFirst) {
                        tts.speak("Where are you? Choose your position",
TextToSpeech.QUEUE_FLUSH, null);
                        isFirst = false;
                  } else
                        tts.speak("You are in the " + name_source + ".
Where do you want to go?", TextToSpeech.QUEUE_FLUSH, null);
            }
      }
```

Στο οποίο παρατηρείται η ύπαρξη του SharedPreferences, καθώς δίνεται η δυνατότητα στο χρήστη να επιλέξει αν θέλει να ακούγονται οι φωνητικές οδηγίες ή όχι. Αν η μεταβλητή boolean voice είναι αληθής τότε θα ακούγονται.

5.8. Βάση δεδομένων

Για τη λειτουργία της εφαρμογής, ήταν απαραίτητη η κατασκευή μιας βάσης δεδομένων. Η βάση αυτή περιλαμβάνει τέσσερα tables. Η βάση δεδομένων ονομάζεται «Allinfo»

```
public static final String DATABASE_NAME = "Allinfo";
```

και τα tables «nodes» «path» «info» «SecretariatInfo».

Η βάση δεδομένων κατασκευάστηκε σε ένα .xml αρχείο το οποίο το διαβάζει η εφαρμογή και δημιουργεί τα tables και τα δεδομένα σαν να είναι κανονική SQLite Database. Παρακάτω παρατίθενται μερικάδεδομένα από κάθε table της βάσης δεδομένων.

- Το table «nodes» περιλαμβάνει τους κόμβους της πλοήγησης. Χρησιμοποιώντας αυτούς του κόμβους ο αλγόριθμος δρομολόγησης, μπορεί και βρίσκει την ελάχιστη διαδρομή προς τον προορισμό. Το «nodes» έχει τον αριθμό του κόμβου, την ονομασία και τον όροφο στον οποίο είναι ο κόμβος καθώς και τις συντεταγμένες x, y και z που ταιριάζουν στο χάρτη.

```
<sql>
<statement>
CREATE TABLE IF NOT EXISTS nodes (
      _id INTEGER PRIMARY KEY AUTOINCREMENT,
      name VARCHAR(50),
      node_name VARCHAR(50),
      numFloor VARCHAR(50),
      pox INTEGER,
      poy INTEGER,
```

```
      poz INTEGER)
</statement>
<statement>INSERT INTO nodes (name, node_name, numFloor, pox, poy, poz)
VALUES('v0', 'Main Entrance', 'Floor 0 - Ground', 942, 875, 0)</statement>
<statement>INSERT INTO nodes (name, node_name, numFloor, pox, poy, poz)
VALUES('v1', 'Amf1', 'Floor 0 - Ground', 1060, 775, 0)</statement>
<statement>INSERT INTO nodes (name, node_name, numFloor, pox, poy, poz)
VALUES('v2', null, 'Floor 0 - Ground', 942, 775, 0)</statement>
<statement>INSERT INTO nodes (name, node_name, numFloor, pox, poy, poz)
VALUES('v3', null, 'Floor 0 - Ground', 942, 700, 0)</statement>
<statement>INSERT INTO nodes (name, node_name, numFloor, pox, poy, poz)
VALUES('v4', null, 'Floor 0 - Ground', 1060, 700, 0)</statement>
<statement>INSERT INTO nodes (name, node_name, numFloor, pox, poy, poz)
VALUES('v5', null, 'Floor 0 - Ground', 1011, 700, 0)</statement>
<statement>INSERT INTO nodes (name, node_name, numFloor, pox, poy, poz)
VALUES('v6', null, 'Floor 0 - Ground', 1300, 700, 0)</statement>
<statement>INSERT INTO nodes (name, node_name, numFloor, pox, poy, poz)
VALUES('v7', 'Exit 1', 'Floor 0 - Ground', 1300, 728, 0)</statement>
```

- Το table «path» στις δύο πρώτες στήλες είναι οι κόμβοι που επικοινωνούν μεταξύ τους και στην τρίτη στήλη είναι το κόστος που θα έχει για να πάει ο χρήστης από τον ένα κόμβο ($1^η$ στήλη) στον άλλο κόμβο ($2^η$ στήλη).

```
<statement>
CREATE TABLE IF NOT EXISTS path (
      _id INTEGER PRIMARY KEY AUTOINCREMENT,
      fromNode VARCHAR(50),
      toNode VARCHAR(50),
      cost INTEGER)
</statement>
<statement>INSERT INTO path (fromNode, toNode, cost) VALUES('v0', 'v2',
100)</statement>
<statement>INSERT INTO path (fromNode, toNode, cost) VALUES('v1', 'v2',
118)</statement>
<statement>INSERT INTO path (fromNode, toNode, cost) VALUES('v1', 'v4',
75)</statement>
<statement>INSERT INTO path (fromNode, toNode, cost) VALUES('v2', 'v0',
100)</statement>
<statement>INSERT INTO path (fromNode, toNode, cost) VALUES('v2', 'v1',
118)</statement>
<statement>INSERT INTO path (fromNode, toNode, cost) VALUES('v2', 'v3',
74)</statement>
<statement>INSERT INTO path (fromNode, toNode, cost) VALUES('v3', 'v2',
74)</statement>
<statement>INSERT INTO path (fromNode, toNode, cost) VALUES('v3', 'v5',
69)</statement>
```

- Το table «info» περιέχει πληροφορίες σχετικά με τα γραφεία των καθηγητών, τηλέφωνα, email και τις συντεταγμένες της θέσης του γραφείου του καθηγητή στο χάρτη.

```
<statement>
CREATE TABLE IF NOT EXISTS info (
      _id INTEGER PRIMARY KEY AUTOINCREMENT,
      firstName VARCHAR(50),
      lastName VARCHAR(50),
      floor VARCHAR(30),
      z INTEGER,
```

```
      x INTEGER,
      y INTEGER,
      phone VARCHAR(30),
      email VARCHAR(30))
</statement>
<statement>INSERT INTO info  VALUES(1,'Michael','Theologou','Floor 2 -
B.2.9', 2, 450, 363, '+302107722532', 'theolog@cs.ntua.gr')</statement>
<statement>INSERT INTO info  VALUES(2,'Efstathios','Sykas','Floor 2 -
B.2.11', 2, 450, 456, '+302107722528', 'sykas@cn.ntua.gr')</statement>
<statement>INSERT INTO info  VALUES(3,'Foto','Afrati','Floor 2 - B.2.5',
2, 450, 175, '+302107722498', 'afrati@softlab.ece.ntua.gr')</statement>
```

- Το table «SecretariatInfo» περιέχει πληροφορίες σχετικά με τις διάφορες γραμματείες, τηλέφωνα, φαξ, email και τις συντεταγμένες της θέσης τους στο χάρτη. Σε κάποιες περιπτώσεις στις οποίες δεν υπάρχουν δεδομένα τοποθετήθηκε μια παύλα "–".

```
<statement>
CREATE TABLE IF NOT EXISTS SecretariatInfo (
      _id INTEGER PRIMARY KEY AUTOINCREMENT,
      name VARCHAR(50),
      floor VARCHAR(30),
      z INTEGER,
      x INTEGER,
      y INTEGER,
      phone1 VARCHAR(30),
      phone2 VARCHAR(30),
      fax VARCHAR(30),
      email VARCHAR(30))
</statement>
<statement>INSERT INTO SecretariatInfo  VALUES(1,'Undergraduate
Secretariat G.1.1','Floor 1', 1, 1495, 735, '+302107723990',
'+302107721889', ' +302107723991', 'secreta@cc.ece.ntua.gr')</statement>
<statement>INSERT INTO SecretariatInfo  VALUES(2,'Postgraduate Secretariat
G.1.2','Floor 1', 1, 1495, 670, '+302107724307', '+302107722224',
'+302107722397', 'graduate@ece.ntua.gr')</statement>
<statement>INSERT INTO SecretariatInfo  VALUES(3,'Secretariat of
Electronic Lab B.1.12', 'Floor 1 - B Fasi', 1, 450, 472, '-', '-', '-', '-
')</statement>

</sql>
```

Όπως παρατηρείται, η βάση δεδομένων έχει γραφτεί σε .xml αρχείο και η οποία όπως αναφέρθηκε αποτελείται από τέσσερα tables, τα οποία αν δεν υπάρχουν δημιουργούνται εκείνη τη στιγμή (CREATE TABLE IF NOT EXISTS). Το κάθε table έχει κάποιες μεταβλητές διαφόρων τύπων (integer, varchar). Επίσης, η κάθε γραμμή καταγραφής δεδομένων αποτελεί ένα statement στο οποία καταγράφονται τα δεδομένα των μεταβλητών αυτών για την κάθε περίπτωση π.χ. INSERT INTO path (fromNode, toNode, cost) VALUES.

Το διάβασμα του .xml αρχείου και η μετατροπή του σε SQLiteDatabase γίνεται με τον παρακάτω κώδικα:

```
public void onCreate(SQLiteDatabase db) {
            String s;
            try {
```

```
                    Toast.makeText(context, "1", 2000).show();
                    InputStream in =
context.getResources().openRawResource(R.raw.sqlall);
                    DocumentBuilder builder =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
                    Document doc = builder.parse(in, null);
                    NodeList statements =
doc.getElementsByTagName("statement");
                    for (int i = 0; i < statements.getLength(); i++) {
                          s =
statements.item(i).getChildNodes().item(0).getNodeValue();
                          db.execSQL(s);
                    }
              } catch (Throwable t) {
                    Toast.makeText(context, t.toString(), 50000).show();
              }
        }
```

Αυτό που κάνει είναι να πάρει το sqlall.xml αρχείο από το φάκελο /r/raw, να το ανοίξει και να το διαβάσει. Έπειτα ακολουθεί ο xml parser ο οποίος επεξεργάζεται το περιεχόμενο, αναγνωρίζει τις εσωτερικές οντότητες, και ελέγχει αν κάθε οντότητα είναι καλός ορισμένη (well-formed). Τα δεδομένα από τα “statement” διαβάζονται και εκτελούν μια SQL στην db η οποία είναι SQLiteDatabase.

Κεφάλαιο 6

Μελλοντικές επεκτάσεις

Στο κεφάλαιο αυτό, θα αναφερθούν οι πιθανές μελλοντικές επεκτάσεις και βελτιώσεις της εφαρμογής. Η εφαρμογή αυτή μπορεί να χρησιμοποιηθεί και σε άλλα κτήρια. Η διαδικασία επέκτασης της εφαρμογής της σε άλλα κτήρια αποτελείται από δύο στάδια, τα οποία είναι:

1. Να περαστούν οι χάρτες των νέων κτηρίων σε μορφή εικόνας.

2. Να κατασκευαστεί μια νέα βάση δεδομένων σε .xml αρχείο. Η βάση θα αποτελείται από δύο πίνακες (tables). Στον πρώτο πίνακα θα καταχωρηθούν οι κόμβοι δηλαδή ονομασία κόμβου και οι συντεταγμένες του. Κατόπιν, στο δεύτερο πίνακα θα καταχωρηθούν ποιοι κόμβοι επικοινωνούν με άλλους κόμβους και το κόστος της μετάβασης από τον ένα κόμβο στον άλλο.

Συνεπώς, αν γίνουν αυτές οι αλλαγές, δηλαδή αντικατασταθούν οι υπάρχοντες χάρτες με τους χάρτες άλλου κτηρίου και τα δεδομένα του αρχείου sqlall.xml αντικατασταθούν με τα νέα δεδομένα, τότε μπορεί η εφαρμογή να τρέξει και να πλοηγήσει τον χρήστη στα νέα κτήρια.

Στη συγκεκριμένη εφαρμογή θεωρείται δεδομένη η θέση του χρήστη, η οποία καταχωρείται από τον ίδιο το χρήστη κατά τη διάρκεια της πλοήγησης. Επομένως, η εύρεση της θέσης του χρήστη χρησιμοποιώντας μία από τις προαναφερθείσες μεθόδους εντοπισμού θέσης σε εσωτερικό χώρο αποτελεί μονόδρομο περαιτέρω επέκτασης της εφαρμογής. Έχοντας τη θέση του χρήστη μπορεί να πραγματοποιηθεί πλοήγηση σε πραγματικό χρόνο. Σε μια τέτοια περίπτωση μπορεί επίσης να ελέγχεται αν ο χρήστης έχει απομακρυνθεί από την διαδρομή που πρέπει να ακολουθήσει. Αν έχει απομακρυνθεί και είναι εκτός διαδρομής τότε μπορεί να γίνεται επαναπροσδιορισμός της ελάχιστης διαδρομής προς τον προορισμό.

Επιπλέον, με βάση τη θέση του χρήστη θα ήταν πολύ σημαντικό να του εμφάνιζε η εφαρμογή τα πιο κοντινά προς αυτόν σημεία ενδιαφέροντος, χωρίς να χρειάζεται να ψάξει που υπάρχουν και αν είναι κοντά του. Αυτή η υπηρεσία θα μπορούσε να χρησιμοποιηθεί και σε ένα σουπερμάρκετ για την εύρεση προϊόντων ή σε ένα πολυκατάστημα για τη εύρεση της πιο κοντινής προς τον χρήστη τουαλέτας ή εξόδου.

Για άτομα με ειδικές ανάγκες, θα πρέπει να υπάρχει επιλογή για το αν μπορούν να χρησιμοποιήσουν σκαλιά ή αν επιθυμούν τη χρησιμοποίηση ασανσέρ, οπότε δε θα πρέπει να βρεθεί η πιο σύντομη διαδρομή αλλά μια διαδρομή που να συμβαδίζει με τους περιορισμούς που θέτει ο χρήστης. Συνεπώς, θα πρέπει να υπάρχει μενού για να θέτει ο χρήστης τους περιορισμούς που επιθυμεί.

Σημαντικό κομμάτι της εφαρμογής αποτελούν οι φωνητικές οδηγίες. Οι φωνητικές οδηγίες είναι μόνο στα αγγλικά. Συνεπώς, χρήσιμο θα ήταν να δινόταν η δυνατότητα στο χρήστη να επιλέξει τη γλώσσα των φωνητικών οδηγιών. Αντίστοιχα, θα μπορούσαν να αλλάζουν και τα οπτικά μηνύματα στην επιλεγμένη γλώσσα.

Βιβλιογραφία

[1] Andrew S. Tanenbaum, «Δίκτυα Υπολογιστών»,
Εκδόσεις Κλειδάριθμος, 2003.

[2] James F. Kurose, Keith W. Ross, "Computer Networking: A Top-Down Approach Featuring the Internet (5th edition)", 2009.

[3] Μ. Ε. Θεολόγου, «Δίκτυα Κινητών και Προσωπικών Επικοινωνιών»,
Εκδόσεις Τζιόλα, Αθήνα, 2008.

[4] Hui Liu, H. Darabi, P. Banerjee, and Jing Liu, "Survey of Wireless Indoor Positioning Techniques and Systems," IEEE Transactions Systems, Man, and Cybernetics, Nov. 2007.

[5] Tsung-Nan Lin and Po-Chiang Lin, "Performance Comparison of Indoor Positioning Techniques based on Location Fingerprinting in Wireless Networks," in Proc. International Conference on Wireless Networks, Communications and Mobile Computing, June, 2005.

[6] P. Bahl and V. N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in Proc. IEEE INFOCOM 2000, Mar.

[7] M. Brunato and R. Battiti, "Statistical learning theory for location fingerprinting in wireless lans," Comput. Netw. ISDN Syst., 2005.

[8] Justin Stook, "Planning an indoor navigation service for a smartphone with Wi-Fi fingerprinting localization", 2011.

[9] Vlad Badea, Rikard Ericksson "Indoor navigation with pseudolites (fake GPS sat.)", 2005.

[10] Dimitris Saragas, Nathan Webb, "Indoor Navigation System for Handheld Devices", 2009.

[11] Hui Liu, Student Member, IEEE, Houshang Darabi, Member, IEEE, Pat Banerjee, and Jing Liu, "Survey of Wireless Indoor Positioning Techniques and Systems", 2007.

[12] Björn Greßmann, Helge Klimek, Volker Turau, "Towards Ubiquitous Indoor Location Based Services and Indoor Navigation", Institute of Telematics Hamburg University of Technology, Hamburg, Germany, 2010.

[13] Busra OZDENIZCI, Kerem OK, Vedat COSKUN, Mehmet N. AYDIN, "Development of an Indoor Navigation System Using NFC Technology", Department of Information Technologies, ISIK University, Istanbul, Turkey, 2011.

[14] V. Zeimpekis, G. Giaglis, G. Lekakos, "A Taxonomy of Indoor and Outdoor Positioning Techniques for Mobile Location Services", 2003.

[15] Marco Porretta, Paolo Nepa, Giuliano Manara, Filippo Giannetti, “Location, Location, Location”, June 2008.

[16] Nicola Lenihan, “A Local Optimal User Position System for Indoor Wireless Devices”, 2004.

[17] TelecomCircle, “Introduction to Location Based Services”, http://www.telecomcircle.com/2009/06/introduction-to-lbs.

[18] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische mathematik, 1(1), 269–271.

[19] Παναγιώτης Σ. Γεώργας, «Μοντελοποίηση Συστημάτων Εντοπισμού Θέσης Εσωτερικού Χώρου με χρήση “Δαχτυλικών Αποτυπωμάτων”», 2009.

[20] Αλέξανδρος Σ. Ανδρονικάκης, «Σύστημα Προσδιορισμού θέσης Σταθμών Βάσης σε Δίκτυα Κινητών Επικοινωνιών», 2010.

[21] Θεόδωρος Η. Μπούρχας, «Ανάπτυξη Πλατφόρμας Δυναμικής Παροχής Υπηρεσιών Βασισμένων στη Θέση των Χρηστών», 2011.

[22] Android Developers- http://developer.android.com/guide/index.html

[23] Kathy Sierra, Bert Bates, «Head First Java», O’REILLY,2005.

[24] Shane Conder, Lauren Darcey, “Android Wireless Application Development”, 2010

[25] S. Y. Hashimi, D. MacLean, S. Komatineni, “Pro Android 3”, Apress, 2011.

[26] Grant Allen, Mike Owens, “The Definitive Guide to SQLite”, Apress, 2010.

Παράρτημα

Πηγαίος Κώδικας Εφαρμογής

Start.java

```
package com.ece;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.preference.PreferenceManager;

public class Start extends Activity {

	MediaPlayer ourSong;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		setContentView(R.layout.start);
		ourSong = MediaPlayer.create(Start.this, R.raw.song);

		SharedPreferences getPrefs =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
		boolean music = getPrefs.getBoolean("checkboxMusic", true);
		if (music == true)
			ourSong.start();

		Thread timer = new Thread() {
			public void run() {
				try {
					sleep(4000);
					System.gc();
				} catch (InterruptedException e) {
					e.printStackTrace();
				} finally {
					startActivity(new
Intent("com.ece.MAPACTIVITY"));
				}
			}
		};
		timer.start();
	}

	@Override
	protected void onPause() {
		// TODO Auto-generated method stub
		super.onPause();
		ourSong.release();
		System.gc();
```

```
                finish();
        }
}
```

MapActivity.java

```
package com.ece;

import java.util.Locale;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.media.AudioManager;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.speech.tts.TextToSpeech;
import android.speech.tts.TextToSpeech.OnInitListener;
import android.view.Gravity;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.data.MapIntent;
import com.data.PassData;
import com.data.PointsFloor;
import com.view.MapView;
import com.view.Navigation;

public class MapActivity extends Activity implements OnInitListener {

        boolean isView = true;
        boolean isFirst = true;
        TextToSpeech tts;
        Navigation view;
        PassData flooraction;
        int floor;
        String nametxt;
        public String source, destination, name_source, name_destination;
        public int x1, y1, z1, x2, y2, z2, xs[], ys[], zs[];
        int i = 0;
        boolean isPaused = false;;
```

```
PointsFloor drawpoints = null;
int[] pinakas = new int[1000];
int indexCovered = 0;
String[] updown = { "0", "1", "2", "3" };
int[] updownX = new int[5];
int[] updownY = new int[5];
int l = 0;
boolean isOK = true;

public void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	Bundle extras = this.getIntent().getExtras();
	if (extras != null) {
		isFirst = false;
		isView = extras.getBoolean("view");
		if (isView) {
			String action = extras.getString("action");
			String ok = extras.getString("ok");
			if (action != null) {
				int floor = extras.getInt("floor");
				z2 = floor;
				showFloor(floor);
			} else if (ok != null) {
				int floor = extras.getInt("z");
				z2 = floor;
				int x = extras.getInt("x");
				int y = extras.getInt("y");
				String nametxt = extras.getString("name");
				drawpoints = new PointsFloor(x, y, floor, nametxt);
				showFloor(floor);
			} else
				showFloor(0);
		} else if (extras != null) {
			source = extras.getString("source");
			name_source = extras.getString("name_source");
			x1 = extras.getInt("x1");
			y1 = extras.getInt("y1");
			z1 = extras.getInt("z1");
			destination = extras.getString("destination");
			name_destination = extras.getString("name_destination");
			x2 = extras.getInt("x2");
			y2 = extras.getInt("y2");
			z2 = extras.getInt("z2");
			xs = extras.getIntArray("xs");
			ys = extras.getIntArray("ys");
			zs = extras.getIntArray("zs");
			showFloor(z1);
			isPaused = false;
		}
	} else {
```

```
                        SharedPreferences getPrefs =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
                        String select = getPrefs.getString("floorlist", "0");
                        int selectFloor = 0;
                        try {
                                selectFloor = Integer.parseInt(select);
                        } catch (NumberFormatException nfe) {
                                // Handle parse error.
                        }
                        showFloor(selectFloor);
                }
                tts = new TextToSpeech(this, this);
        }

        public void onInit(int status) {
                tts.setLanguage(Locale.US);
                if (isFirst) {
                        String message = "Hello, I am Ntuadroid and I will help you to find
your destination. You are on the Floor " + flooraction.getdata();
                        onlySpeak(message);
                        Toast toast = Toast.makeText(this, message, 25000);
                        toast.setGravity(Gravity.BOTTOM, 0, 0);
                        toast.show();
                } else if (isView)
                        onlySpeak("Floor " + flooraction.getdata());
                else {
                        message("Follow the blue line in order to reach your destination");
                }
        }

        public void change(final boolean changeStatus) {
                isPaused = changeStatus;
        }

        public void onlySpeak(final String text) {
                SharedPreferences getPrefs =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
                boolean voice = getPrefs.getBoolean("checkboxVoice", true);
                if (voice == true) {
                        tts.speak(text, TextToSpeech.QUEUE_FLUSH, null);
                }
        }

        public void message(final String message) {
                if (isOK) {
                        SharedPreferences getPrefs =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
                        boolean voice = getPrefs.getBoolean("checkboxVoice", true);
                        if (voice == true) {
                                tts.speak(message, TextToSpeech.QUEUE_FLUSH, null);
```

```
                        }
                        this.runOnUiThread(new Runnable() {
                                public void run() {
                                        Toast toast = Toast.makeText(MapActivity.this,
message, 8000);
                                        toast.setGravity(Gravity.BOTTOM, 0, 0);
                                        toast.show();
                                }
                        });
                }
        }

        public void showFloor(int floor) {

                if (isView) {
                        if (drawpoints == null) {
                                int x = 0;
                                int y = 0;
                                nametxt = null;
                                setContentView(new MapView(this, null, floor, x, y,
nametxt));
                        } else {
                                setContentView(new MapView(this, null, floor,
drawpoints.getcoX(), drawpoints.getcoY(), drawpoints.getS()));
                        }
                } else {
                        view = new Navigation(this, null, x1, y1, z1, x2, y2, z2, xs, ys, zs,
name_source, name_destination, this);
                        setContentView(view);
                }

                flooraction = new PassData(floor);
                if (!isFirst) {
                        Toast toast = Toast.makeText(this, "Floor " + floor, 4000);
                        toast.setGravity(Gravity.BOTTOM, 0, 0);
                        toast.show();
                }
        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
                super.onCreateOptionsMenu(menu);
                menu.add(0, 0, 0, "Find
Destination").setIcon(android.R.drawable.ic_menu_directions);
                menu.add(0, 1, 0, "Select
Floor").setIcon(android.R.drawable.ic_dialog_map);
                menu.add(0, 2, 0, "POI").setIcon(android.R.drawable.ic_dialog_info);
                menu.add(0, 3, 0,
"Settings").setIcon(android.R.drawable.ic_menu_preferences);
                menu.add(0, 4, 0, "Volume").setIcon(android.R.drawable.ic_input_add);
```

```
        menu.add(0, 5, 0, "Back").setIcon(android.R.drawable.ic_input_delete);
        return true;

}

public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
        case 0:
                message("Where are you? Choose your position");
                Intent intent = new Intent(this, com.search.SourceList.class);
                intent.putExtra("Floor1", z2);
                startActivity(intent);
                if (!isView) {
                        isOK = false;
                        finish();
                        break;
                }
                finish();
                return true;

        case 1:
                onlySpeak("Select floor");
                Intent intent1 = new Intent(this, com.ece.Floors.class);
                intent1.putExtra("Floor1", z2);
                startActivity(intent1);
                if (!isView) {
                        isOK = false;
                        finish();
                        System.exit(0);
                        break;
                }
                finish();
                return true;

        case 2:
                onlySpeak("Point of Interest");
                Intent intent2 = new Intent(this, com.ece.Poi.class);
                intent2.putExtra("Floor1", z2);
                startActivity(intent2);
                if (!isView) {
                        isOK = false;
                        finish();
                        System.exit(0);
                        break;
                }
                finish();
                return true;

        case 3:
                onlySpeak("Settings");
```

```
                        Intent intent3 = new Intent(this, com.ece.Prefs.class);
                        intent3.putExtra("Floor1", z2);
                        startActivity(intent3);
                        return true;

                case 4:
                        onlySpeak("Volume");
                        final AlertDialog.Builder alert2 = new AlertDialog.Builder(this);

                        // get the tool to change the media volume of the phone
                        final AudioManager am = (AudioManager)
getSystemService(Context.AUDIO_SERVICE);
                        double maxVol =
am.getStreamMaxVolume(AudioManager.STREAM_MUSIC);
                        final double curVol =
am.getStreamVolume(AudioManager.STREAM_MUSIC);

                        LinearLayout ll = new LinearLayout(this);
                        ll.setOrientation(LinearLayout.VERTICAL);
                        final VolumeBar b = new VolumeBar(this, tts, am);

                        b.measure(150, 30);
                        b.layout(0, 0, 150, 30);
                        b.forceLayout();
                        b.setPadding(15, 10, 15, 10);

                        // Set up the bar to start at the current volume
                        b.setProgress((int) (b.getMax() * curVol / maxVol));
                        TextView tv = new TextView(this);
                        tv.setText("Speech Volume:");
                        ll.addView(tv);
                        ll.addView(b);
                        alert2.setView(ll);

                        alert2.setPositiveButton("Ok", new
DialogInterface.OnClickListener() {
                                public void onClick(DialogInterface dialog, int whichButton)
{

                                }
                        });
                        alert2.setNegativeButton("Cancel", new
DialogInterface.OnClickListener() {
                                public void onClick(DialogInterface dialog, int whichButton)
{
                                        dialog.cancel();

        am.setStreamVolume(AudioManager.STREAM_MUSIC, (int) curVol, 0);
                                }
                        });
```

```
                    Dialog alertDialog = alert2.create();
                    alertDialog.setTitle("Options");
                    alertDialog.getWindow().setLayout(15, 10);
                    alertDialog.show();

                    return true;

                case 5:
                    onlySpeak("Back");
                    isOK = false;
                    finish();
                    System.exit(0);
                    break;
                }

                return false;
        }

        public int changeSpeed() {
                SharedPreferences getPrefs =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
                String select = getPrefs.getString("speed", "30");
                int speed = 30;
                try {
                    speed = Integer.parseInt(select);
                } catch (NumberFormatException nfe) {
                    // Handle parse error.
                }
                return speed;
        }

        public void onBackPressed() {
                if (!isView) {
                    startActivity(new Intent("com.search.SOURCELIST"));
                    finish();
                    System.exit(0);
                } else {
                    startActivity(MapIntent.openFloor(this.getApplicationContext(), 0));
                    finish();
                    return;
                }
                return;
        }

}
```

MapView.java

```
package com.view;

import java.text.NumberFormat;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Matrix;
import android.graphics.Paint;
import android.graphics.Picture;
import android.graphics.Typeface;
import android.graphics.drawable.PaintDrawable;
import android.util.AttributeSet;
import android.view.GestureDetector;
import android.view.Gravity;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.view.GestureDetector.OnGestureListener;
import android.view.ViewGroup.LayoutParams;
import android.widget.LinearLayout;
import android.widget.Scroller;
import android.widget.TextView;
import android.widget.ZoomButtonsController;
import android.widget.ZoomButtonsController.OnZoomListener;

import com.ece.R;

public class MapView extends View implements OnGestureListener, OnZoomListener {

	int floor = 0;
	int x = 0;
	int y = 0;
	int i = 0;
	String txtname;
	private static float[] SRC = new float[] { 0, 0 };

	private static final int WIDTH = 1599;
	private static final int HEIGHT = 1023;
	private static final float MIN_ZOOM = 0.1f;
	private static final float MAX_ZOOM = 3;

	boolean notTouched = true;
	float followX, followY;
	private float mX;
	private float mY;
```

```
        private Scroller mScroller;
        private GestureDetector mGestureDetector;
        private float mScale;
        private Picture mPicture;
        private ZoomButtonsController mZoomController;
        private NumberFormat mZoomFormat;
        private TextView mZoomLabel;
        private Matrix mMatrix;
        private float[] mDst;
        public Bitmap image;
        BitmapFactory.Options options;

        public static Bitmap image0;
        public static Bitmap image1;
        public static Bitmap image2;
        public static Bitmap image3;

        public static void initBitmaps(View this_) {
                if (image0 != null)
                        return;
                Bitmap image;
                BitmapFactory.Options options = new BitmapFactory.Options();
                image = BitmapFactory.decodeResource(this_.getResources(),
R.drawable.floor0, options);
                image0 = Bitmap.createScaledBitmap(image, WIDTH, HEIGHT, true);
                image.recycle();
                image = BitmapFactory.decodeResource(this_.getResources(),
R.drawable.floor1, options);
                image1 = Bitmap.createScaledBitmap(image, WIDTH, HEIGHT, true);
                image.recycle();
                image = BitmapFactory.decodeResource(this_.getResources(),
R.drawable.floor2, options);
                image2 = Bitmap.createScaledBitmap(image, WIDTH, HEIGHT, true);
                image.recycle();
                image = BitmapFactory.decodeResource(this_.getResources(),
R.drawable.floor3, options);
                image3 = Bitmap.createScaledBitmap(image, WIDTH, HEIGHT, true);
                image.recycle();
        }

        public MapView(Context context, AttributeSet attrs, int floor, int x, int y, String
txtname) {
                super(context, attrs);
                this.floor = floor;
                this.x = x;
                this.y = y;
                if ((x == 0) && (y == 0)) {
                        notTouched = false;
                }
                followX = x;
```

```
        followY = y;
        this.txtname = txtname;
        initBitmaps(this);
        if (floor == 0) {
                image = image0;
        } else if (floor == 1) {
                image = image1;
        } else if (floor == 2) {
                image = image2;
        } else {
                image = image3;
        }

        mScroller = new Scroller(context);
        mGestureDetector = new GestureDetector(this);
        mScale = 0.5f;
        mZoomController = new ZoomButtonsController(this);
        mZoomController.setAutoDismissed(true);
        mZoomController.setOnZoomListener(this);
        mZoomController.setZoomSpeed(1);
        mZoomController.setZoomInEnabled(mScale < MAX_ZOOM);
        mZoomController.setZoomOutEnabled(mScale > MIN_ZOOM);
        makeZoomLabel(context, mZoomController);

        mZoomFormat = NumberFormat.getPercentInstance();
        mZoomLabel.setText("Zoom: " + mZoomFormat.format(mScale));

        mPicture = makeContent();
        setVerticalScrollBarEnabled(true);
        setHorizontalScrollBarEnabled(true);
        mMatrix = new Matrix();
        mDst = new float[2];
        System.gc();

}

protected void onDetachedFromWindow() {
        super.onDetachedFromWindow();
        mZoomController.setVisible(false);
}

public boolean onTouchEvent(MotionEvent event) {
        return mGestureDetector.onTouchEvent(event);
}

protected void onDraw(Canvas canvas) {
        canvas.save();
        if (mScroller.computeScrollOffset()) {
                mX = mScroller.getCurrX();
                mY = mScroller.getCurrY();
```

```
                invalidate();
            }

            mMatrix.reset();
            mMatrix.preTranslate(mX * mScale, mY * mScale);

            int w = getWidth();
            int h = getHeight();
            float pivotX = Math.max(Math.min(-mX, w / 2), 2 * w - WIDTH - mX);
            float pivotY = Math.max(Math.min(-mY, h / 2), 2 * h - HEIGHT - mY);
            mMatrix.preScale(mScale, mScale, pivotX, pivotY);
            if (notTouched) {
                mMatrix.postTranslate(w / 2f - mScale * followX, h / 2f - mScale *
followY);
            }
            canvas.concat(mMatrix);

            // draw content
            mPicture.draw(canvas);
            canvas.restore();
        }

        protected int computeHorizontalScrollExtent() {
            return Math.round(computeHorizontalScrollRange() * getWidth() / (WIDTH
* mScale));
        }

        protected int computeHorizontalScrollOffset() {
            mMatrix.mapPoints(mDst, SRC);
            float x = -mDst[0] / mScale;
            return Math.round(computeHorizontalScrollRange() * x / WIDTH);
        }

        protected int computeVerticalScrollExtent() {
            return Math.round(computeVerticalScrollRange() * getHeight() / (HEIGHT
* mScale));
        }

        protected int computeVerticalScrollOffset() {
            mMatrix.mapPoints(mDst, SRC);
            float y = -mDst[1] / mScale;
            return Math.round(computeVerticalScrollRange() * y / HEIGHT);
        }

        public boolean onDown(MotionEvent e) {
            mZoomController.setVisible(false);
            return true;
        }
```

```
        public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float
velocityY) {
                int minX = (int) (getWidth() - WIDTH);
                int minY = (int) (getHeight() - HEIGHT);
                mScroller.fling((int) mX, (int) mY, (int) velocityX, (int) velocityY, minX, 0,
minY, 0);
                invalidate();
                return true;
        }

        public void onLongPress(MotionEvent e) {
                mZoomController.setVisible(true);
                notTouched = false;
        }

        public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float
distanceY) {
                mX -= distanceX / mScale;
                mY -= distanceY / mScale;
                mX = Math.max(getWidth() - WIDTH, Math.min(0, mX));
                mY = Math.max(getHeight() - HEIGHT, Math.min(0, mY));
                notTouched = false;
                invalidate();
                return true;
        }

        public void onShowPress(MotionEvent e) {
        }

        public boolean onSingleTapUp(MotionEvent e) {
                return true;
        }

        public void onVisibilityChanged(boolean visible) {
        }

        public void onZoom(boolean zoomIn) {
                mScale += zoomIn ? 0.1 : -0.1;
                mScale = Math.min(MAX_ZOOM, Math.max(MIN_ZOOM, mScale));
                mZoomLabel.setText("Zoom: " + mZoomFormat.format(mScale));
                System.out.println("onzoom");
                invalidate();

                mZoomController.setZoomInEnabled(mScale != MAX_ZOOM);
                mZoomController.setZoomOutEnabled(mScale != MIN_ZOOM);
        }

        private void makeZoomLabel(Context context, ZoomButtonsController
zoomController) {
                ViewGroup container = zoomController.getContainer();
```

```
                View controls = zoomController.getZoomControls();
                LayoutParams p0 = controls.getLayoutParams();
                container.removeView(controls);
                LinearLayout layout = new LinearLayout(context);
                layout.setOrientation(LinearLayout.VERTICAL);
                mZoomLabel = new TextView(context);
                mZoomLabel.setPadding(12, 0, 12, 0);
                mZoomLabel.setTypeface(Typeface.DEFAULT_BOLD);
                mZoomLabel.setTextColor(0xff000000);
                PaintDrawable d = new PaintDrawable(0xeeffffff);
                d.setCornerRadius(6);
                mZoomLabel.setBackgroundDrawable(d);
                mZoomLabel.setTextSize(20);
                mZoomLabel.setGravity(Gravity.CENTER_HORIZONTAL);
                LinearLayout.LayoutParams p1 = new
LinearLayout.LayoutParams(LayoutParams.WRAP_CONTENT,
LayoutParams.WRAP_CONTENT);
                p1.gravity = Gravity.CENTER_HORIZONTAL;
                layout.addView(mZoomLabel, p1);
                layout.addView(controls);
                container.addView(layout, p0);
        }

        private Picture makeContent() {

                Picture picture = new Picture();
                Paint paint = new Paint();
                Paint points = new Paint();
                Paint txt = new Paint();
                points.setColor(Color.BLUE);
                points.setStyle(Paint.Style.FILL_AND_STROKE);
                points.setStrokeWidth(10);
                txt.setColor(Color.BLACK);
                txt.setTextSize(20);
                Canvas c = picture.beginRecording(WIDTH, HEIGHT);
                c.drawBitmap(image, 0, 0, paint);
                if ((x != 0) && (x != 428) && (x != 438) && (x != 460) && (x != 423) &&
(x != 563) && (y != 0)) {
                        c.drawCircle(x, y, 4, points);
                        c.drawText("Here is: " + txtname, x - 75, y - 30, txt);
                } else if (x == 428) {
                        c.drawCircle(x, y, 4, points);
                        c.drawText("Here is: " + txtname, x - 50, y - 30, txt);
                        if (floor < 3) {
                                c.drawCircle(992, 637, 4, points);
                                c.drawText("Here is: " + txtname, 950, 620, txt);
                        }
                        if (floor == 0) {
                                c.drawCircle(290, 737, 4, points);
                                c.drawText("Here is: " + txtname, 240, 720, txt);
```

```
                }
            } else if (x == 438) {
                c.drawCircle(x, y, 4, points);
                c.drawText("Here is: " + txtname, x - 75, y - 30, txt);
                c.drawCircle(1004, 90, 4, points);
                c.drawText("Here is: " + txtname, 930, 75, txt);
            } else if (x == 563) {
                c.drawCircle(x, y, 4, points);
                c.drawText("Here is: " + txtname + " 3", x - 75, y - 30, txt);
                c.drawCircle(1547, 700, 4, points);
                c.drawText("Here is: " + txtname, 1472, 685, txt);
                c.drawCircle(1300, 728, 4, points);
                c.drawText("Here is: " + txtname + " 1", 1250, 713, txt);
                c.drawCircle(1300, 670, 4, points);
                c.drawText("Here is: " + txtname + " 2", 1250, 665, txt);
                c.drawCircle(563, 660, 4, points);
                c.drawText("Here is: " + txtname + " 4", 513, 645, txt);
                c.drawCircle(465, 580, 4, points);
                c.drawText("Here is: " + txtname + " 5", 390, 565, txt);
                c.drawCircle(465, 490, 4, points);
                c.drawText("Here is: " + txtname + " 6", 390, 475, txt);
                c.drawCircle(942, 875, 4, points);
                c.drawText("Here is: Main Entrance - Exit", 820, 860, txt);

            } else if (x == 460) {
                c.drawCircle(x, y, 4, points);
                c.drawText("Here is: " + txtname, x - 50, y - 30, txt);
                if (floor < 3) {
                    c.drawCircle(1030, 575, 4, points);
                    c.drawText("Here is: " + txtname, 980, 560, txt);
                }
            } else if (x == 423) {
                c.drawCircle(x, y, 4, points);
                c.drawText("Here is: " + txtname, x - 50, y - 30, txt);
                if (floor == 0) {
                    c.drawCircle(1380, 672, 4, points);
                    c.drawText("Here is: " + txtname, 1330, 657, txt);
                    c.drawCircle(1030, 625, 4, points);
                    c.drawText("Here is: " + txtname, 980, 610, txt);
                } else if (floor < 3) {
                    c.drawCircle(1030, 625, 4, points);
                    c.drawText("Here is: " + txtname, 980, 610, txt);
                }
            }

            picture.endRecording();
            System.gc();
            return picture;
        }
}
```

Navigation.java

```
package com.view;

import java.text.NumberFormat;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Matrix;
import android.graphics.Paint;
import android.graphics.Path;
import android.graphics.Picture;
import android.graphics.Typeface;
import android.graphics.drawable.PaintDrawable;
import android.util.AttributeSet;
import android.view.GestureDetector;
import android.view.Gravity;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.view.GestureDetector.OnGestureListener;
import android.view.ViewGroup.LayoutParams;
import android.widget.LinearLayout;
import android.widget.Scroller;
import android.widget.TextView;
import android.widget.ZoomButtonsController;
import android.widget.ZoomButtonsController.OnZoomListener;

import com.ece.MapActivity;

public class Navigation extends View implements OnGestureListener, OnZoomListener {

	int numberAgain = 0;
	int floor = 0;
	int x1 = 0;
	int y1 = 0;
	int x2 = 0;
	int y2 = 0;
	int z2 = 0;
	int i = 0;
	int l = 0;
	int u = 0;
	int[] pinakas = new int[5000];
	int indexCovered = 0;
	String name_source, name_destination;
	String[] updown = { "0", "1", "2", "3", "4", "5", "6", "7", "8" };
	int[] updownX = new int[5];
```

```
        int[] updownY = new int[5];
        int[] xs = {};
        int[] ys = {};
        int[] zs = {};
        boolean startAgain = true;
        boolean isFirst = true;
        int number = 0;
        int j = 0;
        int k = 0;
        int count = 0;
        int count1 = 0;
        int counter = 0;
        int counter1 = 0;
        int counter2 = 0;
        boolean isAnimation = true;
        float followX, followY;
        boolean pass = true;
        boolean passfloor = true;
        boolean passagain = false;
        boolean changefloor = true;
        boolean changeFloorAfter = false;
        boolean changeFloorBefore = false;
        boolean FirstTime = true;
        private static float[] SRC = new float[] { 0, 0 };

        private static final int WIDTH = 1599;
        private static final int HEIGHT = 1023;
        private static final float MIN_ZOOM = 0.1f;
        private static final float MAX_ZOOM = 3;

        private float mX;
        private float mY;
        private Scroller mScroller;
        private GestureDetector mGestureDetector;
        private float mScale;
        private Picture mPicture;
        private ZoomButtonsController mZoomController;
        private NumberFormat mZoomFormat;
        private TextView mZoomLabel;
        private Matrix mMatrix;
        private float[] mDst;
        public Bitmap image0, image1, image2, image3;
        BitmapFactory.Options options;
        private MapActivity activity;

        public Navigation(Context context, AttributeSet attrs, int x1, int y1, int floor, int x2,
int y2, int z2, int[] xs, int[] ys, int[] zs, String name_source,
                        String name_destination, MapActivity activity) {
                super(context, attrs);
                this.floor = floor;
```

```
                this.z2 = z2;
                this.x1 = x1;
                this.y1 = y1;
                this.x2 = x2;
                this.y2 = y2;
                this.z2 = z2;
                this.xs = xs;
                this.ys = ys;
                this.zs = zs;
                this.name_source = name_source;
                this.name_destination = name_destination;
                this.activity = activity;
                MapView.initBitmaps(this);
                image0 = MapView.image0;
                image1 = MapView.image1;
                image2 = MapView.image2;
                image3 = MapView.image3;

                System.gc();
                mScroller = new Scroller(context);
                mGestureDetector = new GestureDetector(this);
                mScale = 0.5f;
                mZoomController = new ZoomButtonsController(this);
                mZoomController.setAutoDismissed(true);
                mZoomController.setOnZoomListener(this);
                mZoomController.setZoomSpeed(1);
                mZoomController.setZoomInEnabled(mScale < MAX_ZOOM);
                mZoomController.setZoomOutEnabled(mScale > MIN_ZOOM);
                makeZoomLabel(context, mZoomController);

                mZoomFormat = NumberFormat.getPercentInstance();
                mZoomLabel.setText("Zoom: " + mZoomFormat.format(mScale));

                coveredLength = 0f;
                totalLength = getLengthUpToPoint(xs.length - 1);
                setVerticalScrollBarEnabled(true);
                setHorizontalScrollBarEnabled(true);
                mMatrix = new Matrix();
                mDst = new float[2];
                find();
                makepicture();
                System.gc();

        }

        private void find() {
                for (int w = 0; w <= (xs.length - 1); w++) {
                        if (((xs[w] == 1032) && (ys[w] == 575)) || ((xs[w] == 465) &&
(ys[w] == 543))) {
                                updown[l] = "Elevator";
```

```
                        updownX[l] = xs[w];
                        updownY[l] = ys[w];
                        l++;
                    } else if (((xs[w] == 1032) && (ys[w] == 637)) || ((xs[w] == 425)
&& (ys[w] == 543)) || ((xs[w] == 1405) && (ys[w] == 670))) {
                        updown[l] = "Stairs";
                        updownX[l] = xs[w];
                        updownY[l] = ys[w];
                        l++;
                    } else if ((xs[w] == 1405) && (ys[w] == 728)) {
                        updown[l] = "Stairs";
                        updownX[l] = xs[w];
                        updownY[l] = ys[w];
                        l++;
                    }
                }
            }

            private float getLengthUpToPoint(int index) {
                float l = 0f;
                for (int i = 1; i <= index; i++) {
                    x1 = xs[i - 1];
                    y1 = ys[i - 1];
                    x2 = xs[i];
                    y2 = ys[i];
                    l += Math.sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));
                }
                return l;
            }

            private int getIndexCoveredByLength(float length) {
                for (int i = 1; i < xs.length; i++) {
                    x1 = xs[i - 1];
                    y1 = ys[i - 1];
                    x2 = xs[i];
                    y2 = ys[i];
                    length -= Math.sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));
                    if (length <= 0)
                        return i - 1;
                }
                return xs.length - 1;
            }

            protected void onDetachedFromWindow() {
                super.onDetachedFromWindow();
                mZoomController.setVisible(false);
            }

            public boolean onTouchEvent(MotionEvent event) {
                return mGestureDetector.onTouchEvent(event);
```

```
	}

	protected void onDraw(Canvas canvas) {
		canvas.save();
		if (mScroller.computeScrollOffset()) {
			mX = mScroller.getCurrX();
			mY = mScroller.getCurrY();
			invalidate();
		}

		mMatrix.reset();

		mMatrix.preTranslate(mX * mScale, mY * mScale);

		int w = getWidth();
		int h = getHeight();
		float pivotX = Math.max(Math.min(-mX, w / 2), 2 * w - WIDTH - mX);
		float pivotY = Math.max(Math.min(-mY, h / 2), 2 * h - HEIGHT - mY);
		mMatrix.preScale(mScale, mScale, pivotX, pivotY);
		mMatrix.postTranslate(w / 2f - mScale * followX, h / 2f - mScale *
followY);
		canvas.concat(mMatrix);

		// draw content
		mPicture.draw(canvas);
		canvas.restore();
	}

	protected int computeHorizontalScrollExtent() {
		return Math.round(computeHorizontalScrollRange() * getWidth() / (WIDTH
* mScale));
	}

	protected int computeHorizontalScrollOffset() {
		mMatrix.mapPoints(mDst, SRC);
		float x = -mDst[0] / mScale;
		return Math.round(computeHorizontalScrollRange() * x / WIDTH);
	}

	protected int computeVerticalScrollExtent() {
		return Math.round(computeVerticalScrollRange() * getHeight() / (HEIGHT
* mScale));
	}

	protected int computeVerticalScrollOffset() {
		mMatrix.mapPoints(mDst, SRC);
		float y = -mDst[1] / mScale;
		return Math.round(computeVerticalScrollRange() * y / HEIGHT);
	}
```

```
        public boolean onDown(MotionEvent e) {
                mZoomController.setVisible(false);
                return true;
        }

        public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float
velocityY) {
                int minX = (int) (getWidth() - WIDTH);
                int minY = (int) (getHeight() - HEIGHT);
                mScroller.fling((int) mX, (int) mY, (int) velocityX, (int) velocityY, minX, 0,
minY, 0);
                invalidate();
                return true;
        }

        public void onLongPress(MotionEvent e) {
                mZoomController.setVisible(true);
        }

        public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float
distanceY) {
                mX -= distanceX / mScale;
                mY -= distanceY / mScale;
                mX = Math.max(getWidth() - WIDTH, Math.min(0, mX));
                mY = Math.max(getHeight() - HEIGHT, Math.min(0, mY));
                invalidate();
                return true;
        }

        public void onShowPress(MotionEvent e) {
        }

        public boolean onSingleTapUp(MotionEvent e) {
                isPaused = !isPaused;
                activity.change(isPaused);
                if (isPaused)
                        activity.message("Pause");
                else
                        activity.message("Resume");
                return true;
        }

        public void onVisibilityChanged(boolean visible) {
        }

        public void onZoom(boolean zoomIn) {
                mScale += zoomIn ? 0.1 : -0.1;
                mScale = Math.min(MAX_ZOOM, Math.max(MIN_ZOOM, mScale));
                mZoomLabel.setText("Zoom: " + mZoomFormat.format(mScale));
                invalidate();
```

```
            mZoomController.setZoomInEnabled(mScale != MAX_ZOOM);
            mZoomController.setZoomOutEnabled(mScale != MIN_ZOOM);
        }

        private void makeZoomLabel(Context context, ZoomButtonsController
zoomController) {
            ViewGroup container = zoomController.getContainer();
            View controls = zoomController.getZoomControls();
            LayoutParams p0 = controls.getLayoutParams();
            container.removeView(controls);
            LinearLayout layout = new LinearLayout(context);
            layout.setOrientation(LinearLayout.VERTICAL);
            mZoomLabel = new TextView(context);
            mZoomLabel.setPadding(12, 0, 12, 0);
            mZoomLabel.setTypeface(Typeface.DEFAULT_BOLD);
            mZoomLabel.setTextColor(0xff000000);
            PaintDrawable d = new PaintDrawable(0xeeffffff);
            d.setCornerRadius(6);
            mZoomLabel.setBackgroundDrawable(d);
            mZoomLabel.setTextSize(20);
            mZoomLabel.setGravity(Gravity.CENTER_HORIZONTAL);
            LinearLayout.LayoutParams p1 = new
LinearLayout.LayoutParams(LayoutParams.WRAP_CONTENT,
LayoutParams.WRAP_CONTENT);
            p1.gravity = Gravity.CENTER_HORIZONTAL;
            layout.addView(mZoomLabel, p1);
            layout.addView(controls);
            container.addView(layout, p0);
        }

        final long interval = 100;
        int speed = 30;
        float coveredLength, totalLength;
        private Thread timer;
        private boolean isPaused = false;;

        private void makepicture() {

            mPicture = makeContent();
            timer = new Thread() {
                public void run() {
                    try {
                        sleep(6000);
                        while (true) {
                            sleep(interval);
                            if (changeFloorBefore) {
                                isPaused = !isPaused;
                                activity.change(isPaused);
```

```
                                                        if ((updownY[u] == 728) &&
(xs[indexCovered - 1] == 1300)) {
                                                                activity.message("Go
downstairs");
                                                                changeFloorBefore = false;
                                                                sleep(2000);
                                                        } else if ((updownY[u] == 728) &&
(xs[indexCovered - 1] == 1495)) {
                                                                activity.message("Go upstairs");
                                                                changeFloorBefore = false;
                                                                sleep(2000);
                                                        } else {
                                                                activity.message("Use " +
updown[u] + " to change floor. Go to Floor " + zs[indexCovered + 1]);
                                                                sleep(5000);
                                                        }
                                                        activity.message("Tap on the screen to
continue your navigation");
                                                        u = 2;
                                                }

                                                speed = activity.changeSpeed();
                                                if (!isPaused)
                                                        coveredLength += speed * (interval /
1000f);
                                                if (coveredLength > totalLength) {
                                                        i = 0;
                                                        activity.message("Congratulations, you
reached your destination");
                                                        return;
                                                }
                                                mPicture = makeContent();
                                                postInvalidate();
                                                // System.gc();
                                        }
                                } catch (InterruptedException e) {
                                        e.printStackTrace();
                                }
                        }
                };
                timer.start();
                timer = null;
        }

        public void pause() {
                isPaused = true;
        }

        public void resume() {
                isPaused = false;
```

```
}

private Picture makeContent() {

	Path mPath = new Path();
	Paint lines = new Paint();
	Paint points = new Paint();
	lines.setColor(Color.BLUE);
	lines.setStyle(Paint.Style.STROKE);
	lines.setStrokeWidth(4);
	points.setColor(Color.RED);
	points.setStyle(Paint.Style.FILL_AND_STROKE);
	points.setStrokeWidth(5);
	Picture picture = new Picture();
	Paint paint = new Paint();
	Canvas c = picture.beginRecording(WIDTH, HEIGHT);

	indexCovered = getIndexCoveredByLength(coveredLength);

	pinakas[i] = indexCovered;
	if ((i > 0) && (pinakas[i] - pinakas[i - 1] > 1)) {
		indexCovered--;
		pinakas[i] = indexCovered;
	}
	if ((indexCovered >= 1) && (pinakas[i] != pinakas[i - 1])) {
		float x2 = xs[indexCovered + 1] - xs[indexCovered];
		float y2 = ys[indexCovered + 1] - ys[indexCovered];
		float x1 = xs[indexCovered] - xs[indexCovered - 1];
		float y1 = ys[indexCovered] - ys[indexCovered - 1];
		float cross_prod = x1 * y2 - x2 * y1;
		if (Math.abs(cross_prod) > 0.01f)
			if (cross_prod > 0)
				activity.message("Turn right");
			else
				activity.message("Turn left");
	}

	i++;

	if (zs[indexCovered] == 0) {
		c.drawBitmap(image0, 0, 0, paint);
	} else if (zs[indexCovered] == 1) {
		c.drawBitmap(image1, 0, 0, paint);
	} else if (zs[indexCovered] == 2) {
		c.drawBitmap(image2, 0, 0, paint);
	} else {
		c.drawBitmap(image3, 0, 0, paint);
	}

	if ((zs[indexCovered + 1] == zs[indexCovered]) && (pass)) {
```

```
                    mPath.moveTo(xs[0], ys[0]);
                    for (int i = 1; i <= indexCovered; i++) {
                            mPath.lineTo(xs[i], ys[i]);
                    }
                    if (indexCovered < xs.length - 0.1) {
                            float l = getLengthUpToPoint(indexCovered);
                            float dl = coveredLength - l;
                            float x1 = xs[indexCovered];
                            float y1 = ys[indexCovered];
                            float x2 = xs[indexCovered + 1];
                            float y2 = ys[indexCovered + 1];
                            float d = (float) Math.sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) *
(y1 - y2));
                            float t = dl / d;
                            mPath.lineTo(x1 + t * (x2 - x1), y1 + t * (y2 - y1));
                            c.drawCircle(x1 + t * (x2 - x1), y1 + t * (y2 - y1), 3, points);
                            followX = x1 + t * (x2 - x1);
                            followY = y1 + t * (y2 - y1);
                    }
                    counter = indexCovered;

                    Paint source = new Paint();
                    source.setColor(Color.BLACK);
                    source.setTextSize(25);
                    c.drawText(name_source, xs[0] - 30, ys[0] - 10, source);
                    c.drawCircle(xs[0], ys[0], 3, points);
                    if (updown[0] != "0") {
                            Paint updownpaint = new Paint();
                            updownpaint.setColor(Color.BLACK);
                            updownpaint.setTextSize(25);
                            c.drawText(updown[0], updownX[0] - 30, updownY[0] - 10,
updownpaint);
                            c.drawCircle(updownX[0], updownY[0], 3, points);
                    }

                    if (updown[1] != "1") {
                            Paint updownpaint = new Paint();
                            updownpaint.setColor(Color.BLACK);
                            updownpaint.setTextSize(25);
                            c.drawText(updown[1], updownX[1] - 30, updownY[1] - 10,
updownpaint);
                            c.drawCircle(updownX[1], updownY[1], 3, points);
                    }

                    if (zs[indexCovered] == zs[xs.length - 1]) {
                            Paint destination = new Paint();
                            destination.setColor(Color.BLACK);
                            destination.setTextSize(25);
```

```
                        c.drawText(name_destination, xs[xs.length - 1] - 45,
ys[xs.length - 1] - 15, destination);
                        c.drawCircle(xs[xs.length - 1], ys[xs.length - 1], 3, points);
                    }

                    if ((xs[indexCovered] == 1405) && (ys[indexCovered] == 728) &&
(FirstTime)) {
                        changeFloorBefore = true;
                        FirstTime = false;
                    }

                } else {

                    if ((zs[indexCovered + 1] == zs[indexCovered]) && (passagain)) {

                        changeFloorBefore = false;

                        if ((xs[indexCovered] == 1405) && (ys[indexCovered] ==
728) && (FirstTime)) {
                            changeFloorBefore = true;
                            FirstTime = false;
                        }

                        mPath.moveTo(xs[counter], ys[counter]);
                        for (int i = counter; i <= indexCovered; i++) {
                            mPath.lineTo(xs[i], ys[i]);
                        }
                        if (indexCovered < xs.length - 0.1) {
                            float l = getLengthUpToPoint(indexCovered);
                            float dl = coveredLength - l;
                            float x1 = xs[indexCovered];
                            float y1 = ys[indexCovered];
                            float x2 = xs[indexCovered + 1];
                            float y2 = ys[indexCovered + 1];
                            float d = (float) Math.sqrt((x1 - x2) * (x1 - x2) + (y1 -
y2) * (y1 - y2));
                            float t = dl / d;
                            mPath.lineTo(x1 + t * (x2 - x1), y1 + t * (y2 - y1));
                            c.drawCircle(x1 + t * (x2 - x1), y1 + t * (y2 - y1), 3,
points);
                            followX = x1 + t * (x2 - x1);
                            followY = y1 + t * (y2 - y1);
                        }

                        if (zs[indexCovered + 1] == zs[xs.length - 1]) {
                            Paint destination = new Paint();
                            destination.setColor(Color.BLACK);
                            destination.setTextSize(25);
                            c.drawText(name_destination, xs[xs.length - 1] - 45,
ys[xs.length - 1] - 15, destination);
```

```
                                        c.drawCircle(xs[xs.length - 1], ys[xs.length - 1], 3,
points);
                                }

                                if (updown[1] != "1") {
                                        Paint updownpaint = new Paint();
                                        updownpaint.setColor(Color.BLACK);
                                        updownpaint.setTextSize(25);
                                        c.drawText(updown[1], updownX[1] - 30,
updownY[1] - 10, updownpaint);
                                        c.drawCircle(updownX[1], updownY[1], 3, points);
                                }
                                if (updown[2] != "2") {
                                        Paint updownpaint = new Paint();
                                        updownpaint.setColor(Color.BLACK);
                                        updownpaint.setTextSize(25);
                                        c.drawText(updown[2], updownX[2] - 30,
updownY[2] - 10, updownpaint);
                                        c.drawCircle(updownX[2], updownY[2], 3, points);
                                }

                                if (updown[3] != "3") {
                                        Paint updownpaint = new Paint();
                                        updownpaint.setColor(Color.BLACK);
                                        updownpaint.setTextSize(25);
                                        c.drawText(updown[3], updownX[3] - 30,
updownY[3] - 10, updownpaint);
                                        c.drawCircle(updownX[3], updownY[3], 3, points);
                                }
                                counter1 = indexCovered;

                        } else if (changefloor) {
                                changeFloorBefore = true;
                                changefloor = false;
                                pass = false;

                                mPath.moveTo(xs[0], ys[0]);
                                for (int i = 1; i <= indexCovered; i++) {
                                        mPath.lineTo(xs[i], ys[i]);
                                }

                                Paint source = new Paint();
                                source.setColor(Color.BLACK);
                                source.setTextSize(25);
                                c.drawText(name_source, xs[0] - 30, ys[0] - 10, source);
                                c.drawCircle(xs[0], ys[0], 3, points);
                                if (updown[0] != "0") {
                                        Paint updownpaint = new Paint();
                                        updownpaint.setColor(Color.BLACK);
                                        updownpaint.setTextSize(25);
```

```
                                        c.drawText(updown[0], updownX[0] - 30,
updownY[0] - 10, updownpaint);
                                        c.drawCircle(updownX[0], updownY[0], 3, points);
                                }

                                if (updown[1] != "1") {
                                        Paint updownpaint = new Paint();
                                        updownpaint.setColor(Color.BLACK);
                                        updownpaint.setTextSize(25);
                                        c.drawText(updown[1], updownX[1] - 30,
updownY[1] - 10, updownpaint);
                                        c.drawCircle(updownX[1], updownY[1], 3, points);
                                }

                                if (zs[indexCovered] == zs[xs.length - 1]) {
                                        Paint destination = new Paint();
                                        destination.setColor(Color.BLACK);
                                        destination.setTextSize(25);
                                        c.drawText(name_destination, xs[xs.length - 1] - 70,
ys[xs.length - 1] - 15, destination);
                                        c.drawCircle(xs[xs.length - 1], ys[xs.length - 1], 3,
points);
                                }

                        } else if (passfloor) {

                                counter = indexCovered;
                                counter2 = counter;
                                changeFloorBefore = false;
                                passagain = true;
                                passfloor = false;

                                mPath.moveTo(xs[counter], ys[counter]);
                                for (int i = counter; i <= indexCovered; i++) {
                                        mPath.lineTo(xs[i], ys[i]);

                                }
                                if (indexCovered < xs.length - 0.1) {
                                        float l = getLengthUpToPoint(indexCovered);
                                        float dl = coveredLength - l;
                                        float x1 = xs[indexCovered];
                                        float y1 = ys[indexCovered];
                                        float x2 = xs[indexCovered + 1];
                                        float y2 = ys[indexCovered + 1];
                                        float d = (float) Math.sqrt((x1 - x2) * (x1 - x2) + (y1 -
y2) * (y1 - y2));
                                        float t = dl / d;
                                        mPath.lineTo(x1 + t * (x2 - x1), y1 + t * (y2 - y1));
                                        c.drawCircle(x1 + t * (x2 - x1), y1 + t * (y2 - y1), 3,
points);
```

```
                followX = x1 + t * (x2 - x1);
                followY = y1 + t * (y2 - y1);
            }

            Paint source = new Paint();
            source.setColor(Color.BLACK);
            source.setTextSize(25);
            c.drawText(name_source, xs[0] - 30, ys[0] - 10, source);
            c.drawCircle(xs[0], ys[0], 3, points);

        } else if ((!changeFloorBefore) && (passagain)) {
            changeFloorBefore = true;
            passagain = false;

            mPath.moveTo(xs[counter], ys[counter]);
            for (int i = counter; i <= indexCovered; i++) {
                mPath.lineTo(xs[i], ys[i]);

            }

            Paint source = new Paint();
            source.setColor(Color.BLACK);
            source.setTextSize(25);
            c.drawText(name_source, xs[0] - 30, ys[0] - 10, source);
            c.drawCircle(xs[0], ys[0], 3, points);

            if (updown[1] != "1") {
                Paint updownpaint = new Paint();
                updownpaint.setColor(Color.BLACK);
                updownpaint.setTextSize(25);
                c.drawText(updown[1], updownX[1] - 30,
updownY[1] - 10, updownpaint);
                c.drawCircle(updownX[1], updownY[1], 3, points);
            }
            if (updown[2] != "2") {
                Paint updownpaint = new Paint();
                updownpaint.setColor(Color.BLACK);
                updownpaint.setTextSize(25);
                c.drawText(updown[2], updownX[2] - 30,
updownY[2] - 10, updownpaint);
                c.drawCircle(updownX[2], updownY[2], 3, points);
            }

            if (updown[3] != "3") {
                Paint updownpaint = new Paint();
                updownpaint.setColor(Color.BLACK);
                updownpaint.setTextSize(25);
                c.drawText(updown[3], updownX[3] - 30,
updownY[3] - 10, updownpaint);
                c.drawCircle(updownX[3], updownY[3], 3, points);
```

```
                        }

                        counter1 = indexCovered;

                    } else {

                        changeFloorBefore = false;

                        if (k == 0) {
                            changeFloorAfter = true;
                        } else {
                            changeFloorAfter = false;
                        }
                        k++;

                        mPath.moveTo(xs[counter1 + 1], ys[counter1 + 1]);
                        for (int i = counter1 + 1; i <= indexCovered; i++) {
                            mPath.lineTo(xs[i], ys[i]);
                        }
                        if (indexCovered < xs.length - 0.1) {
                            float l = getLengthUpToPoint(indexCovered);
                            float dl = coveredLength - l;
                            float x1 = xs[indexCovered];
                            float y1 = ys[indexCovered];
                            float x2 = xs[indexCovered + 1];
                            float y2 = ys[indexCovered + 1];
                            float d = (float) Math.sqrt((x1 - x2) * (x1 - x2) + (y1 -
y2) * (y1 - y2));
                            float t = dl / d;
                            mPath.lineTo(x1 + t * (x2 - x1), y1 + t * (y2 - y1));
                            c.drawCircle(x1 + t * (x2 - x1), y1 + t * (y2 - y1), 3,
points);
                            followX = x1 + t * (x2 - x1);
                            followY = y1 + t * (y2 - y1);
                        }

                        Paint destination = new Paint();
                        destination.setColor(Color.BLACK);
                        destination.setTextSize(25);
                        c.drawText(name_destination, xs[xs.length - 1] - 70,
ys[xs.length - 1] - 15, destination);
                        c.drawCircle(xs[xs.length - 1], ys[xs.length - 1], 3, points);

                        if (updown[3] != "3") {
                            Paint updownpaint = new Paint();
                            updownpaint.setColor(Color.BLACK);
                            updownpaint.setTextSize(25);
                            c.drawText(updown[3], updownX[3] - 30,
updownY[3] - 10, updownpaint);
                            c.drawCircle(updownX[3], updownY[3], 3, points);
```

```
                        }
                    }
                }
                c.drawPath(mPath, lines);
                picture.endRecording();
                if (number < 3) {
                    System.gc();
                    number++;
                }
                return picture;
        }
}
```

Floors.java

```
package com.ece;

import android.app.ListActivity;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.CheckedTextView;
import android.widget.ListView;

import com.data.MapIntent;

public class Floors extends ListActivity {

        String classes[] = { "0 Floor - Ground", "1st Floor", "2nd Floor", "3rd Floor" };

        int selected;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                // TODO Auto-generated method stub
                super.onCreate(savedInstanceState);
                Bundle extras = this.getIntent().getExtras();
                selected = extras.getInt("Floor1");

                ArrayAdapter<String> adapter = new ArrayAdapter<String>(Floors.this,
android.R.layout.simple_list_item_checked, classes) {
                        @Override
                        public View getView(int position, View convertView, ViewGroup
parent) {
                                CheckedTextView v = (CheckedTextView)
super.getView(position, convertView, parent);
                                v.setChecked(position == selected);
```

```
                        return v;
                    }
                };
                setListAdapter(adapter);
        }

        @Override
        protected void onListItemClick(ListView l, View v, int position, long id) {
                // TODO Auto-generated method stub
                super.onListItemClick(l, v, position, id);

                if (position == 0) {
                        startActivity(MapIntent.openFloor(this.getApplicationContext(), 0));

                } else if (position == 1) {
                        startActivity(MapIntent.openFloor(this.getApplicationContext(), 1));

                } else if (position == 2) {
                        startActivity(MapIntent.openFloor(this.getApplicationContext(), 2));

                } else {
                        startActivity(MapIntent.openFloor(this.getApplicationContext(), 3));
                }

        }

        public void onBackPressed() {
                startActivity(MapIntent.openFloor(this.getApplicationContext(), selected));
                finish();
                return;
        }
}
```

Prefs.java

```
package com.ece;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.SharedPreferences;
import android.content.SharedPreferences.OnSharedPreferenceChangeListener;
import android.os.Bundle;
import android.preference.EditTextPreference;
import android.preference.ListPreference;
import android.preference.Preference;
import android.preference.PreferenceActivity;
import android.preference.PreferenceCategory;
import android.preference.PreferenceManager;
```

```
import android.preference.Preference.OnPreferenceClickListener;
import android.view.LayoutInflater;

public class Prefs extends PreferenceActivity implements
OnSharedPreferenceChangeListener {

	int i = 0;
	int floor;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		Bundle extras = this.getIntent().getExtras();
		floor = extras.getInt("Floor1");
		addPreferencesFromResource(R.xml.prefs);
		// Get the custom preference
		Preference info = (Preference) findPreference("info");
		Preference about = (Preference) findPreference("about");

		info.setOnPreferenceClickListener(new OnPreferenceClickListener() {

			public boolean onPreferenceClick(Preference preference) {
				i = 1;
				InstructionsDialog();
				return true;
			}

		});

		about.setOnPreferenceClickListener(new OnPreferenceClickListener() {

			public boolean onPreferenceClick(Preference preference) {
				i = 2;
				InstructionsDialog();
				return true;
			}

		});

		PreferenceManager.setDefaultValues(Prefs.this, R.xml.prefs, false);

		for (int i = 0; i < getPreferenceScreen().getPreferenceCount(); i++) {
			initSummary(getPreferenceScreen().getPreference(i));
		}
	}

	@Override
	protected void onResume() {
		super.onResume();
```

```
		// Set up a listener whenever a key changes

	getPreferenceScreen().getSharedPreferences().registerOnSharedPreferenceChangeLi
stener(this);
	}

	@Override
	protected void onPause() {
		super.onPause();
		// Unregister the listener whenever a key changes

	getPreferenceScreen().getSharedPreferences().unregisterOnSharedPreferenceChange
Listener(this);
	}

	public void onSharedPreferenceChanged(SharedPreferences sharedPreferences,
String key) {
		updatePrefSummary(findPreference(key));
	}

	private void initSummary(Preference p) {
		if (p instanceof PreferenceCategory) {
			PreferenceCategory pCat = (PreferenceCategory) p;
			for (int i = 0; i < pCat.getPreferenceCount(); i++) {
				initSummary(pCat.getPreference(i));
			}
		} else {
			updatePrefSummary(p);
		}

	}

	private void updatePrefSummary(Preference p) {
		if (p instanceof ListPreference) {
			ListPreference listPref = (ListPreference) p;
			p.setSummary(listPref.getEntry());
		}
		if (p instanceof EditTextPreference) {
			EditTextPreference editTextPref = (EditTextPreference) p;
			p.setSummary(editTextPref.getText());
		}

	}

	public void InstructionsDialog() {
		AlertDialog.Builder ad = new AlertDialog.Builder(this);
		ad.setIcon(R.drawable.icon);
		if (i == 1) {
			ad.setTitle("Info");
			ad.setView(LayoutInflater.from(this).inflate(R.layout.info, null));
```

```
		} else {
			ad.setTitle("About");
			ad.setView(LayoutInflater.from(this).inflate(R.layout.about, null));
		}

		ad.setPositiveButton("OK", new
android.content.DialogInterface.OnClickListener() {
			public void onClick(DialogInterface dialog, int arg1) {
				// OK, go back to Main menu
			}
		});

		ad.setOnCancelListener(new DialogInterface.OnCancelListener() {
			public void onCancel(DialogInterface dialog) {
				// OK, go back to Main menu
			}
		});

		ad.show();
	}

}
```

VolumeBar.java

```
package com.ece;

import android.media.AudioManager;
import android.speech.tts.TextToSpeech;
import android.view.MotionEvent;
import android.widget.SeekBar;

/**
 *An object for changing the phone's volume This object adds text to speech to
 * test the new volume
 */
public class VolumeBar extends SeekBar {

	private TextToSpeech tts;
	AudioManager am;

	public VolumeBar(MapActivity mapActivity, TextToSpeech tts, AudioManager am)
{
		super(mapActivity);
		this.tts = tts;
		this.am = am;

	}
```

```
        public boolean onTouchEvent(MotionEvent event) {
                if (event.getAction() == MotionEvent.ACTION_UP) {
                        double curVol = getProgress();
                        double maxVol = getMax();
                        am.setStreamVolume(AudioManager.STREAM_MUSIC, (int)
(curVol / maxVol * am.getStreamMaxVolume(AudioManager.STREAM_MUSIC)), 0);
                        tts.speak("", TextToSpeech.QUEUE_FLUSH, null);
                        tts.speak("sound", TextToSpeech.QUEUE_FLUSH, null);
                }
                return super.onTouchEvent(event);
        }
}
```

Poi.java

```
package com.ece;

import android.app.ListActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

import com.data.MapIntent;

public class Poi extends ListActivity {

        String classes[] = { "Professors' Offices", "Amfitheatre", "Classrooms",
"Secretariat", "Library", "Labs", "Cafeteria", "WC", "Elevators", "Stairs",
                        "Exit", "Safety Exit" };

        int floor;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setListAdapter(new ArrayAdapter<String>(Poi.this,
android.R.layout.simple_list_item_1, classes));
                Bundle extras = this.getIntent().getExtras();
                if (extras != null) {
                        floor = extras.getInt("Floor1");
                } else
                        floor = 0;
        }
```

```
	@Override
	protected void onListItemClick(ListView l, View v, int position, long id) {
		super.onListItemClick(l, v, position, id);

		if (position == 0) {
			startActivity(new Intent("com.search.PROFESSORLIST"));

		} else if (position == 1) {
			startActivity(new Intent("com.databaselist.AMFITHEATRELIST"));

		} else if (position == 2) {
			startActivity(new Intent("com.databaselist.CLASSROOMSLIST"));

		} else if (position == 3) {
			startActivity(new Intent("com.search.SECRETARIATLIST"));

		} else if (position == 4) {
			startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
400, 737, 0, "Library"));

		} else if (position == 5) {
			startActivity(new Intent("com.databaselist.LABSLIST"));

		} else if (position == 6) {
			startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
1495, 735, 0, "Cafeteria"));

		} else if (position == 7) {
			startActivity(new Intent("com.databaselist.WCLIST"));

		} else if (position == 8) {
			startActivity(new Intent("com.databaselist.ELEVATORLIST"));

		} else if (position == 9) {
			startActivity(new Intent("com.databaselist.STAIRSLIST"));

		} else if (position == 10) {

			Toast toast = Toast.makeText(this, "All Exits are on the Floor 0 -
Ground", 5000);
			toast.setGravity(Gravity.BOTTOM, 0, 0);
			toast.show();
			startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
563, 745, 0, "Exit"));

		} else {
			startActivity(new Intent("com.databaselist.SAFETYEXITLIST"));
		}
	}
```

```
	public void onBackPressed() {
		startActivity(MapIntent.openFloor(this.getApplicationContext(), floor));
		finish();
		return;
	}
}
```

DatabaseHelper.java

```
package com.dijkstra;

import java.io.InputStream;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.Document;
import org.w3c.dom.NodeList;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.widget.Toast;

import com.ece.R;

public class DatabaseHelper extends SQLiteOpenHelper {

	public static final String DATABASE_NAME = "Allinfo";
	protected Context context;

	public DatabaseHelper(Context context) {
		super(context, DATABASE_NAME, null, 1);
		this.context = context;
	}

	@Override
	public void onCreate(SQLiteDatabase db) {
		String s;
		try {
			Toast.makeText(context, "1", 2000).show();
			InputStream in =
context.getResources().openRawResource(R.raw.sqlall);
			DocumentBuilder builder =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
			Document doc = builder.parse(in, null);
			NodeList statements = doc.getElementsByTagName("statement");
```

```
                        for (int i = 0; i < statements.getLength(); i++) {
                                s =
statements.item(i).getChildNodes().item(0).getNodeValue();
                                db.execSQL(s);
                        }
                } catch (Throwable t) {
                        Toast.makeText(context, t.toString(), 50000).show();
                }
        }

        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
                db.execSQL("DROP TABLE IF EXISTS info");
                db.execSQL("DROP TABLE IF EXISTS path");
                db.execSQL("DROP TABLE IF EXISTS nodes");
                db.execSQL("DROP TABLE IF EXISTS SecretariatInfo");
                onCreate(db);
        }

}
```

DijkstraAlgorithm.java

```
package com.dijkstra;

import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.Map;
import java.util.PriorityQueue;

public class DijkstraAlgorithm {

        public Map<String, Vertex> vertices = new HashMap<String, Vertex>();

        public void computePaths(Vertex source) {
                source.minDistance = 0.;
                PriorityQueue<Vertex> vertexQueue = new PriorityQueue<Vertex>();
                vertexQueue.add(source);

                while (!vertexQueue.isEmpty()) {
                        Vertex u = vertexQueue.poll();

                        // Visit each edge exiting u
                        for (Edge e : u.adjacencies) {
                                Vertex v = e.getTarget();
                                double weight = e.getWeight();
                                double distanceThroughU = u.minDistance + weight;
```

```
                        if (distanceThroughU < v.minDistance) {
                            vertexQueue.remove(v);
                            v.minDistance = distanceThroughU;
                            v.previous = u;
                            vertexQueue.add(v);
                        }
                    }
                }
            }

            public ArrayList<Vertex> getShortestPathTo(Vertex target) {
                ArrayList<Vertex> path = new ArrayList<Vertex>();
                for (Vertex vertex = target; vertex != null; vertex = vertex.previous)
                    path.add(vertex);

                Collections.reverse(path);
                return path;
            }

}
```

Edge.java

```
package com.dijkstra;

public class Edge {
        private Vertex target;;
        private double weight;

        public Edge(Vertex target, double weight) {
            this.setTarget(target);
            this.setWeight(weight);
        }

        public void setTarget(Vertex target) {
            this.target = target;
        }

        public Vertex getTarget() {
            return target;
        }

        public void setWeight(double weight) {
            this.weight = weight;
        }

        public double getWeight() {
```

```
		return weight;
	}
}
```

Vertex.java

```
package com.dijkstra;

import java.util.ArrayList;
import java.util.List;

public class Vertex implements Comparable<Vertex> {
	public final String name;
	// public Edge[] adjacencies;
	public List<Edge> adjacencies = new ArrayList<Edge>();
	public double minDistance = Double.POSITIVE_INFINITY;
	public Vertex previous;
	public int x, y, z;

	public Vertex(String argName) {
		name = argName;
	}

	public String toString() {
		return name;
	}

	public int compareTo(Vertex other) {
		return Double.compare(minDistance, other.minDistance);
	}

}
```

MapIntent.java

```
package com.data;

import android.content.Context;
import android.content.Intent;

public class MapIntent extends Intent {
```

```
	private static Class<?> clazz;
	static {
		try {
			clazz = Class.forName("com.ece.MapActivity");
		} catch (ClassNotFoundException e) {
			e.printStackTrace();
		}
	}

	private MapIntent(Context context) {
		super(context, clazz);
	}

	public static MapIntent openFloor(Context context, int floor) {
		MapIntent mapIntent = new MapIntent(context);
		mapIntent.putExtra("action", "open floor");
		mapIntent.putExtra("floor", floor);
		mapIntent.putExtra("view", true);
		return mapIntent;
	}

	public static MapIntent DrawPoints(Context context, int x, int y, int z, String s) {
		MapIntent mapIntent = new MapIntent(context);
		mapIntent.putExtra("ok", "show position");
		mapIntent.putExtra("x", x);
		mapIntent.putExtra("y", y);
		mapIntent.putExtra("z", z);
		mapIntent.putExtra("name", s);
		mapIntent.putExtra("view", true);
		return mapIntent;
	}

	public static MapIntent SrcDest(Context context, String source, int x1, int y1, int z1,
String destination, int x2, int y2, int z2, int[] xs, int[] ys,
			int[] zs, String name_source, String name_destination) {
		MapIntent mapIntent = new MapIntent(context);
		mapIntent.putExtra("points", "show position");
		mapIntent.putExtra("source", source);
		mapIntent.putExtra("x1", x1);
		mapIntent.putExtra("y1", y1);
		mapIntent.putExtra("z1", z1);
		mapIntent.putExtra("destination", destination);
		mapIntent.putExtra("x2", x2);
		mapIntent.putExtra("y2", y2);
		mapIntent.putExtra("z2", z2);
		mapIntent.putExtra("xs", xs);
		mapIntent.putExtra("ys", ys);
		mapIntent.putExtra("zs", zs);
		mapIntent.putExtra("name_source", name_source);
		mapIntent.putExtra("name_destination", name_destination);
```

```
                mapIntent.putExtra("view", false);
                return mapIntent;
        }
}
```

PassData.java

```
package com.data;

public class PassData {

        private int data;

        public PassData(int data) {
                super();
                this.data = data;
        }

        public int getdata() {
                return data;
        }

        public void setdata(int data) {
                this.data = data;
        }
}
```

PassValues.java

```
package com.data;

public class PassValues {

        private String label;

        private String data;

        private int type;

        public static final int ACTION_CALL = 1;
        public static final int ACTION_EMAIL = 2;
        public static final int ACTION_MAP = 3;

        public PassValues(String label, String data, int type) {
                super();
```

```
		this.label = label;
		this.data = data;
		this.type = type;
	}

	public String getLabel() {
		return label;
	}

	public void setLabel(String label) {
		this.label = label;
	}

	public String getData() {
		return data;
	}

	public void setData(String data) {
		this.data = data;
	}

	public int getType() {
		return type;
	}

	public void setType(int type) {
		this.type = type;
	}

}
```

PointsFloor.java

```
package com.data;

public class PointsFloor {

	private int coX;

	private int coY;

	private int coZ;

	private String s;

	public PointsFloor(int coX, int coY, int coZ, String s) {
		super();
```

```
		this.coX = coX;
		this.coY = coY;
		this.coZ = coZ;
		this.s = s;
	}

	public int getcoX() {
		return coX;
	}

	public void setcoX(int coX) {
		this.coX = coX;
	}

	public int getcoY() {
		return coY;
	}

	public void setcoY(int coY) {
		this.coY = coY;
	}

	public int getcoZ() {
		return coZ;
	}

	public void setcoZ(int coZ) {
		this.coZ = coZ;
	}

	public String getS() {
		return s;
	}

	public void setS(String s) {
		this.s = s;
	}
}
```

SourceList.java

```
package com.search;

import java.util.Locale;

import android.app.ListActivity;
import android.content.Intent;
import android.content.SharedPreferences;
```

```
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.speech.tts.TextToSpeech;
import android.speech.tts.TextToSpeech.OnInitListener;
import android.view.Gravity;
import android.view.View;
import android.widget.EditText;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;
import android.widget.Toast;

import com.data.MapIntent;
import com.dijkstra.DatabaseHelper;
import com.ece.R;

public class SourceList extends ListActivity implements OnInitListener {

	protected EditText searchText1;
	protected EditText searchText2;
	protected SQLiteDatabase db;
	protected Cursor cursor1;
	protected Cursor cursor2;
	TextToSpeech tts;
	String name_source;
	protected ListAdapter adapter1;
	protected ListAdapter adapter2;
	int floor;

	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main1);
		Bundle extras = this.getIntent().getExtras();
		if (extras != null) {
			floor = extras.getInt("Floor1");
		} else
			floor = 0;
		searchText1 = (EditText) findViewById(R.id.searchText1);
		db = (new DatabaseHelper(this)).getWritableDatabase();
	}

	public void onListItemClick(ListView parent, View view, int position, long id) {
		Intent intent = new Intent(this, DestinationList.class);
		Cursor cursor1 = (Cursor) adapter1.getItem(position);
		intent.putExtra("SOURCE_ID",
cursor1.getInt(cursor1.getColumnIndex("_id")));
		intent.putExtra("source",
cursor1.getString(cursor1.getColumnIndex("name")));
```

```
                intent.putExtra("name_source",
cursor1.getString(cursor1.getColumnIndex("node_name")));
                intent.putExtra("x1", cursor1.getInt(cursor1.getColumnIndex("pox")));
                intent.putExtra("y1", cursor1.getInt(cursor1.getColumnIndex("poy")));
                intent.putExtra("z1", cursor1.getInt(cursor1.getColumnIndex("poz")));
                name_source = cursor1.getString(cursor1.getColumnIndex("node_name"));
                tts = new TextToSpeech(this, this);
                Toast toast = Toast.makeText(this, "You are in the " + name_source + ".
Where do you want to go?", 6000);
                toast.setGravity(Gravity.BOTTOM, 0, 0);
                toast.show();
                startActivity(intent);

        }

        public void search(View view) {
                // || is the concatenation operation in SQLite
                cursor1 = db.rawQuery("SELECT _id, name, node_name, numFloor, pox,
poy, poz FROM nodes WHERE node_name LIKE ?", new String[] { "%"
                                + searchText1.getText().toString() + "%" });
                startManagingCursor(cursor1);
                adapter1 = new SimpleCursorAdapter(this, R.layout.nodeslist, cursor1, new
String[] { "node_name", "numFloor" }, new int[] { R.id.name, R.id.floorZ });
                setListAdapter(adapter1);

        }

        @Override
        public void onInit(int status) {
                tts.setLanguage(Locale.US);
                SharedPreferences getPrefs =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
                boolean voice = getPrefs.getBoolean("checkboxVoice", true);
                if (voice == true) {
                        tts.speak("You are in the " + name_source + ". Where do you want to
go?", TextToSpeech.QUEUE_FLUSH, null);
                }
        }

        @Override
        protected void onStop() {
                db.close();
                super.onStop();
        }

        public void onBackPressed() {
                startActivity(MapIntent.openFloor(this.getApplicationContext(), floor));
                finish();
                return;
        }
```

}

DestinationList.java

```
package com.search;

import java.util.List;
import java.util.Locale;

import android.app.ListActivity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.speech.tts.TextToSpeech;
import android.speech.tts.TextToSpeech.OnInitListener;
import android.view.Gravity;
import android.view.View;
import android.widget.EditText;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;
import android.widget.Toast;

import com.data.MapIntent;
import com.dijkstra.DatabaseHelper;
import com.dijkstra.DijkstraAlgorithm;
import com.dijkstra.Edge;
import com.dijkstra.Vertex;
import com.ece.R;

public class DestinationList extends ListActivity implements OnInitListener {

	protected EditText searchText2;
	protected SQLiteDatabase nodes_db, edges_db;
	protected Cursor cursor2;
	protected ListAdapter adapter2;
	String source, name_source;
	int id1, x1, y1, z1;
	boolean ok = true;
	TextToSpeech tts;
	String name_destination;

	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
```

```
		Bundle extras = this.getIntent().getExtras();
		source = extras.getString("source");
		id1 = extras.getInt("SOURCE_ID");
		name_source = extras.getString("name_source");
		x1 = extras.getInt("x1");
		y1 = extras.getInt("y1");
		z1 = extras.getInt("z1");
		setContentView(R.layout.main2);
		searchText2 = (EditText) findViewById(R.id.searchText2);
		nodes_db = (new DatabaseHelper(this)).getWritableDatabase();
		edges_db = (new DatabaseHelper(this)).getWritableDatabase();
	}

	public void onListItemClick(ListView parent, View view, int position, long id) {

		Cursor CursorIntent = (Cursor) adapter2.getItem(position);
		String destination =
CursorIntent.getString(CursorIntent.getColumnIndex("name"));
		name_destination =
CursorIntent.getString(CursorIntent.getColumnIndex("node_name"));
		int x2 = CursorIntent.getInt(CursorIntent.getColumnIndex("pox"));
		int y2 = CursorIntent.getInt(CursorIntent.getColumnIndex("poy"));
		int z2 = CursorIntent.getInt(CursorIntent.getColumnIndex("poz"));

		ok = true;
		if ((x1 == x2) & (y1 == y2) & (z1 == z2)) {
			Toast toast = Toast.makeText(this, "You chose the same source and
destination. Choose again. Where do you want to go?", 6000);
			toast.setGravity(Gravity.BOTTOM, 0, 0);
			toast.show();
			ok = false;
		}
		tts = new TextToSpeech(this, this);

		// Dijkstra
		DijkstraAlgorithm algo = new DijkstraAlgorithm();

		Cursor nodes = nodes_db.rawQuery("SELECT * FROM nodes", null);
		startManagingCursor(nodes);
		nodes.moveToFirst();
		while (!nodes.isAfterLast()) {
			Vertex vertex = new
Vertex(nodes.getString(nodes.getColumnIndex("name")));
			vertex.x = nodes.getInt(nodes.getColumnIndex("pox"));
			vertex.y = nodes.getInt(nodes.getColumnIndex("poy"));
			vertex.z = nodes.getInt(nodes.getColumnIndex("poz"));
			algo.vertices.put(nodes.getString(nodes.getColumnIndex("name")),
vertex);
			nodes.moveToNext();
		}
```

```
                Cursor edges = edges_db.rawQuery("SELECT * FROM path", null);
                startManagingCursor(edges);
                edges.moveToFirst();
                while (!edges.isAfterLast()) {
                        String fromStr =
edges.getString(edges.getColumnIndex("fromNode"));
                        String toStr = edges.getString(edges.getColumnIndex("toNode"));
                        int cost = edges.getInt(edges.getColumnIndex("cost"));
                        Vertex from = algo.vertices.get(fromStr);
                        Vertex to = algo.vertices.get(toStr);
                        from.adjacencies.add(new Edge(to, cost));
                        edges.moveToNext();
                }

                algo.computePaths(algo.vertices.get(source));
                List<Vertex> sortestPath =
algo.getShortestPathTo(algo.vertices.get(destination));
                int[] xs = new int[sortestPath.size()];
                int[] ys = new int[sortestPath.size()];
                int[] zs = new int[sortestPath.size()];
                int i = 0;
                for (Vertex v : sortestPath) {
                        xs[i] = v.x;
                        ys[i] = v.y;
                        zs[i] = v.z;
                        i++;
                }
                if (ok == true) {
                        Toast toast = Toast.makeText(this, "You chose " + name_destination
+ ".", 6000);
                        toast.setGravity(Gravity.BOTTOM, 0, 0);
                        toast.show();
                        startActivity(MapIntent.SrcDest(this.getApplicationContext(), source,
x1, y1, z1, destination, x2, y2, z2, xs, ys, zs, name_source,
                                        name_destination));
                }
        }

        public void search(View view) {
                // || is the concatenation operation in SQLite
                cursor2 = nodes_db.rawQuery("SELECT _id, name, node_name, numFloor,
pox, poy, poz FROM nodes WHERE node_name LIKE ?", new String[] { "%"
                                + searchText2.getText().toString() + "%" });
                startManagingCursor(cursor2);
                adapter2 = new SimpleCursorAdapter(this, R.layout.nodeslist, cursor2, new
String[] { "node_name", "numFloor" }, new int[] { R.id.name, R.id.floorZ });
                setListAdapter(adapter2);
        }
```

```
	@Override
	public void onInit(int status) {
		tts.setLanguage(Locale.US);
		SharedPreferences getPrefs =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
		boolean voice = getPrefs.getBoolean("checkboxVoice", true);
		if (voice == true) {
			if (ok == true) {
				tts.speak("You chose " + name_destination,
TextToSpeech.QUEUE_FLUSH, null);
			} else
				tts.speak("You chose the same source and destination. Choose
again. Where do you want to go?", TextToSpeech.QUEUE_FLUSH, null);
		}
	}

	@Override
	protected void onStop() {
		nodes_db.close();
		edges_db.close();
		super.onStop();
	}

	public void onBackPressed() {
		startActivity(new Intent("com.search.SOURCELIST"));
		finish();
		System.exit(0);
		return;
	}

}
```

ProfessorList.java

```
package com.search;

import android.app.ListActivity;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;

import com.dijkstra.DatabaseHelper;
```

```
import com.ece.R;

public class ProfessorList extends ListActivity {

	protected EditText searchText;
	protected SQLiteDatabase db;
	protected Cursor cursor;
	protected ListAdapter adapter;

	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main_prof_search);
		searchText = (EditText) findViewById(R.id.searchText);
		db = (new DatabaseHelper(this)).getWritableDatabase();
	}

	public void onListItemClick(ListView parent, View view, int position, long id) {
		Intent intent = new Intent(this, ProfessorDetails.class);
		Cursor cursor = (Cursor) adapter.getItem(position);
		intent.putExtra("INFO_ID", cursor.getInt(cursor.getColumnIndex("_id")));
		startActivity(intent);
	}

	public void search(View view) {
		// || is the concatenation operation in SQLite
		cursor = db.rawQuery("SELECT _id, firstName, lastName, floor FROM info
WHERE firstName || ' ' || lastName  || ' ' || floor LIKE ?", new String[] { "%"
				+ searchText.getText().toString() + "%" });
		startManagingCursor(cursor);
		adapter = new SimpleCursorAdapter(this, R.layout.prof_list, cursor, new
String[] { "firstName", "lastName", "floor" }, new int[] { R.id.firstName,
				R.id.lastName, R.id.floorN });
		setListAdapter(adapter);
	}

	public void onBackPressed() {
		startActivity(new Intent("com.ece.POI"));
		finish();
		return;
	}

	@Override
	protected void onStop() {

		db.close();
		super.onStop();
	}

}
```

ProfessorDetails.java

```
package com.search;

import java.util.ArrayList;
import java.util.List;

import android.app.ListActivity;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;

import com.data.MapIntent;
import com.data.PassValues;
import com.data.PointsFloor;
import com.dijkstra.DatabaseHelper;
import com.ece.R;

public class ProfessorDetails extends ListActivity {

	protected TextView profNameText;
	protected TextView titleText;
	protected List<PassValues> actions;
	protected List<PointsFloor> positions;
	protected PassValuesAdapter adapter;
	protected int profId;
	protected int positionX;
	protected int positionY;
	protected int positionZ;
	protected Intent i = null;
	protected String name;
	SQLiteDatabase db;

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.prof_details);
		profId = getIntent().getIntExtra("INFO_ID", 0);
```

```
		db = (new DatabaseHelper(this)).getWritableDatabase();
		Cursor cursor = db.rawQuery("SELECT _id, firstName, lastName, floor, z,
x, y, phone, email FROM info  WHERE _id = ?", new String[] { "" + profId });
		startManagingCursor(cursor);
		if (cursor.getCount() == 1) {
			cursor.moveToFirst();

			profNameText = (TextView) findViewById(R.id.profName);

	profNameText.setText(cursor.getString(cursor.getColumnIndex("firstName")) + " "
+ cursor.getString(cursor.getColumnIndex("lastName")));

			titleText = (TextView) findViewById(R.id.floorNum);
			titleText.setText(cursor.getString(cursor.getColumnIndex("floor")));

			actions = new ArrayList<PassValues>();
			positions = new ArrayList<PointsFloor>();

			String officePhone =
cursor.getString(cursor.getColumnIndex("phone"));
			if (officePhone != null) {
				actions.add(new PassValues("Call office", officePhone,
PassValues.ACTION_CALL));
			}

			String email = cursor.getString(cursor.getColumnIndex("email"));
			if (email != null) {
				actions.add(new PassValues("Email", email,
PassValues.ACTION_EMAIL));
			}

			positionX = cursor.getInt(cursor.getColumnIndex("x"));
			positionY = cursor.getInt(cursor.getColumnIndex("y"));
			positionZ = cursor.getInt(cursor.getColumnIndex("z"));
			actions.add(new PassValues("Find Position on the Map", null,
PassValues.ACTION_MAP));

			name = cursor.getString(cursor.getColumnIndex("firstName")) + " "
+ cursor.getString(cursor.getColumnIndex("lastName"));
		}

		adapter = new PassValuesAdapter();
		setListAdapter(adapter);
	}

	public void onListItemClick(ListView parent, View view, int position, long id) {

		PassValues action = actions.get(position);

		Intent intent;
```

```
                switch (action.getType()) {

                case PassValues.ACTION_CALL:
                        Uri callUri = Uri.parse("tel:" + action.getData());
                        intent = new Intent(Intent.ACTION_CALL, callUri);
                        startActivity(intent);
                        break;

                case PassValues.ACTION_EMAIL:
                        intent = new Intent(android.content.Intent.ACTION_SEND);
                        intent.setType("plain/text");
                        intent.putExtra(android.content.Intent.EXTRA_EMAIL, new String[]
{ action.getData() });
                        startActivity(intent);
                        break;

                case PassValues.ACTION_MAP:

                        startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
positionX, positionY, positionZ, name));
                        break;

                }
        }

        class PassValuesAdapter extends ArrayAdapter<PassValues> {

                PassValuesAdapter() {
                        super(ProfessorDetails.this, R.layout.passvalues_list, actions);
                }

                @Override
                public View getView(int position, View convertView, ViewGroup parent) {
                        PassValues action = actions.get(position);
                        LayoutInflater inflater = getLayoutInflater();
                        View view = inflater.inflate(R.layout.passvalues_list, parent, false);
                        TextView label = (TextView) view.findViewById(R.id.label);
                        label.setText(action.getLabel());
                        TextView data = (TextView) view.findViewById(R.id.data);
                        data.setText(action.getData());
                        return view;
                }
        }

        public void onBackPressed() {
                startActivity(new Intent("com.search.PROFESSORLIST"));
                finish();
                return;
        }
```

}

SecretariatList.java

```
package com.search;

import android.app.ListActivity;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;

import com.dijkstra.DatabaseHelper;
import com.ece.R;

public class SecretariatList extends ListActivity {

	protected EditText searchText, test;
	protected SQLiteDatabase db;
	protected Cursor cursor, cursor1;
	protected ListAdapter adapter;
	String[] result;
	boolean isSearched = false;
	int[] id = new int[15];

	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main_secretariat_search);
		searchText = (EditText) findViewById(R.id.searchText);
		db = (new DatabaseHelper(this)).getWritableDatabase();
		getData();
		setListAdapter(new ArrayAdapter<String>(this,
android.R.layout.test_list_item, result));
	}

	public void onListItemClick(ListView parent, View view, int position, long id1) {
		Intent intent = new Intent(this, SecretariatDetails.class);
		if (isSearched) {
			Cursor cursor = (Cursor) adapter.getItem(position);
```

```
                        intent.putExtra("INFO1_ID",
cursor.getInt(cursor.getColumnIndex("_id")));
                } else {
                        intent.putExtra("INFO1_ID", id[position]);
                }
                startActivity(intent);
        }

        public void search(View view) {
                // || is the concatenation operation in SQLite
                isSearched = true;
                cursor = db.rawQuery("SELECT _id, name, floor FROM SecretariatInfo
WHERE name || ' ' || floor LIKE ?", new String[] { "%"
                                + searchText.getText().toString() + "%" });
                startManagingCursor(cursor);
                adapter = new SimpleCursorAdapter(this, R.layout.prof_list, cursor, new
String[] { "name", "floor" }, new int[] { R.id.firstName, R.id.floorN });
                setListAdapter(adapter);
        }

        public String[] getData() {
                // TODO Auto-generated method stub
                String[] columns = new String[] { "_id", "name", "floor" };
                Cursor c = db.query("SecretariatInfo", columns, null, null, null, null, null);

                // Initialise the result variable
                startManagingCursor(c);
                result = new String[c.getCount()];

                int name = c.getColumnIndex("name");
                int floor = c.getColumnIndex("floor");
                int counter = 0;
                for (c.moveToFirst(); !c.isAfterLast(); c.moveToNext()) {
                        result[counter] = c.getString(name) + "\n" + c.getString(floor) + "\n";
                        id[counter] = c.getInt(c.getColumnIndex("_id"));
                        counter++;
                }
                return result;
        }

        @Override
        protected void onStop() {

                db.close();
                super.onStop();
        }
}
```

SecretariatDetails.java

```
package com.search;

import java.util.ArrayList;
import java.util.List;

import android.app.ListActivity;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;

import com.data.MapIntent;
import com.data.PassValues;
import com.data.PointsFloor;
import com.dijkstra.DatabaseHelper;
import com.ece.R;

public class SecretariatDetails extends ListActivity {

	protected TextView profNameText;
	protected TextView titleText;
	protected List<PassValues> actions;
	protected List<PointsFloor> positions;
	protected PassValuesAdapter adapter;
	protected int profId;
	protected int positionX;
	protected int positionY;
	protected int positionZ;
	protected Intent i = null;
	protected String name;
	SQLiteDatabase db;

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.prof_details);

		profId = getIntent().getIntExtra("INFO1_ID", 0);
		db = (new DatabaseHelper(this)).getWritableDatabase();
```

```
		Cursor cursor = db.rawQuery("SELECT _id, name, floor, z, x, y, phone1,
phone2, fax, email FROM SecretariatInfo  WHERE _id = ?", new String[] { ""
				+ profId });
		startManagingCursor(cursor);
		if (cursor.getCount() == 1) {
			cursor.moveToFirst();

			profNameText = (TextView) findViewById(R.id.profName);

	profNameText.setText(cursor.getString(cursor.getColumnIndex("name")));

			titleText = (TextView) findViewById(R.id.floorNum);
			titleText.setText(cursor.getString(cursor.getColumnIndex("floor")));

			actions = new ArrayList<PassValues>();
			positions = new ArrayList<PointsFloor>();

			String officePhone1 =
cursor.getString(cursor.getColumnIndex("phone1"));
			if (officePhone1 != null) {
				actions.add(new PassValues("Call office 1", officePhone1,
PassValues.ACTION_CALL));
			}

			String officePhone2 =
cursor.getString(cursor.getColumnIndex("phone2"));
			if (officePhone2 != null) {
				actions.add(new PassValues("Call office 2", officePhone2,
PassValues.ACTION_CALL));
			}

			String fax = cursor.getString(cursor.getColumnIndex("fax"));
			if (fax != null) {
				actions.add(new PassValues("Fax", officePhone2,
PassValues.ACTION_CALL));
			}

			String email = cursor.getString(cursor.getColumnIndex("email"));
			if (email != null) {
				actions.add(new PassValues("Email", email,
PassValues.ACTION_EMAIL));
			}

			positionX = cursor.getInt(cursor.getColumnIndex("x"));
			positionY = cursor.getInt(cursor.getColumnIndex("y"));
			positionZ = cursor.getInt(cursor.getColumnIndex("z"));
			actions.add(new PassValues("Find Position on the Map", null,
PassValues.ACTION_MAP));

			name = cursor.getString(cursor.getColumnIndex("name"));
```

```
		}

		adapter = new PassValuesAdapter();
		setListAdapter(adapter);
	}

	public void onListItemClick(ListView parent, View view, int position, long id) {

		PassValues action = actions.get(position);

		Intent intent;

		switch (action.getType()) {

		case PassValues.ACTION_CALL:
			Uri callUri = Uri.parse("tel:" + action.getData());
			intent = new Intent(Intent.ACTION_CALL, callUri);
			startActivity(intent);
			break;

		case PassValues.ACTION_EMAIL:
			intent = new Intent(android.content.Intent.ACTION_SEND);
			intent.setType("plain/text");
			intent.putExtra(android.content.Intent.EXTRA_EMAIL, new String[]
{ action.getData() });
			startActivity(intent);
			break;

		case PassValues.ACTION_MAP:

			startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
positionX, positionY, positionZ, name));
			break;

		}
	}

	class PassValuesAdapter extends ArrayAdapter<PassValues> {

		PassValuesAdapter() {
			super(SecretariatDetails.this, R.layout.passvalues_list, actions);
		}

		@Override
		public View getView(int position, View convertView, ViewGroup parent) {
			PassValues action = actions.get(position);
			LayoutInflater inflater = getLayoutInflater();
			View view = inflater.inflate(R.layout.passvalues_list, parent, false);
			TextView label = (TextView) view.findViewById(R.id.label);
			label.setText(action.getLabel());
```

```
                TextView data = (TextView) view.findViewById(R.id.data);
                data.setText(action.getData());
                return view;
            }

        }

        public void onBackPressed() {
            startActivity(new Intent("com.ece.POI"));
            finish();
            return;
        }

}
```

AmfitheatreList.java

```
package com.databaselist;

import android.app.ListActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

import com.data.MapIntent;

public class AmfitheatreList extends ListActivity {

        String classes[] = { "Amf1", "Amf2", "Amf3", "Amf4", "Amf5" };

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            // TODO Auto-generated method stub
            super.onCreate(savedInstanceState);
            setListAdapter(new ArrayAdapter<String>(AmfitheatreList.this,
android.R.layout.simple_list_item_1, classes));
            Toast toast = Toast.makeText(this, "All Amfitheatres are on the Floor 0 -
Ground.", 8000);
            toast.setGravity(Gravity.BOTTOM, 0, 0);
            toast.show();
        }

        @Override
        protected void onListItemClick(ListView l, View v, int position, long id) {
```

```
		// TODO Auto-generated method stub
		super.onListItemClick(l, v, position, id);

		if (position == 0) {
			startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
1090, 820, 0, "Amf1"));

		} else if (position == 1) {
			startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
744, 740, 0, "Amf2"));

		} else if (position == 2) {
			startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
709, 740, 0, "Amf3"));

		} else if (position == 3) {
			startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
709, 657, 0, "Amf4"));

		} else {
			startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
744, 657, 0, "Amf5"));
		}
	}

	public void onBackPressed() {
		startActivity(new Intent("com.ece.POI"));
		finish();
		return;
	}
}
```

ClassRoomsList.java

```
package com.databaselist;

import android.app.ListActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

import com.data.MapIntent;
```

```
public class ClassRoomsList extends ListActivity {

	String classes[] = { "Classroom A.0.16-001", "Classroom A.0.15-002", "Classroom
A.0.14-003", "Classroom A.0.13-004", "Classroom A.0.12-005",
			"Classroom A.0.11-006", "Classroom A.0.10-007", "Classroom
B.0.1-013", "Classroom B.0.2-012", "Classroom B.0.3-0011", "Classroom B.0.4-010",
			"Classroom B.0.5-009", "Classroom B.0.6-008" };

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		setListAdapter(new ArrayAdapter<String>(ClassRoomsList.this,
android.R.layout.simple_list_item_1, classes));
		Toast toast = Toast.makeText(this, "All Classrooms are on the Floor 0 -
Ground", 5000);
		toast.setGravity(Gravity.BOTTOM, 0, 0);
		toast.show();
	}

	@Override
	protected void onListItemClick(ListView l, View v, int position, long id) {
		// TODO Auto-generated method stub
		super.onListItemClick(l, v, position, id);

		if (position == 0) {
			startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
1032, 476, 0, "Classroom"));

		} else if (position == 1) {
			startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
1032, 442, 0, "Classroom"));

		} else if (position == 2) {
			startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
1032, 348, 0, "Classroom"));

		} else if (position == 3) {
			startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
1032, 255, 0, "Classroom"));

		} else if (position == 4) {
			startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
1032, 160, 0, "Classroom"));

		} else if (position == 5) {
			startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
992, 255, 0, "Classroom"));

		} else if (position == 6) {
```

```
                        startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
992, 442, 0, "Classroom"));

                } else if (position == 7) {
                        startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
379, 435, 0, "Classroom"));

                } else if (position == 8) {
                        startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
379, 400, 0, "Classroom"));

                } else if (position == 9) {
                        startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
379, 246, 0, "Classroom"));

                } else if (position == 10) {
                        startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
379, 213, 0, "Classroom"));

                } else if (position == 11) {
                        startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
409, 264, 0, "Classroom"));

                } else {
                        startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
409, 382, 0, "Classroom"));

                }
        }

        public void onBackPressed() {
                startActivity(new Intent("com.ece.POI"));
                finish();
                return;
        }
}
```

ElevatorList.java

```
package com.databaselist;

import android.app.ListActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.widget.ArrayAdapter;
```

```
import android.widget.ListView;
import android.widget.Toast;

import com.data.MapIntent;

public class ElevatorList extends ListActivity {

	String classes[] = { "0 Floor - Ground", "1st Floor", "2nd Floor", "3rd Floor" };

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		setListAdapter(new ArrayAdapter<String>(ElevatorList.this,
android.R.layout.simple_list_item_1, classes));
		Toast toast = Toast.makeText(this, "Select Floor to see all Elevators", 5000);
		toast.setGravity(Gravity.BOTTOM, 0, 0);
		toast.show();
	}

	@Override
	protected void onListItemClick(ListView l, View v, int position, long id) {
		// TODO Auto-generated method stub
		super.onListItemClick(l, v, position, id);
		startActivity(MapIntent.DrawPoints(this.getApplicationContext(), 460, 543,
position, "Elevator"));
	}

	public void onBackPressed() {
		startActivity(new Intent("com.ece.POI"));
		finish();
		return;
	}
}
```

LabsList.java

```
package com.databaselist;

import android.app.ListActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;
```

```
import com.data.MapIntent;

public class LabsList extends ListActivity {

	String classes[] = { "Mobile and Personal Communications Lab - B.2.2",
"Electronics Lab B.1.1", "Microelectronics Lab B.1.4",
			"Computer Networks Lab - B.2.1", "Network Management –
Intelligent Networks Lab B.3.20", "PC LAB A.1.13-1", "PC LAB A.1.12-2", "PC LAB
A.1.11-3",
			"PC LAB A.1.11-4", "LAB Shmmy A.1.9-5", "LAB Shmmy A.2.9-
10", "Lab - B.2.4", "Lab B.2.19", "Lab B.2.20", "Lab - B.3.1", "Lab - B.3.2",
			"Lab - B.3.3", "Lab - B.3.4" };

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		setListAdapter(new ArrayAdapter<String>(LabsList.this,
android.R.layout.test_list_item, classes));
		Toast toast = Toast.makeText(this, "Select Lab to show the position on the
map", 10000);
		toast.setGravity(Gravity.BOTTOM, 0, 0);
		toast.show();
	}

	@Override
	protected void onListItemClick(ListView l, View v, int position, long id) {
		// TODO Auto-generated method stub
		super.onListItemClick(l, v, position, id);

		if (position == 0) {
			startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
425, 335, 2, "Mobile and Personal Communications Lab - B.2.2"));

		} else if (position == 1) {
			startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
425, 428, 1, "Electronics Lab B.1.1"));

		} else if (position == 2) {
			startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
425, 220, 1, "Microelectronics Lab B.1.4"));

		} else if (position == 3) {
			startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
425, 428, 2, "Computer Networks Lab B.2.1"));

		} else if (position == 4) {
			startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
425, 831, 3, "Network Management – Intelligent Networks Lab B.3.20"));
```

```
                } else if (position == 5) {
                        startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
1032, 476, 1, "PC LAB A.1.13-1"));

                } else if (position == 6) {
                        startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
1032, 442, 1, "PC LAB A.1.12-2"));

                } else if (position == 7) {
                        startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
1032, 348, 1, "PC LAB A.1.11-3"));

                } else if (position == 8) {
                        startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
1032, 255, 1, "PC LAB A.1.11-4"));

                } else if (position == 9) {
                        startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
1032, 160, 1, "LAB Shmmy A.1.9-5"));

                } else if (position == 10) {
                        startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
1032, 476, 2, "LAB Shmmy A.2.9-10"));

                } else if (position == 11) {
                        startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
425, 220, 2, "Lab - B.2.4"));

                } else if (position == 12) {
                        startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
425, 879, 2, "Lab - B.2.19"));

                } else if (position == 13) {
                        startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
425, 831, 2, "Lab - B.2.20"));

                } else if (position == 14) {
                        startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
425, 428, 3, "Lab - B.3.1"));

                } else if (position == 15) {
                        startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
425, 335, 3, "Lab - B.3.2"));

                } else if (position == 16) {
                        startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
425, 313, 3, "Lab - B.3.3"));

                } else {
```

```
                        startActivity(MapIntent.DrawPoints(this.getApplicationContext(),
425, 220, 3, "Lab - B.3.4"));

                }
        }

        public void onBackPressed() {
                startActivity(new Intent("com.ece.POI"));
                finish();
                return;
        }
}
```

SafetyExitList.java

```
package com.databaselist;

import android.app.ListActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

import com.data.MapIntent;

public class SafetyExitList extends ListActivity {

        String classes[] = { "0 Floor - Ground", "1st Floor", "2nd Floor", "3rd Floor" };

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                // TODO Auto-generated method stub
                super.onCreate(savedInstanceState);
                setListAdapter(new ArrayAdapter<String>(SafetyExitList.this,
android.R.layout.simple_list_item_1, classes));
                Toast toast = Toast.makeText(this, "Select Floor to see all Safety Exits",
5000);
                toast.setGravity(Gravity.BOTTOM, 0, 0);
                toast.show();
        }

        @Override
        protected void onListItemClick(ListView l, View v, int position, long id) {
```

```
		// TODO Auto-generated method stub
		super.onListItemClick(l, v, position, id);
		startActivity(MapIntent.DrawPoints(this.getApplicationContext(), 438, 150,
position, "Safety Exit"));
	}

	public void onBackPressed() {
		startActivity(new Intent("com.ece.POI"));
		finish();
		return;
	}
}
```

StairsList.java

```
package com.databaselist;

import android.app.ListActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

import com.data.MapIntent;

public class StairsList extends ListActivity {

	String classes[] = { "0 Floor - Ground", "1st Floor", "2nd Floor", "3rd Floor" };

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		setListAdapter(new ArrayAdapter<String>(StairsList.this,
android.R.layout.simple_list_item_1, classes));
		Toast toast = Toast.makeText(this, "Select Floor to see all Stairs", 5000);
		toast.setGravity(Gravity.BOTTOM, 0, 0);
		toast.show();
	}

	@Override
	protected void onListItemClick(ListView l, View v, int position, long id) {
		// TODO Auto-generated method stub
```

```
		super.onListItemClick(l, v, position, id);
		startActivity(MapIntent.DrawPoints(this.getApplicationContext(), 423, 543,
position, "Stairs"));
	}

	public void onBackPressed() {
		startActivity(new Intent("com.ece.POI"));
		finish();
		return;
	}
}
```

WCList.java

```
package com.databaselist;

import android.app.ListActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

import com.data.MapIntent;

public class WCList extends ListActivity {

	String classes[] = { "0 Floor - Ground", "1st Floor", "2nd Floor", "3rd Floor" };

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		setListAdapter(new ArrayAdapter<String>(WCList.this,
android.R.layout.simple_list_item_1, classes));
		Toast toast = Toast.makeText(this, "Select Floor to see all WC", 5000);
		toast.setGravity(Gravity.BOTTOM, 0, 0);
		toast.show();
	}

	@Override
	protected void onListItemClick(ListView l, View v, int position, long id) {
		// TODO Auto-generated method stub
```

```
                super.onListItemClick(l, v, position, id);

                startActivity(MapIntent.DrawPoints(this.getApplicationContext(), 428, 622,
position, "WC"));
        }

        public void onBackPressed() {
                startActivity(new Intent("com.ece.POI"));
                finish();
                return;
        }
}
```