ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Π. Μ. Σ. Διδακτική της Τεχνολογίας και Ψηφιακά Συστήματα

Κατεύθυνση: Ψηφιακές Επικοινωνίες και Δίκτυα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

"Ανάλυση και υλοποίηση στην πλατφόρμα Android εφαρμογής δωρεάν ανταλλαγής μηνυμάτων -chat- με την τεχνολογία Bluetooth"

Εισηγήτρια: Λεμονιά Ταμβάκη

Επιβλέπων: Γεώργιος Ευθύμογλου, Επίκουρος Καθηγητής

Διπλωματική Εργασία υποβληθείσα στο Τμήμα Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς ως μέρους των απαιτήσεων για την απόκτηση Μεταπτυχιακού Διπλώματος Ειδίκευσης στη Διδακτική της Τεχνολογίας και Ψηφιακών Συστημάτων

Πειραιάς, Μάιος 2015

UNIVERSITY OF PIRAEUS

DEPARTMENT OF DIGITAL COMMUNICATIONS

MASTER PROGRAM IN DIGITAL COMMUNICATIONS AND NETWORKS

MASTER THESIS TITLE

"Analysis and implementation of application for free chat via Bluetooth with Android"

By Lemonia Tamvaki

Master Thesis submitted to the Department of Digital Communications of the University of Piraeus in partial fulfillment of the requirements for the degree of Master of Arts in Digital Communications And Networks

Piraeus, Greece, May 2015

Αφιερώνεται στην οικογένεια μου και στην μνήμη του πατέρα μου και του παππού μου που δεν είναι πια κοντά μας.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον Επίκουρο Καθηγητή κο Γεώργιο Ευθύμογλου για την καθοδήγηση του, την υποστήριξη του και τις σωστές υποδείξεις του κατά τη διάρκεια της συγγραφής της διπλωματική μου εργασίας.

Ευχαριστίες επίσης οφείλονται στην κα Βέρα-Αλεξάνδρα Σταυρουλάκη και στον κο Απόστολο Μηλιώνη για τη βοήθεια που μου παρείχαν ώστε μέσω της διπλωματικής μου εργασίας να μπορέσω να κατανοήσω το θεωρητικό υπόβαθρο του θέματος που ανέλαβα.

Τέλος θα ήθελα να εκφράσω την ευγνωμοσύνη μου στους γονείς μου, στα αδέλφια μου, στον κο Διονύσιο Κοντογιάννη, προϊστάμενο μου στην ΕΡΤ και σε όσους με στήριξαν όλη αυτή τη χρονική περίοδο για την βοήθεια τους, για την κατανόηση τους και για την αμέριστη συμπαράσταση τους σ' αυτή την προσπάθεια μου.

Περίληψη

Το “Bluetooth” είναι μια πολλά υποσχόμενη τεχνολογία για ασύρματες επικοινωνίες σε προσωπικό και τοπικό επίπεδο. Ένα “Bluetooth scatternet” αποτελείται από επικαλυπτόμενα “piconets” το καθένα από τα οποία αποτελείται από ένα μικρό αριθμό συσκευών οι οποίες μοιράζονται το ίδιο κανάλι.

Η παρούσα πτυχιακή εργασία έχει ως στόχο την ανάλυση της λειτουργίας της εφαρμογής ‘Bluetooth Chat’ για την πλατφόρμα Android. Ο σκοπός της εφαρμογής είναι η ελευθερη επικοινωνία δυο συσκευών, οι οποίες χρησιμοποιούν το λειτουργικό σύστημα Android, μέσω της τεχνολογίας Bluetooth. Η λειτουργία του Bluetooth γίνεται με τις ραδιοσυχνότητες και χρησιμοποιεί τη ζώνη των ραδιοσυχνοτήτων των 2.4 GHz η οποία έχει παγκοσμία καθιερωθεί για να χρησιμοποιείται από τη βιομηχανική, επιστημονική και ιατρική κοινότητα (ISM - Industrial, Scientific, Medicine) ελεύθερα, χωρίς αδειοδότηση. Η ζώνη των συχνοτήτων ISM έχει εύρος 83.5MHz και είναι διαθέσιμη στις συσκευές ανεξάρτητα από τη γεωγραφική θέση στην οποία βρίσκονται.

Το θεωρητικό μέρος της παρούσας εργασίας ασχολείται με την πλατφόρμα Android, την αρχιτεκτονική και το περιβάλλον ανάπτυξης της καθώς και τα χρησιμοποιούμενα εργαλεία στην ανάπτυξη της εφαρμογής για κινητά. Τα σημαντικά στοιχεία και οι διαδικασίες της ανάπτυξης της Android εφαρμογής περιγράφεται στην ανάπτυξη της διαδικασίας.

Στο 1^{ο} κεφάλαιο της εργασίας, αναφέρονται γενικές πληροφορίες για την αρχιτεκτονική, τα τεχνικά χαρακτηριστικά και τις ιδιαιτερότητες της τεχνολογίας Bluetooth.

Το 2^{ο} κεφάλαιο περιέχει στοιχεία σύγκρισης της παραπάνω τεχνολογίας με άλλες συναφείς τεχνολογίες που έχουν αναπτυχθεί στο χώρο των ασύρματων τεχνολογιών και παρουσιάζει τη θέση που κατέχει σήμερα η τεχνολογία αυτή στην αγορά αλλά και τις προσδοκίες του εμπορικού κόσμου για την μελλοντική της εξέλιξη. Ακόμη στο 2^{ο} κεφάλαιο δίνεται μία ερμηνεία για το πως κατάφερε να σκαρφαλώσει στην αγοραστική θέση που βρίσκεται σήμερα η υπό εξέταση τεχνολογία.

Στο 3^{ο} κεφάλαιο της εργασίας αναφέρονται γενικές πληροφορίες για την αρχιτεκτονική του λειτουργικού συστήματος ‘Android’ καθώς και το γεγονός επικράτησης του στην αγορά έναντι των άλλων τεχνολογιών. Στο 4^{ο} κεφάλαιο γίνεται

μια σύντομη αναφορά στην αρχιτεκτονική μιας εφαρμογής ‘Android’ και στα δομικά στοιχεία που την απαρτίζουν.

Στο 5ο κεφάλαιο αναφέρονται οι απαιτήσεις τόσο στο υλικό (hardware) όσο και στο λογισμικό (software) για την ανάπτυξη της εφαρμογής αυτής. Στο κεφάλαιο αυτό παρουσιάζονται ακόμη οι δυσχέρειες και τα προβλήματα που προέκυψαν και πως αυτά αντιμετωπίστηκαν.

Το 6^{ο} κεφάλαιο περιέχει τα ‘UML’ διαγράμματα που αναπτυχτήκαν πριν την υλοποίηση και την κωδικοποίηση της εφαρμογής. Ακόμη παρουσιάζονται σταδιακά τα βήματα της ανάλυσης και της σχεδίασης που οδήγησαν στην ανάπτυξη αυτού του κώδικα.

Το 7^{ο} κεφάλαιο παρουσιάζει την υλοποίηση του κώδικα της εφαρμογής σε ορισμένα βασικά σημεία όπως: Α) ο τρόπος επίτευξης της ασύρματης σύνδεσης μέσω Bluetooth, Β) η αποστολή και λήψη μηνυμάτων, Γ) η αποθήκευση (ή όχι) των μηνυμάτων αυτών σε δύο βάσεις ‘sqLite’, Δ) η αναζήτηση παλαιότερων μηνυμάτων από τη βάση δεδομένων και η διαγραφή τους από τη βάση. Στο κεφάλαιο ακόμη παρουσιάζεται η λειτουργία της εφαρμογής και απεικονίζονται οι οθόνες των συσκευών κατά τη λειτουργία αυτή.

Στο 8^{ο} και τελευταίο κεφάλαιο γίνεται μία επισκόπηση της διπλωματικής εργασίας και κατατίθενται τα συμπεράσματα που συνάχθηκαν κατά τη διάρκεια της εκπόνησης της.

Abstract

Bluetooth is a promising technology for personal/local area wireless communications. A Bluetooth scatternet is comprised of overlapping piconets, each comprising a small number of devices that share the same channel.

This thesis aims to analyze the operation of Bluetooth Chat application used in the Android platform. The purpose of the application is to establish via Bluetooth technology the free communication between two devices that use the Android operating system. The Bluetooth function is done with the radio frequency and uses the radio frequency band of 2.4 GHz which has a worldwide established for use by industrial, scientific and medical community (ISM - Industrial, Scientific, Medicine) freely, without authorization.

The theoretical part of this paper examines the Android platform, its architecture and development environment as well as the tools used in the Android mobile application development. The key elements and procedures of the Android application development are described in the development process.

In the first chapter, general information refers to the architecture, specifications and characteristics of the Bluetooth technology.

In the second chapter these data are compared with other relevant technologies developed in the wireless technologies field. The chapter shows the position of this technology on the market today and the expectations of the commercial world for future development. It also gives an explanation of how this technology managed to gain its current commercial share on purchasing current location.

In the third chapter general information are mentioned about the architecture of the Android operating system and its market dominance over other technologies.

The fourth chapter is a brief reference to the architecture of an Android app and the structural elements its composed of. In Chapter 5 requirements both hardware and the software are mentioned for the development of this application. It also presents the encountered difficulties and problems and how they were treated.

The sixth chapter contains the UML diagrams that were formed before the implementation and consolidation of the application. It attempts to present in part the stages of analysis and design which led to the development of this code.

The seventh chapter presents the implementation of application code in key points, such as: A) How to achieve the wireless connection via Bluetooth. B) How to send and receive messages. C) How these messages can be saved (or not) of on two bases sqLite. D) How to search past messages from the database and delete them from the base. The chapter also presents the application and displays the devices during this operation.

Finally, in the eighth and final chapter on overview of the thesis and its conclusions during the development are mentioned.

"Ανάλυση και υλοποίηση στην πλατφόρμα Android εφαρμογής δωρεάν ανταλλαγής μηνυμάτων -chat- με την τεχνολογία Bluetooth"

Σημαντικοί όροι στα ελληνικά:

AM: Ενεργό μέλος (Active Member)

BD_ADDR: Διεύθυνση συσκευής Bluetooth (Bluetooth Device Address)
BTSF: Αλγόριθμος BTSF

CAC: Κωδικός πρόσβασης καναλιού (Channel Access Code)
CLK: Ρολόι συγχρονισμού της συσκευής (The Master clock)
CRC: Έλεγχος Κυκλικού Πλεονασμού (Cyclic Redundancy Check)

DAC: Κωδικός Πρόσβασης Συσκευής (Device Access Code)
DH: Πακέτα Data High (Rate)
DM: Πακέτα Data Medium (Rate)
DPSK: Διαφορική μετατόπιση του φασματος (Differential phase-shift keying)

DVM: Εικονική μηχανή Dalvic (Dalvic Virtual Machine)

EDR: Ενισχυμένη ταχύτητα δεδομένων (Enhanced Data Rate)

FEC: Εμπρόσθια διόρθωση σφάλματος (Forward Error Correction)
FHSS: Διασπορά φάσματος με εναλλαγή συχνότητας (Frequency Hop Spread Spectrum)

IEEE: Ινστιτούτο Ηλεκτρολόγων και Ηλεκτρονικών Μηχανικών (Institute of Electrical and Electronic Engineers)
ISM: Βιομηχανική, επιστημονική και ιατρική κοινότητα (Industrial Scientific and Medica)

JDK: Εργαλείο ανάπτυξης κώδικα της Java (Java Development Kit)

LAN: Τοπικό δίκτυο (Local Area Network)
LMP: Πρωτόκολλο διαχείρησης συνδέσεων (Link manager protocol)

OBEX: Πρωτόκολλο Ανταλλαγής Αντικειμένων (OBject EXchange (protocol))

PAN: Προσωπικά δίκτυα (Personal Area Network)
PC: Προσωπικός υπολογιστής (Personal Computer)
P2P: Σύνδεση σημείο με σημείο (point to point) ή κόμβο με κόμβο (peer to peer)

QoS: Ποιότητα παροχής υπηρεσιών (Quality of Service)

SDP: Πρωτόκολλο ανίχνευσης υπηρεσιών (Service Discovery Protocol)
SIG: (Bluetooth) Special Interest Group

TDD: Τεχνική Time - Division Duplex
TDM: Πολυπλεξία διαίρεσης χρόνο (Time Division Multiplexing)
TDMA: Πολλαπλή προς

WAP: Πρωτόκολλο Ασύρματων Εφαρμογών (Wireless Application Protocol)
WLAN: Ασύρματο Τοπικό Δίκτυο (Wireless Local Area Network)

Συντμήσεις σημαντικών όρων:

Β.Δ.: Βάση Δεδομένων
Β.Τ.: Υπηρεσία Bluetooth

Δ.Π.Χ.: Διάγραμμα Περιπτώσεων Χρήσης
Δ.Δ.: Διάγραμμα Δραστηριοτήτων
Δ.Κ.: Διάγραμμα Κλάσεων

Η/Υ: Ηλεκτρονικός Υπολογιστής

Λ.Σ.: Λειτουργικό Σύστημα

"Analysis and implementation of application for free chat via Bluetooth with Android"

Keywords:

ACL: Asynchronous Connectionless
AM: Active Member
API: Application Programming Interface

BD_ADDR: Bluetooth Device Address
BTSF αλγόριθμο

CAC: Channel Access Code
CLK: The Master clock
CRC: Cyclic Redundancy Check

DAC: Device Access Code
DH: Data High (Rate)
DM: Data Medium (Rate)
DQPSK: Differential Quaternary Phase Shift Keying
DPSK: Differential phase-shift keying

DVM: Dalvic Virtual Machine

EDR: Enhanced Data Rate

FEC: Forward Error Correction
FHSS: Frequency Hop Spread Spectrum

GFSK: Gaussian Frequency Shift Keying

HCI: Host Controller Interface

IAC: Inquiry Access Code
IEEE: Institute of Electrical and Electronic Engineers
ISM: Industrial Scientific and Medical (band)

JDK: Java Development Kit

LAN: Local Area Network
L2CAP: Logical link control and adaptation protocol
LMP: Link manager protocol

MAC: Medium Access Control

OBEX: OBject EXchange (protocol)

PAN: Personal Area Network
PC: Personal Computer

P2P: -point to point- or -peer to peer-

QoS: Quality of Service

SCO: Synchronous Connection Oriented
SDP: Service Discovery Protocol
SIG: (Bluetooth) Special Interest Group

TDD: Time Division Duplex
TDM: Time Division Multiplexing
TDMA: Time division multiple access

USB: Universal Serial Bus

WAP: Wireless Application Protocol
WLAN: Wireless Local Area Network

Πίνακας περιεχομένων

Ευρετήριο πινάκων

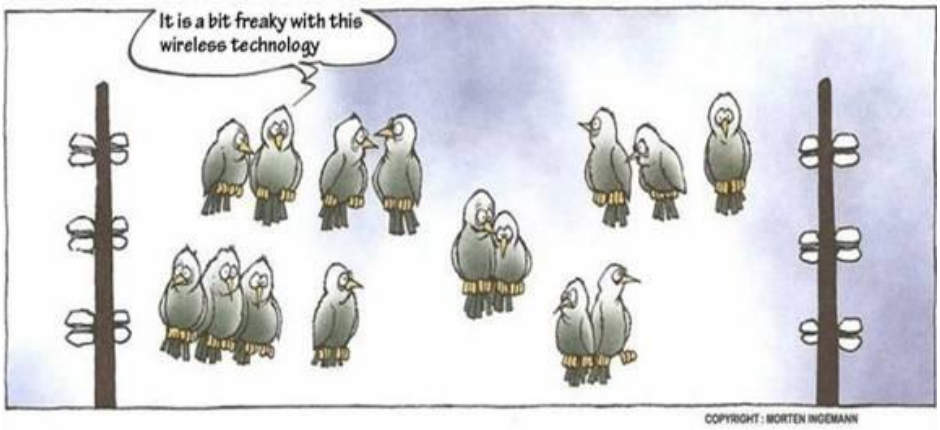
Ευρετήριο διαγραμμάτων

Ευρετήριο εικόνων

1 Η προδιαγραφή του Bluetooth

1.1 Τεχνολογίες ασύρματης δικτύωσης

Σ' ένα ασύρματο δίκτυο η επικοινωνία των συσκευών γίνεται μέσω των ηλεκτρομαγνητικών κυμάτων. Η πληροφορία που θα αποσταλεί, διαμορφώνει κατάλληλα ένα φέρον το οποίο εκπέμπεται στον αέρα. Η συχνότητα του φέροντος αλλά και ο τρόπος διαμόρφωσης, είναι χαρακτηριστικά καθοριστικής σημασίας στην επιλογή του σωστού δικτύου πράγμα που επηρεάζει παραμέτρους όπως είναι η εμβέλεια, η μέγιστη διαμεταγωγή, η αντοχή στο θόρυβο και τις παρεμβολές.



Εικόνα 1.1 Ασύρματη δικτύωση (σκίτσο του Morten Ingemann)

Η χρήση υπερύθρων ακτινών στα ασύρματα τοπικά δίκτυα (Wireless Local Area Networks) κρίθηκε ακατάλληλη λόγω της μικρής εμβέλειας που μπορούν να καλύψουν καθώς και της απαίτησης για οπτική επαφή μεταξύ πομπού και δέκτη που υπάρχει για τη λειτουργία τους. Ακατάλληλη κρίθηκε και η χρήση των μικροκυμάτων χαμηλής ισχύος, λόγω του μεγάλου κόστους των πομποδεκτών. Για την ασύρματη επικοινωνία επιλέχθηκαν οι ράδιο-συχνότητες (ή ραδιοκύματα *Radio frequency*), οι οποίες έχουν χρησιμοποιηθεί σε πολλές εφαρμογές καθώς η τεχνογνωσία τους είναι ιδιαίτερα αναπτυγμένη, ως η πλέον ενδεδειγμένη λύση.

Ο τρόπος διαμόρφωσης στα συστήματα ασύρματης δικτύωσης διαφέρει ανάλογα με την τεχνολογία και το πρότυπο που ακολουθείται. Στο φυσικό επίπεδο χρησιμοποιούνται κυρίως τρία είδη πρόσβασης: η Διαμόρφωση με Εναλλαγή της Συχνότητας FHSS (Frequency Hopping Spread Spectrum), η Διαμόρφωση της Ευθείας Ακολουθίας DSSS (Direct Sequence Spread Spectrum) και η Ορθογωνική

Πολυπλεξία με Διαίρεση Συχνότητας OFDM (Orthogonal Frequency Division Multiplexing).

Η διαμόρφωση και η μετάδοση του φάσματος επιτυγχάνεται με δύο σημαντικές τεχνικές διασποράς του φάσματος, η μία είναι η διαμόρφωση με εναλλαγή (ή αναπήδηση) συχνότητας (Frequency Hopping Spread Spectrum ή FHSS) και η άλλη τεχνική είναι η διασπορά ευθείας ακολουθίας (Direct Sequence Spread Spectrum ή DSSS). Κάθε τεχνική ορίζει διαφορετικό τρόπο με τον οποίο μπορεί να συνδυαστεί ο κώδικας διασποράς (ή μετάδοσης) με το σήμα της πληροφορίας.

Στην διαμόρφωση με εναλλαγή της συχνότητας FHSS, το διαθέσιμο εύρος συχνοτήτων διαχωρίζεται σε κανάλια. Ο πομπός εκπέμπει τα δεδομένα αλλάζοντας κανάλι σε περιοδικά (ή προκαθορισμένα) χρονικά διαστήματα. Η σειρά των καναλιών που θα χρησιμοποιηθούν για τη μετάδοση των πληροφοριών είναι γνωστή μόνο στον πομπό και στο δέκτη που επικοινωνούν. Το διαθέσιμο εύρος συχνοτήτων, ο αριθμός των καναλιών και ο προσδιορισμός του χρονικού διαστήματος της διάθεσης- απασχόλησης του κάθε καναλιού είναι παράμετροι που καθορίζονται από την εκάστοτε υλοποίηση.

Στην διαμόρφωση της ευθείας ακολουθίας DSSS, αποστέλλεται σε ένα προκαθορισμένο κανάλι μια σειρά από bit. Κάθε bit του σήματος της πληροφορίας «μετασχηματίζεται» με ένα προκαθορισμένο κώδικα οποίος καλείται κώδικας μετάδοσης (spreading code), σε μια ακολουθία bits του μεταδιδόμενου σήματος. Όσο περισσότερα bits έχει ο κωδικός μετάδοσης τόσο πιο ανθεκτικός γίνεται στις παρεμβολές. Τα πλεονάζοντα bit μπορούν να χρησιμοποιηθούν στον έλεγχο και στη διόρθωση των σφαλμάτων μειώνοντας την ανάγκη της εκ νέου αποστολής πακέτων και αυξάνοντας έτσι την αξιοπιστία του συστήματος.

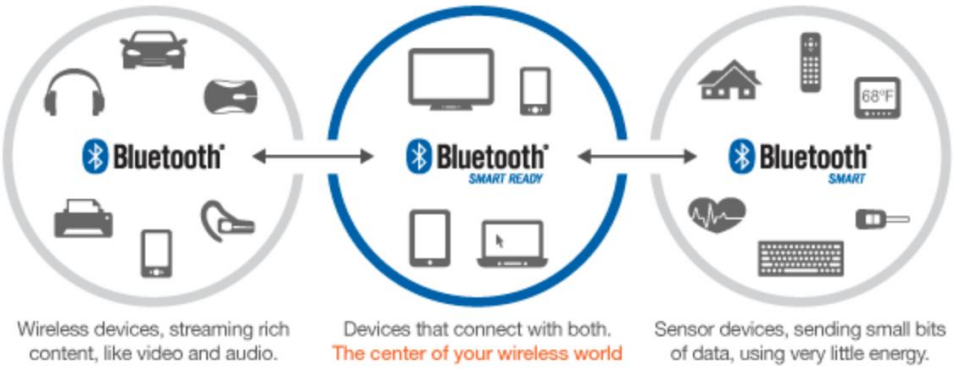
Η Ορθογωνική Πολυπλεξία με Διαίρεση Συχνότητας, OFDM, θεωρείται ότι είναι η καταλληλότερη μέθοδος για τη μετάδοση υψηλού ρυθμού δεδομένων λόγω της μεγάλης ανοχής που έχει στη διασυμβολική παρεμβολή. Στην πολυπλεξία της ορθογώνιας διαίρεσης της συχνότητας το σήμα διαιρείται σε πολλαπλά σήματα χαμηλότερης ισχύος. Οι συχνότητες που αποδίδονται στα κανάλια έχουν τέτοια απόσταση ώστε τα σήματα που στέλνονται να είναι ορθογώνια μεταξύ τους, δηλαδή να μην υπάρχουν παρεμβολές μεταξύ των συχνοτήτων. Τα σήματα μεταδίδονται συγχρόνως σε διαφορετικές συχνότητες. Ταυτόχρονα ο δέκτης επεξεργάζεται και

αυτός τα σήματα με σκοπό την ανάκτηση του αρχικού σήματος. Έτσι όσο μεγαλώνει ο αριθμός των υποκαναλιών που διαιρούμε το σήμα μας, τόσο αυξάνει η μέγιστη διαμεταγωγή που μπορούμε να επιτύχουμε αλλά αυξάνει ταυτόχρονα και η πιθανότητα της απώλειας δεδομένων.

Μερικές από τις επικρατέστερες τεχνολογίες ασύρματης δικτύωσης είναι ήδη διαθέσιμες στον καταναλωτή, έχοντας εδραιώσει τη θέση τους στην παγκόσμια αγορά, ενώ κάποιες άλλες βρίσκονται στο στάδιο της παραγωγής και διάθεσης αναζητώντας το δικό τους μερίδιο από την αγορά. Στις επόμενες παραγράφους αυτού του κεφαλαίου, θα εξετάσουμε την τεχνολογία του Bluetooth η οποία κατατάσσεται στην κατηγορία των ασύρματων δικτύων προσωπικής εμβέλειας WPAN (Wireless Personal Area Networks) και αναφέρονται συνοπτικά τα τεχνικά χαρακτηριστικά της, ο τρόπος λειτουργίας της και η ασφάλεια που παρέχει στους χρήστες.

1.2 Η τεχνολογία του Bluetooth

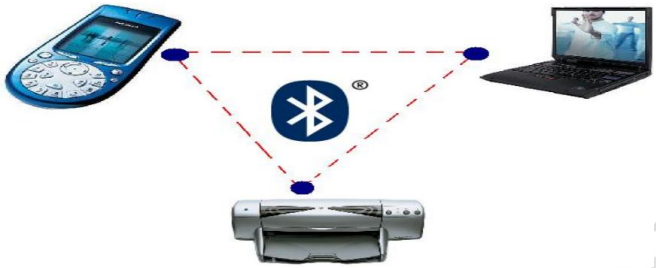
Η τεχνολογία του δικτύου του Bluetooth δημιουργήθηκε από τις εταιρίες κινητής τηλεφωνίας Ericsson, Nokia, Motorola, Intel, IBM, Toshiba και 3COM. Πρόκειται για μια τεχνολογία που ανήκει στην κατηγορία των ασύρματων προσωπικών δικτύων (WPAN) και δημιουργήθηκε με σκοπό την αντικατάσταση των καλωδίων της σύνδεσης στις φορητές συσκευές, την χαμηλή κατανάλωση ενέργειας και την δημιουργία εύρωστων δικτύων.



Εικόνα 1.2 Συνδεσιμότητα συσκευών σε δίκτυα Bluetooth

Η τεχνολογίας αυτή χρησιμοποιώντας τις ραδιοσυχνότητες χαμηλής ισχύος δημιουργεί δίκτυα μικρής εμβέλειας και ως εκ τούτου το πρωτόκολλο 802.15 που

αναπτύχθηκε από την IEEE για τα ασύρματα προσωπικά δίκτυα είναι βασισμένο σε αυτή.



Εικόνα 1.3 Σενάριο χρήσης του Bluetooth

Το Bluetooth είναι μια προδιαγραφή (IEEE 802.15.1) σχεδιασμένη κυρίως για να υποστηρίξει την ασύρματη δικτύωση των προσωπικών και περιφερειακών συσκευών όπως είναι τα τηλέφωνα, τα PDA, οι εκτυπωτές, οι υπολογιστές και άλλες συσκευές.

Εικόνα 1.4 Σύνδεση περιφερειακών συσκευών με Bluetooth

Όταν μια συσκευή Bluetooth βρεθεί στην εμβέλεια μιας άλλης συσκευής, τότε μπορεί να δημιουργηθεί ένα αυτοοργανωμένο δίκτυο ή κατ' απαίτηση –αδόμητο- δίκτυο (ad hoc) το οποίο είναι ένας αποκεντρωμένος τύπος ασύρματου δικτύου για την ανταλλαγή πληροφοριών. Μπορούν να συνδεθούν μέχρι και οχτώ συσκευές (η μια συσκευή αναλαμβάνει το ρόλο του κύριου κόμβου) σε ένα δίκτυο Bluetooth (piconet) για την ανταλλαγή πληροφοριών. [1], [2]

1.3Ιστορικά στοιχεία, η ονομασία και το λογότυπο

Στις μέρες μας, η τεχνολογία "Bluetooth" αποτελεί αναφορά στις προδιαγραφές σχεδιασμού της ασύρματης σύνδεσης Bluetooth. Η μη ύπαρξη κάποιου ευρέως αποδεκτού πρότυπου για τα WPAN μέχρι τα τέλη της δεκαετίας του 1990 οδήγησε στην ανάπτυξη της τεχνολογίας αυτής. Πρώτη η Ericsson έθεσε τις βάσεις για την ανάπτυξη μίας τεχνολογίας η οποία θα επέτρεπε τον σχηματισμό τοπικών δικτύων πολύ μικρής εμβέλειας με σκοπό την ασύρματη και την κατ’ απαίτηση –αδόμητη- (ad

hoc) δικτύωση ετερογενών φορητών συσκευών. Το πρότυπο που προέκυψε υιοθετήθηκε στη συνέχεια από την IEEE ως το πρότυπο 802.15 για τα WPAN.

Εικόνα 1.5

Οι σχεδιαστές του προτύπου αυτού ήταν του 802.15 ήταν βέβαιοι ότι το νέο πρότυπο θα επικρατούσε στην αγορά λόγω της δωρεάν διάθεσης του. Το κύριο χαρακτηριστικό του προτύπου αυτού ήταν η σύνδεση των ετερογενών δικτύων πράγμα που έπαιξε σημαντικό ρόλο στην επιλογή του ονόματος του.

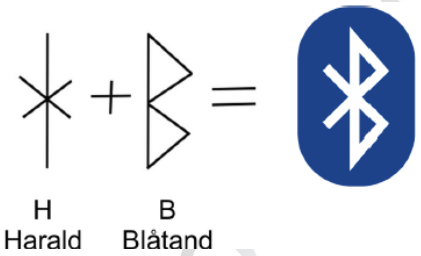
Το όνομα Bluetooth υιοθετήθηκε επίσημα το 1998 και είναι δανεισμένο από το βασιλιά της Δανίας Harald Blaatand ή Χάραλντ ο Κυανόδους, γνωστός και με το ψευδώνυμο "Bluetooth", ο οποίος έζησε στα τέλη του 10ου αιώνα μ.Χ. Ο Χάραλντ ήταν ο πρωτότοκος γιός του βασιλιά Γκορμ που κυβερνούσε για πολλά έτη τη Γιουτλάνδη, τη μεγαλύτερη χερσόνησο της Δανίας. Η ανατροφή του υπήρξε ανάλογη της καταγωγής του και των παραδόσεων της φυλής των Βίκινγκς. Το μεγαλύτερο μέρος των Σκανδιναβών υπηκόων του ασχολούνταν με τις αγροτικές εργασίες. Μέσα σε αυτό το κοινωνικό περιβάλλον μεγάλωσε και ανδρώθηκε ο Harald Blatand (Κυανόδους), το όνομα του οποίου έχει τις ρίζες του σε δύο αρχαίες δανέζικες λέξεις: bla (σκουρόδερμος) και tan (γενναίος άνδρας). Αυτά περίπου αναφέρονται σε ένα δελτίο τύπου της Ericsson το οποίο δημοσιεύθηκε το 1999.

Εικόνα 1.6 Ο βασιλιάς Harald Blatand (Κυανόδους) [3]

Ο Χάραλντ υπήρξε ένας αρκετά δραστήριος βασιλιάς, οραματιζόταν την ένωση των γειτονικών του σκανδιναβικών κρατών και ο μόνος τρόπος να το πετύχει ήταν οι στρατιωτικές επιχειρήσεις. Υπάρχουν ιστορικές αναφορές που επιβεβαιώνουν το γεγονός ότι κατάφερε να επικρατήσει της Δανίας και της Νορβηγίας και να

δημιουργήσει έτσι ένα μεγάλο σκανδιναβικό κράτος. Το πρωτόκολλο επικοινωνίας συσκευών Bluetooth πήρε το όνομά του από αυτόν το Δανό βασιλιά, φαινομενικά λόγω της ικανότητας του να κάνει διαφορετικές φατρίες να επικοινωνούν και να συνεργάζονται μεταξύ τους. Σύμφωνα με το μύθο, ο Δανός βασιλιάς κέρδισε το παρατσούκλι "Bluetooth" από την αγάπη του για τα βατόμουρα (αγγλ. Blackberry), τα οποία κατά την κατανάλωση τους χρωμάτιζαν μπλε τα δόντια του.

Η ιδέα αυτού του ονόματος προτάθηκε το 1997 από τον Jim Kardach (κατά το χρόνο που διάβαζε ένα ιστορικό μυθιστόρημα του Frans Gunnar Bengtsson 's «Πλοία Βίκινγκς και ο βασιλιάς Harald Bluetooth») ο οποίος ανέπτυξε ένα σύστημα που επιτρέπει στα κινητά τηλέφωνα να συνδέονται με τους υπολογιστές.



Εικόνα 1.7 Το λογότυπο του Bluetooth με τους ρούνους Η και Β [3]

Το λογότυπο του Bluetooth αποτελείται από τους σκανδιναβικούς ρούνους, Η και Β (Hagall) (ᚼ) και (Bjarkan) (ᛒ) των αρχικών του ονόματος του βασιλιά Harold Bluetooth. Το 1998 δημιουργείται και επίσημα η κοινοπραξία των εταιρειών Ericson, Intel, International Business Machines (IBM), Nokia και Toshiba με την επωνυμία Bluetooth Special Interest Group (SIG) με κύριο στόχο την ανάπτυξη των ανοικτών προδιαγραφών της ασύρματης τεχνολογίας του Bluetooth. Τα επόμενα χρόνια πολλές άλλες εταιρίες ενσωματώθηκαν στην ομάδα του Bluetooth SIG με την μόνη υποχρέωση να υποστηρίζουν στις συσκευές που κατασκευάζουν την τεχνολογία του Bluetooth. Συμφώνα με την επίσημη ιστοσελίδα της τεχνολογίας αυτής η ομάδα του Bluetooth SIG σήμερα απαριθμεί περίπου στις 25.000 εταιρείες μέλη. [2] [4] [3] [5] [6]

1.4 Η αρχιτεκτονική της τεχνολογίας του Bluetooth

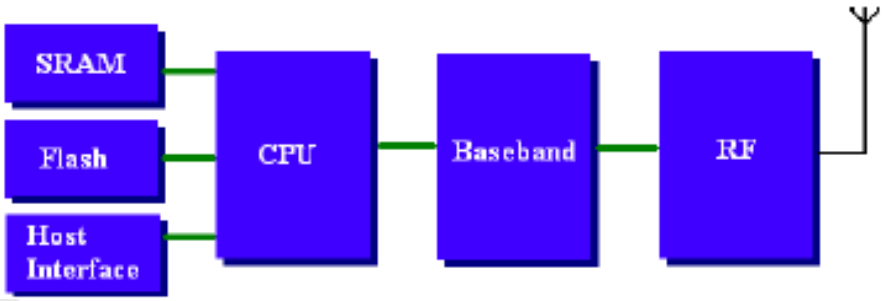
Τα WPAN εντάσσονται στην κατηγορία των ασύρματων δικτύων. Είναι προσωπικά δίκτυα πολύ μικρής εμβέλειας που στόχο έχουν στην ασύρματη και ad

hoc δικτύωση ετερογενών φορητών συσκευών. Το σπουδαιότερο πρότυπο σ' αυτό το χώρο είναι η οικογένεια των πρωτοκόλλων Bluetooth που σχεδιάστηκε από μία ομάδα εταιρειών και υιοθετήθηκε στη συνέχεια από την IEEE ως το πρότυπο 802.15 για WPAN.

Το hardware της σύνδεσης του Bluetooth μπορεί να είναι ενσωματωμένο σε ένα τσιπ ή σε μια ασύρματη μονάδα. Αν και ο αρχικός στόχος ήταν η κατασκευή ενός chip, πολλοί προμηθευτές χρησιμοποιούν σήμερα ένα chip για το baseband και ένα chip για το RF, λόγω των δυσχερειών της ενσωμάτωσης μέρους του RF σε chip CMOS.

Εικόνα 1.8 Μια μονάδα Bluetooth

Αυτό το μηχανικό μέρος χειρίζεται τη διαδικασία της μετάδοσης και της λήψης, καθώς απαιτείται η ψηφιακή επεξεργασία του σήματος στο πρωτόκολλο της βασικής ζώνης. Οι λειτουργίες του περιλαμβάνουν τη δημιουργία της σύνδεσης, την υποστήριξη της ασύγχρονης σύνδεσης (δεδομένα) και των σύγχρονων συνδέσεων (3 κανάλια για τη φωνή), την διόρθωση των σφαλμάτων και τον έλεγχο της ταυτότητας ή την αυθεντικοποίηση.



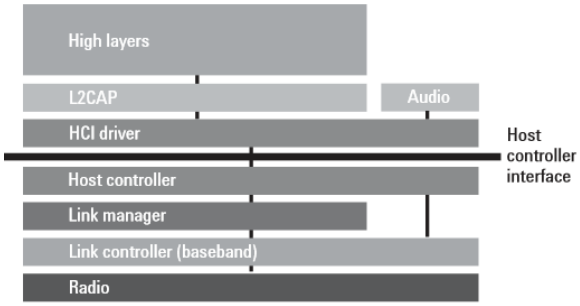
Εικόνα 1.9 Το hardware του Bluetooth

Η φυσική σύνδεση πραγματοποιείται στην ασύρματη μονάδα (radio –φυσικό επίπεδο).

Η μονάδα ελέγχου ζεύξης (Link controller (baseband)) του υποεπίπεδου Mac αναλαμβάνει τις υπηρεσίες ελέγχου, επίσης είναι υπεύθυνη για την αποστολή και

λήψη των πακέτων των δεδομένων μέσω της ασύρματης ζεύξης. Η μετάδοση των δεδομένων γίνεται μέσω καναλιών αναμετάδοσης τύπου ACL (Asynchronous Connectionless) ή SCO (Synchronous Connection Oriented) (βλ. παράγραφο 1.6.1.6).

Ο διαχειριστής της σύνδεσης του μηχανικού μέρους με το λογισμικό, Link Manager, εμπεριέχεται στον επεξεργαστή (CPU) της βασικής ζώνης και διαχειρίζεται τις εργασίες της δικτύωσης. Εκτελεί τον εντοπισμό των συσκευών, την εγκατάσταση της σύνδεσης, την αυθεντικοποίηση, την ασφάλεια των συνδέσεων και τη διαμόρφωση του συνδέσμου. Οι διαχειριστές της σύνδεσης δύο συσκευών επικοινωνούν χρησιμοποιώντας το πρωτόκολλο διαχείρισης της σύνδεσης Link Management Protocol ή LMP (βλ. παράγραφο 1.5 Έλεγχος και διαχείριση της στοίβας των πρωτοκόλλων του Bluetooth).



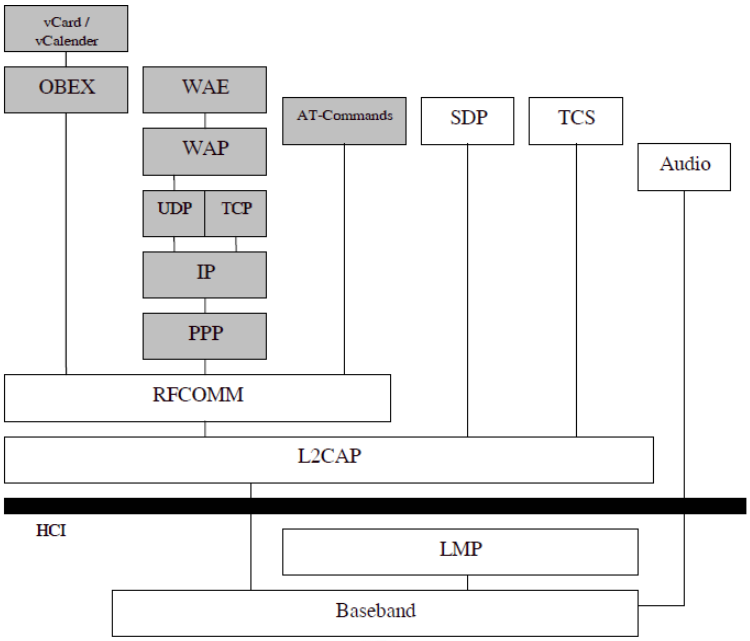
Εικόνα 1.10 Η διεπαφή του ελεγκτή HCI (Host Controller Interface) [5]

Η διεπαφή του κεντρικού ελεγκτή HCI (Host Controller Interface) παρέχεται από το hardware του Bluetooth, η διεπαφή αυτή ουσιαστικά παρεμβάλλεται ανάμεσα στο λογισμικό και τους ελεγκτές του υλικού μέρους (hardware) προκειμένου να επιτευχθεί η επικοινωνία του λογισμικού και του υλικού μέρος της συσκευής. [1] [2] [5] [6]

1.5 Έλεγχος και διαχείριση της στοίβας των πρωτοκόλλων του Bluetooth

Όπως έχει αναφερθεί και στην παράγραφο 1.2 "Η τεχνολογία του Bluetooth" το πρωτόκολλο του δικτύου του Bluetooth δημιουργήθηκε για να καταργήσει τα καλώδια στη σύνδεση των συσκευών, να υποστηρίξει την δικτύωση ad hoc, να παρέχει δωρεάν σύνδεση μεταξύ των συσκευών και να έχει χαμηλή κατανάλωση ενέργειας. Για να μπορέσει το πρωτόκολλο του Bluetooth να υλοποιήσει όλες αυτές

τις απαιτήσεις έχει μια στοίβα πρωτοκόλλων (Bluetooth Protocol Stack) που περιέχει πρωτόκολλα όπως το πρωτόκολλο της ασύρματης σύνδεσης, το πρωτόκολλο τηλεφώνου, το πρωτόκολλο ανακάλυψης υπηρεσιών κλπ (Εικόνα 1.27).



Εικόνα 1.11 Η στοίβα του πρωτοκόλλου Bluetooth [1]

Τα πρωτόκολλα του πυρήνα είναι πρωτόκολλα τεσσάρων επιπέδων:

Βασική Ζώνη (Baseband) Ορίζει τις λεπτομέρειες της ασύρματης σύνδεσης στο Φυσικό Επίπεδο (ή RF) όπως είναι η συχνότητα, το σχήμα της διαμόρφωσης, την ισχύ της μετάδοσης. Περιλαμβάνει τη διεπαφή Link Control (βλ. παράγραφο 1.4 Η αρχιτεκτονική της τεχνολογίας του Bluetooth). Επίσης καθορίζει το πρωτόκολλο για την εγκαθίδρυση της σύνδεσης σε ένα piconet, τη διευθυνσιοδότηση, τον τύπο των πακέτων, τη σειρά της μετάδοσης, τη χρονική ακολουθία, τον έλεγχο της ισχύος και την κωδικοποίηση του καναλιού.

Πρωτόκολλο Διαχείρισης Ζεύξης (Link Management Protocol, LMP) Είναι υπεύθυνο για την εγκατάσταση της σύνδεσης μεταξύ των συσκευών, για την κρυπτογράφηση της πληροφορίας και την ασφάλεια της σύνδεσης. Επιτρέπει την ανακάλυψη υπηρεσιών, οι οποίες ανιχνεύονται από άλλες Bluetooth συσκευές που βρίσκονται εντός της εμβέλειας της συσκευής. Το πρωτόκολλο LMP αναλαμβάνει υπηρεσίες ελέγχου και διαχείρισης του δικτύου, όπως είναι: η πιστοποίηση των κόμβων, ο συγχρονισμός του

ρολογιού για το FHSS, η προσαρμογή της ισχύος εκπομπής με βάση τη λαμβανόμενη ισχύ, η επιλογή του κατάλληλου τύπου πλαισίου MAC ανάλογα με το κανάλι, η εγκαθίδρυση συνδέσεων SCO κλπ.

Πρωτόκολλο Ελέγχου και Προσαρμογής Λογικών Ζεύξεων (Logical link control and adaptation protocol, L2CAP) Προσαρμόζει τα πρωτόκολλα του ανωτέρου επιπέδου στο επίπεδο της βασικής ζώνης. Είναι επίσης υπεύθυνο για την πολυπλεξία των δεδομένων του ανώτερου επιπέδου. Το πρωτόκολλο L2CAP ενεργοποιείται μόνο για τις συνδέσεις ACL, υλοποιεί τις λογικές συνδέσεις πάνω από τις φυσικές ζεύξεις, καθορίζει τα κριτήρια ποιότητας των υπηρεσιών Qos για κάθε σύνδεση, πολυπλέκει πολλές λογικές συνδέσεις σε μία φυσική κλπ.

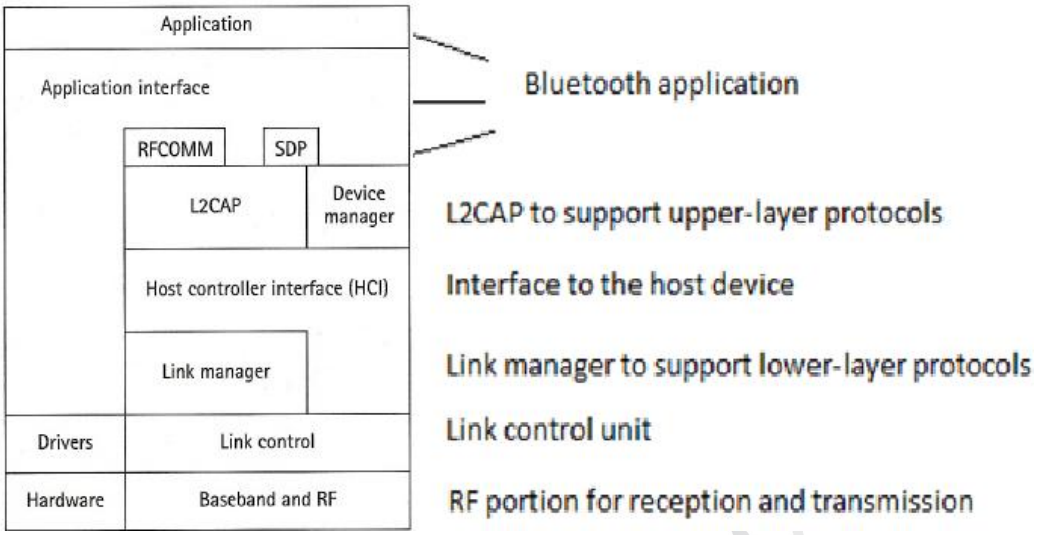
Πρωτόκολλο Ανακάλυψης Υπηρεσιών (Service discovery protocol, SDP) Για να μπορέσουν να συνδεθούν οι συσκευές πρέπει να υποστηρίζουν τις ίδιες υπηρεσίες. Το πρωτόκολλο αυτό "ρωτά" τις συσκευές για πιθανές διαθέσιμες υπηρεσίες και λειτουργεί χωρίς την εγκατάσταση σύνδεσης.

Αυτά είναι τα τέσσερα επίπεδα που περιέχουν τα πρωτόκολλα του πυρήνα της προδιαγραφής του Bluetooth. Εκτός απ' αυτά τα βασικά πρωτόκολλα του πυρήνα υπάρχουν και άλλα πρωτόκολλα τα οποία υποστηρίζουν τα προφίλ που υλοποιεί η προδιαγραφή του Bluetooth. Μερικά από τα πρωτοκολλά αυτά είναι:

Πρωτόκολλο Αντικατάστασης Καλωδίων (Radio Frequency Communication, RFCOMM) είναι ένα πρωτόκολλο επικοινωνίας σχεδιασμένο να προσομοιώνει τις σειριακές θύρες.

Πρωτόκολλο Τηλεφωνικού Ελέγχου (Telephony Control Specification Binary, TCS-Bin) είναι ένα πρωτόκολλο σχεδιασμένο για τον έλεγχο του τηλέφωνου

Πρωτόκολλο Ανταλλαγής Αντικειμένων (OBject EXchange Protocol, OBEX) είναι ένα πρωτόκολλο σχεδιασμένο για την ανταλλαγή αντικειμένων όπως είναι τα αρχεία, ο συγχρονισμός δεδομένων κλπ

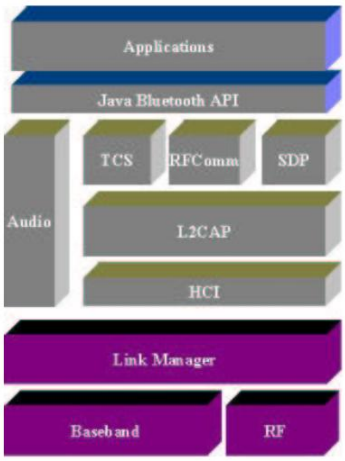


Εικόνα 1.12 Τα συστατικά στοιχεία της στοίβας του πρωτοκόλλου Bluetooth

Οι υπηρεσίες και τα προφίλ που υποστηρίζει η προδιαγραφή του Bluetooth υλοποιούνται από τα πρωτόκολλα της τεχνολογίας αυτής. [1], [2], [6]

1.6 Τεχνικά χαρακτηριστικά της τεχνολογίας του Bluetooth

Οι βασικότερες προδιαγραφές του Bluetooth αφορούν το φυσικό επίπεδο και το υποεπίπεδο MAC. Στο υποεπίπεδο MAC έχουν δημιουργηθεί διαφορετικά πρωτόκολλα για την υποστήριξη διαφορετικών εφαρμογών τα οποία ονομάζονται **προφίλ**.



Εικόνα 1.13 Οι βασικότερες προδιαγραφές του Bluetooth στο φυσικό επίπεδο και στο υποεπίπεδο MAC

Η ομάδα SIG του Bluetooth έχει παρουσιάσει τέτοιες παραμετροποιημένες εκδοχές του προτύπου για την υποστήριξη των εμπορικών εφαρμογών (π.χ. προφίλ

ασύρματου τηλεφώνου, προφίλ πρόσβασης σε LAN, προφίλ εκτύπωσης, φωτογραφίας, αυτοκινήτου κλπ). Κάθε προφίλ περιλαμβάνει πρότυπα για όλα τα επίπεδα και προσφέρει λύσεις για τη διασύνδεση με ετερογενή δίκτυα μεγαλύτερης κλίμακας.

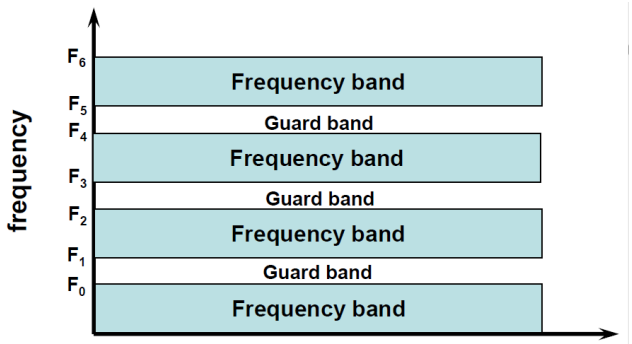
Η τεχνολογία Bluetooth είναι δομημένη σε δύο πακέτα προδιαγραφών. Το πρώτο πακέτο περιέχει τις προδιαγραφές του πυρήνα (βλ. παράγραφο 1.6.1), οι οποίες περιγράφουν τον τρόπο με τον οποίο η τεχνολογία λειτουργεί στα χαμηλότερα επίπεδα των ραδιοσυχνοτήτων, RF και της βασικής ζώνης, baseband (έως και το επίπεδο του Link Manager). Το δεύτερο πακέτο περιέχει τις προδιαγραφές του προφίλ χρήσης (βλ. παράγραφο 1.6.2) οι οποίες εστιάζουν στο πώς πρέπει να φτιαχτούν διαλειτουργικές συσκευές στηριζόμενες στις προδιαγραφές του πυρήνα. Σ' αυτό το κεφάλαιο εξετάζουμε τις προδιαγραφές του πυρήνα ενώ στα επόμενα κεφάλαια γίνεται παρουσίαση της εφαρμογής του bluetooth chat. [1] [2] [7] [8]

1.6.1 Προδιαγραφές Πυρήνα

Στις επόμενες παραγράφους αναφέρονται ο τρόπος μετάδοσης της πληροφορίας, η ζώνη των συχνοτήτων και ο τρόπο διάθεσης των συχνοτήτων αυτών στις συσκευές, το σχήμα της διαμόρφωσης, και η εμβέλεια της περιοχής κάλυψης της τεχνολογίας του Bluetooth. [2]

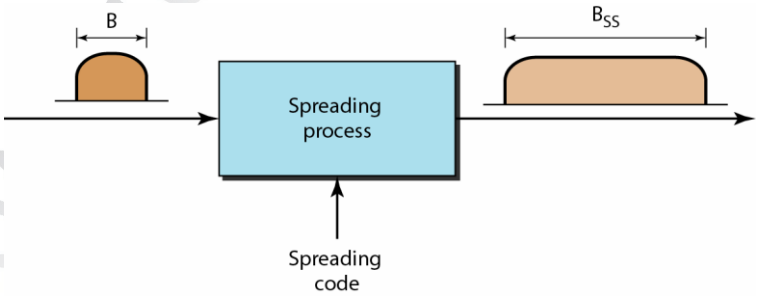
1.6.1.1 Ζώνη συχνοτήτων λειτουργίας και τρόπος μετάδοσης δεδομένων

Στο φυσικό επίπεδο (RF) πραγματοποιείται η μετάδοση του σήματος μέσω των ραδιοσυχνοτήτων. Το Bluetooth λειτουργεί στις συχνότητες 2400 - 2483,5 MHz συμπεριλαμβανομένων και των ζωνών προστασίας, guard bands. Τα κανάλια διαχωρίζονται μεταξύ τους με μικρές ζώνες αχρησιμοποίητων συχνοτήτων τις ζώνες ασφαλείας (guard bands) - για την αποφυγή επικάλυψης μεταξύ σημάτων διαφορετικών καναλιών.



Εικόνα 1.14 Ζώνες ασφαλείας και ζώνες συχνοτήτων [9]

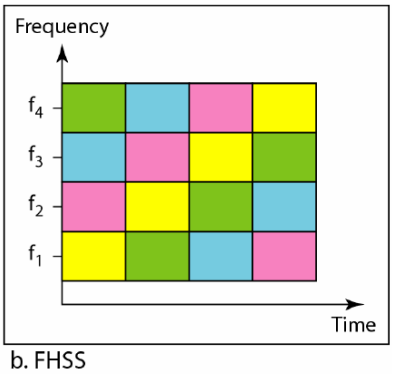
Η ζώνη των ραδιοσυχνοτήτων των 2.4 GHz έχει παγκοσμία καθιερωθεί για να χρησιμοποιείται από τη βιομηχανική, επιστημονική και ιατρική κοινότητα (ISM - Industrial, Scientific, Medicine) ελεύθερα, χωρίς αδειοδότηση. Η ζώνη των συχνοτήτων ISM έχει εύρος 83.5MHz και είναι διαθέσιμη στις συσκευές ανεξάρτητα από τη γεωγραφική θέση στην οποία βρίσκονται. Για να μπορέσουν να συνδεθούν δύο συσκευές είναι απαραίτητο να είναι συντονισμένες στην ίδια συχνότητα και να μοιράζονται το ίδιο κανάλι. Η μετάδοση της πληροφορίας στο ασύρματο μέσο γίνεται με την χρήση της μεθόδου της φασματικής διασποράς με την εναλλαγή συχνότητας FHSS (Frequency Hopping Spread Spectrum). Στη διασπορά φάσματος, Spread Spectrum, το εύρος ζώνης που ανατίθεται σε κάθε κόμβο πρέπει να είναι κατά πολύ μεγαλύτερο από αυτό που χρειάζεται (redundancy). Το αρχικό σήμα με τη διαδικασία της φασματικής διασποράς μετατρέπεται στο τελικό σήμα. Το τελικό σήμα είναι ανεξάρτητο από το αρχικό σήμα.



Εικόνα 1.15 Διασπορά Φάσματος – Spread Spectrum [9]

Η μέθοδος της φασματικής διασποράς με εναλλαγή της συχνότητας FHSS επιτυγχάνεται με «άλματα» από συχνότητα σε συχνότητα με ψευδοτυχαίο τρόπο. Ο δέκτης ακολουθεί τα ίδια συγχρονισμένα άλματα και με τον ίδιο ψευδοτυχαίο τρόπο για να κάνει την ανάκτηση του σήματος.

Η επιλογή της μεθόδου FHSS αλλά και της αμφίδρομης επικοινωνίας γίνεται για να περιοριστούν στο ελάχιστο οι παρεμβολές από παρεμφερείς συσκευές.



Εικόνα 1.16 Φασματική διασπορά με FHSS [9]

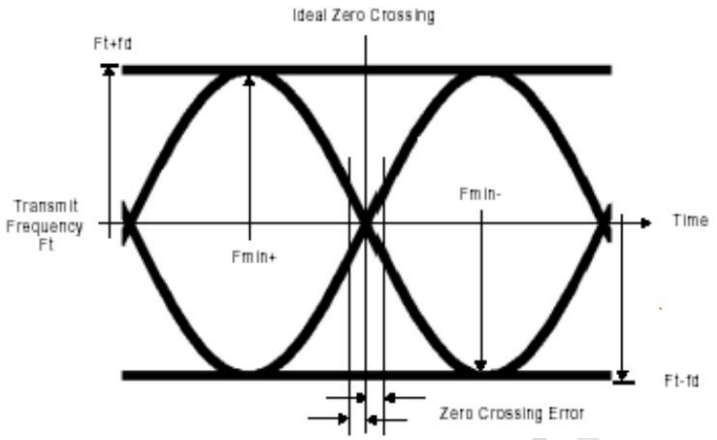
Η τεχνολογία του Bluetooth χωρίζει το φάσμα που του έχει διατεθεί (2.402GHz – 2.48GHz) σε 79 κανάλια. Το εύρος ζώνης του κάθε καναλιού είναι 1 MHz. Το πρώτο κανάλι ξεκινά από τα 2402 MHz και συνεχίζει με βήματα του 1 MHz έως τα 2480 MHz. Τα δεδομένα που πρέπει να αποσταλούν διαιρούνται σε πακέτα. Κάθε φορά που μεταδίδει ένα πακέτο δεδομένων αλλάζει η συχνότητα (εκτελεί 1600 εναλλαγές συχνότητας ανά δευτερόλεπτο) σύμφωνα με ένα προσυμφωνημένο pattern μεταξύ πομπού και δέκτη. Έτσι αποφεύγονται οι παρεμβολές από τα άλλα σήματα των γειτονικών συχνοτήτων και ο θόρυβος.

Οι προδιαγραφές της συγκεκριμένης τεχνολογίας υποστηρίζονται από την ομάδα SIG. Ένα πρόβλημα των προδιαγραφών του Bluetooth είναι ότι, λόγω της μετάδοσης στην ελεύθερη ζώνη συχνοτήτων των 2,4 GHz, οι συσκευές που το υποστηρίζουν αδυνατούν να χρησιμοποιήσουν ταυτόχρονα τα περισσότερα πρωτόκολλα της οικογένειας IEEE 802.11, γιατί εμφανίζονται σοβαρά προβλήματα παρεμβολών. [2], [9], [10]

1.6.1.2 Σχήματα διαμόρφωσης

Στην έκδοση v1.0 του πρωτοκόλλου του Bluetooth για την μετάδοση του σήματος χρησιμοποιούνταν η διαμόρφωση GFSK (Gaussian Frequency-Shift keying) ως το μόνο διαθέσιμο σχήμα διαμόρφωσης. Η νεότερη έκδοση του πρωτοκόλλου του δικτύου του Bluetooth 2.0 + EDR χρησιμοποιεί τη διαμόρφωση π/4- DQPSK

(Differential Quadrature Phase Shift Keying) και τη διαμόρφωση 8DPSK μεταξύ συμβατών συσκευών.

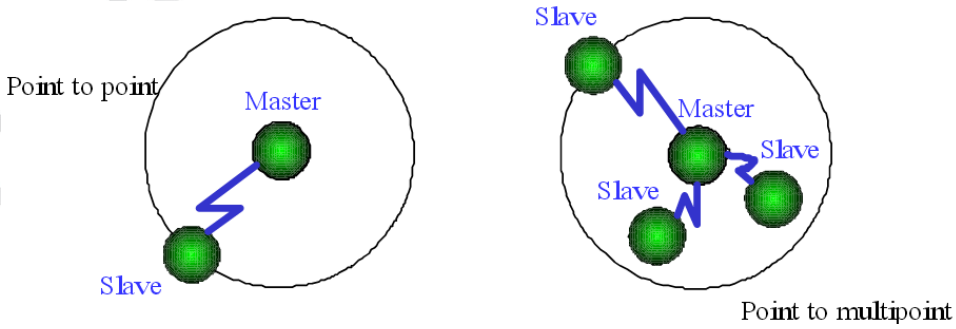


Εικόνα 1.17 Διαμόρφωση GFSK [Bluetooth SIG] [11]

Ο τύπος της διαμόρφωσης που χρησιμοποιείται για την μετάδοση του σήματος στο Φυσικό Επίπεδο καθορίζει την μορφή των πλαισίων που δημιουργούνται στο υποεπίπεδο MAC (βλ. παράγραφο 1.6.1.8 "Πλαίσια MAC και συγχρονισμός των κόμβων Master και Slave").

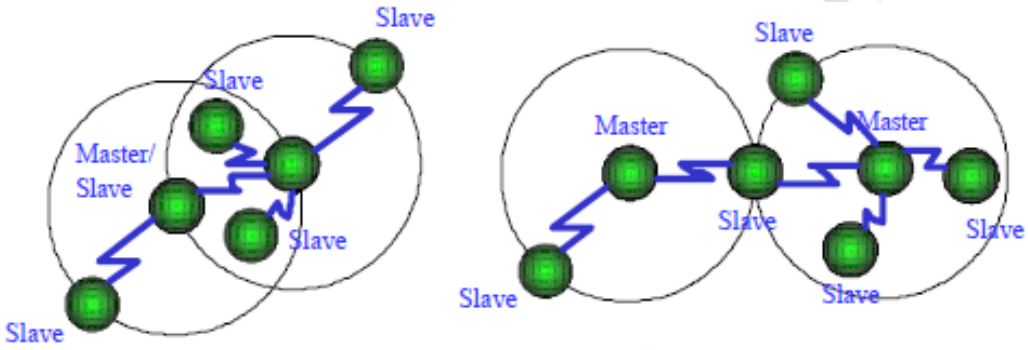
1.6.1.3 Δομή, τοπολογία δικτύου

Η βασική δομική μονάδα ενός δικτύου Bluetooth είναι το piconet. Όλοι οι κόμβοι του piconet (το πολύ μέχρι επτά συσκευές Slaves) μοιράζονται τον ίδιο κώδικα διασποράς και υπόκεινται στον έλεγχο ενός κοινού κύριου κόμβου ή Master. Τα Piconets μπορεί να είναι στατικά ή να σχηματίζονται δυναμικά καθώς οι συσκευές κινούνται μέσα και έξω από το φάσμα του ενός από το άλλο.



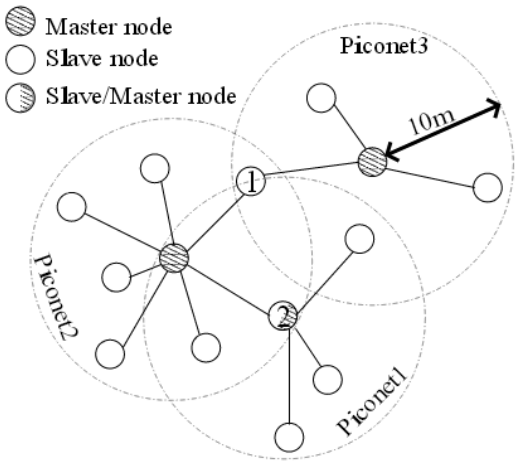
Εικόνα 1.18 α) Point to Point piconet, β) Point to Multipoint piconet

Το δίκτυο του Bluetooth επιτρέπει τις απευθείας συνδέσεις P2P (-point to point, σημείο με σημείο- ή -peer to peer, κόμβο με κόμβο-) από συσκευή προς συσκευή. Η εικόνα που ακολουθεί μας παρουσιάζει δύο σενάρια σύνδεσης συσκευών σε piconet δίκτυα. Δύο ή περισσότερα piconets μπορούν να βρίσκονται στον ίδιο χώρο, με τους κόμβους να μπορούν να συμμετέχουν ταυτόχρονα σε παραπάνω από ένα piconet και να επικοινωνούν μεταξύ τους δημιουργώντας ένα δίκτυο scatternet μεγαλύτερης κλίμακας. Οι προδιαγραφές του πυρήνα Bluetooth Core επιτρέπουν τη σύνδεση δύο ή περισσότερων piconets ώστε να σχηματίσουν ένα scatternet.

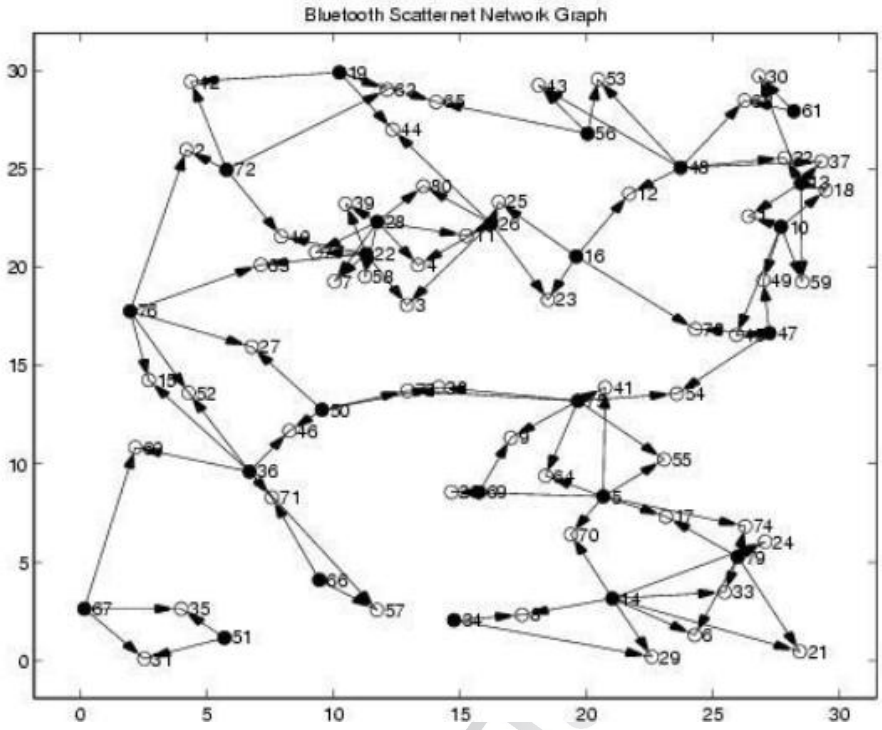


Εικόνα 1.19 α) Scatternet Master/Slave και β) Scatternet Slave/Slave

Μία συσκευή μπορεί ταυτόχρονα να συμμετέχει σε δύο piconets έχοντας το ρόλο του slave (Εικόνα 1.19α). Σπανιότερο αλλά όχι αδύνατο, όπως φαίνεται στις εικόνες 1.18 β) και 1.19, είναι μία συσκευή να συμμετέχει σε δύο δίκτυα διατηρώντας το ρόλο του Master στο ένα δίκτυο ενώ στο άλλο έχει το ρόλο του slave.



Εικόνα 1.20 Συμμετοχή κόμβου με διαφορετικούς ρόλους Master/Slave σε Scatternet (επίπεδο μοντέλο) [12]



Εικόνα 1.21 Γράφος δικτύου Bluetooth Scatternet (με τον αλγόριθμο BTSF)

Στην εικόνα 1.20 παρουσιάζεται ένας γράφος με τις συνδέσεις των κόμβων ενός δικτύου Bluetooth scatternet όπως αυτός έχει διαμορφωθεί με βάση τον αλγόριθμο BTSF. [2], [12]

1.6.1.4 Καταστάσεις κόμβων

Οι συνδεδεμένοι κόμβοι μπορούν να βρεθούν σε μία από τις επόμενες πέντε καταστάσεις οι οποίες ορίζονται στις προδιαγραφές της τεχνολογίας του Bluetooth:

Κατάσταση αναμονής (Stand by): δεν μεταφέρονται δεδομένα και το σήμα είναι απενεργοποιημένο (switched off).

Κατάσταση κράτησης (Hold): η συσκευή παύει να υποστηρίζει την ACL (Asynchronous Connectionless) μετάδοση για ένα ορισμένο χρονικό διάστημα, για να γίνει η αποδέσμευση του bandwidth λόγω συγχρονισμού (paging) κλπ. Η συσκευή Slave μπορεί να μεταφέρει μόνο φωνή, με σύνδεση SCO (Synchronous Connection Oriented), και επιπλέον έχει μειωμένη κατανάλωση ισχύος. Η συσκευή αφουγκράζεται το κανάλι για να καταλάβει αν θα πρέπει να βγει από την κατάσταση αυτή, διατηρεί τη

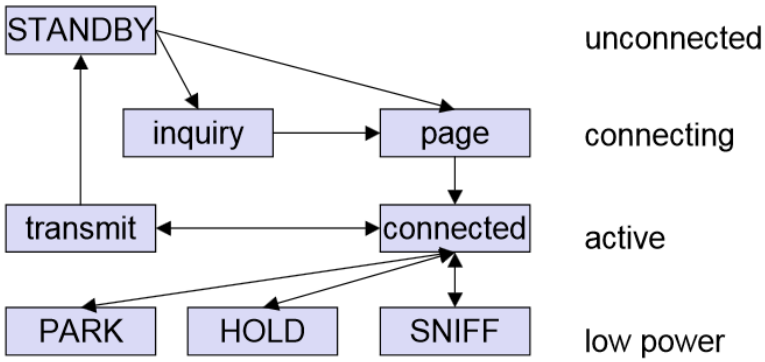
διεύθυνση της ως ενεργό μέλος (AM -Active Member) και πραγματοποιεί μία φορά το συγχρονισμό της με το Master.

Κατάσταση όσφρησης (Sniff): η slave συσκευή, ακούει σε συγκεκριμένες χρονοθυρίδες, αφουγκράζεται την διεύθυνση του ως ενεργού μέλους της AM για ένα προκαθορισμένο χρονικό διάστημα, χρονοθυρίδα (time-slot). Τότε κάνει την μετάδοση των δεδομένων της μόνο γι' αυτή τη χρονοθυρίδα και μετά απενεργοποιείται μέχρι την επόμενη όσφρηση της χρονοθυρίδας.

Κατάσταση στάθμευσης (Parked): η συσκευή Slave είναι μέλος του δικτύου αλλά δεν ακούει το κανάλι και δεν ανταλλάσσει δεδομένα. Η συσκευή που βρίσκεται στην κατάσταση αυτή παραμένει συγχρονισμένη με την συσκευή master και μπορεί να επιστρέψει στην ενεργή κατάσταση χωρίς να εκκινήσει την διαδικασία για την εγκατάσταση της σύνδεσης.

Ενεργή κατάσταση (Active): οι συσκευές Slave συμμετέχουν ενεργά στο δίκτυο ανταλλάσσοντας δεδομένα. Η κατάσταση λειτουργίας είναι ενεργή και η συσκευή Slave κατά την είσοδο της στην κατάσταση σύνδεσης συγχρονίζεται με το ρολόι του Master. Ο Master μεταδίδει ένα POLL μήνυμα προκειμένου να ελέγξει τη σύνδεση.

Παρακάτω δίνεται το διάγραμμα μετάβασης καταστάσεων στις οποίες μπορεί να βρεθεί ένας κόμβος που συμμετέχει σε ένα δίκτυο Bluetooth.



Διάγραμμα 1.1 Διάγραμμα μετάβασης καταστάσεων ενός κόμβου σε ένα δίκτυο Bluetooth

Οι κόμβοι Active, Sniff και Hold αναγνωρίζονται από διευθύνσεις 3-bit (έως 7 ενεργοί Slaves), ενώ οι κόμβοι Parked από διευθύνσεις 8-bit (έως 256 ανενεργοί Slaves). Οι καταστάσεις sniff, hold και parked είναι καταστάσεις χαμηλής ενέργειας.

Ακολουθεί ένας πίνακας με ενδεικτικές τιμές του CSR Bluecore 01 με τις κατανάλωσης ισχύος των καταστάσεων Park και Active.

Πίνακας 1.1 Σύγκριση της διαχεόμενης ισχύος στις καταστάσεις park και active

	Modes of Activity	
Product	Power Dissipation in Park Mode	Power Dissipation in Active Mode
CSR Bluecore 01	0.3 mW	0.6 – 135 mW

Στην επόμενη παράγραφο παρουσιάζεται ο τρόπος λειτουργίας του δικτύου με βάση τις καταστάσεις που περιγράψαμε παραπάνω. [1] [11]

1.6.1.5 Τρόπος λειτουργίας του δικτύου Bluetooth

Τα δίκτυα Piconets μπορεί να είναι στατικά ή να σχηματίζονται δυναμικά καθώς οι συσκευές κινούνται μέσα και έξω από το φάσμα του ενός ή του άλλου δικτύου. Μια συσκευή που δεν συμμετέχει σε κάποιο δίκτυο piconet βρίσκεται σε κατάσταση αναμονής (προεπιλεγμένη κατάσταση χαμηλής κατανάλωσης). Η συσκευή αφήνει την κατάσταση αναμονής για την αποστολή ή τη λήψη ενός μηνύματος inquiry ή μιας εντολής συγχρονισμού page. Η αποστολή ενός μηνύματος inquiry γίνεται για να ζητηθούν πληροφορίες όταν η διεύθυνση μιας targeted συσκευής είναι άγνωστη. Μία ερώτηση πληροφοριών θα πρέπει να ακολουθείται από μια εντολή συγχρονισμού του ρολογιού, page, προκειμένου η συσκευή-αποστολέας να συντονιστεί με τη συσκευή που θα συνδεθεί. Μια εντολή page περιέχει ένα συγκεκριμένο Device-Access-Code που χρησιμοποιείται για τη σύνδεση μιας απομακρυσμένης συσκευής. Όταν η απομακρυσμένη συσκευή ανταποκριθεί, και οι δύο συσκευές εισέρχονται σε κατάσταση σύνδεσης, η συσκευή που κάνει την έναρξη της σύνδεσης αναλαμβάνει το ρόλο της συσκευής master και η ανταποκρινόμενη συσκευή ενεργεί ως slave.

Αφού περιέλθει σε κατάσταση σύνδεσης, η συσκευή του slave θα συγχρονιστεί με το ρολόι της συσκευής του master και θα ενημερωθεί με το σωστό πρότυπο εναλλαγής της συχνότητα. Σε αυτό το σημείο, οι διαχειριστές της σύνδεσης ανταλλάσουν εντολές για την απόκτηση των πληροφοριών της συσκευής προκειμένου να δημιουργηθεί η σύνδεση.

Ο κόμβος master στέλνει μηνύματα για να κρατήσει συγχρονισμένο το δίκτυο piconet. Οι συσκευές slaves "ακούν" κάθε χρονοθυρίδα που εκπέμπει η συσκευή master ούτως ώστε να συγχρονιστούν με αυτή.

Σε κάθε ενεργή συσκευή slave εκχωρείτε μια διεύθυνση ενεργού μέλους η AM_ADDR και συμμετέχει πλέον ως ενεργό μέλος του piconet. Η συσκευή slave "ακούει" όλες τις χρονοθυρίδες του master για να πληροφορηθεί αν έχει διευθυνσιοδοτηθεί από τη συσκευή master. Για τους κόμβους slave υπάρχουν τρεις καταστάσεις χαμηλής ενέργειας: η κατάσταση sniff, η κατάσταση hold, και η κατάσταση park.

Ο κόμβος master μπορεί να μεταδώσει σε συσκευές που βρίσκονται στην κατάσταση sniff κατά τη διάρκεια συγκεκριμένων χρονοθυρίδων. Για το λόγο αυτό οι συσκευές αφουγκράζονται ή ακούν μόνο κατά τη διάρκεια αυτών των ειδικών χρονοθυρίδων και "κοιμούνται" τον υπόλοιπο χρόνο.

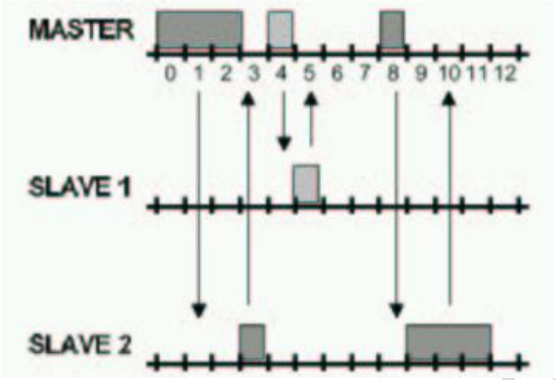
Ένας κόμβος slave που βρίσκεται σε κατάσταση αναμονής (hold) δεν λαμβάνει κανένα ασύγχρονο πακέτο και "ακούει" μόνο για να μπορέσει αν θα επανέλθει ξανά στην ενεργή κατάσταση.

Μία συσκευή slave που βρίσκεται σε κατάσταση park σταματά να "ακούει" και εγκαταλείπει την διεύθυνση του ενεργού μέλους που της είχε εκχωρηθεί.

Όλες οι συσκευές συγχρονίζονται-μοιράζονται το ρολόι της συσκευής master. Η τελευταία μοιράζει στους σταθμούς slave την πρόσβαση στο κοινό μέσο (τον ελεύθερο χώρο) με τη μέθοδο TDMA/TDD (Time division multiple access/Time Division Duplex), στην οποία ο χρόνος διαμερίζεται σε χρονοθυρίδες. Η ανταλλαγή των πακέτων πραγματοποιείται με βάση το ρολόι της master συσκευής, το οποίο χτυπά στο διάστημα των 312,5 μs. Δύο χτύποι του ρολογιού συνθέτουν μια χρονοθυρίδα των 625 μs, δύο τέτοιες χρονοθυρίδες συνθέτουν ένα ζεύγος χρονοθυρίδων των 1250 μs.

Στην απλή περίπτωση των πακέτων single-slot ο κόμβος master μεταδίδει δεδομένα στις ζυγές χρονοθυρίδες και λαμβάνει σε μονά slots. Ο κόμβος slave, αντίθετα, λαμβάνει στα ζυγά slots και μεταδίδει στα μονά slots. Τα πακέτα μπορεί να είναι μεγέθους 1, 3 ή 5 slots, αλλά σε όλες τις περιπτώσεις η εκπομπή της συσκευής

master θα ξεκινήσει σε ζυγά slots και η μετάδοση της συσκευής slave σε μονές χρονοθυρίδες.



Εικόνα 1.22 Frequency Hopping - Time Division Duplex Bluetooth MAC

Η συσκευή Master εκπέμπει στις περιττές χρονοθυρίδες και οι συσκευές Slaves στις άρτιες (εναλλάξ), κάθε κόμβος που θέλει να εκπέμψει λαμβάνει περιοδικά από τον κόμβο Master το δικαίωμα μετάδοσης σε 1, 3 ή 5 συνεχόμενες χρονοθυρίδες και κατά τη διάρκεια εκπομπής ενός πλαισίου δεν γίνεται εναλλαγή της συχνότητας. Οι κόμβοι Slaves μεταδίδουν μόνο στον κόμβο Master, ο οποίος στέλνει τα πλαίσιά που παρέλαβε από τις συσκευές slave προς τον τελικό παραλήπτη.

Οι συσκευές μπορούν με κοινή συμφωνία, να αλλάξουν ρόλους και ο σκλάβος μπορεί να γίνει η συσκευή master (για παράδειγμα, μια συσκευή ασύρματων ακουστικών ξεκινά μια σύνδεση με ένα τηλέφωνο τα ακουστικά είναι συσκευή η οποία αναγκαστικά ξεκινά τη σύνδεση ως master συσκευή, αλλά μπορεί στη συνέχεια να αλλάξει ρολό και να ‘προτιμά’ το ρόλο της slave συσκευής).

Τα δεδομένα μπορούν να μεταφερθούν μεταξύ της συσκευής master και μιας άλλης συσκευής. Η μετάβαση από τη μία συσκευή στην άλλη γίνεται σύμφωνα με το μοντέλο της κυκλικής διαδοχής, round-robin. Η συσκευή master είναι αυτή που επιλέγει ποια συσκευή slave θα διευθυνσιοδοτήσει, ενώ μία slave συσκευή "ακούει" όλα τα slots. Το να είναι μία συσκευή master έχει λιγότερο βάρος από το να είναι συσκευή slave. Είναι δυνατόν να είναι μία master συσκευή με επτά υποτελείς συσκευές slave. Είναι όμως δύσκολο μια slave συσκευή να είναι υποτελείς κάτω από πολλές master συσκευές. [2], [12]

1.6.1.6 Τύποι καναλιών

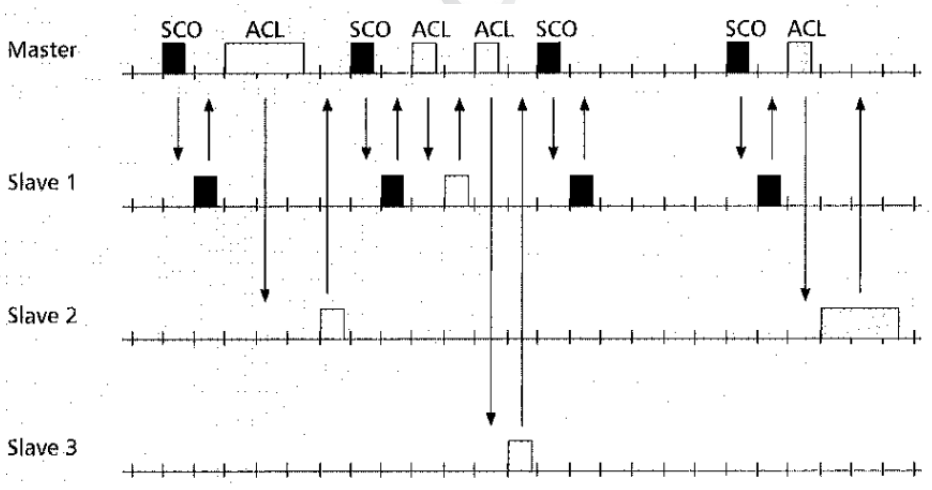
Τα δεδομένα κατακερματίζονται και ενθυλακώνονται μέσα στα πακέτα ή πλαίσια (frames) του πρωτοκόλλου του Bluetooth (παράγραφος 1.6.1.8 "Πλαίσια MAC και συγχρονισμός των κόμβων Master και Slave") πριν γίνει η μετάδοση τους στο φυσικό επίπεδο μέσω της ζεύξης. Στο επίπεδο Mac η Μονάδα Ελέγχου Ζεύξης (Link controller (baseband)) είναι υπεύθυνη για τον έλεγχο και την αποστολή και τη λήψη των πακέτων δεδομένων μέσω της ασύρματης ζεύξης. Η Μονάδα Ελέγχου Ζεύξης του Bluetooth παρέχει δύο τύπους ζεύξεων για τη μετάδοση των δεδομένων, την Ασύγχρονη ζεύξη χωρίς σύνδεση ACL (Asynchronous Connectionless) και την Σύγχρονη ζεύξη SCO (Synchronous Connection Oriented). Η τεχνολογία του Bluetooth διαθέτει μία ασύγχρονη σύνδεση για την διάδοση των απλών δεδομένων και τρεις σύγχρονες συνδέσεις που τυπικά χρησιμοποιούνται για τη μεταφορά φωνής.

Σύγχρονες ζεύξεις SCO (Synchronous Connection Oriented). Η σύνδεση SCO παρέχει ένα συμμετρικό κανάλι επικοινωνίας (δηλαδή το κανάλι έχει την ίδια ρυθμαπόδοση στην αποστολή και τη λήψη του) μεταξύ των συσκευών master και slave για την μετάδοση των δεδομένων (συνήθως φωνής). Η ζεύξη SCO ακολουθεί ως προς την υλοποίηση της την τεχνική της Μεταγωγής Κυκλώματος (circuit switching). Η master συσκευή μπορεί να υποστηρίξει μέχρι τρεις σύγχρονες ζεύξεις ενώ η συσκευή slave μπορεί να υποστηρίξει μέχρι τρεις ζεύξεις με την ίδια συσκευή master (Εικόνα 1.23) και μέχρι δύο ζεύξεις με διαφορετικές master συσκευές. Η ζεύξη SCO δημιουργείται ύστερα από απαίτηση της κύριας συσκευής προς την εξαρτημένη συσκευή και με την μεσολάβηση του Διαχειριστή Ζεύξης (Link Manager). Κάθε κόμβος μπορεί να δεσμεύσει μόνο μία χρονοθυρίδα, Οι μεταδόσεις της σύνδεσης αυτής εξαρτώνται χρονικά και για το λόγο αυτό δεν γίνονται επανεκπομπές ή επιβεβαιώσεις των αποστολών των πακέτων αλλά χρησιμοποιούν αλγορίθμους ανίχνευσης και διόρθωσης σφαλμάτων (FEC).

Ασύγχρονες ζεύξεις χωρίς σύνδεση ACL (Asynchronous Connectionless). Η ζεύξη ACL, σε αντίθεση με την ζεύξη SCO ακολουθεί ως προς την υλοποίηση της την τεχνική της Μεταγωγής Πακέτων (Packet

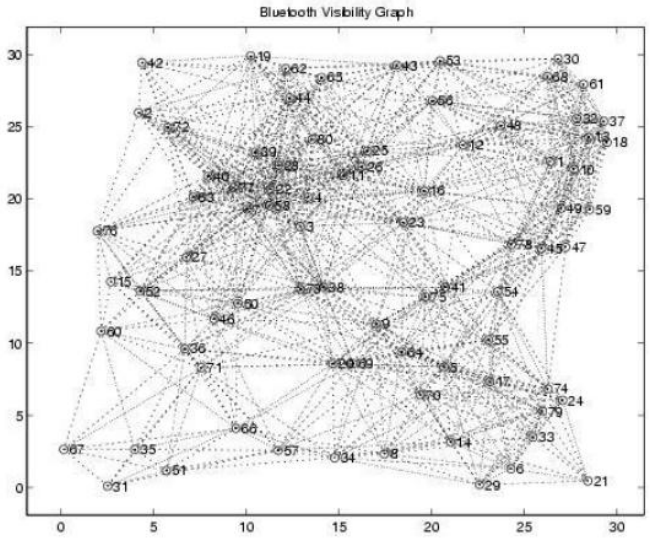
switching). Η ζεύξη ACL δημιουργείται μεταξύ δύο συσκευών με την έναρξη της επικοινωνίας. Η master συσκευή μπορεί να υποστηρίξει μία μόνο σύνδεση με την ίδια συσκευή slave, ενώ μπορεί να εξυπηρετεί πολλές τέτοιες συνδέσεις με διαφορετικές όμως συσκευές. Η μετάδοση των δεδομένων γίνεται με τη μορφή πακέτων τα οποία μπορεί να είναι broadcast ή διευθηνσιοδοτούμενα προς μια συσκευή. Τυπικά χρησιμοποιούνται για τη μετάδοση δεδομένων, κάθε κόμβος μπορεί να δεσμεύσει 1, 3 ή 5 χρονοθυρίδες για την εκπομπή ενός πλαισίου. Η διαχείριση της κίνησης σε ένα κανάλι ACL γίνεται από την κύρια συσκευή. Τέλος σε αντίθεση με την ζεύξη SCO τα πακέτα της ACL ζεύξη αναμεταδίδονται για να εξασφαλίσουν την ορθή αποστολή τους.

Η τεχνολογία του Bluetooth διαθέτει μία ασύγχρονη σύνδεση για την διάδοση των απλών δεδομένων και τρεις σύγχρονες συνδέσεις που τυπικά χρησιμοποιούνται για τη μεταφορά φωνής.



Εικόνα 1.23 Ένα παράδειγμα μίξης σύγχρονων SCO συνδέσεων και ασύγχρονων ACL συνδέσεων σε ένα piconet

Τα δεδομένα στην τεχνολογία του Bluetooth μεταδίδονται σε πακέτα. Στην παραπάνω εικόνα δίνεται ένα παράδειγμα ανταλλαγής πακέτων μέσω σύγχρονων SCO καναλιών και ασύγχρονων ACL καναλιών σε ένα piconet δίκτυο. Με αυτό τον τρόπο γίνεται η επικοινωνία του master κόμβου με τους δύο slave κόμβους για να επιτευχθεί η μετάδοση της φωνής μέσω των σύγχρονων συνδέσεων και η ανταλλαγή δεδομένων μέσω της ασύγχρονης σύνδεσης.



Εικόνα 1.24 Γράφημα συνδεσιμότητας ενός δικτύου Bluetooth Scatternet

Στην εικόνα 23 παρουσιάζεται ένα γράφημα της συνδεσιμότητας ενός δικτύου Bluetooth scatternet στο οποίο απεικονίζεται η πυκνότητα των συνδέσεων. Στην ενότητα που ακολουθεί περιγράφονται οι διαδικασίες που ακολουθούνται για να δημιουργηθεί η σύνδεση των κόμβων σε ένα δίκτυο Bluetooth Scatternet. [2], [13]

1.6.1.7 Η δημιουργία του δικτύου και οι μεταβολές της δικτυακής τοπολογίας

Η δημιουργία ενός δικτύου Bluetooth ξεκινάει με την διαδικασία της 'Ανίχνευσης'. Σε ένα άγνωστο περιβάλλον, ένας κόμβος που επιθυμεί να γίνει κύριος κόμβος (διαδικασία Inquiry) ψάχνει για να εντοπίσει τις συσκευές που βρίσκονται μέσα στην περιοχή της εμβέλειας του και θέλουν να συνδεθούν μ' αυτόν, έτσι σχηματίζεται ένα piconet δίκτυο. Ο κύριος κόμβος είναι υπεύθυνος για τις μεταβολές της δικτυακής τοπολογίας όπως είναι οι εισαγωγές και οι αποχωρήσεις κόμβων και ο συγχρονισμός τους (διαδικασία Paging). Οι ενέργειες για τη διαδικασία της ανίχνευσης και της σύνδεσης των συσκευών σε ένα δίκτυο Bluetooth ακολουθούν την εξής σειρά:

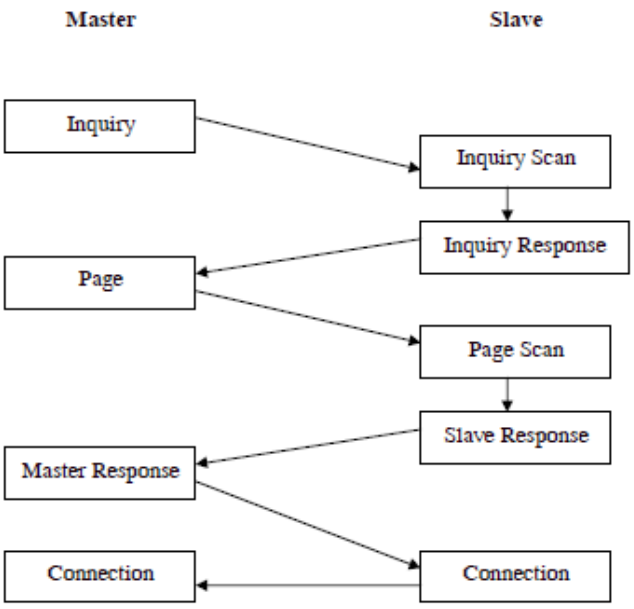
Ο εν δυνάμει Master κόμβος εκκινεί τη διαδικασία ανίχνευσης πιθανών Slaves κόμβων εκπέμποντας ένα Inquiry μήνυμα που περιέχει τον κώδικα πρόσβασης IAC (Inquiry Access Code).

Κάθε κόμβος που λαμβάνει ένα τέτοιο μήνυμα απαντά με την αποστολή ενός πλαισίου που περιέχει τη διεύθυνση του και τις πληροφορίες που απαιτούνται για τον συγχρονισμό του με το ρολόι του κύριου κόμβου και περιμένει την παραλαβή ενός μηνύματος Page.

Ο Master κόμβος λαμβάνει αυτά τα πλαίσια των Slaves κόμβων, χρησιμοποιεί τις διευθύνσεις των τελευταίων για να υπολογίσει τον κώδικα διασποράς του Frequency Hopping και αποστέλλει στους Slaves κόμβους που εντοπίστηκαν ένα μήνυμα Page που περιέχει τον κώδικα πρόσβασης συσκευής DAC (Device Access Code).

Οι Slaves κόμβοι απαντούν με τον κώδικα IAC (ένα είδος πιστοποίησης) και ο Master κόμβος τους στέλνει τον κώδικα της διασποράς.

Οι Slaves κόμβοι επιβεβαιώνουν τη λήψη των μηνυμάτων, συνδέονται με τον κύριο κόμβο και σχηματίζεται το piconet.[2]



Εικόνα 1.25 Τυπική διαδικασία ανίχνευσης και σύνδεσης ενός δικτύου Bluetooth

1.6.1.8 Πλαίσια MAC και συγχρονισμός των κόμβων Master και Slave

Τα δεδομένα που εισέρχονται στο Επίπεδο Σύνδεσης Δεδομένων (Data Link Layer του μοντέλου του OSI) από το υψηλότερο Επίπεδο Δικτυού (Network Layer) οργανώνονται σε πλαίσια (frames). Για να μεταδοθούν τα δεδομένα (data ή φωνής)

τεμαχίζονται σε μικρότερα πακέτα και ενθυλακώνονται μέσα σε πλαίσια του υποεπιπέδου LLC (Logical Link Control). Το κάθε πλαίσιο του LLC έχει τη δική του επικεφαλίδα η οποία μεταφέρει στοιχεία για τη λογική σύνδεση του LLC και το μήκος του ωφέλιμου φορτίου του πλαισίου. Τα πακέτα που προέρχονται από το LLC προωθούνται στο κατώτερο υποεπίπεδο MAC όπου γίνεται η ενθυλάκωση τους μέσα σε πλαίσια του επιπέδου MAC με την προσθήκη των πεδίων Access code και Header (Εικόνα 1.26). Τα υποεπίπεδα LLC και MAC ανήκουν στο Επίπεδο Σύνδεσης Δεδομένων το οποίο μας παρέχει την αξιόπιστη μεταφορά των δεδομένων πάνω στα φυσικά μέσα. Εφόσον δημιουργηθεί το πλαίσιο MAC διαβιβάζεται στο αμέσως κατώτερο επίπεδο, το Φυσικό Επίπεδο (Physical Layer), για την αποστολή του πακέτου μέσω των ραδιοσυχνοτήτων. Όπως αναφέρθηκε στην παράγραφο 1.6.1.6 «1.6.1.6Τύποι καναλιών» η επικοινωνία των δικτύων του Bluetooth γίνεται με την αποστολή πακέτων μέσω των ζεύξεων SCO (Synchronous Connection Oriented) και ACL (Asynchronous Connectionless). Για τις ζεύξεις SCO υπάρχουν τέσσερις διαφορετικοί τύποι πλαισίων και για τις ζεύξεις ACL έξι τύποι.

Όλοι οι τύποι πλαισίων έχουν ένα κώδικα πρόσβασης (Access code). Υπάρχουν τρεις τύποι κώδικα πρόσβασης: ο IAC (Inquiry Access Code), ο DAC (Device Access Code) και ο CAC (Channel Access Code). Ο κώδικας αυτός χρησιμοποιείται για τον συγχρονισμό του Master κόμβου με τους κόμβους Slaves που απαιτείται κατά τη διαδικασία της εναλλαγής των συχνοτήτων του FHSS και την ταυτοποίηση των πακέτων στο Φυσικό Επίπεδο. Ο κώδικα πρόσβασης ακόμα χρησιμοποιείται για τις διαδικασίες inquiry, paging και park της λειτουργίας του πρωτοκόλλου του Bluetooth.

Η επικεφαλίδα του πλαισίου (Header - επίπεδο σύνδεσης ή υποεπίπεδο MAC) περιέχει τις διευθύνσεις της πηγής και του προορισμού (συσκευή προέλευσης και προορισμού αντίστοιχα) καθώς και πληροφορίες για τον τύπο του πακέτου (15 τύποι πακέτων) και το μήκος του ωφέλιμου φορτίου.

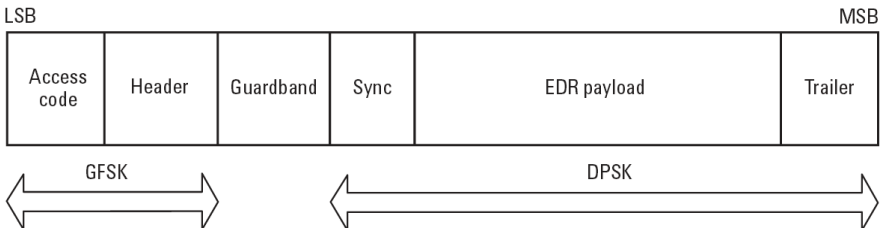
Το ωφέλιμο φορτίο (payload) μπορεί να περιέχει απλά δεδομένα, δεδομένα φωνής ή πληροφορίες έλεγχου. Ο τύπος της ζεύξης καθορίζει τη μορφή του ωφέλιμου φορτίου. Το ωφέλιμο φορτίο μπορεί να περιέχει επιπρόσθετες πληροφορίες για τον εντοπισμό σφαλμάτων. Ο κώδικα Ελέγχου Κυκλικού Πλεονασμού CRC (Cyclic Redundancy Check) ο οποίος προστίθεται στο πακέτο για να προσδιορίσει αν το

ωφέλιμο φορτίο είναι σωστό ή όχι χρησιμοποιείται για τον εντοπισμό σφαλμάτων των πακέτων των απλών δεδομένων. Ενώ για τον εντοπισμό και τη διόρθωση των σφαλμάτων στα δεδομένων της φωνής χρησιμοποιείται ο κώδικας FEC (Forward Error Correction).

Η μορφή του πλαισίου είναι ανάλογη με τη διαμόρφωση (βλ. παράγραφο 1.6.1.2 "Σχήματα διαμόρφωσης") που ακολουθείται για την μετάδοση του πακέτου. Στην περίπτωση που η διαμόρφωση είναι τύπου GFSK (Gaussian Frequency-Shift keying) τότε η δομή του πλαισίου MAC είναι αυτή που απεικονίζεται στην Εικόνα 1.26 και περιέχει τα πεδία του κώδικα πρόσβασης, της επικεφαλίδας και του ωφέλιμου φορτίου.

Εικόνα 1.26 Βασική μορφή πλαισίου MAC του δικτύου Bluetooth (διαμόρφωση GFSK) [5]

Το πρωτόκολλο του Bluetooth 2.0 + EDR, χρησιμοποιεί τη διαμόρφωση DPSK η οποία παρουσιάζει διαφορά στην μετατόπιση της φάσης. Αυτό έχει ως αποτέλεσμα τη διαφοροποίηση των πλαισίων MAC από τη βασική μορφή τους. Η μορφή του πλαισίου MAC της έκδοσης Bluetooth 2.0 + EDR παρουσιάζεται στην εικόνα που ακολουθεί και περιέχει εκτός από τα πεδία του κώδικα πρόσβασης, της επικεφαλίδας και του ωφέλιμου φορτίου, τη ζώνη ασφαλείας ή ζώνη φύλαξης (guard band), το πεδίο Sync το οποίο περιέχει μια ακολουθία συμβόλων που χρησιμοποιείται για το συγχρονισμό των συμβόλων και της φάσης ανάλογα με τον τύπο (DQPSK ή 8DPSK) της διαμόρφωσης που χρησιμοποιείται στα πακέτα EDR και τέλος το πεδίο της ουράς (trailer) που περιέχει το αποτέλεσμα του κώδικα CRC.



Εικόνα 1.27 Τύπος πλαισίου Bluetooth [5]

Όπως αναφέρεται στην ιστοσελίδα της wikipedia: "Οι διαφορετικοί τύποι πλαισίων MAC του Bluetooth διαφοροποιούνται στο ωφέλιμο φορτίο: οι τύποι SCO είναι ο High Quality Voice 1, στον οποίον τα 2/3 του μήκους του ωφέλιμου φορτίου του πλαισίου είναι κώδικας FEC (με αποτέλεσμα μικρή απώλεια πλαισίων αλλά μειωμένο ρυθμό μετάδοσης δεδομένων) και τα υπόλοιπα πληροφορίες φωνής, ο High Quality Voice 2, στον οποίον το 1/3 του μήκους του ωφέλιμου φορτίου του πλαισίου είναι κώδικας FEC και τα υπόλοιπα πληροφορίες φωνής, ο High Quality Voice 3, στο ωφέλιμο φορτίο του οποίου υπάρχουν μόνο πληροφορίες φωνής, και ο Data-Voice, ο οποίος έχει κώδικα FEC ίσο με το 1/3 του συνολικού μήκους του πλαισίου, κώδικα CRC, την κεφαλίδα LLC που προαναφέρθηκε, πληροφορίες φωνής και πληροφορίες δεδομένων. Οι τύποι ACL από την άλλη υποστηρίζουν διαφορετικούς ρυθμούς μετάδοσης και απώλειας πακέτων (όπου υψηλός ρυθμός μετάδοσης σημαίνει μικρό μήκος κώδικα FEC και άρα υψηλή απώλεια πακέτων), μεταφέρουν μόνο πληροφορίες δεδομένων και διακρίνονται σε Data Medium Rate (DM) 1,3 και 5, καθώς και Data High Rate (DH) 1,3 και 5." Πηγή: ιστοσελίδα wikipedia http://el.wikipedia.org/wiki/Bluetooth

Πίνακας 1.2 DM και DH πακέτα

Packet Type	Max. Payload /bytes	FEC	Max. Symmetric Data Rate /kbps	Asymmetric Data Rate Forward /kbps	Asymmetric Data Rate Reverse /kbps
DM1	17	2/3	108.8	108.8	108.8
DH1	27	None	172.8	172.8	172.8
DM3	121	2/3	258.1	387.2	54.4
DH3	183	None	390.4	585.6	86.4
DM5	224	2/3	286.7	477.8	36.3
DH5	339	None	433.9	723.2	57.6

Στον παραπάνω πίνακα φαίνετε η διαφορά της χωρητικότητας των ωφέλιμων φορτίων των διαφορετικών τύπων των πακέτων. [1] [2] [5]

1.6.1.9 Εμβέλεια μετάδοσης και κλάσεις ισχύος

Όπως αναφέρθηκε προηγουμένως στην παράγραφο 1.6.1.1 «**Ζώνη συχνοτήτων λειτουργίας και τρόπος μετάδοσης** δεδομένων» το Bluetooth λόγω της μεθόδου της φασματικής διασποράς με την εναλλαγή της συχνότητας FHSS (Frequency-Hopping Spread Spectrum) που χρησιμοποιεί εφαρμόζει την τακτική εναλλαγής της συχνότητας η οποία καθορίζεται ψευδοτυχαία από τον κύριο κόμβο,

και προδιαγράφει τρία επίπεδα ισχύος της εκπομπής από τα οποία εξαρτάται και η εμβέλεια της μετάδοσης (πάντα μικρότερη των 10 μέτρων σε PAN).

Η εκπεμπόμενη ισχύς που χρησιμοποιείται για την ζεύξη ρυθμίζεται ανάλογα με την απόσταση στην οποία βρίσκονται οι συσκευές του πομπού και του δέκτη έτσι ώστε να χρησιμοποιείται η ελάχιστη εκπεμπόμενη ισχύς που απαιτείται για την ζεύξη αυτή. Η αυξομείωση της εκπεμπόμενης ισχύος έχει ως αποτέλεσμα την χαμηλή κατανάλωση ενέργειας στις μπαταρίες των συσκευών. Η ρύθμιση της εκπεμπόμενης ισχύς καθορίζεται από την συσκευή του κύριου κόμβου. Στην περίπτωση που η συσκευή του δέκτη βρίσκεται σε απόσταση μικρότερη από την μέγιστη δυνατή απόσταση που μπορεί να καλύψει η συσκευή του πομπού συμφώνα με τις προδιαγραφές της εμβέλεια της συσκευής, ο πομπός μειώνει την ισχύς του σήματος του ανάλογα με την απόσταση που έχει από των δέκτη για την μετάδοση του σήματος. Ένα άλλο πλεονέκτημα από την αυξομείωση της εκπεμπόμενης ισχύς μεταξύ των συσκευών του πομπού και του δέκτη είναι η μείωση των παρεμβολών στο δίκτυο του Bluetooth. Το πραγματικό εύρος της απόστασης ποικίλει ανάλογα με τις συνθήκες της διάδοσης, το υλικό της κάλυψης, τη διαμόρφωση της κεραίας και την κατάστασης της μπαταρίας. Οι περισσότερες εφαρμογές Bluetooth έχουν δημιουργηθεί με γνώμονα τη χρήση τους σε εσωτερικό περιβάλλον, όπου η εξασθένηση του σήματος λόγω των τοίχων και των αντανακλάσεων των σημάτων μειώνει κατά πολύ την απόσταση από αυτή που έχει καθοριστεί από την line-of-sight απόσταση που καθορίζουν οι προδιαγραφές του δικτύου του Bluetooth.

Πίνακας 1.3 Κλάσεις Ισχύος Bluetooth

Κλάση Ενέργειας	Απόσταση (m)	Μεγίστη επιτρεπόμενη ισχύς	Μέση εκπεμπόμενη ισχύς	Ελάχιστη εκπεμπόμενη ισχύς	Έλεγχος ισχύος
Κλάση 1	~100+	100 mW (20 dBm)	N/A	100 mW (20 dBm)	Pmin<+4 dBm to Pmax Optional: Pmin[2)] to Pmax
Κλάση 2	~10	2,5 mW (4 dBm)	1 mW (0 dBm)	2,5 mW (4 dBm)	Optional: Pmin[2)] to Pmax
Κλάση 3	~1	1 mW (0 dBm)	N/A	1 mW (0 dBm)	Optional: Pmin[2)] to Pmax

Η κάθε συσκευή χαρακτηρίζεται ανάλογα με την εκπεμπόμενη ισχύς που υποστηρίζει σύμφωνα με τις προδιαγραφές του πρωτοκόλλου του Bluetooth. Θεωρώντας ως κριτήριο την εκπεμπόμενη ισχύ (που προϋποθέτει την ανάλογη μπαταρία της συσκευή) η οποία καθορίζει την απόσταση μετάδοσης διακρίνονται

τρεις κλάσεις συσκευών σύμφωνα με τις προδιαγραφές του πρωτοκόλλου του Bluetooth. Οι κλάσεις αυτές παρουσιάζονται στον Πίνακας 1.3 Κλάσεις Ισχύος BluetoothΠίνακας 1.3.

Η ακτίνα λειτουργίας του Bluetooth ξεκινάει από το 1 m για τις συσκευές που ανήκουν στην 1η κλάση και φτάνει μέχρι τα 100 m για τις συσκευές της 3ης κλάσης. Η πιο διαδεδομένη κατηγορία συσκευών των προδιαγραφών του δικτύου του Bluetooth είναι αυτή που χρησιμοποιείται για την υλοποίηση των δικτύων WPAN που περιλαμβάνει τις συσκευές της 2ης κλάσης η οποία έχει ακτίνα δράσης περίπου τα 10 m. Αξίζει να σημειωθεί ότι δεν απαιτείται η οπτική επαφή του πομπού και του δέκτη για την επίτευξη της ζεύξης.

Πίνακας 1.4 Οι εκδόσεις του πρωτοκόλλου του Bluetooth [2]

Version	Data rate	Maximum application throughput
Version 1.2	1 Mbit/s	>80 kbit/s
Version 2.0 + EDR	3 Mbit/s	>80 kbit/s
Version 3.0 + HS	24 Mbit/s	See Version 3.0+HS.
Version 4.0	See Version 4.0LE.	

Στην ιστοσελίδα της wikipedia στο λήμμα στο οποίο γίνεται αναφορά στο δίκτυο του Bluetooth σημειώνεται ότι: "… Συνδέοντας δύο συσκευές της 1ης κλάσης, οι οποίες έχουν υψηλή ευαισθησία και υψηλή ισχύ, μπορεί να επιτραπούν αποστάσεις που υπερβαίνουν κατά πολύ τα 100m, ανάλογα με τις απαιτήσεις της διακομιστικής ικανότητας που έχει η εφαρμογή." Πηγή: wikipedia, http://en.wikipedia.org/wiki/Bluetooth

1.6.1.10 Σύνοψη των τεχνικών χαρακτηριστικών

Ο σχεδιασμός της τεχνολογίας του Bluetooth έγινε με γνώμονα το χαμηλό κόστος, το μεγάλο εύρος λειτουργίας, τη χαμηλή κατανάλωση ενέργειας και τη δωρεάν διάθεση του στο ευρύ κοινό. Το κλειδί της επιτυχίας της τεχνολογίας αυτής βρίσκεται στα χαρακτηριστικά της, τα οποία παρουσιάστηκαν στο κεφάλαιο αυτό. Στον πίνακα που ακολουθεί περιέχονται τα χαρακτηριστικά αυτά καθώς και το τρόπος αντιμετώπισης των ζητημάτων που προέκυψαν.

Πίνακας 1.5 Σύνοψη των βασικών χαρακτηριστικών του δικτύου του Bluetooth

Τεχνικές προκλήσεις	Χαρακτηριστικά/Αντιμετώπιση
Ζώνη συχνοτήτων (παγκόσμια)	Αντιμετώπιση: 2.4GHz Ζώνη συχνοτήτων ISM
Μετάδοση σήματος	Διασπορά φάσματος με μετατόπιση της συχνότητας (FHSS)
Τεχνικές διαμόρφωσης του σήματος	GFSK και DPSK
Μέγιστη εκπεμπόμενη ισχύς	>20 dBm
Συνολικός ρυθμός μετάδοσης δεδομένων	0,721 - 1 Mbps
Εμβέλεια	10-100 m
Πλήθος υποστηριζόμενων συσκευών	8 συσκευές / piconet (7 slaves / master)
Διαθέσιμα κανάλια για δεδομένα φωνής	3
Παρεμβολές από άλλες συσκευές που χρησιμοποιούν τη ζώνη ISM και άλλες συσκευές Bluetooth	Αντιμετώπιση: FHSS, Κώδικας διόρθωσης σφαλμάτων (Error correction coding)
Χαμηλή κατανάλωση ενέργειας	Αντιμετώπιση: Έλεγχος κατανάλωσης ενέργειας, καταστάσεις εξοικονόμησης ενέργειας, Μεταβλητό μήκος πακέτων
Οικονομικό κόστος	Αντιμετώπιση: FHSS, TDMA, low receiver sensitivity, Relaxed link budget, Low IF
Ασφάλεια • Αυθεντικοποίηση (ασφάλεια δεδομένων) • Κρυπτογράφηση (ασφάλεια δεδομένων)	Αντιμετώπιση: FHSS, ασφάλεια στο Επίπεδο Σύνδεσης Κλειδί των 128 bits Κλειδί παραμετροποιημένο από 8 - 128 bits
Υψηλή πιθανότητα σφάλματος της ασύρματης σύνδεσης	Αντιμετώπιση: ARQ, FEC και CVSD (δεδομένα φωνής)
Μετάδοση δεδομένων Φωνής/Data	Αντιμετώπιση: Circuit/Packet Switching

Σε όλες τις προδιαγραφές των τεχνικών δικτύωσης των συσκευών τα πιο σημαντικά στοιχεία είναι η μείωση της ισχύς και το χαμηλό κόστος τα οποία είναι καθοριστικής σημασίας για την ανάπτυξη των εφαρμογών.

1.6.2 Προδιαγραφές του προφίλ χρήσης

Οι προδιαγραφές του δικτύου του Bluetooth V1.0b περιλαμβάνουν τα παρακάτω προφίλ τα οποία προσδιορίζουν και τις εφαρμογές που υποστηρίζει η έκδοση αυτή του Bluetooth:

- General Access – Υποστηρίζει τις υπηρεσίες της αναζήτησης άλλων συσκευών Bluetooth, της διαχείρισης του επιπέδου σύνδεσης (Link Management) και της ασφάλειας μεταξύ των συσκευών Bluetooth.
- Service Discovery – Διενεργεί την 'Αναζήτηση' των διαθέσιμων υπηρεσιών σε γειτονικές συσκευές Bluetooth.
- Cordless Telephony – Το προφίλ αυτό δίνει την δυνατότητα σε ένα κινητό τηλέφωνο να λειτουργεί σαν ασύρματο τηλέφωνο όταν βρεθεί στην εμβέλεια του σταθμού βάσης.
- Intercom – Το προφίλ αυτό δίνει την δυνατότητα σε δύο κινητά που διαθέτουν την τεχνολογία Bluetooth να επικοινωνούν μεταξύ τους χωρίς να γίνεται χρήση του δικτύου που ανήκουν αλλά σύνδεση μέσω της τεχνολογίας Bluetooth.
- Serial Port – Προσομοίωση σειριακής θύρας.
- Headset – Καθορισμός εκπομπής και λήψης δεδομένων φωνής πάνω σε μια σύνδεση Bluetooth.
- Dial-Up Networking – Καθορισμός σύνδεσης μεταξύ ενός υπολογιστή και ενός κινητού τηλεφώνου.
- Fax – Προφίλ για την αποστολή και λήψη φαξ από υπολογιστή μέσω κινητού τηλεφώνου.
- LAN Access – Προφίλ για την δημιουργία ενός δικτύου PAN χρησιμοποιώντας το πρωτόκολλο PPP.
- Generic Object Exchange (OBEX) – Υπηρεσίες που χρησιμοποιούνται από άλλα προφίλ (File Transfer, Object Push, Synchronization).
- Object Push – Προφίλ για την αποστολή και λήψη επαγγελματικών καρτών και ραντεβού από μια συσκευή σε μια άλλη.
- Μεταφορά αρχείων (File Transfer) – Διαχείριση (δημιουργία, διαγραφή, επισκόπηση, μεταφορά) ενός συστήματος αρχείων και φακέλων από μια συσκευή σε μια άλλη.

- Συγχρονισμός συσκευών (Synchronization) – Το προφίλ αυτό δίνει την δυνατότητα συγχρονισμού δύο συσκευών την στιγμή που θα βρεθεί η μία εντός της εμβέλειας της άλλης.

Όλα τα παραπάνω προφίλ που αναφέρθηκαν δικαιολογούν γιατί τα δίκτυα της τεχνολογίας του Bluetooth είναι τόσο δημοφιλή στο ευρύ κοινό. Πηγή: wikipedia, http://en.wikipedia.org/wiki/Bluetooth [2]

1.6.2.1 Το μοντέλο χρήσης της τεχνολογίας του Bluetooth

Το μοντέλο χρήσης της τεχνολογίας του Bluetooth είναι βασισμένο στην ασύρματη διασύνδεση των συσκευών και εστιάζει σε τρεις μεγάλες κατηγορίες:

- Σημεία Πρόσβασης Φωνής/Δεδομένων
- Διασύνδεση Περιφερειακών Συσκευών
- Προσωπικά Δίκτυα

Τα σημεία πρόσβασης Φωνής/Δεδομένων (Voice/Data Access Points) ήταν το αρχικό μοντέλο χρήσης της προδιαγραφής του Bluetooth το οποίο αναφέρεται στην διασύνδεση ενός υπολογιστή (desktop υπολογιστή, φορητού, PDA) και μιας έξυπνης συσκευής. Για παράδειγμα, ένας υπολογιστής που υποστηρίζει την υπηρεσία του Bluetooth, μπορεί να συνδεθεί στο Internet για την αποστολή και λήψη email μέσω ενός κινητού τηλεφώνου που διαθέτει την υπηρεσία του Bluetooth,. Το κινητό τηλέφωνο στην προκειμένη περίπτωση λειτουργεί ως ένα προσωπικό σημείο πρόσβασης, Access Point.

Η δεύτερη κατηγορία χρήσης αναφέρεται στη διασύνδεση των περιφερειακών συσκευών. Στην κατηγορία αυτή ανήκουν οι περιφερειακές συσκευές του υπολογιστή όπως είναι τα πληκτρολόγια, τα ποντίκια, τα joysticks και τα ακουστικά τα οποία συνδέονται ασύρματα με την κεντρική μονάδα, μέσω της υπηρεσίας σύνδεσης του Bluetooth.

Στην τρίτη κατηγορία χρήσης του Bluetooth εντάσσονται τα προσωπικά δίκτυα και συγκεκριμένα τα ομότιμα προσωρινά δίκτυα (ad hoc). Εδώ καλύπτονται οι ανάγκες για δικτύωση των συσκευών ενός ατόμου ή μιας μικρής ομάδας ατόμων. Ένα PDA

που υποστηρίζει την υπηρεσία του Bluetooth, μπορεί να κάνει συγχρονισμό του ημερολογίου, των επαφών και των ραντεβού του κατόχου του, με τον υπολογιστή του γραφείου ή της γραμματέας. [6], [11]

1.7 Εξοικονόμηση ενέργειας

Η κατανάλωση ενέργειας αποτελεί έναν από τους σημαντικότερους παράγοντες στην απόδοση των ad-hoc δικτύων, ειδικότερα δε των δικτύων bluetooth, και σχετίζεται με τη διάρκεια ζωής τους. Η τεχνολογία της κατασκευής των μπαταριών βελτιώνεται συνεχώς παράλληλα όμως οι ανάγκες για ενέργεια αυξάνονται συνεχώς, αφού αυξάνεται η ζήτηση σε εύρος ζώνης και η ποιότητα των υπηρεσιών.

Η διαθέσιμη ενέργεια και η κατανάλωση της αποτελεί κρίσιμο σημείο της λειτουργίας των ad-hoc δικτύων. Ένας κόμβος που ανήκει σε ένα δίκτυο ad-hoc μπορεί για λόγους εξοικονόμησης ενέργειας να επιλέξει να μην παρέχει κάποιες υπηρεσίες οι οποίες απαιτούνται από το δίκτυο. Στα δίκτυα του bluetooth η εξοικονόμηση ενέργειας επιτυγχάνεται και με την μετάβαση του κόμβου στις καταστάσεις χαμηλής ενέργειας (Standby, Hold, Sniff, Park) που συμπεριλαμβάνονται στις προδιαγραφές του Bluetooth. [6] [11]

1.8 Ασφάλεια στο δίκτυο του Bluetooth

Παράγοντες όπως είναι η συνεχής αλλαγή της τοπολογίας, τα μικρά αποθέματα της ενέργειας, η μνήμη αποθήκευσης και το εύρος της ζώνης στα ad-hoc δίκτυα, συμβάλλουν στην μείωση της ασφάλειας στα ad-hoc δίκτυα απ' ότι στα υπόλοιπα ασύρματα δίκτυα. Ένας άλλος παράγοντας που δρα ανασταλτικά στην ασφάλεια που παρουσιάζουν τα δίκτυα αυτά είναι η έλλειψη της κεντρικής εσωτερικής δομής που τα χαρακτηρίζει η οποία καθιστά δύσκολη την πιστοποίηση των δεδομένων. Η αύξηση των επιθέσεων στα δίκτυα αυτά είναι ένα ζήτημα που απαιτεί ακόμα μεγαλύτερη ασφάλεια. Τα θέματα ασφαλείας που λαμβάνονται υπ' όψη είναι τα εξής:

> Η Διαθεσιμότητα εξασφαλίζει την παροχή των υπηρεσιών παρά τις επιθέσεις άρνησης παροχής.

Η Εμπιστευτικότητα εξασφαλίζει ότι οι πληροφορίες δεν αποκαλύπτονται σε μη εξουσιοδοτημένες οντότητες.

Η Ακεραιότητα εξασφαλίζει ότι τα μεταδιδόμενα μηνύματα δεν παραποιούνται.

Η Πιστοποίηση ταυτοποιεί την οντότητα με την οποία επικοινωνεί ένας κόμβος.

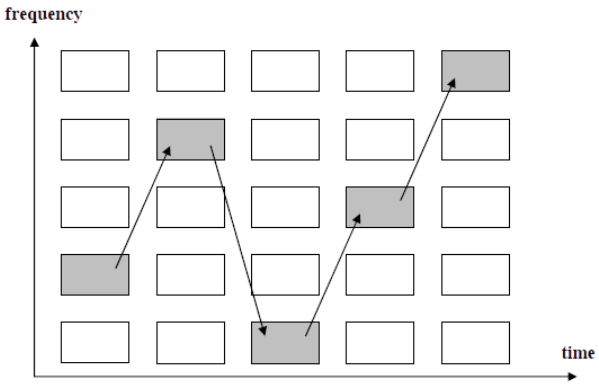
Υπάρχουν αρκετοί τρόποι της αντιμετώπισης των επιθέσεων όπως είναι η διαγραφή ή η τροποποίηση των μηνυμάτων, λόγω όμως της ελαττωματικής αρχιτεκτονικής που έχουν τα ad-hoc δικτύων, τα καθίστα ευάλωτα σε επιθέσεις ασφαλείας.

Υπάρχουν αλγόριθμοι που χρησιμοποιούνται για την ανίχνευση και την απομόνωση των ελαττωματικών κόμβων από το δίκτυο. Οι αλγόριθμοι αυτοί δεν βρίσκουν εύκολα εφαρμογή στα ad-hoc δίκτυα τα οποία έχουν συνεχείς αλλαγές στην τοπολογία τους, αφού απαιτείται από τους αλγόριθμους οι επαναλαμβανόμενες μεταδόσεις μηνυμάτων σε όλους του κινητούς κόμβους.

Σε ένα ασύρματο δίκτυο όπως είναι το δίκτυο του Bluetooth, τα δεδομένα μεταφέρονται ελεύθερα στον αέρα και μπορούν να υποκλαπούν από οποιονδήποτε βρεθεί στην εμβέλεια της συσκευής που εκπέμπει. Η ασφάλεια είναι ένας πολύ σημαντικός παράγοντας και μπορεί να καθορίσει την επιτυχία ή όχι ενός πρωτοκόλλου. Η ασφάλεια στο Bluetooth επιτυγχάνεται με τρεις τρόπους:

- Ψευδοτυχαία εναλλαγή συχνότητας
- Αυθεντικοποίηση
- Κρυπτογράφηση

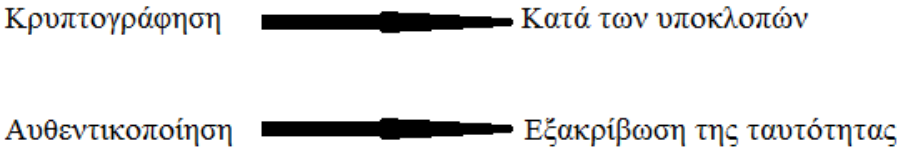
Η εναλλαγή συχνότητας είναι μία τεχνική διασποράς του φάσματος (παράγραφος 1.6.1.1) που στοχεύει στην ελάττωση του θορύβου. Παρόλα αυτά είναι επίσης ένας καλός τρόπος αποτροπής των ωτακουστών, γιατί είναι πολύ δύσκολο για έναν ωτακουστή να παγιδεύσει τις αλλαγές της συχνότητας που πραγματοποιούνται κατά τη μετάδοση του σήματος σε ένα δίκτυο Bluetooth. Η εικόνα που ακολουθεί δείχνει πως μία συχνότητα ενός καναλιού του δικτύου του Bluetooth μπορεί να μεταβάλλεται ανά το χρόνο.



Εικόνα 1.28 Η συχνότητα ενός καναλιού Bluetooth μεταβάλλεται ψευδο-τυχαία

Η τεχνολογία του Bluetooth έχει διάφορα επίπεδα και μηχανισμούς ασφαλείας για να εξασφαλίσει ότι τα δεδομένα που μεταδίδονται δεν μπορούν να υποκλαπούν. Για να το πετύχει αυτό χρησιμοποιεί τέσσερα βασικά κλειδιά για την κρυπτογράφηση και για την εξακρίβωση των άλλων συσκευών του δικτύου του Bluetooth και άλλα δύο μυστικά κλειδιά, το κλειδί για την εξακρίβωση-αυθεντικοποίηση (authentication key) της συσκευής από τις άλλες συσκευές και το κλειδί της κρυπτογράφησης (encryption key) τα οποία μπορούν να είναι από 8bit έως 128bit.

Η αυθεντικοποίηση επιτρέπει σε ένα χρήστη να περιορίσει την συνδεσιμότητα σε συγκεκριμένες συσκευές. Η κρυπτογράφηση κάνει τα δεδομένα αναγνώσιμα μόνο σε εξουσιοδοτημένες συσκευές, εμποδίζοντας έτσι τους ωτακουστές.



Εικόνα 1.29 Αυθεντικοποίηση και κρυπτογράφηση

Όλες οι συσκευές που διαθέτουν την υπηρεσία του Bluetooth εφαρμόζουν ένα Γενικό Προφίλ Πρόσβασης (Generic Access Profile), το οποίο ορίζει ένα μοντέλο ασφαλείας που περιέχει τρία διαφορετικά επίπεδα ασφαλείας:

Επισφαλής κατάσταση ή -μη ασφαλείς- λειτουργία (Insecure mode ή Non-Security) – Η συσκευή δεν χρησιμοποιεί καμιά διαδικασία ασφαλείας.

Υπηρεσία επιπέδου επιβολής ασφάλειας (Service-level enforced security). Δεν απαιτούνται διαδικασίες ασφαλείας που έχουν πραγματοποιηθεί πριν

από τη δημιουργία ενός καναλιού. Σε αυτό το επίπεδο η άδεια για την χρήση μιας συσκευής εξαρτάται από το είδος της υπηρεσίας που ζητείτε.

Επιβολή ασφάλειας σε επίπεδο-σύνδεσης (Link-level enforced security). Οι διαδικασίες ασφαλείας που ξεκίνησαν πριν από την εγκαθίδρυση συνδέσμου ολοκληρώνονται. Αυτό το επίπεδο ασφαλείας απαιτεί διαδικασίες εξακρίβωσης προτού δοθεί η δυνατότητα χρήσης κάποιας υπηρεσίας της συσκευής.

Οι δικλείδες ασφαλείας του Bluetooth είναι σημαντικές για το project αυτό, γιατί σχετίζεται με τον έλεγχο πρόσβασης. Πάραυτα, αυτή η αυξημένη ασφάλεια δεν είναι αρκετά αποτελεσματική. [2] [6]

1.9 Σύνοψη Bluetooth

Στον παρακάτω πίνακα παρουσιάζονται τα κυριότερα χαρακτηριστικά και τα πλεονεκτήματά του πρωτόκολλου του Bluetooth.

Πίνακας 1.6 Χαρακτηριστικά Bluetooth

Χαρακτηριστικά	Πλεονεκτήματα
Χρήση συχνότητας 2.4GHz (ISM Band).	Οι συσκευές Bluetooth λειτουργούν σε παγκόσμιο επίπεδο.
Υποστήριξη μέχρι 8 συσκευών από ένα piconet, όπου η μία αναλαμβάνει το ρόλο της συσκευής master και οι υπόλοιπες είναι εξαρτημένες συσκευές slave.	Πολλαπλά piconets συνδέονται μεταξύ τους μέσω των συσκευών που αναλαμβάνουν το ρόλο της συσκευής master, αυξάνοντας έτσι τον αριθμό των συνδεδεμένων συσκευών.
Το Bluetooth επιτρέπει την ασύρματη σύνδεση του κινητού τηλεφώνου.	Απλοποίηση της σύνδεσης στο Internet ή στο εταιρικό δίκτυο εξαλείφοντας την ανάγκη των καλωδίων.
Οι συσκευές που υποστηρίζουν την προδιαγραφή του Bluetooth μπορούν να επικοινωνούν σε εμβέλεια έως και 10m.	Περιορίζοντας την εμβέλεια των συσκευών στα 10m, μειώνονται και οι απαιτήσεις των συσκευών σε ενέργεια, κάνοντας το Bluetooth πρακτικό για συσκευές που λειτουργούν με μπαταρία. Τα 10m είναι αρκετά για τα δίκτυα WPAN, για τα οποία σχεδιάστηκε το Bluetooth. Περιορίζονται έτσι τις παρεμβολές από τις άλλες συνδέσεις του

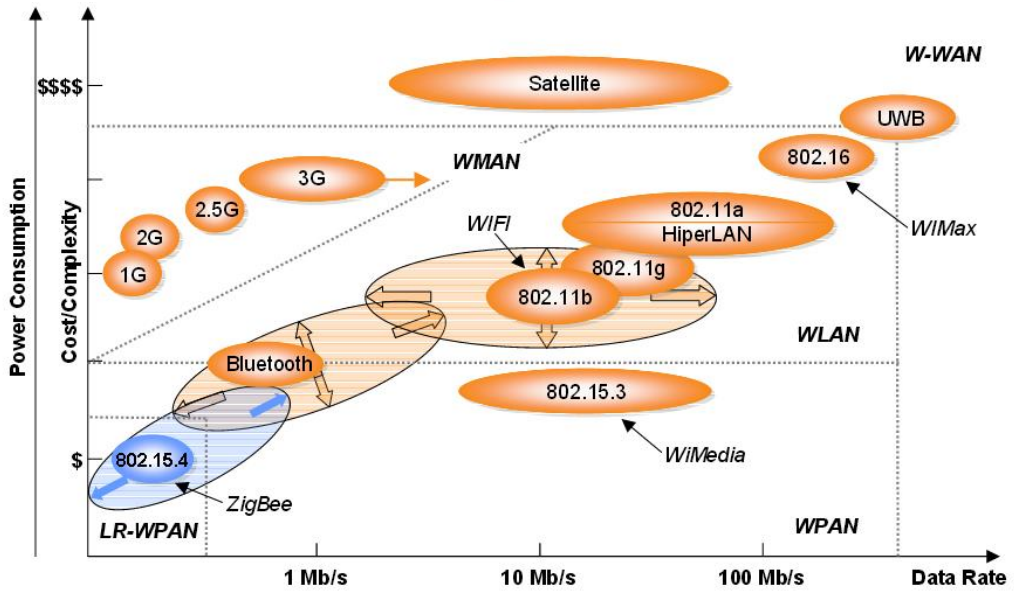
	δικτύου του Bluetooth.
Η εξακρίβωση και η κρυπτο-γράφηση με τη χρήση Public και Private κλειδιών είναι σημαντικά στοιχεία της ασφάλειας του προτύπου του Bluetooth.	Παρέχεται υψηλού βαθμού ασφάλεια στην επικοινωνία μεταξύ των συσκευών του Bluetooth.
Δεν απαιτείται οπτική επαφή μεταξύ των συσκευών Bluetooth για να γίνει μια σύνδεση.	Παρέχεται μεγαλύτερη ευελιξία και ευκολία στη χρήση σε αντίθεση με άλλες τεχνολογίες όπως είναι η τεχνική του IrDA. Είναι δυνατόν να εγκατασταθούν συνδέσεις ακόμα και όταν υπάρχουν εμπόδια στη μέση.
Πρόγραμμα απόκτησης του σήματος Bluetooth για τις συσκευές που υποστηρίζουν τις προδιαγραφές του Bluetooth.	Το πρόγραμμα αυτό θα παρέχει στην βιομηχανία και στην αγορά ένα μηχανισμό ο οποίος θα αναγνωρίζει τις συσκευές που μπορούν να συνδέονται μεταξύ τους μέσω του Bluetooth. θα διασφαλιστεί η διάφανης λειτουργία μεταξύ των συσκευών που υποστηρίζουν το Bluetooth και θα είναι πιο εύκολή η χρήση των υπηρεσιών που υποστηρίζει στον τελικό χρήστη.

2 Η τεχνολογία Bluetooth, ο ανταγωνισμός και η επικράτηση της στην αγορά

Στο κεφάλαιο αυτό παρουσιάζεται ο τρόπος επικράτησης της προδιαγραφής του Bluetooth έναντι των άλλων συναφών τεχνολογιών, τα στοιχεία που συνέβαλαν στην εδραίωση του και οι εμπορικές προοπτικές της τεχνολογίας αυτής.

2.1 Το Bluetooth εναντίον των ανταγωνιστριών τεχνολογιών

Στις παρακάτω ενότητες παρουσιάζονται συνοπτικά οι ασύρματες τεχνολογίες των PAN δικτύων οι οποίες έχουν συναφή χαρακτηριστικά με την προδιαγραφή του bluetooth και που αποτελούν τις κύριες ανταγωνίστριες τεχνολογίες του bluetooth στην αγορά. Αναφέρονται τα κυριότερα χαρακτηριστικά τους αλλά και τα πλεονεκτήματα και τα μειονεκτήματα που παρουσιάζουν έναντι του πρωτόκολλου του Bluetooth.



Εικόνα 2.1 Το Bluetooth και οι ανταγωνίστριες τεχνολογίες

Η παραπάνω εικόνα έχει δημοσιευτεί από τον Dr. Jose A. Gutierrez και την εταιρεία Eaton Corporation το 2005 σε μια παρουσίαση με θέμα «IEEE Std. 802.15.4 Enabling Pervasive Wirless Sensor Networks" οι διαφάνειες τις οποίας έχουν αναρτηθεί στην ιστοσελίδα του Πανεπιστημίου του Berkeley, http://www.cs.berkeley.edu.

2.1.1 Bluetooth vs Wi-Fi

Τα δίκτυα του Bluetooth και του Wi-Fi (Wireless Fidelity, προδιαγραφή IEEE 802.11) είναι δυο διαφορετικές προδιαγραφές που έχουν αναπτυχθεί για την υλοποίηση της ασύρματης δικτύωσης. Συνεπώς τα δίκτυα αυτά παρουσιάζουν κάποιες ομοιότητες στην εφαρμογή τους όπως είναι η δημιουργία ασύρματων δικτύων και η μεταφορά αρχείων. Το δίκτυο του Wi-Fi προορίζεται για την αντικατάσταση της καλωδίωσης των τοπικών δικτύων στους χώρους εργασίας (WLAN). Το δίκτυο του Bluetooth προορίζεται για την δημιουργία δικτύων WPAN, για την αντικατάσταση της καλωδίωσης και για τον εντοπισμό της θέσης στις εφαρμογές που αναπτύσσονται για τις έξυπνες οικιακές συσκευές (θερμοστάτες, κ.τ.λ.), για την μεταφορά δεδομένων ανάμεσα σε δύο συσκευές όταν η ταχύτητα δεν είναι καθοριστικός παράγοντας.

Πίνακας 2.1 Bluetooth vs Wi-Fi

	Bluetooth	**WiFi**
Χρήση ενέργειας	Μέτρια	Υψηλή
Ρυθμός δεδομένων	Μέτριο προς υψηλό	Υψηλό
Κάλυψη	Υψηλή	Υψηλή
Κόστος HW	Χαμηλό προς μέτριο	Υψηλό
Ρυθμός μετάδοσης δεδομένων	2.1Mbps	600 Mbps
Ασφάλεια	Υψηλή	Υψηλή
Αρχή προδιαγραφών	Bluetooth SIG	IEEE, WECA
Πλεονεκτήματα	Τεχνολογία mainstream με υψηλή πρόσληψη	Τεχνολογία mainstream με υψηλή πρόσληψη
Μειονεκτήματα	Ζητήματα ενέργειας (τυπική διάρκεια μπατα-ρίας 1-2 εβδομάδες)	Χαμηλή ακρίβεια, ζητήματα πολύ μεγάλης κατανάλωσης ενεργείας (τυπική διάρκεια μπαταρίας 1-2 ημέρες)

Το Wi-Fi δίκτυο αποτελεί την αντίστοιχη ασύρματη έκδοση ενός ενσύρματου Ethernet δικτύου και για να λειτουργήσει χρειάζεται έναν router και τη ρύθμιση-διαμόρφωση των κοινών πόρων. Χρησιμοποιείται κυρίως για τη μετάδοση των αρχείων και την εγκατάσταση των συνδέσεων των audio συσκευών (για παράδειγμα, ακουστικά και hands-free συσκευές). Το Wi-Fi χρησιμοποιεί τις ίδιες ραδιοσυχνότητες με το Bluetooth, αλλά διαθέτει μεγαλύτερη ισχύ μετάδοσης, αυτό

έχει ως αποτέλεσμα υψηλότερους ρυθμούς μετάδοσης δεδομένων και καλύτερο εύρος από το σταθμό βάσης.

Αν και η προδιαγραφή του Bluetooth εκπέμπει στην ίδια ζώνη συχνοτήτων (2,4 GHz) με τα πρότυπα της κατηγορίας 802.11 της IEEE, όμως παρόλα αυτά η τεχνολογία του Bluetooth δεν είναι το κατάλληλη για να αντικαταστήσει το πρότυπο του Wi-Fi. Αν και οι δύο τεχνολογίες στοχεύουν στην ασύρματη δικτύωση, η δικτύωση αυτή που παρέχουν αφορά διαφορετικούς τύπους δικτύων. Σε σύγκριση με το Wi-Fi, η δικτύωση που παρέχει το Bluetooth αφορά τα δίκτυα WPAN και είναι πιο αργή, πιο περιορισμένη σε εύρος και υποστηρίζει λιγότερες συσκευές. [2]

2.1.2 Bluetooth vs HomeRF

Το HomeRF είναι μια ασύρματη τεχνολογία η οποία δημιουργήθηκε το 1998 από την ομάδα εργασίας του HomeRF με σκοπό την ασύρματη δικτύωση των οικιακών συσκευών και χρησιμοποιείται συνήθως για να συνδέσει desktop. Η τεχνολογία του HomeRF λειτουργεί στη ζώνη συχνοτήτων των 2,4 GHz του ISM. Η προδιαγραφή αυτή μειονεκτεί στην υλοποίηση των δικτύων ad-hoc σε σχέση με την προδιαγραφή του Bluetooth.

Πίνακας 2.2 Bluetooth vs HomeRF

	Bluetooth	**HomeRF**
Τεχνολογία	2,4 GHz FHSS	2,4 GHz FHSS
Κατανάλωση ενέργειας	Μέτρια	Υψηλή
Κάλυψη	100m (classA)	50m
Κόστος	Χαμηλό προς μέτριο	Υψηλό
Ρυθμός μετάδοσης δεδομένων	1 Mbps	10 Mbps
Ασφάλεια	Υψηλή	Υψηλή
Αρχή προδιαγραφών	Bluetooth SIG	IEEE, WECA
Πλήθος κόμβων	8	127
Αντιμετώπιση παρεμβολών	Μέτρια	Υψηλή

Το HomeRF παρέχει καλύτερη ασφάλεια από την τεχνολογία του Bluetooth και είναι ανθεκτικό στις παρεμβολές των γειτονικών κόμβων. Ένα άλλο μειονέκτημα της τεχνολογίας του HomeRF είναι η κατανάλωση ισχύος η οποία είναι πολύ μεγαλύτερη απ' αυτή του Bluetooth. Η προδιαγραφή του HomeRF δεν είναι πολύ προσιτή στο αγοραστικό κοινό γιατί δεν διατίθεται δωρεάν όπως συμβαίνει με την τεχνολογία του Bluetooth. [14]

2.1.3 Bluetooth vs High Rate WPAN

Στο πρότυπο IEEE 802.15.3 περιλαμβάνονται τα δίκτυα High-Rate-Wireless Personal Area Networks (HR-WPAN) τα οποία στοχεύουν στην ασύρματη δικτύωση των σταθερών και κινητών συσκευές μέσα σε ένα προσωπικό χώρο λειτουργίας. Ο κύριος σκοπός της δημιουργίας του προτύπου του High-Rate-WPAN είναι η υλοποίηση ενός ασύρματου δικτύου το οποίο θα είναι διαθέσιμο σε χαμηλό κόστος, θα παρουσιάζει χαμηλή πολυπλοκότητα και κατανάλωση ενέργειας και θα παρέχει συνδεσιμότητα υψηλού εύρους δεδομένων στις ασύρματες προσωπικές συσκευές.

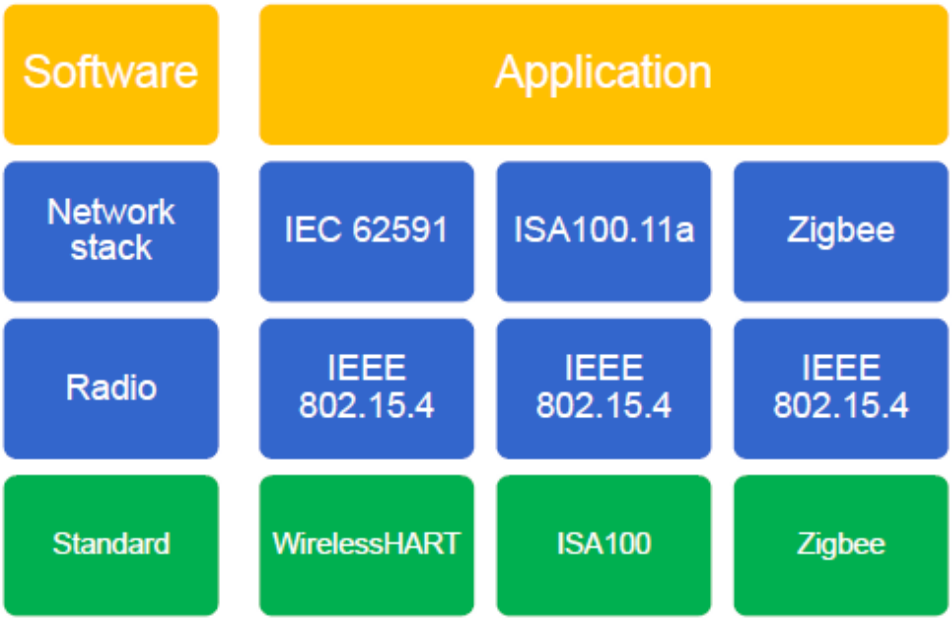
Πίνακας 2.3 Bluetooth vs High Rate WPAN

Χαρακτηριστικά	Bluetooth	High Rate WPAN
Ζώνη συχνοτήτων	2,4 GHz	2,4 GHz
Bandwidth	1 MHz	15 MHz
Πλήθος καναλιών	79	5
Μέγιστο εύρος (Mbps)	0,72	55
Εύρος μετάδοσης	10 m	10 m
Εφαρμογή	WPAN	HR-WPAN

2.1.4 Bluetooth vs Low Rate WPAN

Η προδιαγραφή του δικτύου του LR WPAN (Low Rate WPAN) υλοποιείται με το πρότυπο 802.15.4 της IEEE το οποίο παρουσιάστηκε το 2003 και υποστηρίζεται από την ομάδα εργασίας IEEE 802.15. Το δίκτυο του LR WPAN στοχεύει στην ασύρματη δικτύωση των προσωπικών δικτύων τα οποία υποστηρίζουν το χαμηλό ρυθμό μετάδοσης των δεδομένων, τη μεγάλη διάρκεια της μπαταρίας η οποία μπορεί να

είναι από λίγους μήνες μέχρι χρόνια, την πολύ χαμηλή πολυπλοκότητα και το χαμηλό κόστος των εφαρμογών.



Εικόνα 2.2 Τα πρότυπα WirelessHART, ISA100.11a και ZigBee

Η εικόνα έχει δημοσιευτεί στην ιστοσελίδα του Πανεπιστημίου του Aalto (https://noppa.aalto.fi/noppa/kurssi/as-74.3199/luennot/AS-74_3199_summary_of_course.pdf) και είναι υλικό από το μάθημα με θέμα «AS-74.3199 Wireless Automation» του τμήματος των Ηλεκτρολόγων Μηχανικών.

Η κατηγορία του προτύπου 802.15.4 έχει πολλές υποκατηγορίες. Η δημιουργία των δικτύων ZigBee και WirelessHART τα οποία παρουσιάζονται στις επόμενες παραγράφους στηρίχθηκε πάνω στο πρότυπο αυτό. [15]

2.1.4.1 Bluetooth vs ZigBee

Η ανάπτυξη του δικτύου αυτού στηρίχτηκε στο πρότυπο 802.15.4 της IEEE. Η προδιαγραφή του ZigBee λειτουργεί στις ραδιοσυχνότητες των 868 MHz του ISM στην Ευρώπη, των 915 MHz στις ΗΠΑ και στην Αυστραλία και στα 2,4 GHz στις υπόλοιπες χώρες του κόσμου. Το πρότυπο του ZigBee σχεδιάστηκε με γνώμονα την αποδοτικότητα των ασύρματων επικοινωνιών, τους χαμηλούς ρυθμούς μετάδοσης των δεδομένων, την χαμηλή κατανάλωση της ενέργειας, την ασφάλεια και την αξιοπιστία. Δόθηκε έμφαση σε πέντε βασικές κατηγορίες εφαρμογών: στους

αυτοματισμούς των σπιτιών και των κτηρίων, στην έξυπνη ενέργεια, στις υπηρεσίες των τηλεπικοινωνιών και στην υγεία.

Πίνακας 2.4 Bluetooth vs ZigBee

Χαρακτηριστικά	Bluetooth	ZigBee
Ζώνη συχνοτήτων	2,4 GHz	868 MHz, 915 MHz, 2.4 GHz
Bandwidth	1 MHz	20-250 KHz
Ρυθμός μετάδοσης δεδομένων	1 Mbps	250 Kbps
Πλήθος κόμβων	8	65534
Ασφάλεια	Μέτρια	Μέτρια
Εύρος μετάδοσης	10 m	70 m
Χρόνος ζωής μπαταρίας	1-7 ημέρες	100-1000+ ημέρες
Εφαρμογή	WPAN	LR-WPAN

Οι ταχύτητες μετάδοσης των δεδομένων ποικίλουν από 20 kbs/s στη ζώνη συχνοτήτων των 868 MHz έως τα 250 kbs/s στη ζώνη συχνοτήτων των 2,4 GHz. Το κόστος της προδιαγραφής του ZigBee είναι χαμηλό παρόλα αυτά η τεχνολογία αυτή δεν είναι διαθέσιμη δωρεάν στις συσκευές κινητής τηλεφωνίας και στους Η/Υ και είναι ένα δίκτυο που σχεδιαστικέ περισσότερο για τον έλεγχο και τις ανάγκες του αυτοματισμού των συσκευών. [16]

2.1.4.2 Bluetooth vs WirelessHART

Το δίκτυο του WirelessHART κατασκευάστηκε από την ένωση των εταιρειών HART Communications Foundation (HCF) με βάση το πρότυπο 802.15.4 της IEEE και περιλαμβάνει είναι ένα ανοιχτό βιομηχανικό πρότυπο που έχει σχεδιαστεί για να ανταποκρίνεται στις ιδιαίτερες απαιτήσεις της ασύρματης επικοινωνίας σε τοπικό επίπεδο στον τομέα της παραγωγής.

Η προδιαγραφή του WirelessHART είναι μια ασύρματη τεχνολογία δικτύωσης που χρησιμοποιείται ευρέως στη βιομηχανία για την ενεργοποίηση, την παρακολούθηση και τον έλεγχο των αισθητήρων. Η τεχνολογία του WirelessHART χαρακτηρίζεται

από την αξιοπιστία, την ασφάλεια, την αποδοτικότητα και την ευκολία στη χρήση της.

Πίνακας 2.5 Bluetooth vs WirelessHART

Χαρακτηριστικά	WirelessHART	Bluetooth
Ζώνη συχνοτήτων	2,4 GHz	2,4 GHz
Ρυθμός μετάδοσης δεδομένων	250 Kbps	1 Mbps
Πλήθος κόμβων	30000	8
Ασφάλεια	Υψηλή	Μέτρια
Εύρος μετάδοσης	250 m	10 m
Διάρκεια μπαταρίας	100-1000+ ημέρες	1-7 ημέρες
Εφαρμογή	LR-WPAN	WPAN

Το WirelessHART βασίζεται στο πρωτόκολλο Highway Addressable Remote Transducer Protocol (HART) το οποίο επιτρέπει τη σύνδεση και τον έλεγχο των συσκευών ενώ διαθέτει και εργαλεία για τη διαμόρφωση των δικτύων και την ενσωμάτωση νέων ασύρματων συσκευών σ΄ αυτά. Η τεχνολογία αυτή λειτουργεί στη ζώνη των 2,4 GHz του ISM και δεν είναι διαθέσιμη δωρεάν ενώ το κόστος της θεωρείται υψηλό. [17]

2.1.5 Bluetooth vs Infrared Data Association (IrDA)

Η τεχνολογία της ασύρματης δικτύωσης με τη χρήση των υπέρυθρων ακτινών, Infrared Data Association (IrDA), χρησιμοποιείται για τη μεταφορά δεδομένων υψηλής ταχύτητας, σε μικρές αποστάσεις όπου υπάρχει οπτική επαφή (Line-of-Sight LOS) και σε συνδέσεις point-to-point στις οποίες απαιτείται πολύ χαμηλός ρυθμός σφάλματος bit (bit error rate -BER). Η τεχνολογία αυτή σχεδιάστηκε για να υποστηρίξει τη σύνδεση των υπολογιστών και των κινητών τηλεφώνων με τις περιφερειακές συσκευές. Παρουσιάζει μεγάλη ανοχή σε παρεμβολές ενώ το γεγονός ότι δεν επηρεάζεται από τις παρεμβολές των γειτονικών κόμβων επιτρέπει την συνύπαρξη της με άλλα δίκτυα.

Η αύξηση της χωρητικότητας και των δυνατοτήτων των συσκευών κινητής τηλεφωνίας οδήγησε στην δημιουργία του προτύπου IrDA Giga-IR το οποίο υλοποιήθηκε μετά από βελτιώσεις που έγιναν στο επίπεδο των πρωτοκόλλων και στο

σχεδιασμό των πομποδεκτών κάνοντας την τεχνολογία αυτή πιο αποδοτική και αυξάνοντας την απόσταση που μπορεί να καλύψει στα 3 μέτρα.

Πίνακας 2.6 Bluetooth vs IrDA

Χαρακτηριστικά	IrDA	Bluetooth
Μετάδοση σήματος	Υπέρυθρες ακτίνες	Ζώνη συχνοτήτων 2,4 GHz
Ρυθμό μετάδοσης	4-16 Mbps	1 Mbps
Πλήθος κόμβων	point-to-point	8
Ασφάλεια	Υψηλή	Μέτρια
Εμβέλεια	1-3 m	1-100 m
Εφαρμογή		WPAN

Η τεχνολογία αυτή δεν επηρεάζει τις άλλες ασύρματες επικοινωνίες αλλά και δεν επηρεάζεται από τις παρεμβολές των γειτονικών κόμβων επιτρέποντας έτσι τη συνύπαρξη της με τα άλλα δίκτυα. Το μεγαλύτερο πλεονέκτημα του δικτύου IrDA έναντι του Bluetooth είναι ο υψηλός ρυθμός μετάδοσης των δεδομένων, ο οποίος το καθιστά κατάλληλο για εφαρμογές υψηλής ταχύτητας. Το Bluetooth παρέχει στους χρήστες μεγαλύτερη κινητικότητα γιατί μπορεί να διαπεράσει ρούχα και μαλακά χωρίσματα. Οι υπέρυθρες ακτίνες απαιτούν για την λειτουργία τους την οπτική επαφή (line-of-sight) του πομπού με τον δέκτη για την μετάδοση των δεδομένων. Η ανταλλαγή δεδομένων μεταξύ πολλών συσκευών δεν είναι δυνατή με τις υπέρυθρες αφού μπορούν να υποστηρίξουν μόνο τις συνδέσεις point-to-point μεταξύ των συσκευών. Και οι δύο τεχνολογίες έχουν τα πλεονεκτήματα και τα μειονεκτήματά τους και δεν μπορεί πλήρως να αντικαταστήσει η μία την άλλη, μπορούν όμως να συνυπάρξουν συμπληρώνοντας η μία την άλλη. [18]

2.1.6 Bluetooth vs Wireless LANs

Το πρότυπο IEEE 802.11 ή WLAN (Wireless LAN) είναι μια άλλη προδιαγραφή για την εγκαθίδρυση μιας ασύρματης δικτύωσης η οποία χρησιμοποιεί για τη μετάδοση του σήματος την ελεύθερη ζώνη των 2.4 GHz.

Η τεχνολογία αυτή δημιουργήθηκε για τη μετάδοση μόνο των δεδομένων και δεν υποστηρίζει την μετάδοση του σήματος της φωνής. Μπορεί να δημιουργήσει ad hoc

συνδέσεις αλλά στην πραγματικότητα αναπαράγει ένα δίκτυο Ethernet χωρίς καλωδίωση.

Πίνακας 2.7 Bluetooth vs Wireless LANs

	IEEE 802.11b & 802.11a	**Bluetooth**
Μετάδοση σήματος	IEEE 802.11b - 2.4 GHz IEEE 802.11a - 5GHHz	2.4 GHz
Ρυθμό μετάδοσης	11 Mbps- 54 Mbps	1-2 Mbps
Εμβέλεια	100 m - 802.11b 20 m - 802.11a	1-100 m
Παρεμβολές	2.4 GHz (ζώνη με προβλήματα παρεμβολών)	2.4 GHz (ζώνη με προβλήματα παρεμβολών)
Κόστος	Υψηλότερο από το κόστος του Bluetooth	

Η τεχνολογία αυτή δεν λειτουργεί το ίδιο καλά στην υποστήριξη της ad hoc δικτύωσης όσο η τεχνολογία του Bluetooth, διότι η εγκατάσταση της ασύρματης δικτύωσης WLAN είναι πιο πολύπλοκη. Η ανάπτυξη των ad hoc δικτύων είναι ένα από τα δυνατά σημεία της τεχνολογίας του Bluetooth. Για το λόγο αυτό, οι δύο τεχνολογίες μπορούν να συνυπάρξουν και να ενισχύσουν η μία την άλλη. Ένα παράδειγμα τέτοιας συνύπαρξης είναι η δημιουργία ενός δικτύου στο οποίο η σύνδεση των περιφερειακών συσκευών με τα PC's θα γίνεται με Bluetooth ενώ η σύνδεση των PC's με το LAN θα γίνεται με την τεχνολογία WLAN

2.2 Γιατί επικράτησε η τεχνολογία Bluetooth

Οι τεχνολογίες της ασύρματης δικτύωσης που παρουσιαστήκαν στις προηγούμενες παραγράφους (παράγραφος 2.1, Το Bluetooth εναντίον των ανταγωνιστριών τεχνολογιών) έχουν δημιουργηθεί για να καλύψουν εξειδικευμένες ανάγκες. Η συνύπαρξη των τεχνολογιών αυτών είναι εφικτή και σε ορισμένες περιπτώσεις απαραίτητη.

Ο λόγος που ενέπνευσε την ομάδα SIG του Bluetooth στην δημιουργία αυτής της τεχνολογίας ήταν η ασύρματη δικτύωση των περιφερειακών συσκευών των Η/Υ και των έξυπνων συσκευών κινητής τηλεφωνίας. Στην ασύρματη δικτύωση των περιφερειακών συσκευών των Η/Υ στοχεύει και η τεχνολογία των δικτύων IrDA, με

αποτέλεσμα οι αυτές οι δύο τεχνολογίες να είναι ανταγωνίστριες μεταξύ τους. Η προδιαγραφή του δικτύου IrDA ήταν ήδη από την δεκαετία του 1990 μια δημοφιλής τεχνολογία για την διασύνδεση των περιφερειακών συσκευών με PC, αλλά περιορίζεται σοβαρά από την μικρή απόσταση σύνδεσης του 1 m και την απαίτηση της απευθείας οπτικής επαφής (LOS) που πρέπει να πληρείται για την επίτευξη της επικοινωνίας. Ο περιορισμός αυτός καταργεί τη χρήση IrDA για έναν υπολογιστή που δεν έχει οπτική επαφή, και οι συσκευές επικοινωνίας είναι κοντά, αλλά δεν έχουν απευθείας οπτική επαφή με τον υπολογιστή. Οι δυσκολίες αυτές μπορούν να ξεπεραστούν με την χρήση του προτύπου του Bluetooth.

Λόγω της φύσης των ραδιοσυχνοτήτων, το Bluetooth δεν υπόκειται σε τέτοιους περιορισμούς. Επιπλέον για τις ασύρματες συνδέσεις μέχρι τα 10 m (κλάση Α: έως 100 m, αυξάνοντας την ισχύ του πομπού), οι συσκευές δεν χρειάζεται να είναι σε απευθείας οπτική επαφή και μπορεί να επιτευχθεί η διασύνδεση τους ακόμα και μέσα από λεπτούς τοίχους ή άλλα μη μεταλλικά αντικείμενα. Αυτό επιτρέπει σε ένα κινητό τηλέφωνο που βρίσκεται μέσα σε μια τσέπη ή σ' ένα χαρτοφύλακα, να λειτουργεί ως modem για ένα φορητό υπολογιστή ή ένα PDA.

Η ομάδα SIG του Bluetooth έχει σχεδιάσει την τεχνολογία αυτή με σκοπό να είναι προσιτή στο ευρύ κοινό. Εταιρείες όπως είναι η Ericsson, η IBM, η Intel, η Nokia η Toshiba, η 3Com, η Microsoft και Motorola οι οποίες κατέχουν το μεγαλύτερο κομμάτι στις πωλήσεις των έξυπνων κινητών συσκευών και των συσκευών των Η/Υ ενσωματώνουν την τεχνολογία του Bluetooth στα προϊόντα τους με κόστος χαμηλότερο από 5 ευρώ/μονάδα. Στον αντίποδα του χαμηλού κόστους βρίσκεται η περιορισμένη απόσταση σύνδεσης και οι χαμηλές ταχύτητες μετάδοσης.

Οι τεχνολογίες που χρησιμοποιούν για τη μετάδοση του σήματος τις υπέρυθρες ακτίνες προϋποθέτουν την οπτική επαφή των συσκευών. Οι τεχνολογίες που μεταδίδουν το σήμα μέσω των ραδιοσυχνοτήτων αντιμετωπίζουν προβλήματα παρεμβολών, υποστηρίζουν μικρότερους ρυθμούς μετάδοσης και έχουν μεγαλύτερες απαιτήσεις στην κατανάλωση ενέργειας.

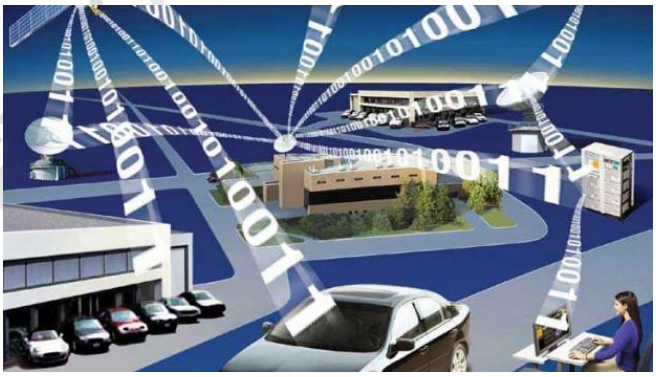
Τέλος, το βασικό πλεονέκτημα του Bluetooth είναι η ικανότητά του να χειρίζεται ταυτόχρονα τις μεταδόσεις δεδομένων και φωνής. Μπορεί να υποστηρίζει ένα ασύγχρονο κανάλι δεδομένων και μέχρι τρία σύγχρονα κανάλια φωνής, ή ένα κανάλι που να υποστηρίζει τη φωνή και τα δεδομένα. Η δυνατότητα αυτή σε συνδυασμό με

την ad hoc διασύνδεση των συσκευών και την ανακάλυψη νέων υπηρεσιών το καθιστούν μια εξαιρετική λύση για την σύνδεση των συσκευών της κινητής τηλεφωνίας με τις εφαρμογές του διαδικτύου. Αυτός ο συνδυασμός επιτρέπει τέτοιου είδους καινοτόμες λύσεις, όπως τη σύνδεση μίας κινητής συσκευής με το hands-free headset για την υποστήριξη των φωνητικών κλήσεων, την ανταλλαγή αρχείων εικόνας, video ή ήχου μεταξύ συσκευών κινητής τηλεφωνίας και Η/Υ, τη δυνατότητα εκτύπωσης σε φαξ, και τον αυτόματο συγχρονισμό του tablet, του laptop, και της εφαρμογής του κατάλογου διευθύνσεων του κινητού τηλεφώνου. [18]

2.3Στοιχεία που συνέβαλλαν στην εδραίωση του Bluetooth

Η ανάγκη για την αντικατάσταση της καλωδίωσης στα δίκτυα οδήγησε την ομάδα SIG του Bluetooth στην ανάπτυξη της τεχνολογίας αυτής. Η ελεύθερη διάθεση των εργαλείων ανάπτυξης λογισμικού, Software Development Kits (SDKs) για την ανάπτυξη των εφαρμογών και οι ολοένα και χαμηλότερες τιμές της κατασκευής των Bluetooth chips συνέβαλαν στην επικράτηση του προτύπου του Bluetooth στην αγορά.

Οι συνεχώς αυξανόμενες ανάγκες των καταναλωτών και η εξέλιξη των υλικών υποδομών και του λογισμικού οδήγησαν στην ανάπτυξη και στη βελτίωση της προδιαγραφής του Bluetooth.



Εικόνα 2.3 Μετάδοση δεδομένων

Η συνεχής και ραγδαία εξέλιξη των προϊόντων των δικτύων 3G και 4G και των κυψελωτών συστημάτων δημιουργούν νέες εμπορικές ευκαιρίες για την τεχνολογία του Bluetooth.

2.4 Τα οφέλη από την επιλογή του Bluetooth

Παρακάτω, παρουσιάζονται τα δέκα σημαντικότερα πλεονεκτήματα και οι λόγοι για την χρησιμοποίηση της τεχνολογίας του Bluetooth.

Ασύρματη δικτύωση. Η εξάλειψη των καλωδίων στην δικτύωση των συσκευών και η βελτίωση των ζητημάτων που προέκυψαν στην ασφάλεια από την εξάλειψη των καλωδίων.

Η τεχνολογία του Bluetooth είναι οικονομική. Η κατασκευή των Bluetooth chips από τις εταιρείες είναι οικονομική και αυτό έχει ως αποτέλεσμα στη μείωση του κόστους από τις εταιρείες των παρόχων στη διάθεση της τεχνολογίας αυτής στην αγορά. Η εξοικονόμηση αυτή περνά από την εταιρεία στους πελάτες.

Εύκολη διαδικασία σύνδεσης. Όταν δύο ή περισσότερες συσκευές βρεθούν εντός της περιοχής της εμβέλειας (10 μέτρα) θα αρχίσει αυτόματα η επικοινωνία τους χωρίς να απαιτείται κάποια πολύπλοκη διαδικασία για την σύνδεση τους.

Τυποποιημένο πρωτόκολλο. Το πρότυπο της ασύρματης δικτύωσης του Bluetooth εγγυάται ένα υψηλό επίπεδο συμβατότητας μεταξύ των συσκευών δηλαδή μπορούν να συνδεθούν συσκευές μεταξύ τους, ακόμα και αν δεν είναι το ίδιο μοντέλο.

Μείωση παρεμβολών. Οι συσκευές που υποστηρίζουν το πρότυπο του Bluetooth κάνουν διαμόρφωση του αρχικού σήματος με την τεχνική της διασποράς του φάσματος με την αναπήδηση της συχνοτήτων (frequency hopping) και η μετάδοση των δεδομένων πραγματοποιείται με σήματα χαμηλής ισχύος. Όλα αυτά συμβάλουν στην αποφυγή των παρεμβολών που δημιουργούνται από τις άλλες ασύρματες συσκευές.

Χαμηλή κατανάλωση ενέργειας. Η μετάδοση των σημάτων χαμηλής ισχύος έχει ως αποτέλεσμα η τεχνολογία του Bluetooth να καταναλώνει για τη λειτουργία της πολύ λίγη ενέργεια δηλαδή η μπαταρία έχει μεγαλύτερη διάρκεια ζωής. Αυτό είναι ένα πλεονέκτημα για τις φορητές συσκευές.

Κοινή χρήση φωνής και δεδομένων. Το πρότυπο του Bluetooth επιτρέπει στις συμβατές συσκευές την ανταλλαγή των δεδομένων και της φωνής.

Άμεση εγκαθίδρυση PAN (Personal Area Network) δικτύου. Μπορούν να συνδεθούν έως και επτά συσκευές Bluetooth μεταξύ τους, σχηματίζοντας ένα piconet ή PAN.

Δυνατότητα αναβάθμισης. Παρέχεται η δυνατότητα της αναβάθμισης των παλιότερων εκδόσεων του Bluetooth με τις νεότερες εκδόσεις, οι οποίες προσφέρουν πολλά νέα πλεονεκτήματα και συμβατότητα με παλαιότερες εκδόσεις.

Παγκόσμιο πρότυπο ασύρματης επικοινωνίας. Το πρότυπο της ασύρματης επικοινωνίας του Bluetooth λειτουργεί στις ίδιες συχνότητες σε όλες τις περιοχές του πλανήτη, για το λόγο αυτό είναι μια τόσο δημοφιλής τεχνολογία.

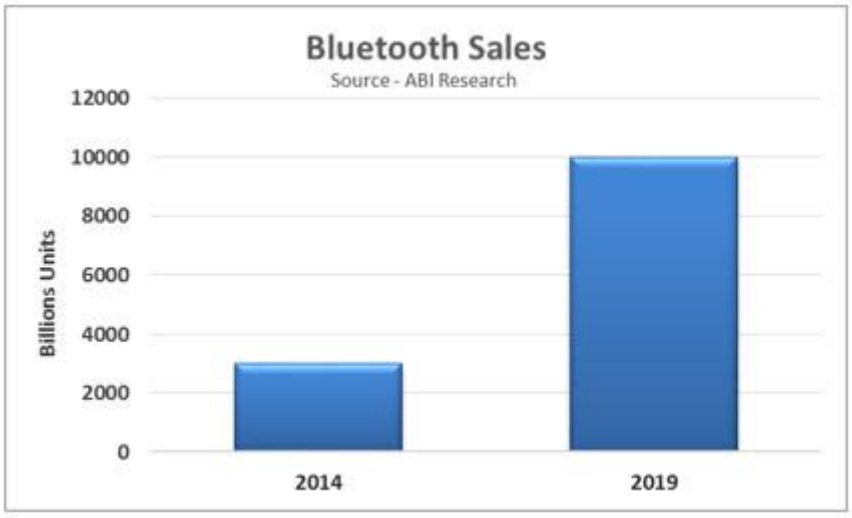
Εικόνα 2.4 Bluetooth world

2.5 Εμπορικές προοπτικές

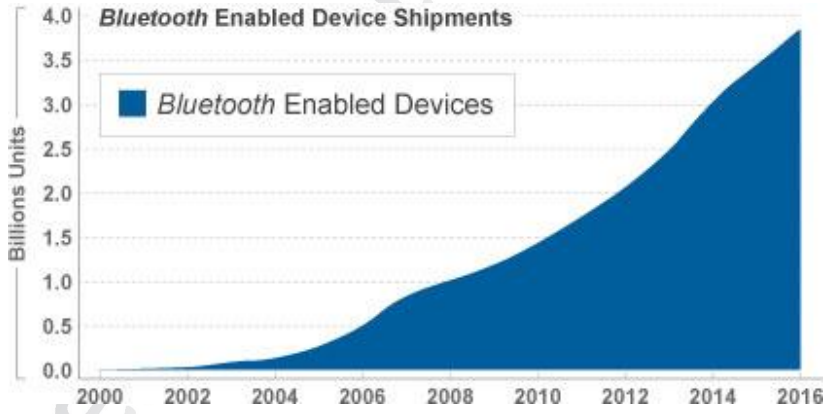
Η ανάπτυξη της τεχνολογίας, η αύξηση της παραγωγής των εμπορικών προϊόντων όπως είναι τα PC's, τα laptops και τα έξυπνα κινητά τηλέφωνα, η συνεχής ελάττωση του κόστους και του μεγέθους αυτών των συσκευών οδήγησε στην αύξηση της δημοτικότητα τους. Η ανάγκη της δικτύωσης αυτών των συσκευών καλύπτεται από την τεχνολογία του Bluetooth η οποία καταργεί τις καλωδιώσεις και παρέχει το μέσο για τη σύνδεση των μονάδων μεταξύ τους, δημιουργώντας μικρά PAN δίκτυα..

Η ομάδας της SIG του Bluetooth το 2005 αριθμούσε περίπου 2500 μέλη εταιρείες, ενώ συμφώνα με πρόσφατη ανακοίνωση της η ομάδα αυτή σήμερα έχει ξεπεράσει τις 25.000 εταιρείες μέλη παγκοσμίως. Η ραγδαία εξέλιξη των προϊόντων και οι καινοτόμες ιδέες των μελών της ομάδας της SIG τροφοδοτούν την έκρηξη της τεχνολογίας του Bluetooth στην παγκόσμια αγορά. Πάνω από 3 δισεκατομμύρια προϊόντων Bluetooth διατέθηκαν στην αγορά το προηγούμενος έτος (2014) ενώ συμφώνα με την ίδια ερεύνα της εταιρείας ABI Research οι πωλήσεις των προϊόντων που υποστηρίζουν την τεχνολογία του Bluetooth αναμένεται να φτάσουν τα 10

δισεκατομμύρια το 2019 (Διάγραμμα 2.1 Παγκόσμιες πωλήσεις συσκευών που υποστηρίζουν το Bluetooth). Ο μέσος ετήσιος ρυθμός αύξησης (compound annual growth rate - CAGR) των πωλήσεων υπολογίζεται στα 35,1% σε σχέση με την εν λόγω χρονική περίοδο.

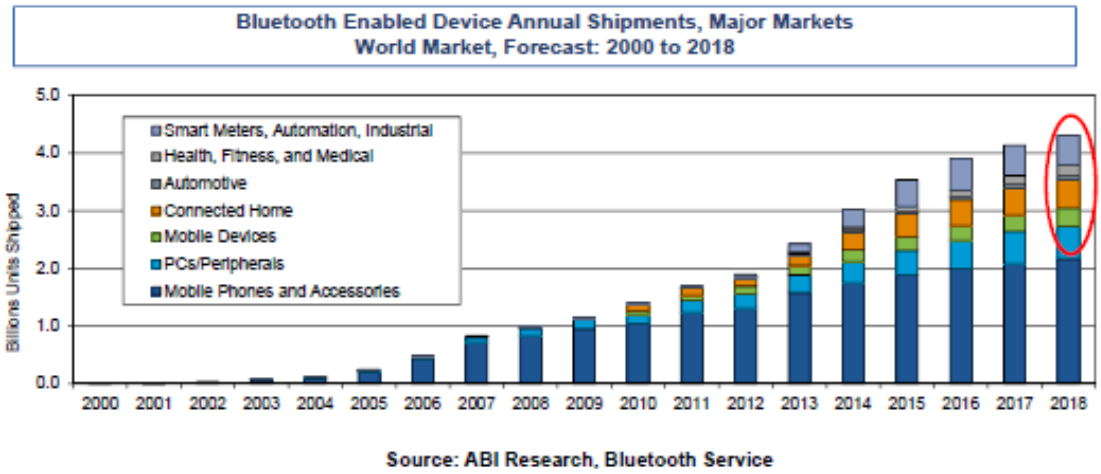


Διάγραμμα 2.1 Παγκόσμιες πωλήσεις συσκευών που υποστηρίζουν το Bluetooth[19]



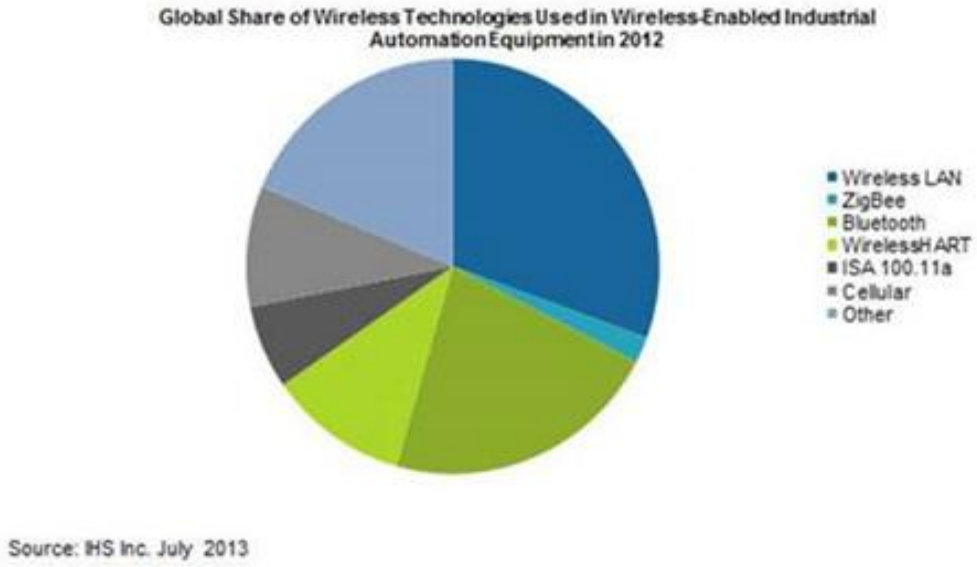
Διάγραμμα 2.2 Διάθεση συσκευών με δυνατότητα Bluetooth [20]

Συμφώνα με τα στατιστικά στοιχεία της εταιρείας *ABI Research* (Διάγραμμα 2.2) τα οποία παρουσιαστήκαν στην Σαγκάη το 2013, μέχρι το τέλος του 2012 είχαν διατεθεί στην παγκόσμια αγορά περισσότερα από 9 δισεκατομμύρια συσκευές που υποστηρίζουν την τεχνολογία του Bluetooth, ενώ από την ίδια ερεύνα προβλεπόταν ότι ως το τέλος του 2014 θα αποστέλλονταν ακόμα 3 δισ. συσκευές. Η παρουσίαση και η διάθεση στην αγορά της έκδοσης 4.0 του Bluetooth δημιούργησε νέες ευκαιρίες και οι συνολικές πωλήσεις των συσκευών αυτών αναμένεται να αυξηθούν κατά 2,9 δισ. ευρώ ετησίως μέχρι το 2016.



Διάγραμμα 2.3 Ανάπτυξη της αγοράς του Bluetooth [21]

Τον Μαρτίου του 2013 παρουσιάστηκαν τα αποτελέσματα της έρευνας με θέμα την ανάπτυξη της ετήσιας διάθεσης των συσκευών που παρέχουν την τεχνολογία του Bluetooth στην παγκόσμια αγορά που διεξήχθη από την εταιρεία *ABI Research* (Διάγραμμα 2.3). Συμφώνα με την ερεύνα αυτή υπολογίστηκε ότι οι συσκευές που παρείχαν τη δυνατότητα σύνδεσης με Bluetooth οι οποίες είχαν μέχρι τότε πουληθεί ήταν απαριθμούσαν τα 10 δισεκατομμύρια. Από την ίδια ερεύνα προβλέπεται ότι οι συνολικές πωλήσεις θα υπερβούν τα 20 δισεκατομμύρια συσκευές μέχρι το 2016 και τα 30 δισεκατομμύρια συσκευές μέχρι το 2018.

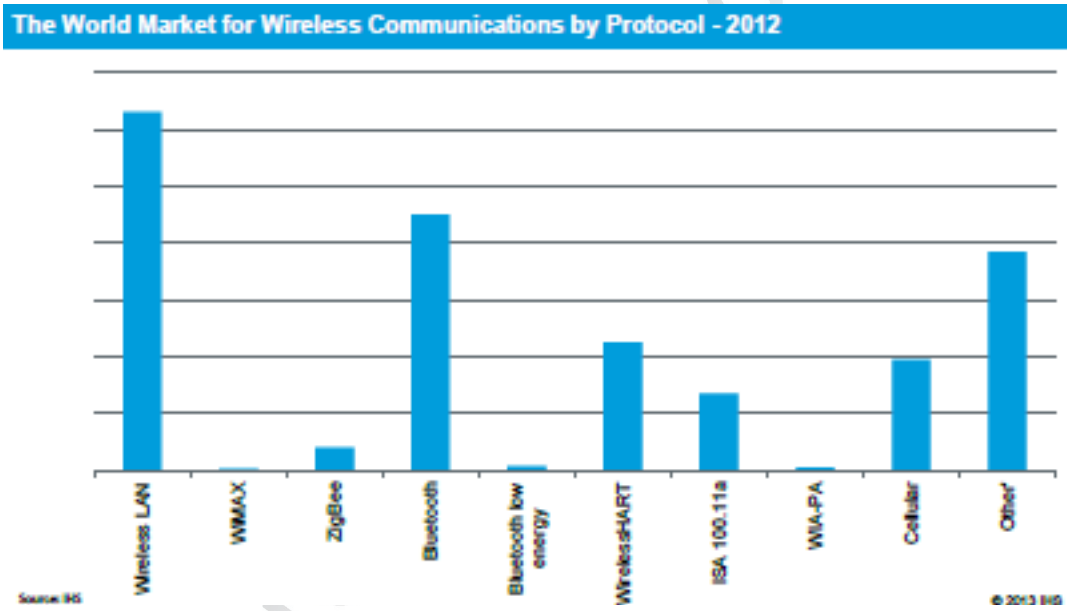


Διάγραμμα 2.4 Το παγκόσμιο μερίδιο των ασύρματων τεχνολογιών στην αγορά των Βιομηχανικών Εξοπλισμών Αυτοματισμού με Ασύρματη-δικτύωση [22]

Το μέγεθος της επικράτησης της τεχνολογίας του Bluetooth (χρωματισμός με έντονο πράσινο) στην αγορά των Βιομηχανικών Εξοπλισμών Αυτοματισμού με

Ασύρματη δικτύωση (Διάγραμμα 2.4) διαφαίνεται από την στατιστική έρευνα που διεξήχθη από την εταιρεία IHS Inc. η οποία δημοσιεύτηκε τον Ιούλιο του 2013.

Το ενδιαφέρον για την επικράτησης των τεχνολογιών της ασύρματης δικτύωσης ήταν αρκετά μεγάλο από τα προηγούμενα χρόνια. Σύμφωνα με την έκθεση που εξήχθη το 2012 από την IMS Research για λογαριασμό του IHS (NYSE: IHS) με τίτλο "The World Market for Wireless Communications in Factory and Process Automation" [22], το 2012 το πρωτόκολλο του WLAN έφτανε σε πωλήσεις το ποσοστό του 31 % των συνολικών πωλήσεων του βιομηχανικού εξοπλισμού αυτοματισμών με ασύρματη-δικτύωση και το πρωτόκολλο του Bluetooth κατείχε το ποσοστό του 22 % της παγκόσμιας αγοράς.



Διάγραμμα 2.5 Η παγκόσμια αγορά των ασύρματων επικοινωνιών το 2012 [23]

Το παραπάνω διάγραμμα επιτρέπει τη σύγκριση των ασύρματων πρωτοκόλλων του κάθε προϊόντος. Παρατηρώντας το διαφαίνεται ξεκάθαρα η επικράτηση των δύο τεχνολογιών που αναφέραμε παραπάνω έναντι των υπόλοιπων τεχνολογιών που διεκδικούν το δικό τους μερίδιο στην αγορά των ασύρματων επικοινωνιών, από το 2012. [4] [24]

2.6 Το Bluetooth αποτελεί μια καλή επιλογή για την ανάπτυξη του project

Η επιλογή του προτύπου του Bluetooth για την υλοποίηση του project που αναπτύσσουμε έγινε λαμβάνοντας υπόψη τα ακόλουθα σημεία:

- Τις ελάχιστες διαστάσεις του Hardware (ευχρηστία).
- Τη δωρεάν διάθεση, χαμηλό κόστος των εξαρτημάτων του Bluetooth.
- Τις χαμηλές ενεργειακές απαιτήσεις για την σύνδεση και τη λειτουργία του Bluetooth
- Την αυξημένη ασφάλεια.
- Το μέτριο εύρος.

Το χαμηλό κόστος και το μικρό μέγεθος του Bluetooth το κάνουν μια δημοφιλή τεχνολογία με μεγάλη αποδοχή από τους κατασκευαστές και τους καταναλωτές. Η χαμηλή κατανάλωση ενέργειας είναι ιδιαίτερα σημαντική για το συγκεκριμένο project γιατί η εφαρμογή μπορεί να βρίσκεται σε λειτουργία για μεγάλη χρονική διάρκεια διατηρώντας την ασύρματη σύνδεση μέσω του δικτύου του Bluetooth. Αυτό αυξάνει τον κύκλο ζωής της μπαταρίας. Η ασφάλεια που παρέχει η προδιαγραφή του Bluetooth είναι ικανοποιητική για τις ανάγκες της εφαρμογής αυτής. Το σχετικά χαμηλό εύρος μετάδοσης δεδομένων που υποστηρίζει η προδιαγραφή του Bluetooth δεν αποτελεί πρόβλημα γιατί τα δεδομένα που μεταδίδονται στο project αυτό είναι σε μικρές ποσότητες δεδομένων.

2.7 Σύνοψη

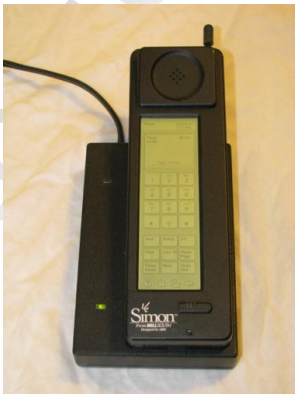
Τα μειονεκτήματα που παρουσιάζει η τεχνολογία του B.T. δεν είναι αρκετά για να οδηγήσουν αυτή την τεχνολογία στον αποκλεισμό της από την αγορά. Το αντίθετο μάλιστα, η επικράτηση του πρότυπου του B.T. έναντι των άλλων συναφών τεχνολογιών στις μέρες μας είναι ξεκάθαρη. Όλες οι έξυπνες τερματικές συσκευές πλέον παρέχουν την υπηρεσία αυτή στους χρήστες διότι αποτελεί μια οικονομική λύση σύνδεσης και ανταλλαγής δεδομένων.

3 Το λειτουργικό σύστημα κινητών τερματικών Android

Στο κεφάλαιο αυτό γίνεται αναφορά στα λειτουργικά συστήματα των κινητών τερματικών και περιγράφεται η πλατφόρμα του Android πάνω στην οποία υλοποιείται η εφαρμογή του Bluetooth chat. Επίσης γίνεται μία σύντομη αναφορά στις έξυπνες συσκευές (smartphone) των κινητών τηλεφώνων.

3.1 Η εμφάνιση των πρώτων smart phones

Το 1992 η IBM κατασκευάζει τη "Simon" ένα τηλέφωνο που διέθετε επαναφορτιζόμενη μπαταρία, είχε οθόνη αφής (touch screen), επέτρεπε την πρόσβαση σε ηλεκτρονικό ταχυδρομείο (email) και διέθετε τα χαρακτηριστικά ενός PDA. Η συσκευή αυτή είναι μια συσκευή που συνδυάζει τις λειτουργίες του κινητού τηλεφώνου και του PDA.



Εικόνα 3.1 Η Simon πάνω στη βάση φόρτισης της. [25]

Αν και η συσκευή Simon είχε κατασκευαστεί από το 1992 εντούτοις ο όρος του έξυπνου τηλεφώνου «smartphone» πρωτοαναφέρεται από την Ericsson το 1997 όταν παρουσιάζει τη συσκευή GS 88 με την ονομασία "Penelope".

Η διάκριση μεταξύ των smartphones και των χαρακτηριστικών των τηλεφώνων είναι ασαφής και δεν υπάρχει επίσημος ορισμός για το τι συνιστά τη διαφορά μεταξύ τους. Μία από τις πιο σημαντικές διαφορές είναι ότι κατασκευάζετα ένα πλήθος από προηγμένες εφαρμογές διεπαφών (APIs) των smartphones για την εκτέλεση τρίτων εφαρμογών, οι οποίες μπορούν να έχουν καλύτερη ενσωμάτωση με το λειτουργικό σύστημα και το υλικό των τηλεφώνων από την ενσωμάτωση των εφαρμογών που παρουσιάζουν με τα συνηθισμένα τηλεφώνα. [26], [25]

3.2 Ιστορικά στοιχεία και οικονομική εξέλιξη του Android

Η εταιρία Android ιδρύθηκε τον Οκτώβριο του 2003 στο Πάλο Άλτο της Καλιφόρνιας από τους Andy Rubin, Rich Miner, Nick Sears και Chris White. Σκοπός της εταιρείας ήταν σύμφωνα με τα λόγια του Rubin, η ανάπτυξη, "έξυπνων" κινητών

Εικόνα 3.2 Το λογότυπο "Android" της εταιρείας

συσκευών που θα έχουν μεγαλύτερη επίγνωση για τη θέση και τις προτιμήσεις του ιδιοκτήτη τους».

Στις 17 Αυγούστου του 2005 η Google εξαγόρασε την εταιρεία Android και την έκανε θυγατρική εταιρεία της. Οι Rubin, Miner και White παρέμειναν στην εταιρεία της Android Inc μετά την εξαγορά της. Με αυτή την κίνηση η Google σχεδίασε την είσοδο της στην αγορά της κινητής τηλεφωνίας και μία ομάδα με επικεφαλή τον Rubin ανέπτυξε μια πλατφόρμα για κινητές συσκευές που λειτουργούσε πάνω στον πυρήνα του Linux.

Εικόνα 3.3 Το εμπορικό σήμα Android Robot της εταιρείας

Στις 5 Νοεμβρίου του 2007, ιδρύεται η κοινοπραξία εταιρειών τεχνολογίας Open Handset Alliance (OHA) [27] η οποία αποτελούνταν από 84 εταιρείες αρχικά, μεταξύ των οποίων ήταν οι εταιρείες Google, HTC, Sony, Samsung, Sprint Nextel, T-Mobile, Qualcomm και της Texas Instruments. Στο δελτίο τύπου που ανήρτησε στην ιστοσελίδα της (http://www.openhandsetalliance.com/press_110507.html) η κοινοπραξία εταιρειών OHA υποστήριζε ότι στόχο της ήταν η ανάπτυξη ανοικτών προτύπων για τις φορητές συσκευές.

Εικόνα 3.4 Μερικοί εκ των εταίρων της κοινοπραξίας Open Handset Alliance

Η OHA παρουσιάσε την ίδια ημέρα το πρώτο προϊόν της, την 1η πλατφόρμα Android η οποία ήταν ένα ευέλικτο σύστημα με δυνατότητα αναβάθμισης. Η Google παράλληλα παρήγαγε μια σειρά από εξαρτήματα hardware και εφαρμογές λογισμικού κερδίζοντας έτσι αρκετές συνεργασίες. Το πρώτο διαθέσιμο για πώληση τηλέφωνο

που χρησιμοποιούσε το λειτουργικό σύστημα Android ήταν το HTC Dream, που κυκλοφόρησε στις 22 Οκτωβρίου 2008.

Το 2010, η Google εγκαινίασε τη σειρά συσκευών Nexus (smartphones -Nexus One και tablets) η οποία χρησιμοποιούσε στο λειτουργικό σύστημα Android.

Από το 2008 και μετά το Λ.Σ Android έχει δεχθεί πληθώρα αναβαθμίσεων διαθέτοντας στην αγορά καινούργιες εκδόσεις με αποτέλεσμα τη συνεχή βελτίωση του. Οι συσκευές των smartphones και τα tablets εξελίσσονται διαρκώς και οι απαιτήσεις του κόσμου για καλύτερη ποιότητα προϊόντων και υπηρεσιών QoS ολοένα και αυξάνονται. Όλα τα παραπάνω οδηγούν στην ραγδαία αύξηση των πωλήσεων των συσκευών αυτών.

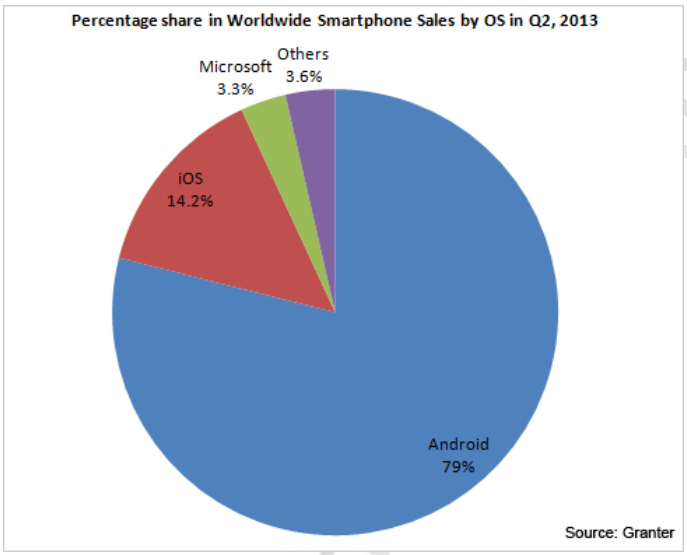
Πίνακας 3.1 Ιστορικό πωλήσεων συσκευών ανά εταιρεία (σε εκατ. μονάδες)

Year	Android (Google)	Blackberry (RIM)	iOS (Apple)	Linux (other than Android)	Palm/WebOS (Palm/HP)	Symbian (Nokia)	Asha Full Touch (Nokia)	Windows Mobile/Phone (Microsoft)	Bada (Samsung)	Other
2007		11.77	3.3	11.76	1.76	77.68		14.7		
2008		23.15	11.42	11.26	2.51	72.93		16.5		
2009	6.8	34.35	24.89	8.13	1.19	80.88		15.03		
2010	67.22	47.45	46.6			111.58		12.38		
2011	219.52	51.54	89.26			93.41		8.77		14.24
2012-Q1	81.07	9.94	33.12			12.47		2.71	3.84	1.24
2012-Q2	104.8	7.4	26.0	3.5		6.8		5.4		0.1
2012-Q3	122.5	9.0	23.6			4.4	6.5	4.1	5.1	0.7
2012-Q4	144.7	7.3	43.5			2.6	9.3	6.2	2.7	0.7
2013-Q1	162.1	6.3	37.4			__	--	7.0	--	--
2013-Q2	177.9	6.2	31.9			.631	--	7.4	.838	.471

Αυτή η έκρηξη της αγοράς απεικονίζεται στον παραπάνω πίνακα στον οποίο διαφαίνεται η επικράτηση του Android και της Google έναντι των άλλων εταιρειών. Οι τιμές των πωλήσεων δίνονται ανά τρίμηνο (Q1, Q2 κλπ) για τα έτη 2012 και 2013, πράγμα που μας επιτρέπει την σύγκριση των πωλήσεων συμφώνα με το αντίστοιχο τρίμηνο.

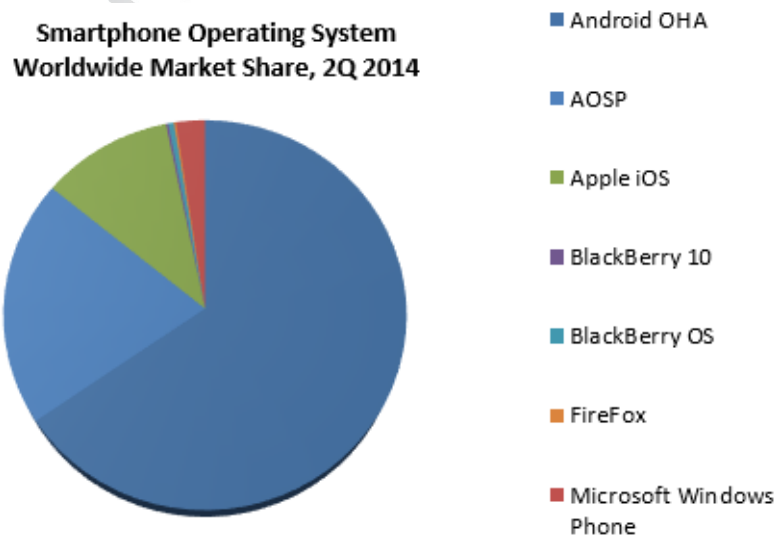
Στο διάγραμμα 3.1 φαίνεται το μερίδιο των πωλήσεων των Smartphones (σε τελικούς χρήστες) που αντιστοιχούσε στην κάθε εταιρεία κατά το δεύτερο τρίμηνο του 2013 από την παγκόσμια αγορά (πηγή: "Market Share Analysis: Mobile Phones, Worldwide, 2Q13″ της Gartner Inc. (NYSE: IT)). Οι πωλήσεις των συσκευών με τα

Λ.Σ. του Android και του iOS εξακολουθούν να ηγούνται της αγοράς με ποσοστό μεριδίου 79% και 14,2% αντίστοιχα. Η εκτίμηση των αναλυτών ήταν ότι οι πωλήσεις κατά το δεύτερο τρίμηνο του 2013 θα ήταν αυξημένες εντούτοις η σταδιακή μείωση των τιμών των smart phones κατά το δεύτερο εξάμηνο του 2013 και των επόμενων ετών θα αυξήσει την αγοραστική κίνηση.



Διάγραμμα 3.1 Πωλήσεις Smartphones παγκοσμίως κατά το 2ο τρίμηνο του 2013

Η εταιρεία ερευνών ABI Research αναφέρει σε έρευνα της (Διάγραμμα 3.2) ότι οι παγκόσμιες πωλήσεις των Smartphones που χρησιμοποιούν τα Λ.Σ. του Android OHA και του Android Open Source Project (AOSP) το δεύτερο τρίμηνο του 2014 (2Q 2014) αυξήθηκαν κατά 20% σε σχέση με το πρώτο τρίμηνο.



Διάγραμμα 3.2 Οι παγκόσμιες πωλήσεις των Smartphones το 2ο τρίμηνο του 2014

Η κοινοπραξία των εταιρειών τεχνολογίας της OHA με το Λ.Σ. Android εξακολουθεί να ηγείται με ποσοστό 65% της παγκόσμιας αγοράς. Συνολικά οι πωλήσεις των συσκευών Android OHA και AOSP έχουν ξεπεράσει τα 278 εκατομμύρια συσκευές καλύπτοντας έτσι το εντυπωσιακό μερίδιο του 86% της παγκόσμιας αγοράς. [28], [27], [29], [30]

3.3 Τι είναι το Android?

Το Android είναι μια ανοιχτή και ελεύθερη πλατφόρμα για κινητά τηλέφωνα η οποία περιλαμβάνει ένα λειτουργικό σύστημα που τρέχει πάνω στον πυρήνα του Linux, το απαραίτητο ενδιάμεσο λογισμικό, τις βιβλιοθήκες και τις βασικές εφαρμογές που παρέχονται από την Google. Έχει σχεδιαστεί κυρίως για τις φορητές συσκευές με οθόνη αφής όπως είναι τα smartphones και τα tablets. Ο κώδικας αυτός είναι δημοσιευμένος στον ιστότοπο του Android (http://source.android.com/) και είναι διαθέσιμος σε όλους,

Εικόνα 3.5 Android

Το android sdk παρέχει δωρεάν στους προγραμματιστές τα απαραίτητα εργαλεία για να την ανάπτυξη προγραμμάτων σε γλώσσα προγραμματισμού Java. Συμφώνα με την εταιρεία Android για τη διαθεσιμότητα του κώδικα αναφέρεται μεταξύ άλλων ότι "Δημιουργήσαμε το Android στο πλαίσιο των δικών μας εμπειριών για τις εφαρμογές των κινητών τηλεφώνων. Θέλαμε να βεβαιωθούμε ότι δεν υπάρχει κομβικό σημείο αποτυχίας, ώστε να μπορεί μια βιομηχανία να ορίζει ή να ελέγχει τις καινοτομίες κάποιου άλλου. Γι 'αυτό δημιουργήσαμε το Android και κάναμε τον πηγαίο κώδικα του ανοιχτό." Πηγή http://source.android.com/ [31], [32]

3.4 Οι εκδόσεις του Android

Η αρχική πλατφόρμα του Android έχει δεχτεί πολλές ενημερώσεις μέχρι να φτάσει στην σημερινή της μορφή. Η νέα έκδοση προκύπτει μετά από αναβαθμίσεις που πραγματοποιούνται στην προηγούμενη έκδοση, προστίθενται νέα χαρακτηριστικά και

βελτιώνονται τα προϋπάρχοντα χαρακτηριστικά, επιλύονται προβλήματα και αστοχίες ή ελλείψεις που παρουσιάζει ο κώδικας.

Εικόνα 3.6 Android Version Code LOLLIPOP

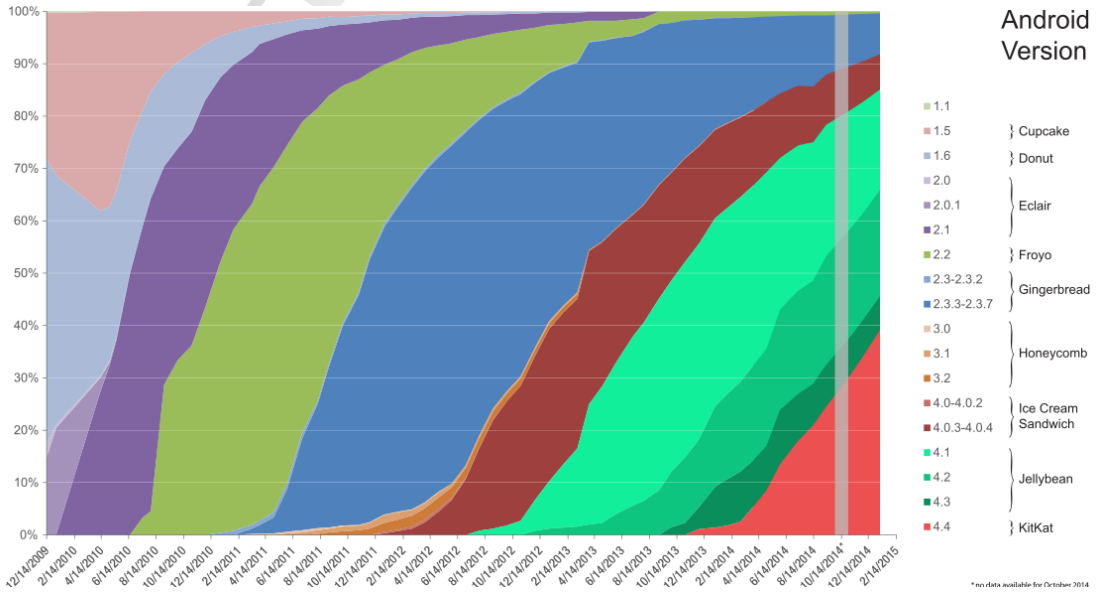
Σε κάθε ενημέρωση του Λ.Σ. του Android αντιστοιχεί μία ονομασία της έκδοσης (Version Code) που βασίζεται σε κάποιο επιδόρπιο, το αρχικό γράμμα της ονομασίας της νέας έκδοσης είναι το επόμενο γράμμα της αλφαβήτου από το πρώτο γράμμα της ονομασίας της προηγούμενης έκδοσης.

Πίνακας 3.2 Οι εκδόσεις του λειτουργικού συστήματος Android

Πλατφόρμα	Επίπεδο API	Έκδοση (VERSION_CODE)
Android 5.0	20	LOLLIPOP
Android 4.4	19	KITKAT
Android 4.3	18	JELLY_BEAN_MR2
Android 4.2, 4.2.2	17	JELLY_BEAN_MR1
Android 4.1, 4.1.1	16	JELLY_BEAN
Android 4.0.3, 4.0.4	15	ICE_CREAM_SANDWICH_MR1
Android 4.0, 4.0.1, 4.0.2	14	ICE_CREAM_SANDWICH
Android 3.2	13	HONEYCOMB_MR2
Android 3.1.x	12	HONEYCOMB_MR1

Android 3.0.x	11	HONEYCOMB
Android 2.3.4, 2.3.3	10	GINGERBREAD_MR1
Android 2.3, 2.3.1, 2.3.2	9	GINGERBREAD
Android 2.2.x	8	FROYO
Android 2.1.x	7	ECLAIR_MR1
Android 2.0.1	6	ECLAIR_0_1
Android 2.0	5	ECLAIR
Android 1.6	4	DONUT
Android 1.5	3	CUPCAKE
Android 1.1	2	BETA (BASE_1_1)
Android 1.0	1	ALPHA (BASE)

Κάθε έκδοση της πλατφόρμας του Android περιλαμβάνει αντίστοιχο πλαίσιο εφαρμογών που προσδιορίζεται απο έναν ακέραιο που ονομάζεται "API Level". Παρέχεται η δυνατότητα ενημέρωσης της έκδοση της πλατφόρμας του Android. Οι ενημερώσεις των εκδόσεων της πλατφόρμας του Android έχουν σχεδιαστεί με τέτοιο τρόπο ώστε να εξασφαλίζεται η συμβατότητα της νεότερης έκδοσης με τις προηγούμενες εκδόσεις.



Εικόνα 3.7 Η παγκόσμια κατανομή των εκδόσεων του android

Στο διάγραμμα 3.7 παρουσιάζεται η παγκόσμια κατανομή των εκδόσεων του Λ.Σ. του Android από το Δεκέμβριο του 2009 μέχρι και σήμερα. Σύμφωνα με τα παραπάνω στοιχεία από τον Ιανουάριο του 2015, η έκδοση "KitKat" της πλατφόρμας 4.4 του Λ.Σ. Android είναι η πιο ευρέως χρησιμοποιούμενη έκδοση όλων των εκδόσεων του Android, έχει εγκατασταθεί στο ποσοστό του 39,1% του συνόλου των συσκευών με Λ.Σ. Android σε όλο τον κόσμο. Οι εκδόσεις του Android "Jelly Bean" (4.1-4.3.1) κατέχουν το μεγαλύτερο μερίδιο της παγκόσμιας αγοράς το οποίο ανέρχεται στο ποσοστό του 46% των συσκευών με Λ.Σ. Android. [33]

3.5 Τα χαρακτηριστικά του Λ.Σ. του Android

Τα γενικά χαρακτηριστικά του Λ.Σ. Android έγιναν η αιτία της ραγδαίας διάδοσης του και της επικράτησης του στην παγκόσμια αγορά. Η δομή των εφαρμογών είναι τέτοια που επιτρέπει την επαναχρησιμοποίηση των τμημάτων του κώδικα. Η γλώσσα προγραμματισμού της java χρησιμοποιείται για την ανάπτυξη των Android εφαρμογών.

Η ενσωμάτωση του περιηγητή (browser) που βασίζεται στην μηχανή Blink (σε παλιότερες εκδόσεις στο WebKit) επιτρέπει την πλοήγηση στο διαδίκτυο. Η ανάπτυξη των Βάσεων Δεδομένων (Β.Δ.) γίνεται με τη γλώσσα SQLite η οποία θεωρείται «ελαφρύτερη» από την κλασσική γλώσσα SQL στη διαχείριση των Β.Δ.

Το Λ.Σ. του Android, υποστηρίζει τα πολυμέσων για τους τύπους των αρχείων του ήχου, της εικόνας και του βίντεο. Υποστηρίζει την υπηρεσία του streaming media και τις εξωτερικές πηγές (π.χ. SD card, USB flash drives και USB HDDs). Επιτρέπει την ταυτόχρονη εκτέλεση πολλών εργασιών multitasking στις εφαρμογές, το αποτύπωμα της οθόνης screen capture, τις βιντεοκλήσεις, είναι πολυγλωσσικό, υποστηρίζει την οθόνη αφής και είναι προσιτό σε ανθρώπους με προβλήματα όρασης.

Ως προς τη συνδεσιμότητα το Android είναι συμβατό με τις τεχνολογίες επικοινωνίας συμπεριλαμβανομένων των GSM/EDGE, Wi-Fi, Bluetooth, LTE, CDMA, EV-DO, UMTS, NFC, IDEN και WiMAX. Το Android υποστηρίζει την υπηρεσία του tethering, η οποία επιτρέπει σε ένα τηλέφωνο να χρησιμοποιηθεί ως ασύρματο/ενσύρματο Wi-Fi hotspot.

Οι συσκευές Android υποστηρίζουν υλικό (hardware) όπως είναι οι βίντεοκάμερες, οι οθόνες αφής, το GPS, τα γυροσκόπια, οι αισθητήρες πίεσης, τα θερμόμετρα, η φωτογραφική μηχανή, η πυξίδα, ο μετρητής επιτάχυνσης και άλλα. Τα γραφικά είναι βελτιωμένα και παρέχονται χάρη στην ειδική βιβλιοθήκη των δισδιάστατων (2D) γραφικών, ενώ παράλληλα τα τρισδιάστατα γραφικά (3D) βασίζονται στις προδιαγραφές του OpenGL ES 1.0.

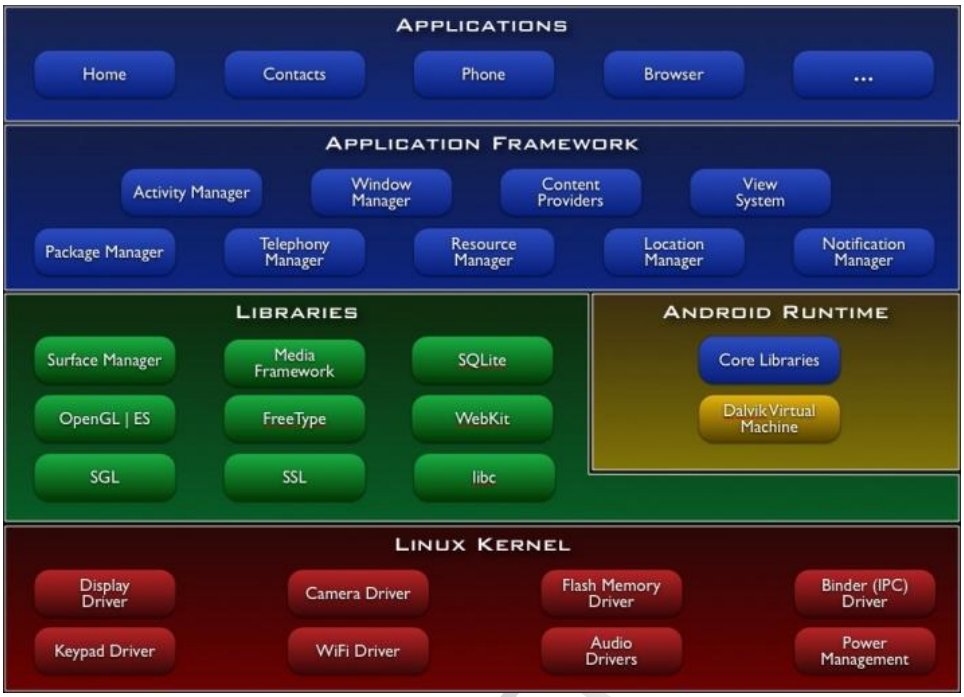
Όλα τα παραπάνω χαρακτηριστικά καθώς και το πλούσιο περιβάλλον προγραμματισμού δικαιολογούν την μέχρι σήμερα επικράτηση της πλατφόρμας του Android στην παγκόσμια αγορά και κάνουν το Android ένα ισχυρό εργαλείο. [31]

3.6 Η αρχιτεκτονική του Λ.Σ. Android

Το Android **δεν** είναι το υλικό μέρος, hardware, είναι μια στοίβα λογισμικού που χρησιμοποιείται από τις έξυπνες συσκευές κινητής τηλεφωνίας και τα tablets.

Τι περιλαμβάνει όμως αυτή η στοίβα; Η στοίβα αυτή είναι αποτελείται από το λειτουργικό σύστημα (η πλατφόρμα πάνω στην οποία "τρέχουν" όλα), το ενδιάμεσο λογισμικό (που επιτρέπει στις εφαρμογές να συνδέονται με το δίκτυο και να επικοινωνούν μεταξύ τους) και τις βασικές εφαρμογές (τα προγράμματα που τρέχει το κινητό τερματικό). Το λειτουργικό σύστημα Android αποτελείται από 3 επίπεδα, θεωρώντας το Android Runtime (ART) μέρος του ενδιάμεσου λογισμικού. Το Android runtime (ART) εκτελεί τη διαχείριση του χρόνου εκτέλεσης των εφαρμογών και των υπηρεσιών του συστήματος του Android και αποτελείται από τον διερμηνέα Dalvik Virtual machine και τις βιβλιοθήκες του πυρήνα, Core Java libraries. Ο διερμηνέας ή αλλιώς η εικονική μηχανή της Java, Dalvic (Dalvic Virtual Machine), χρησιμοποιείται από τις παλιότερες εκδόσεις του Λ.Σ. του Android για την εκτέλεση του κώδικα (βλέπε παράγραφο 3.6.2 Το περιβάλλον εκτέλεσης εφαρμογών του Android Runtime). Ο διερμηνέας Dalvic έχει χαμηλές απαιτήσεις σε μνήμη και σχεδιάστηκε για να επιτρέπει την ταυτόχρονη εκτέλεση πολλών εικονικών μηχανών (Virtual Machine ή VM) και για να εκτελεί τα Dalvik Executables (DEX) τα οποία είναι συμπιεσμένα μέσα στο Android Package (APK) και τα οποία χρησιμοποιούνται για τη λειτουργία των εφαρμογών σε συσκευές με περιορισμένους πόρους. Ο χρόνος εκτέλεσης, Android Runtime, παρουσιάστηκε στην έκδοση 4.4 Kit Kat και

χρησιμοποιείται ως διερμηνέας για τις νεότερες εκδόσεις του Λ.Σ. του Android.



Εικόνα 3.8 Αρχιτεκτονική του Android [28]

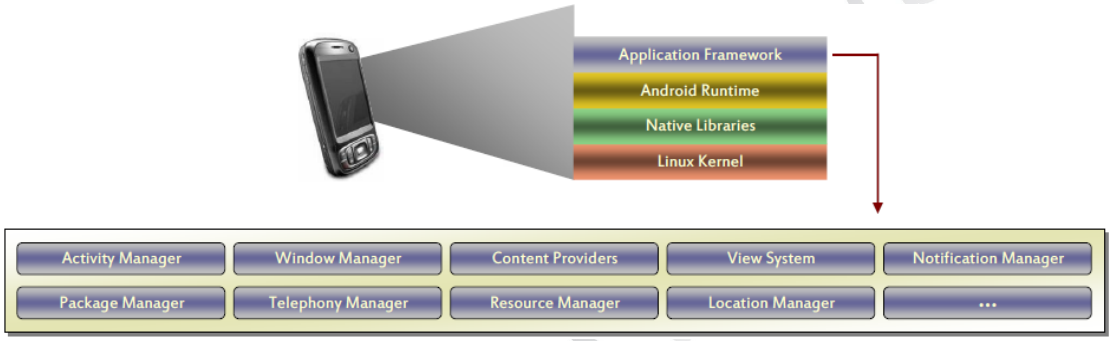
Τα ανώτερα επίπεδα χρησιμοποιούν τις υπηρεσίες που παρέχονται από το κατώτερα κάθε φορά επίπεδα. Στην παρακάτω ενότητα παρουσιάζονται τα συστατικά του κάθε επιπέδου αρχίζοντας από το χαμηλότερο επίπεδο προς το υψηλότερο. [28]

3.6.1 Οι εφαρμογές

Στο ανώτερο επίπεδο της στοίβας λογισμικού του Android βρίσκονται οι εφαρμογές (Applications ή API). Το Android διατίθεται μαζί με ένα σύνολο εφαρμογών του πυρήνα οι οποίες περιλαμβάνουν έναν client για email, το λογισμικό για την ανταλλαγή μηνυμάτων SMS, το ημερολόγιο, τους χάρτες, το πρόγραμμα περιήγησης, το βιβλίο επαφών και πολλές άλλες εφαρμογές που είναι εγκατεστημένες στην κάθε συσκευή από τους κατασκευαστές. Ο κώδικας των περισσότερων εφαρμογών είναι γραμμένος στη γλώσσα προγραμματισμού JAVA. Οι εφαρμογές είναι τα στοιχειά της στοίβας που βλέπει ο χρήστης. Ο χρήστης έχει την δυνατότητα να κατεβάσει διάφορες εφαρμογές από το Android Market.

3.6.1.1 Το πλαίσιο εφαρμογών

Κάτω από το σύνολο των εφαρμογών και μέσα στο ανώτερο επίπεδο της στοίβας βρίσκεται το πλαίσιο εφαρμογής (Application Framework). Η πλατφόρμα του Android επιτρέπει στους προγραμματιστές τη δημιουργία καινοτόμων εφαρμογων, τους παρέχει την επικοινωνία με το hardware της συσκευής, την πρόσβαση σε πληροφορίες τοποθεσίας, την εκτέλεση των υπηρεσιών στον παρασκήνιο και πολλές άλλες δυνατότητες.



Εικόνα 3.9 Application Framework [26]

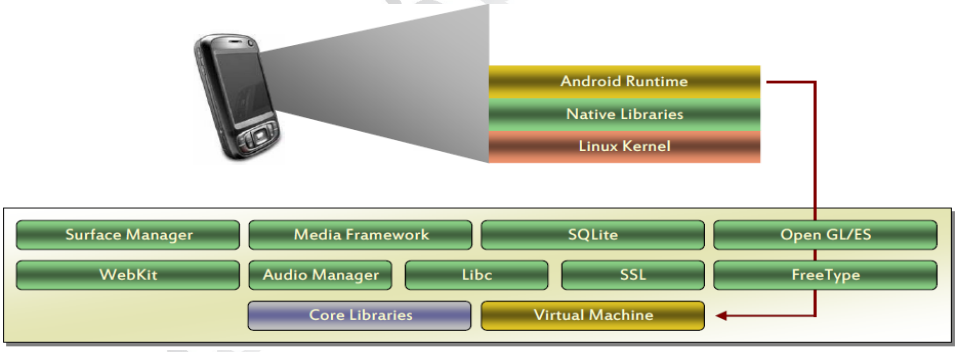
Οι προγραμματιστές έχουν πλήρη πρόσβαση στα ίδιο πλαίσιο εφαρμογών που χρησιμοποιείται από τις βασικές εφαρμογές της συσκευής. Το πλαίσιο εφαρμογών είναι εγκατεστημένο στο Λ.Σ. του Android αλλά μπορεί να εμπλουτιστεί και να ενημερωθεί αφού η αρχιτεκτονική των εφαρμογών Android έχει σχεδιαστεί με σκοπό την επαναχρησιμοποίηση των συστατικών στοιχείων των εφαρμογών αλλά και τη αντικατάσταση τους από τον χρήστη με όποια άλλα συστατικά στοιχεία αυτός επιθυμεί. Τα πιο σημαντικά συστατικά του πλαισίου εφαρμογών είναι:

- Ο Διαχειριστής Πακέτων (Package Manager) κρατάει πληροφορίες για τις εφαρμογές που υπάρχουν στο σύστημα.
- Ο Διαχειριστής Παραθύρων (Window Manager) διαχειρίζεται τα παράθυρα όλων των εφαρμογών.
- Ο View System παρέχει όλα τα προγραμματιστικά εργαλεία, widgets, για την ανάπτυξη της εφαρμογής.
- Οι Πάροχοι Περιεχομένου (Content Providers) οι οποίοι επιτρέπουν στις εφαρμογές να έχουν πρόσβαση σε άλλες εφαρμογές ή να μοιράζονται δεδομένα μεταξύ τους.

- Ο Διαχειριστής Πόρων (Resource Manager) ο οποίος παρέχει πρόσβαση στους πόρους των προγραμμάτων οι οποίοι δεν έχουν σχέση με την κωδικοποίηση. Τέτοιοι πόροι είναι οι κωδικοί χρωμάτων, οι αλφαριθμητικοί χαρακτήρες και τα έτοιμα αρχεία οθονών (XML).
- Ο Διαχειριστής Τοποθεσίας (Location Manger) ο οποίος χρησιμοποιείται για να κρατάει την τοποθεσία του τηλεφώνου.
- Ο Διαχειριστής Κοινοποιήσεων (Notification Manager) ο οποίος επιτρέπει στις εφαρμογές να εμφανίζουν διάφορες ειδοποιήσεις πάνω στην γραμμή κατάστασης της συσκευής, χωρίς να διακόπτει ο χρήστης την εργασία του.
- Ο Διαχειριστής Δραστηριοτήτων (Activity Manager) ο οποίος διαχειρίζεται τον κύκλο της ζωής των εφαρμογών. [34]

3.6.2 Το περιβάλλον εκτέλεσης εφαρμογών του Android Runtime

Στο ίδιο επίπεδο με τις εγγενείς βιβλιοθήκες συναντούμε και το περιβάλλον εκτέλεσης εφαρμογών του Android Runtime (ART). Το περιβάλλον εκτέλεσης εφαρμογών περιλαμβάνει τον διερμηνέα ή την εικονική μηχανή Dalvik (Dalvik Virtual machine) και τις βιβλιοθήκες του πυρήνα της Java (Core Java libraries).



Εικόνα 3.10 Ο διερμηνέας Dalvic (Dalvic Virtual Machine) [26]

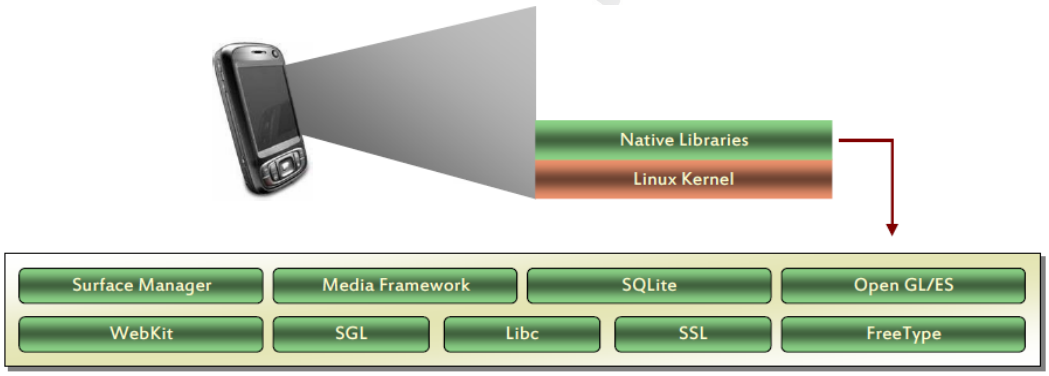
Οι βιβλιοθήκες του πυρήνα είναι μια σειρά από βασικές βιβλιοθήκες οι οποίες παρέχουν τις περισσότερες από τις διαθέσιμες λειτουργιές των βασικών βιβλιοθηκών της γλώσσας προγραμματισμού της JAVA.

Ο διερμηνέας Dalvik είναι μια βελτιστοποιημένη υλοποίηση της εικονικής μηχανής της JAVA, Java Virtual Machine (JVM) για φορητές συσκευές που παρέχεται από την Google. Ο διερμηνέας Dalvik χρησιμοποιείται σε συσκευές Android για την εκτέλεση των εφαρμογών και είναι σχεδιασμένος έτσι ώστε να

επιτρέπει την ταυτόχρονη εκτέλεση πολλαπλών εικονικών μηχανών Virtual Machines. Είναι ιδανικός στις περιπτώσεις που απαιτείται χαμηλή επεξεργαστική ισχύ και υπάρχουν μειωμένες δυνατότητες μνήμης. Πιο συγκεκριμένα εκτελεί αρχεία της μορφής Dalvik Executables (.dex) τα οποία προέρχονται από αρχεία κλάσεων .class και βιβλιοθηκών .jar τα οποία είναι συμπιεσμένα μέσα στο πακέτο της Android εφαρμογής (Android Package, APK). Τα αρχεία .dex καταλαμβάνουν ελάχιστη μνήμη. [26]

3.6.3 Οι εγγενής βιβλιοθήκες

Στο μεσαίο επίπεδο της στοίβας περιλαμβάνεται ένα σύνολο από βιβλιοθήκες (Native Libraries), γραμμένες στις γλώσσες προγραμματισμού C και C++, οι οποίες χρησιμοποιούνται από τα συστατικά του συστήματος του Android. Οι βιβλιοθήκες αυτές έχουν μεταγλωττιστεί σύμφωνα με την αρχιτεκτονική του υλικού της συσκευή.



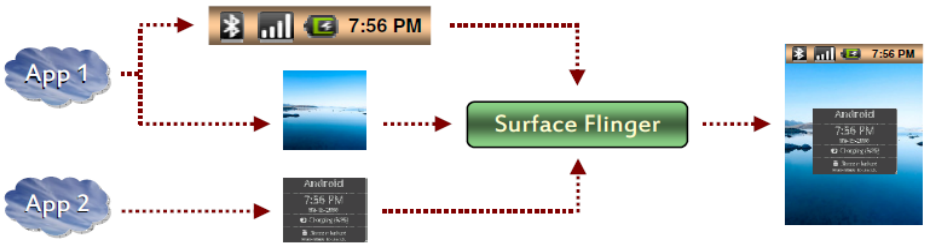
Εικόνα 3.11 Οι εγγενείς Native Libraries [26]

Οι δυνατότητες των βιβλιοθηκών αυτών δεν είναι φανερές στους χρήστες από μόνες τους, παρά μόνο όταν κληθούν από προγράμματα του πλαισίου των εφαρμογών. Οι δυνατότητες των βιβλιοθηκών είναι διαθέσιμες στους προγραμματιστές από το πλαίσιο εφαρμογών και παρέχουν την άδεια ελεύθερου λογισμικού (BSD Licence). Έχουν μικρό μέγεθος και γρήγορη απόκριση. Χρησιμοποιούνται για την υποστήριξη των εφαρμογών του Android.

Τα πιο σημαντικά συστατικά των εγγενών βιβλιοθηκών είναι:

- Ο Διαχειριστής Επιφανειών (Surface Manager) ο οποίος αναλαμβάνει τη διαχείριση της σύνθεσης των παραθύρων σε ένα πλαίσιο και την

αποθήκευσή τους σε ένα off-screen frame buffer. Μπορεί να συνδυάσει 2D και 3D γραφικά και άλλα συστατικά μέχρι να σχεδιαστεί η τελική οθόνη που προβάλλετε στον χρήστη.



Εικόνα 3.12 Surface Manager [26]

- Ο Διαχειριστής Ήχου (Audio Manager) ο οποίος επεξεργάζεται πολλαπλές ροές ήχου και τις μεταφέρει στις κατάλληλες εξόδους (PCM out paths). Μπορεί να διαχειριστεί πολλαπλούς τύπους συσκευών και να μεταφέρει τις ροές στην εκάστοτε συσκευή εξόδου, όπως φαίνεται και στην παρακάτω εικόνα.

Εικόνα 3.13 Ο Audio Manager [26]

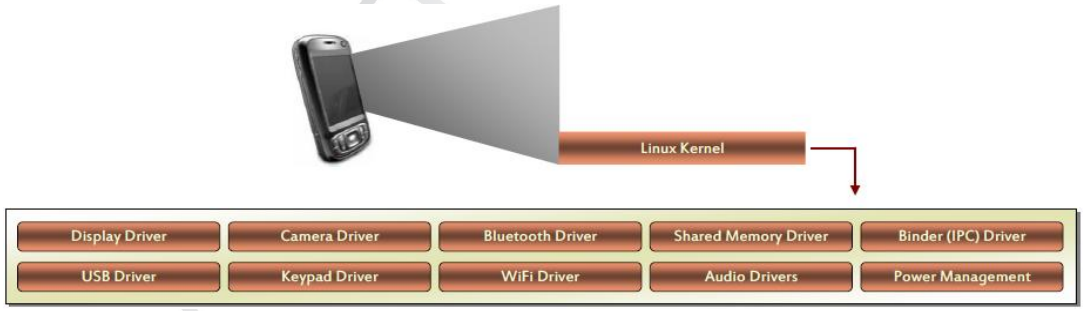
- Η γλώσσα διαχείρισης Βάσεων Δεδομένων SQLite η οποία χρησιμοποιείται για την δημιουργία, την αποθήκευση και την διαχείριση των δεδομένων σε μια σχεσιακή βάση. Η SQLite έχει λιγότερες απαιτήσεις μνήμης και πόρων και μειωμένες δυνατότητες από την SQL.
- Το πλαίσιο πολυμέσων Media Framework παρέχει διάφορους codecs πολυμέσων που επιτρέπουν την εγγραφή και την αναπαραγωγή των διαφόρων μορφών πολυμέσων (media).
- Το εργαλείο του WebKit είναι μια μηχανή φυλλομετρητή (web browser) που χρησιμοποιείται για την εμφάνιση των πολυμεσικών συστατικών HTML.
- Η βιβλιοθήκη με βελτιωμένα γραφικά 2D/3D βασισμένη στο OpenGL ES.
- Οι βιβλιοθήκες Hardware Abstraction Libraries χρησιμοποιούνται για την δημιουργία της διεπαφής όταν το Android απαιτεί την υλοποίηση "driver" για την υποστήριξη κάποιου υλικού.

Εικόνα 3.14 Οι βιβλιοθήκες Hardware Abstraction Libraries [26]

Ο προγραμματιστής έχει τη δυνατότητα να υλοποιήσει έναν αριθμό από προτυποποιημένα APIs και μπορεί να παρέμβει για να εισάγει στην πλατφόρμα όλες τις διαθέσιμες υλοποιήσεις που θεωρεί χρήσιμες. [26]

3.6.4 Πυρήνας Linux Kernel

Το Android έχει χτιστεί πάνω στον πυρήνα του λειτουργικού συστήματος Linux (έκδοση 2.6 του Linux) στον οποίο έχουν γίνει κάποιες αρχιτεκτονικές αλλαγές. Αυτό σημαίνει ότι ο πυρήνας του Λ.Σ. του Android είναι Linux, χωρίς όμως να μπορούν να εκτελεστούν πακέτα του Linux πάνω στο περιβάλλον του Android. Τα δύο αυτά Λ.Σ. είναι διαφορετικά. Ο πυρήνας του Linux αλληλεπιδρά με το μηχανικό μέρος μέσω των οδηγών (drivers) των μηχανικών εξαρτημάτων. Οι drivers είναι προγράμματα (software) που διαχειρίζονται και επικοινωνούν με το υλικό (hardware). Για παράδειγμα, οι συσκευές περιέχουν στα μηχανικά εξαρτήματα τους το chip που παρέχει την υπηρεσία του Bluetooth. Ο πυρήνας του Linux πρέπει να περιλαμβάνει ένα πρόγραμμα οδήγησης του Bluetooth για να επικοινωνεί με το chip του Bluetooth.



Εικόνα 3.15 Linux Kernel [26]

Ο πυρήνας του Linux πάνω στον οποίο δημιουργήθηκε το Λ.Σ. του Android δεν εμπεριέχει όλες τις λειτουργίες του Linux. Στο σύστημα του Android προστέθηκαν κάποιες επιπλέον λειτουργίες όπως είναι το Alarm, το "Android shared memory", ο "Kernel memory killer", ο Kernel Debugger και άλλα.

Ο πυρήνας Linux αρχικά σχεδιάστηκε για τους υπολογιστές στις μέρες μας όμως αποτελεί το βασικό συστατικό των περισσοτέρων υπερυπολογιστών αλλά και πολλών

άλλων συσκευών. Το Android επίσης χρησιμοποιεί τον πυρήνα του Linux για υπηρεσίες όπως είναι η ασφάλεια, η διαχείριση μνήμης, η διαχείριση διεργασιών, η δικτύωση και άλλες υπηρεσίες. [26]

3.7 Πλεονεκτήματα και μειονεκτήματα του Android

Παρακάτω συνοψίζονται τα πλεονεκτήματα και τα μειονεκτήματα του λειτουργικού συστήματος Android.

Τα σημαντικότερα πλεονεκτήματα του Android είναι τα ακόλουθα:

- Είναι ανοικτό, επειδή είναι βασισμένο σε Linux open source έτσι ώστε να μπορεί να αναπτυχθεί από τον κανέναν.
- Εύκολη πρόσβαση στο Android App Market: οι ιδιοκτήτες των Android συσκευών ασχολούνται πολύ με το τηλέφωνο και τους δίνεται η δυνατότητα να κατεβάσουν δωρεάν εφαρμογές από το Android App Market.
- Είναι ένα δημοφιλές λειτουργικό σύστημα. Διαφορετικό από το iOS το οποίο περιορίζεται ως προς τη συμβατότητα στο iPhone από την Apple, ενώ το Android έχει πολλούς κατασκευαστές, με κύριους την HTC και τη Samsung.
- Υποστηρίζει όλες τις υπηρεσίες της Google.
- Παρέχονται στους χρήστες ενημερωμένοι χάρτες του Android.

Τα μειονεκτήματα του Android που εντοπίζονται είναι:

- Η σύνδεση στο Internet είναι βασική προϋπόθεση για τη λειτουργία του συστήματος του Android. Για τον εντοπισμό της θέσης του χρήστη και την εμφάνιση των χαρτών πρέπει να υπάρχει σύνδεση Internet GPRS. Ο συγχρονισμός των εφαρμογών της συσκευής με τις εφαρμογές άλλων συσκευών και η ενημέρωση των εφαρμογών απαιτούν την σύνδεση με το διαδίκτυο.
- Οι βελτιώσεις του κώδικα και οι αστοχίες που εντοπίζονται στον κώδικα των εκδόσεων του Android αργούν να αντιμετωπιστούν από την Google.
- Στο Android Market εντοπίζονται μολυσμένες εφαρμογές με malware.
- Η δωρεάν διάθεση των εφαρμογών έχει ως αποτέλεσμα την προσθήκη πολλών διαφημίσεων.

- Η γρήγορη αποφόρτιση της μπαταρίας που οφείλεται στο λειτουργικό σύστημα το οποίο απασχολείτε στο παρασκήνιο με πολλές "διαδικασίες".

Το λειτουργικό σύστημα του Android είναι ανοιχτού κώδικα που σημαίνει ότι μπορεί να τροποποιηθεί ώστε να λειτουργεί σύμφωνα με τις απαιτήσεις σας χωρίς να υπάρχουν νομικά ζητήματα.

3.8 Σύνοψη

Μερικά από τα γενικά χαρακτηριστικά του Λ.Σ. android καθώς και τα πλεονεκτήματα που το χαρακτηρίζουν είναι τα στοιχεία που αποτέλεσαν το εφαλτήριο της ταχύτατης και ραγδαίας εξέλιξης του, πράγμα που δικαιολογεί την επικράτηση του με διαφορά έναντι των άλλων Λ.Σ. στην παγκόσμια αγορά τα τελευταία χρόνια.

4 Τα δομικά στοιχειά μιας Android εφαρμογής

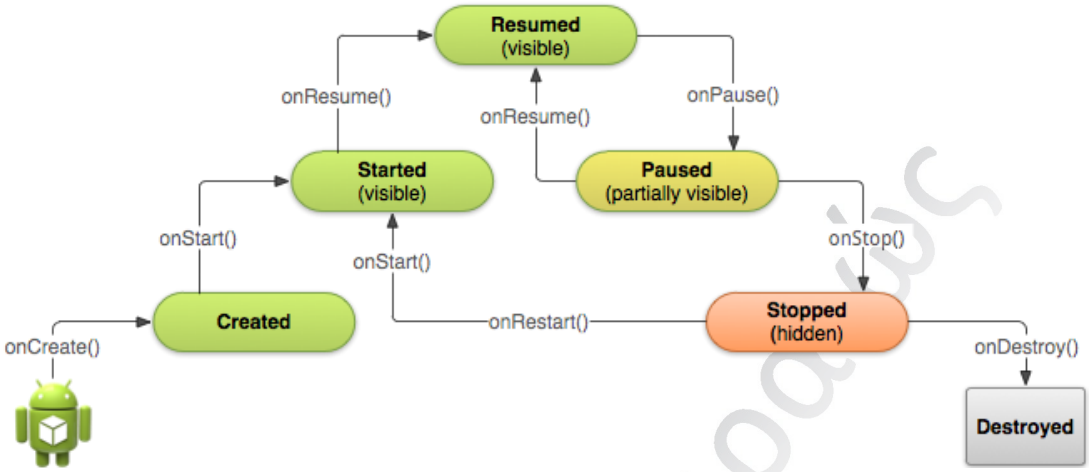
Η κατασκευή μιας εφαρμογής Android ακολουθεί μια δομή και η οποία αποτελείται από κάποια συστατικά στοιχεία (application components) που ως προς τη γενική τους μορφή είναι όμοια για όλες τις εφαρμογές και χρησιμοποιούνται για συγκεκριμένες διαδικασίες. Κάθε εφαρμογή Android από την έναρξη της λειτουργίας της, μέχρι και τον τερματισμό της ακολουθεί ένα κύκλο ζωής. Η εφαρμογή στα ενδιάμεσα στάδια του κύκλου της ζωής της μπορεί να είναι αδρανής ή ενεργή και να ανταποκρίνεται στις επιθυμίες του χρήστη ακολουθώντας τα μηνύματα των προθέσεων Intent.

Τα βασικότερα συστατικά στοιχεία της εφαρμογής είναι οι δραστηριότητες (activities), οι προθέσεις (intents), οι υπηρεσίες (services), οι παραλήπτες μηνυμάτων (broadcast receivers) και οι πάροχοι περιεχομένου (content providers). [31]

4.1 Η δραστηριότητα (activity) και ο κύκλος της ζωής της

Η δραστηριότητα (activity) θεωρείται το βασικότερο συστατικό κάθε Android εφαρμογής. Τα περισσότερα activities παρουσιάζουν μία όψη "view" στο χρήστη, είτε για την προβολή κάποιων γραφικών, είτε για την ανάκτηση των δεδομένων. Η activity μπορεί να δημιουργήσει στιγμιότυπα ενός ή περισσότερων views. Κάθε view έχει κάποια γραφικά στοιχεία (widgets) που καταλαμβάνουν είτε ολόκληρη την οθόνη είτε μέρος αυτής. Κάθε δραστηριότητα έχει το δική της οθόνη αλληλεπίδρασης με το χρήστη, την διεπαφή του χρήστη (user interface). Δηλαδή για κάθε δραστηριότητα υπάρχει τουλάχιστον ένα αρχείο γραφικού περιβάλλοντος .xml μέσα στον φάκελο των πόρων (\res\layout) του κώδικα της εφαρμογής. Κατά την εκκίνηση της εφαρμογής η πρώτη οθόνη που βλέπει ο χρήστης αντιστοιχεί στην αρχική (main) activity. Τη διαχείριση της εναλλαγής των οθονών και κατά συνέπεια των δραστηριοτήτων αναλαμβάνει μια στοίβα δραστηριοτήτων τύπο LIFO (Last In First Out) για τη μεταφορά από τη μία οθόνη της εφαρμογής στην άλλη. Η δραστηριότητα που είναι ορατή στην οθόνη βρίσκεται στην κορυφή της στοίβας και όταν ο χρήστης επιλέξει να επιστρέψει σε προηγούμενη κατάσταση πατώντας το πλήκτρο επιστροφής τότε η εφαρμογή επιστρέφει στην προηγούμενη ενέργεια, τότε η δραστηριότητα

βγαίνει από την πρώτη θέση της λίστας στην οποία βρισκόταν και την πρώτη θέση καταλαμβάνει η οθόνη της προηγούμενης δραστηριότητας.



Εικόνα 4.1 Οι καταστάσεις στις οποίες μπορεί να περιέλθει μία εφαρμογή android [35]

Η παραπάνω εικόνα απεικονίζει αυτούς τους βρόγχους και τα μονοπάτια που ακολουθεί μία δραστηριότητα μεταξύ των καταστάσεων στις οποίες μπορεί να βρεθεί.

Μία δραστηριότητα μπορεί να βρεθεί στις ακόλουθες καταστάσεις:

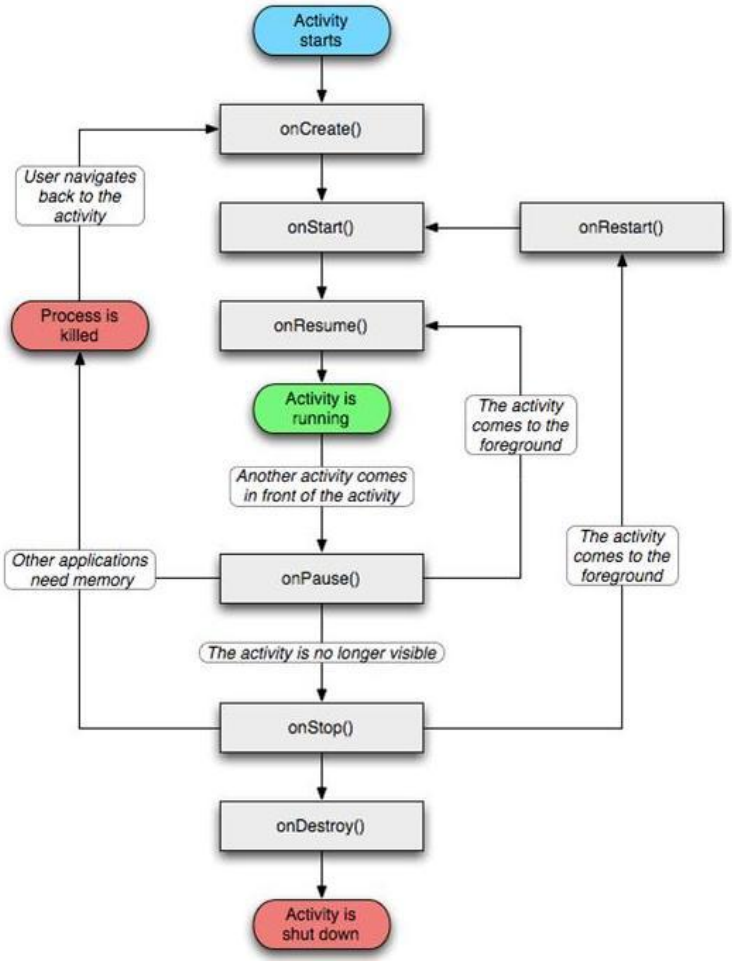
Να είναι **ενεργή** (active ή running) και να είναι ορατά τα αποτελέσματα της εκτέλεσης της στον χρήστη. Σε αυτή την κατάσταση, η δραστηριότητα είναι σε πρώτο πλάνο και ο χρήστης μπορεί να αλληλεπιδράσει με αυτή.

Να τεθεί **σε παύση** (paused) σε αυτή την κατάσταση, η δραστηριότητα είναι μερικώς ορατή επισκιάζεται από μια άλλη δραστηριότητα, η άλλη δραστηριότητα που είναι σε πρώτο πλάνο είναι ημιδιαφανής ή δεν καλύπτει ολόκληρη την οθόνη. Η δραστηριότητα που είναι σε παύση δεν λαμβάνει είσοδο από το χρήστη και δεν μπορεί να εκτελέσει οποιοδήποτε κώδικα.

Να είναι **σταματημένη** (stopped) σε αυτή την κατάσταση, η δραστηριότητα είναι εντελώς κρυμμένη και δεν είναι ορατή στο χρήστη, θεωρείται ότι εκτελείται στο παρασκήνιο (background). Καθώς η δραστηριότητα σταματά, αν και όλες οι πληροφορίες αυτής της κατάστασης, όπως οι μεταβλητές διατηρούνται, εντούτοις δεν μπορεί να εκτελέσει οποιοδήποτε κώδικα.

Εικόνα 4.2 Μέθοδοι κλήσης μιας δραστηριότητα [31]

Η δραστηριότητα μεταβαίνει από κατάσταση σε κατάσταση (Εικόνα 4.3) με την κλήση των συναρτήσεων onCreate(), onStart(), onRestart(), onResume(), onPause(), onStop() και onDestroy(). Αυτές οι μέθοδοι ορίζουν ολόκληρο τον κύκλο ζωής της δραστηριότητας.



Εικόνα 4.3 Ο κύκλος ζωής μιας δραστηριότητας (activity) [31]

Η μέθοδος onCreate() εκτελείται για μία και μόνο φορά κατά τη ν εκκίνηση της δραστηριότητας. Μέσα στη μέθοδο αυτήν πραγματοποιείται με την κλήση της setContentView() η σύνδεση της δραστηριότητας με το γραφικό περιβάλλον της διεπαφής της με τον χρήστη.

Με την κλήση της onResume() γίνεται η αρχικοποίηση και η ανάκτηση των δεδομένων της δραστηριότητας. Στη συνάρτηση αυτή τοποθετούνται εντολές που θα επαναληφθούν όπως είναι η εντολή για την επανεκκίνηση των αρχείων βίντεο, ήχου και εικόνας, η εντολή για την επανεμφάνιση μηνυμάτων κλπ.

Η onPause() αναλαμβάνει τη διακοπή της δραστηριότητας και την αποθήκευση των δεδομένων.

Η onDestroy() καλείται στο τέλος της ζωής της δραστηριότητας για τον τερματισμό της και την απελευθέρωση των πόρων που είχε δεσμεύσει κατά τη διάρκεια της λειτουργίας της.

Με τις μεθόδους startActivity() και finish() ο χρήστης μπορεί να επιστρέψει σε στιγμιότυπα activity που εμφανίζονται και έπειτα καταργούνται όταν εκτελείται η δραστηριότητα της οθόνης χωρίς να επανεκκινήσει την εφαρμογή. Τέτοιο παράδειγμα activity είναι η αρχική οθόνη της εκκίνησης της εφαρμογής. [31], [35]

4.2 Οι προθέσεις (intents)

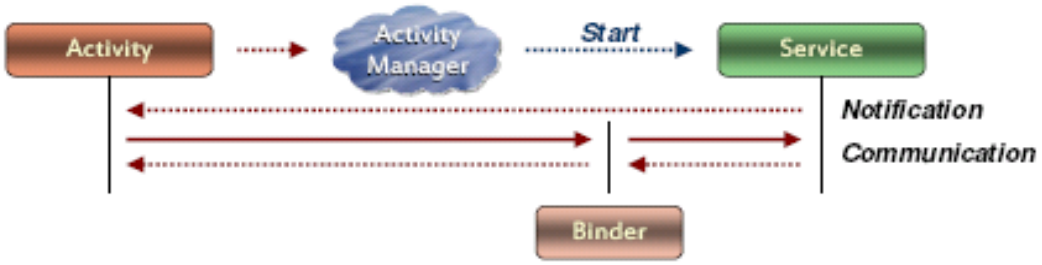
Η πρόθεση (intent) είναι το στοιχείο που περιέχει την πρόθεση του χρήστη για την εκτέλεση μίας ενέργειας. Μεταφέρει ένα απλό μήνυμα που αναφέρεται στην πρόθεση να γίνει μια ενέργεια.

Εικόνα 4.4 Οι προθέσεις (intents) [31]

Οι προθέσεις χρησιμοποιούνται για να ενεργοποιήσουν κάποια δραστηριότητα και είναι προφανές ότι μία πρόθεση προπορεύεται μιας δραστηριότητας. [31]

4.3 Οι υπηρεσίες (services)

Οι υπηρεσίες (services) δεν είναι ορατές στον χρήστη, είναι απρόσωπα στοιχεία που εκτελούνται στο background (music player, network download κλπ), αλλά είναι απαραίτητες για την λειτουργία της εφαρμογής πχ ενεργοποίηση του Bluetooth, η σύνδεση με το wi-fi κλπ.



Εικόνα 4.5 Οι υπηρεσίες (services) [31]

Μία υπηρεσία (service) υλοποιείται με παρόμοιο τρόπο με αυτόν της δραστηριότητας (activity). Κάθε υπηρεσία για να μπορεί να χρησιμοποιηθεί από την εφαρμογή πρέπει να δηλωθεί στο αρχείο Manifest της εφαρμογής (βλέπε παράγραφο 7.4.1 Ο ρόλος του αρχείου AndroidManifest.XML). Οι υπηρεσίες επιτελούν τις ακόλουθες βασικές λειτουργίες. Η λειτουργία της έναρξης της εκάστοτε υπηρεσίας με την κλήση της μεθόδου startService() από κάποια δραστηριότητα για να ξεκινήσει η υπηρεσία και να συνεχίσει την λειτουργία της ακόμα και αν η δραστηριότητα που την κάλεσε έχει τερματιστεί. Για το λόγο αυτό η υπηρεσία δεν επιστρέφει κάποιο αποτέλεσμα στην δραστηριότητα από την οποία είχε κληθεί. Η λειτουργία της σύνδεσης της υπηρεσίας, με την κλήση της bindServise(), με κάποιο άλλο στοιχείο της εφαρμογής με το οποίο η υπηρεσία αλληλεπιδρά (πχ όπως είναι η ανταλλαγή των μηνυμάτων). Όταν το στοιχείο της αλληλεπίδρασης τερματιστεί τότε παύει να λειτουργεί και η υπηρεσία. Ο τερματισμός λειτουργίας της υπηρεσίας γίνεται με την κλήση της stopSelf(). [31]

4.4 Οι παραλήπτες μηνυμάτων (broadcast receiver)

Οι παραλήπτες μηνυμάτων (broadcast receivers) είναι συστατικά στοιχεία που περιμένουν να ενεργοποιηθούν από ένα συμβάν. Ενημερώνονται για κάποιο συμβάν με τη λήψη διάφορων ειδοποιήσεων από το Λ.Σ. και αντιδρούν με την εκπομπή των ανάλογων μηνυμάτων. Οι broadcast receivers δεν παρέχουν user interface, αλλά ενημερώνουν μέσω των status bar notifications, το χρήστη για κάποια ενέργεια όπως είναι η χαμηλή στάθμη της μπαταρίας ή την αλλαγή της γλώσσας ή η χρήση του flash κατά την λήψη των φωτογραφιών κλπ. Οι εφαρμογές μπορούν με τη χρήση των παραληπτών μηνυμάτων να στείλουν μία ειδοποίηση σε άλλες εφαρμογές για να ενημερώσουν ότι π.χ. κάποια δεδομένα έχουν κατέβει στη συσκευή και είναι έτοιμα προς χρήση.

Οι παραλήπτες μηνυμάτων ενεργοποιούνται από τα broadcast intents και χρησιμοποιούν φίλτρα (intent filters) για να διαχωρίσουν το intent που τους ενδιαφέρει. Για να μπορούν να χρησιμοποιηθούν οι παραλήπτες μηνυμάτων από την εφαρμογή πρέπει δηλώνονται στο αρχείο .java ή στο αρχείο manifest xml. [31]

4.5 Οι πάροχοι περιεχομένου (content providers)

Οι πάροχοι περιεχομένου (content providers) χρησιμοποιούνται για την αποθήκευση των δεδομένα και τη διάθεση τους στις άλλες εφαρμογές. [31]

4.6 Σύνοψη

Στο κεφάλαιο αυτό παρουσιάστηκαν τα δομικά στοιχεία από τα οποία αποτελείται μια εφαρμογής android, πού χρησιμεύουν τα στοιχεία αυτά και ποιες είναι οι δυνατότητες τους. Παρουσιάστηκε η αρχιτεκτονική και ο τρόπος λειτουργίας κάθε εφαρμογής android.

5 Απαιτήσεις μηχανικού εξοπλισμού και λογισμικού

Στο κεφάλαιο αυτό αναφέρονται οι απαιτήσεις του μηχανικού εξοπλισμού και το λογισμικό που χρησιμοποιήθηκαν για την ανάπτυξη του κώδικα του project του BluetoothChat. Αναφέρονται οι ιστοσελίδες στις οποίες βρίσκεται αναρτημένο το λογισμικού. Παρέχονται σημαντικές πληροφορίες της ρύθμισης των συσκευών κατά την διαδικασία της ανάπτυξης της εφαρμογής.

5.1 Οι απαιτήσεις της εφαρμογής ως προς τον μηχανικό (hardware) εξοπλισμό

Ο μηχανικός εξοπλισμός (hardware) που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής είναι ο ακόλουθος:

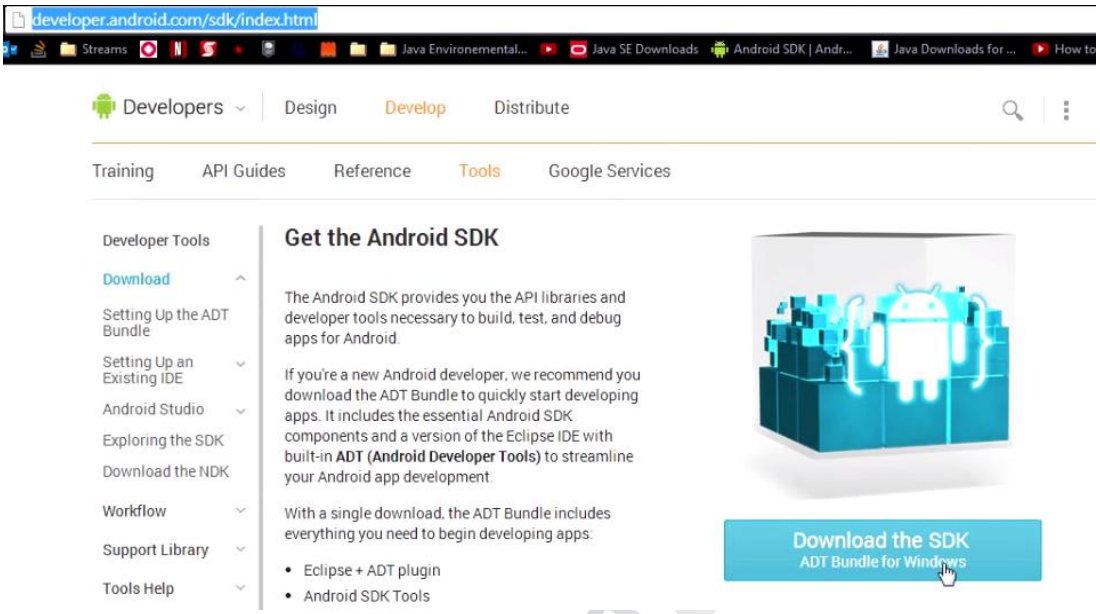
- ένας Η/Υ (Dell, Inspiron N5110, RAM 4GB στα 1333 MHz και HDD 500 GB) με Λ.Σ. Windows 7 (64-bit). Στον Η/Υ εγκαταστάθηκε το λογισμικό της επόμενης παραγράφου το οποίο παρέχεται δωρεάν.
- μια συσκευή κινητής τηλεφωνίας με λειτουργικό σύστημα Android έκδοσης 2.0.1 (API 6) ή μεταγενέστερη, με επεξεργαστή (CPU) 600 MHz, εσωτερική μνήμη 150 MB και εσωτερική μνήμη 150 MB. Ο χώρος που καταλαμβανει η εφαρμογή κατά την εγκατάσταση της είναι 867 Kb. Η συσκευή θα πρέπει να υποστηρίζει την υπηρεσία του BT.

5.2 Οι απαιτήσεις της εφαρμογής ως προς το λογισμικό (software)

Οι βασικές απαιτήσεις όσον αφορά το λογισμικό για την κατασκευή εφαρμογών android είναι η εγκατάσταση του εκτελέσιμου αρχείου Java, η εγκατάσταση του Android SDK και η εγκατάσταση του Eclipse IDE και του Eclipse plug in (βοηθητικά).

Για την ανάπτυξη αυτής της εφαρμογής έγινε εγκατάσταση του εργαλείου ανάπτυξης εφαρμογών (Integrated Development Environment, IDE) Eclipse για να μπορούμε να μεταγλωττίσουμε και να εκτελέσουμε τον κώδικα της εφαρμογής. Επιπλέον εγκαταστάθηκε το εργαλείο ανάπτυξης λογισμικού Android Development

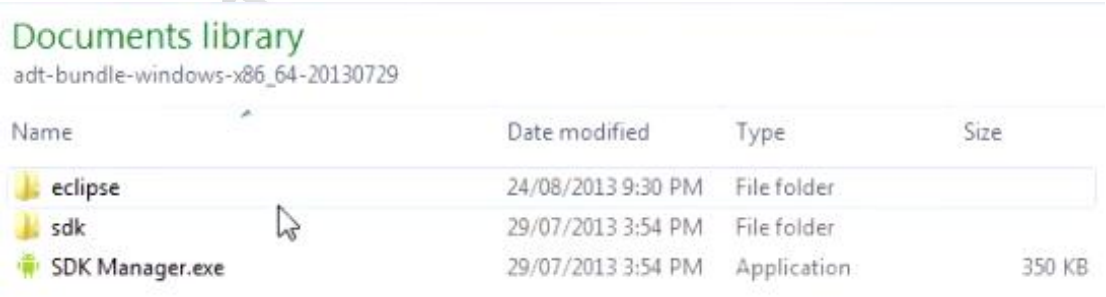
Tools (ADT) που διατίθεται δωρεάν από την ιστοσελίδα των Android Developers στην οποία παρέχονται και οδηγίες για την εγκατάσταση του εργαλείου αυτού.



Εικόνα 5.1 Η ιστοσελίδα του Android Developer

Πριν ξεκινήσει η μεταφόρτωση του πακέτου εμφανίζεται μία οθόνη για αποδοχή των όρων της εταιρείας και πρέπει να επιλέγουν τα 64 bit ή τα 32 bit ανάλογα με την αρχιτεκτονική του Η/Υ. Είναι σημαντικό να επιλέγουν τα σωστά bit λειτουργίας του υπολογιστή γιατί αλλιώς θα υπάρξει δυσλειτουργία στην εγκατάσταση του eclipse και του jdk.

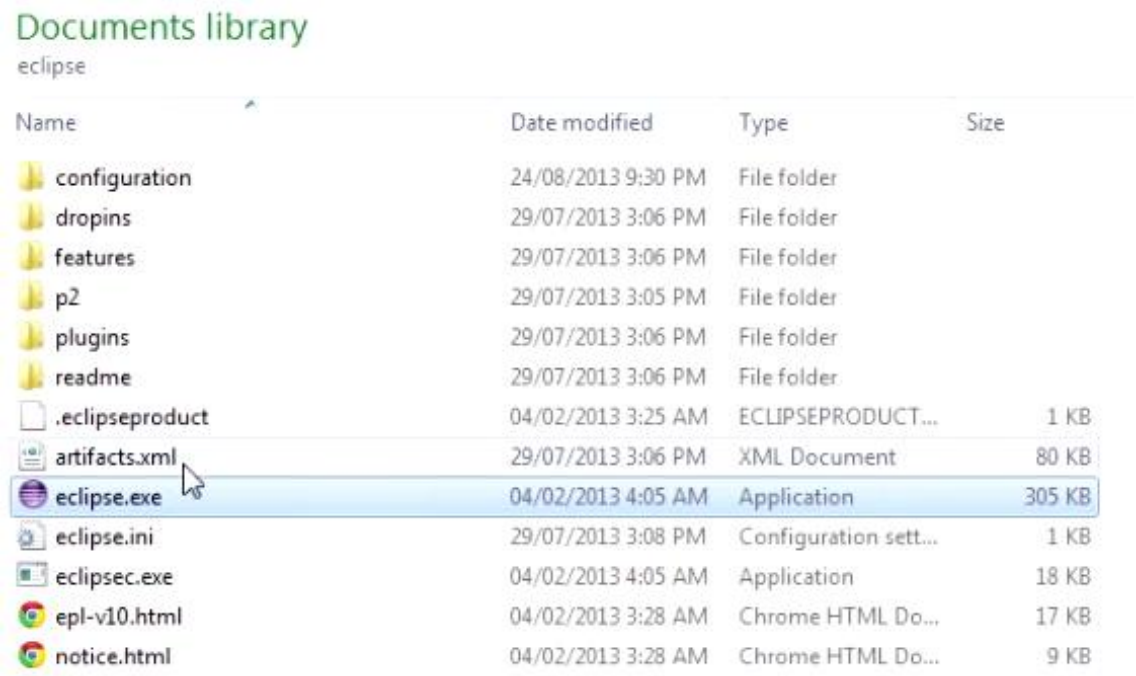
Αφού αποσυμπιέσουμε το αρχείο που κατέβηκε από τη σελίδα των Android Developers παρατηρούμε ότι περιέχονται οι κατάλογοι του eclipse και του sdk και το εκτελέσιμο αρχείο SDK Manager.



Εικόνα 5.2 ο κατάλογος με το eclipse, το sdk και το αρχείο SDK Manager

Μέσα στον φάκελο του eclipse βρίσκεται το εκτελέσιμο αρχείο του πακέτου eclipse.exe. Με διπλό κλικ πάνω στο αρχείο eclipse.exe γίνεται η εκκίνηση της εφαρμογής και επειδή είναι η πρώτη φορά της εκκίνησης της ένα παράθυρο διαλόγου

ζητά να οριστεί η θέση του καταλόγου workspace στον οποίο θα αποθηκεύονται τα πακέτα των εφαρμογών που πρόκειται να δημιουργηθούν με το eclipse.



Εικόνα 5.3 Ο κατάλογος του eclipse

Το πιο σημαντικό κομμάτι της λειτουργίας του Eclipse IDE είναι η εγκατάσταση του εργαλείου ανάπτυξης λογισμικού ADT που συμπεριλαμβάνεται στο πακέτο εγκατάστασης. Ταυτόχρονα με την διαδικασία της εγκατάστασης και της ενημέρωση του ADT με την νεότερη έκδοση του γίνεται και η εγκατάσταση του SDK.

Το εργαλείο ανάπτυξης λογισμικού Java Development Kit επιτρέπει τον προγραμματισμό σε γλώσσα προγραμματισμού Java μέσα από το περιβάλλον του Eclipse. Από την ιστοσελίδα της oracle: http://www.oracle.com/technetwork/java/javase/downloads/index.html είναι διαθέσιμο δωρεάν το εργαλείο του jdk.

Για την εγκατάσταση του εργαλείου jdk επιλέγουμε το εικονίδιο «Java DOWNLOAD» για να μεταφερθούμε στην επόμενη σελίδα. Η αποδοχή των όρων μας επιτρέπει να επιλέξουμε το Java SE Development Kit συμφώνα με το Λ.Σ. του Η/Υ και τα bits λειτουργίας του.

www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Java SE Development Kit 8u31

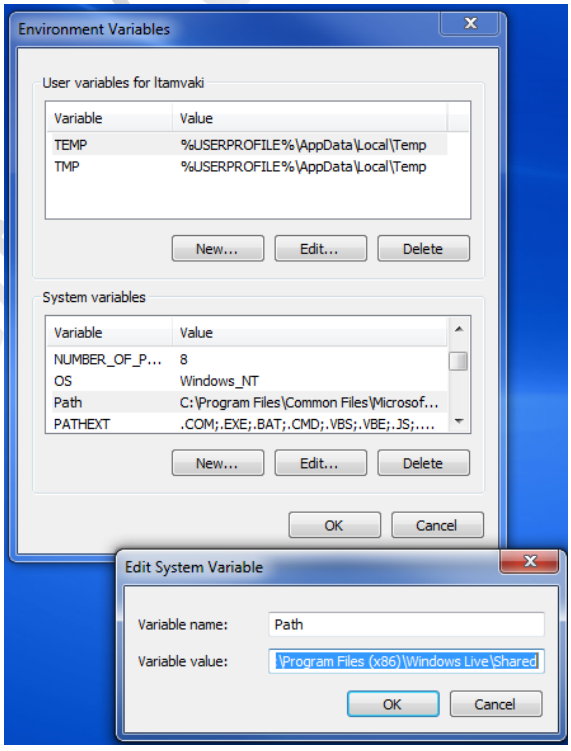
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux x86	135.24 MB	jdk-8u31-linux-i586.rpm
Linux x86	154.91 MB	jdk-8u31-linux-i586.tar.gz
Linux x64	135.62 MB	jdk-8u31-linux-x64.rpm
Linux x64	153.45 MB	jdk-8u31-linux-x64.tar.gz
Mac OS X x64	209.17 MB	jdk-8u31-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	136.91 MB	jdk-8u31-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	97.11 MB	jdk-8u31-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	137.51 MB	jdk-8u31-solaris-x64.tar.Z
Solaris x64	94.82 MB	jdk-8u31-solaris-x64.tar.gz
Windows x86	157.96 MB	jdk-8u31-windows-i586.exe
Windows x64	170.36 MB	jdk-8u31-windows-x64.exe

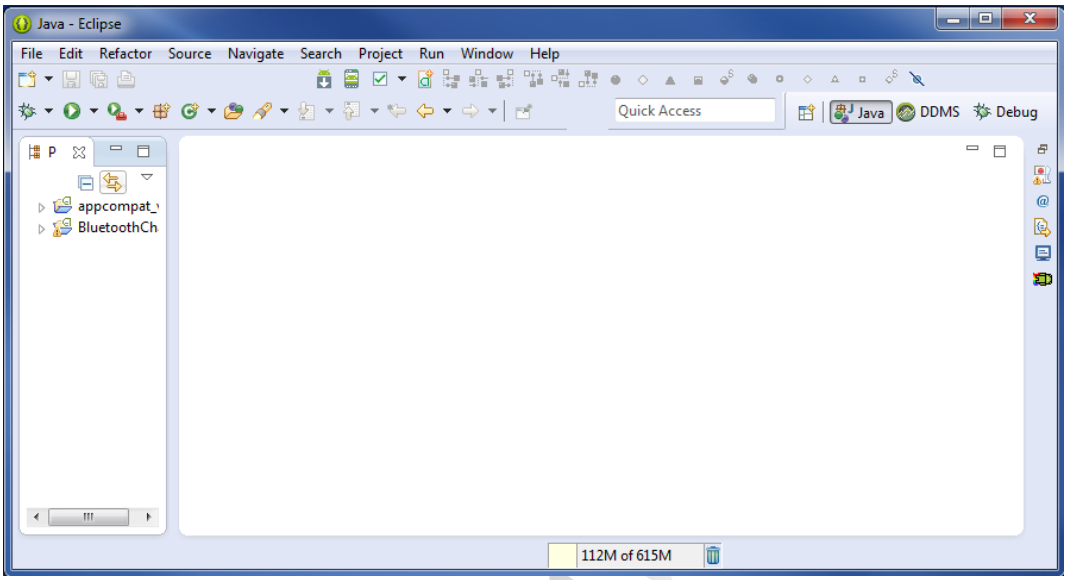
Εικόνα 5.4 Η ιστοσελίδα της Oracle

Η εγκατάσταση του jdk γίνεται με διπλό klik πάνω στο αρχείο του jdk. Πρέπει να δηλώσουμε τη διαδρομή (path) του jdk στις μεταβλητές περιβάλλοντος (environment variables) του Η/Υ. Για να γίνει αυτό απλά εντοπίζουμε τη διαδρομή του αρχείου javaw.exe και την προσθέτουμε στις environment variables στο πεδίο path.



Εικόνα 5.5 Environment variables και το πεδίο path

Η κατασκευή των εφαρμογών Android επιτυγχάνεται με τη «συνεργασία» του κώδικα java ο οποίος χρησιμοποιείται για την υλοποίηση των ενεργειών και των xml αρχείων που περιέχουν το σχεδιασμό των διεπαφών της εφαρμογής.



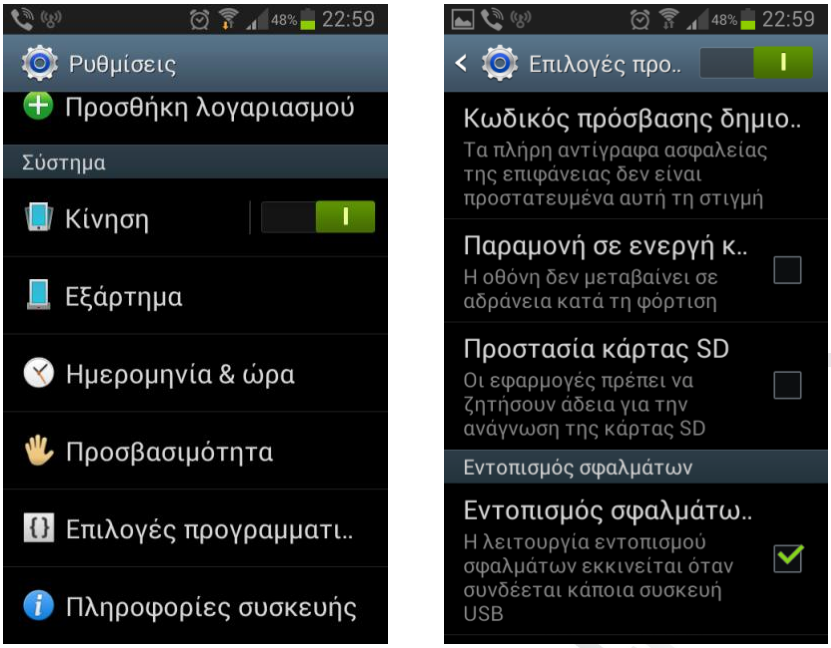
Εικόνα 5.6 Το περιβάλλον ανάπτυξης λογισμικού Eclipse

Τα εργαλεία του SDK μεταγλωττίζουν τον κώδικα μαζί με τα δεδομένα και τα αρχεία των πόρων σε ένα πακέτο το Android package το οποίο είναι ένα συμπιεσμένο αρχείο με την κατάληξη .apk. Ο κώδικας σε ένα αρχείο .apk θεωρείται ως μια εφαρμογή και είναι το αρχείο που χρησιμοποιεί η συσκευή για να εγκαταστήσει την εφαρμογή. [36], [37]

5.3 Ρύθμιση της συσκευής κινητής τηλεφωνίας

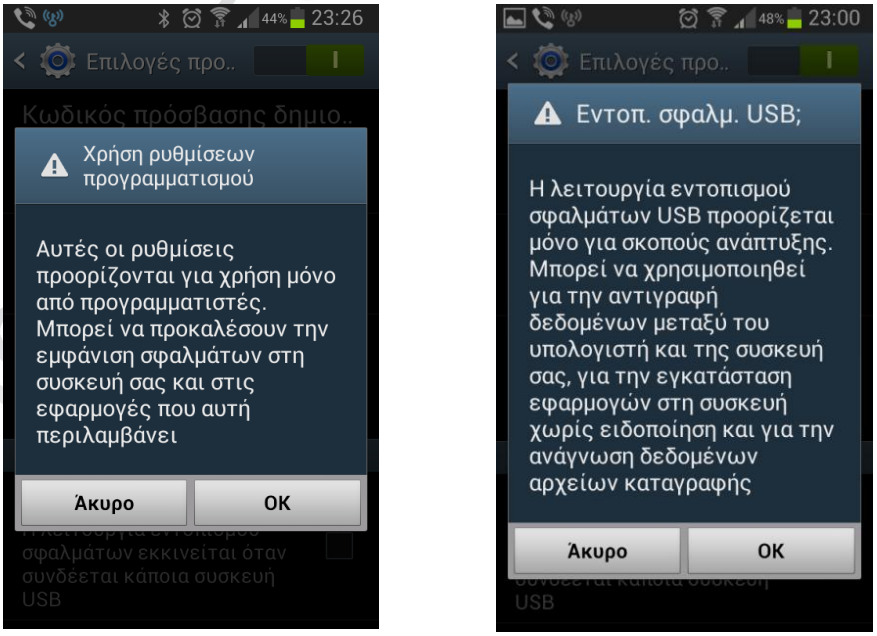
Οι μηχανές προσομοίωσης του AVD δεν υποστηρίζουν την τεχνολογία του BT για το λόγο αυτό χρησιμοποιήθηκε μια συσκευή κινητής τηλεφωνίας αφού πρώτα έγιναν οι κατάλληλες ρυθμίσεις.

Στη συσκευή κινητής τηλεφωνίας (λειτουργικό σύστημα Android) χρειάζεται να γίνουν κάποιες ρυθμίσεις πριν χρησιμοποιηθεί ως εργαλείο για την ανάπτυξη του λογισμικού Android. Στις "Ρυθμίσεις" στο κάτω μέρος εμφανίζονται οι "Επιλογές προγραμματισμού". Στην οθόνη των επιλογών προγραμματισμού στο πάνω μέρος σύρουμε το διακόπτη ώστε να ενεργοποιηθούν οι επιλογές προγραμματισμού (πράσινος διακόπτης)



Εικόνα 5.7 Α) "Ρυθμίσεις" και Β) "Επιλογές προγραμματισμού"

Αμέσως εμφανίζεται το παράθυρο διαλόγου που μας ενημερώνει για τη "Χρήση ρυθμίσεων προγραμματισμού". Μετά την αποδοχή του παράθυρου διαλόγου εμφανίζεται η οθόνη των επίλογων προγραμματισμού τσεκάρουμε την τελευταία επιλογή "Εντοπισμός σφαλμάτων". Ένα νέο παράθυρο διαλόγου "Εντοπισμός σφαλμάτων USB" εμφανίζεται, επιλέγουμε "OK". Η συσκευή κινητής τηλεφωνίας έχει ρυθμιστεί έτσι ώστε να παίξει το ρόλο του προσομοιωτή της εφαρμογής κατά τη διαδικασία της εκσφαλμάτωσης από το Eclipse.



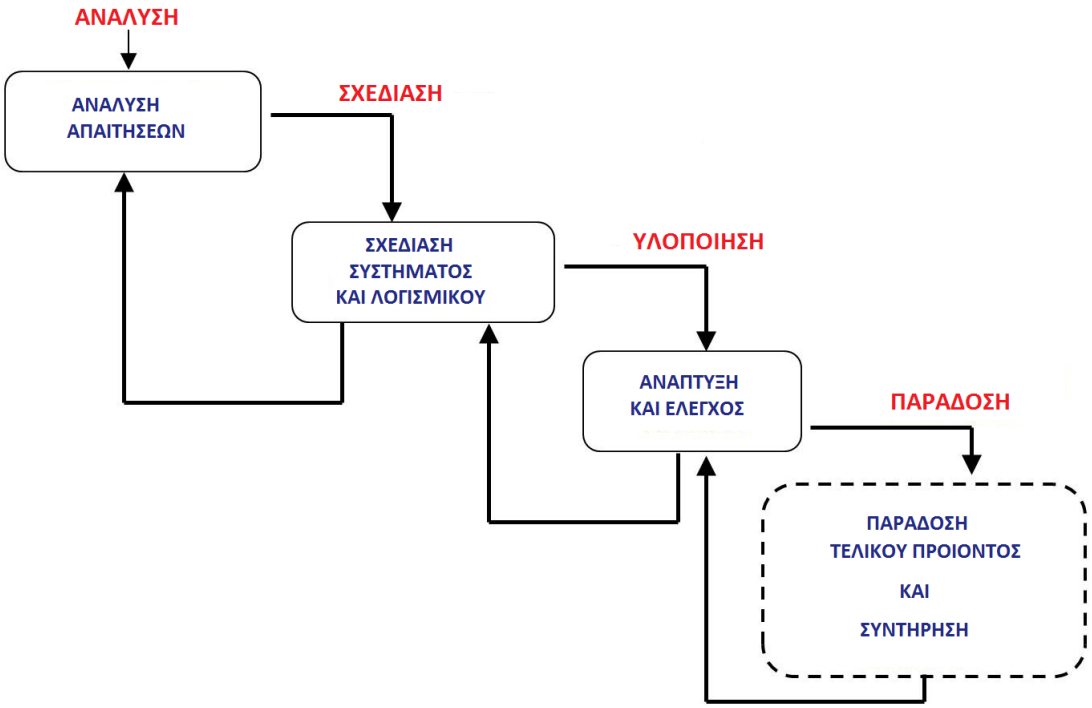
Εικόνα 5.8 Α) "Χρήση ρυθμίσεων προγραμματισμού" και Β) "Εντοπισμός σφαλμάτων USB"

5.4 Σύνοψη

Στο κεφάλαιο αυτό παρουσιάστηκαν οι απαιτήσεις σε hardware και software για την ανάπτυξη της εφαρμογής. Οι ιστοσελίδες της Oracle και των Android Developers παρέχουν δωρεάν τα προγραμματιστικά εργαλεία και αναλυτικές οδηγίες για την εγκατάσταση και τη χρήση τους. Συμφώνα με όλα τα προηγούμενα μπορούμε να συμπεράνουμε ότι η κατασκευή μιας εφαρμογής android δεν είναι ένα δαπανηρό εγχείρημα.

6 Ανάλυση και σχεδίαση της εφαρμογής πριν την υλοποίηση

Σε αυτό το κεφάλαιο παρουσιάζονται οι φάσεις της ανάλυσης και της σχεδίασης της εφαρμογής του Bluetooth chat. Αρχικά γίνεται η ανάλυση των απαιτήσεων, στη συνέχεια λαμβάνοντας υπ' όψη τα ευρήματα της ανάλυσης προχωρούσε στην σχεδίαση της εφαρμογής, τέλος εφόσον ολοκληρωθούν οι δύο προηγούμενες φάσεις ακολουθεί η φάση της υλοποίησης στην οποία γίνεται η κωδικοποίηση της εφαρμογής. Η ανάπτυξη της εφαρμογής ολοκληρώνεται με τον έλεγχο και την τεκμηρίωση του τελικού συστήματος πριν τη παράδοση του. Το γραμμικό μοντέλο (linear life cycle model) που χρησιμοποιήθηκε για την ανάπτυξη του συστήματος είναι το «μοντέλο του καταρράκτη» (waterfall model) που απεικονίζεται στην επόμενη εικόνα.



Εικόνα 6.1 Το «μοντέλο του καταρράκτη» (waterfall model)

Κάθε λογισμικό πριν φτάσει στο στάδιο της παραγωγής του κώδικα ακολουθεί μια διαδικασία ανάλυσης του τρόπου ανάπτυξης του, τόσο μέσα από κείμενα αλλά και διαγράμματα και ότι άλλο θεωρηθεί απαραίτητο για την κατανόηση του συστήματος. Η ενοποιημένη γλώσσα σχεδιασμού (*unified modeling language*, UML) είναι μια γραφική γλώσσα για την οπτική παράσταση, τη διαμόρφωση των προδιαγραφών και την τεκμηρίωση των συστημάτων. [38]

6.1 Η UML μοντελοποίηση

Η UML (Unified Modeling Language) είναι μια γλώσσα που περιγράφει την κατάρτιση των προδιαγραφών του λογισμικού και τη δημιουργία και την τεκμηρίωση των τμημάτων του λογισμικού. Αναπαριστά με οπτικό τρόπο (visualization) τα τμήματα του λογισμικού και χρησιμοποιείται για την μοντελοποίηση εταιρικών και άλλων συστημάτων που δεν αφορούν το λογισμικό. Η UML περιγράφει τη δυναμική δομή και τη στατική δομή του συστήματος. Η δυναμική δομή του συστήματος περιλαμβάνει διαγράμματα όπως είναι το Διάγραμμα των Περιπτώσεων Χρήσης (Use case diagram), το Διαγράμματα των Δραστηριοτήτων (Activity diagram), το Διάγραμμα Ακολουθίας (Sequence diagram), το Διάγραμμα Συνεργασίας (Collaboration diagram), το Διάγραμμα Καταστάσεων (Statechart diagram). Η στατική δομή του συστήματος περιλαμβάνει το Διάγραμμα Κλάσεων (Class diagram), το Διάγραμμα Αντικειμένων (Object diagram), το Διάγραμμα Συστατικών (Component diagram), και το Διάγραμμα Διάταξης (Deployment diagram).

Η χρήσης της γλώσσας UML και των διαγραμμάτων της στοχεύει στην μοντελοποίηση των συστημάτων συμφώνα με τις αρχές των αντικειμενοστραφών μοντέλων, το συνταίριασμα των σκέψεων και της πρακτικής εφαρμογής τους ενώ αποτελεί μια μοντελοποιημένη γλώσσα που μπορεί να χρησιμοποιηθεί τόσο από τον άνθρωπο όσο κι από τις μηχανές. [39], [38]

6.2 Ανάλυση

Η φάση της ανάλυσης είναι πολύ κρίσιμη γιατί στη φάση αυτή αναλύονται οι απαιτήσεις του συστήματος οι οποίες θα χρησιμοποιηθούν στην επομένη φάση της σχεδίασης. Οι απαιτήσεις του συστήματος καταγράφονται είτε σε κείμενο, είτε σε λίστα είτε σε πίνακα προκειμένου να οριοθετηθούν οι εργασίες που θα εκτελεί το σύστημα. Στη φάση της Ανάλυσης κατασκευάζονται τα διαγράμματα των Περιπτώσεων Χρήσης (use-case), των Κλάσεων (class) και των Δραστηριοτήτων (activity). Η φάση της ανάλυσης και ειδικά το στάδιο της προδιαγραφής είναι η πρώτη φάση ανάπτυξης του συστήματος.

6.2.1 Η προδιαγραφή των απαιτήσεων του έργου Bluetooth Chat

Στην προδιαγραφή των απαιτήσεων ορίζεται «ποιό είναι το πρόβλημα;». Τι καλείται να υλοποιήσει η εφαρμογή αυτή;

Στο σημείο αυτό καθορίζονται οι στόχοι, διαμορφώνεται το πλαίσιο εργασίας, καταγράφονται και αναλύονται οι απαιτήσεις που υπάρχουν. Τι δυνατότητες θα παρέχει η εφαρμογή; Πρέπει να μπορεί ο χρήστης να επικοινωνεί με τους φίλους του δωρεάν χωρίς να χρειάζεται να έχει πρόσβαση σε δίκτυο wifi ή στο Internet. Να επιτυγχάνεται η ανταλλαγή μηνυμάτων σε συνέδρια, στο χώρο εργασίας, στην τάξη, σε festivals και clubs.

Ζητείται να υλοποιηθεί, ένας ασφαλής, γρήγορος, εύκολος και δωρεάν τρόπος επικοινωνίας. Το κείμενο των προδιαγραφών της εφαρμογής που ακολουθεί είναι το προϊόν αυτής της φάσης.

Πίνακας 6.1 Οι προδιαγραφές της εφαρμογής

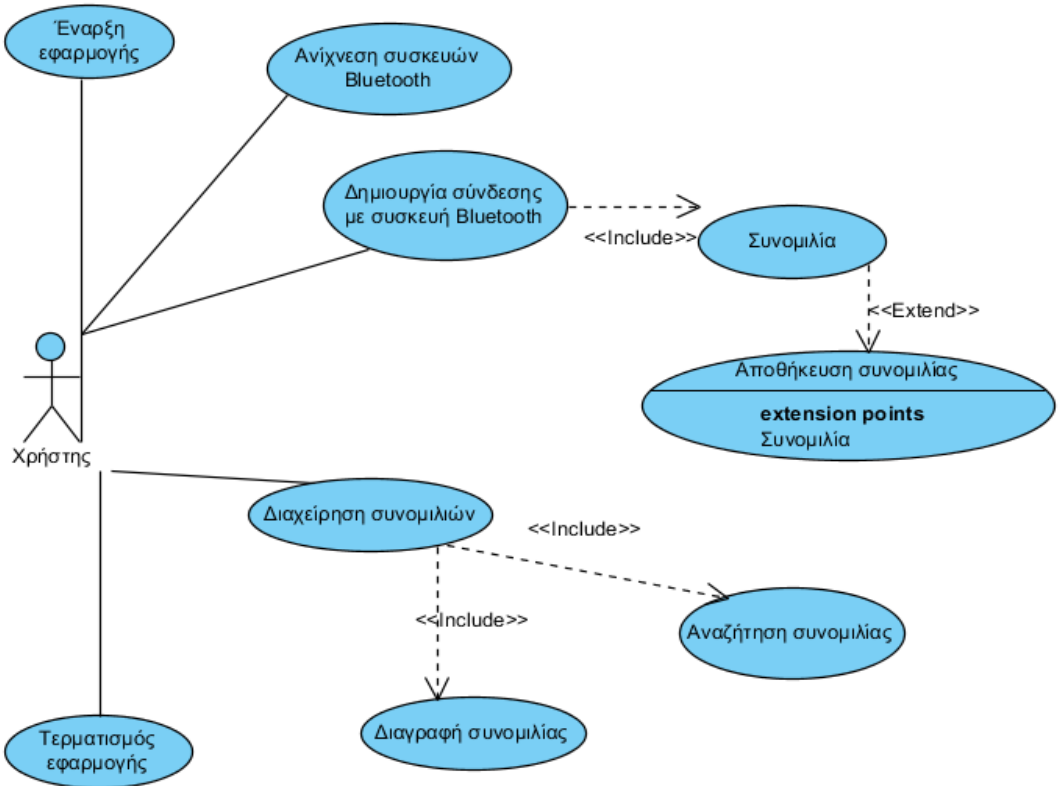
Οι προδιαγραφές της εφαρμογής
Η εφαρμογή πρέπει να υλοποιεί τις ακόλουθες εργασίες: 1. Ανίχνευση των συσκευών που έχουν ενεργοποιημένη την υπηρεσία του Bluetooth εντός της απόστασης λειτουργίας της συσκευής. 2. Αντιστοίχιση (paired) των συσκευών που έχουν ενεργοποιημένη την υπηρεσία του bluetooth μέσω του τοπικού προσαρμογέα του bluetooth 3. Εγκαθίδρυση ενός καναλιού RFCOMM και ενεργοποίηση της υπηρεσίας του bluetooth 4. Σύνδεση με άλλες συσκευές μέσω της διαδικασίας της ανακάλυψης υπηρεσιών. 5. Ανταλλαγή μηνυμάτων μέσω της υπηρεσίας του bluetooth 6. Αποθήκευση των συνομιλιών σε Βάση Δεδομένων (Database) 7. Αναζήτηση μιας συνομιλίας από τη ΒΔ. 8. Ταξινόμηση των αποθηκευμένων συνομιλιών α) με αλφαβητική σειρά (ως προς το όνομα του συνομιλητή) και β) με χρονική σειρά ως προς την ημερομηνία καταχώρησης της συζήτησης. 9. Διαγραφή κάποιας συνομιλίας από τη ΒΔ.

Αφού καταλήξουμε στο τελικό κείμενο των προδιαγραφών ακολουθεί ο ορισμός του πλάνου εργασίας. Οι πρώτες πέντε εργασίες υλοποιούνται στο πλαίσιο της

εγκατάστασης και της λειτουργίας της προδιαγραφής του bluetooth. Ο κώδικας αυτός διατίθεται δωρεάν στην ιστοσελίδα των Android developers (https://android.googlesource.com/platform/development/25b6aed7b2e01ce7bdc0dfa1a79eaf009ad178fe/samples/BluetoothChat/src/com/example/android/BluetoothChat/BluetoothChatService.java). Ενώ οι υπόλοιπες εργασίες (6-9) υλοποιούν τη ΒΔ και τη διαχείριση των συνομιλιών που έχουν καταγραφεί. [38]

6.2.2 Το Διάγραμμα Περιπτώσεων Χρήσης της εφαρμογής

Το Διάγραμμα των Περιπτώσεων Χρήσης (Δ.Π.Χ., Use Case Diagram) είναι ένα από τα αρχικά διαγράμματα της γλώσσας μοντελοποίησης UML. Οι περιπτώσεις χρήσης περιγράφουν την δυναμική συμπεριφορά του συστήματος και απεικονίζει τον τρόπο με τον οποίο οι χειριστές χρησιμοποιούν το σύστημα. Εστιάζει στην λειτουργική άποψη και την εξωτερική όψη του συστήματος.



Διάγραμμα 6.1 Το Διάγραμμα Περιπτώσεων Χρήσης της εφαρμογής BluetoothChat

Το Δ.Π.Χ. αποτελεί το σημείο αναφοράς γιατί χρησιμοποιείται για να καθοδηγήσει την ανάπτυξη του συστήματος σε όλες τις φάσεις της ανάπτυξης του. Τυχόν λάθη που μπορεί να εντοπιστούν στην σχεδίαση του λειτουργούν ανασταλτικά

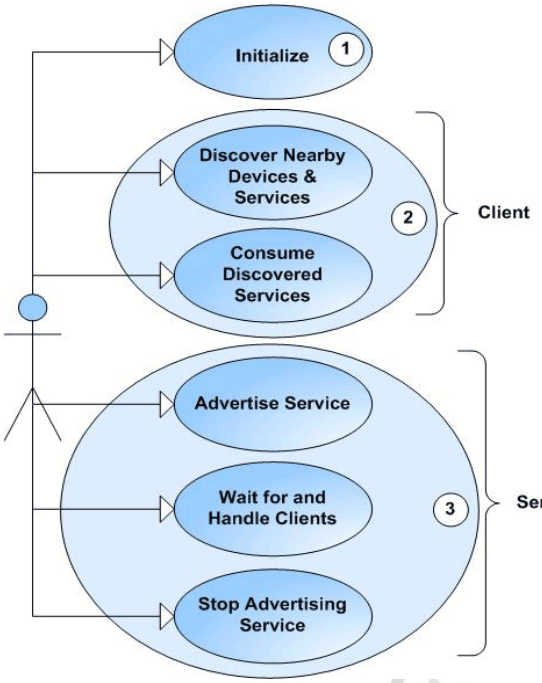
στην ανάπτυξη του συστήματος. Όσο πιο σαφές, περιεκτικό και αναλυτικό είναι το Δ.Π.Χ. τόσο πιο εύκολα και γρήγορα γίνεται η υλοποίηση του συστήματος. Το Δ.Π.Χ. περιλαμβάνει ένα σύνολο από σενάρια που στοχεύουν στην επίτευξη ενός σκοπού του χρήστη και περιέχει τις περιπτώσεις χρήσης (use cases), τους χειριστές (actors) και τις συσχετίσεις.

Το Δ.Π.Χ που παρουσιάζεται παρακάτω είναι το γενικό διάγραμμα όλων των δραστηριοτήτων των χρηστών της εφαρμογής.Το Δ.Π.Χ. που παρουσιάζεται παραπάνω η πρώτη περίπτωση χρήσης αναπαριστά την διαδικασία της ενεργοποίησης της εφαρμογής η οποία περιλαμβάνει τον κορμό των λειτουργιών που επιτελεί η εφαρμογή. Μια άλλη λειτουργία είναι η «Ανίχνευση των συσκευών Bluetooth» και η επιλογή της συσκευής με την οποία θα γίνει η συνδεση με τη δημιουργία ενός καναλιού με το πρωτόκολλο RFCOMM. Επιπλέον, στο Δ.Π.Χ., εμφανίζεται η διαδικασία της «Διαχείρισης των συνομιλιών» (αναζήτηση και διαγραφή των συνομιλιών) και του «Τερματισμού» της εφαρμογής για την έξοδο του χρήστη από την εφαρμογή. [40]

6.2.3 Το Διάγραμμα Περιπτώσεων Χρήσης της ενεργοποίησης και της διαχείρισης της σύνδεσης του Bluetooth

Το Δ.Π.Χ. των λειτουργιών της εφαρμογής μπορεί να επιμεριστεί σε περισσότερα Δ.Π.Χ. τα οποία θα περιέχουν μερικές από τις λειτουργίες της εφαρμογής. Αυτά τα Δ.Π.Χ. είναι πιο αναλυτικά και εστιάζουν στις συγκεκριμένες λειτουργίες με περισσότερες λεπτομέρειες.

Το Δ.Π.Χ. που ακολουθεί είναι μια τέτοια περίπτωση και παρουσιάζει την λειτουργία της ενεργοποίησης και της διαχείρισης της σύνδεσης του Bluetooth που παράγεται αν αναλύσουμε την αντίστοιχη λειτουργία του προηγούμενου Δ.Π.Χ..Το διάγραμμα 6.2 απεικονίζει τις διαδικασίες που διενεργούνται τόσο από την πλευρά του client όσο και από την πλευρά του server.



Διάγραμμα 6.2 Το Διάγραμμα Περιπτώσεων Χρήσης της ενεργοποίησης και διαχείρισης της σύνδεσης Bluetooth

Μια εφαρμογή που χρησιμοποιεί την τεχνολογία Bluetooth πρέπει να μπορεί να υποστηρίζει τη σύνδεση μίας συσκευής είτε αναλαμβάνοντας να συνδεθεί ως πελάτης είτε ως διακομιστής. Ή μπορεί να συμπεριφερθεί σαν ένα peer-to-peer τερματικό σημείο αναλαμβάνοντας και τους δύο ρόλους του πελάτη και του διακομιστή.

Εκκίνηση (Initialize) - Κάθε εφαρμογή με δυνατότητα Bluetooth, τόσο από την πλευρά του διακομιστή όσος και του πελάτη, πρέπει πρώτα να προετοιμάσει τη στοίβα Bluetooth.

Πελάτης (Client) - Ένας πελάτης καταναλώνει απομακρυσμένες υπηρεσίες. Στη αρχή ανιχνεύει τις εντός εμβέλειας συσκευές, τότε για κάθε συσκευή που έχει ανιχνεύσει ψάχνει να εντοπίσει τις υπηρεσίες που τον ενδιαφέρουν.

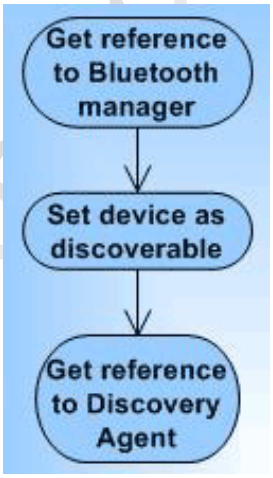
Διακομιστής (Server) - Ένας server κάνει διαθέσιμες τις υπηρεσίες στους πελάτες. Τις καταχωρεί στη βάση δεδομένων ανίχνευσης υπηρεσιών Service Discovery Database (SDDB), στην πραγματικότητα τις «διαφημίζει» ως διαθέσιμες. Στη συνέχεια περιμένει για εισερχόμενες συνδέσεις, αποδέχεται όλες αυτές που εισέρχονται, και εξυπηρετεί τους πελάτες που τις αιτούν. Τέλος, όταν η υπηρεσία δεν είναι πλέον απαραίτητη η εφαρμογή την αφαιρεί από τη βάση δεδομένων SDDB. [40] [41]

6.2.4 Το Διάγραμμα Δραστηριότητας της ενεργοποίησης και της δημιουργίας-διαχείρισης της σύνδεσης Bluetooth

Το Διάγραμμα Δραστηριότητας (Activity Diagram) μοντελοποιεί τη ροή της εργασίας, αναπαριστώντας τις διάφορες καταστάσεις εκτέλεσης ενός υπολογισμού (Bohm & Jacopini). Περιγράφει την ροή των εργασιών μέσα στο σύστημα και τα βασικά του στοιχεία είναι οι Δραστηριότητες (activities), οι Ενέργειες (actions) και οι Μεταβάσεις (transitions). Παρουσιάζει τη ροή του ελέγχου μεταξύ δραστηριοτήτων του ιδίου αντικειμένου ή πολλών αντικειμένων.

Το διάγραμμα δραστηριοτήτων (Εικόνα 6.2) χρησιμοποιείται για την περιγραφή της περίπτωσης χρήσης που εξετάσαμε προηγουμένως τόσο από την πλευρά του server όσο και από την πλευρά του client.

Αρχικά τόσο ο διακομιστής όσο και ο πελάτης εκτελούν την διαδικασία της Αρχικοποίησης (Initialize) όπως αναφέρθηκε παραπάνω και η οποία είναι κοινή και για τους δύο.

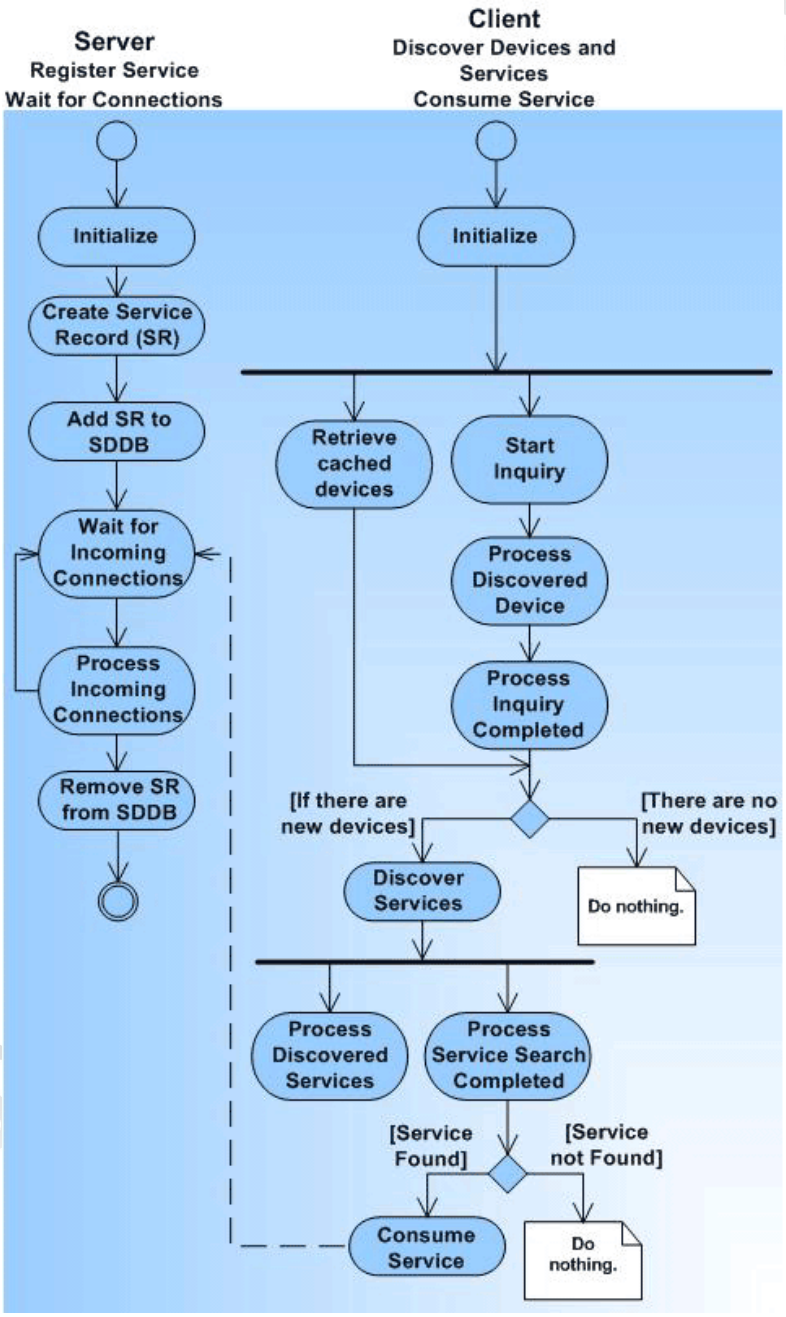


Εικόνα 6.2 Αρχικοποίηση του Bluetooth

Στη συνέχεια ανάλογα με τον ρόλο που έχει αναλάβει η συσκευή εκτελεί τις ακόλουθες δραστηριότητες.

Ο διακομιστής Δημιουργεί τις Εγγραφές των Υπηρεσιών SR (Create Service Record (SR)) που δίνατε να παρέχει στους μελλοντικούς πελάτες του, Καταχωρεί τις Εγγραφές αυτές στη βάση SDDB (Add SR to SDDB) όπως περιγράφεται παραπάνω και μετά τίθεται σε Κατάσταση Αναμονής Εισερχομένων Συνδέσεων (Wait for Incoming Connections) μέχρι να εμφανιστούν συσκευές που θα αιτηθούν τις

παρεχόμενες υπηρεσίες του. Όσο ο server εκτελεί τις παραπάνω δραστηριότητες ο client εκτελεί τη διαδικασία συγχρονισμού, προσπαθεί να Ανακτήσει τις Προσωρινά Αποθηκευμένες Συσκευές (Retrieve cashed devices), (που είχε εντοπίσει από προηγούμενες ανιχνεύσεις) ενώ ταυτόχρονα στέλνει αιτήματα ανίχνευσης των εντός εμβέλειας συσκευών μέσα από την Διαδικασία Ανίχνευσης των Συσκευών (Process Discovered Device).



Διάγραμμα 6.3 Το Διαγράμματα Δραστηριότητων του client και του server για τη χρήση Bluetooth

Αφού ολοκληρώσει τη διαδικασία Ανίχνευσης των Συσκευών (Process inquiry Completed) ο client προχώρα στην ανίχνευση των υπηρεσιών (Discover Services), εκτελεί συγχρονισμό, ολοκληρώνει την διαδικασία των ανιχνεύσιμων υπηρεσιών (Process Service Search Completed) και στέλνει αίτημα για να καταναλώσει-χρησιμοποιήσει κάποια από τις παρεχόμενες υπηρεσίες του server. Ο server (εφόσον μπορεί να διαθέσει την αιτούμενη υπηρεσία στον πελάτη) αποδέχεται το αίτημα του client βγαίνει από την κατάσταση αναμονής εισερχομένων συνδέσεων στην οποία βρισκόταν μέχρι να λάβει κάποιο αίτημα σύνδεσης και εκτελεί τη διαδικασία των εισερχόμενων συνδέσεων, ενώ μετά την ολοκλήρωση της σύνδεσης επιστρέφει στην κατάσταση αναμονής εισερχόμενων συνδέσεων μέχρι να λάβει νέο αίτημα κατανάλωσης-χρήσης υπηρεσίας από κάποιον άλλον πελάτη. Όταν ο server δεν παρέχει πλέον την υπηρεσία τότε αφαιρεί την εγγραφή SR από τη βάση SDDB (Remove SR from SDDB).

Εν ολίγοις, με την ανάληψη του ρόλου του διακομιστή η εφαρμογή προετοιμάζει μια υπηρεσία και περιμένει για τις συνδέσεις, και ο πελάτης αφού ανιχνεύσει συσκευές και υπηρεσίες, στη συνέχεια συνδέεται με κάποια συγκεκριμένη συσκευή ώστε να καταναλώνει μια συγκεκριμένη υπηρεσία. [40] [41]

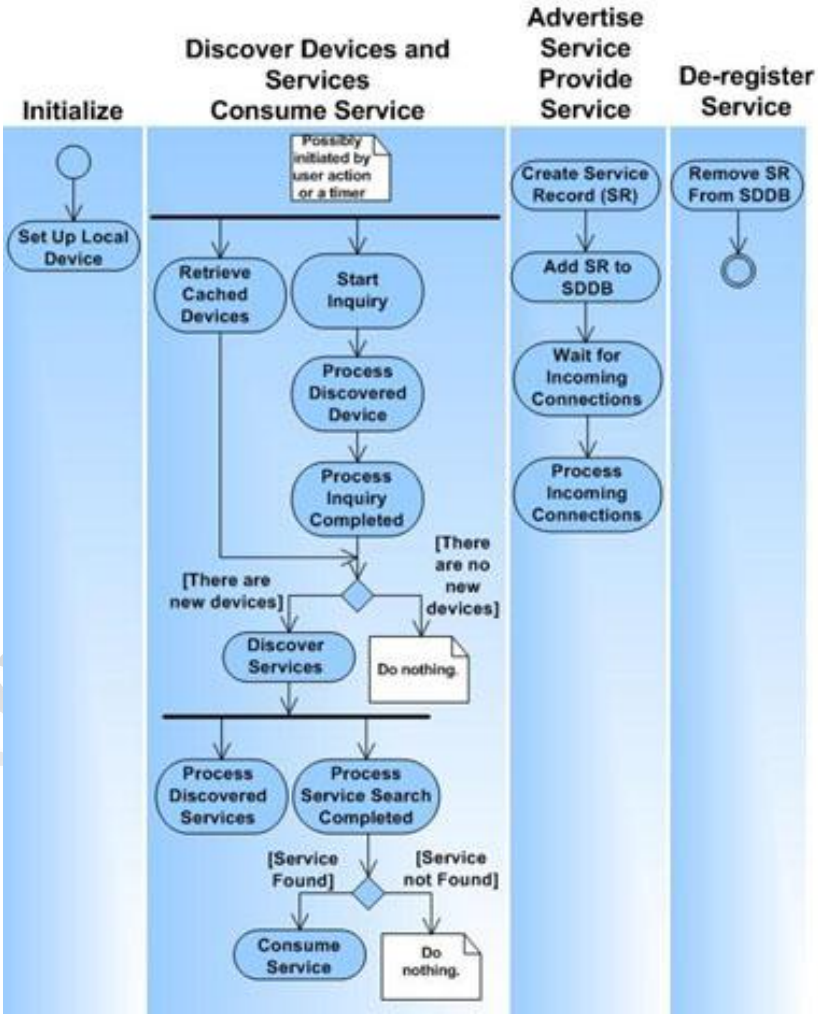
6.3 Σχεδίαση

Στη φάση της σχεδίασης καθορίζεται από ποια βασικά δομικά στοιχεία θα αποτελείται το λογισμικό (π.χ. δομές δεδομένων, κλάσεις, συναρτήσεις. Η φάση της σχεδίασης, βασισμένη στα αποτελέσματα που προέκυψαν από την προηγούμενη φάση της ανάλυσης, ξεκινά μια δομημένη λεπτομερή περιγραφή της εφαρμογής. Η σχεδίαση γίνεται προς την κατεύθυνση της λειτουργικότητας (πχ διεπαφές, περιεχόμενο, εικονίδια κλπ) και προς τη κατεύθυνση της τεχνικής σχεδίασης για τη δημιουργία της αρχιτεκτονικής και της δομής της εφαρμογής. Στη φάση αυτή τα διαγράμματα Κλάσεων (class) και Δραστηριοτήτων (activity) της προηγούμενης φάσης συμπληρώνονται με περισσότερες λεπτομέρειες ενώ δημιουργούνται τα διαγράμματα Ακολουθίας (sequence) και Καταστάσεων (state). Στο στάδιο αυτό λαμβάνονται αποφάσεις για την πλατφόρμα, το λογισμικό ανάπτυξης, τις Βάσεις Δεδομένων, τη μορφή των αρχείων κλπ.

Η πλατφόρμα του Android υποστηρίζει τη λειτουργία των Bluetooth δικτύων τα οποία επιτρέπουν τη δωρεάν ασύρματη ανταλλαγή δεδομένων μεταξύ των συσκευών που διαθέτουν την υπηρεσία του Bluetooth. Το πλαίσιο λειτουργίας αυτής της εφαρμογής παρέχει πρόσβαση στις λειτουργίες Bluetooth μέσω του Android Bluetooth API. Η εφαρμογή του Bluetooth chat επιτρέπει στις φορητές συσκευές να συνδέονται μέσω του Bluetooth για την ανταλλαγή μηνυμάτων και έχει σχεδιαστεί για κοινωνικούς σκοπούς.

6.3.1 Το τελικό Διάγραμμα Δραστηριότητων της δημιουργίας και διαχείρισης της σύνδεσης Bluetooth

Τα Δ.Δ. που προήχθησαν τμηματικά κατά την διαδικασία της ανάλυσης τώρα συνθέτονται στην δημιουργία ενός ενιαίου Δ.Δ.

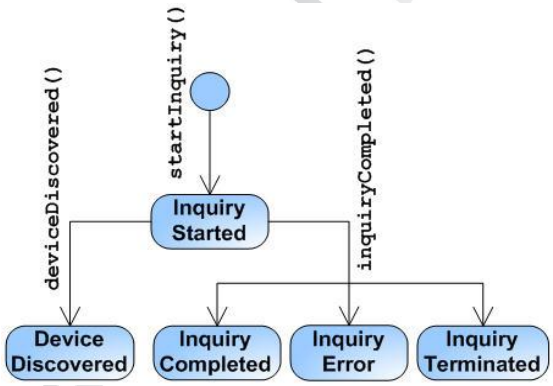


Διάγραμμα 6.4 Το Διάγραμμα Δραστηριοτήτων των εφαρμογών που διαθέτουν Bluetooth

Στο παραπάνω Δ.Δ. γίνεται η διάκριση των διαδικασιών που υλοποιούνται, όπως αυτές αναφέρονται στα προηγούμενα Διαγράμματα των Περιπτώσεων Χρήσης και των Δραστηριοτήτων χωρίς όμως να γίνεται διάκριση των ρόλων των συσκευών. Είναι το Δ.Δ. που προκύπτει από το Δ.Π.Χ. της ενεργοποίησης και διαχείρισης της σύνδεσης Bluetooth. [41]

6.3.2 Το Διάγραμμα Καταστάσεων της διαδικασίας ανίχνευσης συσκευών

Η διαδικασία της ανίχνευσης συσκευών είναι όμοια για όλες τις εφαρμογές που κάνουν χρήση της τεχνολογίας Bluetooth. Το Διάγραμμα Καταστάσεων της διαδικασίας ανίχνευσης συσκευών που ακολουθεί βρίσκεται στη σελίδα της oracle και εξηγεί τη λειτουργία αυτής της διαδικασίας.



Διάγραμμα 6.5 Το Διάγραμμα Κατάστασεων της ανίχνευσης συσκευών

Από την πλευρά της συσκευής που εκτελεί την ανίχνευση DiscoveryAgent εκτελούνται οι μέθοδοι εντοπισμού και ακύρωσης της ανακάλυψης της συσκευής:

- η μέθοδος retrieveDevices() ανακτά τις ήδη αντιστοιχισμένες από παλιότερη διαδικασία σύνδεσης γνωστές συσκευές που βρίσκονται σε κοντινή απόσταση.
- η μέθοδος startInquiry() θα ξεκινήσει την ανακάλυψη ή ανίχνευση των νέων κοντινών συσκευών.
- η μέθοδος cancelInquiry() ακυρώνει οποιαδήποτε διαδικασία ανίχνευσης που βρίσκεται σε εξέλιξη.
- η μέθοδος deviceDiscovered() δηλώνει αν έχει ανακαλυφθεί μια συσκευή.

η inquiryCompleted() δηλώνει αν η έρευνα έχει πετύχει, προκάλεσε ένα σφάλμα, orbeen ακυρωθεί.

Η Ανίχνευση των Συσκευών ξεκινά με την κλήση της μεθόδου startInquiry(). Καθώς προχωρά η διαδικασία, ο Bluetooth Discovery Agent επικαλείται τις μεθόδους επιστροφής κλήσης deviceDiscovered() και inquiryCompleted(), ανάλογα με την περίπτωση. [40], [41]

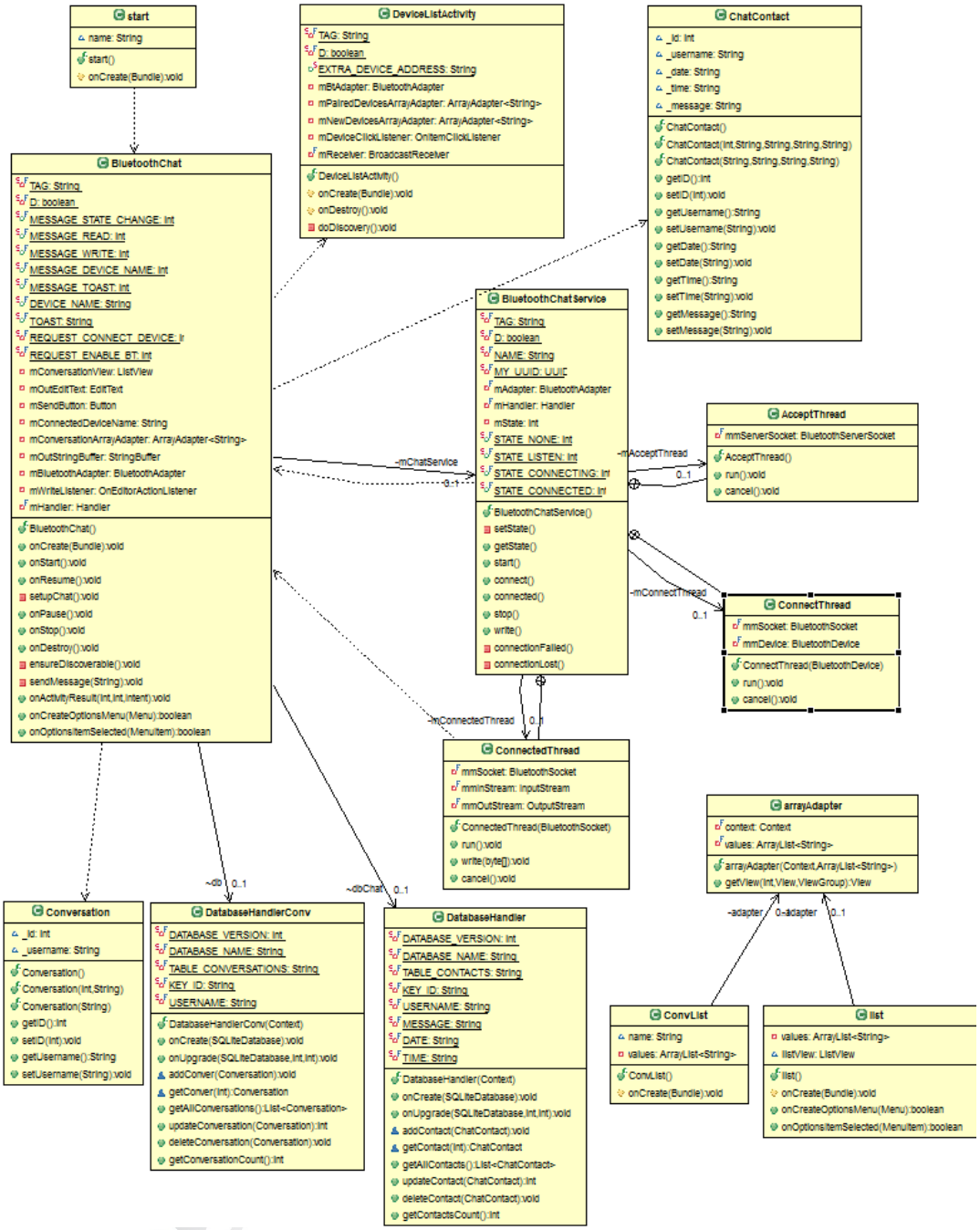
6.3.3 Το Διάγραμμα Κλάσεων της εφαρμογής

Στη κατηγορία των διαγραμμάτων δομής συμπεριλαμβάνεται το Διάγραμμα Κλάσεων (Δ.Κ.) το οποίο είναι το βασικότερο των διαγραμμάτων της UML. Ένα Δ.Κ. δείχνει την στατική δομή των κλάσεων του συστήματος και των σχέσεων μεταξύ τους.

Το Δ.Κ. αποτελείται από κλάσεις (classes), διαπροσωπίες (interfaces), συνεργασίες (collaborations) και συσχετίσεις (relationships). Και χρησιμοποιείται κατά την διάρκεια της ανάλυσης για να περιγράψει, τις λειτουργικές απαιτήσεις (functional requirements). Κατά τη διάρκεια του σχεδιασμού για να περιγράψει, το λεξιλόγιο του συστήματος (system's vocabulary), τις συσχετίσεις (relationships) και το λογικό σχήμα της βάσης δεδομένων (logical database schema) και τέλος κατά τη διάρκεια της υλοποίησης.

Στο Δ.Κ. του κώδικα της εφαρμογής διαφαίνονται όλες οι κλάσεις της εφαρμογής και οι συσχετίσεις που υφίστανται, δηλαδή ο τρόπος επικοινωνίας των κλάσεων.

Η κλάση ορίζει ένα σύνολο αντικειμένων. Η δομή της κάθε κλάσης αποτελείται από τρία μέρη στο πρώτο αναγράφεται το όνομα της κλάσης, στο δεύτερο περιλαμβάνονται τα πεδία (attributes) οι σταθερές και οι μεταβλητές καθώς και ο τύπος με τον οποίο δηλώνονται. Τέλος στο τρίτο μέρος περιέχονται οι μέθοδοι (operations) και δημιουργοί (constructors) της κλάσης.



Διάγραμμα 6.6 Το Διάγραμμα Κλάσεων του κώδικα της εφαρμογής

6.4Σύνοψη

Τα αποτελέσματα των προηγούμενων φάσεων της ανάλυσης και της σχεδίασης αξιοποιούνται στη φάση της υλοποίησης. Στη φάση αυτή γίνεται η συγγραφή και η διόρθωση του κώδικα. Όπως γίνεται αντιληπτό όσο πιο λεπτομερές και πιο ορθό είναι το προϊόν των προηγούμενων φάσεων τόσο πιο εύκολη και χωρίς προβλήματα είναι η φάση αυτή.

7 Υλοποίηση της εφαρμογής του BluetoothChat

Μετά την ανάλυση και τη σχεδίαση των απαιτήσεων ακολουθεί η φάση της υλοποίησης της εφαρμογής. Με βάση τα διαγράμματα που παρήχθησαν στο προηγούμενο κεφάλαιο θα κατασκευαστεί ο κώδικας της εφαρμογής. Σ' αυτό το κεφάλαιο παρουσιάζεται και περιγράφεται συνοπτικά ο κώδικας της εφαρμογής. Η συγγραφή του κώδικα όπως έχει προαναφερθεί στην παράγραφο 5.2 «Οι απαιτήσεις της εφαρμογής ως προς το λογισμικό (software)» πραγματοποιήθηκε με το περιβάλλον ανάπτυξης λογισμικού Eclipse το οποίο περιλαμβάνει το Java IDE και το ADT. Για λόγους οργάνωσης ο κώδικας της εφαρμογής περιέχεται στο παράρτημα της πτυχιακής εργασίας.

7.1 Η δομή του κώδικα της εφαρμογής του BluetoothChat

Κάθε εφαρμογή Android αποτελείται από τα αρχεία του κώδικα της java τα οποία υλοποιούν τις ενέργειες που εκτελεί η εφαρμογή και τα αρχεία του κώδικα xml τα οποία υλοποιούν το γραφικό περιβάλλον της εφαρμογής. Ο κώδικας της εφαρμογής του BluetoothChat μπορεί να επιμεριστεί σε τρία μέρη.

Στο 1^{ο} μέρος γίνεται η παρουσίαση της αρχικής οθόνης και ποιές επιλογές έχει ο χρήστης κατά την είσοδο του στο API.

Στο 2^{ο} μέρος υλοποιούνται οι βασικές εργασίες για την επικοινωνία μέσω της υπηρεσίας του Bluetooth. Αρχικά γίνεται ο έλεγχος για την υποστήριξη της υπηρεσίας του Bluetooth από τη συσκευή και η εγκατάσταση (setup) του. Αμέσως μετά ανακτώνται (αν υπάρχουν) και παρουσιάζονται, οι συσκευές με τις οποίες η συσκευή είχε συνδεθεί στο παρελθόν. Στη συνέχεια εντοπίζονται οι νέες συσκευές που είναι διαθέσιμες εντός της εμβέλειας της περιοχής ανίχνευσης της συσκευής και πραγματοποιείται η μεταφορά των δεδομένων των νέων συσκευών. Τέλος δημιουργείται ένα socket για τη σύνδεση της συσκευής με τη συσκευή που έχει επιλέξει ο χρήστης από τη λίστα των συσκευών η οποία περιέχει τις συσκευές με τις οποίες είχε συνδεθεί παλιότερα και τις νέες συσκευές που εντοπίστηκαν. Ο κώδικας όλων των Bluetooth APIs είναι διαθέσιμος στα παραδείγματα του SDK (<*sdk*>/platforms/android-<*version*>/samples/).

Το 3^{ο} μέρος του κώδικα δημιουργεί και διαχειρίζεται τις Βάσεις Δεδομένων. Η εφαρμογή χρησιμοποιεί δύο ΒΔ για την αποθήκευση και τη διαχείριση των επαφών και των συνομιλιών.

7.2 Η αρχιτεκτονική της εφαρμογής του BluetoothChat

Πριν προχωρήσουμε στην ανάλυση του κώδικα της εφαρμογής, θα ήταν χρήσιμο να παρουσιάσουμε πρώτα τη δομή του πακέτου, πως δηλαδή οργανώνονται τα αρχεία και ποια είναι η κύρια λειτουργία τους. Όλα τα αρχεία της εφαρμογής βρίσκονται στο πακέτο **com.example.android.BluetoothChat.**

Κατά τη δημιουργία του Android Project από το περιβάλλον ανάπτυξης λογισμικού του Eclipse το οποίο χρησιμοποιήθηκε για την συγγραφή του κώδικα της εφαρμογής, δημιουργείται ταυτόχρονα και η δομή του project όπως παρουσιάζεται στην Εικόνα 7.1 Η δομή του πακέτου BluetoothChat με τους εξής φακέλους:

Android 4.3: Πρόκειται για τις βιβλιοθήκες που επιλέγονται αυτόματα από το σύστημα, όταν δηλωθεί η ελάχιστη έκδοση του Λ.Σ. Android με την οποία είναι συμβατή η εφαρμογή. Αρχικά, κατά την δημιουργία ενός project μιας εφαρμογής Android εμφανίζονται παράθυρα διαλόγου με διάφορες επιλογές, μια απ' αυτές είναι το Build Target της εφαρμογής στο οποίο δηλώνεται η ελάχιστη έκδοση του Λ.Σ. του Android με την οποία θα είναι συμβατή η εφαρμογή. Το σύστημα τοποθετεί τις κατάλληλες βιβλιοθήκες για τη λειτουργία της εφαρμογής στην ελάχιστη έκδοση που δηλώσαμε (ανάλογη εγκατάσταση των εκδόσεων αυτών πρέπει να γίνει και στο ADT Plugin). Η εφαρμογή που δημιουργούμε είναι συμβατή με τις συσκευές που λειτουργούν με την έκδοση 4.3 του Android και τις νεότερες εκδόσεις του. Η δομή του πακέτου BluetoothChat περιέχει τους παρακάτω φακέλους.

src: Περιέχει όλα τα αρχεία του κώδικα της java, περιλαμβάνει τις κλάσεις της java της εφαρμογής.

gen: Περιέχει τα αρχεία R.java και BuildConfig.java τα οποία κατασκευάζονται και μορφοποιούνται αυτόματα από το σύστημα. Το αρχείο R.java είναι μια κλάση η οποία χρησιμοποιείται για την σύνδεση των πόρων (αρχεία του

φακέλου res) με τον κώδικα της java της εφαρμογής. Το αρχείο R.java παρουσιάζεται αναλυτικότερα παρακάτω στην παράγραφο 7.3.3 «Η κλάση R.java – πρόσβαση στους πόρους».

assets: Στο φάκελο αυτό μπορούμε να αποθηκεύσουμε οποιοδήποτε άλλο αρχείο θέλουμε ή ακόμα μπορούμε να δημιουργήσουμε υποφακέλους. Είναι ένας βοηθητικός φάκελος για την αποθήκευση των πόρων παρόμοιος με τον φάκελο res.

res: Ο φάκελος αυτός περιέχει υποφακέλους με τους πόρους της εφαρμογής (application resources)**.** Οι σημαντικότεροι αυτών των υποφακέλων είναι οι υποφάκελοι drawable, layout, menu και ο values.

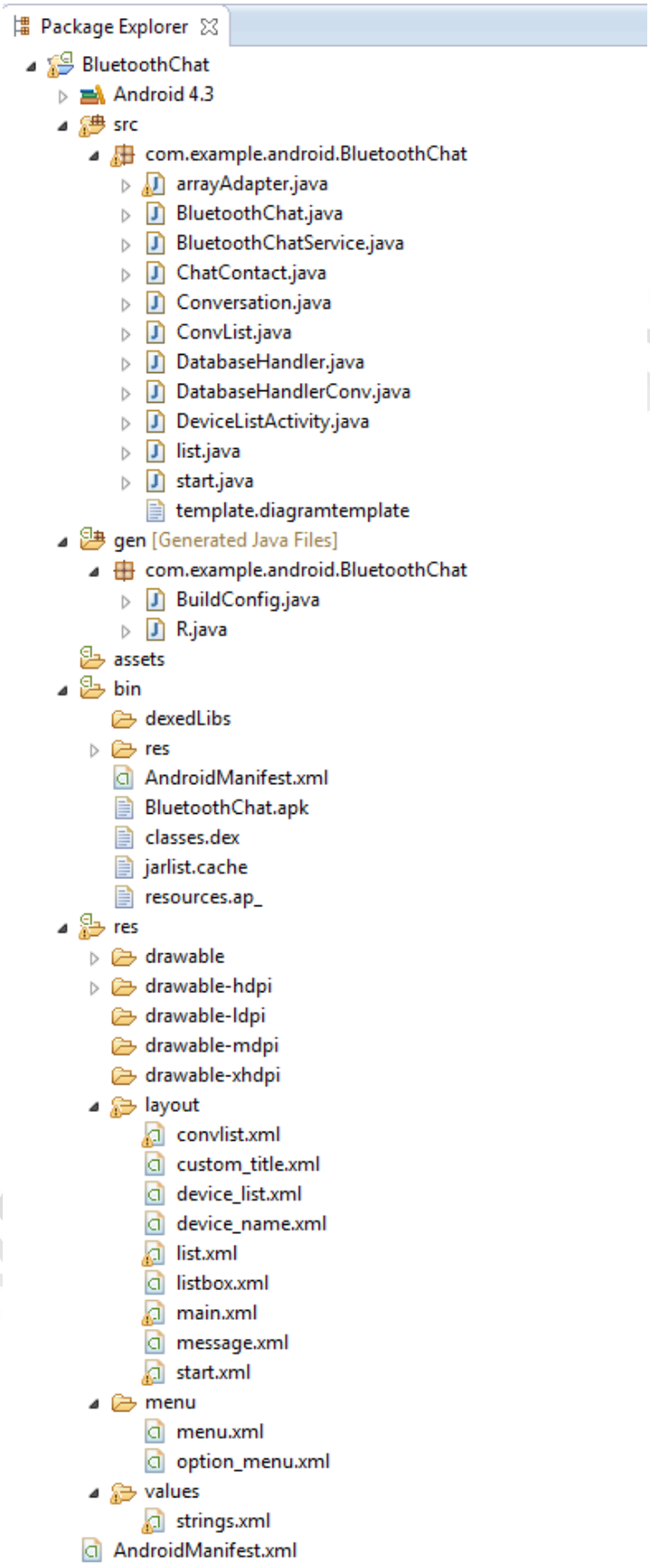
drawable: Περιέχει τις εικόνες που χρησιμοποιούνται στην εφαρμογή**.**

layout: Περιέχει τα αρχεία που καθορίζουν την όψη και τα γραφικά της εφαρμογής.

menu: τοποθετούνται τα μενού του κώδικα (πχ option menu) τα οποία θεωρούνται μέρος των πόρων.

values: αποθηκεύονται οι μεταβλητές του κώδικα.

AndroidManifest.xml: Το AndroidManifest.xml δεν είναι φάκελος αλλά ένα σημαντικό αρχείο του κώδικα της κάθε εφαρμογής. Σε αυτό το αρχείο ορίζονται οι activities, οι content providers, οι services και τα intent receiver's της εφαρμογής. Επίσης στο αρχείο αυτό δηλώνονται οι άδειες των υπηρεσιών που απαιτούνται από την εκάστοτε εφαρμογή. Το αρχείο αυτό παρουσιάζεται αναλυτικότερα στην παράγραφο 7.4.1 «Ο ρόλος του αρχείου AndroidManifest.XML». [31]



Εικόνα 7.1 Η δομή του *πακέτου* BluetoothChat

7.3 Εκκίνηση της εφαρμογής

7.3.1 Η οθόνη εκκίνησης της εφαρμογή, το αρχείο start.xml

Τα αρχεία που υλοποιούν το γραφικό μέρος της εφαρμογής θεωρούνται ως πηγές και για το λόγο αυτό αποθηκεύονται στο φάκελο res (resource) και ειδικά στο φάκελο των γραφικών layout.

Το αρχείο start.xml (BluetoothChat\res\layout\start.xml) που παρουσιάζεται στην παρακάτω εικόνα, περιέχει το σχεδιαστικό κομμάτι της αρχικής οθόνης.

Εικόνα 7.2 Η αρχική οθόνη της εφαρμογής

Στο σημείο αυτό παρατίθεται ο κώδικας του γραφικού μέρους της αρχικής οθόνης ο οποίος είναι γραμμένος στη γλώσσα προγραμματισμού xml. Τα βασικά του σημεία του κώδικα αυτού είναι ο προσδιορισμός του backround με τη χρήση της εικόνας back η οποία βρίσκεται στο φάκελο res (resource) και ειδικότερα στον υποφάκελο των εικόνων drawable. Στη συνέχεια σχεδιάζονται τα κουμπιά button1 και button2, CHAT και CONVERSATIONS αντίστοιχα. Ορίζεται η περιγραφή, η θέση, το backround των κουμπιών καθώς και το χρώμα, το στυλ και το μέγεθος της γραμματοσειράς που θα χρησιμοποιηθεί.

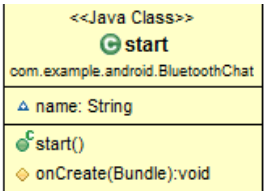
*start.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/back" >

    <Button
        android:id="@+id/button1"
        android:layout_width="200dp"
        android:layout_height="50dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="150dp"
        android:background="@drawable/but"
        android:text="CHAT"
        android:textColor="#ffffff"
        android:textStyle="bold"
        android:textSize="20sp" />

    <Button
        android:id="@+id/button2"
        android:layout_width="200dp"
        android:layout_height="50dp"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/button1"
        android:layout_marginTop="100dp"
        android:background="@drawable/but"
        android:text="CONVERSATIONS"
        android:textColor="#ffffff"
        android:textStyle="bold"
        android:textSize="20sp" />

</RelativeLayout>
```

Εικόνα 7.3 Ο κώδικας του αρχείου Start.xml

7.3.2 Η κλάση start

Στην κλάση αυτή δημιουργείται η Βάση Δεδομένων και αρχικοποιείται με την εισαγωγή κάποιων δεδομένων που προσομοιώνουν προηγούμενες συζητήσεις έτσι ώστε να υπάρχουν κάποιες συνομιλίες κατά την δημιουργία του API. Εκτενέστερη περιγραφή της δημιουργίας της ΒΔ γίνεται στις επόμενες παραγράφους. Στην παρακάτω εικόνα παρουσιάζεται η δομή της κλάσης start στην οποία διακρίνονται στο πρώτο τμήμα το όνομα της κλάσης και του πακέτου, στο επόμενο τμήμα δηλώνονται οι μεταβλητές και οι σταθερές της κλάσης, στη συγκεκριμένη περίπτωση δηλώνεται η μεταβλητή name ενώ στο τελευταίο τμήμα δηλώνονται ο constructor της start και η μέθοδος onCreate().



Εικόνα 7.4 Η κλάση start

Στην εικόνα που ακολουθεί παρατίθεται ο κώδικας της κλάσης αυτής. Η κλάση start επεκτείνει (extend) μια δραστηριότητα. Όλες οι λειτουργίες που επιτελεί η start εκτελούνται μέσα στη μέθοδο onCreate. Η μέθοδος onCreate είναι η μόνη μέθοδος που της κλάσης start πλην του constructor της κλάσης. Στην Εικόνα 7.5 στο σημείο -1- με την κλήση της setContentView(R.layout.start) γίνεται η σύνδεση της κλάσης start με το αρχείο start.xml το οποίο περιέχει το γραφικό περιβάλλον της αρχικής οθόνης της εφαρμογής και που παρουσιάστηκε στην προηγούμενη παράγραφο. Αμέσως μετά δημιουργείται το αντικείμενο db το οποίο είναι τύπου DatabaseHandler για την κατασκευή και διαχείριση της ΒΔ. Στο σημείο -2- της Εικόνα 7.5 ο μηχανισμός ελέγχου της if (db.getAllContacts().isEmpty()) ελέγχει αν η βάση db είναι κενή, τότε εισάγονται οι εγγραφές στη βάση. Με παρόμοιο τρόπο δημιουργείται, στο σημείο -3- του παρακάτω κώδικα, η ΒΔ db1 η οποία αποθηκεύει τα ονόματα των επαφών που πραγματοποιούν τις συνομιλίες.

```
public class start extends Activity {
    String name;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.start);
        final DatabaseHandler db = new DatabaseHandler(this);

        if (db.getAllContacts().isEmpty()) {
            db.addContact(new ChatContact("Maria","13/10/2013","15:40:43","Maria: \n hello "));
            db.addContact(new ChatContact("Maria","13/10/2013","15:41:43","Me: \n hi, ti kaneis? "));
            db.addContact(new ChatContact("Maria","13/10/2013","15:42:43","Maria: \n kala mole barieme "));

            db.addContact(new ChatContact("Nikos","14/10/2013","15:40:43","Nikos: \n simera stis 5 "));
            db.addContact(new ChatContact("Nikos","14/10/2013","15:41:43","Me: \n ok "));

            db.addContact(new ChatContact("Maria","17/10/2013","15:40:43","Maria: \n Pos pige me ton Niko? "));
            db.addContact(new ChatContact("Maria","17/10/2013","15:41:43","Me: \n Kala , ligo baretos "));
            db.addContact(new ChatContact("Maria","17/10/2013","15:42:43","Maria: \n Giati to les? mia xara pedi fenete "));
            db.addContact(new ChatContact("Maria","17/10/2013","15:43:43","Me: \n den ksero, den perasa, nistaksa "));
            db.addContact(new ChatContact("Maria","17/10/2013","15:44:43","Maria: \n a kalaaaaa, katalaba... pame gia ala "));

            db.addContact(new ChatContact("Manos","19/10/2013","15:40:43","Me: \n Geia sou Mano, pos paei i ergasia? "));
            db.addContact(new ChatContact("Manos","19/10/2013","15:41:43","Manos: \n asta eimai gia kafe...etsi paei. "));

            db.addContact(new ChatContact("John","07/02/2014","09:30:43","John: \n Ti egine me ti nerit? Bgike prokiriksi?"));
            db.addContact(new ChatContact("John","07/02/2014","09:30:57","Me: \n Nai xtes, alla exei liges thesis "));
            db.addContact(new ChatContact("John","07/02/2014","09:31:15","John: \n Giati etsi? "));
            db.addContact(new ChatContact("John","07/02/2014","09:31:39","Me: \n den ksero, ti na po, xalia "));
            db.addContact(new ChatContact("John","07/02/2014","09:32:03","John: \n a kalaaaaa, katalaba... "));

            final DatabaseHandlerConv db1 = new DatabaseHandlerConv(this);
            db1.addConver(new Conversation("Maria"));
            db1.addConver(new Conversation("Nikos"));
            db1.addConver(new Conversation("Manos"));
            db1.addConver(new Conversation("John"));
        }
        Button chat1 = (Button) findViewById(R.id.button1);
        Button conv1 = (Button) findViewById(R.id.button2);

        chat1.setOnClickListener(new OnClickListener() {
             public void onClick(View v) {
                Intent intent1 = new Intent();
                intent1.setClass(start.this, BluetoothChat.class);
                startActivity(intent1);
            }
        });

        conv1.setOnClickListener(new OnClickListener() {
             public void onClick(View v) {
                Intent intent1 = new Intent();
                intent1.setClass(start.this, list.class);
                startActivity(intent1);
            }
        });
    }
```

Εικόνα 7.5 Ο κώδικας της κλάσης Start

Στη συνέχεια στο σημείο 4 της προηγούμενης εικόνας πραγματοποιείται η σύνδεση των κουμπιών button1 και button2 τα οποία βρίσκονται στο γραφικό περιβάλλον του αρχείου start.xml (βλ. προηγούμενη παράγραφο) με τα κουμπιά chat1 και conv1 του κώδικα της java τα οποία θα εκτελέσουν αντίστοιχα τις διαδικασίες της συνομιλίας και της αναζήτησης των συνομιλιών που έχουν αποθηκευτεί στη ΒΔ.

Οι Android developers χρησιμοποιούν συχνά ανώνυμες εσωτερικές κλάσεις για να ορίσουν εξειδικευμένους listeners, οι οποίοι χρησιμοποιούνται για να καταγράφουν τις επιστροφές κλήσης μιας συγκεκριμένης συμπεριφοράς όταν συμβαίνει ένα γεγονός. Για παράδειγμα, όταν ο χρήστης κάνει κλικ πάνω σε ένα στοιχείο που εμφανίζεται σε μία οθόνη, ο προγραμματιστής καλεί την μέθοδο setOnClickListener(), η οποία λαμβάνει μία μόνο παράμετρο: ένα αντικείμενο View.OnClickListener.

Οι προγραμματιστές χρησιμοποιούν την τεχνική της ανώνυμης εσωτερικής κλάσης για τη δημιουργία, τον ορισμό και τη χρήση του View.OnClickListener όπως συμβαίνει στα σημεία -5- και -6- του παραπάνω κώδικα.

Η δημιουργία της πρόθεσης intent γίνεται για τον καθορισμό με τη μέθοδο setClass της onCreate στην οποία περιέχεται το στοιχείο της κλάσης από την οποία ξεκινάει η πρόθεση και η κλάση στην οποία βρίσκεται η λειτουργία που θα ενεργοποιηθεί με την επιλογή της πρόθεσης αυτής.

```
intent1.setClass(start.this, BluetoothChat.class);
```

ή

```
intent1.setClass(start.this, list.class);
```

Και στις δύο περιπτώσεις ενεργοποιείται η δραστηριότητα startActivity() για να ξεκινήσει ο κύκλος της ζωής της εφαρμογής.

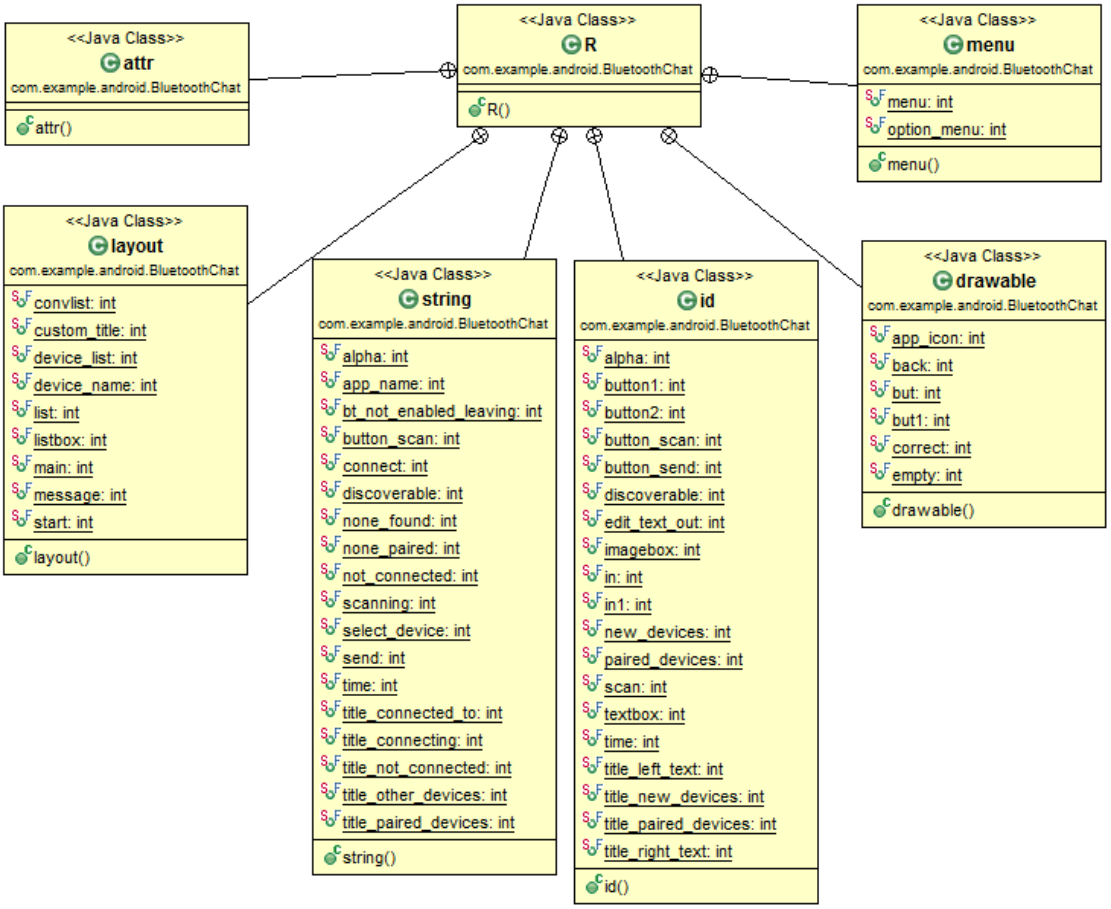
```
startActivity(intent1);
```

Έτσι γίνεται η σύνδεση των δύο κουμπιών που εμφανίζονται στο αρχείο start.xml με τη λειτουργία που πρέπει να επιτελέσουν, δηλαδή με τον κώδικα της java.

7.3.3 Η κλάση R.java – πρόσβαση στους πόρους

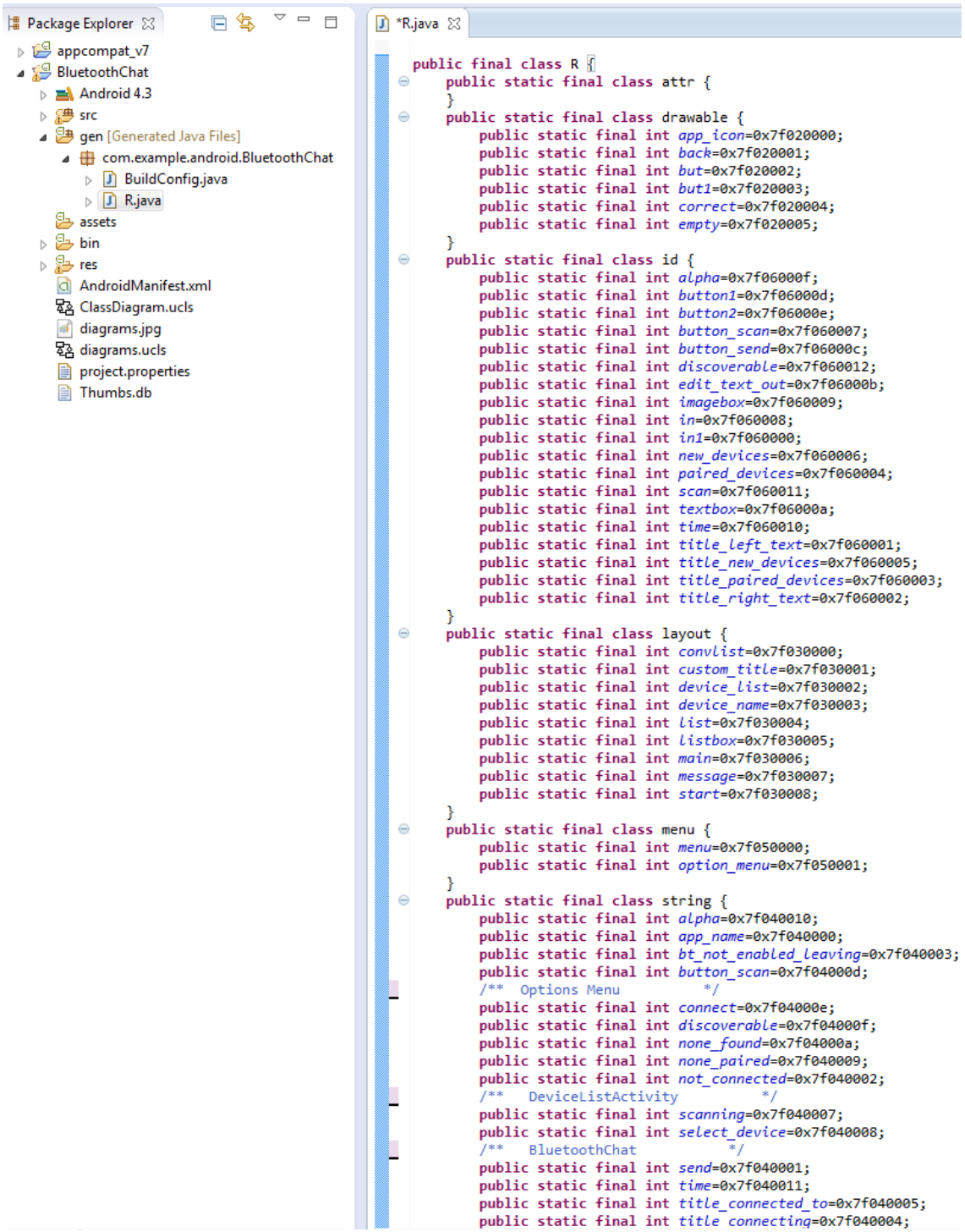
Η κλάση R είναι ένα σημαντικό κομμάτι του κώδικα κάθε android εφαρμογής. Η κλάση αυτή δημιουργείται αυτόματα από το σύστημα κατά τη δημιουργία του project. Δεν ενημερώνεται από τον προγραμματιστή αλλά ενημερώνεται αυτόματα από το σύστημα ανάλογα με τις αλλαγές που γίνονται κατά τη διαδικασία συγγραφής του κώδικα του API.

Η κλάση R βρίσκεται στο φάκελο BluetoothChatFinally\BluetoothChat\gen\ και περιέχει τις κλάσεις των πόρων anim, animator, array, attr, bool, color, dimen, drawable, fraction, id, integer, interpolator, layout, menu, mipmap, plurals, raw, string, style, styleable, xml. Γι' αυτή την εφαρμογή το περιβάλλον ανάπτυξης λογισμικού Eclipse δημιούργησε την κλάση R η οποία περιέχει τις υποκλάσεις attr, drawable, id, layout, menu και string (Εικόνα 7.6) γιατί μόνο τέτοιους πόρους χρειάζεται για τη λειτουργία της η εφαρμογή του BluetoothChat.



Εικόνα 7.6 Οι υποκλάσεις της κλάσης R

Οι πόροι που χρησιμοποιεί η εφαρμογή πρέπει να δηλωθούν με το αναγνωριστικά id τους στην κλάση R του API.



Εικόνα 7.7 Ο κώδικας του αρχείου R της java

Όταν η εφαρμογή περάσει τη διαδικασία του compile, το εργαλείο AAPT δημιουργεί την κλάση R (Εικόνα 7.7), η οποία περιέχει τα αναγνωριστικά id's όλων των πόρων στο φάκελο res (resources). Για κάθε είδος πόρου, υπάρχει μια υποκλάση

στην R (για παράδειγμα, η υποκλάση R.drawable είναι η υποκλάση που περιέχει όλους τους πόρους σχεδιασμού) και για κάθε πόρο αυτού του τύπου, υπάρχει ένα ακέραιος αριθμός (για παράδειγμα για τον πόρο R.drawable.app_icon ο 0x7f020000). Αυτός ο ακέραιος είναι το αναγνωριστικό id του πόρου που χρησιμοποιείται για την ανάκτηση των πόρων.

Ένα αναγνωριστικό πόρου αποτελείται από:

- τον τύπο του πόρου (π.χ. string, drawable ή layout)
- το όνομα της πηγής, το οποίο μπορεί να είναι είτε το όνομα του αρχείου χωρίς την επέκταση του, είτε η τιμή ενός χαρακτηριστικού του XML αρχείου π.χ. android:name

Η πρόσβαση μιας πηγής μπορεί να γίνει με δύο τρόπους είτε μέσα από τον κώδικα της java είτε μέσα από ένα xml αρχείο.

7.4 Σύνδεση στο δίκτυο Bluetooth

Η συσκευή στην οποία θα τρέξει εφαρμογή αυτή θα πρέπει να υποστηρίζει την υπηρεσία του Bluetooth και να διαθέτει το κανάλι RFCOMM. Στις προδιαγραφές της συσκευής θα πρέπει να περιλαμβάνονται τα σχετικά προφίλ του Bluetooth π.χ. τα A2DP, AVRCP, OBEX, κ.ο.κ., τα οποία είναι κατάλληλα για τη συσκευή (βλέπε παράγραφο 1.6.2 «Προδιαγραφές του προφίλ χρήσης»). [42]

7.4.1 Ο ρόλος του αρχείου AndroidManifest.XML

Το αρχείο AndroidManifest.xml (BluetoothChat\AndroidManifest.xml) βρίσκεται στο πακέτο **com.example.android.BluetoothChat** και είναι απαραίτητο συστατικό κάθε εφαρμογής Android. Περιέχει σημαντικές πληροφορίες της εφαρμογής που αφορούν το Λ.Σ. του Android οι οποίες απαιτούνται από το σύστημα προκειμένου να μπορέσει να τρέξει τον κώδικα της εφαρμογής. Τα διάφορα συστατικά στοιχεία της εφαρμογής, όπως είναι οι δραστηριότητες (activities), οι υπηρεσίες (services), οι πάροχοι περιεχομένου κλπ, ορίζονται στο αρχείο αυτό. Επίσης το αρχείο AndroidManifest χρησιμοποιείται για τη δήλωση των διαφορετικών ειδών άδειών για

τη χρήση των υπηρεσιών όπως είναι η πρόσβαση στο bluetooth, η χρήση της κάμερας, η επικοινωνία με το NFC (Near Field Communication) και άλλες υπηρεσίες.

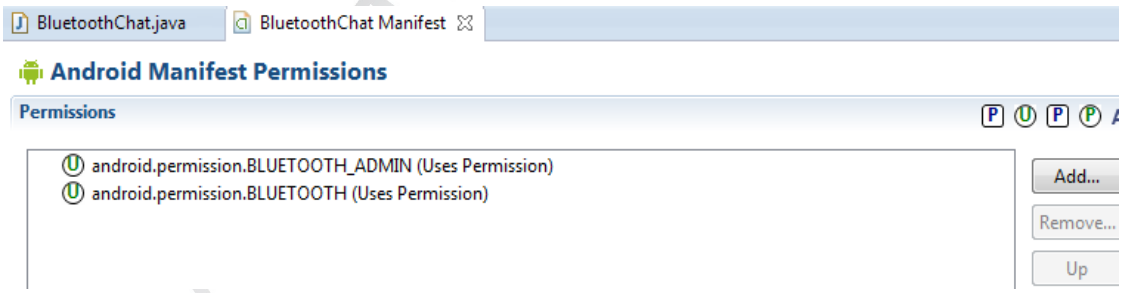
Οι εφαρμογές Android για λόγους συμβατότητας μπορούν να μεταγλωττιστούν από διαφορετικές εκδόσεις του εργαλείου SDK (software development kit). Η σήμανση <uses-sdk> καθορίζει το ελάχιστο SDK που απαιτείται να διαθέτει η συσκευή για να δομηθεί και να εκτελεστεί σωστά η εφαρμογή ενώ το targetSdkVersion (Εικόνα 7.9 σημείο 1), δηλώνει τη μέγιστη έκδοση του SDK.

<uses-sdk android:minSdkVersion="**6**"

android:targetSdkVersion="**18**" />

Επίσης στο αρχείο AndroidManifest δηλώνονται οι άδειες (permissions) των συσκευών που πρόκειται να χρησιμοποιηθούν από την εκάστοτε εφαρμογή. Για την προσπέλαση του Bluetooth είναι απαραίτητο να δηλωθούν οι άδειες χρήσης του bluetooth καθώς και του Bluetooth Administrator. Το αρχείο AndroidManifest της εφαρμογής του BluetoothChat έχει μετονομαστεί σε BluetoothChat Manifest.

Η εκχώρηση των αδειών μπορεί να γίνει με δυο τρόπους είτε μέσα από το παράθυρο του Android Manifest Permissions όπως φαίνεται στην ακόλουθη εικόνα, είτε μέσα από τον κώδικα όπως περιγράφεται στο σημείο 1 της Εικόνα 7.9.



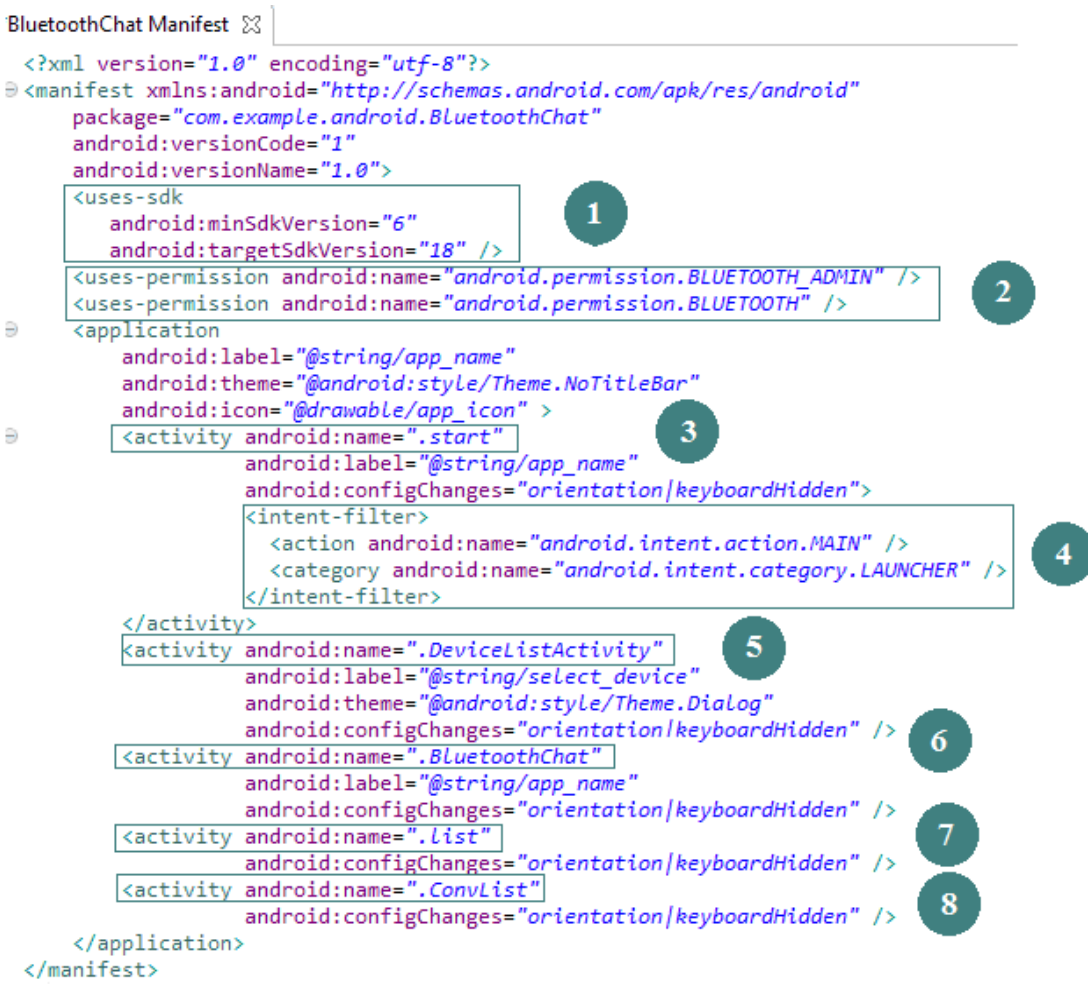
Εικόνα 7.8 Το παράθυρο του Android Manifest Permissions για την εκχωρηση αδειών

Για τον εντοπισμό της υπηρεσίας του Bluetooth η εφαρμογή πρέπει να διαθέτει την άδεια χρήσης (permission) του BLUETOOTH και την άδεια του BLUETOOTH_ADMIN για τη διαχείριση των ρυθμίσεων του Bluetooth ή την ανίχνευση των συσκευών.

<uses-permission android:name="**android.permission.BLUETOOTH_ADMIN**" />

<uses-permission android:name="**android.permission.BLUETOOTH**" />

Για την εκχώρηση των αδειών χρήσης απαιτούνται οι παραπάνω γραμμές του κώδικα οι οποίες διακρίνονται και στην Εικόνα 7.9 στο σημείο 2.



Εικόνα 7.9 Ο κώδικας του BluetoothChat Manifest.xml

Οι εφαρμογές Android αποτελούνται από διάφορες δραστηριότητες (βλέπε παράγραφο 4.1 «Η δραστηριότητα (activity) και ο κύκλος της ζωής της»). Κάθε δραστηριότητα πρέπει να δηλωθεί στο αρχείο manifest με το όνομα της κλάσης στην οποία έχει δημιουργηθεί, για να μπορέσει να εκτελεστεί από την συσκευή. Η δραστηριότητα εκτελεί μια συγκεκριμένη εργασία που πρέπει να ολοκληρωθεί στην οθόνη της δραστηριότητας. Κάθε δραστηριότητα έχει τη δική της σήμανση <activity> μέσα στο αρχείο manifest. Η παρακάτω εντολή ορίζει την κλάση της δραστηριότητας με το όνομα start.

<activity android:name="**.start**"

Στο αρχείο AndroidManifest δηλώνεται η δραστηριότητα start (Εικόνα 7.9 σημείο 3) η οποία είναι η αρχική activity της εφαρμογής, καθώς και η επόμενη activity που θα κληθεί μετά από την επιλογή του χρήστη. Στο αρχείο αυτό δηλώνονται επίσης οι προθέσεις που ενδεχομένως να χρησιμοποιηθούν. Στην εφαρμογή αυτή ο χρηστής έχει τη δυνατότητα να επιλέξει την ενεργοποίηση της υπηρεσία του BluetoothChat η οποία θα καλέσει τις δραστηριότητες DeviceListActivity και BluetoothChat (Εικόνα 7.9 σημεία 5 και 6 αντίστοιχα).

activity android:name="**.DeviceListActivity**"

activity android:name="**.BluetoothChat**"

Στην περίπτωση που ο χρήστης επιλέξει την διαχείριση των συνομιλιών τότε θα κληθούν οι activities list και ConvList (Εικόνα 7.9 σημεία 7 και 8 αντίστοιχα).

activity android:name="**.list**"

activity android:name="**.ConvList**"

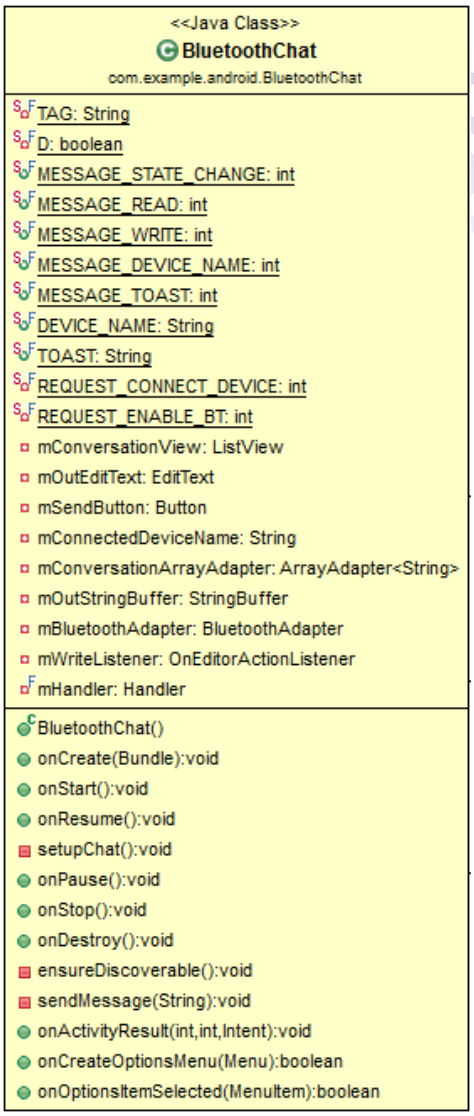
Ο κώδικας των xml αρχείων και των αρχείων του κώδικα της java αναλύονται στις επόμενες παραγράφους. [40]

7.4.2 Εγκατάσταση του Bluetooth

Πριν η εφαρμογή ξεκινήσει την επικοινωνία μέσω Bluetooth, θα πρέπει να γίνει έλεγχος εάν υποστηρίζεται η τεχνολογία αυτή από τη συσκευή και μετά να βεβαιωθούμε ότι η υπηρεσία αυτή είναι ενεργοποιημένη. Αν το Bluetooth δεν υποστηρίζεται από τη συσκευή, τότε θα πρέπει να απενεργοποιηθεί οποιαδήποτε λειτουργία του. Στην περίπτωση που υποστηρίζεται η υπηρεσία του Bluetooth από την συσκευή αλλά δεν έχει ενεργοποιηθεί τότε ο χρήστης έχει τη δυνατότητα να επιλέξει την ενεργοποίηση της υπηρεσίας αυτής χωρίς να χρειαστεί να βγει από την εφαρμογή. Αυτή η εγκατάσταση επιτυγχάνεται με τη χρήση της κλάσης BluetoothAdapter σε δύο στάδια. [43]

7.4.2.1 Η κλάση BluetoothChat

Η κλάση BluetoothChat είναι η κλάση που διαχειρίζεται το τμήμα του κώδικα που αφορά τη διαδικασία της συνομιλίας με την ανταλλαγή γραπτών μηνυμάτων. Η δομή της κλάσης παρουσιάζεται στην επόμενη εικόνα, στο δεύτερο μέρος της δομής της διακρίνουμε τις μεταβλητές και τις σταθερές της. Στο τελευταίο τμήμα περιέχονται ο constructor της BluetoothChat και οι μέθοδοι της κλάσης οι οποίες αναλύονται στις επόμενες παραγράφους.



Εικόνα 7.10 Η δομή της κλάσης BluetoothChat

Μεταξύ των μεθόδων αυτών διακρίνουμε:

- τις μεθόδους onCreate, onStart, onResume, onPause, onStop και onDestroy, οι οποίες είναι οι μέθοδοι που συνθέτουν τον κύκλο ζωής της εφαρμογής.

- την μέθοδο setupChat() για την εγκατάσταση του chat και την μέθοδο ensureDiscoverable() που επιτελεί την ανιχνευσιμότητα της συσκευής από τις άλλες συσκευές και
- την μέθοδο sendMessage(String) για την αποστολή των μηνυμάτων.
- την μέθοδο onActivityResult,
- την μέθοδο onCreateOptionsMenu και
- την μέθοδο onOptionsItemSelected

Η περιγραφή του κώδικα καθώς και οι λειτουργίες τις οποίες επιτελούν όλες οι παραπάνω μέθοδοι αναλύονται στις επόμενες παραγράφους. Στο σημείο 1 της Εικόνα 7.11 δηλώνονται και αρχικοποιούνται οι σταθερές, οι μεταβλητές, τα στοιχεία των views και οι ΒΔ. Στη συνέχεια καλείται η συνάρτηση onCreate, η οποία ενεργοποιείται κατά την εκκίνηση της εφαρμογής και η οποία περιγράφεται εκτενώς στην επόμενη παράγραφο.

BluetoothChat.java

```
package com.example.android.BluetoothChat;

import java.util.Calendar;

/**
 * This is the main Activity that displays the current chat session.
 */
public class BluetoothChat extends Activity {
    // Debugging
    private static final String TAG = "BluetoothChat";
    private static final boolean D = true;

    // Message types sent from the BluetoothChatService Handler
    public static final int MESSAGE_STATE_CHANGE = 1;
    public static final int MESSAGE_READ = 2;
    public static final int MESSAGE_WRITE = 3;
    public static final int MESSAGE_DEVICE_NAME = 4;
    public static final int MESSAGE_TOAST = 5;

    // Key names received from the BluetoothChatService Handler
    public static final String DEVICE_NAME = "device_name";
    public static final String TOAST = "toast";

    // Intent request codes
    private static final int REQUEST_CONNECT_DEVICE = 1;
    private static final int REQUEST_ENABLE_BT = 2;

    // Layout Views
    private ListView mConversationView;
    private EditText mOutEditText;
    private Button mSendButton;
```

1

Εικόνα 7.11 Η κλάση BluetoothChat (σημείο 1)

Η λειτουργία του Android στην πραγματικότητα περιγράφει έναν «κύκλο ζωής της δραστηριότητας», περιγράφει τα διάφορα στάδια της εφαρμογή. Για παράδειγμα, όταν μια εφαρμογή ξεκινάει, τρεις διαφορετικές μέθοδοι του κύκλου της «ζωής» της

συνυπάρχουν η OnCreate(), η OnStart() και η onResume(). Η αρχικοποίηση των σταθερών μιας κλάσης της εφαρμογής εμφανίζεται στην μέθοδο onCreate() γιατί αυτές πρέπει να πάρουν αρχικές τιμές κατά την εκκίνηση και για μία μόνο φορά (final). Στην μέθοδο onResume() αρχικοποιούνται οι μεταβλητές που πρέπει να λάβουν επανειλημμένα αρχικές τιμές στην περίπτωση που η συσκευή μπαίνει σε κατάσταση αναμονής ή χάσει τα περιεχόμενα της.

7.4.2.1.1 Ο έλεγχος για την υποστήριξη του BT από τη συσκευή

Η συνάρτηση onCreate (Εικόνα 7.12 σημείο 2), ενεργοποιείται κατά την εκκίνηση της εφαρμογής και χρησιμοποιείται για την αρχικοποίηση ενεργειών κατά την έναρξη της εφαρμογής και για μία μόνο φορά. Στην onCreate περιέχονται οι ενέργειες που δεν ξαναρχικοποιούνται όταν η συσκευή μπαίνει σε κατάσταση αναμονής. Στη μέθοδο onCreate(), κάποια μέρη του κώδικα δημιουργούνται αυτόματα από τον οδηγό κατά την κατασκευή του project. Οι εντολές αυτές είναι η super.onCreate(savedInstanceState) και η setContentView(R.layout.activity_main). Με τη setContentView γίνεται η σύνδεση με το αρχείο main.xml που είναι μια πηγή του σχεδιαστικού κώδικα, layout, το οποίο παρατίθεται στην παράγραφο 7.4.2.1.3 «Ο κώδικας του αρχείου πόρων της Main.xml». Έπειτα παρεμβάλλεται ο κώδικας που δημιουργήσαμε αφού πρώτα αρχικοποιηθούν οι δύο ΒΔ db και dbChat που ορίστηκαν προηγουμένως.

Η BluetoothAdapter απαιτείται από όλες τις δραστηριότητες που συμμετέχουν στην λειτουργία του Bluetooth. Για να χρησιμοποιηθεί η BluetoothAdapter, καλούμε μέσα από την δραστηριότητα onCreate τη static μέθοδο getDefaultAdapter(). Αυτή επιστρέφει μια private (ορατή μόνο στην κλάση) σταθερά BluetoothAdapter που αναπαριστά τη συσκευή που διαθέτει Bluetooth adapter, δηλαδή το σήμα του Bluetooth. Υπάρχει μόνο ένας Bluetooth adapter για ολόκληρο το σύστημα, και η εφαρμογή μπορεί να αλληλεπιδρά με την χρήση αυτού του αντικειμένου.

```
// Name of the connected device
private String mConnectedDeviceName = null;
// Array adapter for the conversation thread
private ArrayAdapter<String> mConversationArrayAdapter;
// String buffer for outgoing messages
private StringBuffer mOutStringBuffer;
// Local Bluetooth adapter
private BluetoothAdapter mBluetoothAdapter = null;
// Member object for the chat services
private BluetoothChatService mChatService = null;

DatabaseHandlerConv db;
DatabaseHandler dbChat;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    if(D) Log.e(TAG, "+++ ON CREATE +++");

    db = new DatabaseHandlerConv(this);
    dbChat = new DatabaseHandler(this);

    setContentView(R.layout.main);

    // Get local Bluetooth adapter
    mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

    // If the adapter is null, then Bluetooth is not supported
    if (mBluetoothAdapter == null) {
        Toast.makeText(this, "Η υπηρεσία Bluetooth δεν υποστηρίζεται.", Toast.LENGTH_LONG).show();
        finish();
        return;
    }
}
```

2

Εικόνα 7.12 Η κλάση BluetoothChat (σημείο 2)

Στο αντικείμενο mBluetoothAdapter, που έχει λάβει ήδη από την BluetoothChat την αρχική τιμή null, τώρα φορτώνεται ο adapter με την BluetoothAdapter.getDefaultAdapter. Ο μηχανισμός της if ελέγχει αν ο adapter έχει την τιμή null, δηλαδή εάν η τιμή του mBluetoothAdapter δεν έχει μεταβληθεί από την getDefaultAdapter. Αν η μέθοδος getDefaultAdapter() επιστρέψει την τιμή null, τότε σημαίνει ότι η συσκευή δεν υποστηρίζει την τεχνολογία Bluetooth και εμφανίζεται ένα toast μήνυμα ότι το Bluetooth δεν υποστηρίζεται από τη συσκευή (hardware).

```
public synchronized void onPause() {
    super.onPause();
    if(D) Log.e(TAG, "- ON PAUSE -");
}

@Override
public void onStop() {
    super.onStop();
    if(D) Log.e(TAG, "-- ON STOP --");
}

@Override
public void onDestroy() {
    super.onDestroy();
    // Stop the Bluetooth chat services
    if (mChatService != null) mChatService.stop();
    if(D) Log.e(TAG, "--- ON DESTROY ---");
}
```

3

Εικόνα 7.13 Η κλάση BluetoothChat (σημείο 3)

Στην περίπτωση που η συσκευή δεν υποστηρίζει την τεχνολογία του Bluetooth η ροή του προγράμματος μεταφέρεται στην onDestroy (Εικόνα 7.13, σημείο 3).

7.4.2.1.2 Ενεργοποίηση της λειτουργίας του Bluetooth

Στη συνέχεια, θα πρέπει να διασφαλιστεί ότι το Bluetooth είναι ενεργοποιημένο, ο έλεγχος αυτός πραγματοποιείται μέσα στη δραστηριότητα onStart(). Στη μέθοδο onStart(), στο σημείο 4 της Εικόνα 7.14, οδηγείται η ροή του κώδικα όταν η εφαρμογή εκκινείτε με την onCreate().

```
@Override
public void onStart() {
    super.onStart();
    if(D) Log.e(TAG, "++ ON START ++");

    // If BT is not on, request that it be enabled.
    // setupChat() will then be called during onActivityResult
    if (!mBluetoothAdapter.isEnabled()) {
        Intent enableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(enableIntent, REQUEST_ENABLE_BT);
    // Otherwise, setup the chat session
    } else {
        if (mChatService == null) setupChat();
    }
}

@Override
public synchronized void onResume() {
    super.onResume();
    if(D) Log.e(TAG, "+ ON RESUME +");

    // Performing this check in onResume() covers the case in which BT was
    // not enabled during onStart(), so we were paused to enable it...
    // onResume() will be called when ACTION_REQUEST_ENABLE activity returns.
    if (mChatService != null) {
        // Only if the state is STATE_NONE, do we know that we haven't started already
        if (mChatService.getState() == BluetoothChatService.STATE_NONE) {
          // Start the Bluetooth chat services
          mChatService.start();
        }
    }
}
```

4

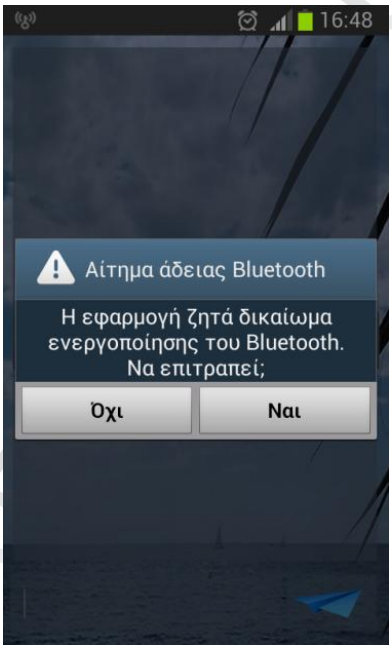
5

Εικόνα 7.14 Η κλάση BluetoothChat (σημεία 4 και 5)

Η μέθοδος isEnabled() ελέγχει αν η υπηρεσία του Bluetooth είναι ενεργοποιημένη. Εάν αυτή η μέθοδος επιστρέψει false, τότε η υπηρεσία είναι απενεργοποιημένη. Αυτό σημαίνει ότι υποστηρίζεται μεν από τη συσκευή αλλά δεν είναι ενεργοποιημένο. Η συνθήκη της if με την isEnable ελέγχει αν ο mBluetoothAdapter δεν είναι ενεργοποιημένος, τότε αν δεν είναι διαθέσιμος δημιουργείται μια πρόθεση intent. Ένα αντικείμενο intent ενθυλακώνει μια αίτηση εργασίας που χρησιμοποιείται από το λειτουργικό σύστημα Android. Στη συγκεκριμένη περίπτωση δημιουργείται το αντικείμενο πρόθεσης enableIntent για την αποστολή μιας έκκλησης ενεργοποίησης

του BT προς το χρήστη. Για να αιτηθεί η ενεργοποίηση της τεχνολογίας Bluetooth, καλείται η μέθοδος startActivityForResult() με την ACTION_REQUEST_ENABLE. Τότε θα σταλεί ένα αίτημα ενεργοποίησης της σύνδεσης Bluetooth μέσα από τις ρυθμίσεις του συστήματος (χωρίς να διακοπεί η λειτουργία της εφαρμογής). Εάν το BT είναι διαθέσιμο τότε προχωράει σε έναν επόμενο έλεγχο για την ενεργοποίηση του chat με την setupChat.

Θα εμφανιστεί ένα παράθυρο διάλογου ζητώντας την άδεια του χρήστη για να ενεργοποιηθεί η σύνδεση Bluetooth, όπως φαίνεται παρακάτω στην Εικόνα 7.15. Αν ο χρήστης επιλέξει "Ναι", το σύστημα θα αρχίσει την ενεργοποίηση της σύνδεσης Bluetooth και η ροή του προγράμματος θα επιστρέψει στην εφαρμογή, είτε η διαδικασία της ενεργοποίησης του Bluetooth ολοκληρωθεί είτε όχι.



Εικόνα 7.15 Αίτημα άδειας ενεργοποίησης Bluetooth

Σε περίπτωση που το BT δεν βρέθηκε διαθέσιμο κατά την εκτέλεση της onStart() τότε αναλαμβάνει η onResume()τη διάθεση του (Εικόνα 7.14, σημείο 5). Η onResume() θα κληθεί μετά την επιστροφή της ACTION_REQUEST_ENABLE. Ελέγχει αν η μεταβλητή mChatService δεν είναι null και μόνο όταν έχει λάβει η mChatService.getState() την κατάσταση STATE_NONE ξεκινά την υπηρεσία του mChatService.start().

Η σταθερά REQUEST_ENABLE_BT που μεταφέρεται μέσω της startActivityForResult() είναι ένας τοπικά ορισμένος ακέραιος, μεγαλύτερος του 0,

τον οποίο το σύστημα τον επιστρέφει μέσω της onActivityResult() στην παράμετρο requestCode.

```
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    if(D) Log.d(TAG, "onActivityResult " + resultCode);
    switch (requestCode) {
    case REQUEST_CONNECT_DEVICE:
        // When DeviceListActivity returns with a device to connect
        if (resultCode == Activity.RESULT_OK) {
            // Get the device MAC address
            String address = data.getExtras()
                                 .getString(DeviceListActivity.EXTRA_DEVICE_ADDRESS);
            // Get the BLuetoothDevice object
            BluetoothDevice device = mBluetoothAdapter.getRemoteDevice(address);
            // Attempt to connect to the device
            mChatService.connect(device);
        }
        break;
    case REQUEST_ENABLE_BT:
        // When the request to enable Bluetooth returns
        if (resultCode == Activity.RESULT_OK) {
            // Bluetooth is now enabled, so set up a chat session
            setupChat();
        } else {
            // User did not enable Bluetooth or an error occured
            Log.d(TAG, "BT not enabled");
            Toast.makeText(this, R.string.bt_not_enabled_leaving, Toast.LENGTH_SHORT).show();
            finish();
        }
    }
}
```

6

Εικόνα 7.16 Η κλάση BluetoothChat (σημείο 6)

Εάν επιτευχθεί η διαθεσιμότητα του Bluetooth τότε η δραστηριότητα λαμβάνει την τιμή του αποτελέσματος RESULT_OK στην onActivityResult() (Εικόνα 7.16, σημείο 6).

Εικόνα 7.17 Η υπηρεσία του Bluetooth δεν είναι διαθέσιμη

Εάν το Bluetooth δεν είναι διαθέσιμο είτε λόγω κάποιου λάθους είτε γιατί ο χρήστης απάντησε "Όχι" στην ερώτηση της σύνδεσης με την υπηρεσία του BT, τότε ο κώδικας επιστρέφει τη RESULT_CANCELED.

Ενεργοποιώντας την ανίχνευση των συσκευών αυτόματα θα ενεργοποιηθεί και το Bluetooth. [44]

7.4.2.1.3 Ο κώδικας του αρχείου πόρων της Main.xml

Το αρχείο main.xml απεικονίζει τη διεπαφή του χρήστη για την ανταλλαγή των μηνυμάτων (Εικόνα 7.19). Όπως αναφέρθηκε στις προηγούμενες παραγράφους η σύνδεση του κώδικά της κλάσης BluetoothChat με το αρχείο main.xml το οποίο είναι μια πηγή του σχεδιαστικού κώδικα πραγματοποιείτε με τη χρήση της setContentView στη μέθοδο onCreate της BluetoothChat.

Στην Εικόνα 7.18 παρατίθεται ο κώδικας του αρχείου main.xml. Στον κώδικα αυτόν ο μηχανισμός διάταξης των στοιχείων, LinearLayout, αναλαμβάνει την γραμμική τοποθέτηση των πόρων του αρχείου στο γραφικό περιβάλλον της διεπαφής της Main. Ορίζεται ο κάθετος προσανατολισμός, orientation, των στοιχείων της οθόνης με την orientation="vertical" καθώς και το ύψος layout_height και το πλάτος layout_width που θα καταλαμβάνει η διεπαφή στην οθόνη. Ο προσανατολισμός της οθόνης μπορεί να είναι είτε οριζόντιος είτε κάθετος, που σημαίνει ότι μεταβάλλεται ανάλογα με τις διαστάσεις της οθόνης είτε ως προς το πλάτος είτε ως προς το ύψος, αντίστοιχα. Ο προσανατολισμός της οθόνης ρυθμίζει την εμφάνιση της διεπαφής σε διαφορετικές συσκευές που λειτουργούν σε διαφορετικές κατευθύνσεις από προεπιλογή, και την εμφάνιση της διεπαφής στην ίδια την συσκευή όταν ο προσανατολισμός μπορεί να αλλάξει κατά το χρόνο εκτέλεσης της εφαρμογής, όταν ο χρήστης περιστρέφει τη συσκευή. Η εικόνα back που βρίσκεται μέσα στον φάκελο drawable εμφανίζεται με την background="@drawable/but" στο backround της οθόνης.

Στο επάνω μέρος της οθόνης εμφανίζεται η λίστα των μηνυμάτων (Εικόνα 7.18) η οποία σχηματίζεται με το μηχανισμό ελέγχου της ListView. Όμοια με την διεπαφή ορίζονται στη ListView το ύψος, το πλάτος και το backround της λίστας. Με την

stackFromBottom η εμφάνιση των στοιχείων της λίστας θα ξεκινά με την τοποθέτηση των νεότερων στοιχείων από το κάτω μέρος της οθόνης προς τα πάνω.

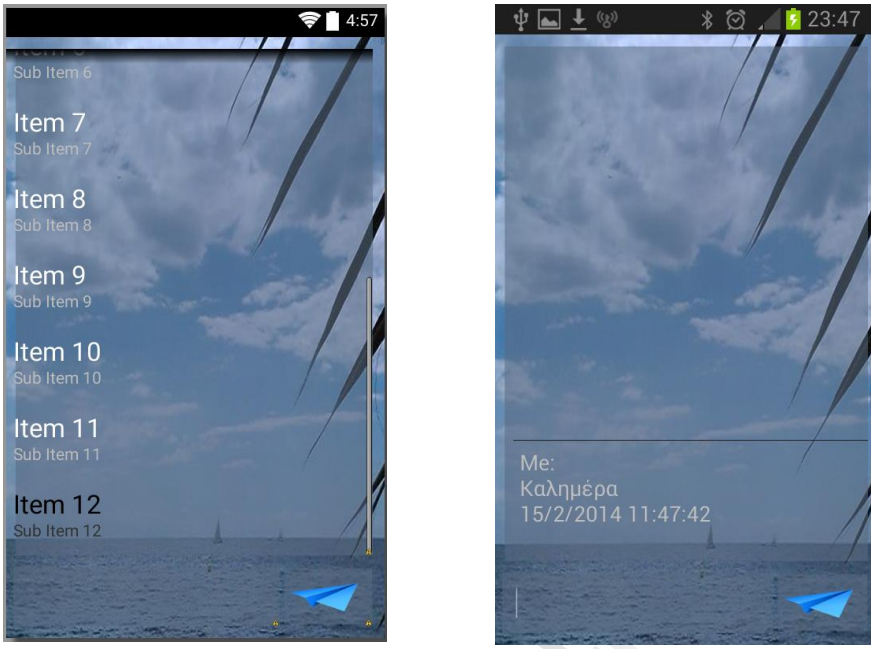
main.xml

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/back" >
    <ListView android:id="@+id/in"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:stackFromBottom="true"
        android:background="@drawable/but"

        android:layout_marginRight="10dp"
        android:layout_marginTop="10dp"
        android:layout_marginBottom="10dp"
        android:transcriptMode="alwaysScroll"
        android:layout_weight="1"
    />
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginRight="10dp"

        android:layout_marginBottom="10dp"
        >
        <EditText
             android:id="@+id/edit_text_out"
            android:layout_width="0dp"
            android:layout_height="50dp"
            android:layout_weight="1"
            android:background="@drawable/but"
            android:textColor="#ffffff"

            android:layout_gravity="bottom"
        />
        <Button android:id="@+id/button_send"
            android:layout_width="75dp"
            android:layout_height="50dp"
            android:layout_marginLeft="3dp"
             android:background="@drawable/but1"
        />
    </LinearLayout>
</LinearLayout>
```

Εικόνα 7.18 Ο κώδικας του αρχείου main.xml

Τα περιθώρια της λίστας από το πάνω, το κάτω και το δεξί μέρος της οθόνης δηλώνονται με τη χρήση των layout_marginRight, layout_marginTop, και layout_marginBottom αντίστοιχα. Η μονάδα dp που χρησιμοποιείται είναι μία εικονική μονάδα pixel που σχετίζεται με την πυκνότητα των pixels και χρησιμοποιείται για τον καθορισμό των διαστάσεων ή της θέσης των στοιχείων κατά το σχεδιασμό των διεπαφών.

Η ενεργοποίηση της μπάρας κύλισης για την εμφάνιση των στοιχεία της λίστας γίνεται με την transcriptMode="alwaysScroll".



Εικόνα 7.19 Η γραφική απεικόνιση αρχείου main.xml α) όπως παράγεται από eclipse και β) όπως εμφανίζεται στη συσκευή του χρήστη

Στο κάτω μέρος της Εικόνα 7.19 διαφαίνεται το πλαίσιο εισαγωγής του κειμένου (EditText) και το κουμπί (Button) το οποίο δίνει την εντολή της αποστολής του μηνύματος. Η σχεδίαση αυτών των δύο στοιχείων γίνεται με το μηχανισμό της γραμμικής διάταξης LinearLayout, και ορίζεται ο οριζόντιος προσανατολισμός τους με την orientation="horizontal". Όμοια με τα προηγούμενα ορίζονται και εδώ το πλάτος, το ύψος και τα περιθώρια των στοιχείων μέσα στην οθόνη της διεπαφής. Για το σχεδιασμό του πλαισίου του κειμένου EditText δηλώνεται ο νέος πόρος edit_text_out με την id="@+id/edit_text_out" ο οποίος θα χρησιμοποιηθεί ως μεταβλητή από το σύστημα του Android για την καταχώρηση του μηνύματος. Ο πόρος αυτός είναι ένας νέος πόρος ο οποίος δεν είναι δηλωμένος στην κλάση R.java (παράγραφος 7.3.3). Ο πόρος του θα πρέπει να δημιουργηθεί από το σύστημα και να δηλωθεί στο αρχείο R.java (**public static final int** *edit_text_out*=0x7f06000b;) και για το λόγο αυτό χρησιμοποιείται το σύμβολο + κατά τη δήλωση του. Στη συνέχεια δηλώνονται το background="@drawable/but", το χρώμα του κειμένου textColor="#ffffff" και η θέση του πλαισίου layout_gravity="bottom".

Ο σχεδιασμός του κουμπιού Button είναι παρόμοιος με τον σχεδιασμό του πλαισίου. Το στοιχείο του κουμπιού δημιουργείται και δηλώνεται στο αρχείο των πόρων R.java (public static final int edit_text_out=0x7f06000b;) με την id="@+id/button_send". Οι διαστάσεις του πλάτους και του ύψους του κουμπιού

ορίζονται με τις layout_width="75dp" και layout_height="50dp" αντίστοιχα. Το αριστερό περιθώριο του κουμπιού με την διεπαφή ορίζεται με την layout_marginLeft="3dp" και το backround ορίζεται με την background="@drawable/but1".

7.4.2.1.4 Η ανιχνευσιμότητα της συσκευής

Προκειμένου μια συσκευή να είναι ανιχνεύσιμη (enabling discoverability) από τις άλλες συσκευές καλείται η μέθοδος startActivityForResult(Intent, int) με την πρόθεση ACTION_REQUEST_DISCOVERABLE. Αφού κληθεί η μέθοδος startActivityForResult θα εμφανιστεί ένα αίτημα διάθεσης της ανιχνευσιμότητας της συσκευής μέσω των ρυθμίσεων του συστήματος (χωρίς να διακοπεί η λειτουργία της εφαρμογής). Η συσκευή από προεπιλογή θα είναι ανιχνεύσιμη για το χρονικό διάστημα 120 δευτερολέπτων. Η αλλαγή της χρονικής διάρκειας της ανιχνευσιμότητας της συσκευής επιτυγχάνεται με την χρήση της EXTRA_DISCOVERABLE_DURATION. Η μέγιστη διάρκεια της ανιχνευσιμότητας της συσκευής που μπορεί να οριστεί σε μία εφαρμογή είναι 3600 δευτερόλεπτα και για την τιμή της διάρκειας ίση με 0 η συσκευή είναι πάντα ανιχνεύσιμη. Αν δοθεί στη διάρκεια οποιαδήποτε άλλη τιμή κάτω από το 0 ή πάνω από το 3600 τότε η διάρκεια της ανιχνευσιμότητας της συσκευής ρυθμίζεται αυτόματα στα 120 δευτερόλεπτα. Για παράδειγμα, το παρακάτω τμήμα του κώδικα ορίζει τη διάρκεια αυτή στα 300' (Εικόνα 7.20, σημείο 7).

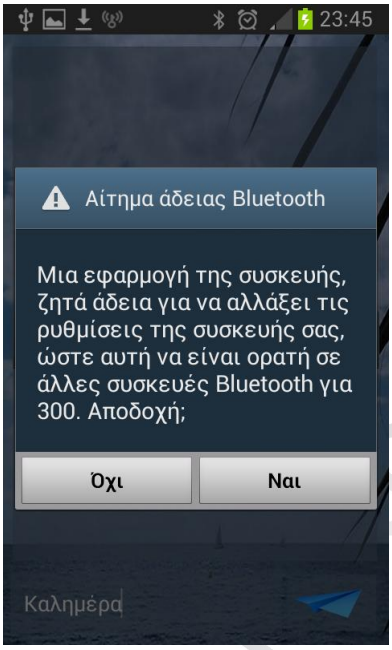
```
private void ensureDiscoverable() {
    if(D) Log.d(TAG, "ensure discoverable");
    if (mBluetoothAdapter.getScanMode() !=
        BluetoothAdapter.SCAN_MODE_CONNECTABLE_DISCOVERABLE) {
        Intent discoverableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
        discoverableIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION, 300);
        startActivity(discoverableIntent);
    }
}
```

7

Εικόνα 7.20 Η μέθοδος ensureDiscoverable της BluetoothChat (σημείο 7)

Στην Εικόνα 7.21 φαίνεται το παράθυρο διαλόγου που ζητά από το χρήστη την άδεια για να γίνει ανιχνεύσιμη η συσκευή. Αν ο χρήστης απαντήσει "ναι", τότε η συσκευή θα είναι ανιχνεύσιμη για το χρονικό διάστημα που ορίζεται από την εφαρμογή. Η κλήση της onActivityResult() επιστρέφει στη δραστηριότητά μια τιμή στην οποία είναι καταχωρημένη η διάρκεια που είναι ανιχνεύσιμη η συσκευή. Εάν ο

χρήστης απαντήσει "Όχι" στο παραπάνω αίτημα ή αν συμβεί κάποιο σφάλμα, τότε το αποτέλεσμα RESULT_CANCELED θα επιστραφεί στη δραστηριότητα.



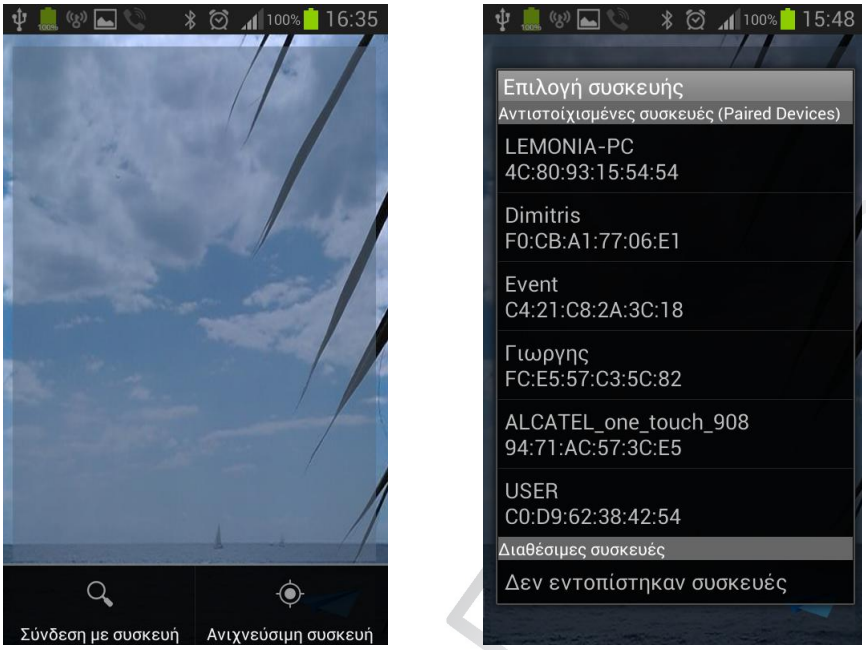
Εικόνα 7.21 Το αίτημα άδειας για την ανιχνευσιμοτήτα της συσκευής

Σημείωση: Αν η υπηρεσία του Bluetooth δεν έχει ενεργοποιηθεί στη συσκευή, η διαδικασία της ανιχνεύσιμης συσκευής θα το ενεργοποιήσει αυτόματα.

Η συσκευή θα παραμείνει σιωπηλά σε κατάσταση ανίχνευσης για καθορισμένο χρονικό διάστημα. Για να ειδοποιηθεί η συσκευή όταν αλλάξει η κατάσταση ανίχνευσης, πρέπει να γίνει στο BroadcastReceiver ενημέρωση με την πρόθεση ACTION_SCAN_MODE_CHANGED. Η ενημέρωση αυτή του BroadcastReceiver γίνεται από τα πρόσθετα πεδία EXTRA_SCAN_MODE και EXTRA_PREVIOUS_SCAN_MODE. Στα πεδία αυτά είναι καταχωρημένες αντίστοιχα η νέα και η παλιά κατάσταση λειτουργίας και οι τιμές που μπορούν να λάβουν είναι η SCAN_MODE_CONNECTABLE_DISCOVERABLE όταν η συσκευή είναι ανιχνεύσιμη, η SCAN_MODE_CONNECTABLE όταν η συσκευή είναι σε μη ανιχνεύσιμη κατάσταση, αλλά εξακολουθεί να είναι σε θέση να λαμβάνει συνδέσεις, ή η τιμή SCAN_MODE_NONE η οποία δείχνει ότι η συσκευή είναι σε μη ανιχνεύσιμη κατάσταση και δεν μπορεί να λαμβάνει συνδέσεις.

Η διαθεσιμότητα της ανίχνευσης μιας συσκευής είναι απαραίτητη μόνο όταν γίνει αίτησή για να φιλοξενηθεί ένα server socket το οποίο δέχεται εισερχόμενες

συνδέσεις, επειδή οι απομακρυσμένες συσκευές πρέπει να μπορούν να ανιχνεύσουν τη συσκευή πριν ξεκινήσει η σύνδεση.



Εικόνα 7.22 α) Το option menu, β) Επιλογή συσκευής για σύνδεση

Το option menu επιτρέπει στο χρήστη να καταστήσει τη συσκευή ανιχνεύσιμη και να επιλέξει τη σύνδεση της με άλλες συσκευές. Τα μενού από το σύστημα Android θεωρούνται ως πόροι και για το λόγο αυτό δηλώνονται στο αρχείο R.java των πόρων.

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.option_menu, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
    case R.id.scan:
        // Launch the DeviceListActivity to see devices and do scan
        Intent serverIntent = new Intent(this, DeviceListActivity.class);
        startActivityForResult(serverIntent, REQUEST_CONNECT_DEVICE);
        return true;
    case R.id.discoverable:
        // Ensure this device is discoverable by others
        ensureDiscoverable();
        return true;
    }
    return false;
}
```

8

Εικόνα 7.23 Το option menu της κλάσης BluetoothChat (σημείο 8)

Ο κώδικας του option menu παρουσιάζεται στην Εικόνα 7.23. Η δημιουργία του option menu γίνεται σημείο 8 της παραπάνω εικόνας με το αντικείμενο inflater που

είναι τύπου MenuInflater το οποίο δημιουργείται μέσα στην συνάρτηση onCreateOptionsMenu(Menu menu) της κλάσης BluetoothChat. [45], [46]

7.4.2.1.5 Η μετάδοση των μηνυμάτων

Η ανταλλαγή και η διαχείριση των μηνυμάτων γίνεται αντίστοιχα από τις μεθόδους setupChat, sendMessage και Handler της κλάσης BluetoothChat.

7.4.2.1.5.1 Η μέθοδος setupChat της κλάσης BluetoothChat

Η μέθοδος setupChat (Εικόνα 7.24, σημείο 9) αναλαμβάνει την αρχικοποίηση των στοιχείων που εμπλέκονται στην ανταλλαγή των μηνυμάτων. Δημιουργεί το αντικείμενο mConversationArrayAdapter τύπου arrayAdapter το οποίο χρησιμοποιεί το αρχείο πόρου message.xml που περιέχει το σχεδιαστικό κομμάτι του κώδικα και εισάγει τη λίστα mConversationView των μηνυμάτων που ανταλλάχθηκαν κατά τη διάρκεια της συνομιλίας.

```
private void setupChat() {
    Log.d(TAG, "setupChat()");

    // Initialize the array adapter for the conversation thread
    mConversationArrayAdapter = new ArrayAdapter<String>(this, R.layout.message);
    mConversationView = (ListView) findViewById(R.id.in);
    mConversationView.setAdapter(mConversationArrayAdapter);

    // Initialize the compose field with a listener for the return key
    mOutEditText = (EditText) findViewById(R.id.edit_text_out);
    mOutEditText.setOnEditorActionListener(mWriteListener);

    // Initialize the send button with a listener that for click events
    mSendButton = (Button) findViewById(R.id.button_send);
    mSendButton.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            // Send a message using content of the edit text widget
            TextView view = (TextView) findViewById(R.id.edit_text_out);
            String message = view.getText().toString();
            sendMessage(message);
        }
    });

    // Initialize the BluetoothChatService to perform bluetooth connections
    mChatService = new BluetoothChatService(this, mHandler);

    // Initialize the buffer for outgoing messages
    mOutStringBuffer = new StringBuffer("");
}
```

9

Εικόνα 7.24 Η μέθοδος setupChat της κλάσης BluetoothChat (σημείο 9)

Στη συνέχεια αρχικοποιεί το πεδίο mOutEditText τύπου editText το οποίο χρησιμοποιείται για την αποθήκευση του μηνύματος προς αποστολή και καλεί την setOnEditorActionListener με την παράμετρο το mWriteListener. Η setOnEditorActionListener ενεργοποιεί έναν ειδικό listener ο οποίος καλείται όταν εγγραφεί κάτι στο πλαίσιο κειμένου και πατηθεί το κουμπί της αποστολής στην οθόνη. Έπειτα δηλώνεται και αρχικοποιείται το κουμπί για την αποστολή του μηνύματος και γίνεται η σύνδεση του με την setOnClickListener η οποία με τη σειρά της καλεί την sendMessage για να αναλάβει την αποστολή του μηνύματος.

Για την σύνδεση με το BT και την αποστολή του μηνύματος δημιουργείται και αρχικοποιείται το αντικείμενο mChatService που είναι τύπου BluetoothChatService. Στο τέλος της μεθόδου setupChat καθαρίζεται ο buffer mOutStringBuffer που χρησιμοποιείται για την προσωρινή καταχώρηση των εξερχόμενων μηνυμάτων.

7.4.2.1.5.2 Ο κώδικας του αρχείου πόρου της Message.xml

Το αρχείο Message.xml είναι μια πηγή. Είναι ένας βοηθητικός πόρος ο οποίος χρησιμοποιείται από την μέθοδο setupChat της κλάσης BluetoothChat, για την αποστολή του μηνύματος.

message.xml

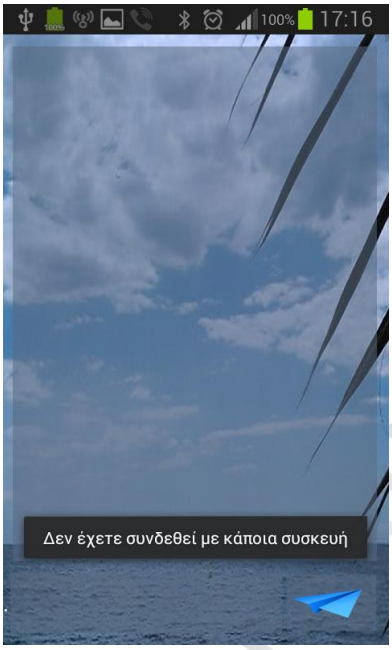
```
<?xml version="1.0" encoding="utf-8"?>

<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="18sp"
    android:padding="5dp"
/>
```

Εικόνα 7.25 Ο κώδικας της Message.xml

7.4.2.1.5.3 Η μέθοδος sendMessage της κλάσης BluetoothChat

Η μέθοδος sendMessage ελέγχει στο σημείο 10 της Εικόνα 7.27 με την συνθήκη ελέγχου if την ύπαρξη ή όχι σύνδεσης με την υπηρεσία του BT. Εάν από τον παραπάνω έλεγχο διαπιστωθεί ότι η mChatService δεν έχει λάβει την κατάσταση STATE_CONNECTED από την BluetoothChatService τότε εμφανίζεται στην οθόνη

όπως φαίνεται στην παρακάτω εικόνα ένα tost μήνυμα «Δεν έχετε συνδεθεί με κάποια συσκευή».



Εικόνα 7.26 Ενημέρωση μη ύπαρξης σύνδεσης

Ο χρήστης ενημερώνεται ότι η συσκευή δεν είναι συνδεδεμένη με κάποια άλλη συσκευή για να μπορέσει να γίνει η αποστολή του μηνύματος.

```
private void sendMessage(String message) {                                    10
    // Check that we're actually connected before trying anything
    if (mChatService.getState() != BluetoothChatService.STATE_CONNECTED) {
        Toast.makeText(this, R.string.not_connected, Toast.LENGTH_SHORT).show();
        return;
    }

    // Check that there's actually something to send
    if (message.length() > 0) {
        // Get the message bytes and tell the BluetoothChatService to write
        byte[] send = message.getBytes();
        mChatService.write(send);

        // Reset out string buffer to zero and clear the edit text field
        mOutStringBuffer.setLength(0);
        mOutEditText.setText(mOutStringBuffer);
    }
}

// The action listener for the EditText widget, to listen for the return key  11
private TextView.OnEditorActionListener mWriteListener =
    new TextView.OnEditorActionListener() {
    public boolean onEditorAction(TextView view, int actionId, KeyEvent event) {
        // If the action is a key-up event on the return key, send the message
        if (actionId == EditorInfo.IME_NULL && event.getAction() == KeyEvent.ACTION_UP) {
            String message = view.getText().toString();
            sendMessage(message);
        }
        if(D) Log.i(TAG, "END onEditorAction");
        return true;
    }
};
```

Εικόνα 7.27 Η μέθοδος sendMessage της BluetoothChat (σημεία 10 και 11)

Στην συνέχεια πραγματοποιείται από την επόμενη συνθήκη if ένας ακόμη έλεγχος για το αν υπάρχει κάποιο μήνυμα προς αποστολή. Για το λόγο αυτό υπολογίζεται το μήκος του μηνύματος από την συνάρτηση length. Στην περίπτωση που η length επιστρέψει τιμή μεγαλύτερη από το 0 γίνεται η εγγραφή του μηνύματος στη διεπαφή UI αφού προηγουμένως το περιεχόμενο του μηνύματος message έχει μετατραπεί σε bytes. Τέλος διαγράφονται τα περιεχόμενα του buffer mOutStringBuffer και καθαρίζεται το πεδίο της mOutEditText. Μετά τη μέθοδο sendMessage ακολουθεί ο ακροατής δράσης (listener) OnEditorActionListener του widget EditText (Εικόνα 7.27, σημείο 11) ο οποίος περιμένει την απόκριση από την αποστολή του μηνύματος.

7.4.2.1.5.4 Η κλάση Handler της BluetoothChat

Η κλάση Handler είναι αυτή που διαχειρίζεται τα μηνύματα που ανταλλάσσονται από τους χρήστες για την πραγματοποίηση των συνομιλιών και από την εφαρμογή για την ενημέρωση του χρήστη. Ο έλεγχος των μηνυμάτων πραγματοποιείται από τη συνθήκη ελέγχου switch η οποία δέχεται ως παράμετρο τη μεταβλητή msg τύπου Message. Στην Εικόνα 7.28 στο σημείο 12 η μεταβλητή msg φορτώνει στην what, έναν κωδικό μηνύματος ο οποίος ορίζεται από το χρήστη για να μπορεί ο παραλήπτης μπορεί να αναγνωρίσει τι περιέχει το μήνυμα. Ο κάθε Handler μπορεί να ορίσει τους δικούς του κωδικούς για να μην σύγχυση των κωδικών του με τους κωδικούς των άλλων handlers.

Στον πρώτο έλεγχο της "case MESSAGE_STATE_CHANGE:" εξετάζεται αν η κατάσταση της σύνδεσης έχει μεταβληθεί. Η σταθερά MESSAGE_STATE_CHANGE = 1 και συγκρίνεται η τιμή της με την τιμή που έχει λάβει η msg.what. Σε μία δεύτερη συνθήκη ελέγχου "switch (msg.arg1)" αποδίδεται στη msg η τιμή arg1 που είναι ένας μικρός ακέραιος. Αν αυτή η τιμή της msg=arg1 είναι ίση με αυτή της STATE_CONNECTED=3 που έχει επιστραφεί από την BluetoothChatService, δηλαδή αν υπάρχει σύνδεση, τότε καθαρίζεται ο mConversationArrayAdapter και η ροή του προγράμματος μεταφέρεται εκτός της Handler. Όμοια γίνεται ο έλεγχος για την STATE_CONNECTING=2, την STATE_LISTEN=1 και την STATE_NONE=0. Αν ο έλεγχος στις περιπτώσεις των STATE_CONNECTED, STATE_CONNECTING και STATE_NONE είναι αληθής τότε η ροη του προγράμματος με την εντολή break μεταφέρεται εκτός της handler. Ο

έλεγχος της STATE_LISTEN είναι αληθής όταν λάβει τιμή ίση με 1 στην περίπτωση αυτή δεν υπάρχει break για να οδηγήσει την ροή του κώδικα έξω από τον έλεγχο τότε το σύστημα περιμένει μέχρι να ολοκληρωθεί η κατάσταση listen.

```
// The Handler that gets information back from the BluetoothChatService
@SuppressLint("HandlerLeak")
private final Handler mHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        switch (msg.what) {
        case MESSAGE_STATE_CHANGE:
            if(D) Log.i(TAG, "MESSAGE_STATE_CHANGE: " + msg.arg1);
            switch (msg.arg1) {
            case BluetoothChatService.STATE_CONNECTED:
                mConversationArrayAdapter.clear();
                break;
            case BluetoothChatService.STATE_CONNECTING:
                break;
            case BluetoothChatService.STATE_LISTEN:
            case BluetoothChatService.STATE_NONE:
                break;
            }
            break;
        case MESSAGE_WRITE:
            byte[] writeBuf = (byte[]) msg.obj;
            Calendar c = Calendar.getInstance();
            int sec = c.get(Calendar.SECOND);
            int min = c.get(Calendar.MINUTE);
            int hor = c.get(Calendar.HOUR);

            int xrono = c.get(Calendar.YEAR);
            int mina = c.get(Calendar.MONTH) + 1;
            int mera = c.get(Calendar.DATE);
            // construct a string from the buffer
            String writeMessage = new String(writeBuf);
            mConversationArrayAdapter.add("Me:  "+"\n" + writeMessage + "\n" + mera +"/"+mina+"/"
                                          +xrono+" "+hor+":"+min+":"+sec+"\n");
            dbChat.addContact(new ChatContact(mConnectedDeviceName, mera +"/"+mina+"/"
                                          +xrono, hor+":"+min+":"+sec, "Me:  \n" + writeMessage));
            break;
        case MESSAGE_READ:
            byte[] readBuf = (byte[]) msg.obj;
            Calendar c1 = Calendar.getInstance();
            int sec1 = c1.get(Calendar.SECOND);
            int min1 = c1.get(Calendar.MINUTE);
            int hor1 = c1.get(Calendar.HOUR);

            int xrono1 = c1.get(Calendar.YEAR);
            int mina1 = c1.get(Calendar.MONTH) + 1;
            int mera1 = c1.get(Calendar.DATE);
            // construct a string from the valid bytes in the buffer
            String readMessage = new String(readBuf, 0, msg.arg1);
            mConversationArrayAdapter.add(mConnectedDeviceName+":  "+"\n" + readMessage+ "\n" + mera1 +"/"+mina1+"/"
                                                                  +xrono1+" "+hor1+":"+min1+":"+sec1+"\n");
            dbChat.addContact(new ChatContact(mConnectedDeviceName,  mera1 +"/"+mina1+"/"+xrono1, hor1+":"+min1+":"
                                                                  +sec1, mConnectedDeviceName+": "+"\n"+ readMessage));
            break;
        case MESSAGE_DEVICE_NAME:
            // save the connected device's name
            mConnectedDeviceName = msg.getData().getString(DEVICE_NAME);
            Toast.makeText(getApplicationContext(), "Connected to "
                           + mConnectedDeviceName, Toast.LENGTH_SHORT).show();
            db.addConver(new Conversation(mConnectedDeviceName));
            break;
        case MESSAGE_TOAST:
            Toast.makeText(getApplicationContext(), msg.getData().getString(TOAST),
                           Toast.LENGTH_SHORT).show();
            break;
        }
    }
```

Εικόνα 7.28 Η μέθοδος handler της BluetoothChat (σημεία 12, 13, 14 και 15)

Στον επόμενο έλεγχο (case MESSAGE_WRITE:) στο σημείο 13 της Εικόνα 7.28 εξετάζεται αν έχει εγγραφεί κάποιο μήνυμα τότε η MESSAGE_WRITE = 3 και συγκρίνεται με την τιμή που έχει λάβει η msg.what. Εάν η τιμή της msg.what = 3 τότε ο πίνακας από byte writeBuf περιέχει το μήνυμα msg.obj το οποίο γράφτηκε από τον

χρήστη για να αποσταλεί στον παραλήπτη. Οι χρονικές παράμετροι του μηνύματος ανακτώνται από το σύστημα. Το περιεχόμενο του μηνύματος αποθηκεύεται με τη μορφή string στην μεταβλητή writeMessage. Το περιεχόμενο της μεταβλητής writeMessage συμπληρώνεται με τις χρονικές παραμέτρους και εμφανίζεται στη λίστα της mConversationArrayAdapter στην οθόνη. Η εγγραφή του μηνύματος συμπληρωμένη με τις χρονικές παραμέτρους καταχωρείται στη ΒΔ dbChat και η ροη του προγράμματος μεταφέρεται εκτός της handler.

Η περίπτωση της λήψης ενός εισερχόμενου μηνύματος (case MESSAGE_READ:) εξετάζεται στο σημείο 14 της Εικόνα 7.28 και η MESSAGE_READ = 2. Όμοια με τα προηγούμενα συγκρίνεται με την τιμή που έχει λάβει η msg.what. Εάν και η τιμή της msg.what = 2 αυτό σημαίνει ότι ο πίνακας readBuf που αποθηκεύει byte περιέχει το μήνυμα msg.obj (το obj χρησιμοποιείται αντί της setData) το οποίο στάλθηκε από τον παραλήπτη. Οι χρονικές παράμετροι του μηνύματος ανακτώνται από το σύστημα. Το όνομα του αποστολέα βρίσκεται στην μεταβλητή τύπου string mConnectedDeviceName. Το περιεχόμενο του μηνύματος αποθηκεύεται με τη μορφή string στην μεταβλητή readMessage. Το μηνυμα συμπληρωμένο με τις χρονικές παραμέτρους και το όνομα της συσκευής του αποστολέα καταχωρείται στη λίστα της mConversationArrayAdapter και εμφανίζεται στην οθόνη, τέλος προστίθεται η εγγραφή στη ΒΔ dbChat και η ροη του προγράμματος μεταφέρεται εκτός της handler.

Στο σημείο 15 της Εικόνα 7.28 ο έλεγχος της case MESSAGE_DEVICE_NAME: γίνεται για την ανάκτηση των ονομάτων των συνδεδεμένων συσκευών. Η MESSAGE_DEVICE_NAME = 4 στην περίπτωση που και η msg.what έχει την ίδια τιμή εμφανίζεται ένα τοστ μήνυμα με το όνομα της συσκευής, η εγγραφή του ονόματος της συσκευής προστίθεται στη ΒΔ db και η ροη του προγράμματος μεταφέρεται εκτός της handler. [47]

7.4.2.1.6 Οι μέθοδοι onPause() και onStop() της BluetoothChat

Η ανάλυση της κλάσης BluetoothChat ολοκληρώνεται με την παράθεση του κώδικα των μεθόδων onPause() και onStop().

```
@Override
public synchronized void onPause() {
    super.onPause();
    if(D) Log.e(TAG, "- ON PAUSE -");
}

@Override
public void onStop() {
    super.onStop();
    if(D) Log.e(TAG, "-- ON STOP --");
}


@Override
public void onDestroy() {
    super.onDestroy();
    // Stop the Bluetooth chat services
    if (mChatService != null) mChatService.stop();
    if(D) Log.e(TAG, "--- ON DESTROY ---");
}
```

Εικόνα 7.29 Οι μέθοδοι onPause(), onStop() και onDestroy() της BluetoothChat

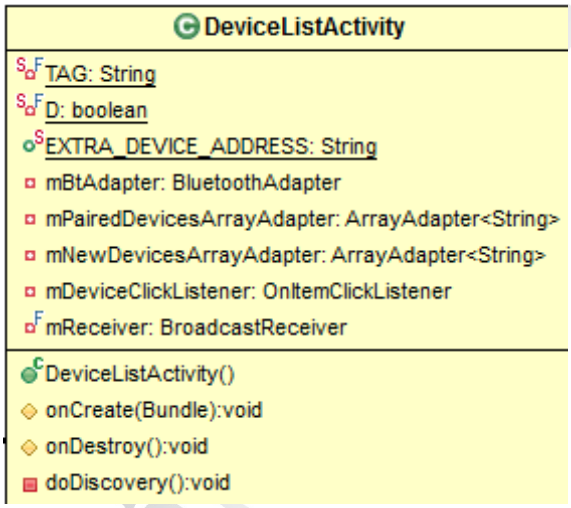
Η μέθοδος onPause() εμφανίζει στο Log το μήνυμα "- ON PAUSE -" όταν η εφαρμογή περιέλθει σε αυτήν την κατάσταση. Παρόμοια οι μέθοδοι onStop() και onDestroy εμφανίζουν στο LOG τα μηνύματα "-- ON STOP --" και "--- ON DESTROY ---" αντίστοιχα όταν η εφαρμογή περιέλθει σε κάποια απ αυτές τις καταστάσεις. Η μέθοδος onDestroy επιπλέον τερματίζει και την επικοινωνία.

7.4.3 Εύρεση – Αναζήτηση ή ανίχνευση των συσκευών

Η διαδικασία της ανίχνευσης των συσκευών είναι μια διαδικασία σάρωσης της περιοχής η οποία βρίσκεται εντός της εμβέλειας της συσκευής, με σκοπό τον εντοπισμό των συσκευών που έχουν ενεργοποιημένη την υπηρεσία του Bluetooth (αυτό μερικές φορές αναφέρεται ως «ανίχνευση», «έρευνα» ή «σάρωση») και στη συνέχεια, ζητά κάποιες πληροφορίες από τις συσκευές που ανιχνεύτηκαν. Η συσκευή που διαθέτει την υπηρεσία του Bluetooth και βρίσκεται εντός της περιοχής ανίχνευσης θα ανταποκριθεί στο αίτημα της ανίχνευσης μόνο αν είναι προς το παρόν ενεργή η ανιχνευσιμότητα της. Αν μια συσκευή είναι ανιχνεύσιμη, θα ανταποκριθεί στο αίτημα ανίχνευσης ανταλλάσοντας κάποιες πληροφορίες, όπως είναι το όνομα της συσκευής, την κλάση της, και τη μοναδική διεύθυνση MAC δηλαδή το μοναδικό αριθμός της κάρτας του δικτύου της. Χρησιμοποιώντας αυτές τις πληροφορίες, η συσκευή που εκτελεί την ανίχνευση μπορεί να εκκινήσει μια σύνδεση με την συσκευή που έχει ανιχνευθεί.

7.4.3.1 Η κλάση DeviceListActivity

Η κλάση DeviceListActivity αναλαμβάνει να εμφανίσει στην οθόνη τις συσκευές που έχουν εντοπιστεί μετά από τη διαδικασία της αναζήτησης αλλά και αυτών των συσκευών (αν υπάρχουν) με τις οποίες είχε συνδεθεί παλιότερα. Δημιουργεί μια διεπαφή (interface) στην οποία εμφανίζονται οι ήδη συνδεδεμένες συσκευές και ένα κουμπί για την αναζήτηση νέων συσκευών. Το σχεδιαστικό κομμάτι της εμφάνισης περιγράφεται αναλυτικότερα σε επόμενες παραγράφους.



Εικόνα 7.30 Η δομή της κλάσης DeviceListActivity

Η δραστηριότητα DeviceListActivity όπως φαίνεται και από την Εικόνα 7.30 υλοποιεί εκτός του κατασκευαστή της DeviceListActivity άλλες τρεις μεθόδους:

- Τη μέθοδο onCreate(Bundle) μέσα στην οποία αρχικοποιείται η activity και η περιγραφή της οποίας ακολουθεί.
- Τη μέθοδο onDestroy(), η οποία τερματίζει την activity όταν ο χρήστης επιθυμεί να την εγκαταλείψει.
- Και τη μέθοδο doDiscovery(), που πραγματοποιεί την αναζήτηση των συσκευών.

Η κλάση DeviceListActivity όπως και η BluetoothChat ξεκινά με τη δήλωση και την αρχικοποίηση των σταθερών που θα χρησιμοποιηθούν στη συνέχεια. Δημιουργεί το αντικείμενο mBtAdapter που είναι τύπου BluetoothAdapter, (Εικόνα 7.31, σημείο 1) το οποίο χρησιμεύει στην εμφάνιση των απομακρυσμένων συσκευών Bluetooth με

τις οποίες είναι εφικτή η σύνδεση. Το αντικείμενο mBtAdapter κάνει χρήση του μοναδικού Adapter ο οποίος είναι κοινός για όλες τις κλάσεις της εφαρμογής και αποτελεί το σημείο εισόδου για όλες τις αλληλεπιδράσεις του Bluetooth. Χρησιμοποιώντας τό, είναι δυνατή η ανίχνευση των άλλων συσκευών που διαθέτουν την υπηρεσία του Bluetooth και η διερεύνηση της λίστας των προηγούμενων συνδεδεμένων (ζευγών) συσκευών. Με τη DeviceListActivity δίνεται υπόσταση σε μια BluetoothDevice χρησιμοποιώντας τη διεύθυνση MAC για τη δημιουργία ενός BluetoothServerSocket.

```
public class DeviceListActivity extends Activity {
    // Debugging
    private static final String TAG = "DeviceListActivity";
    private static final boolean D = true;

    // Return Intent extra
    public static String EXTRA_DEVICE_ADDRESS = "device_address";

    // Member fields
    private BluetoothAdapter mBtAdapter;
    private ArrayAdapter<String> mPairedDevicesArrayAdapter;
    private ArrayAdapter<String> mNewDevicesArrayAdapter;
```

1

Εικόνα 7.31 Η κλάση DeviceListActivity (σημείο 1)

Στη συνέχεια δημιουργούνται δύο πινάκες τύπου ArrayAdapter στους οποίους αποθηκεύονται οι συσκευές που είχαν αντιστοιχιστεί παλιότερα (mPairedDevicesArrayAdapter) με τη συσκευή αλλά και οι νέες συσκευές που προέκυψαν από την διαδικασία της ανίχνευσης (mNewDevicesArrayAdapter).

7.4.3.1.1 Η μέθοδος onCreate(Bundle) της DeviceListActivity

Η μέθοδος onCreate εκτελεί την αρχικοποίηση της activity. Με την κλήση της requestWindowFeature (Εικόνα 7.32, σημείο 2) δημιουργείται η διεπαφή. Μέσω της διεπαφής αυτής που περιέχει τη λίστα των διαθέσιμων συσκευών για σύνδεση πραγματοποιείται η επικοινωνία της εφαρμογής με τον χρήστη.

Η κλάση DeviceListActivity επεκτείνει μια δραστηριότητα (extends Activity). Μια δραστηριότητα (activity) εκτελεί μια ενέργεια που ενδέχεται να ζητήσει ο χρήστης. Κάθε activity αλληλεπιδρά με τον χρήστη, συνεπώς μια Activity class συνυπάρχει με ένα παράθυρο που περιέχει το περιβάλλον αλληλεπίδρασης (User Interface -UI) με τον χρήστη. Η σύνδεση του κώδικα με το γραφικό περιβάλλον αλληλεπίδρασης πραγματοποιείτε με την μέθοδο setContentView(View v). Στη συγκεκριμένη

περίπτωση η σύνδεση του κώδικα της DeviceListActivity αλληλεπιδρά με το γραφικό περιβάλλον της device_list η περιγραφή της οποίας γίνεται παρακάτω στην παράγραφο 7.4.3.1.7 "Η device_list.xml".

Στη μέθοδο onCreate() αρχικοποιείται το κουμπί της ανίχνευσης, scanButton, που καλεί τη doDiscovery(), η οποία αναλύεται παρακάτω, για να ξεκινήσει η ανίχνευση των slave συσκευών (Εικόνα 7.32, σημείο 3). Το κουμπί scanButton συνδέεται με έναν listener και ενεργοποιεί την μέθοδο onClick.

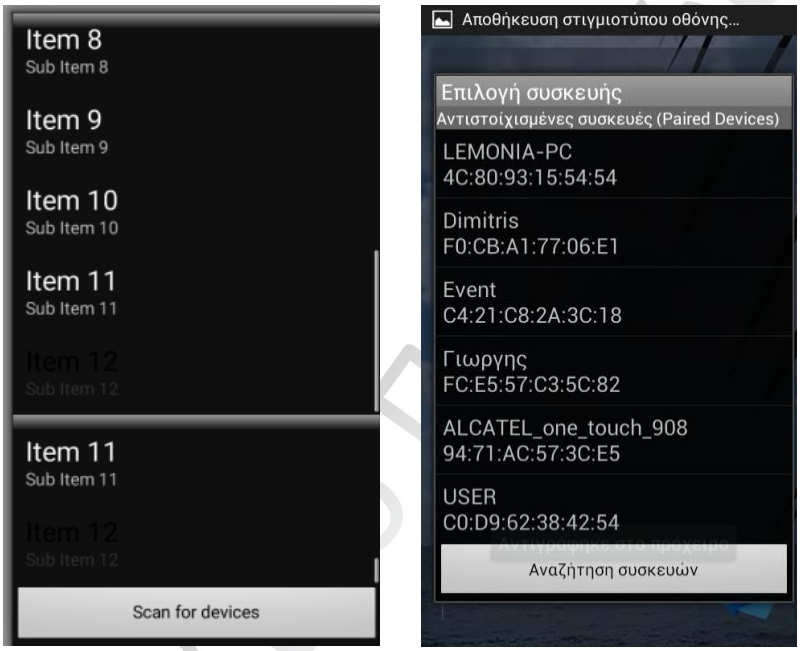
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Setup the window
    requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);
    setContentView(R.layout.device_list);

    // Set result CANCELED incase the user backs out
    setResult(Activity.RESULT_CANCELED);

    // Initialize the button to perform device discovery
    Button scanButton = (Button) findViewById(R.id.button_scan);
    scanButton.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            doDiscovery();
            v.setVisibility(View.GONE);
        }
    });

    // Initialize array adapters. One for already paired devices and
    // one for newly discovered devices
    mPairedDevicesArrayAdapter = new ArrayAdapter<String>(this, R.layout.device_name);
    mNewDevicesArrayAdapter = new ArrayAdapter<String>(this, R.layout.device_name);

    // Find and set up the ListView for paired devices
    ListView pairedListView = (ListView) findViewById(R.id.paired_devices);
    pairedListView.setAdapter(mPairedDevicesArrayAdapter);
    pairedListView.setOnItemClickListener(mDeviceClickListener);

    // Find and set up the ListView for newly discovered devices
    ListView newDevicesListView = (ListView) findViewById(R.id.new_devices);
    newDevicesListView.setAdapter(mNewDevicesArrayAdapter);
    newDevicesListView.setOnItemClickListener(mDeviceClickListener);

    // Register for broadcasts when a device is discovered
    IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
    this.registerReceiver(mReceiver, filter);

    // Register for broadcasts when discovery has finished
    filter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
    this.registerReceiver(mReceiver, filter);
```

2 3 4 5

Εικόνα 7.32 Η μέθοδος onCreate() της DeviceListActivity (σημεία 2-5)

Στο σημείο 4 της Εικόνα 7.32 αρχικοποιούνται τα διανύσματα mPairedDevicesArrayAdapter και mNewDevicesArrayAdapter που χρησιμοποιούνται για την αποθήκευση των συσκευών που έχουν αντιστοιχιστεί από προηγούμενες διαδικασίες ανίχνευσης καθώς και αυτών που θα εντοπιστούν. Με τον τρόπο αυτό

δημιουργείται μία λίστα συσκευών, όπως αυτή που εμφανίζεται στην παρακάτω εικόνα, η οποία αρχικά περιέχει μόνο τις συσκευές που είναι ήδη γνωστές από παλιότερη σύζευξη. Η συσκευή στη φάση αυτή περιμένει την απόκριση του χρήστη. Ένας listener είναι στη διάθεση της κάθε συσκευής ο οποίος μόλις ενεργοποιηθεί με την επιλογή της συσκευής δημιουργεί μια σύνδεση με την συσκευή. Εάν επιλέγει κάποια συσκευή από τη λίστα τότε ένας listener ενεργοποιείται με το onClick event.



Εικόνα 7.33 α) Σχεδιασμός με το eclipse β) Λίστα συσκευών

Το κουμπί «Αναζήτηση συσκευών» ξεκινά τη διαδικασία της ανίχνευσης των νέων συσκευών και την εμφάνιση τους στην οθόνη, στη λίστα των νέων συσκευών.

Στην Εικόνα 7.32 του κώδικα στο σημείο 5 δημιουργείται ένα φίλτρο πάνω στις λίστες ώστε να μπορεί να αναζητηθεί μια συσκευή με το ID της.

7.4.3.1.1.1 Αίτηση της λίστας των αντιστοιχισμένων συσκευών

Πριν από την ανίχνευση των συσκευών, αιτείται η λίστα των αντιστοιχισμένων συσκευών για να διαπιστωθεί αν η συσκευή με την οποία επιθυμούμε να συνδεθούμε είναι ήδη γνωστή από παλιότερη σύνδεση. Για να το γίνει αυτό, καλείται, στο σημείο 6 της Εικόνα 7.34, η getBondedDevices() η οποία θα επιστρέψει το σύνολο των BluetoothDevices που αντιπροσωπεύουν τις συζευγμένες συσκευές.

```
// Get the local Bluetooth adapter
mBtAdapter = BluetoothAdapter.getDefaultAdapter();

// Get a set of currently paired devices
Set<BluetoothDevice> pairedDevices = mBtAdapter.getBondedDevices();

// If there are paired devices, add each one to the ArrayAdapter
if (pairedDevices.size() > 0) {
    findViewById(R.id.title_paired_devices).setVisibility(View.VISIBLE);
    for (BluetoothDevice device : pairedDevices) {
        mPairedDevicesArrayAdapter.add(device.getName() + "\n" + device.getAddress());
    }
} else {
    String noDevices = getResources().getText(R.string.none_paired).toString();
    mPairedDevicesArrayAdapter.add(noDevices);
}
```

6 7 8

Εικόνα 7.34 Αίτηση της λίστας των αντιστοιχισμένων συσκευών

Στη συνθήκη ελέγχου if εξετάζεται αν το μέγεθος του διανύσματος mPairedDevicesArrayAdapter το οποίο διατηρεί τις συσκευές που έχουν παλιότερα αντιστοιχιστεί είναι μεγαλύτερο από το 0, που σημαίνει ότι υπάρχουν συσκευές με τις οποίες συνδέθηκε παλιότερα η συσκευή. Με τη φωλιασμένη συνθήκη for εντοπίζεται κάθε μία από αυτές τις συσκευές και προστίθεται στον πίνακα mPairedDevicesArrayAdapter μία εγγραφή με το όνομα και τη MAC διεύθυνση της καθεμιάς (Εικόνα 7.34, σημείο 7). Το μόνο που χρειάζεται από το αντικείμενο BluetoothDevice, προκειμένου να ξεκινήσει μια σύνδεση είναι η διεύθυνση MAC. Σ' αυτό τον κώδικα, η διεύθυνση MAC αποθηκεύεται σε μια ArrayAdapter λίστας η οποία εμφανίζεται στο χρήστη. Η διεύθυνση MAC μπορεί αργότερα να εξαχθεί για να πραγματοποιηθεί η σύνδεση. Στον παραπάνω κώδικα η προσθήκη του ονόματος και της διεύθυνσης MAC στο διάνυσμα mPairedDevicesArrayAdapter γίνεται με την add και η ανάκτηση του ονόματος και της διεύθυνσης MAC γίνεται από την getName() και την getAddress() αντίστοιχα.

Στην αντίθετη περίπτωση, όταν από τον έλεγχος της if διαπιστωθεί ότι το μέγεθος του διανύσματος mPairedDevicesArrayAdapter είναι μηδενικό (Εικόνα 7.34, σημείο 8) που σημαίνει ότι δεν υπάρχουν εντοπισμένες συσκευές από προηγούμενες ανιχνεύσεις τότε στη λίστα των αντιστοιχισμένων συσκευών εμφανίζεται το μήνυμα «Δεν υπάρχουν συσκευές για αντιστοίχηση». [48]

7.4.3.1.2 Η μέθοδος onDestroy() της κλάσης DeviceListActivity

Η μέθοδος onDestroy() αναλαμβάνει τον τερματισμό - καταστροφή της activity όταν ο χρήστης επιθυμεί να την εγκαταλείψει. Με τη συνθήκη if γίνεται έλεγχος μήπως η διαδικασία ανίχνευσης νέων συσκευών είναι σε εξέλιξη για να την σταματήσει με την cancelDiscovery().

```
@Override
protected void onDestroy() {
    super.onDestroy();

    // Make sure we're not doing discovery anymore
    if (mBtAdapter != null) {
        mBtAdapter.cancelDiscovery();
    }

    // Unregister broadcast listeners
    this.unregisterReceiver(mReceiver);
}
```

Εικόνα 7.35 Η μέθοδος onDestroy() της κλάσης DeviceListActivity

Ενώ τέλος με την unregisterReceiver() αποδεσμεύει τους listeners που χρησιμοποιούνταν από την DeviceListActivity.

7.4.3.1.3 Η ανίχνευση των νέων συσκευών με την doDiscovery

Η επιλογή του κουμπιού «Αναζήτηση συσκευών» από το χρήστη ενεργοποιεί έναν listener ο οποίος καλεί την μέθοδο doDiscovery και ξεκινά η διαδικασία της ανίχνευσης των νέων συσκευών.

Εικόνα 7.36 Η οθόνης «Αναζήτηση συσκευών…»

Στο σημείο 9 του κώδικα της Εικόνα 7.37 γίνεται αλλαγή του τίτλου του UI από «Επιλογή συσκευής» σε «Αναζήτηση συσκευών…». Η μεταβολή της ένδειξης του τίτλου αλλά και το εικονίδιο του κύκλου που περιστρέφεται δίπλα στον τίτλο σημαίνει ότι η διαδικασία της ανίχνευσης βρίσκεται σε εξέλιξη όπως φαίνεται στην Εικόνα 7.36 που ακολουθεί.

Κάτω από τη λίστα των ήδη αντιστοιχισμένων συσκευών θα εμφανιστεί η λίστα με τον τίτλο «Διαθέσιμες συσκευές» στην οποία θα τοποθετηθούν οι νέες συσκευές που θα εντοπιστούν από τη διαδικασία της ανίχνευσης.

Η συνθήκη if ελέγχει αν έχει ήδη πραγματοποιηθεί η διαδικασία της ανίχνευσης τότε ακυρώνει, σταματάει κάθε άλλη διαδικασία ανίχνευσης που επιχειρείται.

```
/**
 * Start device discover with the BluetoothAdapter
 */
private void doDiscovery() {
    if (D) Log.d(TAG, "doDiscovery()");

    // Indicate scanning in the title
    setProgressBarIndeterminateVisibility(true);
    setTitle(R.string.scanning);

    // Turn on sub-title for new devices
    findViewById(R.id.title_new_devices).setVisibility(View.VISIBLE);

    // If we're already discovering, stop it
    if (mBtAdapter.isDiscovering()) {
        mBtAdapter.cancelDiscovery();
    }

    // Request discover from BluetoothAdapter
    mBtAdapter.startDiscovery();
}
```

9

10

Εικόνα 7.37 Η μέθοδος startDiscovery()της κλάσης DeviceListActivity

Για την ανίχνευση των συσκευών (Discovering devices), καλείται η μέθοδος startDiscovery() στο σημείο 10 της Εικόνα 7.37. Ξεκινά η διαδικασία της ανακάλυψης των απομακρυσμένων συσκευών. Η διαδικασία αυτή είναι ασύγχρονη και η μέθοδος θα επιστρέψει αμέσως μια boolean λογική τιμή, που δηλώνει κατά πόσον η ανακάλυψη έχει ξεκινήσει με επιτυχία. Η διαδικασία ανακάλυψης συνήθως περιλαμβάνει ένα αίτημα ανίχνευσης (διάρκειας περίπου 12 δευτερολέπτων), που ακολουθείται από μια λίστα με τα ονόματα των συσκευών που βρέθηκαν κατά την διαδικασία της ανίχνευσης. [49]

7.4.3.1.4 Ο ακροατής δραστηριότητας on-click listener των συσκευών

Στην παράγραφο αυτή παρουσιάζεται ο listener που εξυπηρετεί όλες τις συσκευές που περιέχονται στις λίστες ListViews των συσκευών που παλιότερα είχαν ανιχνευτεί και των νέων συσκευών που εντοπίστηκαν. Με την επιλογή της συσκευής για σύζευξη ταυτόχρονα δημιουργείται ο listener mDeviceClickListener και μπαίνει σε λειτουργία. Καλείται η μέθοδος onItemClick η οποία πρέπει να καλείται όταν ένα στοιχείο πατηθεί από τον χρήστη.

```
// The on-click listener for all devices in the ListViews
private OnItemClickListener mDeviceClickListener = new OnItemClickListener() {
    public void onItemClick(AdapterView<?> av, View v, int arg2, long arg3) {
        // Cancel discovery because it's costly and we're about to connect
        mBtAdapter.cancelDiscovery();

        // Get the device MAC address, which is the last 17 chars in the View
        String info = ((TextView) v).getText().toString();
        String address = info.substring(info.length() - 17);

        // Create the result Intent and include the MAC address
        Intent intent = new Intent();
        intent.putExtra(EXTRA_DEVICE_ADDRESS, address);

        // Set result and finish this Activity
        setResult(Activity.RESULT_OK, intent);
        finish();
    }
};
```

11

12

Εικόνα 7.38 Ο mDeviceClickListener listener εξυπηρετεί όλες τις συσκευές της λίστας

Η εκτέλεση της διαδικασίας ανίχνευσης της συσκευής είναι μια βαριά διαδικασία για τον Bluetooth adapter που καταναλώνει αρκετούς από τους πόρου του. Αφού επιλεγεί μια συσκευή για τη σύνδεση, πριν γίνει η σύνδεση πρέπει να σταματήσει η ανίχνευση των συσκευών με τη χρήση της cancelDiscovery(). Επίσης, στην περίπτωση που έχει γίνει ήδη μια σύνδεση με κάποια από τις συσκευές, τότε εάν επιχειρηθεί η διαδικασίας της ανίχνευσης αυτό μπορεί να επιφέρει σημαντική μείωση του διαθέσιμου bandwidth της σύνδεση, οπότε δεν θα πρέπει να εκτελείτε η διαδικασία της ανίχνευσης, ενώ η συσκευή βρίσκεται ήδη σε σύνδεση.

Αυτό που απαιτείται προκειμένου να ξεκινήσει μια σύνδεση είναι η διεύθυνση MAC της συσκευής που έχει επιλεγεί για τη σύνδεση. Στη συγκεκριμένη περίπτωση αυτή αποθηκεύεται σε έναν πίνακα ArrayAdapter που είναι διαθέσιμος στο χρήστη μέσω της ListViews. Η διεύθυνση MAC (17 χαρακτήρες) εξάγεται προκειμένου

αργότερα να ξεκινήσει η σύνδεση. Στο σημείο 11 της Εικόνα 7.38 ανακτάται από τη μεταβλητή info (τύπου string) η διεύθυνση MAC της κάρτας δικτύου της συσκευής, αποθηκεύεται στην μεταβλητή address και μπορεί πλέον να χρησιμοποιηθεί για τη εγκαθίδρυση της σύνδεσης Bluetooth.

Στο σημείο 12 της Εικόνα 7.38 δημιουργείται η πρόθεση (intent) στην οποία περιέχεται η διεύθυνση MAC και αμέσως μετά τερματίζεται η δραστηριότητα. [48]

7.4.3.1.5 Ο BroadcastReceiver της κλάσης DeviceListActivity

Ο παραλήπτης μηνυμάτων, broadcast receiver, όπως αναφέρεται και στην παράγραφο 4.4 "Οι παραλήπτες μηνυμάτων (broadcast receiver)", είναι το συστατικό στοιχείο που αναλαμβάνει τη λήψη διάφορων ειδοποιήσεων και αντιδρά στην εκπομπή αυτών των ειδοποιήσεων. Οι broadcast receivers δεν έχουν κάποιο γραφικό περιβάλλον. Παρ' όλα αυτά μπορούν να εκκινήσουν ένα activity σαν απάντηση σε κάποια πληροφορία που έλαβαν ή μπορούν να κάνουν χρήση του Notification Manager για να στείλουν κάποια ειδοποίηση στο χρήστη.

Για να ξεκινήσει η διαδικασία της ανίχνευσης των συσκευών καλείται η startDiscovery() μέσα από την μέθοδο doDiscovery() (παράγραφος 7.4.3.1.3). Η διαδικασία αυτή είναι ασύγχρονη και η μέθοδος θα επιστρέψει μία boolean τιμή η οποία δηλώνει ότι η διαδικασία της ανίχνευσης έχει ξεκινήσει επιτυχώς.

Τότε η εφαρμογή ορίζει έναν BroadcastReceiver για την πρόθεση (intent) action προκειμένου να λαμβάνει πληροφορίες για κάθε συσκευή που ανιχνεύει. Για κάθε συσκευή που εντοπίζεται, το σύστημα θα μεταδώσει την πρόθεση (intent) με το πεδίο ACTION_FOUND. Αυτή η πρόθεση επιπλέον μεταφέρει τα πεδία EXTRA_DEVICE και EXTRA_CLASS που περιέχουν το BluetoothDevice και το BluetoothClass αντίστοιχα.

Ένα αντικείμενο BroadcastReceiver ισχύει μόνο κατά τη διάρκεια της κλήσης της μεθόδου onReceive (Context, Intent). Μόλις η ροή του κώδικα επιστρέψει απ' τη λειτουργία της onReceive, το σύστημα θεωρεί ότι το αντικείμενο πρέπει να τερματίσει και να αναστείλει τη λειτουργία του. Για το λόγο αυτό είναι πολύ σημαντικό το τι θα περιλαμβάνει η onReceive(Context, Intent) προς υλοποίηση.

Ο διαχωρισμός των παλιότερα αντιστοιχισμένων συσκευών και των νέων συσκευών γίνεται από την μέθοδος onReceive(). Η μέθοδος onReceive(), στο σημείο 13 της Εικόνα 7.39, περιέχει την μεταβλητή action που είναι τύπου string και η οποία διατηρεί την επιλογή του χρήστη ως προς την ενεργοποίηση του BT. Η μεταβλητή action λειτουργεί σαν σημαία (flag) για τη διενέργεια ή όχι της διαδικασίας της ανίχνευσης των συσκευών. Στη συνθήκη if (Εικόνα 7.39, σημείο 14) εξετάζεται αν κάποια νέα συσκευή έχει εντοπιστεί τότε δημιουργείται το αντικείμενο device τύπου BluetoothDevice μέσα από το οποίο ενεργοποιείται η πρόθεση intent με παράμετρο την EXTRA_DEVICE. Στην περίπτωση που η συσκευή είχε από προηγούμενη φορά αντιστοιχιστεί (paired) τότε παραλείπετε γιατί ήδη βρίσκεται στη λίστα των συσκευών που είχαν αντιστοιχιστεί παλιότερα. Μέσα στην φωλιασμένη συνθήκη έλεγχου if που ακολουθεί (Εικόνα 7.39, σημείο 15) εξετάζεται αν η κατάσταση της συσκευής που επιστρέφεται με την getBondState **δεν** είναι ίση με την BOND_BONDED πράγμα που σημαίνει ότι η συσκευή αυτή δεν είχε αντιστοιχιστεί παλιότερα, τότε το όνομα και η διεύθυνση της συσκευής (τα οποία ανακτώνται αντίστοιχα από την getName και την getAddress) προσθέτονται στη λίστα με τις νέες συσκευές την οποία διαχειρίζεται η mNewDeviceArrayAdapter.

```
private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();

        // When discovery finds a device
        if (BluetoothDevice.ACTION_FOUND.equals(action)) {
            // Get the BluetoothDevice object from the Intent
            BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            // If it's already paired, skip it, because it's been listed already
            if (device.getBondState() != BluetoothDevice.BOND_BONDED) {
                mNewDevicesArrayAdapter.add(device.getName() + "\n" + device.getAddress());
            }
        // When discovery is finished, change the Activity title
        } else if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action)) {
            setProgressBarIndeterminateVisibility(false);
            setTitle(R.string.select_device);
            if (mNewDevicesArrayAdapter.getCount() == 0) {
                String noDevices = getResources().getText(R.string.none_found).toString();
                mNewDevicesArrayAdapter.add(noDevices);
            }
        }
    }
};
```

13 14 15 16 17

Εικόνα 7.39 Ο BroadcastReceiver της κλάσης DeviceListActivity

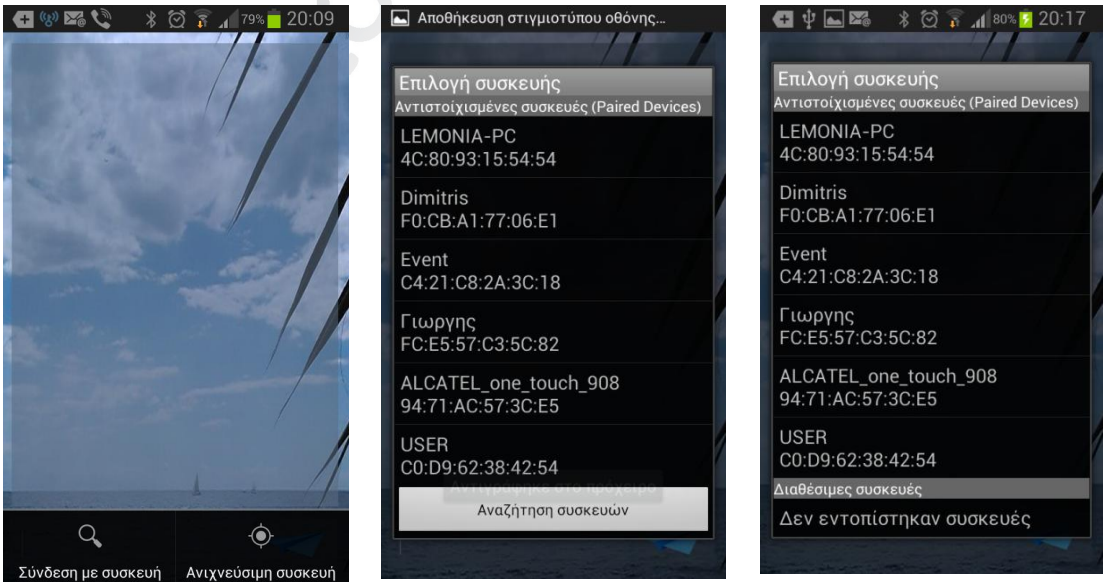
Όταν λήξει το χρονικό διάστημα της ανίχνευσης των συσκευών (12 δευτερόλεπτα περίπου) και ολοκληρωθεί η ανίχνευση, η τιμή της action ισούται με ACTION_DISCOVERY_FINISHED και ο τίτλος της δραστηριότητας παίρνει την τιμή της selected_device (Εικόνα 7.39, σημείο 16).

Ένας νέος έλεγχος ξεκινά, στο σημείο 17 της Εικόνα 7.39, για τον υπολογισμό με την getCount του πλήθους των συσκευών που έχουν εντοπιστεί στη λίστας των νέων συσκευών mNewDeviceArrayAdapter. Στην περίπτωση που η getCount επιστρέψει την τιμή 0, αυτό σημαίνει ότι δεν εντοπίστηκαν νέες συσκευές από τη διαδικασία της ανίχνευσης, η λίστα των νέων συσκευών mNewDeviceArrayAdapter ενημερώνεται για το μη εντοπισμό νέων συσκευών και εμφανίζεται στο UI το σχετικό μήνυμα.

Για να ξεκινήσει μια σύνδεση με το αντικείμενο BluetoothDevice πρέπει να είναι γνωστή η MAC διεύθυνση του. Σε αυτό το σημείο του κώδικα, η διεύθυνση MAC μαζί με άλλες πληροφορίες όπως είναι το όνομα της συσκευής αποθηκεύονται σε μία ArrayAdapter η οποία εμφανίζεται στην οθόνη του χρήστη. Η διεύθυνση MAC μπορεί αργότερα να ανακτηθεί από τη λίστα, για να ξεκινήσει η σύνδεση. [48], [50]

7.4.3.1.6 Η γραφική απεικόνιση της λίστας των συσκευών

Αρχικά ο χρήστης επιλέγει από το options menu τη "Σύνδεση με συσκευές" (Εικόνα 7.40 α). Οι συσκευές καταχωρούνται σε δύο διαφορετικές λίστες (Εικόνα 7.40 γ). Η πρώτη λίστα περιέχει τις συσκευές με τις οποίες η συσκευή είχε συνδεθεί παλιότερα. Στην περίπτωση που δεν υπάρχει στη λίστα η συσκευή με την οποία επιθυμεί ο χρήστης να συνδεθεί, επιλέγει την "Αναζήτηση συσκευών" (Εικόνα 7.40 β).

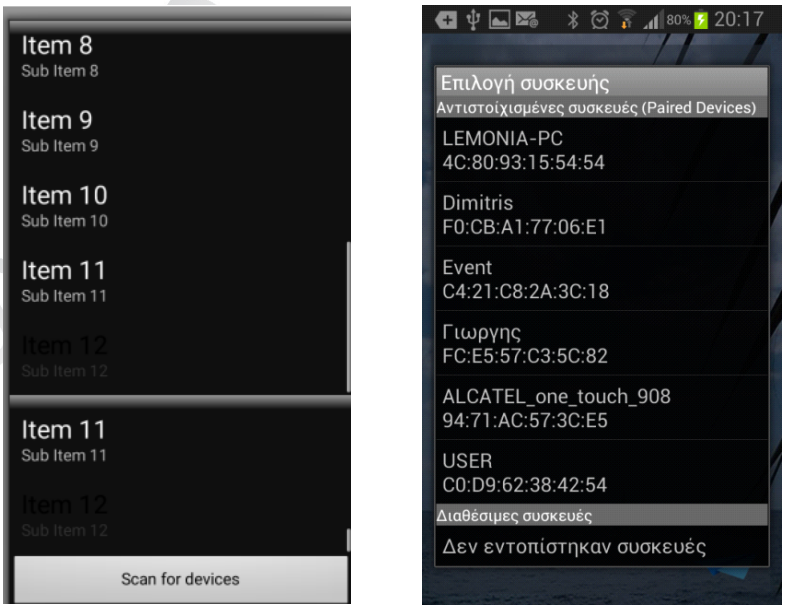


Εικόνα 7.40 α) Το option menu, β) η λίστα των συσκευών πριν την "Αναζήτηση συσκευών και γ) η λίστα των συσκευών μετά τη διαδικασία της αναζήτησης

Η εφαρμογή ξεκινά την ανίχνευση των νέων συσκευών που βρίσκονται εντός της περιοχής της εμβέλειας της και διαθέτουν ενεργοποιημένη την υπηρεσία του BT. Με τον τρόπο αυτό συμπληρώνεται η δεύτερη λίστα με τις νέες συσκευές που ανιχνεύτηκαν. Η υλοποίηση του γραφικού μέρους αυτών των ενεργειών βρίσκεται στο αρχείο device_list.xml. Στην επόμενη παράγραφο παρουσιάζεται ο σχεδιασμός του και ο κώδικας που παράγεται μέσα από το περιβάλλον ανάπτυξης λογισμικού eclipse. [51]

7.4.3.1.7 Η device_list.xml

Όταν ξεκινά η διαδικασία της ανίχνευσης των συσκευών η οθόνη της διεπαφής του χρηστή (User Interface) "Επιλογή συσκευής" περιλαμβάνει δύο λίστες συσκευών προκειμένου ο χρήστης να επιλέξει τη συσκευή με την οποία επιθυμεί να πραγματοποιήσει σύνδεση. Όπως φαίνεται στην εικόνα που ακολουθεί η πρώτη λίστα "Αντιστοιχισμένες συσκευές (Paired Devices)" περιέχει τις συσκευές που ο χρήστης είχε πραγματοποιήσει παλιότερα σύνδεση και έτσι είχε γίνει η αντιστοίχιση μεταξύ των συσκευών. Στη δεύτερη λίστα "Διαθέσιμες συσκευές" καταχωρούνται οι συσκευές που εντοπίζονται από τη διαδικασία της ανίχνευσης αλλά δεν έχει γίνει αντιστοίχιση τους και κατά συνέπεια η σύνδεση τους με τη συσκευή που διενεργεί την ανίχνευση, στη λίστα αυτή περιέχονται οι νέες συσκευές.



Εικόνα 7.41 α)Σχεδιασμός του γραφικού με το eclipse β) η τελική μορφή του device_list.xml

device_list.xml

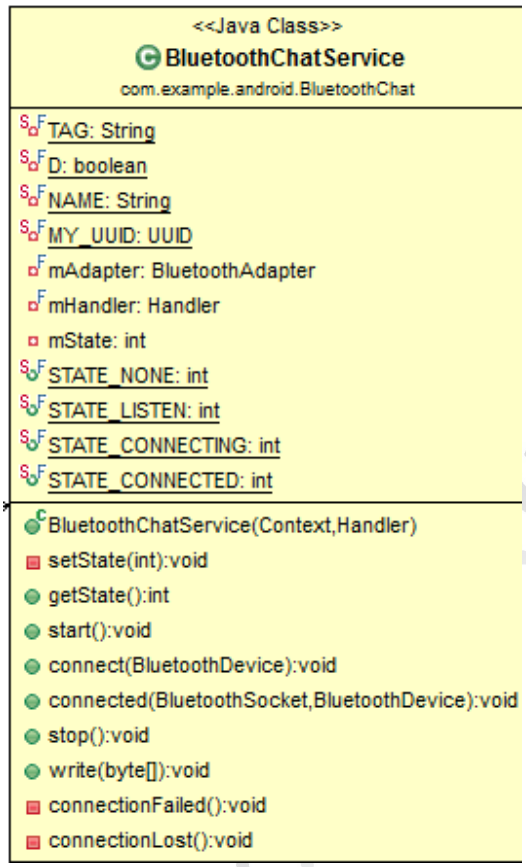
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    >
    <TextView android:id="@+id/title_paired_devices"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/title_paired_devices"
        android:visibility="gone"
        android:background="#666"
        android:textColor="#fff"
        android:paddingLeft="5dp"
    />
    <ListView android:id="@+id/paired_devices"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:stackFromBottom="true"
        android:layout_weight="1"
    />
    <TextView android:id="@+id/title_new_devices"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/title_other_devices"
        android:visibility="gone"
        android:background="#666"
        android:textColor="#fff"
        android:paddingLeft="5dp"
    />
    <ListView android:id="@+id/new_devices"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:stackFromBottom="true"
        android:layout_weight="2"
    />
    <Button android:id="@+id/button_scan"
```

Εικόνα 7.42 Ο κώδικας του αρχείου device_list.xml

Στον παραπάνω κώδικα παρατηρούμε ότι δημιουργούνται δύο αντικείμενα τύπου λίστας τύπου ListView. Η πρώτη λίστα περιέχει τις συσκευές με τις οποίες ο χρήστης είχε συνδεθεί παλιότερα ενώ στη δεύτερη λίστα θα περιλαμβάνονται οι συσκευές που θα εντοπιστούν κατά τη διαδικασία της αναζήτησης νέων συσκευών.

7.4.4 Η κλάση BluetoothChatService

Η κλάση BluetoothChatService είναι αυτή που πραγματοποιεί την εγκατάσταση και τη διαχείριση της σύνδεσης του Bluetooth με τις άλλες συσκευές. Στην Εικόνα 7.43 παρουσιάζεται η δομή της κλάσης. Η κλάση περιλαμβάνει εκτός από τη μέθοδο του κατασκευαστή της, constractor BluetoothChatServise, τις μεθόδους setState(int), getState(), start(), stop(), write(byte[]), connect(BluetoothDevice), connected(BluetoothSocket, BluetoothDevice), connectionFailed() και connectionLost() οι λειτουργίες των οποίων αναλύονται στις επόμενες παραγράφους.



Εικόνα 7.43 Η δομή της κλάσης BluetoothChatService

Η BluetoothChatService δημιουργεί ένα thread που ακούει τις εισερχόμενες συνδέσεις, ένα άλλο thread για τη δημιουργία της σύνδεσης με μία συσκευή και ένα thread για την παρουσίαση των δεδομένων που μεταδίδονται κατά τη σύνδεση.

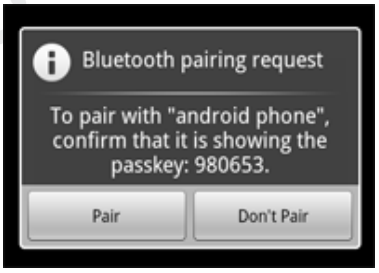
7.4.4.1 Η σύνδεση δύο συσκευών

Για να συνδεθούν δύο συσκευές (Connecting Devices) μέσω της εφαρμογής, θα πρέπει πρώτα να γίνει η εγκατάσταση της εφαρμογής και στις δύο συσκευές. Οι συσκευές θα λειτουργήσουν με βάση το πρότυπο του server και του client, η μια συσκευή θα αναλάβει το ρόλο του server ο οποίος αρχικά θα λαμβάνει αιτήματα δημιουργίας socket και στη συνέχεια θα αποδέχεται ή θα απορρίπτει ανάλογα μια σύνδεση και η άλλη συσκευή στο ρόλο του client πρέπει να ξεκινήσει τη σύνδεση (χρησιμοποιώντας τη διεύθυνση MAC της συσκευής server). Η δημιουργία μίας σύνδεσης έχει επιτευχθεί όταν ο διακομιστής και ο πελάτης έχουν από ένα συνδεδεμένο BluetoothSocket στο ίδιο RFCOMM (Radio frequency communication) κανάλι.

Η απόκτηση του BluetoothSocket από τον server και τον client γίνεται με διαφορετικούς τρόπους. Ο server θα αποκτήσει τη σύνδεση με το BluetoothSocket όταν αποδεχτεί το αίτημα του client για μια εισερχόμενη σύνδεση ενώ ο client θα το λάβει τη σύνδεση με το BluetoothSocket όταν ξεκινήσει ένα RFCOMM κανάλι προς τον server. Σε αυτό το σημείο, κάθε συσκευή μπορεί να λάβει και να στείλει δεδομένα με τη μορφή streaming.

Εικόνα 7.44 Socket Bluetooth

Μια τεχνική υλοποίησης του Bluetooth Socket είναι η αυτόματη προετοιμασία κάθε συσκευής στην ανάληψη του ρόλου του διακομιστή, έτσι ώστε η καθεμία συσκευή να έχει ένα ανοικτό server socket και να αφουγκράζεται τις κλήσεις για σύνδεση. Στη συνέχεια, η συσκευή μπορεί είτε να αρχικοποιήσει μια σύνδεση με μια άλλη συσκευή και να αναλάβει το ρόλο του πελάτη είτε να φιλοξενήσει "host" μία σύνδεση και να ανοίξει ένα server socket ύστερα από αίτημα που έχει στείλε η άλλη συσκευή και να αναλάβει το ρόλο του διακομιστή.



Εικόνα 7.45 Το παράθυρο διαλόγου ζευγοποίησης (pairing) του Bluetooth

Σημείωση: Αν οι δύο συσκευές δεν έχουν αντιστοιχιστεί στο παρελθόν, τότε το πλαίσιο του Android (framework) αυτόματα θα εμφανίσει το αίτημα αντιστοίχισης στο χρήστη κατά τη διάρκεια της διαδικασίας της σύνδεσης, όπως φαίνεται στην Εικόνα 7.45. Κατά την προσπάθεια της σύνδεσης των συσκευών, η εφαρμογή δεν ελέγχει αν οι συσκευές είναι συνδεδεμένες. Η προσπάθεια της δημιουργίας του καναλιού σύνδεσης RFCOMM θα μπλοκαριστεί μέχρι ο χρήστης να αντιστοιχιστεί

με επιτυχία, η δημιουργία του καναλιού σύνδεσης RFCOMM θα αποτύχει, εάν ο χρήστης απορρίψει την αντιστοίχιση, ή όταν αποτύχει η αντιστοίχιση ή λήξει ο χρόνος της αντιστοίχισης. [51]

7.4.4.1.1 Δημιουργία σύνδεσης με την ιδιότητα του διακομιστή (server)

Για να δημιουργηθεί μια σύνδεσης μεταξύ δύο συσκευών πρέπει η μία από τις συσκευές να αναλάβει το ρόλο του διακομιστή (server) διατηρώντας ένα ανοιχτό BluetoothServerSocket. Ο διακομιστής έχει ανοιχτό το server socket (Εικόνα 7.47, σημείο 1) και ακούει τα αιτήματα σύνδεσης που στέλνουν οι γειτονικές συσκευές. Όταν ο διακομιστής αποδεχτεί ένα αιτήματα σύνδεσης τότε παρέχει τη σύνδεση BluetoothSocket στη συσκευή του πελάτη. Όταν εγκατασταθεί η σύνδεση του BluetoothSocket από το BluetoothServerSocket, τότε το BluetoothServerSocket έχει τη δυνατότητα (και επιβάλλεται) να απορρίπτει τα αιτήματα των εισερχόμενων συνδέσεων, εκτός αν πρόκειται να αποδεχτεί περισσότερες συνδέσεις. Αφού δημιουργηθεί η σύνδεση του BluetoothSocket, το BluetoothServerSocket κλείνει και αποδεσμεύονται οι πόροι του.

Παρακάτω περιγράφεται η βασική διαδικασία για τη δημιουργία ενός server socket και η αποδοχή μιας τέτοιας σύνδεσης:

1. Η δημιουργία ενός BluetoothServerSocket γίνεται με την κλήση της μεθόδου listenUsingRfcommWithServiceRecord (NAME, MY_UUID).
   Η σταθερά NAME (τύπου string) περιέχει ένα αναγνωρίσιμο όνομα της υπηρεσίας, για το οποίο το σύστημα αυτόματα θα δημιουργήσει μία νέα εγγραφή Service Discovery Protocol (SDP) στη βάση δεδομένων της συσκευής (το όνομα είναι αυθαίρετο και μπορεί απλά να είναι το όνομα της εφαρμογής). Το MY_UUID είναι ένα αναγνωριστικό UUID (Universally Unique Identifier) που περιλαμβάνεται στην εγγραφή SDP. Δηλαδή, όταν ο πελάτης προσπαθήσει να συνδεθεί με αυτή τη συσκευή, θα έχει ένα αναγνωριστικό UUID που προσδιορίζει με μοναδικό τρόπο την υπηρεσία με την οποία θέλει να συνδεθεί (Εικόνα 7.47, σημείο 2).

Αυτά τα αναγνωριστικά UUIDs πρέπει να ταιριάζουν για να γίνει αποδεκτή η σύνδεση (στο επόμενο βήμα).

Τι είναι το UUID;

Το Universally Unique Identifier (UUID) είναι ένα string ID (128-bit) που χρησιμοποιείται για να προσδιορίσει την υπηρεσία του Bluetooth της εφαρμογής με μοναδικό τρόπο. Το UUID είναι αρκετά μεγάλο και μπορεί να επιλεγεί οποιαδήποτε τυχαία ακολουθία χωρίς να δημιουργεί σύγχυση.

```
// Unique UUID for this application
private static final UUID MY_UUID = UUID.fromString("fa87c0d0-afac-11de-8a39-0800200c9a66");
```

Εικόνα 7.46 Ο κώδικας του Universally Unique Identifier

2. Εφόσον δεν υπάρχει κάποια σύνδεση, ξεκινά η ακρόαση των αιτημάτων σύνδεσης με την κλήση της μεθόδου accept().

   Η κλήση της μεθόδου accept() είναι μια κλήση αποκλεισμού δηλαδή θα επιστρέψει τιμή είτε όταν μία σύνδεση γίνει αποδεκτή ή όταν συμβεί κάποιο σφάλμα και εμφανιστεί μία exception (Εικόνα 7.47, σημείο 3). Μία σύνδεση γίνεται αποδεκτή μόνο όταν μια απομακρυσμένη συσκευή έχει στείλει ένα αίτημα σύνδεσης το οποίο περιέχει το UUID και το UUID ταιριάζει με κάποια από τις εγγραφές που έλαβε ο διακομιστής όταν άκουγε τα εισερχόμενα αιτήματα. Όταν είναι επιτυχής η κλήση της accept() θα επιστρέψει μία σύνδεση BluetoothSocket.
3. Αν έχει εγκαθιδρυθεί μία σύνδεση και δεν πρόκειται να γίνει αποδοχή άλλων συνδέσεων τότε καλείται η μέθοδος close() όπως φαίνεται στο σημείο 5 του κώδικα της παρακάτω εικόνας.

   Με την κλήση της close() γίνεται η αποδέσμευση του server socket αφού δεν θα γίνει η αποδοχή άλλων συνδέσεων και η αποδέσμευση όλων των πόρων που κατανάλωνε το server socket, αλλά δεν κλείνει τη σύνδεση του BluetoothSocket που επιστρεψε η accept(). Το πρωτόκολλο επικοινωνίας RFCOMM επιτρέπει μόνο ένα συνδεδεμένο πελάτη ανά κανάλι τη φορά, οπότε στις περισσότερες περιπτώσεις είναι λογικό να καλείται η μέθοδος close() στην BluetoothServerSocket αμέσως μετά την αποδοχή μιας σύνδεσης socket (Εικόνα 7.47, σημείο 4).

Η κλήση της accept() δεν θα πρέπει να εκτελεστεί στο thread της main activity UI, διότι είναι μια κλήση αποκλεισμού και θα εμποδίσει κάθε άλλη αλληλεπίδραση με

την εφαρμογή. Για την ακύρωση μίας κλήσης αποκλεισμού όπως είναι η accept(), πρέπει να γίνει η κλήση της close() στην BluetoothServerSocket (ή την BluetoothSocket) από άλλο νήμα.

```
private class AcceptThread extends Thread {
    // The local server socket
    private final BluetoothServerSocket mmServerSocket;

    public AcceptThread() {
        BluetoothServerSocket tmp = null;

        // Create a new listening server socket
        try {
            tmp = mAdapter.listenUsingRfcommWithServiceRecord(NAME, MY_UUID);
        } catch (IOException e) {
            Log.e(TAG, "listen() failed", e);
        }
        mmServerSocket = tmp;
    }
    public void run() {
        if (D) Log.d(TAG, "BEGIN mAcceptThread" + this);
        setName("AcceptThread");
        BluetoothSocket socket = null;
        // Listen to the server socket if we're not connected
        while (mState != STATE_CONNECTED) {
            try {
                // This is a blocking call and will only return on a
                // successful connection or an exception
                socket = mmServerSocket.accept();
            } catch (IOException e) {
                Log.e(TAG, "accept() failed", e);
                break;
            }
            // If a connection was accepted
            if (socket != null) {
                synchronized (BluetoothChatService.this) {
                    switch (mState) {
                    case STATE_LISTEN:
                    case STATE_CONNECTING:
                        // Situation normal. Start the connected thread.
                        connected(socket, socket.getRemoteDevice());
                        break;
                    case STATE_NONE:
                    case STATE_CONNECTED:
                        // Either not ready or already connected. Terminate new socket.
                        try {
                            socket.close();
                        } catch (IOException e) {
                            Log.e(TAG, "Could not close unwanted socket", e);
                        }
                        break;
                    }
                }
            }
        }
        if (D) Log.i(TAG, "END mAcceptThread");
    }

    public void cancel() {
        if (D) Log.d(TAG, "cancel " + this);
        try {
            mmServerSocket.close();
        } catch (IOException e) {
            Log.e(TAG, "close() of server failed", e);
        }
    }
}
```

Εικόνα 7.47 Παρουσίαση ενός νήματος από τη πλευρά του server για την αποδοχή εισερχόμενων συνδέσεων

Στον παραπάνω κώδικα μόνο μία εισερχόμενη σύνδεση γίνεται αποδεκτή. Όταν μια σύνδεση γίνει δεκτή και δημιουργηθεί το BluetoothSocket, η εφαρμογή στέλνει το BluetoothSocket που υλοποιεί τη σύνδεση σε ένα ξεχωριστό νήμα και κλείνει το BluetoothServerSocket και το βρόχο.

Όταν η accept() επιστρέψει το BluetoothSocket, το socket είναι ήδη συνδεδεμένο, για το λόγο αυτό δεν πρέπει να καλείται η connect() (όπως συμβαίνει με τη διαδικασία της σύνδεση από την πλευρά του πελάτη).

Η κλήση της getRemoteDevice() χρησιμοποιείται για την δημιουργίας μιας σύνδεσης και ουσιαστικά ξεκινά το νήμα της σύνδεσης.

Στο σημείο 6 του κώδικα της παραπάνω εικόνας, η μέθοδος cancel() καλείται προκειμένου να κλείσει κάθε σύνδεση που επιχειρείται από οποιοδήποτε νήμα μέσα στην BluetoothChatService. [51]

7.4.4.1.2 Δημιουργία σύνδεσης με την ιδιότητα του πελάτη (client)

Η όλη διαδικασία από την πλευρά του πελάτη (client) γίνεται με την δημιουργία του νήματος ConnectThread στο οποίο αρχικά δημιουργούνται δύο αντικείμενα το BluetoothSocket και το BluetoothDevice (Εικόνα 7.48, σημείο 1).

Για να ξεκινήσει η συσκευή του πελάτη μια σύνδεση με μια απομακρυσμένη συσκευή (μια συσκευή που κρατά ένα ανοιχτό server socket), θα πρέπει πρώτα να δημιουργηθεί ένα αντικείμενο BluetoothDevice που αντιστοιχεί στην απομακρυσμένη συσκευή η οποία έχει επιλεγεί κατά την διαδικασία της "Αναζήτησης συσκευών" (Εικόνα 7.48, σημείο 2). Παράλληλα αρχικοποιείται η μεταβλητή tmp που είναι τύπου BluetoothSocket με την τιμή null (Εικόνα 7.48, σημείο 2). Αμέσως μετά χρησιμοποιείται το αντικείμενο BluetoothDevice για τη δημιουργία ενός BluetoothSocket και την έναρξη της σύνδεσης.

Η βασική διαδικασία είναι η ακόλουθη:

1. Με την κλήση της μεθόδου createRfcommSocketToServiceRecord (MY_UUID) δημιουργείται μία σύνδεση BluetoothSocket χρησιμοποιώντας το BluetoothDevice που αντιπροσωπεύει τη συσκευή που

εντοπιστικέ από τη διαδικασία της αναζήτησης των συσκευών (Εικόνα 7.48, σημείο 3).

Προετοιμάζεται η σύνδεση του BluetoothSocket με την οποία θα συνδεθεί η συσκευή BluetoothDevice (Εικόνα 7.48, σημείο 4). Το UUID που περνάει εδώ πρέπει να ταιριάζει με το UUID που χρησιμοποιείται από τον εξυπηρετητή-server όταν ανοίξει το BluetoothServerSocket του (με τη κλήση της μεθόδου listenUsingRfcommWithServiceRecord(String, UUID)).

2. Εκκίνηση της σύνδεσης με την κλήση της μεθόδου connect().

   Με την κλήση της μεθόδου connect(), όπως φαίνεται στο σημείο 6 της εικόνας που ακολουθεί, το σύστημα ξεκινά μία αναζήτηση SDP στην απομακρυσμένη συσκευή, με σκοπό το ταίριασμα του UUID. Αν η αναζήτηση είναι επιτυχής και η απομακρυσμένη συσκευή αποδεχτεί τη σύνδεση, θα αποδοθεί το κανάλι RFCOMM προς χρήση κατά τη διάρκεια της σύνδεσης και η connect() θα έχει εκτελεστεί με επιτυχία. Η κλήση της connect() είναι μια κλήση αποκλεισμού. Εάν, για οποιοδήποτε λόγο, η σύνδεση αποτύχει ή παρέλθει η λήξη του χρόνου της μεθόδου connect() (μετά από περίπου 12 δευτερόλεπτα) τότε θα εμφανιστεί μια exception. Αφού διαπιστωθεί κάποιο σφάλμα με την εμφάνιση της exception τότε θα κληθεί η connectionFailed() και θα ξεκινήσει η διαδικασία κλεισίματος του socket με την κλήση της close() όπως φαίνεται στο σημείο 7 του κώδικα της Εικόνα 7.48.

   Επειδή η μέθοδος connect() είναι μια κλήση αποκλεισμού, η διαδικασία σύνδεσης θα πρέπει να γίνεται πάντα σε νήμα ξεχωριστό από αυτό της main activity.

Σημείωση: Η συσκευή θα πρέπει να μην εκτελεί την διαδικασία της ανίχνευσης συσκευών κατά την κλήση της μεθόδου connect(). Αν η διαδικασία της ανίχνευσης είναι σε εξέλιξη, τότε η προσπάθεια της σύνδεσης θα επιβραδυνθεί σημαντικά και είναι πιθανό να αποτύχει.

```
private class ConnectThread extends Thread {
    private final BluetoothSocket mmSocket;
    private final BluetoothDevice mmDevice;

    public ConnectThread(BluetoothDevice device) {
        mmDevice = device;
        BluetoothSocket tmp = null;

        // Get a BluetoothSocket for a connection with the
        // given BluetoothDevice
        try {
            tmp = device.createRfcommSocketToServiceRecord(MY_UUID);
        } catch (IOException e) {
            Log.e(TAG, "create() failed", e);
        }
        mmSocket = tmp;
    }

    public void run() {
        Log.i(TAG, "BEGIN mConnectThread");
        setName("ConnectThread");
        // Always cancel discovery because it will slow down a connection
        mAdapter.cancelDiscovery();
        // Make a connection to the BluetoothSocket
        try {
            // This is a blocking call and will only return on a
            // successful connection or an exception
            mmSocket.connect();
        } catch (IOException e) {
            connectionFailed();
            // Close the socket
            try {
                mmSocket.close();
            } catch (IOException e2) {
                Log.e(TAG, "unable to close() socket during connection failure", e2);
            }
            // Start the service over to restart listening mode
            BluetoothChatService.this.start();
            return;
        }
        // Reset the ConnectThread because we're done
        synchronized (BluetoothChatService.this) {
            mConnectThread = null;
        }
        // Start the connected thread
        connected(mmSocket, mmDevice);
    }

    public void cancel() {
        try {
            mmSocket.close();
        } catch (IOException e) {
            Log.e(TAG, "close() of connect socket failed", e);
        }
```

Εικόνα 7.48 Παρουσίαση ενός νήματος από τη πλευρά του client για την εκκίνηση μίας σύνδεσης

Στο σημείο 5 του κώδικα της παραπάνω εικόνας, καλείται η cancelDiscovery() για να αποδεσμευτούν οι πόροι που δεσμεύονται από την εφαρμογή κατά την διαδικασία της ανίχνευσης. Η cancelDiscovery() καλείται πάντα πριν να γίνει η σύνδεση.

Η κλήση της BluetoothChatService() γίνεται εφόσον έχει τελειώσει η διαδικασία της σύνδεσης για να ξεκινήσει το νήμα της μεταφοράς των δεδομένων του chat, (Εικόνα 7.48, σημείο 8).

Το νήμα της σύνδεσης ξεκινά με την connected(mmSocket, mmDevice) στο σημείο 9 του κώδικα της παραπάνω εικόνας, μέσα από το οποίο περνιούνται ως παράμετροι η mmSocket και η mmDevice.

Η κλήση της cancel() γίνεται όταν μια σύνδεση βρίσκεται σε εξέλιξη για να κλείσει το socket με την διαδικασία κλεισίματος του socket μέσω της close() (Εικόνα 7.48, σημείο 10). Με αυτόν τον τρόπο θα κλείσει αμέσως το socket και θα γίνει η αποδέσμευση όλων των εσωτερικών πόρων. [51]

7.4.4.2 Η διαχείριση της σύνδεσης

Η BluetoothChatServise αναλαμβάνει τη διαχείριση της σύνδεσης. Η ConnectedThread που περιγράφεται σε αυτή την παράγραφο διαχειρίζεται το νήμα (thread) το οποίο τρέχει κατά τη διάρκεια της σύνδεσης με μια απομακρυσμένη συσκευή και εκτελεί την μετάδοση των εισερχόμενων και εξερχόμενων μηνυμάτων. Στην περίπτωση που έχουμε μία σύνδεση δύο (ή περισσότερων) συσκευών, αυτό σημαίνει ότι η καθεμία από τις συσκευές αυτές θα έχει μια σύνδεση BluetoothSocket. Η δήλωση αυτής της μεταβλητής και η δημιουργία του socket γίνεται αντίστοιχα στα σημεία 1 και 2 του κώδικα της παρακάτω εικόνας. Με τον τρόπο αυτό ξεκινά η μετάδοση των δεδομένων μεταξύ των συσκευών. Χρησιμοποιώντας τη σύνδεση BluetoothSocket, η διαδικασία της μεταφοράς των δεδομένων είναι απλή:

1. Για τη διαδικασία της μεταφοράς των δεδομένων χρησιμοποιούνται δύο αντικείμενα τύπου InputStream και OutputStream. Η δήλωση, η δημιουργία και η αρχικοποίηση του InputStream και του OutputStream με τα αντικείμενα mmInStream και mmOutStream γίνεται στο σημείο 3 της Εικόνα 7.49. Τα Streams χειρίζονται τη μεταφορά των δεδομένων μέσω του socket, όπως φαίνεται στο σημείο 4 του κώδικα της παρακάτω εικόνας. Με τη χρήση των μεθόδων getInputStream() και getOutputStream(), τα εισερχόμενα και εξερχόμενα streams αποθηκεύονται αρχικά στα προσωρινά αντικείμενα tmpIn και tmpOut και μετά τα περιεχόμενα των προσωρινών

αυτών μεταβλητών αποθηκεύονται στα αντικείμενα mmInStream και mmOutStream (Εικόνα 7.49, σημείο 5).

```
private class ConnectedThread extends Thread {
    private final BluetoothSocket mmSocket;
    private final InputStream mmInStream;
    private final OutputStream mmOutStream;

    public ConnectedThread(BluetoothSocket socket) {
        Log.d(TAG, "create ConnectedThread");
        mmSocket = socket;
        InputStream tmpIn = null;
        OutputStream tmpOut = null;

        // Get the BluetoothSocket input and output streams
        try {
            tmpIn = socket.getInputStream();
            tmpOut = socket.getOutputStream();
        } catch (IOException e) {
            Log.e(TAG, "temp sockets not created", e);
        }

        mmInStream = tmpIn;
        mmOutStream = tmpOut;
    }
```

1 2 3 4 5

Εικόνα 7.49 Η ConnectedThread της BluetoothChatService (σημεία 1-5)

2. Στη συνέχεια μέσα στη διαδικασία run() γίνεται η ανάγνωση και η εγγραφή των δεδομένων μέσω των streams με τη χρήση των μεθόδων read(byte []) και write(byte []), αντίστοιχα. Σημαντική είναι η χρήση ενός ειδικού νήματος για όλα τα stream της ανάγνωσης και της γραφής, διότι οι κλήσεις των μεθόδων read(byte []) και write(byte []) είναι κλήσεις αποκλεισμού. Για την αποθήκευση των bytes που επιστρέφονται από την read() δημιουργείται μία buffer μεταβλητή τύπου byte (Εικόνα 7.50, σημείο 6).

```
public void run() {
    Log.i(TAG, "BEGIN mConnectedThread");
    byte[] buffer = new byte[1024];
    int bytes;

    // Keep listening to the InputStream while connected
    while (true) {
        try {
            // Read from the InputStream
            bytes = mmInStream.read(buffer);

            // Send the obtained bytes to the UI Activity
            mHandler.obtainMessage(BluetoothChat.MESSAGE_READ, bytes, -1, buffer)
                    .sendToTarget();
        } catch (IOException e) {
            Log.e(TAG, "disconnected", e);
            connectionLost();
            break;
        }
    }
}
```

6 7

Εικόνα 7.50 Η run() της ConnectedThread (σημεία 6, 7)

Ο βρόχος while (Εικόνα 7.50, σημείο 7) δημιουργείται με σκοπό το InputStream να αφουγκράζεται τη λήψη των εισερχόμενων δεδομένων μέχρι να εμφανιστεί μία exception. Στη συνέχεια η read (byte []) θα περιμένει (μπλοκάρει) μέχρι να υπάρξει κάτι για να διαβάσει από το stream.

Η διαδικασία της εγγραφής πραγματοποιείται με την write(byte []) (Εικόνα 7.51, σημείο 8) η οποία συνήθως δεν μπλοκάρει. Η write(byte []) μπορεί να μπλοκάρει τον έλεγχο της ροής, εάν η απομακρυσμένη συσκευή δεν καλεί αρκετά γρήγορα τη read (byte []) για να διαβάσει δεδομένα με αποτέλεσμα να γεμίσουν οι ενδιάμεσοι buffers επειδή ο κύριος βρόχος αφιερώνεται στην ανάγνωση των δεδομένων από την InputStream.

```
public void write(byte[] buffer) {
    try {
        mmOutStream.write(buffer);

        // Share the sent message back to the UI Activity
        mHandler.obtainMessage(BluetoothChat.MESSAGE_WRITE, -1, -1, buffer)
                .sendToTarget();
    } catch (IOException e) {
        Log.e(TAG, "Exception during write", e);
    }
}

public void cancel() {
    try {
        mmSocket.close();
    } catch (IOException e) {
        Log.e(TAG, "close() of connect socket failed", e);
    }
}
```

8

9

Εικόνα 7.51 Η write() και η cancel() της ConnectedThread (σημεία 8, 9)

Ο constructor δημιουργεί τα απαραίτητα streams. Όταν η read(byte []) διαβάσει τα δεδομένα και επιστραφούν τα bytes από το stream τότε τα δεδομένα αποστέλλονται στην κύρια δραστηριότητα του UI χρησιμοποιώντας ένα αντικείμενο τύπου Handler από τη γονική κλάση. Στη συνέχεια πηγαίνει πίσω και περιμένει περισσότερα byte από το stream.

Η αποστολή των εξερχόμενων δεδομένων στην απομακρυσμένη συσκευή γίνεται με την κλήση της μεθόδου write() του νήματος από την κύρια δραστηριότητα.

Η μέθοδος cancel() (Εικόνα 7.51, σημείο 9) του νήματος είναι σημαντική για να μπορεί να τερματιστεί η σύνδεση με το κλείσιμο του BluetoothSocket οποιαδηποτε χρονική στιγμή. Αυτή η μέθοδος θα πρέπει πάντα να καλείται όταν η σύνδεση του Bluetooth τελιώσει. [51]

7.5 Δημιουργία και διαχείριση της ΒΔ

Η Βάση Δεδομένων (ΒΔ) παρέχει έναν γρήγορο και ευέλικτο τρόπο πρόσβασης για τη διαχείριση πολυπλοκότερων δόμων δεδομένων. Στις παραγράφους που ακολουθούν παρουσιάζονται τα σημαντικότερα σημεία των ΒΔ που χρησιμοποιεί η εφαρμογή. Οι Android εφαρμογές χρησιμοποιούν την γλώσσα SQLite για τη δημιουργία και διαχείριση των εργασιών που αφορούν τις ΒΔ

7.5.1 Τι είναι όμως η SQLite;

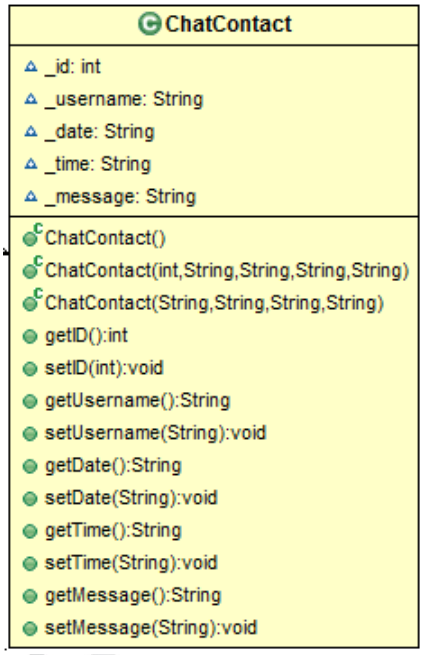
Η SQLite είναι μία γλωσσά διαχείρισης βάσεων δεδομένων, ανοιχτού κώδικα (Open Source), η οποία υποστηρίζει τα χαρακτηριστικά συσχέτισης της SQL όπως είναι η σύνταξη, η μεταφορά και οι έτοιμες δηλώσεις. Επιτρέπει τις δοσοληψίες ACID (Atomicity, Consistency, Isolation, Durability ήτοι ατομικότητα, συνέπεια, απομόνωση, μονιμότητα) που είναι ένα σύνολο ιδιοτήτων το οποίο εγγυάται ότι οι συναλλαγές στην ΒΔ λειτουργούν αξιόπιστα. Δεν απαιτεί ξεχωριστή εγκατάσταση ή συντήρηση, είναι ενιαίο dll που αναπτύσσεται παράλληλα με το API. Αποθηκεύει σε μοναδικό αρχείο (crossplatform compatible). Σε σχέση με την SQL η χρήση της SQLite έχει μικρές απαιτήσεις σε χώρο και μνήμη (325KiB fully configured 190KiB ο βασικός πυρήνας). Είναι γραμμένη σε ANSIC πράγμα που παρέχει διαθέσιμα bindings για πάρα πολλές άλλες γλώσσες.

Εικόνα 7.52 Η SQLite

Απαιτεί λιγότερη μνήμη (περίπου 250 KByte) κατά το χρόνο εκτέλεσης του κώδικα. Η SQLite υποστηρίζει τους τύπους δεδομένων των κειμένων, των ακέραιων και των πραγματικών αριθμών, τα δεδομένων που εμφανίζονται με οποιαδήποτε άλλη μορφή θα πρέπει να μετατραπούν στις παραπάνω μορφές δεδομένων. [52], [53]

7.5.2 Η κλάση ChatContact

Η κλάση ChatContact χρησιμοποιείται για τη δημιουργία των οντοτήτων των χρηστών που ανταλλάσουν τα chat μηνύματα. Ο κώδικας της java δημιουργεί τη δομή της ΒΔ, τα πεδία της ΒΔ θα χρησιμοποιηθούν αργότερα για να γίνει η διαχείριση και η αποθήκευση των δεδομένων στη βάση. Οι μεταβλητές της ΒΔ είναι η _id που είναι ακέραιου τύπου και αντιστοιχεί στον κωδικό που χρησιμοποιείται ως πρωτεύον κλειδί και τα αλφαριθμητικά _username, _date, _time και _message.



Εικόνα 7.53 Η δομή της κλάσης ChatContact

Η δομή της ChatContact που παρουσιάζεται στην Εικόνα 7.53 περιέχει τις μεθόδους:

- τρεις κατασκευαστές constructors ChatContact
- την getID και την setID για την εισαγωγή και την εξαγωγή της παραμέτρου του κωδικού, _id, στη ΒΔ
- την getUsername και την setUsername για την εισαγωγή και την εξαγωγή της παραμέτρου του ονόματος, _username, στη ΒΔ
- την getDate και την setDate για την εισαγωγή και την εξαγωγή της παραμέτρου της ημερομηνίας, _date, στη ΒΔ
- την getTime και την setTime για την εισαγωγή και την εξαγωγή της παραμέτρου της ώρας, _time, στη ΒΔ

- την getMessage και την setMessage για την εισαγωγή και την εξαγωγή της παραμέτρου του μηνύματος, _message, στη ΒΔ

```
package com.example.android.BluetoothChat;
public class ChatContact {
    //private variables
    int _id;
    String _username;
    String _date;
    String _time;
    String _message;

    // Empty constructor
    public ChatContact(){

    }
    // constructor
    public ChatContact(int id, String username, String date, String time, String msg){
        this._id = id;
        this._username = username;
        this._date = date;
        this._time = time;
        this._message = msg;
    }
    // constructor
    public ChatContact(String username, String date, String time, String msg){
        this._username = username;
        this._date = date;
        this._time = time;
        this._message = msg;
    }
    // getting ID
    public int getID(){
        return this._id;
    }
    // setting id
    public void setID(int id){
        this._id = id;
    }
    // getting username
    public String getUsername(){
        return this._username;
    }
    // setting username
    public void setUsername(String username){
        this._username = username;
    }
    // getting date
    public String getDate(){
        return this._date;
    }
    // setting date
    public void setDate(String date){
        this._date = date;
    }
    // getting time
    public String getTime(){
        return this._time;
    }
    // setting time
    public void setTime(String time){
        this._time = time;
    }
    // getting message
    public String getMessage(){
        return this._message;
    }
    // setting message
    public void setMessage(String msg){
        this._message = msg;
    }
}
```

1

Εικόνα 7.54 Η κλάση ChatContact

Για το σκοπό της δημιουργίας των επαφών χρησιμοποιούνται τρεις διαφορετικοί constructors, η τεχνική αυτή είναι γνωστή ως τεχνική της υπερφόρτωσης (overloading) μίας μεθόδου και υποστηρίζεται από την γλώσσα προγραμματισμού της Java. Συμφώνα με την τεχνική αυτή πολλαπλές μέθοδοι μπορούν να μοιράζονται το ίδιο όνομα αλλά δεν μπορούν να υπάρχουν παραπάνω από μία με το ίδιο αποτύπωμα. Με λίγα λόγια η υπερφόρτωση μεθόδων δημιουργεί μεθόδους με το ίδιο όνομα αλλά με διαφορετικά ορίσματα τόσο ως προς το πλήθος όσο και ως προς το είδος των ορισμάτων. Στην περίπτωση μας η υπερφόρτωση εντοπίζεται στη δήλωση του κατασκευαστή constructor.

Όπως φαίνεται και στο σημείο 1 της Εικόνα 7.54 δημιουργούνται τρεις μέθοδοι κατασκευαστών, η μία απ' αυτές είναι ένας άδειος constructor ο οποίος δεν δέχεται κανένα όρισμα και χρησιμοποιείται αρχικά κατά τη δήλωση της ΒΔ όπου δεν απαιτείται η ύπαρξη των πεδίων. Η δεύτερη μέθοδος κατασκευαστή δέχεται τα ορίσματα _id, _username, _date, _time και _message ως παραμέτρους εισόδου όταν δημιουργείται μία εγγραφή, και η τρίτη δέχεται ως παραμέτρους εισόδου τα ορίσματα _username, _date, _time και _message, δηλαδή και οι τρεις αυτοί κατασκευαστές έχουν διαφορετικό αποτύπωμα.

Οι επόμενες μέθοδοι της κλάσης χρησιμοποιούνται για την εισαγωγή και την ανάκτηση των πεδίων _username, _id, _date, _time και _message των χρηστών.

Πίνακας 7.1 Η δομή της ΒΔ της ChatContact

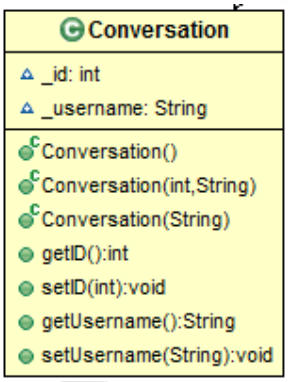
Πίνακας: ChatContact		
Πεδίο	**Τύπος**	**Κλειδί**
_id	int	Primary
_username	String	
_date	String	
_time	String	
_message	String	

Στον προηγούμενο πίνακα παρουσιάζεται η δομή της ΒΔ που δημιουργείται με την κλάση της ChatContact. [54], [55]

7.5.3 Η κλάση Conversation

Παρόμοια με την δημιουργία της κλάση ChatContact είναι και η δημιουργία της κλάσης Conversation η οποία χρησιμοποιείται βοηθητικά για την ταξινόμηση των ονομάτων των χρηστών που ανταλλάσουν τα chat μηνύματα. Η κλάση Conversation όπως φαίνεται στην Εικόνα 7.55 περιέχει τις μεθόδους:

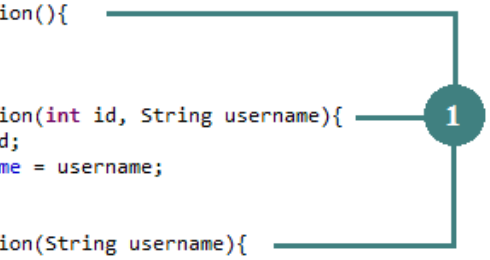
- τρεις κατασκευαστές constructors Conversation
- την getID και την setID για την εισαγωγή και την εξαγωγή της παραμέτρου του κωδικού, _id, στη ΒΔ
- την getUsername και την setUsername για την εισαγωγή και την εξαγωγή της παραμέτρου του ονόματος, _username, στη ΒΔ



Εικόνα 7.55 Η δομή της κλάσης Conversation

Και εδώ εφαρμόζεται η τεχνική της υπερφόρτωσης των μεθόδων του κατασκευαστή constructor. Στην Εικόνα 7.56 στο σημείο 1, δημιουργούνται τρεις μέθοδοι κατασκευαστών, ο ένας απ' αυτούς είναι ένας άδειος constructor, ο δεύτερος κατασκευαστής δέχεται όλα τα ορίσματα της κλάσης αυτής τα οποία είναι το _id και το _username ως παραμέτρους εισόδου, και τέλος ο τρίτος κατασκευαστής δέχεται ως μόνο όρισμα το _username. Κάθε κατασκευαστής έχει διαφορετικό αποτύπωμα.

```
public class Conversation {
    //private variables
    int _id;
    String _username;

    // Empty constructor
    public Conversation(){

    }
    // constructor
    public Conversation(int id, String username){
        this._id = id;
        this._username = username;
    }
    // constructor
    public Conversation(String username){
        this._username = username;
    }
    // getting ID
    public int getID(){
        return this._id;
    }
    // setting id
    public void setID(int id){
        this._id = id;
    }
    // getting username
    public String getUsername(){
        return this._username;
    }
    // setting username
    public void setUsername(String username){
        this._username = username;
    }
}
```



Εικόνα 7.56 Ο κώδικας της κλάσης Conversation

Οι επόμενες μέθοδοι της κλάσης Conversation χρησιμοποιούνται για την εισαγωγή και την ανάκτηση των πεδίων _username και _id των χρηστών.

Πίνακας 7.2 Η δομή του πίνακα Conversation

Πίνακας: Conversation		
Πεδίο	**Τύπος**	**Κλειδί**
_id	int	Primary
_username	String	

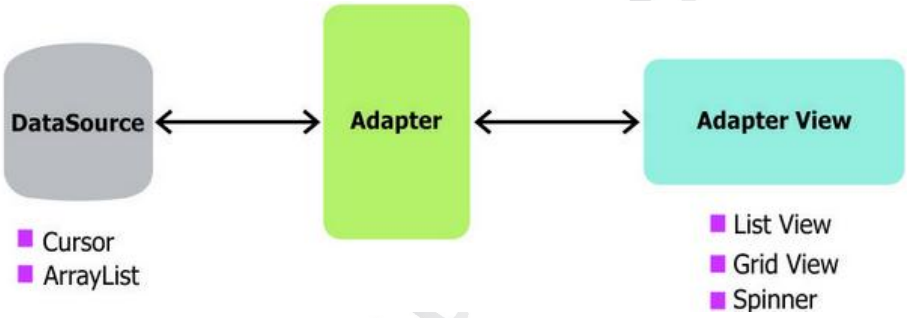
Στον Πίνακας 7.2 παρουσιάζεται η δομή της ΒΔ που δημιουργείται με την κλάση Conversation. [54], [55]

7.5.4 Η κλάση arrayAdapter

Στην ανάπτυξη εφαρμογών Android, για την εμφάνιση μιας κατακόρυφης κυλιόμενης λίστας στοιχείων χρησιμοποιείται μια λίστα αντικειμένων ListView στην οποία έχουν προστεθεί τα δεδομένα με τη συνδρομή ενός adapter (προσαρμογέας ή

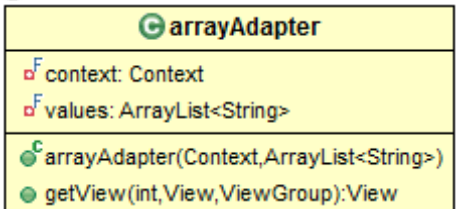
μετασχηματιστής). Το Λ.Σ. Android παρέχει εργαλεία (widgets) για την δημιουργία των adapter, τα σημαντικότερα απ' αυτά είναι το ArrayAdapter και το CursorAdapter. Ο adapter που ενδείκνυται για τη μετατροπή ενός πίνακα αντικειμένων, ArrayList, σε αντικείμενα View που φορτώνονται σε μία ListView, είναι ο ArrayAdapter. Ο ArrayAdapter μπορεί να διαχειριστεί δεδομένα σε πίνακες ή λίστες (Εικόνα 7.57). Είναι ο σύνδεσμος ανάμεσα σε μια πηγή δεδομένων όπως η ArrayList και την οπτική αναπαράσταση της ListView και ρυθμίζει δύο θέματα:

- Ποιος πίνακας θα τροφοδοτήσει με δεδομένα τη λίστα
- Πώς θα μετατραπεί το κάθε δεδομένο του πίνακα σε ένα αντίστοιχο αντικείμενο View



Εικόνα 7.57 Ο ArrayAdapter διαχειριζεται δεδομένα σε πίνακες ή λίστες

Η κλάση της arrayAdapter περιέχει τις σταθερές της context (final) που είναι τύπου Contex και της values που είναι τύπου ArrayList, ενώ στην περιοχή που δηλώνονται οι μέθοδοι διακρίνουμε τον κατασκευαστή της arrayAdapter και τη μέθοδο getView.



Εικόνα 7.58 Η δομή της κλάσης arrayAdapter

Στον κώδικα της κλάσης δηλώνονται τα συστατικά στοιχεία της arrayAdapter που θα χρησιμοποιηθούν. Αυτά τα συστατικά υπάρχουν έτοιμα και είναι διαθέσιμα από την Java και το Λ.Σ. του Android και μπορούν να χρησιμοποιηθούν από την κλάση της arrayAdapter αφού πρώτα δηλωθούν με import στην αρχή του κώδικα της. Τέτοια συστατικά στοιχεία της κλάσης της arrayAdapter είναι: η ArrayList η οποία είναι ένα συστατικό που παρέχεται από την Java, η κλάση Content η οποία παρέχεται από το

Λ.Σ. του Android και επιτρέπει την πρόσβαση σε συγκεκριμένους πόρους και κλάσεις της εφαρμογής, και οι κλήσεις λειτουργιών σε επίπεδο εφαρμογής όπως είναι η έναρξη των activities, η εκπομπή και η λήψη intent's, κ.λπ. Από το Λ.Σ. του Android παρέχονται επίσης για τη διαχείριση των views τα συστατικά LayoutInflater, View και ViewGroup, καθώς και τα εργαλεία του ArrayAdapter, του ImageView και του TextView.

```
import java.util.ArrayList;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;

public class arrayAdapter extends ArrayAdapter<String> {
  private final Context context;
  private final ArrayList<String> values;

  public arrayAdapter(Context context, ArrayList<String> values) {
    super(context, R.layout.listbox, values);
    this.context = context;
    this.values = values;
  }

@Override
  public View getView(int position, View convertView, ViewGroup parent) {
    LayoutInflater inflater = (LayoutInflater) context
        .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    View rowView = inflater.inflate(R.layout.listbox, parent, false);
    TextView textView = (TextView) rowView.findViewById(R.id.textbox);
    ImageView imageView = (ImageView) rowView.findViewById(R.id.imagebox);
    textView.setText(values.get(position));
    // Change the icon for Windows and iPhone
    String s = values.get(position);

    return rowView;
  }
}
```

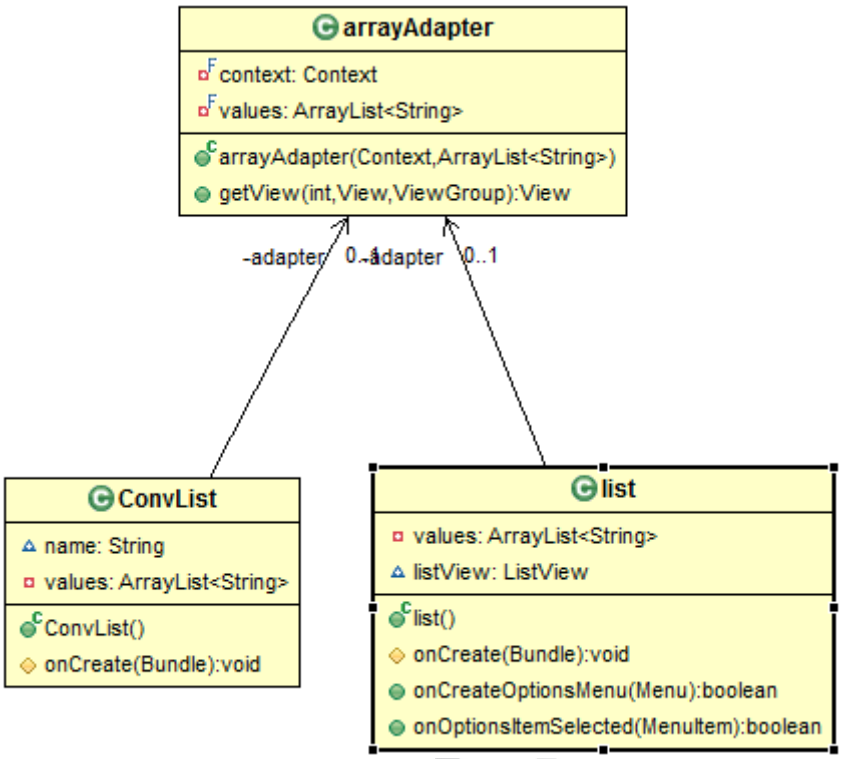
1

2

Εικόνα 7.59 Ο κώδικας της arrayAdapter

Για να μπορεί να χρησιμοποιηθεί το αντικείμενο του ArrayAdapter πρέπει πρώτα να γίνει η αρχικοποίηση του στη ListView και να οριστούν το context (activity), οι πόροι XML και ο πίνακας των δεδομένων (Εικόνα 7.59, σημείο 1). Μετά πρέπει να οριστεί στον adapter η διαδικασία μετατροπής του αντικειμένου της Java σε View με τη μέθοδο getView (Εικόνα 7.59, σημείο 2). Η κλάση ArrayAdapter μπορεί να διαχειριστεί μια λίστα ή έναν πίνακα που περιέχει αντικείμενα της java. Κάθε αντικείμενο τοποθετείτε σε μια εγγραφή του view με τη rowView.

Αφού δημιουργηθεί η κλάση του arrayAdapter σύμφωνα με τις απαιτήσεις της εφαρμογής, σειρά έχει η σύνδεση του adapter με τη ListView.



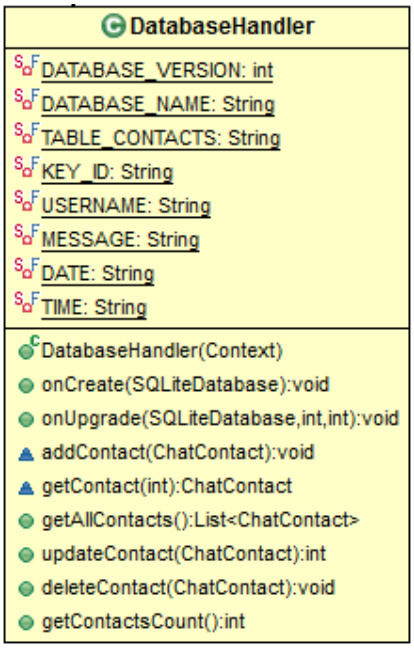
Διάγραμμα 7.1 Το Διάγραμμα Κλάσεων της σύνδεσης του arrayAdapter με τις κλάσεις ConvList και list

Το αντικείμενο arrayAdapter της εφαρμογής που εξετάζουμε λαμβάνει στοιχεία από τις κλάσεις ConvList και list όπως απεικονίζεται από το παραπάνω διάγραμμα. [56]

7.5.5 Η κλάση DatabaseHandler

Αυτή η κλάση είναι υπεύθυνη για την δημιουργία της βάσης δεδομένων και διαχειρίζεται την επικοινωνία του χρήστη με την βάσης δεδομένων. Η java, όπως και άλλες αντικειμενοστραφείς γλώσσες προγραμματισμού, υποστηρίζουν την κληρονομικότητα μεταξύ των κλάσεων, η οποία επιτρέπει σε μια κλάση (παιδί) να κληρονομήσει τις ιδιότητες της κλάσης του γονέα.

Αυτό συμβαίνει και με την κλάση DatabaseHandler η οποία κάνει επέκταση (extend) την κλάση SQLiteOpenHelper. Η SQLiteOpenHelper είναι μια αφηρημένη (abstract) κλάση που υπάρχει στα Android και χρησιμοποιείται για την αλληλεπίδραση του κώδικα και του αρχείου της βάσης δεδομένων.



Εικόνα 7.60 Η δομή της κλάσης DatabaseHandler

Η δομή της κλάσης DatabaseHandler (Εικόνα 7.60) περιέχει τις μεθόδους:

- την DatabaseHandler που είναι ο κατασκευαστής (constructor) της κλάσης
- την onCreate(SQLiteDatabase), για την δήλωση της ΒΔ κατά την φάση της δημιουργίας της εφαρμογής
- την onUpgrade(SQLiteDatabase,int,int)
- την addContact(ChatContact) για την εισαγωγή μιας εγγραφής στη ΒΔ
- την getContact(int) για την ανάκτηση μιας εγγραφής από τη ΒΔ με τη χρήση του πρωτεύοντος κλειδιού _id
- την getAllContacts(int) για την ανάκτηση όλων των εγγραφών
- την updateContact(ChatContact) για την ενημέρωση μιας εγγραφής
- την deleteContact(ChatContact) για την διαγραφή μιας εγγραφής
- την getContactsCount για την ανάκτηση του πρωτεύοντος κλειδιού

Πίνακας 7.3 Η δομή του πίνακα contacts

Πινάκας: contacts		
Πεδίο	**Τύπος**	**Κλειδί**
KEY_ID	INT	PRIMARY KEY
USERNAME	TEXT	
DATE	TEXT	
TIME	TEXT	
MESSAGE	TEXT	

Η δομή του πίνακα contacts αποτελείται από πέντε στήλες και παρουσιάζεται στον Πίνακας 7.3. Η πρώτη στήλη είναι αυτή του μοναδικού πρωτεύοντος κλειδιού (KEY_ID) η οποία είναι υποχρεωτική και σύμφωνα μ' αυτή γίνεται η διαχείριση των δεδομένων που αποθηκεύονται στη ΒΔ. Η στήλη USERNAME χρησιμοποιείται για την καταχώρηση των ονομάτων των χρηστών, η στήλη MESSAGE αποθηκεύει τα μηνύματα που ανταλλάσσονται μεταξύ των χρηστών και οι στήλες DATE και TIME που χρησιμοποιούνται για την αποθήκευση της ημερομηνίας και της ώρας αποστολής των μηνυμάτων αντίστοιχα.

```
public class DatabaseHandler extends SQLiteOpenHelper {
    // All Static variables
    // Database Version
    private static final int DATABASE_VERSION = 1;

    // Database Name
    private static final String DATABASE_NAME = "contactsManager";

    // Contacts table name
    private static final String TABLE_CONTACTS = "contacts";

    // Contacts Table Columns names
    private static final String KEY_ID = "id";
    private static final String USERNAME = "nickname";
    private static final String MESSAGE = "message";
    private static final String DATE = "date";
    private static final String TIME= "time";

    public DatabaseHandler(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    // Creating Tables
    @Override
    public void onCreate(SQLiteDatabase db) {
        String CREATE_CONTACTS_TABLE = "CREATE TABLE " + TABLE_CONTACTS + "("
                + KEY_ID + " INTEGER PRIMARY KEY," + USERNAME + " TEXT,"
                + DATE + " TEXT," + TIME + " TEXT,"
                + MESSAGE + " TEXT" + ")";
        db.execSQL(CREATE_CONTACTS_TABLE);
    }

    // Upgrading database
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // Drop older table if existed
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_CONTACTS);

        // Create tables again
        onCreate(db);
    }
```

Εικόνα 7.61 Ο κώδικα της κλάσης DatabaseHandler (σημεία 1-7)

Αρχικά στα σημεία 1 και 2 της Εικόνα 7.61 του κώδικα καθορίζεται η έκδοση DATABASE_VERSION της ΒΔ και δηλώνεται το όνομα DATABASE_NAME της ΒΔ. Η έκδοση της ΒΔ στον κώδικα είναι 1 και είναι η ελάχιστη έκδοση που μπορεί να χρησιμοποιηθεί πράγμα που παρέχει συμβατότητα με όλους τους AVD SDK. Ακολουθεί η δήλωση και η αρχικοποίηση του πίνακα TABLE_CONTACTS και ο

ορισμός των στηλών του πίνακα που δηλώνονται με τον τύπο των σταθερών (final), όπως φαίνεται στο σημείο 3 της Εικόνα 7.61.

Στο σημείο 4 της Εικόνα 7.61 δηλώνεται η ΒΔ. Η λειτουργία της ΒΔ μέσα στον κύκλο της ζωής της εφαρμογής ξεκινά από την onCreate (Εικόνα 7.61, σημείο 5).

Η μέθοδος onUpgrade() στο σημείο 6 της Εικόνα 7.61 διαγράφει τον πίνακα στην περίπτωση που αυτός υπήρχε και τον δημιουργεί ξανά από την αρχή, αυτό αποτελεί μία καλή τακτική για να μην υπάρχουν «σκουπίδια» που μπορεί να είχαν απομείνει από παλιότερες εκτελέσεις της εφαρμογής. Αφού δημιουργηθεί ο πίνακας contacts καλείται η μέθοδος onCreate() για τη δημιουργία της ΒΔ (Εικόνα 7.61, σημείο 7).

Οι επόμενες μέθοδοι χρησιμοποιούνται για τη διαχείριση των βασικότερων λειτουργιών της Εισαγωγής, της Εξαγωγής-Ανάγνωσης, της Ενημέρωσης και της Διαγραφής της ΒΔ.

Η εισαγωγή των δεδομένων στον πίνακα γίνεται με την χρήση της μεθόδου addContact() η οποία δέχεται ως παράμετρο ένα αντικείμενο τύπου ChatContact το οποίο περιέχει την εγγραφή ενός μηνύματος που έχει αποσταλεί ή ληφθεί (Εικόνα 7.62, σημείο 8). Καλείται η μέθοδος getWritableDatabase για την παραχώρηση του δικαιώματος εγγραφής στη βάση db. Τα δεδομένα που περιέχει η μεταβλητή ChatContact αποθηκεύονται στην μεταβλητή values και με την μέθοδο insert της SQL καταχωρείτε η εγγραφή με τις τιμές που έχουν αποθηκευτεί στη μεταβλητή values στη ΒΔ db. Αφού εισαχθούν τα δεδομένα στη ΒΔ καλείται η μέθοδος close για να κλίσει τη σύνδεση με τη βάση.

```
/**
 * All CRUD(Create, Read, Update, Delete) Operations
 */

// Adding new contact
void addContact(ChatContact contact) {
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(USERNAME, contact.getUsername()); // Contact USERNAME
    values.put(DATE, contact.getDate()); // Contact DATE
    values.put(TIME, contact.getTime()); // Contact TIME
    values.put(MESSAGE, contact.getMessage()); // Contact MESSAGE

    // Inserting Row
    db.insert(TABLE_CONTACTS, null, values);
    db.close(); // Closing database connection
}
```

8

Εικόνα 7.62 Ο κώδικα της κλάσης DatabaseHandler (σημείο 8)

Η συνάρτηση getContact() αναλαμβάνει την ανάκτηση των δεδομένων για ανάγνωση από τη ΒΔ, κάνει την αναζήτηση και ανακτά μία εγγραφή από τον πίνακα contact κάθε φορά που θα κληθεί. Η συνάρτηση getContact() δέχεται την παράμετρο id του κωδικού της εγγραφής που αναζητείται (Εικόνα 7.63, σημείο 9). Καλείται η μέθοδος getReadableDatabase για την παραχώρηση στην εφαρμογή του δικαιώματος ανάγνωσης των δεδομένων που είναι αποθηκευμένα στη ΒΔ. Γίνεται η αναζήτηση της εγγραφής με βάση το κλειδί KEY_ID που έχει περάσει ως παράμετρος (id) κατά την κλήση της getContact και εντοπίζεται η θέση της εγγραφής στη ΒΔ db. Η εγγραφή αποθηκεύεται στη τοπική μεταβλητή contact η οποία επιστρέφεται από τη συνάρτηση της getContact στην εφαρμογή (Εικόνα 7.63, σημείο 10).

```
// Getting single contact
ChatContact getContact(int id) {
    SQLiteDatabase db = this.getReadableDatabase();  9

    Cursor cursor = db.query(TABLE_CONTACTS, new String[] { KEY_ID,
            USERNAME, DATE, TIME, MESSAGE }, KEY_ID + "=?",
            new String[] { String.valueOf(id) }, null, null, null, null);
    if (cursor != null)
        cursor.moveToFirst();

    ChatContact contact = new ChatContact(Integer.parseInt(cursor.getString(0)),
            cursor.getString(1), cursor.getString(2), cursor.getString(3), cursor.getString(4));
    // return contact
    return contact;
}  10
```

Εικόνα 7.63 Ο κώδικάς της κλάσης DatabaseHandler (σημεία 9-10)

Η συνάρτηση getAllContacts() επιστρέφει μια λίστα η οποία περιέχει το σύνολο των εγγραφών που έχουν αποθηκευτεί στη ΒΔ (Εικόνα 7.64, σημείο 11). Η ανάκτηση των εγγραφών γίνεται με την εντολή SELECT της SQL, ο χαρακτήρας μπαλαντέρ * χρησιμοποιείται για να διαβαστούν όλες οι εγγραφές από τον πίνακα TABLE_CONTACTS της βάσης db. Η ΒΔ παραχωρεί το δικαίωμα εγγραφής, getWritableDatabase. Το αποτέλεσμα της εκτέλεσης της εντολής SELECT αποθηκεύεται με τη μορφή των εγγραφών rawQuery στην μεταβλητή cursor. Στη συνέχεια η if ελέγχει για να προσπελαστούν όλα τα δεδομένα που περιέχει η cursor και με την φωλιασμένη συνθήκη while αποθηκεύει κάθε εγγραφή contact, στη λίστα contactList. Η συνάρτηση getAllContacts επιστρέφει τη λίστα contactList η οποία περιέχει το σύνολο των εγγραφών που έχουν αποθηκευτεί στη βάση db.

```
// Getting All Contacts. Display a list of all contacts.
 public List<ChatContact> getAllContacts() {
    List<ChatContact> contactList = new ArrayList<ChatContact>();
    // Select All Query
    String selectQuery = "SELECT  * FROM " + TABLE_CONTACTS;

    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery(selectQuery, null);

    // looping through all rows and adding to list
    // topothetei i sql gia na ta do ero
    if (cursor.moveToFirst()) {
        do {
            ChatContact contact = new ChatContact();
            contact.setID(Integer.parseInt(cursor.getString(0)));
            contact.setUsername(cursor.getString(1));
            contact.setDate(cursor.getString(2));
            contact.setTime(cursor.getString(3));
            contact.setMessage(cursor.getString(4));

            // Adding contact to list
            contactList.add(contact);
        } while (cursor.moveToNext());
    }

    // return contact list
    return contactList;
}
```

11

Εικόνα 7.64 Η μέθοδος getAllContacts() (σημείο 11)

Στο σημείο 12 της Εικόνα 7.65 η μέθοδος updateContact() αναλαμβάνει την ενημέρωση μιας εγγραφής στη ΒΔ. Η συνάρτηση updateContact() δέχεται ως παράμετρο εισόδου μία εγγραφή contact με τις συνομιλίες. Εκχωρεί το δικαίωμα της εγγραφής στη ΒΔ με την εντολή

SQLiteDatabase db = **this**.getWritableDatabase();

για να μπορεί να μεταβάλλει την ΒΔ. Δημιουργεί το αντικείμενο values τύπου ContactValues στο οποίο αποθηκεύονται οι τιμές της εγγραφής contact εκτός της τιμής του κλειδιού και ξεκινάει την ενημέρωση της εγγραφής που έχει επιλέγει συμφώνα με την τιμή της μεταβλητής id που έχει λάβει η contact.getID().

```
    // Updating single contact
    public int updateContact(ChatContact contact) {
        SQLiteDatabase db = this.getWritableDatabase();

        ContentValues values = new ContentValues();
        values.put(USERNAME, contact.getUsername());
        values.put(DATE, contact.getDate());
        values.put(TIME, contact.getTime());
        values.put(MESSAGE, contact.getMessage());

        // updating row
        return db.update(TABLE_CONTACTS, values, KEY_ID + " = ?",
                new String[] { String.valueOf(contact.getID()) });
    }
```

12

Εικόνα 7.65 η μέθοδος updateContact()

Η διαγραφή μιας εγγραφής από τη ΒΔ πραγματοποιείται σημείο 13 της Εικόνα 7.66 με την κλήση της μεθόδου deleteContact(). Αφού διαγραφεί η επιλεγείσα εγγραφή από τη ΒΔ θα πρέπει να κλείσει η σύνδεση με τη βάση.

```
// Deleting single contact
public void deleteContact(ChatContact contact) {
    SQLiteDatabase db = this.getWritableDatabase();
    db.delete(TABLE_CONTACTS, KEY_ID + " = ?",
            new String[] { String.valueOf(contact.getID()) });
    db.close();
}
```

13

Εικόνα 7.66 Η μέθοδος deleteContact()

Η καταμέτρηση του πλήθους των εγγράφων που βρίσκονται στη βάση db πραγματοποιείται με την κλήση της getContactsCount().

```
// Getting contacts Count
public int getContactsCount() {
    String countQuery = "SELECT  * FROM " + TABLE_CONTACTS;
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.rawQuery(countQuery, null);
    cursor.close();

    // return count
    return cursor.getCount();
}
```

14

Εικόνα 7.67 Η μέθοδος getContactsCount()

Η εντολή String countQuery = "SELECT * FROM " + TABLE_CONTACTS; αποθηκεύει στην μεταβλητή τύπου string countQuery όλες τις εγγραφές του πίνακα. Εκχωρείται το δικαίωμα της ανάγνωσης της ΒΔ με την εντολή

SQLiteDatabase db = **this**.getReadableDatabase();

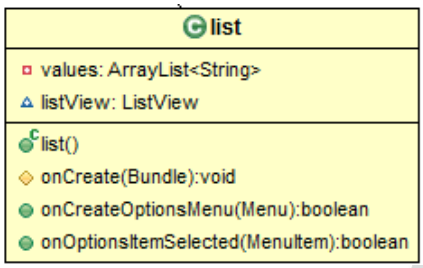
για να επιτραπεί η ανάγνωση από τη ΒΔ. Η μεταβλητή cursor που είναι τύπου Cursor αναλαμβάνει την αποθήκευση των εγγραφών μέσω της εντολής

Cursor cursor = db.rawQuery(countQuery, **null**);

Αμέσως μετά κλείνει η cursor με την εντολή cursor.close(); για να αποδεσμευτούν όλοι οι πόροι που χρησιμοποιούσε μέχρι τότε. Η καταμέτρηση των εγγραφών που έχουν αποθηκευτεί στην cursor γίνεται από την μέθοδο getCount. Το νούμερο που προέκυψε από την καταμέτρηση των εγγραφών επιστρέφεται από τη συνάρτηση getContactsCount() στο πρόγραμμα. [54], [55], [57], [58]

7.5.6 Η κλάση list

Η κλάση αυτή αναλαμβάνει την προβολή στην οθόνη της λίστας των συζητήσεων που έχουν αποθηκευτεί στη ΒΔ. Κάνει extends την Activity γιατί από την αλληλεπίδραση με τον χρήστη αναμένεται να προκύψει κάποια δραστηριότητα.



Εικόνα 7.68 Η δομή της κλάσης list

Η δομή της κλάσης list που απεικονίζεται στην προηγούμενη εικόνα περιέχει τις μεθόδους:

- τον κατασκευαστή της list
- την onCreate(Bundle)
- την onCreateOptionsMenu(Menu)
- την onOprionItemSelected(MenuItem)

Για την προβολή της λίστας των συζητήσεων στην οθόνη απαιτείται η δημιουργία ενός αντικείμενου adapter τύπου arrayAdapter (Εικόνα 7.69, σημείο 1).

```
import java.util.ArrayList;
public class list extends Activity {
    private arrayAdapter adapter;
    private ArrayList<String> values = new ArrayList<String>();
    ListView listView;
```

1

Εικόνα 7.69 Ο κώδικας της κλάσης list (σημείο 1)

Στην Εικόνα 7.70 στο σημείο 2 δημιουργείται η μεταβλητή values η οποία είναι τύπου ArrayList και θα χρησιμοποιηθεί αργότερα μέσα στο κώδικα για να κρατήσει τα δεδομένα που θα ανακτηθούν από τη ΒΔ. Η listView δημιουργείται για τη προβολή της λίστας των επαφών κάνοντας κλήση του ListAdapter. Στην onCreate η οποία δέχεται ως παράμετρο εισόδου την savedInstanceState τύπου Bundle αρχικοποιείται η δραστηριότητα. Στο σημείο αυτό είναι πολύ σημαντικό να γίνει η κλήση της setContentView(R.layout.list) η οποία αναλαμβάνει μέσω της κλάσης R.java να κάνει τη σύνδεση του κώδικα με το αρχείο των γραφικών list.xml που ορίζει το UI (User Interface). Αμέσως μετά με την κλήση της findViewById()

επιτυγχάνεται η ανάκτηση των widgets που θα πρέπει να αλληλεπιδρούν με προγραμματισμό σε αυτή τη διεπαφή (UI). Ο πίνακας conversations (Εικόνα 7.70, σημείο 3) γεμίζει από τη από τη ΒΔ με όλες τις συζητήσεις που έχουν αποθηκευτεί. Στο βρόγχο for που ακολουθεί αποθηκεύονται τα ονόματα των χρηστών που υπήρχαν μέσα στη ΒΔ. Και εν συνεχεία, στο σημείο 4 της Εικόνα 7.70, πραγματοποιείται η προσθήκη των ονομάτων στη λίστα και η προβολή τους στην οθόνη με την ListView.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.list);

    listView = (ListView) findViewById(R.id.in);
    final DatabaseHandlerConv db = new DatabaseHandlerConv(this);
    List<Conversation> conversations = db.getAllConversations();

    for (Conversation cv : conversations) {
        values.add(cv.getUsername()+"\n");
    }

    adapter = new arrayAdapter(list.this, values);
    listView.setAdapter(adapter);

    listView.setOnItemClickListener(new OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view,
           int position, long id) {

          String name0 = listView.getItemAtPosition(position).toString();
          Intent intent1 = new Intent();
          intent1.setClass(list.this, ConvList.class);
          intent1.putExtra("position", name0);
          startActivity(intent1);
        }
    });
```

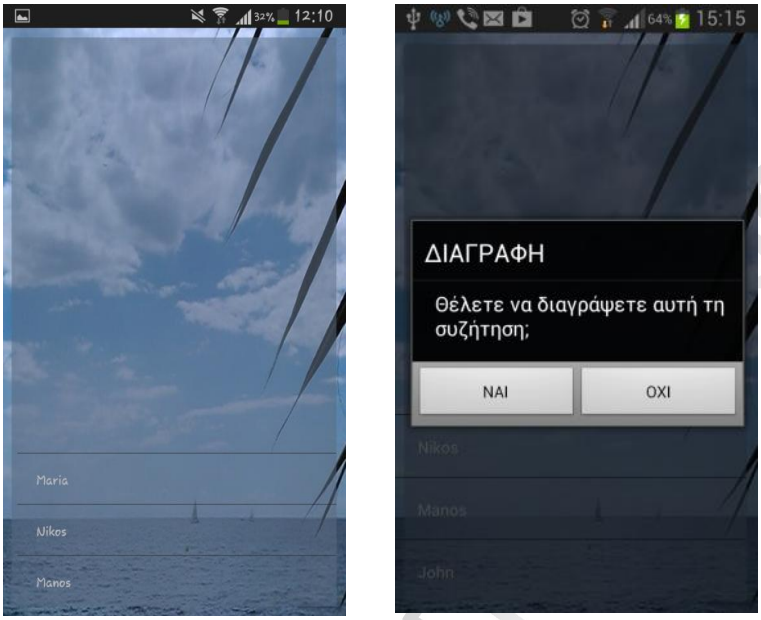
2 3 4 5

Εικόνα 7.70 Ο κώδικας της κλάσης list (σημεία 2-5)

Αφού εμφανιστεί η λίστα με τα ονόματα των επαφών, ο χρήστης έχει δύο δυνατότητες η πρώτη είναι να επιλέξει την προβολή κάποιας συζήτησης που είχε στο παρελθόν με κάποιο άλλο χρήστη και που είναι αποθηκευμένη στη ΒΔ πατώντας μία φορά πάνω στο όνομα. Η δραστηριότητα αυτή υλοποιείται (Εικόνα 7.70, σημείο 5) με την κλήση της setOnItemClickListener. Η εφαρμογή περιμένει μέχρι να επιλέξει ο χρήστης κάποιο όνομα από τη λίστα και «αφουγκράζεται» μέχρι να δοθεί η επιλογή. Τότε ξεκινάει η onItemClick η οποία λαμβάνει τη θέση –position- στην οποία έχει πατήσει ο χρήστης και με την getItemAtPosition γίνεται η επιλογή της συζήτησης ενώ η εμφάνιση της γίνεται από την intent1.setClass(list.this, ConvList.class) με την κλήση της ConvList.class.

Η δεύτερη επιλογή επιτρέπει τη διαγραφή της συνομιλίας από τον χρήστη πατώντας παρατεταμένα σε κάποιο από τα ονόματα. Τότε εμφανίζεται ένα παράθυρο

διαλόγου που ρωτά τον χρήστη αν θέλει να διαγράψει ή όχι τη συνομιλίας αυτή, (Εικόνα 7.71, β).



Εικόνα 7.71 α) Η λίστα με τις συζητήσεις και β) Το παράθυρο διαλόγου για τη διαγραφή των συζητήσεων

Το παρατεταμένο πάτημα του χρήστη πάνω στην οθόνη για την επιλογή της συζήτησης ενεργοποιεί με την κλήση της setOnItemLongClickListener έναν ακροατή συμβάντων. Μέχρι τότε η εφαρμογή βρίσκεται σε ετοιμότητα και περιμένει ένα άγγιγμα της οθόνης από το χρήστη.

Η εμφάνιση του μηνύματος της διαγραφής της συζήτησης γίνεται στο σημείο 6 της Εικόνα 7.72. Η κλάση AlertDialog.Builder χρησιμοποιείται για τη δόμηση του παραθύρου διαλόγου (τίτλος, εμφάνιση μηνύματος, κουμπιά επιλογών). Αυτό το παράθυρο διαλόγου οδηγεί σε δύο επιλογές (μέχρι τρία κουμπιά επιλογών μπορούν να χρησιμοποιηθούν σε ένα παράθυρο διαλόγου).

Στην περίπτωση της θετικής απόκρισης του χρήστη υλοποιείται ο κώδικας που περικλείεται μέσα στην setPositiveButton. Αρχικά η idpos λαμβάνει τη θέση, position, της συζήτησης που έχει επιλεγεί από το χρήστη με το παρατεταμένο κλικ. Η θέση αυτή προσαυξάνεται κατά ένα, position+1, γιατί η τιμή που επιστρέφεται ξεκινά από το μηδέν, στην περίπτωση που δεν γίνει η προσαύξηση αυτή η όποια τροποποίηση θα γίνει στη προηγούμενη εγγραφή. Στο σημείο 7 της Εικόνα 7.72 γίνεται ανάκτηση της ΒΔ db3 που διατηρεί τις συζητήσεις, στη συνέχεια εντοπίζεται η θέση της συζήτησης στη ΒΔ και η συζήτηση διαγράφεται από τη βάση. Με τον ίδιο

τρόπο γίνεται η διαγραφή της επαφής από τη ΒΔ db4 των επαφών (Εικόνα 7.72, σημείο 8). Εφόσον η συζήτηση και η επαφή διαγράφουν από τις αντίστοιχες ΒΔ, γίνεται ξανά η ανάκτηση της ενημερωμένης ΒΔ των συζητήσεων (Εικόνα 7.72, σημείο 9) προκειμένου να προβληθεί η νέα λίστα των συζητήσεων στην οθόνη.

Αν ο χρήστης επιλέξει το κουμπί «ΟΧΙ» τότε η ροή του προγράμματος μεταφέρεται στο σημείο 10 της Εικόνα 7.72 και εκτελείται ο κώδικας που περιέχεται στη setNegativeButton. Τέλος (Εικόνα 7.72, σημείο 11) γίνεται η εμφάνιση του παράθυρου διαλόγου της AlertDialog από την show(), η κλήση της συνάρτησης setOnItemLongClickListener τερματίζεται με την επιστροφή της boolean τιμής true.

```
listView.setOnItemLongClickListener(new OnItemLongClickListener() {
      @Override
      public boolean onItemLongClick(AdapterView<?> parent, View view,
         final int position, long id) {
          new AlertDialog.Builder(list.this)
            .setTitle("ΔΙΑΓΡΑΦΗ")                                          6
            .setMessage("Θέλετε να διαγράψετε αυτή τη συζήτηση;")
            .setPositiveButton("ΝΑΙ", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int which) {
                      int idpos = position+1;
                      String names = null;
                      final DatabaseHandlerConv db3 = new DatabaseHandlerConv(list.this);   7
                      List<Conversation> conversations = db3.getAllConversations();

                      for (Conversation cv : conversations) {
                          if (cv.getID() == idpos ) {
                              names = cv.getUsername();
                              db3.deleteConversation(cv);
                          }
                      }

                      final DatabaseHandler db4 = new DatabaseHandler(list.this);   8
                      List<ChatContact> contact = db4.getAllContacts();

                      for (ChatContact cn1 : contact) {
                          if (cn1.getUsername().equals(names) ) {
                               db4.deleteContact(cn1);
                          }
                      }

                      final DatabaseHandlerConv db9 = new DatabaseHandlerConv(list.this);   9
                      List<Conversation> conversations9 = db9.getAllConversations();
                      values.clear();

                      for (Conversation cv : conversations9) {
                          values.add(cv.getUsername()+"\n");
                      }

                      adapter = new arrayAdapter(list.this, values);
                      listView.setAdapter(adapter);
                }
            })
            .setNegativeButton("ΟΧΙ", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int which) {          10
                    // do nothing
                }
            })
            .show();
          return true;                                                    11
      }
});
```

Εικόνα 7.72 Η setOnItemLongClickListener (σημεία 6-11)

7.5.6.1 Το option menu

Στην Εικόνα 7.73 στο σημείο 12 δημιουργείται το option menu που εμφανίζεται με το άγγιγμα του menu button της συσκευής. Το option menu χρησιμοποιείται όταν πρέπει να συμπεριληφθούν ενέργειες και επιλογές οι οποίες είναι σχετικές με το πλαίσιο τρέχουσας δραστηριότητας (π.χ. όπως η αναζήτηση, το ηλεκτρονικό ταχυδρομείο κλπ). Η εμφάνιση των επιλογών του option menu στην οθόνη εξαρτάται από την έκδοση του Λ.Σ. του Android στην οποία έγινε η ανάπτυξη της εφαρμογής. Αν η εφαρμογή έχει δημιουργηθεί για έκδοση του Λ.Σ. Android 2.3.x (API level 10) ή παλιότερη, τα περιεχόμενα του option menu εμφανίζονται στο κάτω μέρος της οθόνης όταν ο χρήστης πιέζει το menu button της συσκευής. Η εφαρμογή που εξετάζουμε έχει δημιουργηθεί με την έκδοση Android 2.0.1 (android:minSdkVersion="6") για να μπορεί να χρησιμοποιηθεί και σε παλιότερες συσκευές, επομένως το option menu εμφανίζεται στο κάτω μέρος της συσκευής. Αν η εφαρμογή έχει δημιουργηθεί για του λειτουργικού Android 3.0 (API level 11) ή μεταγενέστερη έκδοση, τα περιεχόμενα του option menu εμφανίζονται στο action bar στο πάνω μέρος της οθόνης.

Για να δημιουργηθεί το option menu και να οριστούν οι επιλογές του καλείται η κλάση onCreateOptionsMenu. Η σύνδεση και η εμφάνιση του γραφικού μέρους του μενού επιτυγχάνεται με την inflater.inflate(R.menu.menu, menu). Η λεπτομερέστερη παρουσίαση του xml αρχείου του μενού γίνεται στην παράγραφο 7.5.6.3 "Η ταξινόμηση χρονικά ή αλφαβητικά (menu.xml)".

Η διαχείριση του event το οποίο έχει επιλεγεί από το μενού γίνεται αμέσως μετά από την onOptionsItemSelected(MenuItem item).

Η ροή του κώδικα οδηγείται σε ένα βρόγχο case. Αν ο χρήστης επιλέξει την ταξινόμηση «Αλφαβητικά» τότε εκτελείται το πρώτο μέρος του βρόγχου case R.id.alpha, (Εικόνα 7.73, σημείο 13) και γίνεται η ανάκτηση όλων των συζητήσεων από τη ΒΔ. Στο βρόγχο for εντοπίζονται τα ονόματα των επαφών και αποθηκεύονται στη μεταβλητή τύπου πίνακα, value. Η ταξινόμηση των στοιχείων του πίνακα value γίνεται από την Collections.sort που περιλαμβάνεται στην java.util.Collections. Η προβολή των ταξινομημένων επαφών επιτυγχάνεται με την δημιουργία ενός

αντικειμένου arrayAdapter(list.this, values) και με την κλήση της listView.setAdapter(adapter). Μετά την προβολή της λίστας ο βρόγχος case τερματίζεται επιστρέφοντας την τιμή true.

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.alpha:
            final DatabaseHandlerConv db4 = new DatabaseHandlerConv(list.this);
            List<Conversation> conversations4 = db4.getAllConversations();
            values.clear();

            for (Conversation cv : conversations4) {
                values.add(cv.getUsername()+"\n");
            }

            Collections.sort(values);
            adapter = new arrayAdapter(list.this, values);
            listView.setAdapter(adapter);
            return true;

        case R.id.time:
            final DatabaseHandlerConv db9 = new DatabaseHandlerConv(list.this);
            List<Conversation> conversations9 = db9.getAllConversations();
            values.clear();

            for (Conversation cv : conversations9) {
                values.add(cv.getUsername()+"\n");
            }

            adapter = new arrayAdapter(list.this, values);
            listView.setAdapter(adapter);
            return true;
        }
        return false;
}
```

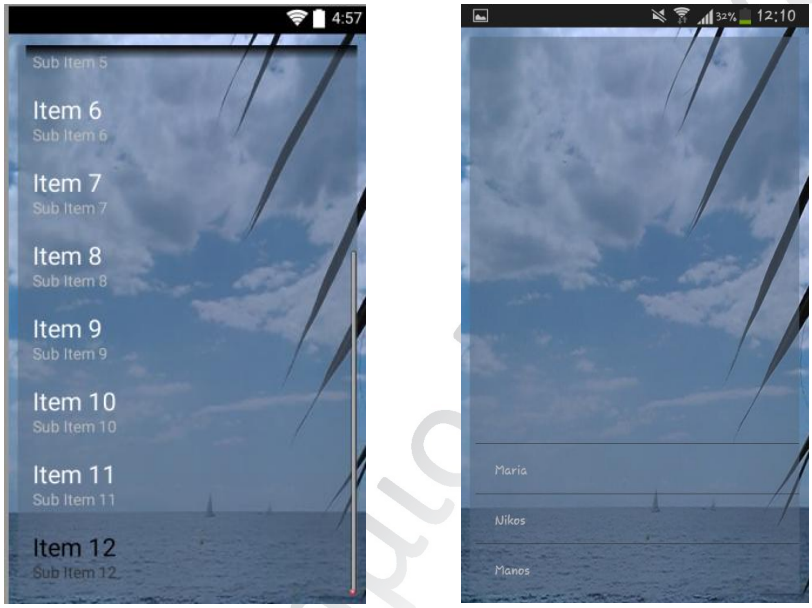
12 13 14

Εικόνα 7.73 Η onCreateOptionsMenu και η onOptionsItemSelected(MenuItem item)

Αν η επιλογή του χρήστη στο μενού είναι «Χρονικά», δηλαδή ζητήσει την χρονική ταξινόμηση των συζητήσεων τότε εκτελείται το δεύτερο μέρος του βρόγχου της case R.id.time (Εικόνα 7.73,σημείο 14). Οι εργασίες που εκτελούνται σε αυτό το μέρος του βρόγχου είναι όμοιες με του προηγούμενου, με τη μόνη διαφορά ότι εδώ παραλείπεται η ταξινόμηση των στοιχείων του πίνακα value από την Collections.sort και οι επαφές εμφανίζονται στη λίστα ακριβώς με τη χρονική σειρά που αποθηκευτήκαν στη ΒΔ. [56]

7.5.6.2 Η λίστα με τις συζητήσεις (list.xml)

Οι περισσότερες οθόνες διεπαφής χρήστη των Android εφαρμογών ορίζονται χρησιμοποιώντας τα ειδικά μορφοποιημένα αρχεία XML. Τα αρχεία διατάξεων XML μπορούν να θεωρηθούν σαν ένας ειδικός τύπος πόρου. Χρησιμοποιούνται γενικά για να ορίσουν πως θα εμφανίζεται ένα τμήμα της οθόνης ή ολόκληρη η οθόνη. Τα αρχεία των πόρων διάταξης αποθηκεύονται με ιεραρχικό τρόπο μέσα στο φάκελο /res/layout.



Εικόνα 7.74 α)Σχεδιασμός γραφικού με το eclipse β) η τελική μορφή του list.xml

Η προβολή της λίστας των συζητήσεων επιτυγχάνεται με τη «συνεργασία» των αντίστοιχων list αρχείων σε κώδικα java και xml. Στην παράγραφο αυτή παρατίθεται ο κώδικας του list.xml αρχείου πόρων καθώς και το γραφικό layout όπως αυτό σχεδιάζεται από το eclipse.

Ο κώδικας του αρχείου list.xml περιέχει το μηχανισμό διάταξης LinearLayout, που χρησιμοποιείται ως υποδοχέας για όλους τους μηχανισμούς ελέγχου της διεπαφής του χρήστη. Στην περίπτωση αυτής της διεπαφής ορίζεται ένας μόνο μηχανισμός ελέγχου ListView για την προβολή της λίστας.

Εδώ ορίζεται το background της διεπαφής που στη συγκεκριμένη εφαρμογή είναι ένας άλλος πόρος, το αρχείο φωτογραφίας back.png, που βρίσκεται στο φάκελο BluetoothChat\res\drawable. Στο xml αρχείο το background αναφέρεται με το όνομα @drawable/back και ορίζεται μέσα στη ListView. Επίσης προβλέπεται η δημιουργία

μπάρας κύλισης transcriptMode="alwaysScroll" δίπλα στη λίστα σε περίπτωση που οι εγγραφές είναι περισσότερες απ' αυτές που μπορούν να χωρέσουν στην οθόνη.

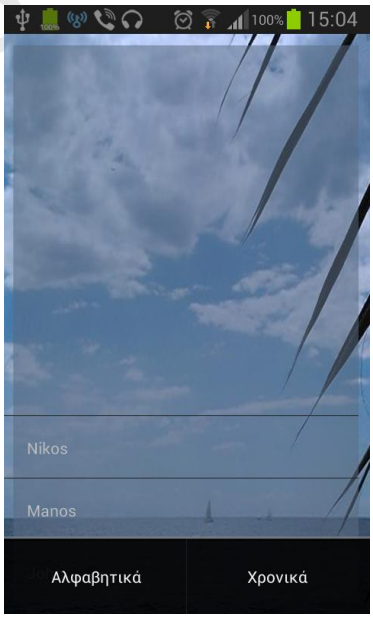
list.xml

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/back" >

    <ListView
        android:id="@+id/in"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:stackFromBottom="true"
        android:layout_marginBottom="10dp"
        android:layout_marginRight="10dp"
        android:layout_marginTop="10dp"
        android:background="@drawable/but"
        android:transcriptMode="alwaysScroll" >
    </ListView>

</LinearLayout>
```

Εικόνα 7.75 Ο κωδικας του list.xml

7.5.6.3 Η ταξινόμηση χρονικά ή αλφαβητικά (menu.xml)

Το αρχείο πόρων menu.xml αναλαμβάνει το σχεδιαστικό κομμάτι του option menu που εμφανίζεται στο κάτω μέρος της οθόνης της συσκευής, για την διαχείριση της αλφαβητικής και της χρονικής ταξινόμησης της λίστας των συζητήσεων.



Εικόνα 7.76 Το γραφικό του option menu

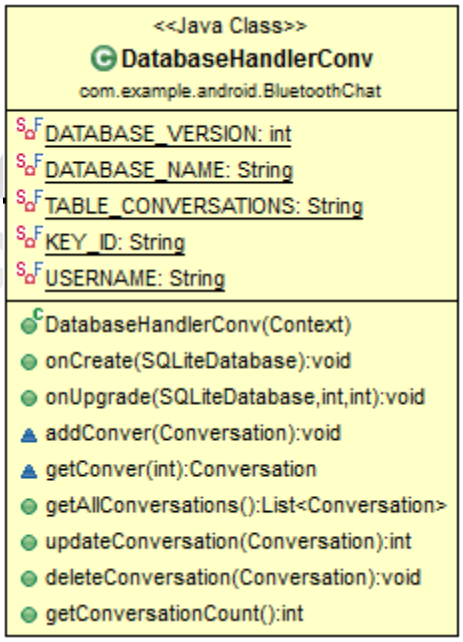
Ο κώδικας του αρχείου menu.xml αναλαμβάνει τη δημιουργία του option menu. Το σημαντικότερο σημείο του κώδικα είναι η δημιουργία των επιλογών του μενού ως στοιχείων item που ορίζονται με ένα μοναδικό αναγνωριστικό id και ένα όνομα τύπου string.

```
<?xml version="1.0" encoding="utf-8"?>

<menu xmlns:android="http://schemas.android.com/apk/res/android">
     <item android:id="@+id/alpha"
           android:title="@string/alpha" />
     <item android:id="@+id/time"
           android:title="@string/time" />
</menu>
```

Εικόνα 7.77 Ο κωδικας xml του γραφικού μέρους του option menu

7.5.7 Η κλάση DatabaseHandleConv

Αυτή η κλάση είναι υπεύθυνη για την δημιουργία και τη διαχείριση της ΒΔ conversationManager (Εικόνα 7.79, σημείο 1) η οποία περιέχει τον πίνακα conversation. Η δομή της, η κατασκευή της και η λειτουργία της είναι παρόμοια με αυτές της κλάσης DatabaseHandler η περιγραφή της οποίας βρίσκεται στην παράγραφο 7.5.5.



Εικόνα 7.78 Η δομή της κλάσης DatabaseHandleConv

Η κλάση DatabaseHandlerConv όπως παρουσιάζεται στην παραπάνω εικόνα περιέχει τις μεθόδους:

- την DatabaseHandlerConv τον κατασκευαστή της, constructor
- την onCreate(SQLiteDatabase), για την δήλωση της ΒΔ κατά την φάση της δημιουργίας της εφαρμογής
- την onUpgrade(SQLiteDatabase,int,int)
- την addConver(Conversation) για την εισαγωγή μιας εγγραφής στη ΒΔ
- την getConver(int) για την ανάκτηση με τη χρήση του πρωτεύοντος κλειδιού _id μιας εγγραφής
- την getAllConversations() για την ανάκτηση όλων των εγγραφών
- την updateConversation(Conversation) για την ενημέρωση μιας εγγραφής
- την deleteConversation(Conversation) για την διαγραφή μιας εγγραφής
- την getConversationCount() για την ανάκτηση του κωδικού του πρωτεύοντος κλειδιού

```
public class DatabaseHandlerConv extends SQLiteOpenHelper {
    // All Static variables
    // Database Version
    private static final int DATABASE_VERSION = 1;

    // Database Name
    private static final String DATABASE_NAME = "conversationManager";

    // Contacts table name
    private static final String TABLE_CONVERSATIONS = "conversations";

    // Contacts Table Columns names
    private static final String KEY_ID = "id";
    private static final String USERNAME = "username";

    public DatabaseHandlerConv(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    // Creating Tables
    @Override
    public void onCreate(SQLiteDatabase dbConv) {
        String CREATE_CONVERSATIONS_TABLE = "CREATE TABLE " + TABLE_CONVERSATIONS
                + "(" + KEY_ID + " INTEGER PRIMARY KEY," + USERNAME + " TEXT"
                + ")";
        dbConv.execSQL(CREATE_CONVERSATIONS_TABLE);
    }

    // Upgrading database
    @Override
    public void onUpgrade(SQLiteDatabase dbConv, int oldVersion, int newVersion) {
        // Drop older table if existed
        dbConv.execSQL("DROP TABLE IF EXISTS " + TABLE_CONVERSATIONS);
        // Create tables again
        onCreate(dbConv);
    }
```

1

2

Εικόνα 7.79 Ο κώδικας της κλάσης DatabaseHandleConv (σημεία 1-3)

Η λειτουργία της είναι παρόμοια με αυτή της κλάσης DatabaseHandler, διαφοροποιείται μόνο ως προς το πλήθος των πεδίων του πίνακα conversations που δημιουργεί.

Η βάση δημιουργεί τον πίνακα conversations ο οποίος περιλαμβάνει δύο στήλες. Την υποχρεωτική στήλη του μοναδικού πρωτεύοντος κλειδιού (KEY_ID) βάση του οποίου γίνονται οι αναζητήσεις και οι ταξινομήσεις των οντοτήτων στη ΒΔ και την στήλη USERNAME στην οποία αποθηκεύονται τα ονόματα των χρηστών. Όμοια με την κλάση DatabaseHandler κατασκευάζεται (Εικόνα 7.79, σημείο 2) η μέθοδος onUpgrade() για τη δημιουργία της ΒΔ.

```
// Adding new conversation
void addConver(Conversation conversation) {
    SQLiteDatabase dbConv = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(USERNAME, conversation.getUsername()); // conversation USERNAME

    // Inserting Row
    dbConv.insert(TABLE_CONVERSATIONS, null, values);
    dbConv.close(); // Closing database connection
}

// Getting single conversation
Conversation getConver(int id) {
    SQLiteDatabase dbConv = this.getReadableDatabase();

    Cursor cursor = dbConv.query(TABLE_CONVERSATIONS, new String[] { KEY_ID,
            USERNAME
            }, KEY_ID + "=?",
            new String[] { String.valueOf(id) }, null, null, null, null);
    if (cursor != null)
        cursor.moveToFirst();

    Conversation conver = new Conversation(Integer.parseInt(cursor.getString(0)),
            cursor.getString(1)
            );
    // return conversation
    return conver;
}

// Getting All conversation
public List<Conversation> getAllConversations() {
    List<Conversation> conversationList = new ArrayList<Conversation>();
    // Select All Query
    String selectQuery = "SELECT  * FROM " + TABLE_CONVERSATIONS;

    SQLiteDatabase dbConv = this.getWritableDatabase();
    Cursor cursor = dbConv.rawQuery(selectQuery, null);

    // looping through all rows and adding to list
    if (cursor.moveToFirst()) {
        do {
            Conversation conver = new Conversation();
            conver.setID(Integer.parseInt(cursor.getString(0)));
            conver.setUsername(cursor.getString(1));
            // Adding conversation to list
            conversationList.add(conver);
        } while (cursor.moveToNext());
    }

    // return conversation list
    return conversationList;
}
```

Εικόνα 7.80 Ο κώδικας της DatabaseHandleConv

Οι μέθοδοι addConver(Conversation), getConver(int) και getAllConversations() χρησιμοποιούνται για τη διαχείριση των βασικότερων λειτουργιών της εισαγωγής, της εξαγωγής-ανάγνωσης, της ενημέρωσης και της διαγραφής της ΒΔ.

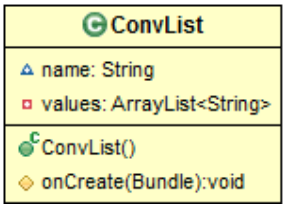
```
// Updating single conversation
public int updateConversation(Conversation conver) {
    SQLiteDatabase dbConv = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(USERNAME, conver.getUsername());

    // updating row
    return dbConv.update(TABLE_CONVERSATIONS, values, KEY_ID + " = ?",
            new String[] { String.valueOf(conver.getID()) });
}

// Deleting single conversation
public void deleteConversation(Conversation conver) {
    SQLiteDatabase db = this.getWritableDatabase();
    db.delete(TABLE_CONVERSATIONS, KEY_ID + " = ?",
            new String[] { String.valueOf(conver.getID()) });
    db.close();
}

// Getting conversation Count
public int getConversationCount() {
    String countQuery = "SELECT  * FROM " + TABLE_CONVERSATIONS;
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.rawQuery(countQuery, null);
    cursor.close();
    // return count
    return cursor.getCount();
}
```

Εικόνα 7.81 Ο κώδικας της DatabaseHandleConv

Τέλος στην Εικόνα 7.81 διακρίνουμε τις μεθόδους: updateConversation deleteConversation(Conversation) και getConversationCount() της κλάσης DatabaseHandleConv.

7.5.8 Η κλάση ConvList

Αυτή η κλάση αναλαμβάνει την προβολή της λίστας των συνομιλιών των χρηστών στην οθόνη της συσκευής. Η δομή της κλάσης αυτής παρουσιάζεται στην Εικόνα 7.82 και περιλαμβάνει τις μεταβλητές name και values που είναι τύπου string και το διάνυσμα της ArrayList.



Εικόνα 7.82 Η δομή της κλάσης ConvList

Στο τμήμα δήλωσης των μεθόδων της κλάσης διακρίνονται ο constructor ConvList και η μέθοδος onCreate(Bundle).

Η κλάση ConvList αναλαμβάνει την προβολή της λίστας των συνομιλιών των χρηστών στην οθόνη της συσκευής. Αρχικά δημιουργούνται (Εικόνα 7.83, σημείο 1) ένα αντικείμενο adapter τύπου arrayAdapter για την εμφάνιση της λίστας στην οθόνη και ένα αντικείμενο value, τύπου ArrayList το οποίο στην ουσία είναι ένας πίνακας στον οποίο αποθηκεύεται η λίστα των συνομιλιών που ανακτώνται από τη ΒΔ db1.

```
package com.example.android.BluetoothChat;

import java.util.ArrayList;
import java.util.List;
import android.app.Activity;
import android.os.Bundle;
import android.widget.ListView;

public class ConvList extends Activity {
    private arrayAdapter adapter;
    String name;
    private ArrayList<String> values = new ArrayList<String>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.convlist);

        ListView listView = (ListView) findViewById(R.id.in1);
        Bundle dat = this.getIntent().getExtras();
        name = dat.getString("position");
        final DatabaseHandler db1 = new DatabaseHandler(this);
        List<ChatContact> contact = db1.getAllContacts();

        for (ChatContact cn1 : contact) {
            if (name.contains(cn1.getUsername()) && (name.length() == (cn1.getUsername().length()+1))) {
              values.add(""+cn1.getMessage()+"\n"+cn1.getDate()+" "+cn1.getTime()+"\n");
            }
        }

        adapter = new arrayAdapter(ConvList.this,  values);
        listView.setAdapter(adapter);
    }
}
```

1 2 3 4 5

Εικόνα 7.83 Ο κώδικας της κλάσης ConvList

Στο σημείο 2 της παραπάνω εικόνας δημιουργείται η onCreate μέσα στην οποία καλείτε η setContentView(R.layout.convlist) για να γίνει η σύνδεση με το αρχείο σχεδίασης της διεπαφής convlist.xml που προβάλει την λίστα των συζητήσεων στην οθόνη. Η μέθοδος findViewById() (Εικόνα 7.83, σημείο 3) κάνει την ανάκτηση της λίστας με βάση το μοναδικό αναγνωριστικό (ID) της με τη χρήση του μηχανισμού ελέγχου ListView.

Στη μεταβλητή name αποθηκεύεται το όνομα της επαφής που αντιστοιχεί στη θέση position που είχε επιλέξει νωρίτερα ο χρήστης. Από την ΒΔ db1 ανακτώνται οι εγγραφές των συζητήσεων και αποθηκεύονται στη λίστα contact (Εικόνα 7.83,

σημείο 4). Στον βρόγχο for που ακολουθεί γίνεται η επιλογή των εγγραφών εκείνων που περιέχουν το όνομα name και αποθηκεύονται στην μεταβλητή value. Τέλος στο σημείο 5 της παραπάνω εικόνας του κώδικα δημιουργείται ένα αντικείμενο τύπου arrayAdapter για την προβολή στην οθόνη της λίστας των εγγραφών που καταχωρήθηκαν στην μεταβλητή value.

7.5.8.1 Η οθόνη με τις συζητήσεις ανά επαφή (convlist.xml)

Το αρχείο διάταξης πόρων convlist.xml περιέχει το σχεδιασμό της διεπαφής που εμφανίζει στην οθόνη μία συζήτηση.

Εικόνα 7.84 Η λίστα των συνομιλιών ανά επαφή

convlist.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/back" >

    <ListView android:id="@+id/in1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:stackFromBottom="true"
        android:paddingLeft="15dp"
        android:layout_marginRight="10dp"
        android:layout_marginTop="10dp"
        android:layout_marginBottom="10dp"
        android:background="@drawable/but"
        android:transcriptMode="alwaysScroll"
    />
</LinearLayout>
```

Εικόνα 7.85 Ο κώδικας της convlist.xml

Για το σχεδιασμό της οθόνης αυτής χρησιμοποιείται η γραμμική μορφή, LinearLayout μέσα στην οποία τοποθετείται η ListView που περιέχει τα δεδομένα που έχουν φορτωθεί με τη μεσολάβηση του arrayAdapter.

7.6 Σύνοψη

Μετά από την ολοκλήρωση της κωδικοποίησης, σειρά έχουν τα στάδια του ελέγχου και της τεκμηρίωσης για να είναι δυνατή η εμπορική διάθεση της εφαρμογής. Η εμπορική διάθεση της εφαρμογής δεν σημαίνει ότι η ενασχόληση με την εφαρμογή έχει τελειώσει. Το σύστημα επιδέχεται τη συνεχή πραγματοποίηση αλλαγών με σκοπό την ενσωμάτωση νέων δυνατοτήτων, την αναβάθμιση του και την αντιμετώπιση των καθημερινών προβλημάτων που ενδέχεται να προκύψουν κατά την πραγματική λειτουργία του ή που διέφυγαν κατά την διαδικασία της ανάπτυξης.

8 Συμπεράσματα

Η τεχνολογία του Bluetooth όπως έχει αναφερθεί στο πρώτο κεφάλαιο «Η προδιαγραφή του Bluetooth», λειτουργεί σε μια ζώνη συχνοτήτων εύρους 83.5MHz η οποία είναι παγκοσμίως αναγνωρισμένη για την ελεύθερη -χωρίς άδεια- χρήση της από την κοινότητα ISM, ενώ οι συσκευές που την ενσωματώνουν μπορούν να λειτουργήσουν χωρίς προβλήματα σε οποιοδήποτε σημείο του πλανήτη. Επίσης όπως έχει προαναφερθεί στο δεύτερο κεφάλαιο με θέμα «Η τεχνολογία Bluetooth, ο ανταγωνισμός και η επικράτηση της στην αγορά», είναι μια δημοφιλής τεχνολογία που ανταγωνίζεται επάξια τις άλλες συναφής τεχνολογίες που βρίσκονται στο χώρο της αγοράς.

Το Λ.Σ. του Android όπως αναπτύχτηκε στο τρίτο κεφάλαιο «Το λειτουργικό σύστημα κινητών τερματικών Android», γνωρίζει ραγδαία εξάπλωση και μέσα σε λίγα χρόνια έχει καταλάβει ήδη τα 2/3 της παγκόσμιας αγοράς. Παράγοντες όπως είναι η φιλικότητα του προς το χρήστη, το γεγονός ότι διατίθεται δωρεάν από τη Google και ότι έχουν αναπτυχθεί πληθώρα εφαρμογών πάνω σ' αυτό, το καθιστούν το δημοφιλέστερο Λ.Σ. για τις έξυπνες τερματικές συσκευές, των τελευταίων χρόνων.

Η ανάπτυξη των εφαρμογών, όπως έχει παρουσιαστεί στο τέταρτο κεφάλαιο «Τα δομικά στοιχειά μιας Android εφαρμογής» θεωρείται σχετικά εύκολη. Η ιστοσελίδα των Android Developers παρέχει δωρεάν κώδικά που καλύπτει τις βασικές λειτουργίες που μπορούν να αναπτυχθούν σε μια εφαρμογή android, έκτος αυτού υπάρχει διαθέσιμο υλικό εκπαίδευσης, οδηγοί εφαρμογών (API Guides), αναφορές, εργαλεία ανάπτυξης λογισμικού, υπηρεσίες της Google και παραδείγματα (π.χ. τα διαγράμματα του κεφαλαίου 6, «Ανάλυση και σχεδίαση της εφαρμογής πριν την υλοποίηση»). Τα προγραμματιστικά εργαλεία όπως αναφέρθηκε στο πέμπτο κεφάλαιο «Απαιτήσεις μηχανικού εξοπλισμού και λογισμικού» είναι εύκολα προσβάσιμα, διατίθενται δωρεάν στο διαδίκτυο συνοδευόμενα από αναλυτικές οδηγίες για την εξοικείωση με τη νέα τεχνολογία. Το Android Market δίνει την δυνατότητα στους προγραμματιστές να δημοσιεύσουν ανέξοδα και εύκολα τις εφαρμογές τους. Η ανάρτηση και μόνο του εκτελέσιμου αρχείου .apk της εφαρμογής στο Android Market είναι αρκετό ώστε η εφαρμογή να είναι προσβάσιμη σε όλους τους χρήστες των Android συσκευών οποιαδήποτε στιγμή και από κάθε σημείο.

Τέλος στο έκτο κεφάλαιο «Υλοποίηση» παρουσιάζεται και αναλύεται ο κώδικας της εφαρμογής του BluetoothChat. Η εφαρμογή του BluetoothChat που παρουσιάστηκε είναι ένα demo και δεν θα μπορούσε να προωθηθεί στην αγορά,. Έχει υλοποιηθεί για την παρουσίασης της διπλωματικής εργασίας. Αυτό σημαίνει ότι μπορεί να χρησιμοποιηθεί για την επίδειξη του τρόπου λειτουργίας του Bluetooth, το πώς γίνεται η διαδικασία της ανταλλαγής των μηνυμάτων και η αποθήκευση τους στη Β.Δ. και πως σχεδιάζονται οι διεπαφές του χρήστη. Η διάθεση της στην αγορά, στο Android Market, είναι εφικτή εφόσον πραγματοποιηθούν ελάχιστες παρεμβάσεις (όπως π.χ. η αποθήκευση δεδομένων σε αρχείο).

Εν κατακλείδι στην παρούσα εργασία αναπτύχθηκε μία εφαρμογή για έξυπνες φορητές συσκευές (smartphones) με Λ.Σ. Android. Η εν λόγω εφαρμογή βασίζεται στην τεχνολογία του Bluetooth. Ο κύριος στόχος της εργασίας ήταν να παρουσιάσει την τεχνολογία το Bluetooth, τον τρόπο ανταλλαγής μηνυμάτων μέσω αυτής της τεχνολογίας καθώς και η αποθήκευση τους στις Β.Δ. Μέσα από την εργασία αυτή αναδείχτηκε πως μπορούν έτοιμα κομμάτια του κώδικά που βρίσκονται στη σελίδα των Android Developers να προσαρτηθούν με τέτοιο τρόπο ώστε να αποτελέσουν τμήματα του κώδικα της εφαρμογής του BluetoothChat.

Αξίζει να σημειωθεί ότι κατά τη διάρκεια της εκπόνησης της εργασίας αποκτήθηκαν σημαντικές γνώσεις για τις τεχνολογίες των έξυπνων κινητών τηλεφώνων και τον τρόπο ανάπτυξης των εφαρμογών τους.

ΒΙΒΛΙΟΓΡΑΦΙΑ - ΑΝΑΦΟΡΕΣ

[1] R. Rajaraman, "Bluetooth and Mobile IP," 2010. [Online]. Available: http://www.ccs.neu.edu/home/rraj/Courses/6710/S10/Lectures/BluetoothMobileIP.pdf.

[2] "wikipedia Bluetooth," 25 Νοεμβρίου 2013. [Online]. Available: http://en.wikipedia.org/wiki/Bluetooth.

[3] "Harald Bluetooth," wikipedia, 14 November 2013. [Online]. Available: http://en.wikipedia.org/wiki/Harald_Bluetooth.

[4] B. S. I. G. (SIG), "bluetooth.com," [Online]. Available: http://www.bluetooth.com/Pages/sig-membership.aspx.

[5] A. Technologies, "Bluetooth Measurement Fundamentals," 12 October 2006. [Online]. Available: http://cp.literature.agilent.com/litweb/pdf/5988-3760EN.pdf.

[6] N. Gupta, Inside Bluetooth Low Energy, Boston: Artech House, 2013.

[7] S. William, Ασύρματες Επικοινωνίες και Δίκτυα, Εκδ. Τζιόλα, 2007.

[8] N. Jain. [Online].

[9] Α. Αγγελική, *Ευρυζωνικά δίκτυα (4) Αποτελεσματική χρήση του φάσματος – Πολυπλεξία και Διασπορά Φάσματος,* Αθήνα: Αλεξίου Αγγελική, Πανεπιστήμιο Πειραιά, 2012.

[10] B. S. I. Group, "Bluetooth.org," 1998. [Online]. Available: https://www.bluetooth.org/en-us.

[11] S. Liu, "Bluetooth Technology," [Online]. Available: http://progtutorials.tripod.com/Bluetooth_Technology.htm.

[12] A. M. J. G. a. H. B. Godfrey Tan, "Forming Scatternets from Bluetooth Personal

Area Networks," MIT Laboratory for Computer Science, 1 OCTOBER 2001. [Online]. Available: http://wind.lcs.mit.edu/projects/blueware/tr826.pdf.

[13] J. ,. E. R. S. B. Haartsen, "The Bluetooth Radio System - IEEE Personal Communications," 2000. [Online]. Available: http://www.csie.ntu.edu.tw/~b92018/course/WMS/The%20Bluetooth%20Radio%20System.pdf.

[14] P. Guprta, "The Wireless Developer Network," [Online]. Available: http://www.wirelessdevnet.com/channels/bluetooth/features/bluetooth_irda5.html.

[15] "wikipedia IEEE_802.15.4," 19 November 2013. [Online]. Available: http://en.wikipedia.org/wiki/IEEE_802.15.4.

[16] "wikipedia ZigBee," 25 November 2013. [Online]. Available: http://en.wikipedia.org/wiki/ZigBee.

[17] "wikipedia WirelessHART," 11 October 2013. [Online]. Available: http://en.wikipedia.org/wiki/WirelessHART.

[18] P. Gupta, "wirelessdevnet bluetooth_irda4," The Wireless Developer Network, [Online]. Available: http://www.wirelessdevnet.com/channels/bluetooth/features/bluetooth_irda4.html.

[19] D. Pheteplace, "Connector + Cable Assembly Supplier," Connector Supplier + Cable Assembly Supplier, 2 February 2015. [Online]. Available: http://www.connectorsupplier.com/facts-figures-wireless-continues-upward-trend/. [Accessed 28 February 2015].

[20] ABI Research, "Bluetooth SIG, Inc," Bluetooth SIG, Inc, 2013. [Online]. Available: http://www.bluetooth.com/Pages/SIG-Membership.aspx. [Accessed 15 Δεκέμβριος 2013].

[21] Jonathan Kaye, Laird Technologies, "EECatalog," EECatalog, 2014. [Online].

Available: http://eecatalog.com/lps/2014/02/07/replace-cable-applications-in-industrial-automation-with-bluetooth-low-energy-ble/. [Accessed 2014].

[22] T. Moore, "Wireless Communications in Factory and Process Automation," IMS Research, Colorado USA, 2013.

[23] T. Moore, "Wireless Communications in Factory and Process Automation," IHS , Colorado USA, 2013.

[24] N. Hunn, "To Ubiquity and Beyond," WiFore Consulting, 2013.

[25] w. Smartphone, "wikipedia Smartphone," wikipedia Smartphone, 26 November 2013. [Online]. Available: http://en.wikipedia.org/wiki/Smartphone.

[26] Β.-Α. Σ. Απόστολος Μηλιώνης, *Λειτουργικά Συστ. και Εφαρμογές Κινητών Τερματικών -Επισκόπηση έξυπνων συσκευών και εισαγωγή στην πλατφόρμα Android,* Πειραιάς: Πανεπιστήμιο Πειραιά, Τμημα Ψηφιακών συστημάτων, 2012.

[27] o. h. a. OHA, "open handset alliance OHA," open handset alliance OHA, 5 November 2007. [Online]. Available: http://www.openhandsetalliance.com.

[28] "wikipedia.org Android operating_system," wikipedia.org Android operating_system, 28 November 2013. [Online]. Available: http://en.wikipedia.org/wiki/Android_%28operating_system%29.

[29] M. Agarwal, "Iamwire," Wirefoot India Technology Private Limited, 14 August 2013. [Online]. Available: http://www.iamwire.com/2013/08/smartphone-sales-surpass-feature-phone-sales-2q13-time-gartner/.

[30] ABI Reaserch, iTersNews, "iTersNews," ABI Reaserch, 06 08 2014. [Online]. Available: http://itersnews.com/?p=82424. [Accessed 07 02 2015].

[31] Β.-Α. Σ. Απόστολος Μηλιώνης, *Λειτοργικά συστ. και εφαρμογές κινητων τερματικών -Mobile Application Ανάπτυξη εφαρμογών με το ANDROID,* Πειραιάς: Πανεπιστήμιο Πειραιώς, Τμήμα Ψηφιακών Συστημάτων, 2012.

[32] A.-O. H. Alliance, "Android," Android , 2013. [Online]. Available: http://source.android.com/.

[33] Wikipedia, the free encyclopedia, "Wikipedia, the free encyclopedia," Wikimedia Foundation, Inc., 13 Φεβρουάριος 2015. [Online]. Available: http://en.wikipedia.org/wiki/Android_version_history. [Accessed 15 Φεβρουάριος 2015].

[34] "developer.android Fragments," 25 Nov 2013. [Online]. Available: http://developer.android.com/reference/android/app/Activity.html#Fragments.

[35] android developers, "android developers," android developers, 2014. [Online]. Available: http://developer.android.com/training/basics/activity-lifecycle/starting.html. [Accessed 2014].

[36] Android Developers, "Android Developers," Android, [Online]. Available: http://developer.android.com/tools/sdk/eclipse-adt.html. [Accessed 2015].

[37] oracle, "oracle," oracle corp, 2013. [Online]. Available: http://www.oracle.com/technetwork/java/javase/downloads/index.html. [Accessed 2013].

[38] Sparx Systems, "Sparx Systems," Sparx Systems Pty Ltd., 2000. [Online]. Available: http://www.sparxsystems.com/uml-tutorial.html. [Accessed 2015].

[39] OMG, "http://www.uml.org/#UML2.0," 03 10 2014. [Online]. Available: http://www.uml.org/#UML2.0.

[40] Oracle Corporation, "Oracle," [Online]. Available: http://www.oracle.com/technetwork/articles/javame/index-140411.html. [Accessed 15 01 2015].

[41] Oracle corp., "Oracle corp.," Bluetooth Interest Group, 31 Oct 2013. [Online]. Available: http://www.oracle.com/technetwork/articles/javame/index-156193.html. [Accessed 17 02 2014].

[42] Google Inc., "Android 5.0 Compatibility Program," 11 Ιανουάριος 2015. [Online]. Available: http://static.googleusercontent.com/media/source.android.com/el//compatibility/android-cdd.pdf. [Accessed 25 Ιανουάριος 2015].

[43] android developers, "android developers2," android developers, [Online]. Available: http://developer.android.com/guide/topics/connectivity/bluetooth.html#Finding Devices. [Accessed 10 Μάρτιος 2014].

[44] Android Developers, "Android Developers," [Online]. Available: http://developer.android.com/guide/topics/connectivity/bluetooth.html#Finding Devices. [Accessed 2014].

[45] J. S. N. To, The Android Developer's Cookbook, Boston: Pearson Education, Inc., 2011.

[46] C. C. A. R. D. Kevin Townsend, Getting Started with Bluetooth Low Energy, O'Reilly Media, 2014.

[47] Android Developers, "Android Developers," Android Developers, 2013. [Online]. Available: http://developer.android.com/reference/android/os/Handler.html. [Accessed Νοέμβριος 2014].

[48] Android Developers, "Android Developers," [Online]. Available: http://developer.android.com/guide/topics/connectivity/bluetooth.html#ConnectingDevices. [Accessed 2014].

[49] Android Developers, "Android Developers," [Online]. Available: http://developer.android.com/reference/android/bluetooth/BluetoothAdapter.html#startDiscovery%28%29. [Accessed 2014].

[50] Android Developers, "Android Developers," Android Developers, 2013. [Online]. Available: http://developer.android.com/reference/android/bluetooth/BluetoothDevice.html

. [Accessed 2014].

[51] Android Developers, "Android Developers," Android Developers, [Online]. Available: http://developer.android.com/guide/topics/connectivity/bluetooth.html. [Accessed 2014].

[52] L. Vogel, "http://www.vogella.com," vogella GmbH, 2009. [Online]. Available: http://www.vogella.com/tutorials/AndroidSQLite/article.html#overview_sqlite. [Accessed 2012].

[53] D. R. Hipp, "SQLite," SQLite, [Online]. Available: http://www.sqlite.org.

[54] Ravi Tamada, "AndroidHive," AndroidHive, 27 Nov 2011. [Online]. Available: http://www.androidhive.info/2011/11/android-sqlite-database-tutorial/. [Accessed 16 05 2013].

[55] L. Vogel, "vogella," 2010. [Online]. Available: http://www.vogella.com/tutorials/AndroidSQLite/article.html. [Accessed 16 05 2013].

[56] GitHub, "https://github.com," GitHub Inc, February 2008. [Online]. Available: https://github.com/codepath/android_guides/wiki/Using-an-ArrayAdapter-with-ListView. [Accessed 28 11 2014].

[57] Android developers, "Android developers," Android developers, [Online]. Available: http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html. [Accessed 19 05 2013].

[58] C. Sessa, 50 Android Hacks, Shelter Island, NY: Manning Publications Co., 2013.

[59] "wikipedia ISA100.11a," 16 November 2013. [Online]. Available: http://en.wikipedia.org/wiki/ISA100.11a.

[60] D. &. P. Deitel H.M, JAVA Προγραμματισμός (μεταφρασμένο) Έκδοση 6η, Γκιούρδας, 2006.

[61] K. I.K, Προγραμματισμός με JAVA, Αθήνα: Κλειδάριθμος, 2003.

[62] J. Friensen, "Learn Java for Android Development e-book," 2010.

[63] "wikipedia," [Online]. Available: http://el.wikipedia.org/wiki/Bluetooth.

[64] N. J. a. B. N. J. Nitin Pabuwal, "An Architectural Framework to deploy Scatternet-based Applications over Bluetooth," May 2003. [Online]. Available: http://research.microsoft.com/en-us/um/people/navendu/mypapers/icc03.pdf.

[65] H. Wang, "Overview of Bluetooth Technology," PENNSTATE, State College, Dept. of Electrical Engineering, 3 July 2001. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.96.3217&rep=rep1&type=pdf.

[66] "wikipedia IrDA," 25 November 2013. [Online]. Available: http://en.wikipedia.org/wiki/IrDA.

[67] "Android-x86 - Porting Android to x86," Google, November 29 2013. [Online]. Available: http://www.android-x86.org/download.

[68] O. P. Policies, "Appendix B. VirtualBox privacy information, Version 5," Oracle , 2012.

[69] OMG, "http://www.uml.org/#UML2.0," 03 10 2014. [Online].

[70] Android Developers, "Android Developers," Android Developers, [Online]. Available: http://developer.android.com/guide/topics/ui/menus.html#options-menu. [Accessed 2014].

ΠΑΡΑΡΤΗΜΑ

Ο πηγαίος κώδικας του αρχείου arrayAdapter.java

```
package com.example.android.BluetoothChat;

import java.util.ArrayList;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;

public class arrayAdapter extends ArrayAdapter<String> {
  private final Context context;
  private final ArrayList<String> values;

  public arrayAdapter(Context context, ArrayList<String> values) {
    super(context, R.layout.listbox, values);
    this.context = context;
    this.values = values;
  }

@Override
  public View getView(int position, View convertView, ViewGroup parent) {
    LayoutInflater inflater = (LayoutInflater) context
        .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    View rowView = inflater.inflate(R.layout.listbox, parent, false);
    TextView textView = (TextView) rowView.findViewById(R.id.textbox);
    ImageView imageView = (ImageView) rowView.findViewById(R.id.imagebox);
    textView.setText(values.get(position));
    // Change the icon for Windows and iPhone
    String s = values.get(position);

    return rowView;
  }
}
```

Ο πηγαίος κώδικας του αρχείου BluetoothChat.java

```
package com.example.android.BluetoothChat;

import java.util.Calendar;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.inputmethod.EditorInfo;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

/**
 * This is the main Activity that displays the current chat session.
 */
public class BluetoothChat extends Activity {
    // Debugging
    private static final String TAG = "BluetoothChat";
    private static final boolean D = true;

    // Message types sent from the BluetoothChatService Handler
    public static final int MESSAGE_STATE_CHANGE = 1;
    public static final int MESSAGE_READ = 2;
    public static final int MESSAGE_WRITE = 3;
    public static final int MESSAGE_DEVICE_NAME = 4;
    public static final int MESSAGE_TOAST = 5;

    // Key names received from the BluetoothChatService Handler
    public static final String DEVICE_NAME = "device_name";
    public static final String TOAST = "toast";

    // Intent request codes
    private static final int REQUEST_CONNECT_DEVICE = 1;
    private static final int REQUEST_ENABLE_BT = 2;

    // Layout Views
    private ListView mConversationView;
    private EditText mOutEditText;
    private Button mSendButton;

    // Name of the connected device
    private String mConnectedDeviceName = null;
    // Array adapter for the conversation thread
```

```
    private ArrayAdapter<String> mConversationArrayAdapter;
    // String buffer for outgoing messages
    private StringBuffer mOutStringBuffer;
    // Local Bluetooth adapter
    private BluetoothAdapter mBluetoothAdapter = null;
    // Member object for the chat services
    private BluetoothChatService mChatService = null;

    DatabaseHandlerConv db;
    DatabaseHandler dbChat;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if(D) Log.e(TAG, "+++ ON CREATE +++");

        db = new DatabaseHandlerConv(this);
        dbChat = new DatabaseHandler(this);

        setContentView(R.layout.main);

        // Get local Bluetooth adapter
        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

        // If the adapter is null, then Bluetooth is not supported
        if (mBluetoothAdapter == null) {
            Toast.makeText(this, "Η υπηρεσία Bluetooth δεν υποστηρίζεται.",
Toast.LENGTH_LONG).show();
            finish();
            return;
        }
    }

    @Override
    public void onStart() {
        super.onStart();
        if(D) Log.e(TAG, "++ ON START ++");

        // If BT is not on, request that it be enabled.
        // setupChat() will then be called during onActivityResult
        if (!mBluetoothAdapter.isEnabled()) {
            Intent enableIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableIntent, REQUEST_ENABLE_BT);
        // Otherwise, setup the chat session
        } else {
            if (mChatService == null) setupChat();
        }
    }

    @Override
    public synchronized void onResume() {
        super.onResume();
        if(D) Log.e(TAG, "+ ON RESUME +");

        // Performing this check in onResume() covers the case in which BT
        // was not enabled during onStart(), so we were paused to enable it
        // onResume() will be called when ACTION_REQUEST_ENABLE activity
        // returns.
```

```
        if (mChatService != null) {
            // Only if the state is STATE_NONE, do we know that we haven't
            // started already
            if (mChatService.getState() ==BluetoothChatService.STATE_NONE){
              // Start the Bluetooth chat services
              mChatService.start();
            }
        }
    }

    private void setupChat() {
        Log.d(TAG, "setupChat()");

        // Initialize the array adapter for the conversation thread
        mConversationArrayAdapter = new ArrayAdapter<String>(this,
R.layout.message);
        mConversationView = (ListView) findViewById(R.id.in);
        mConversationView.setAdapter(mConversationArrayAdapter);

        // Initialize the compose field with a listener for the return key
        mOutEditText = (EditText) findViewById(R.id.edit_text_out);
        mOutEditText.setOnEditorActionListener(mWriteListener);

        // Initialize the send button with a listener that for click events
        mSendButton = (Button) findViewById(R.id.button_send);
        mSendButton.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
               // Send a message using content of the edit text widget
               TextView view = (TextView) findViewById(R.id.edit_text_out);
               String message = view.getText().toString();
               sendMessage(message);
            }
        });

        // Initialize the BluetoothChatService to perform bluetooth
        // connections
        mChatService = new BluetoothChatService(this, mHandler);

        // Initialize the buffer for outgoing messages
        mOutStringBuffer = new StringBuffer("");
    }

    @Override
    public synchronized void onPause() {
        super.onPause();
        if(D) Log.e(TAG, "- ON PAUSE -");
    }

    @Override
    public void onStop() {
        super.onStop();
        if(D) Log.e(TAG, "-- ON STOP --");
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        // Stop the Bluetooth chat services
        if (mChatService != null) mChatService.stop();
```

```
        if(D) Log.e(TAG, "--- ON DESTROY ---");
    }

    private void ensureDiscoverable() {
        if(D) Log.d(TAG, "ensure discoverable");
        if (mBluetoothAdapter.getScanMode() !=
            BluetoothAdapter.SCAN_MODE_CONNECTABLE_DISCOVERABLE) {
            Intent discoverableIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);

discoverableIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION,
300);
            startActivity(discoverableIntent);
        }
    }

    /**
     * Sends a message.
     * @param message  A string of text to send.
     */
    private void sendMessage(String message) {
        // Check that we're actually connected before trying anything
        if (mChatService.getState() !=
BluetoothChatService.STATE_CONNECTED) {
            Toast.makeText(this, R.string.not_connected,
Toast.LENGTH_SHORT).show();
            return;
        }

        // Check that there's actually something to send
        if (message.length() > 0) {
            // Get the message bytes and tell the BluetoothChatService to
            // write
            byte[] send = message.getBytes();
            mChatService.write(send);

            //Reset out string buffer to zero and clear the edit text field
            mOutStringBuffer.setLength(0);
            mOutEditText.setText(mOutStringBuffer);
        }
    }

    // The action listener for the EditText widget, to listen for the
    // return key
    private TextView.OnEditorActionListener mWriteListener =
        new TextView.OnEditorActionListener() {
        public boolean onEditorAction(TextView view, int actionId, KeyEvent
event) {
            // If the action is a key-up event on the return key, send the
            // message
            if (actionId == EditorInfo.IME_NULL && event.getAction() ==
KeyEvent.ACTION_UP) {
                String message = view.getText().toString();
                sendMessage(message);
            }
            if(D) Log.i(TAG, "END onEditorAction");
            return true;
        }
    };
```

```
// The Handler that gets information back from the BluetoothChatService
@SuppressLint("HandlerLeak")
   private final Handler mHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        switch (msg.what) {
        case MESSAGE_STATE_CHANGE:
            if(D) Log.i(TAG, "MESSAGE_STATE_CHANGE: " + msg.arg1);
            switch (msg.arg1) {
            case BluetoothChatService.STATE_CONNECTED:
                mConversationArrayAdapter.clear();
                break;
            case BluetoothChatService.STATE_CONNECTING:
                break;
            case BluetoothChatService.STATE_LISTEN:
            case BluetoothChatService.STATE_NONE:
                break;
            }
            break;
        case MESSAGE_WRITE:
            byte[] writeBuf = (byte[]) msg.obj;
            Calendar c = Calendar.getInstance();
            int sec = c.get(Calendar.SECOND);
            int min = c.get(Calendar.MINUTE);
            int hor = c.get(Calendar.HOUR);

            int xrono = c.get(Calendar.YEAR);
            int mina = c.get(Calendar.MONTH) + 1;
            int mera = c.get(Calendar.DATE);
            // construct a string from the buffer
            String writeMessage = new String(writeBuf);
            mConversationArrayAdapter.add("Me:  "+"\n" + writeMessage +
                    "\n" + mera +"/"+mina+"/" +xrono+"
                     "+hor+":"+min+":"+sec+"\n");
            dbChat.addContact(new ChatContact(mConnectedDeviceName,
                     mera +"/"+mina+"/" +xrono, hor+":"+min+":"+sec,
                     "Me:  \n" + writeMessage));
            break;
        case MESSAGE_READ:
            byte[] readBuf = (byte[]) msg.obj;
            Calendar c1 = Calendar.getInstance();
            int sec1 = c1.get(Calendar.SECOND);
            int min1 = c1.get(Calendar.MINUTE);
            int hor1 = c1.get(Calendar.HOUR);

            int xrono1 = c1.get(Calendar.YEAR);
            int mina1 = c1.get(Calendar.MONTH) + 1;
            int mera1 = c1.get(Calendar.DATE);
            // construct a string from the valid bytes in the buffer
            String readMessage = new String(readBuf, 0, msg.arg1);
            mConversationArrayAdapter.add(mConnectedDeviceName+":
                "+"\n" + readMessage+ "\n" + mera1 +"/"+mina1+"/"
                 +xrono1+" "+hor1+":"+min1+":"+sec1+"\n");
            dbChat.addContact(new ChatContact(mConnectedDeviceName,
                 mera1 +"/"+mina1+"/"+xrono1, hor1+":"+min1+":" +sec1,
                 mConnectedDeviceName+": "+"\n"+ readMessage));
            break;
        case MESSAGE_DEVICE_NAME:
```

```
                // save the connected device's name
                mConnectedDeviceName =msg.getData().getString(DEVICE_NAME);
                Toast.makeText(getApplicationContext(), "Connected to "
                        + mConnectedDeviceName, Toast.LENGTH_SHORT).show();
                db.addConver(new Conversation(mConnectedDeviceName));
                break;
            case MESSAGE_TOAST:
                Toast.makeText(getApplicationContext()
                 msg.getData().getString(TOAST),Toast.LENGTH_SHORT).show();
                break;
            }
        }
    };

    public void onActivityResult(int requestCode, int resultCode, Intent
data) {
        if(D) Log.d(TAG, "onActivityResult " + resultCode);
        switch (requestCode) {
        case REQUEST_CONNECT_DEVICE:
            // When DeviceListActivity returns with a device to connect
            if (resultCode == Activity.RESULT_OK) {
                // Get the device MAC address
                String address = data.getExtras()
                      .getString(DeviceListActivity.EXTRA_DEVICE_ADDRESS);
                // Get the BLuetoothDevice object
                BluetoothDevice device =
                                  mBluetoothAdapter.getRemoteDevice(address);
                // Attempt to connect to the device
                mChatService.connect(device);
            }
            break;
        case REQUEST_ENABLE_BT:
            // When the request to enable Bluetooth returns
            if (resultCode == Activity.RESULT_OK) {
                // Bluetooth is now enabled, so set up a chat session
                setupChat();
            } else {
                // User did not enable Bluetooth or an error occured
                Log.d(TAG, "BT not enabled");
                Toast.makeText(this, R.string.bt_not_enabled_leaving,
                                        Toast.LENGTH_SHORT).show();
                finish();
            }
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.option_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
        case R.id.scan:
          // Launch the DeviceListActivity to see devices and do scan
          Intent serverIntent = new Intent(this, DeviceListActivity.class);
```

```
            startActivityForResult(serverIntent, REQUEST_CONNECT_DEVICE);
            return true;
        case R.id.discoverable:
            // Ensure this device is discoverable by others
            ensureDiscoverable();
            return true;
        }
        return false;
    }

}
```

Ο πηγαίος κώδικας του αρχείου BluetoothChatService.java

```
/
* * Copyright (C) 2009 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.example.android.BluetoothChat;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.UUID;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothServerSocket;
import android.bluetooth.BluetoothSocket;
import android.content.Context;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;

/**
 * This class does all the work for setting up and managing Bluetooth
 * connections with other devices. It has a thread that listens for
 * incoming connections, a thread for connecting with a device, and a
 * thread for performing data transmissions when connected.
 */
public class BluetoothChatService {
    // Debugging
    private static final String TAG = "BluetoothChatService";
    private static final boolean D = true;

    // Name for the SDP record when creating server socket
    private static final String NAME = "BluetoothChat";

    // Unique UUID for this application
    private static final UUID MY_UUID = UUID.fromString("fa87c0d0-afac-
11de-8a39-0800200c9a66");

    // Member fields
    private final BluetoothAdapter mAdapter;
    private final Handler mHandler;
    private AcceptThread mAcceptThread;
    private ConnectThread mConnectThread;
    private ConnectedThread mConnectedThread;
```

```
    private int mState;

    // Constants that indicate the current connection state
    public static final int STATE_NONE = 0;       // we're doing nothing
    public static final int STATE_LISTEN = 1;     // now listening for
incoming connections
    public static final int STATE_CONNECTING = 2; // now initiating an
outgoing connection
    public static final int STATE_CONNECTED = 3;  // now connected to a
remote device

    /**
     * Constructor. Prepares a new BluetoothChat session.
     * @param context  The UI Activity Context
     * @param handler  A Handler to send messages back to the UI Activity
     */
    public BluetoothChatService(Context context, Handler handler) {
        mAdapter = BluetoothAdapter.getDefaultAdapter();
        mState = STATE_NONE;
        mHandler = handler;
    }

    /**
     * Set the current state of the chat connection
     * @param state  An integer defining the current connection state
     */
    private synchronized void setState(int state) {
        if (D) Log.d(TAG, "setState() " + mState + " -> " + state);
        mState = state;

        // Give the new state to the Handler so the UI Activity can update
        mHandler.obtainMessage(BluetoothChat.MESSAGE_STATE_CHANGE, state, -
1).sendToTarget();
    }

    /**
     * Return the current connection state. */
    public synchronized int getState() {
        return mState;
    }

    /**
     * Start the chat service. Specifically start AcceptThread to begin a
     * session in listening (server) mode. Called by the Activity
onResume() */
    public synchronized void start() {
        if (D) Log.d(TAG, "start");

        // Cancel any thread attempting to make a connection
        if (mConnectThread != null) {mConnectThread.cancel();
mConnectThread = null;}

        // Cancel any thread currently running a connection
        if (mConnectedThread != null) {mConnectedThread.cancel();
mConnectedThread = null;}

        // Start the thread to listen on a BluetoothServerSocket
        if (mAcceptThread == null) {
            mAcceptThread = new AcceptThread();
```

```
            mAcceptThread.start();
        }
        setState(STATE_LISTEN);
    }

    /**
     * Start the ConnectThread to initiate a connection to a remote device.
     * @param device  The BluetoothDevice to connect
     */
    public synchronized void connect(BluetoothDevice device) {
        if (D) Log.d(TAG, "connect to: " + device);

        // Cancel any thread attempting to make a connection
        if (mState == STATE_CONNECTING) {
            if (mConnectThread != null) {mConnectThread.cancel();
mConnectThread = null;}
        }

        // Cancel any thread currently running a connection
        if (mConnectedThread != null) {mConnectedThread.cancel();
mConnectedThread = null;}

        // Start the thread to connect with the given device
        mConnectThread = new ConnectThread(device);
        mConnectThread.start();
        setState(STATE_CONNECTING);
    }

    /**
     * Start the ConnectedThread to begin managing a Bluetooth connection
     * @param socket  The BluetoothSocket on which the connection was made
     * @param device  The BluetoothDevice that has been connected
     */
    public synchronized void connected(BluetoothSocket socket,
BluetoothDevice device) {
        if (D) Log.d(TAG, "connected");

        // Cancel the thread that completed the connection
        if (mConnectThread != null) {mConnectThread.cancel();
mConnectThread = null;}

        // Cancel any thread currently running a connection
        if (mConnectedThread != null) {mConnectedThread.cancel();
mConnectedThread = null;}

        // Cancel the accept thread because we only want to connect to one
device
        if (mAcceptThread != null) {mAcceptThread.cancel(); mAcceptThread =
null;}

        // Start the thread to manage the connection and perform
transmissions
        mConnectedThread = new ConnectedThread(socket);
        mConnectedThread.start();

        // Send the name of the connected device back to the UI Activity
        Message msg =
mHandler.obtainMessage(BluetoothChat.MESSAGE_DEVICE_NAME);
        Bundle bundle = new Bundle();
```

```
        bundle.putString(BluetoothChat.DEVICE_NAME, device.getName());
        msg.setData(bundle);
        mHandler.sendMessage(msg);

        setState(STATE_CONNECTED);
    }

    /**
     * Stop all threads
     */
    public synchronized void stop() {
        if (D) Log.d(TAG, "stop");
        if (mConnectThread != null) {mConnectThread.cancel();
mConnectThread = null;}
        if (mConnectedThread != null) {mConnectedThread.cancel();
mConnectedThread = null;}
        if (mAcceptThread != null) {mAcceptThread.cancel(); mAcceptThread =
null;}
        setState(STATE_NONE);
    }

    /**
     * Write to the ConnectedThread in an unsynchronized manner
     * @param out The bytes to write
     * @see ConnectedThread#write(byte[])
     */
    public void write(byte[] out) {
        // Create temporary object
        ConnectedThread r;
        // Synchronize a copy of the ConnectedThread
        synchronized (this) {
            if (mState != STATE_CONNECTED) return;
            r = mConnectedThread;
        }
        // Perform the write unsynchronized
        r.write(out);
    }

    /**
    *Indicate that the connection attempt failed and notify the UI.
     Activity */
    private void connectionFailed() {
        setState(STATE_LISTEN);

        // Send a failure message back to the Activity
        Message msg = mHandler.obtainMessage(BluetoothChat.MESSAGE_TOAST);
        Bundle bundle = new Bundle();
        bundle.putString(BluetoothChat.TOAST, "Unable to connect device");
        msg.setData(bundle);
        mHandler.sendMessage(msg);
    }

    /**
     * Indicate that the connection was lost and notify the UI Activity.
     */
    private void connectionLost() {
        setState(STATE_LISTEN);

        // Send a failure message back to the Activity
```

```
        Message msg = mHandler.obtainMessage(BluetoothChat.MESSAGE_TOAST);
        Bundle bundle = new Bundle();
        bundle.putString(BluetoothChat.TOAST, "Device connection was
lost");
        msg.setData(bundle);
        mHandler.sendMessage(msg);
    }

    /**
    * This thread runs while listening for incoming connections. It behaves
    * like a server-side client. It runs until a connection is accepted
    * (or until cancelled).
    */
    private class AcceptThread extends Thread {
       // The local server socket
        private final BluetoothServerSocket mmServerSocket;

        public AcceptThread() {
            BluetoothServerSocket tmp = null;

            // Create a new listening server socket
            try {
               tmp = mAdapter.listenUsingRfcommWithServiceRecord(NAME,
MY_UUID);
            } catch (IOException e) {
                Log.e(TAG, "listen() failed", e);
            }
            mmServerSocket = tmp;
        }

        public void run() {
            if (D) Log.d(TAG, "BEGIN mAcceptThread" + this);
            setName("AcceptThread");
            BluetoothSocket socket = null;

            // Listen to the server socket if we're not connected
            while (mState != STATE_CONNECTED) {
                try {
                    // This is a blocking call and will only return on a
                    // successful connection or an exception
                    socket = mmServerSocket.accept();
                } catch (IOException e) {
                    Log.e(TAG, "accept() failed", e);
                    break;
                }

                // If a connection was accepted
                if (socket != null) {
                    synchronized (BluetoothChatService.this) {
                        switch (mState) {
                        case STATE_LISTEN:
                        case STATE_CONNECTING:
                          // Situation normal. Start the connected thread.
                          connected(socket, socket.getRemoteDevice());
                          break;
                        case STATE_NONE:
                        case STATE_CONNECTED:
                          // Either not ready or already connected.
Terminate new socket.
```

```
                            try {
                                socket.close();
                            } catch (IOException e) {
                           Log.e(TAG, "Could not close unwanted socket", e);
                            }
                            break;
                        }
                    }
                }
            }
            if (D) Log.i(TAG, "END mAcceptThread");
        }

        public void cancel() {
            if (D) Log.d(TAG, "cancel " + this);
            try {
                mmServerSocket.close();
            } catch (IOException e) {
                Log.e(TAG, "close() of server failed", e);
            }
        }
    }


    /**
     * This thread runs while attempting to make an outgoing connection
     * with a device. It runs straight through; the connection either
     * succeeds or fails.
     */
    private class ConnectThread extends Thread {
        private final BluetoothSocket mmSocket;
        private final BluetoothDevice mmDevice;

        public ConnectThread(BluetoothDevice device) {
            mmDevice = device;
            BluetoothSocket tmp = null;

            // Get a BluetoothSocket for a connection with the
            // given BluetoothDevice
            try {
                tmp = device.createRfcommSocketToServiceRecord(MY_UUID);
            } catch (IOException e) {
                Log.e(TAG, "create() failed", e);
            }
            mmSocket = tmp;
        }

        public void run() {
            Log.i(TAG, "BEGIN mConnectThread");
            setName("ConnectThread");

            // Always cancel discovery because it will slow down a
connection
            mAdapter.cancelDiscovery();

            // Make a connection to the BluetoothSocket
            try {
                // This is a blocking call and will only return on a
                // successful connection or an exception
```

```
                mmSocket.connect();
            } catch (IOException e) {
                connectionFailed();
                // Close the socket
                try {
                    mmSocket.close();
                } catch (IOException e2) {
                    Log.e(TAG, "unable to close() socket during connection
failure", e2);
                }
                // Start the service over to restart listening mode
                BluetoothChatService.this.start();
                return;
            }

            // Reset the ConnectThread because we're done
            synchronized (BluetoothChatService.this) {
                mConnectThread = null;
            }

            // Start the connected thread
            connected(mmSocket, mmDevice);
        }

        public void cancel() {
            try {
                mmSocket.close();
            } catch (IOException e) {
                Log.e(TAG, "close() of connect socket failed", e);
            }
        }
    }

    /**
     * This thread runs during a connection with a remote device.
     * It handles all incoming and outgoing transmissions.
     */
    private class ConnectedThread extends Thread {
        private final BluetoothSocket mmSocket;
        private final InputStream mmInStream;
        private final OutputStream mmOutStream;

        public ConnectedThread(BluetoothSocket socket) {
            Log.d(TAG, "create ConnectedThread");
            mmSocket = socket;
            InputStream tmpIn = null;
            OutputStream tmpOut = null;

            // Get the BluetoothSocket input and output streams
            try {
                tmpIn = socket.getInputStream();
                tmpOut = socket.getOutputStream();
            } catch (IOException e) {
                Log.e(TAG, "temp sockets not created", e);
            }

            mmInStream = tmpIn;
            mmOutStream = tmpOut;
        }
```

```
        public void run() {
            Log.i(TAG, "BEGIN mConnectedThread");
            byte[] buffer = new byte[1024];
            int bytes;

            // Keep listening to the InputStream while connected
            while (true) {
                try {
                    // Read from the InputStream
                    bytes = mmInStream.read(buffer);

                    // Send the obtained bytes to the UI Activity
                    mHandler.obtainMessage(BluetoothChat.MESSAGE_READ,
bytes, -1, buffer)
                            .sendToTarget();
                } catch (IOException e) {
                    Log.e(TAG, "disconnected", e);
                    connectionLost();
                    break;
                }
            }
        }

        /**
         * Write to the connected OutStream.
         * @param buffer  The bytes to write
         */
        public void write(byte[] buffer) {
            try {
                mmOutStream.write(buffer);

                // Share the sent message back to the UI Activity
                mHandler.obtainMessage(BluetoothChat.MESSAGE_WRITE, -1, -1,
buffer)
                        .sendToTarget();
            } catch (IOException e) {
                Log.e(TAG, "Exception during write", e);
            }
        }

        public void cancel() {
            try {
                mmSocket.close();
            } catch (IOException e) {
                Log.e(TAG, "close() of connect socket failed *", e);
            }
        }
    }
}
```

Ο πηγαίος κώδικας του αρχείου `ChatContact.java`

```
package com.example.android.BluetoothChat;

public class ChatContact {
    //private variables
    int _id;
    String _username;
    String _date;
    String _time;
    String _message;

    // Empty constructor
    public ChatContact(){

    }
    // constructor
    public ChatContact(int id, String username, String date, String time,
String msg){
        this._id = id;
        this._username = username;
        this._date = date;
        this._time = time;
        this._message = msg;
    }

    // constructor
    public ChatContact(String username,String date,String time,String msg){
        this._username = username;
        this._date = date;
        this._time = time;
        this._message = msg;

    }
    // getting ID
    public int getID(){
        return this._id;
    }

    // setting id
    public void setID(int id){
        this._id = id;
    }

    // getting username
    public String getUsername(){
        return this._username;
    }

    // setting username
    public void setUsername(String username){
        this._username = username;
    }

    // getting date
    public String getDate(){
        return this._date;
    }
```

```
    // setting date
    public void setDate(String date){
        this._date = date;
    }

    // getting time
    public String getTime(){
        return this._time;
    }

    // setting time
    public void setTime(String time){
        this._time = time;
    }

    // getting message
    public String getMessage(){
        return this._message;
    }

    // setting message
    public void setMessage(String msg){
        this._message = msg;
    }

}
```

Ο πηγαίος κώδικας του αρχείου Conversation.java

```
package com.example.android.BluetoothChat;

public class Conversation {

    //private variables
    int _id;
    String _username;

    // Empty constructor
    public Conversation(){

    }
    // constructor
    public Conversation(int id, String username){
        this._id = id;
        this._username = username;
   }

    // constructor without id
    public Conversation(String username){
        this._username = username;
    }
    // getting ID
    public int getID(){
        return this._id;
    }

    // setting id
    public void setID(int id){
        this._id = id;
    }

    // getting username
    public String getUsername(){
        return this._username;
    }

    // setting username
    public void setUsername(String username){
        this._username = username;
    }
}
```

Ο πηγαίος κώδικας του αρχείου ConvList.java

```
package com.example.android.BluetoothChat;

import java.util.ArrayList;
import java.util.List;
import android.app.Activity;
import android.os.Bundle;
import android.widget.ListView;

public class ConvList extends Activity {
	private arrayAdapter adapter;
	String name;
	private ArrayList<String> values = new ArrayList<String>();

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.convlist);

		ListView listView = (ListView) findViewById(R.id.in1);

		Bundle dat = this.getIntent().getExtras();

	 name = dat.getString("position");

		final DatabaseHandler db1 = new DatabaseHandler(this);

		List<ChatContact> contact = db1.getAllContacts();

		for (ChatContact cn1 : contact) {
		if (name.contains(cn1.getUsername()) && (name.length() ==
(cn1.getUsername().length()+1))) {
		  values.add(""+cn1.getMessage()+"\n"+cn1.getDate()+"
"+cn1.getTime()+"\n");
		}
		 }

		adapter = new arrayAdapter(ConvList.this,  values);
		listView.setAdapter(adapter);
	}
}
```

Ο πηγαίος κώδικας του αρχείου DatabaseHandler.java

```
package com.example.android.BluetoothChat;

import java.util.ArrayList;
import java.util.List;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseHandler extends SQLiteOpenHelper {

    // All Static variables
    // Database Version
    private static final int DATABASE_VERSION = 1;

    // Database Name
    private static final String DATABASE_NAME = "contactsManager";

    // Contacts table name
    private static final String TABLE_CONTACTS = "contacts";

    // Contacts Table Columns names
    private static final String KEY_ID = "id";
    private static final String USERNAME = "nickname";
    private static final String MESSAGE = "message";
    private static final String DATE = "date";
    private static final String TIME= "time";



    public DatabaseHandler(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    // Creating Tables
    @Override
    public void onCreate(SQLiteDatabase db) {
        String CREATE_CONTACTS_TABLE = "CREATE TABLE " + TABLE_CONTACTS +
"("
                + KEY_ID + " INTEGER PRIMARY KEY," + USERNAME + " TEXT,"
                + DATE + " TEXT," + TIME + " TEXT,"
                + MESSAGE + " TEXT" + ")";
        db.execSQL(CREATE_CONTACTS_TABLE);
    }

    // Upgrading database
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        // Drop older table if existed
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_CONTACTS);

        // Create tables again
        onCreate(db);
    }
```

```
    /**
     * All CRUD(Create, Read, Update, Delete) Operations
     */

    // Adding new contact
    void addContact(ChatContact contact) {
        SQLiteDatabase db = this.getWritableDatabase();

        ContentValues values = new ContentValues();
        values.put(USERNAME, contact.getUsername()); // Contact USERNAME
        values.put(DATE, contact.getDate()); // Contact DATE
        values.put(TIME, contact.getTime()); // Contact TIME
        values.put(MESSAGE, contact.getMessage()); // Contact MESSAGE

        // Inserting Row
        db.insert(TABLE_CONTACTS, null, values);
        db.close(); // Closing database connection
    }

    // Getting single contact
    ChatContact getContact(int id) {
        SQLiteDatabase db = this.getReadableDatabase();

        Cursor cursor = db.query(TABLE_CONTACTS, new String[] { KEY_ID,
                USERNAME, DATE, TIME, MESSAGE }, KEY_ID + "=?",
                new String[] { String.valueOf(id) }, null, null, null,
null);
        if (cursor != null)
            cursor.moveToFirst();

        ChatContact contact = new
ChatContact(Integer.parseInt(cursor.getString(0)),
                cursor.getString(1), cursor.getString(2),
cursor.getString(3), cursor.getString(4));
        // return contact
        return contact;
    }

    // Getting All Contacts
    //  emfanizei oti exei i sql
    public List<ChatContact> getAllContacts() {
        List<ChatContact> contactList = new ArrayList<ChatContact>();
        // Select All Query
        String selectQuery = "SELECT  * FROM " + TABLE_CONTACTS;

        SQLiteDatabase db = this.getWritableDatabase();
        Cursor cursor = db.rawQuery(selectQuery, null);

        // looping through all rows and adding to list
        // topothetei i sql gia na ta do ero
        if (cursor.moveToFirst()) {
            do {
                ChatContact contact = new ChatContact();
                contact.setID(Integer.parseInt(cursor.getString(0)));
                contact.setUsername(cursor.getString(1));
                contact.setDate(cursor.getString(2));
                contact.setTime(cursor.getString(3));
                contact.setMessage(cursor.getString(4));
```

```
                // Adding contact to list
                contactList.add(contact);
            } while (cursor.moveToNext());
        }

        // return contact list
        return contactList;
    }

    // Updating single contact
    public int updateContact(ChatContact contact) {
        SQLiteDatabase db = this.getWritableDatabase();

        ContentValues values = new ContentValues();
        values.put(USERNAME, contact.getUsername());
        values.put(DATE, contact.getDate());
        values.put(TIME, contact.getTime());
        values.put(MESSAGE, contact.getMessage());

        // updating row
        return db.update(TABLE_CONTACTS, values, KEY_ID + " = ?",
                new String[] { String.valueOf(contact.getID()) });
    }

    // Deleting single contact
    public void deleteContact(ChatContact contact) {
        SQLiteDatabase db = this.getWritableDatabase();
        db.delete(TABLE_CONTACTS, KEY_ID + " = ?",
                new String[] { String.valueOf(contact.getID()) });
        db.close();
    }


    // Getting contacts Count
    public int getContactsCount() {
        String countQuery = "SELECT  * FROM " + TABLE_CONTACTS;
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery(countQuery, null);
        cursor.close();

        // return count
        return cursor.getCount();
    }

}
```

Ο πηγαίος κώδικας του αρχείου DatabaseHandlerConv.java

```
package com.example.android.BluetoothChat;

import java.util.ArrayList;
import java.util.List;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseHandlerConv extends SQLiteOpenHelper {

    // All Static variables
    // Database Version
    private static final int DATABASE_VERSION = 1;

    // Database Name
    private static final String DATABASE_NAME = "conversationManager";

    // Contacts table name
    private static final String TABLE_CONVERSATIONS = "conversations";

    // Contacts Table Columns names
    private static final String KEY_ID = "id";
    private static final String USERNAME = "username";

    public DatabaseHandlerConv(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    // Creating Tables
    @Override
    public void onCreate(SQLiteDatabase dbConv) {
        String CREATE_CONVERSATIONS_TABLE = "CREATE TABLE " +
TABLE_CONVERSATIONS + "("
                + KEY_ID + " INTEGER PRIMARY KEY," + USERNAME + " TEXT"
                + ")";
        dbConv.execSQL(CREATE_CONVERSATIONS_TABLE);
    }

    // Upgrading database
    @Override
    public void onUpgrade(SQLiteDatabase dbConv, int oldVersion, int
newVersion) {
        // Drop older table if existed
       dbConv.execSQL("DROP TABLE IF EXISTS " + TABLE_CONVERSATIONS);

        // Create tables again
        onCreate(dbConv);
    }

    /**
     * All CRUD(Create, Read, Update, Delete) Operations
     */

    // Adding new contact
    void addConver(Conversation conversation) {
```

```
        SQLiteDatabase dbConv = this.getWritableDatabase();

        ContentValues values = new ContentValues();
        values.put(USERNAME, conversation.getUsername()); // conversation
USERNAME

        // Inserting Row
        dbConv.insert(TABLE_CONVERSATIONS, null, values);
        dbConv.close(); // Closing database connection
    }

    // Getting single contact
    Conversation getConver(int id) {
        SQLiteDatabase dbConv = this.getReadableDatabase();
        Cursor cursor = dbConv.query(TABLE_CONVERSATIONS, new String[] {
KEY_ID,
                USERNAME
                }, KEY_ID + "=?",
                new String[] { String.valueOf(id) }, null, null, null,
null);
        if (cursor != null)
            cursor.moveToFirst();

        Conversation conver = new
Conversation(Integer.parseInt(cursor.getString(0)),
                cursor.getString(1));
        // return contact
        return conver;
    }

    // Getting All Contacts
    public List<Conversation> getAllConversations() {
        List<Conversation> conversationList = new
ArrayList<Conversation>();
        // Select All Query
        String selectQuery = "SELECT  * FROM " + TABLE_CONVERSATIONS;

        SQLiteDatabase dbConv = this.getWritableDatabase();
        Cursor cursor = dbConv.rawQuery(selectQuery, null);
        // looping through all rows and adding to list
        if (cursor.moveToFirst()) {
            do {
             Conversation conver = new Conversation();
             conver.setID(Integer.parseInt(cursor.getString(0)));
             conver.setUsername(cursor.getString(1));
                // Adding contact to list
             conversationList.add(conver);
            } while (cursor.moveToNext());
        }
        // return contact list
        return conversationList;
    }

    // Updating single contact
    public int updateConversation(Conversation conver) {
        SQLiteDatabase dbConv = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(USERNAME, conver.getUsername());
        // updating row
```

```
        return dbConv.update(TABLE_CONVERSATIONS, values, KEY_ID + " = ?",
                new String[] { String.valueOf(conver.getID()) });
    }

    // Deleting single contact
    public void deleteConversation(Conversation conver) {
        SQLiteDatabase db = this.getWritableDatabase();
        db.delete(TABLE_CONVERSATIONS, KEY_ID + " = ?",
                new String[] { String.valueOf(conver.getID()) });
        db.close();
    }

     // Getting contacts Count
    public int getConversationCount() {
        String countQuery = "SELECT  * FROM " + TABLE_CONVERSATIONS;
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery(countQuery, null);
        cursor.close();
        // return count
        return cursor.getCount();
    }
}
```

Ο πηγαίος κώδικας του αρχείου DeviceListActivity.java

```
/*
 * Copyright (C) 2009 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.example.android.BluetoothChat;

import java.util.Set;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.Window;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.AdapterView.OnItemClickListener;

/**
 * This Activity appears as a dialog. It lists any paired devices and
 * devices detected in the area after discovery. When a device is chosen
 * by the user, the MAC address of the device is sent back to the parent
 * Activity in the result Intent.
 */
public class DeviceListActivity extends Activity {
    // Debugging
    private static final String TAG = "DeviceListActivity";
    private static final boolean D = true;

    // Return Intent extra
    public static String EXTRA_DEVICE_ADDRESS = "device_address";

    // Member fields
    private BluetoothAdapter mBtAdapter;
    private ArrayAdapter<String> mPairedDevicesArrayAdapter;
```

```
    private ArrayAdapter<String> mNewDevicesArrayAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Setup the window
        requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);
        setContentView(R.layout.device_list);

        // Set result CANCELED incase the user backs out
        setResult(Activity.RESULT_CANCELED);

        // Initialize the button to perform device discovery
        Button scanButton = (Button) findViewById(R.id.button_scan);
        scanButton.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                doDiscovery();
                v.setVisibility(View.GONE);
            }
        });

        // Initialize array adapters. One for already paired devices and
        // one for newly discovered devices
        mPairedDevicesArrayAdapter = new ArrayAdapter<String>(this,
R.layout.device_name);
        mNewDevicesArrayAdapter = new ArrayAdapter<String>(this,
R.layout.device_name);

        // Find and set up the ListView for paired devices
        ListView pairedListView = (ListView)
findViewById(R.id.paired_devices);
        pairedListView.setAdapter(mPairedDevicesArrayAdapter);
        pairedListView.setOnItemClickListener(mDeviceClickListener);

        // Find and set up the ListView for newly discovered devices
        ListView newDevicesListView = (ListView)
findViewById(R.id.new_devices);
        newDevicesListView.setAdapter(mNewDevicesArrayAdapter);
        newDevicesListView.setOnItemClickListener(mDeviceClickListener);

        // Register for broadcasts when a device is discovered
        IntentFilter filter = new
IntentFilter(BluetoothDevice.ACTION_FOUND);
        this.registerReceiver(mReceiver, filter);

        // Register for broadcasts when discovery has finished
        filter = new
IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
        this.registerReceiver(mReceiver, filter);

        // Get the local Bluetooth adapter
        mBtAdapter = BluetoothAdapter.getDefaultAdapter();

        // Get a set of currently paired devices
        Set<BluetoothDevice> pairedDevices = mBtAdapter.getBondedDevices();

        // If there are paired devices, add each one to the ArrayAdapter
        if (pairedDevices.size() > 0) {
```

```
findViewById(R.id.title_paired_devices).setVisibility(View.VISIBLE);
            for (BluetoothDevice device : pairedDevices) {
                mPairedDevicesArrayAdapter.add(device.getName() + "\n" +
device.getAddress());
            }
        } else {
            String noDevices =
getResources().getText(R.string.none_paired).toString();
            mPairedDevicesArrayAdapter.add(noDevices);
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();

        // Make sure we're not doing discovery anymore
        if (mBtAdapter != null) {
            mBtAdapter.cancelDiscovery();
        }

        // Unregister broadcast listeners
        this.unregisterReceiver(mReceiver);
    }

    /**
     * Start device discover with the BluetoothAdapter
     */
    private void doDiscovery() {
        if (D) Log.d(TAG, "doDiscovery()");

        // Indicate scanning in the title
        setProgressBarIndeterminateVisibility(true);
        setTitle(R.string.scanning);

        // Turn on sub-title for new devices
        findViewById(R.id.title_new_devices).setVisibility(View.VISIBLE);

        // If we're already discovering, stop it
        if (mBtAdapter.isDiscovering()) {
            mBtAdapter.cancelDiscovery();
        }

        // Request discover from BluetoothAdapter
        mBtAdapter.startDiscovery();
    }

    // The on-click listener for all devices in the ListViews
    private OnItemClickListener mDeviceClickListener = new
OnItemClickListener() {
        public void onItemClick(AdapterView<?> av, View v, int arg2, long
arg3) {
            // Cancel discovery because it's costly and we're about to
connect
            mBtAdapter.cancelDiscovery();

            // Get the device MAC address, which is the last 17 chars in
the View
```

```
            String info = ((TextView) v).getText().toString();
            String address = info.substring(info.length() - 17);

            // Create the result Intent and include the MAC address
            Intent intent = new Intent();
            intent.putExtra(EXTRA_DEVICE_ADDRESS, address);

            // Set result and finish this Activity
            setResult(Activity.RESULT_OK, intent);
            finish();
        }
    };

    // The BroadcastReceiver that listens for discovered devices and
    // changes the title when discovery is finished
    private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();

            // When discovery finds a device
            if (BluetoothDevice.ACTION_FOUND.equals(action)) {
                // Get the BluetoothDevice object from the Intent
                BluetoothDevice device =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                // If it's already paired, skip it, because it's been
listed already
                if (device.getBondState() != BluetoothDevice.BOND_BONDED) {
                    mNewDevicesArrayAdapter.add(device.getName() + "\n" +
device.getAddress());
                }
            // When discovery is finished, change the Activity title
            } else if
(BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action)) {
                setProgressBarIndeterminateVisibility(false);
                setTitle(R.string.select_device);
                if (mNewDevicesArrayAdapter.getCount() == 0) {
                    String noDevices =
getResources().getText(R.string.none_found).toString();
                    mNewDevicesArrayAdapter.add(noDevices);
                }
            }
        }
    };

}
```

Ο πηγαίος κώδικας του αρχείου list.java

```
package com.example.android.BluetoothChat;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.ListView;
public class list extends Activity {
   private arrayAdapter adapter;
   private ArrayList<String> values = new ArrayList<String>();
   ListView listView;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.list);

      listView = (ListView) findViewById(R.id.in);
      final DatabaseHandlerConv db = new DatabaseHandlerConv(this);
      List<Conversation> conversations = db.getAllConversations();

      for (Conversation cv : conversations) {
            values.add(cv.getUsername()+"\n");
      }

      adapter = new arrayAdapter(list.this, values);
      listView.setAdapter(adapter);
      listView.setOnItemClickListener(new OnItemClickListener() {
         @Override
         public void onItemClick(AdapterView<?> parent, View view,
            int position, long id) {
            String name0 = listView.getItemAtPosition(position).toString();
            Intent intent1 = new Intent();
            intent1.setClass(list.this, ConvList.class);
            intent1.putExtra("position", name0);
            startActivity(intent1);
         }
      });

      listView.setOnItemLongClickListener(new OnItemLongClickListener() {
         @Override
         public boolean onItemLongClick(AdapterView<?> parent, View view,
            final int position, long id) {
            new AlertDialog.Builder(list.this)
               .setTitle("ΔΙΑΓΡΑΦΗ")
               .setMessage("Θέλετε να διαγράψετε αυτή τη συζήτηση;")
```

```
                .setPositiveButton("NAI", new
DialogInterface.OnClickListener() {
                 public void onClick(DialogInterface dialog, int which) {
                   int idpos = position+1;
                   String names = null;
                   names = ""+parent.getItemAtPosition(position);

                   final DatabaseHandlerConv db3 = new
DatabaseHandlerConv(list.this);
                   List<Conversation> conversations =
db3.getAllConversations();

                   for (Conversation cv : conversations) {
                      Log.e("compare","1: "+cv.getUsername().length()+" 2:
"+names.length());
                      String names1 = cv.getUsername()+"\n";
                      if (names1.equals(names) ) {
                         Log.e("name", ":"+ cv.getUsername());
                         //  names = cv.getUsername();
                         db3.deleteConversation(cv);
                      }
                   }

                   final DatabaseHandler db4 = new
DatabaseHandler(list.this);
                   List<ChatContact> contact = db4.getAllContacts();

                   for (ChatContact cn1 : contact) {
                      if (cn1.getUsername().equals(names) ) {
                         Log.e("mesa", ":");
                         db4.deleteContact(cn1);
                      }
                   }
                   final DatabaseHandlerConv db9 = new
DatabaseHandlerConv(list.this);
                   List<Conversation> conversations9 =
db9.getAllConversations();
                   values.clear();

                   for (Conversation cv : conversations9) {
                    values.add(cv.getUsername()+"\n");
                   }

                   adapter = new arrayAdapter(list.this, values);
                   listView.setAdapter(adapter);
                 }
               })
               .setNegativeButton("OXI", new DialogInterface.OnClickListener(){
              public void onClick(DialogInterface dialog, int which) {
                // do nothing
              }
            })
            .show();
             return true;
          }
          return false;
       }
}
```

Ο πηγαίος κώδικας του αρχείου start.java

```
package com.example.android.BluetoothChat;

import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.app.Activity;
import android.content.Intent;

public class start extends Activity {
   String name;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.start);
      final DatabaseHandler db = new DatabaseHandler(this);

      if (db.getAllContacts().isEmpty()) {
         db.addContact(new
ChatContact("Maria","13/10/2013","15:40:43","Maria: \n hello "));
         db.addContact(new ChatContact("Maria","13/10/2013","15:41:43","Me:
\n hi, ti kaneis? "));
         db.addContact(new
ChatContact("Maria","13/10/2013","15:42:43","Maria: \n kala mole barieme
"));

         db.addContact(new
ChatContact("Nikos","14/10/2013","15:40:43","Nikos: \n simera stis 5 "));
         db.addContact(new ChatContact("Nikos","14/10/2013","15:41:43","Me:
\n ok "));

         db.addContact(new
ChatContact("Maria","17/10/2013","15:40:43","Maria: \n Pos pige me ton
Niko? "));
         db.addContact(new ChatContact("Maria","17/10/2013","15:41:43","Me:
\n Kala , ligo baretos "));
         db.addContact(new
ChatContact("Maria","17/10/2013","15:42:43","Maria: \n Giati to les? mia
xara pedi fenete "));
         db.addContact(new ChatContact("Maria","17/10/2013","15:43:43","Me:
\n den ksero, den perasa, nistaksa "));
         db.addContact(new
ChatContact("Maria","17/10/2013","15:44:43","Maria: \n a kalaaaaa,
katalaba... pame gia ala "));

         db.addContact(new ChatContact("Manos","19/10/2013","15:40:43","Me:
\n Geia sou Mano, pos paei i ergasia? "));
         db.addContact(new
ChatContact("Manos","19/10/2013","15:41:43","Manos: \n asta eimai gia
kafe...etsi paei. "));

         db.addContact(new
ChatContact("John","07/02/2014","09:30:43","John: \n Ti egine me ti nerit?
Bgike prokiriksi?"));
         db.addContact(new ChatContact("John","07/02/2014","09:30:57","Me:
\n Nai xtes, alla exei liges thesis "));
```

```
        db.addContact(new
ChatContact("John","07/02/2014","09:31:15","John: \n Giati etsi? "));
        db.addContact(new ChatContact("John","07/02/2014","09:31:39","Me:
\n den ksero, ti na po, xalia "));
        db.addContact(new
ChatContact("John","07/02/2014","09:32:03","John: \n a kalaaaaa,
katalaba... "));

        final DatabaseHandlerConv db1 = new DatabaseHandlerConv(this);
        db1.addConver(new Conversation("Maria"));
        db1.addConver(new Conversation("Nikos"));
        db1.addConver(new Conversation("Manos"));
        db1.addConver(new Conversation("John"));
      }
      Button chat1 = (Button) findViewById(R.id.button1);
      Button conv1 = (Button) findViewById(R.id.button2);

      chat1.setOnClickListener(new OnClickListener() {
      public void onClick(View v) {
         Intent intent1 = new Intent();
         intent1.setClass(start.this, BluetoothChat.class);
         startActivity(intent1);
      }
    });

        conv1.setOnClickListener(new OnClickListener() {
      public void onClick(View v) {
         Intent intent1 = new Intent();
         intent1.setClass(start.this, list.class);
         startActivity(intent1);
      }
    });
  }
}
```

Ο πηγαίος κώδικας του αρχείου convlist.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/back" >

    <ListView android:id="@+id/in1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:stackFromBottom="true"

        android:layout_marginRight="10dp"
        android:layout_marginTop="10dp"
        android:layout_marginBottom="10dp"
        android:background="@drawable/but"
        android:transcriptMode="alwaysScroll"
    />
</LinearLayout>
```

Ο πηγαίος κώδικας του αρχείου costum_title

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center_vertical"
  >
  <TextView android:id="@+id/title_left_text"
      android:layout_alignParentLeft="true"
      android:ellipsize="end"
      android:singleLine="true"
      style="?android:attr/windowTitleStyle"
      android:layout_width="wrap_content"
      android:layout_height="fill_parent"
      android:layout_weight="1"
    />
    <TextView android:id="@+id/title_right_text"
        android:layout_alignParentRight="true"
        android:ellipsize="end"
        android:singleLine="true"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:textColor="#fff"
        android:layout_weight="1"
    />
</RelativeLayout>
```

Ο πηγαίος κώδικας του αρχείου device_list.xml

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView android:id="@+id/title_paired_devices"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/title_paired_devices"
        android:visibility="gone"
        android:background="#666"
        android:textColor="#fff"
    />
    <ListView android:id="@+id/paired_devices"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:stackFromBottom="true"
        android:layout_weight="1"
    />
    <TextView android:id="@+id/title_new_devices"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/title_other_devices"
        android:visibility="gone"
        android:background="#666"
        android:textColor="#fff"

    />
    <ListView android:id="@+id/new_devices"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:stackFromBottom="true"
        android:layout_weight="2"
    />
    <Button android:id="@+id/button_scan"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/button_scan"
    />
</LinearLayout>
```

Ο πηγαίος κώδικας του αρχείου device_name.xml

```
<?xml version="1.0" encoding="utf-8"?>

<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="18sp"
    android:padding="5dp"
/>
```

Ο πηγαίος κώδικας του αρχείου List.xml

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/back" >

    <ListView
        android:id="@+id/in"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:stackFromBottom="true"
        android:layout_marginBottom="10dp"
        android:layout_marginRight="10dp"
        android:layout_marginTop="10dp"
        android:background="@drawable/but"
        android:transcriptMode="alwaysScroll" >
    </ListView>

</LinearLayout>
```

Ο πηγαίος κώδικας του αρχείου listbox.xml

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <ImageView
        android:id="@+id/imagebox"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:src="@drawable/empty"
        android:visibility="gone"/>

    <TextView
        android:id="@+id/textbox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="18dp"
        android:layout_marginLeft="20dp"
        android:text="Test" />

</LinearLayout>
```

Ο πηγαίος κώδικας του αρχείου message.xml

```
<?xml version="1.0" encoding="utf-8"?>

<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="18sp"
    android:padding="5dp"
/>
```

Ο πηγαίος κώδικας του αρχείου main.xml

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/back" >
    <ListView android:id="@+id/in"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:stackFromBottom="true"
        android:background="@drawable/but"

        android:layout_marginRight="10dp"
        android:layout_marginTop="10dp"
        android:layout_marginBottom="10dp"
        android:transcriptMode="alwaysScroll"
        android:layout_weight="1"
    />
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginRight="10dp"
        android:layout_marginBottom="10dp"
        >
        <EditText
             android:id="@+id/edit_text_out"
            android:layout_width="0dp"
            android:layout_height="50dp"
            android:layout_weight="1"
            android:background="@drawable/but"
            android:textColor="#ffffff"

            android:layout_gravity="bottom"
        />
        <Button android:id="@+id/button_send"
            android:layout_width="75dp"
            android:layout_height="50dp"
            android:layout_marginLeft="3dp"
             android:background="@drawable/but1"
        />
    </LinearLayout>
</LinearLayout>
```

Ο πηγαίος κώδικας του αρχείου start.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/back" >

    <Button
        android:id="@+id/button1"
        android:layout_width="200dp"
        android:layout_height="50dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="150dp"
        android:background="@drawable/but"
        android:text="CHAT"
        android:textColor="#ffffff"
        android:textStyle="bold"
        android:textSize="20sp" />

    <Button
        android:id="@+id/button2"
        android:layout_width="200dp"
        android:layout_height="50dp"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/button1"
        android:layout_marginTop="100dp"
        android:background="@drawable/but"
        android:text="CONVERSATIONS"
        android:textColor="#ffffff"
        android:textStyle="bold"
        android:textSize="20sp" />
</RelativeLayout>
```

Ο πηγαίος κώδικας του αρχείου menu.xml

```
<?xml version="1.0" encoding="utf-8"?>

<menu xmlns:android="http://schemas.android.com/apk/res/android">
     <item android:id="@+id/alpha"
           android:title="@string/alpha" />
     <item android:id="@+id/time"
           android:title="@string/time" />
</menu>
```

Ο πηγαίος κώδικας του αρχείου option_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>

<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/scan"
          android:icon="@android:drawable/ic_menu_search"
          android:title="@string/connect" />
    <item android:id="@+id/discoverable"
          android:icon="@android:drawable/ic_menu_mylocation"
          android:title="@string/discoverable" />
</menu>
```