

Ατομική Διπλωματική Εργασία

**ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ΣΤΗΝ ΠΛΑΤΦΟΡΜΑ GOOGLE
ANDROID ΓΙΑ ΦΟΡΗΤΟ ΣΥΣΤΗΜΑ ΤΗΛΕΚΑΡΔΙΟΛΟΓΙΑΣ**

Στέλιος Μαϊμάρης

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ



ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2015

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

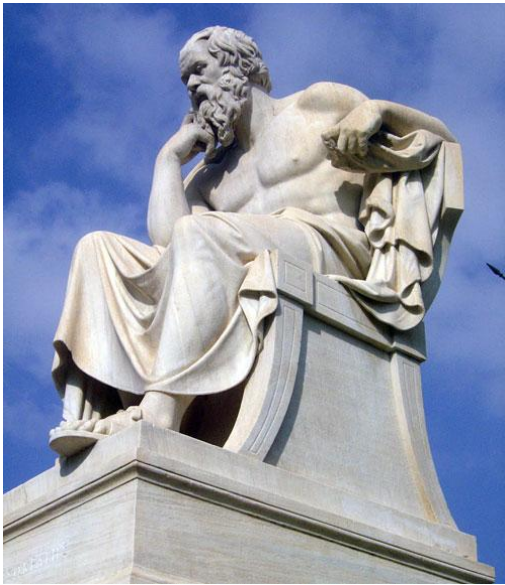
ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ΣΤΗΝ ΠΛΑΤΦΟΡΜΑ GOOGLE ANDROID ΓΙΑ ΦΟΡΗΤΟ ΣΥΣΤΗΜΑ ΤΗΛΕΚΑΡΔΙΟΛΟΓΙΑΣ

Στέλιος Μαϊμάρης

Επιβλέπων Καθηγητής
Δρ. Παττίχης Κωνσταντίνος

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των
απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του
Πανεπιστημίου Κύπρου

Μάιος 2015



Σωκράτης σε σκέψη

http://assets2.bigthink.com/system/idea_thumbnails/49715/headline/socrates.?1363573003

ἔν οἶδα ὅτι οὐδὲν οἶδα

Πλάτωνας, Απολογία Σωκράτη

Μετάφραση:

Ένα ξέρω, ότι δεν ξέρω τίποτα

Ευχαριστίες

Θέλω να εκφράσω τις θερμές μου ευχαριστίες στον καθηγητή Πανεπιστημίου Κύπρου και επιβλέπων καθηγητή της διπλωματικής μου εργασίας Δρ. Παττίχη Κωνσταντίνο καθώς και την ερευνητική ομάδα του εργαστηρίου eHealth, για την καθοδήγηση που μου έδωσαν και για την υπομονή που επέδειξαν σε όλη την διάρκεια της συνεργασίας μας.

Επίσης, μπορεί να φαίνεται δεδομένο για πολλούς από εμάς, αλλά οφείλουμε πολλά σ' αυτά τα δυο άτομα, ένα τεράστιο ευχαριστώ στους Larry Page και Sergey Brin για την μεγαλεπήβολη δημιουργία της ευρετικής μηχανής Google (όπου στηρίζεται και η πλατφόρμα Android), που κατάφεραν να φέρουν ολόκληρη την γνώση του Διαδικτύου στους οικιακούς υπολογιστές μας.

Τέλος θέλω να ευχαριστήσω την οικογένεια μου αλλά και τους φίλους μου για την ηθική και ψυχολογική υποστήριξη που μου προσέφεραν σε όλη την διάρκεια της τετραετής φοίτησης μου στο Πανεπιστήμιο Κύπρου.

Περιεχόμενα:

Κεφάλαιο 1

1.1 Τι είναι το ηλεκτροκαρδιογράφημα.....	1
1.2 Καρδία και Καρδιακός Κύκλος.....	2
1.3 Πως καταγράφεται το ηλεκτροκαρδιογράφημα.....	2
1.3.1 Απαγωγές και τρίγωνο Einthoven.....	2
1.3.2 Επάρματα (PQRST).....	5
1.4 Ηλεκτροκαρδιογραφήματα σε εφαρμογές Android.....	7
1.5 Σκοπός διπλωματικής εργασίας.....	11
1.6 Δομή διπλωματικής εργασίας	14

Κεφάλαιο 2

2.1 Τύποι κινητών συσκευών	14
2.2 Πλατφόρμα Google Android	15
2.2.1 Ιστορική Ανάδρομη	17
2.2.2 Εκδόσεις Android	18
2.2.3 Αρχιτεκτονική Android	26
2.2.3.1 Επίπεδα Αρχιτεκτονικής	28
2.2.3.2 Βιβλιοθήκες	30
2.2.3.3 Επίπεδο Android Runtime	31
2.2.3.4 Πυρήνας Linux	31
2.2.3.5 Συστατικά Στοιχεία Εφαρμογών	31
2.2.3.6 Διεπαφή Χρήστη	32
2.2.3.7 Εργαλεία Διάδρασης (Basic widgets)	36
2.3 Πλατφόρμα Fi-Star	39
2.3.1 Εισαγωγή	39
2.3.2 Αρχιτεκτονική – Enablers	40

Κεφάλαιο 3

3.1 Αποτελέσματα Συζητήσεων	45
3.2 Απαιτήσεις Ασθενών-Ιατρού	46

3.2.1	Απαιτήσεις Ιατρού	46
3.2.2	Απαιτήσεις Ασθενών	47
3.3	Συσκευή Arduino - Shimmer και Αισθητήρων	47
3.3.1	Εισαγωγή πλατφόρμα Arduino	47
3.3.2	Λειτουργία Arduino	50
3.3.3	Ονομασία και μονάδες μετρήσεις των αισθητήρων	57
3.3.4	Λειτουργία Shimmer	59
3.4	Ανάλυση δραστηριοτήτων του συστήματος	60

Κεφάλαιο 4

4.1	Χαρακτηριστικά συστήματος	62
4.2	Λειτουργίες Συστήματος	63
4.2.1	Χρήστης / Ασθενής	63
4.2.1.1	Επιλογή Γλώσσας	63
4.2.1.2	Είσοδος στο Σύστημα	64
4.2.1.3	Εγγραφή Χρήστη	64
4.2.1.4	Επιλογή πλατφόρμας Shimmer / Arduino	64
4.2.1.5	Επιλογή συσκευής Shimmer	64
4.2.1.6	Επιλογή πλατφόρμας Arduino	64
4.2.1.7	Παρουσίαση βιομετρήσεων και καρδιογραφήματος ...	65
4.2.1.8	Παρουσίαση ιστορικού μετρήσεων ανά αισθητήρα	65
4.2.1.9	Αναφορά Περίληψης Μετρήσεων Ασθενή	65
4.2.1.10	Οδηγοί Χρήσεως / Βοήθεια	65
4.2.2	Χρήστης / Ιατρός	65
4.2.2.1	Εγγραφή Ιατρού στο σύστημα	65
4.2.2.2	Είσοδος στο σύστημα	66
4.2.2.3	Παρακολούθηση δεδομένων Ασθενούς	66
4.2.2.4	Επιλογή Χρήστη	66
4.2.2.5	Απεικόνιση Βιομετρήσεων	66
4.2.2.6	Εισαγωγή Σχόλιου	66
4.2.2.7	Περίληψη μετρήσεων ασθενούς	66

Κεφάλαιο 5

5.1 Εισαγωγή	67
5.2 Αναλυτική Σχεδίαση Εφαρμογής Android	68
5.3 Σχεδίαση Βάσης Δεδομένων – Server	74
5.4 Σχεδίαση Web App	77

Κεφάλαιο 6

6.1 Εισαγωγή	79
6.2 Βήματα υλοποίησης συστήματος	80
6.3 Έλεγχοι συστήματος	81
6.3.1 Ανάνηψη από σφάλματα	81
6.3.2 Έλεγχος Ασφάλειας	82
6.3.3 Έλεγχος Ορθότητας Δεδομένων	82
6.4 Παρουσίαση εφαρμογής με Screenshots του τελικού προϊόντος	82
6.4.1 Έναρξη Εφαρμογής	82
6.4.2 Επιλογή Γλώσσας	82
6.4.3 Σύνδεση Χρήστη	83
6.4.4 Εγγραφή Χρήστη	83
6.4.5 Ηλεκτροκαρδιογράφημα από συσκευή Shimmer	86
6.4.6 Παρουσίαση Ηλεκτροκαρδιογραφήματος Shimmer	87
6.4.7 Ηλεκτροκαρδιογράφημα-βιομετρήσεις από Arduino	92
6.4.8 Παρουσίαση βιομετρήσεων από πλατφόρμα Arduino	95
6.5 Αξιολόγηση Συστήματος από χρήστες	99

Κεφάλαιο 7

7.1 Συμπεράσματα	100
7.2 Μελλοντικά Σχέδια	101

Βιβλιογραφία	102
---------------------------	------------

Παράρτημα Α

Παράρτημα Β

Περίληψη

Η ατομική διπλωματική μου εργασία ξεκίνησε με σκοπό την δημιουργία μιας εφαρμογής για κινητά τηλεφωνά, στηριγμένη στην πλατφόρμα Google Android, για την ψηφιακή επεξεργασία ηλεκτροκαρδιογραφήματος από ασθενοφόρα. Με το πέρασμα του χρόνου και μετά από πάμπολλες συζητήσεις με όλα τα εμπλεκόμενα μέλη της ομάδας του Δρ. Παττίχη, φτάνουμε τώρα στην δημιουργία ενός προϊόντος, το οποίο επεκτείνεται πέραν και από των δικών μου προσδοκιών. Η εφαρμογή Android δεν συμπεριλαμβάνει μόνο την επεξεργασία και απεικόνιση του ηλεκτροκαρδιογραφήματος, αλλά ο ασθενής έχει την δυνατότητα να μπορεί να λαμβάνει και να ενημερώνεται για επιπρόσθετες βιομετρήσεις σε πραγματικό χρόνο.

Βασικότερα, ο ασθενής έχει στην διάθεση του μια εξατομικευμένη εφαρμογή άμεσης ενημέρωσης σχετικά με την υγεία του στην άνεση του χώρου του. Έτσι, σημαντικότερος σκοπός μου είναι να προσφέρω αναβαθμισμένες υπηρεσίες ιατρικής φροντίδας και ενημέρωσης σε ασθενείς, χωρίς την αναγκαιότητα των συχνών επισκέψεων σε ιατρικά κέντρα και υπηρεσίες.

Από την άλλη μεριά, οι ιατροί θα μπορούσαν άμεσα να ενημερώνονται για την ιατρική περίληψη του ασθενή, κερδίζοντας έτσι πολύτιμο χρόνο. Έτσι, κρίνεται σημαντικό πως ο ιατρός θα έχει άμεση ανταπόκριση σε κάποιο πρόβλημα προς τον ασθενή, παρακολουθώντας τις πληροφορίες υγείας του ασθενούς, χωρίς την άμεση παρέμβαση του και σε οποιαδήποτε χρονική στιγμή.

Κατανοώντας όλα τα πιο πάνω και σύμφωνα με τις απαιτήσεις των ασθενών/ιατρών εξακριβώσαμε τις απαιτήσεις που χρειάζονται να πληρωθούν για την υποκειμενική ικανοποίηση τόσο των ασθενών όσο και των ιατρών. Από πλευράς των ιατρών, πρέπει να υπάρχει απεικόνιση του ηλεκτροκαρδιογραφήματος σε οποιαδήποτε χρονική στιγμή, καθώς και των άλλων βιομετρήσεων που περιλαμβάνουν καρδιακούς παλμούς, οξυγόνωση του αίματος, καταγραφή ηλεκτρομυογραφήματος, διαγωγιμότητας του δέρματος, ροής αέρα από τις αναπνευστικές οδούς, θερμοκρασίας σώματος, διαστολικής και συστολικής πίεσης. Όλα αυτές οι μετρήσεις θα αποθηκεύονται ως ιστορικό υγείας του ασθενούς και θα ανακτώνται ανά πάσα στιγμή από τον ιατρό.

Από πλευράς ασθενών, γεννιούνται κυρίως απαιτήσεις ευχρηστίας και ασφάλειας του συστήματος. Επιθυμούν ένα εύκολο και εύχρηστο σύστημα, με τα

κατάλληλα γραφικά περιβάλλοντα και με τις απαραίτητες οδηγίες χρήσης, χωρίς εξειδικευμένους τεχνικούς όρους. Οι ασθενείς θα είναι άτομα όλων των ηλικιών, με ή χωρίς κάποια τεχνολογική γνώση, γι' αυτό η κατάλληλη σχεδίαση της εφαρμογής αποτελεί πρωταρχικό σκοπό της διπλωματικής μου. Τέλος, ο ασθενής θα μπορεί να ενημερώνεται για το ιστορικό των μετρήσεων του, καθορίζοντας τον αισθητήρα μέτρησης που επιθυμεί.

Έπειτα, στην φάση των προδιαγραφών καθορίσαμε με ποιο τρόπο το σύστημα θα αρμόζει στους κανόνες λειτουργικότητας και ευχρηστίας, δίνοντας και τα απαραίτητα χαρακτηριστικά του. Στην φάση της σχεδίασης, έδωσα μια αναλυτική προσέγγιση της σχεδίασης της Android εφαρμογής, καθώς και για την σχεδίαση της βάσης δεδομένων με την πλατφόρμα MySQL, την σχεδίαση του Server, που θα τρέχει πάνω τα API, που καθορίζονται από τον EHR και τέλος την σχεδίαση του Web App, όπου ο ιατρός θα μπορεί να παρακολουθούσε το προφίλ του ασθενούς.

Τέλος, φτάνουμε στο βασικότερο κομμάτι της διπλωματικής, την φάση της υλοποίησης, όπου χρησιμοποίησα το ολοκληρωμένο περιβάλλον ανάπτυξης Eclipse, ενσωματώνοντας πάνω plug-ins από το Android Software Development Kit (SDK). Ο προγραμματισμός της εφαρμογής Android πραγματοποιήθηκε στην γλώσσά προγραμματισμού Java. Με την χρήση της MySQL, δημιούργησα την βάση δεδομένων, στην οποία τα δεδομένα αποθηκεύονται και ανακτώνται με μεσολάβηση του Server-API. Η ανάκτηση και αποθήκευση δεδομένων πραγματοποιείται με την χρήση της ασύρματης τεχνολογίας Wi-Fi (μιλώντας για την βάση δεδομένων), ενώ ξεχωριστά για την πλατφόρμα Shimmer χρησιμοποιείται η τεχνολογία Bluetooth, και για το Arduino, η τεχνολογία Wi-Fi hotspot. Όλα τα δεδομένα που καταγράφονται από τις συσκευές φιλτράρονται και αποθηκεύονται στην βάση δεδομένων, για την ανάλογη μεταχείριση τόσο από ασθενείς όσο και από ιατρούς.

Μετά την υλοποίηση, το σύστημα αξιολογήθηκε από το προσωπικό του Δρ. Παττίχη, με πολύ θετική ανταπόκριση. Επίσης, το σύστημα αξιολογήθηκε και από εμένα και από άτομα που κατέχουν ή δεν κατέχουν τεχνολογική γνώση, τα οποία ικανοποιήθηκαν από την ευχρηστία και λειτουργικότητα της όλης εφαρμογής. Σαν επίλογο, αξίζει να σημειωθεί πως ο καθένας μπορεί να χρησιμοποιήσει την εφαρμογή στο κινητό τηλέφωνο του, μακριά από χρονοβόρες διαδικασίες και με χαμηλό κόστος.

Κεφάλαιο 1

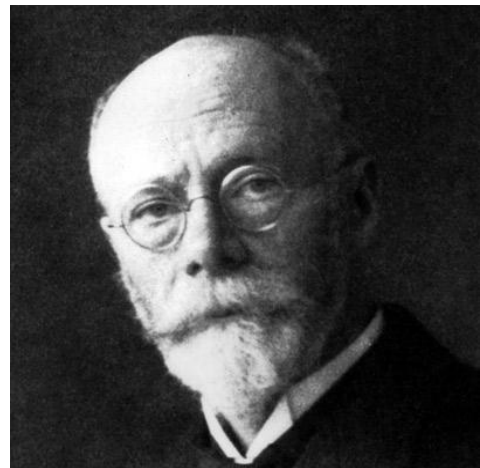
Εισαγωγή

1.1 Τι είναι το ηλεκτροκαρδιογράφημα	1
1.2 Καρδιά και Καρδιακός Κύκλος	2
1.3 Πως καταγράφεται το ηλεκτροκαρδιογράφημα	2
1.3.1 Απαγωγές και τρίγωνο Einthoven	2
1.3.2 Επάρματα (PQRST)	5
1.4 Ηλεκτροκαρδιογραφήματα σε εφαρμογές Android	7
1.5 Σκοπός διπλωματικής εργασίας	11
1.6 Δομή διπλωματικής εργασίας	11

1.1 Τι είναι το ηλεκτροκαρδιογράφημα

<http://www.einthovenlaboratory.com/wp-content/uploads/Willem-Einthoven2.jpg>Το

Ηλεκτροκαρδιογράφημα είναι η εξέταση που χρησιμοποιείται ευρέως στην καθημερινή κλινική πράξη και δίνει πολλές πληροφορίες για την ηλεκτρική δραστηριότητα και επομένως για τη λειτουργία της καρδιάς. Είναι μια απλή, γρήγορη, ανώδυνη, φτηνή εξέταση με αρκετά καλή ειδικότητα και ευαισθησία. Στην πραγματικότητα καταγράφει την ηλεκτρική δραστηριότητα των μυών της παλλόμενης καρδιάς αποδίδοντας, μέσω του ηλεκτροκαρδιογράφου, σε ειδικό χαρτί ή ψηφιακή οθόνη, (όπου ο οριζόντιος άξονας αντιστοιχεί σε χρόνο και ο κάθετος στο ηλεκτρικό δυναμικό), χαρακτηριστικά γραφήματα που ονομάζονται επάρματα.



Εικόνα 1.1: Willem Einthoven
<http://www.einthovenlaboratory.com/wp-content/uploads/Willem-Einthoven2.jpg>

Ο Willem Einthoven ήταν αυτός που ανακάλυψε τις αρχές καταγραφής και ερμηνείας του ηλεκτροκαρδιογραφήματος από το 1893, κερδίζοντας το Νομπέλ ιατρικής το 1924.

1.2. Καρδιά και Καρδιακός Κύκλος

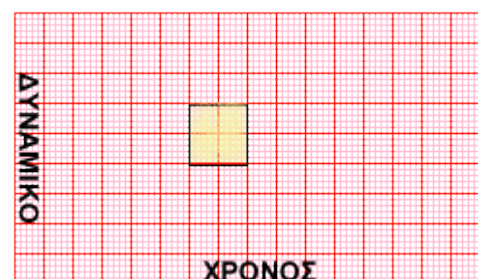
Η καρδιά είναι ένα μυώδες όργανο που συστέλλεται ρυθμικά λειτουργώντας σαν αντλία αίματος με τη βοήθεια του κυκλοφορικού συστήματος. Περιέχει ένα σύστημα εξειδικευμένων μυϊκών ινών, το ερεθισματοαγωγό σύστημα, οι οποίες μπορούν να αυτοδιεγείρονται. Η διέγερση αυτή εξαπλώνεται σε ολόκληρο το μυοκάρδιο και έχει ως αποτέλεσμα τη συστολή του μυοκαρδίου. Το σύστημα αυτό καλείται σύστημα παραγωγής και αγωγής της διέγερσης και αποτελείται από τον φλεβόκομβο, τον κολποκοιλιακό κόμβο, το δεμάτιο του Hiss με τα δύο του σκέλη και τις ίνες Purkinje. Το ερεθισματοαγωγό σύστημα βρίσκεται υπενδοκαρδιακά και η διέγερση γίνεται από μέσα προς τα έξω.

Ο καρδιακός κύκλος αποτελεί την περίοδο από την έναρξη ενός καρδιακού κτύπου, έως την έναρξη του επόμενου κτύπου, ο οποίος περιλαμβάνει τη συστολή, τη σύσπαση των κόλπων και των κοιλιών, που εξωθεί το αίμα και τη διαστολή, την περίοδο κατά την οποία οι κοιλότητες επαναπληρώνονται με αίμα. Φυσιολογικά, οι κόλποι συσπώνται αμέσως πριν από τις κοιλίες. Ο συνήθης κύκλος διαρκεί 0,8 δευτερόλεπτα με 60 έως 85 καρδιακούς κτύπους περίπου ανά λεπτό σε ενήλικα σε κατάσταση ηρεμίας. Η κολπική συστολή διαρκεί 0,1 δευτερόλεπτα, η κοιλιακή συστολή 0,3 δευτερόλεπτα και η διαστολή 0,4 δευτερόλεπτα. Παρ' ότι φαίνεται πως η καρδιά λειτουργεί ακατάπαυστα, στην πραγματικότητα ηρεμεί για ικανό χρονικό διάστημα μέχρι την έναρξη του επόμενου καρδιακού κύκλου.

1.3. Καταγραφή Καρδιογραφήματος

1.3.1 Απαγωγές και τρίγωνο Einthoven

Το ηλεκτροκαρδιογράφημα λαμβάνεται με τη χρήση ειδικής συσκευής, η οποία ονομάζεται ηλεκτροκαρδιογράφος. Στην ουσία πρόκειται για ένα βολτόμετρο που καταγράφει τις διαφορές δυναμικού των ερεθισμάτων, τα οποία παράγονται στην καρδιά και φθάνουν έως την επιφάνεια του σώματος. Αποτελείται από μια κεντρική μονάδα και 10 καλώδια: 4 πλάκες τοποθετούνται στα άνω και κάτω άκρα και 6 ηλεκτρόδια στο θωρακικό



Εικόνα 1.2: Απεικόνιση

Ηλεκτροκαρδιογραφήματος

http://upload.wikimedia.org/wikipedia/commons/a/ab/Empty_ECG_paper.svg

τοίχωμα του εξεταζόμενου (καρδιογράφος 12 απαγωγών).

Μέσω της κεντρικής μονάδας ρυθμίζεται η ενεργοποίηση ή η απενεργοποίηση της συσκευής (ON/OFF), η ταχύτητα καταγραφής (συνήθως 25 mm/sec), η χρήση φίλτρων (για την αποφυγή του «θορύβου»), το μέγεθος και ο τρόπος καταγραφής (auto/manual) του ηλεκτροκαρδιογραφήματος. Εντός της κεντρικής μονάδας υπάρχει ακίδα (συνήθως με μαύρο μελάνι) και τοποθετείται ειδικό χιλιοστομετρικό χαρτί (με τετραγωνάκια που έχουν πλευρά μήκους 1 χιλιοστού). Το χαρτί φέρει δύο άξονες: έναν οριζόντιο (καταγραφή του χρόνου) και έναν κάθετο (καταγραφή του δυναμικού)

Το χαρτί κινείται με ταχύτητα 25 mm/sec (εάν απαιτείται πιο εκτεταμένη καταγραφή, η ταχύτητα ρυθμίζεται στα 50 mm/sec). Επομένως, το 1 mm αντιστοιχεί σε 0,04 sec και τα 5 mm σε 0,20 sec. Στον κάθετο άξονα καταγράφεται το μέγεθος των δυναμικών: το 1 mm αντιστοιχεί σε 0,1 mV και σε ειδικές περιπτώσεις με το μισό ή το διπλάσιο

Ηλεκτροκαρδιογραφικές Απαγωγές

Στο ηλεκτροκαρδιογράφημα καταγράφονται οι κυματομορφές 12 απαγωγών:

- 6 απαγωγές των ακρών για τα ηλεκτρικά δυναμικά της καρδιάς που φθάνουν στα άκρα (I, II, III, aVR, aVL, aVF).
- 6 προκάρδιες απαγωγές για τα δυναμικά που φθάνουν στην πρόσθια επιφάνεια του θώρακα (V1, V2, V3, V4, V5, V6).

Οι απαγωγές I, II, III ονομάζονται διπολικές και καταγράφουν τη διαφορά δυναμικού μεταξύ των άκρων. Στα σύγχρονα ηλεκτροκαρδιογραφήματα δεν χρειάζεται να πραγματοποιούνται και οι 12 απαγωγές, εντούτοις 3 απαγωγές είναι υπεραρκετές.

Τύποι Απαγωγών:

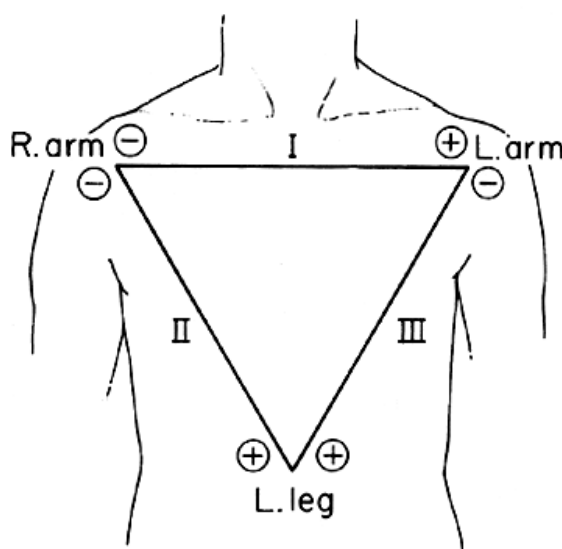
- **Απαγωγή I:** Για την καταγραφή της απαγωγής I, τοποθετείται το αρνητικό ηλεκτρόδιο στο δεξιό άνω άκρο και το θετικό ηλεκτρόδιο στο αριστερό άνω άκρο. Κατά συνέπεια, όταν το ερέθισμα κινείται από τα δεξιά προς τα αριστερά, το καταγραφόμενο έπαρμα είναι θετικό. Η απαγωγή I καταγράφει ηλεκτρικά ερεθίσματα που προέρχονται από το πλάγιο τμήμα της καρδιάς.
- **Απαγωγή II:** Για την καταγραφή της απαγωγής II, το αρνητικό ηλεκτρόδιο του ηλεκτροκαρδιογράφου τοποθετείται στο δεξιό άνω άκρο και το θετικό στο αριστερό κάτω άκρο. Επομένως, όταν το ερέθισμα κινείται από τα δεξιά προς τα

αριστερά, ο ηλεκτροκαρδιογράφος καταγράφει θετικό έπαρμα. Η απαγωγή II καταγράφει δυναμικά από το κατώτερο τμήμα της καρδιάς.

- **Απαγωγή III:** Για την καταγραφή της απαγωγής III, το αρνητικό ηλεκτρόδιο του ηλεκτροκαρδιογράφου τοποθετείται στο αριστερό άνω άκρο και το θετικό στο αριστερό κάτω άκρο. Αυτό σημαίνει ότι ο ηλεκτροκαρδιογράφος καταγράφει θετικό έπαρμα όταν το ερέθισμα κινείται από το αριστερό άνω προς το αριστερό κάτω άκρο. Η απαγωγή III καταγράφει την ηλεκτρική λειτουργία από το κατώτερο τμήμα της καρδιάς.

Το τρίγωνο Einthoven

Όπως βλέπουμε από την εικόνα, γύρω από την περιοχή της καρδιάς υπάρχει ένα ισόπλευρο τρίγωνο, το οποίο ονομάζεται τρίγωνο Einthoven. Το τρίγωνο αυτό υποδεικνύει τα δυο άνω άκρα και το αριστερό κάτω άκρο, που ουσιαστικά αποτελούν τις τρεις γωνίες του τριγώνου που περιβάλλει την καρδιά. Οι δυο γωνίες στο άνω άκρο του τριγώνου παριστάνουν τα σημεία, στα οποία τα δυο άνω άκρα πραγματοποιούν ηλεκτρική σύνδεση με τα υγρά που περιβάλλουν την καρδιά, ενώ η κάτω γωνία αποτελεί το σημείο στο οποίο το αριστερό κάτω άκρο συνδέεται με αυτά τα υγρά.



Εικόνα 1.3: Τρίγωνο Einthoven
<http://www.icbetapp.com/wp-content/uploads/2013/02/3032F02.gif>

Οι **απαγωγές V1, V2, V3, V4, V5, V6** ονομάζονται προκάρδιες ή θωρακικές απαγωγές. Η καταγραφή των αντίστοιχων κυματομορφών πραγματοποιείται με τη χρήση 6 ηλεκτροδίων, τα οποία τοποθετούνται στο πρόσθιο θωρακικό τοίχωμα. Επειδή η καρδιά βρίσκεται πολύ κοντά στο τοίχωμα, κάθε προκάρδια απαγωγή καταγράφει ηλεκτρικά δυναμικά που προέρχονται από περιοχές του μυοκαρδίου αμέσως κάτω από το ηλεκτρόδιο. Για αυτόν τον λόγο, σχετικά μικρές ανωμαλίες των κοιλιών (ιδιαίτερα στο

πρόσθιο κοιλιακό τοίχωμα) προκαλούν συχνά εκσεσημασμένες αλλοιώσεις στις καταγραφές των προκάρδιων απαγωγών.

1.3.2 Τα επάρματα και η ονομασία τους

Όταν μια απαγωγή βλέπει το ερέθισμα να έρχεται προς αυτή καταγράφει ένα θετικό έπαρμα, πάνω δηλαδή από την ισοηλεκτρική γραμμή, ενώ όταν το βλέπει να φεύγει καταγράφει ένα αρνητικό έπαρμα, δηλαδή κάτω από την ισοηλεκτρική γραμμή.



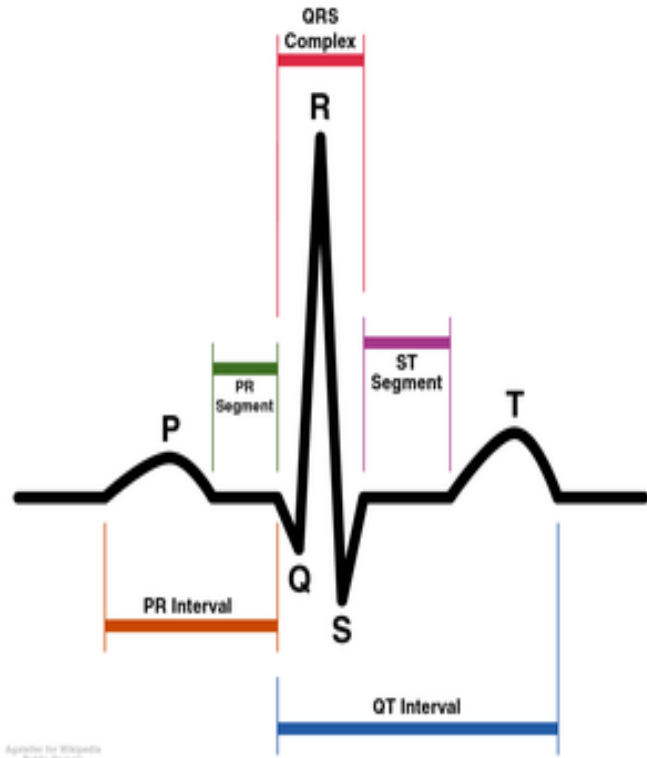
Εικόνες 1.4-1.5: Θετικό/Αρνητικό Έπαρμα
[http://upload.wikimedia.org/wikipedia/commons/thu
mb/9/94/ECG_Vector.svg/1024px-
ECG_Vector.svg.png](http://upload.wikimedia.org/wikipedia/commons/thumb/9/94/ECG_Vector.svg/1024px-ECG_Vector.svg.png)

Η διέγερση των κόλπων ξεκινά φυσιολογικά από το φλεβόκομβο και έχει συνιστάμενη κατεύθυνση προς τον κολποκοιλιακό κόμβο. Το έπαρμα που παράγεται ονομάζεται έπαρμα **P** και είναι θετικό στην απαγωγή II και αρνητικό στην απαγωγή αVR. Το έπαρμα που παράγεται από την εκπόλωση των κόλπων έχει πάντα την ίδια ονομασία είτε είναι θετικό είτε είναι αρνητικό.

Κατά τη διέλευσή της από τον κολποκοιλιακό κόμβο η διέγερση καθυστερεί και στο ηλεκτροκαρδιογράφημα καταγράφεται ισοηλεκτρική γραμμή, το διάστημα **PQ**. Ακολουθεί η διέγερση των κοιλιών η οποία παράγει ένα σύμπλεγμα επαρμάτων, το σύμπλεγμα **QRS**, που προκύπτει από τη διαδοχική διέγερση:

1. Του μεσοκοιλιακού διαφράγματος. Η συνιστάμενη κατεύθυνση των διπόλων είναι από αριστερά προς τα δεξιά. Έτσι η απαγωγή V1 καταγράφει θετικό έπαρμα ενώ η απαγωγή V6 αρνητικό.
2. Και των δύο κοιλιών με συνιστάμενη κατεύθυνση των διπόλων προς τα αριστερά. Έτσι η V1 καταγράφει ένα βαθύ αρνητικό έπαρμα ενώ η V6 ένα ψηλό θετικό.
3. Της υπερκοιλιακής ακρολοφίας με κατεύθυνση προς τα δεξιά. Επομένως η V1 καταγράφει θετικό έπαρμα και η V6 αρνητικό.

- **Έπαρμα P:** εκπόλωση των κόλπων.
- **Διάστημα PR interval:** εκπόλωση των κόλπων έως την αρχή της εκπόλωσης των κοιλιών.
- **Διάστημα PR segment:** διέλευση ερεθίσματος από τον κολποκοιλιακό κόμβο.
- **Σύμπλεγμα QRS:** εκπόλωση των κοιλιών.
- **Διάστημα QT:** πλήρης κύκλος εκπόλωσης και επαναπόλωσης των κοιλιών.



Εικόνα 1.6: PQRSΤ Έπαρμα
<http://upload.wikimedia.org/wikipedia/commons/0/09/ECG-PQRST%2Bpopis.svg>

- **Διάστημα ST:** τέλος εκπόλωσης έως την αρχή της επαναπόλωσης των κοιλιών.
- **Έπαρμα T:** επαναπόλωση των κοιλιών.

Το σύμπλεγμα QRS ονομάζεται πάντοτε έτσι, ανεξάρτητα από τα ονόματα των επιμέρους επαρμάτων που το αποτελούν. Η ονομασία των επιμέρους επαρμάτων όμως ακολουθεί συγκεκριμένους κανόνες.

1. Το πρώτο αρνητικό έπαρμα του συμπλέγματος ονομάζεται έπαρμα Q.
2. Τα θετικά επάρματα ονομάζονται πάντα R. Αν είναι περισσότερα από ένα ονομάζονται R, R1, R2 κλπ
3. Το αρνητικό έπαρμα που ακολουθεί θετικό έπαρμα ονομάζεται έπαρμα S

Μετά την ολοκλήρωση της εκπόλωσης των κοιλιών ακολουθεί ισοηλεκτρική γραμμή, το διάστημα ST. Την εκπόλωση του μυοκαρδίου ακολουθεί η επαναπόλωση του. Η επαναπόλωση των κόλπων παράγει μικρό έπαρμα που δεν γίνεται συνήθως αντιληπτό στο ηλεκτροκαρδιογράφημα διότι συμπίπτει χρονικά με την εκπόλωση των κοιλιών και την παραγωγή του συμπλέγματος QRS.

Η επαναπόλωση όμως των κοιλιών παράγει έπαρμα που μπορεί να είναι θετικό η αρνητικό, πάντοτε όμως ονομάζεται έπαρμα T.

1.4 Ηλεκτροκαρδιογράφημα σε εφαρμογές Android

Παρατηρούμε πως η τεχνολογία των κινητών τηλεφώνων άρχισε να ανταποκρίνεται και σε ελεύθερους επαγγελματίες υγείας και σε άτομα με ιδιαίτερες εξειδικεύσεις στον επιστημονικό χώρο. Ολοένα και δημιουργούνται καινοτόμες εφαρμογές, που ενισχύουν και υποβοηθούν τα άτομα αυτά να αντεπεξέλθουν στις εργασίες τους, δίνοντας το πλεονέκτημα της ευελιξίας και μεταφοράς της συσκευής.

Ψάχνοντας στο Google Store, μπορεί ο καθένας να πραγματοποιήσει εύρεση αυτών των εφαρμογών. Οι εφαρμογές απευθύνονται κυρίως σε φοιτητές ιατρικής ή άλλων επιστημονικών πεδίων, που επιθυμούν να ενημερώνονται για τις νέες εξελίξεις στον χώρο της ιατρικής επιστήμης.

Σημείωση: Όλα τα παρακάτω screenshots πάρθηκαν από το
Google Store, 2015. [<https://play.google.com/store/apps?hl=el>]

Μερικές εφαρμογές Ηλεκτροκαρδιογραφήματος είναι:

Καρδιογράφος



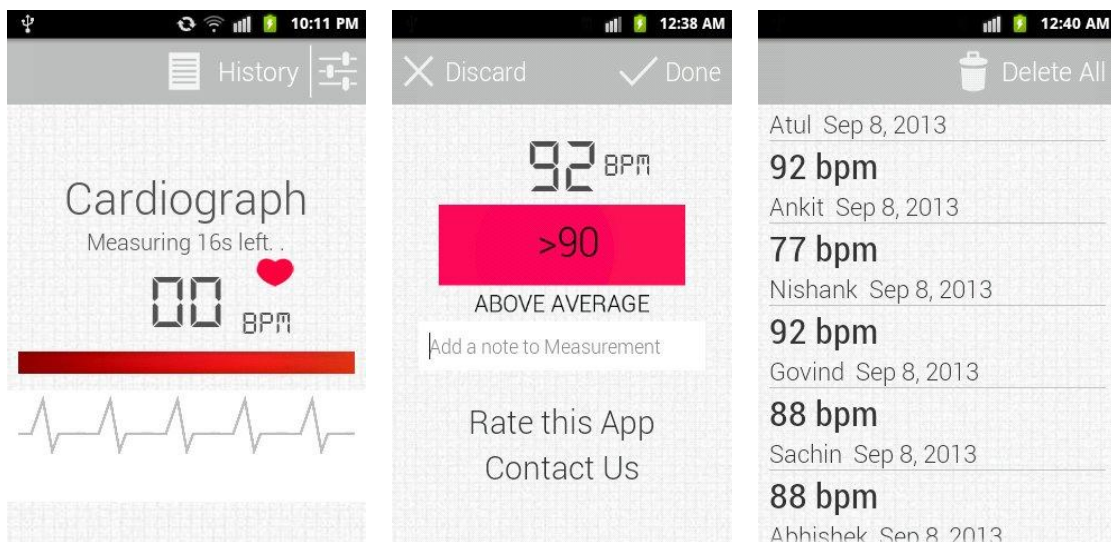
Περιγραφή:

Ο Καρδιογράφος είναι μια εφαρμογή που μετράει τον καρδιακό ρυθμό, αποθηκεύοντας τα αποτελέσματα για μελλοντική αναφορά και σε πολλαπλά προφίλ ανθρώπων. Χρησιμοποιεί την ενσωματωμένη κάμερα ή τον ειδικό αισθητήρα της κινητής συσκευής για να υπολογίσει τον ρυθμό της καρδιάς.

Χαρακτηριστικά της εφαρμογής:

- Μέτρηση καρδιακού ρυθμού
- Παρακολούθηση των αποτελεσμάτων (γραφική αναπαράσταση ECG)
- Πολλαπλά προφίλ ώστε διαφορά άτομα να χρησιμοποιούν την εφαρμογή σε μια κοινόχρηστη συσκευή.
- Εύχρηστο περιβάλλον αλληλεπίδρασης με τον χρήστη

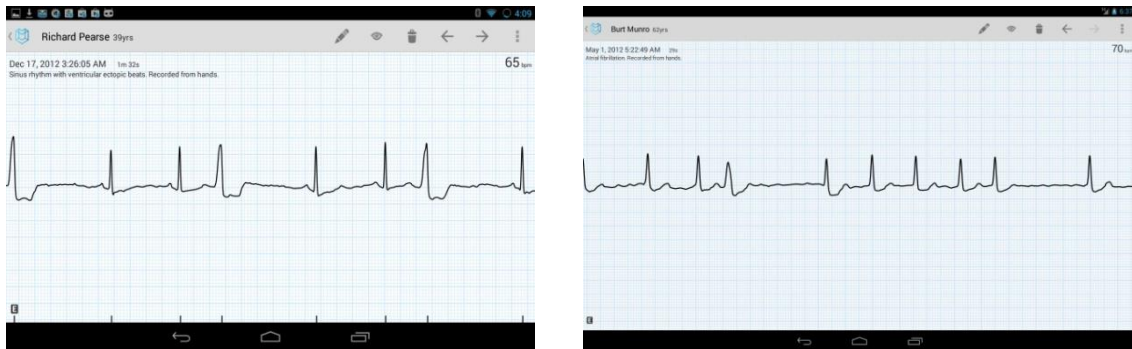
Cardiograph Heart Rate Monitor



Περιγραφή / Χαρακτηριστικά:

Η εφαρμογή διαβάζει τον παλμό του χρήστη, με την βοήθεια της κάμερας και του flash του κινητού τηλέφωνου, όπου ανιχνεύεται η εναλλαγή του χρώματος στο δακτυλικό αποτύπωμα. Διατηρεί εγγραφές με το ιστορικό των προηγούμενων καρδιογραφημάτων του χρήστη, μαζί με το επιπρόσθετο χαρακτηριστικό της αποστολής αυτών των δεδομένων στον προσωπικό ιατρό του χρήστη μέσω ηλεκτρονικού ταχυδρομείου (email).

Alive ECG



Περιγραφή/Χαρακτηριστικά:

Η εφαρμογή διαβάζει τον παλμό του χρήστη, όπως προανέφερα πιο πάνω, με την μονή διαφορά πως η παρουσίαση του ηλεκτροκαρδιογραφήματος γίνεται αναπαραστατικά σε ψηφιακό ηλεκτροκαρδιογραφικό χαρτί. Τα δεδομένα αποθηκεύονται σαν ιστορικό είτε στη συσκευή είτε στην ιστοσελίδα app.alivecor.com.

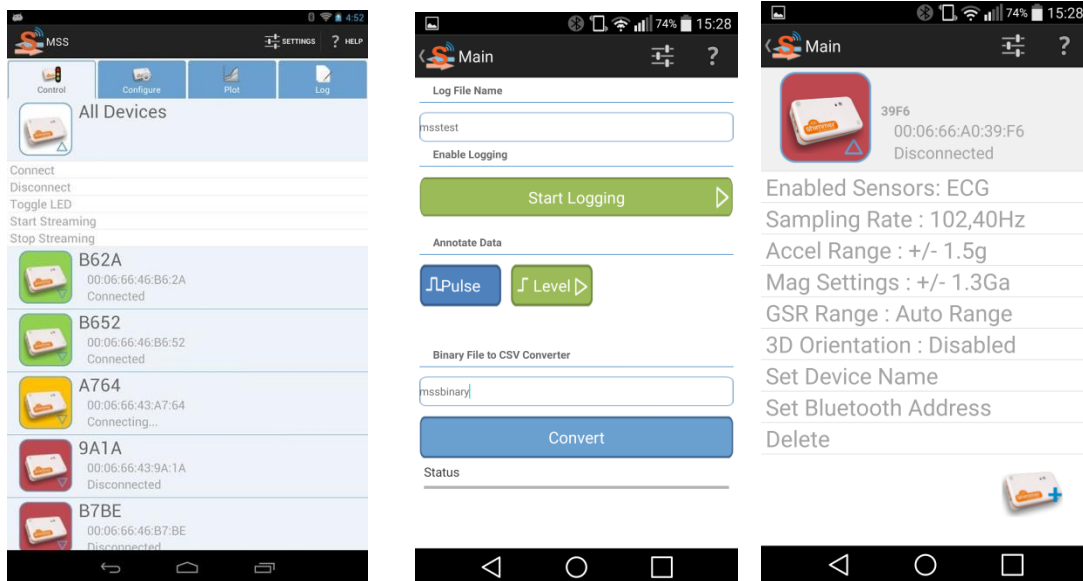
E-Health Sensor Platform



Περιγραφή:

Αυτή η εφαρμογή εκτός από την καταμέτρηση του ηλεκτροκαρδιογραφήματος, επίσης συλλέγει και παρουσιάζει και άλλες βιομετρήσεις, με βάση κάποια είδη αισθητήρων από την πλατφόρμα της E-Health. Η εφαρμογή είναι ειδικά σχεδιασμένη για τις πλατφόρμες Arduino και Raspberry Pie.

Multi Shimmer Sync



Περιγραφή:

Αυτή η εφαρμογή από την εταιρεία Shimmer επιτρέπει την διαμόρφωση και συγχρονισμό συλλογής δεδομένων από πολλαπλούς αισθητήρες καταγραφής ηλεκτροκαρδιογραφήματος των συσκευών Shimmer2R. Η εφαρμογή είναι ιδανική για τους χρήστες που επιθυμούν να αναπτύξουν εφαρμογές, όπου απαιτείται ταυτόχρονη λήψη δεδομένων από έναν αριθμό μονάδων.

Με μια σειρά από λειτουργίες και τα εργαλεία συλλογής δεδομένων, οι χρήστες μπορούν να επιλέξουν ρυθμό δειγματοληψίας, να αποθηκεύσουν και να ρυθμίσουν τα δεδομένα καταγραφής για μελλοντική χρήση, να σχολιάσουν τα δεδομένα, και να διασφαλίσουν την λειτουργικότητα, την αποτελεσματικότητα και την ευκολία χρήσης των δεδομένων. Επίσης, δύναται η χρήση απεικόνισης σε πραγματικό χρόνο του ηλεκτροκαρδιογραφήματος, με την δυνατότητα αποθήκευσης και αναπαραγωγής του. Η δοκιμαστική έκδοση προσφέρει πλήρη λειτουργικότητα καταγραφής όταν χρησιμοποιείται με μία μόνο συσκευή Shimmer.



1.5 Σκοπός της διπλωματικής εργασίας

Η ατομική διπλωματική μου εργασία έχει υλοποιηθεί με σκοπό την παρακολούθηση καρδιογραφημάτων και βιομετρήσεων από ασθενείς, με την χρήση της πλατφόρμας Google Android. Αυτό αποσκοπεί στο να γίνεται η άμεση ενημέρωση του γιατρού, σχετικά με το ηλεκτροκαρδιογράφημα και διαφόρων άλλων ιατρικών μετρήσεων ενός ασθενή, χωρίς να χρειάζεται να είναι πάρων ο γιατρός, αναβαθμίζοντας έτσι τις υπηρεσίες ιατρικής παρακολούθησης και φροντίδας.

Οι ιατροί δεν έχουν την δυνατότητα να ενημερώνονται άμεσα για την κατάσταση ενός ασθενή, ο οποίος μπορεί να βρίσκεται σε σοβαρή κατάσταση. Ο κάθε ιατρός καλείται αρκετά συχνά να εξετάσει διάφορους ασθενείς, παίρνοντας κάθε φορά νέες μετρήσεις. Πέρα από την ταλαιπωρία για επανεξετάσεις, χάνεται και πολύτιμος χρόνος τόσο και από πλευράς ιατρού, ο οποίος διαθέτει ένα βεβιασμένο πρόγραμμα μπροστά του, όσο και από πλευράς ασθενή, του οποίου χάνεται επίσης ο ελεύθερος του χρόνος. Μ' αυτήν την εφαρμογή, ο ιατρός θα ενημερώνεται για το καρδιογράφημα και τις βιομετρήσεις του ασθενή σχεδόν άμεσα (αφού τα δεδομένα θα αποστέλλονται σε μια κοινή βάση δεδομένων, με δυνατότητα ανάκτησης και απεικόνισης τους) και το σημαντικότερο εν κίνηση, ακόμα και αν δεν βρίσκεται στο νοσοκομείο, έχοντας άμεση ανταπόκριση σε κάποιο πρόβλημα αρρυθμίας της καρδιάς ή σωματικής βλάβης του ασθενή.

Λόγω της δυνατότητας ευελιξίας και εύκολης μεταφοράς του κινητού τηλεφώνου, η επιλογή για υλοποίηση της εφαρμογής σε πλατφόρμας Google Android υπήρξε καθοριστική (και για άλλους πολλούς λόγους, τους οποίους θα αναφέρω παρακάτω).

1.6 Δομή Διπλωματικής εργασίας

Κεφάλαιο 1° :

Το πρώτο κεφάλαιο αποτελεί μια εισαγωγή της διπλωματικής μου εργασίας. Αρχικά αναφέρει τι είναι το ηλεκτροκαρδιογράφημα και πως το μετρούμε ή υπολογίζουμε, έπειτα αναφέρει κάποιες εφαρμογές Android που υπάρχουν ήδη στο Google Store και έχουν να κάνουν με τρόπους υπολογισμού και παρουσίασης καρδιογραφήματος, και τέλος θέτει τον σκοπό και την δομή της διπλωματικής.

Κεφάλαιο 2° :

Το δεύτερο κεφάλαιο αφορά τις κινητές συσκευές και πιο συγκεκριμένα τις κινητές συσκευές με λογισμικό Google android. Αρχικά γίνεται μια ανασκόπηση στους τύπους των κινητών συσκευών και στα χαρακτηριστικά τους. Ακολουθεί μια εισαγωγή για την πλατφόρμα Google Android, έπειτα τα χαρακτηριστικά , οι εκδόσεις, η αρχιτεκτονική, ο κύκλος ζωής των εφαρμογών και τέλος τα επιμέρους διαδραστικά εργαλεία, που θα χρησιμοποιηθούν στις εφαρμογές. Επίσης, δίνεται μια σύντομη παρουσίαση της αρχιτεκτονικής της πλατφόρμας Fi-Star (Future Internet Social Technological Alignment in HealthCare), η οποία μελλοντικά θα στηρίζει σημαντικά την τεχνολογική ιατρική περίθαλψη.

Κεφάλαιο 3° :

Το τρίτο κεφάλαιο παρουσιάζει την πρώτη φάση του κύκλου ζωής ενός συστήματος, την φάση της ανάλυσης απαιτήσεων. Αρχικά παρουσιάζονται τα αποτελέσματα των συζητήσεων μεταξύ των μεντόρων μου, εξακριβώνοντας τις απαιτήσεις, που θα καθορίζουν το σύστημα. Ακολουθώντας γίνεται εκτενής αναφορά στις απαιτήσεις από πλευράς ιατρών, ασθενών αλλά και συσκευών. Από πλευράς συσκευών, γίνεται αναφορά στις δυο συσκευές που θα απασχολήσουν την διπλωματική μου, που είναι η συσκευή ηλεκτροκαρδιογραφήματος της Shimmer και η πλατφόρμα Arduino Uno, όπου περιγράφονται οι λειτουργίες τις κάθε μιας, καθώς και η χρήση των διαφόρων αισθητήρων, που ενσωματώνονται πάνω στην πλατφόρμα Arduino. Τέλος, αναλύεται και γενικότερα η δραστηριότητα του συστήματος.

Κεφάλαιο 4° :

Το τέταρτο κεφάλαιο παρουσιάζει την δεύτερη φάση του κύκλου ζωής ενός συστήματος, την φάση της ανάλυσης προδιαγραφών. Εξηγούνται τα χαρακτηριστικά που πρέπει να έχει η εφαρμογή και μια περιγραφή των λειτουργιών της.

Κεφάλαιο 5° :

Το πέμπτο κεφάλαιο παρουσιάζει την τρίτη φάση του κύκλου ζωής ενός συστήματος, την φάση της σχεδίασης του συστήματος. Αναφέρεται πως έγινε η αρχιτεκτονική και η αναλυτική σχεδίαση του συστήματος. Ειδικότερα, αναφέρετε ο τρόπος δημιουργίας της

εφαρμογής Android, καθώς και της βάσης δεδομένων στην MySQL και η σύνδεση της με τον Server(σχεδιασμός των API).

Κεφάλαιο 6° :

Το έκτο κεφάλαιο παρουσιάζει τα βήματα που ακολουθήθηκαν για να υλοποιηθεί το σύστημα καθώς και τους διάφορους ελέγχους που έγιναν για να διαπιστωθεί η ορθότητα του. Τέλος παρουσιάζετε και η αξιολόγηση του συστήματος τόσο από τον ιατρό όσο και από ασθενείς.

Κεφάλαιο 7° :

Το κεφάλαιο αυτό αποτελεί το τελευταίο κεφάλαιο της εργασίας μου. Εδώ παρουσιάζονται τα συμπεράσματα της εργασίας και γίνεται αναφορά σε μελλοντικές εργασίες που μπορούν να γίνουν.

Κεφάλαιο 2

Κινητές Συσκευές Android

2.1 Τύποι κινητών συσκευών	14
2.2 Πλατφόρμα Google Android	15
2.2.1 Ιστορική Ανάδρομη	17
2.2.2 Εκδόσεις Android	18
2.2.3 Αρχιτεκτονική Android	26
2.2.3.1 Επίπεδα Αρχιτεκτονικής	28
2.2.3.2 Βιβλιοθήκες	30
2.2.3.3 Επίπεδο Android Runtime	31
2.2.3.4 Πυρήνας Linux	31
2.2.3.5 Συστατικά Στοιχεία Εφαρμογών	32
2.2.3.6 Διεπαφή Χρήστη	35
2.2.3.7 Εργαλεία Διάδρασης (Basic widgets)	36
2.3 Πλατφόρμα Fi-Star	39
2.3.1 Εισαγωγή	39
2.3.2 Αρχιτεκτονική – Enablers	40

2.1 Τύποι κινητών συσκευών

Με την εξέλιξη της τεχνολογίας, οι κινητές συσκευές έγιναν αναπόσπαστο μέρος της ζωής του σύγχρονου ανθρώπου. Μετά από την κατασκευή των πρώτων τρανζίστορ και μικροεπεξεργαστών, καθώς και με την εξέλιξη των τεχνολογιών δικτύων και τηλεπικοινωνιών, εμφανίζονται για πρώτη φορά στο παγκόσμιο κοινό οι πρώτες φορητές συσκευές τηλεπικοινωνίας, στις αρχές της δεκαετίας του 1980. Με την πάροδο του χρόνου, δημιουργήθηκαν διάφορων τύπων κινητών συσκευών, τις οποίες περιλαμβάνουν τα κινητά τηλέφωνα (Cellular or Mobile phones), τις συσκευές PDA (Portable Digital/data Assistants) και τα smart phones («έξυπνα» τηλέφωνα).

Τα smart phones προκύπτουν από τον συνδυασμό των κινητών συσκευών με τις συσκευές PDA, δίνοντας απεριόριστες λειτουργικότητες στην συσκευή όπως:

την δυνατότητα των τηλεφωνικών κλήσεων και αποστολής μηνυμάτων καθώς επίσης και την δυνατότητα διαχείρισης προσωπικών πληροφοριών, της πλοήγησης στο διαδίκτυο, του ελέγχου του ηλεκτρονικού ταχυδρομείου, την αποθήκευση και διαχείριση αρχείων, φωτογραφιών, βίντεο κτλ σε πιο ανεπτυγμένο βαθμό απ' ότι στα απλά κινητά και αυξημένες επιλογές εφαρμογών για ψυχαγωγία και όχι μόνο. Και όλα αυτές οι λειτουργίες συρρικνωμένες στο να φοράνε στην τσέπη του παντελονιού μας, δίνοντας έμφαση στο σημείο της φορητότητας της συσκευής καθώς και της κινητικότητας. Τα σύγχρονα κινητά τηλέφωνα δεν έχουν τίποτα να ζηλέψουν από ένα ηλεκτρονικό υπολογιστή.

Τα τελευταία χρόνια βιώνουμε μια επανάσταση των «έξυπνων» κινητών τηλεφώνων, τα οποία ενσωματώνουν δυνατότητες που δεν υπήρχαν στα μέχρι σήμερα κινητά τηλέφωνα. Περιλαμβάνουν ολοκληρωμένο λειτουργικό σύστημα, ισχυρό επεξεργαστή ικανό για εκτέλεση χρονοβόρων υπολογισμών, δυνατότητα σύνδεσης στο διαδίκτυο είτε μέσω ασυρμάτων δικτύων (Wi-Fi) είτε δικτύων τρίτης γενιάς (3G / 4G networks), κάμερα υψηλής ανάλυσης, συστήματα εντοπισμού θέσης (GPS), πληθώρα αισθητήρων κ.α. Κάθε είδους αισθητήρας μπορεί να υπάρχει σε ένα Smartphone, όπως αισθητήρας θερμοκρασίας περιβάλλοντος, αισθητήρας μέτρησης θορύβου καθώς και πολλοί άλλοι.

2.2 Πλατφόρμα Google Android

Το λειτουργικό σύστημα της Google για smart phones έκανε την εμφάνισή του το 2008. Το Android είναι μια πλατφόρμα ανοιχτού λογισμικού που αναπτύχθηκε από προγραμματιστές της ίδιας και προγραμματιστές των Intel, HTC, ARM, Motorola και Samsung. Αποτελεί ένα λειτουργικό σύστημα που συναντάται κυρίως σε ενσωματωμένα συστήματα όπως είναι τα κινητά τηλέφωνα, τα tablet αλλά και άλλες κινητές συσκευές. Λόγω αυτής της περιορισμένης χρήσης πόρων από την εγκατάσταση του σε μια τόσο μικρή ηλεκτρονική συσκευή, όπως είναι το κινητό τηλέφωνο, η εταιρεία κατασκευής του λογισμικού, καθώς και ο προγραμματιστής που καλείται να δημιουργήσει εφαρμογές, οφείλουν να αναθεωρήσουν κάποιες δεδομένες λειτουργίες,, που θα συναντούσαμε σε ένα τυπικό ηλεκτρονικό υπολογιστή. Ενώ μέχρι τώρα, τα προγράμματα που εκτελούνταν στους προσωπικούς μας υπολογιστές, είχαν στην διάθεσή τους «απεριόριστη» μνήμη και επεξεργαστική ισχύ πλέον αυτό παύει να ισχύει.

Έτσι, ο προγραμματιστής πρέπει να έχει στο μυαλό του, την φράση “Keep it simple, stupid”, ώστε οι εφαρμογές να μπορούν να λειτουργούν ομαλά μέσα στο «συρρικνωμένο» χώρο του κινητού τηλεφώνου και ο χρήστης να μπορεί να αλληλεπιδρά αποδοτικότερα μ’ αυτές.

Το Android αποτελεί πλατφόρμα ανοιχτού λογισμικού, λόγω της χρήσης του διασήμευ πυρήνα Linux. Κληρονομώντας όλα τα χαρακτηριστικά ασφαλείας του Linux και όλες τις διαχειριστικές τεχνικές μνήμης και επεξεργαστή που αυτό διαθέτει, το Android, καθίσταται ένα αρκετά αξιόπιστο λειτουργικό σύστημα. Έτσι, ο κάθε προγραμματιστής έχει την δυνατότητα να προσαρμόζει τις δίκες του ανάγκες και να προσφέρει υποκειμενικά το δικό του μερίδιο στο κτίσιμο του λογισμικού. Επιπρόσθετα, δύναται και η δυνατότητα να επέμβει και στο εσωτερικό των ανώτερων επιπέδων του λειτουργικού συστήματος. Με μηδενικό κόστος εργαλείων ανάπτυξης δίνεται η δυνατότητα σε οποιονδήποτε να δοκιμάσει την ανάπτυξη εφαρμογών για Android με μοναδικό περιορισμό τον χρόνο εξοικείωσης με το περιβάλλον ανάπτυξης. Βέβαια στην αγορά υπάρχουν και άλλα λειτουργικά συστήματα που προορίζονται για κινητά τηλέφωνα.

Exhibit 1: Global Smartphone OS Shipments and Market Share in Q2 2014

Global Smartphone Operating System Shipments (Millions of Units)	Q2 '13	Q2 '14
Android	186.8	249.6
Apple iOS	31.2	35.2
Microsoft	8.9	8.0
BlackBerry	5.7	1.9
Others	0.5	0.5
Total	233.0	295.2
Global Smartphone Operating System Marketshare %	Q2 '13	Q2 '14
Android	80.2%	84.6%
Apple iOS	13.4%	11.9%
Microsoft	3.8%	2.7%
BlackBerry	2.4%	0.6%
Others	0.2%	0.2%
Total	100.0%	100.0%
Total Growth Year-over-Year %	48.9%	26.7%

Εικόνα 2.1 Στατιστικά Στοιχεία Πωλήσεων Κινητών για το έτος 2014
<http://cdn.bgr.com/2014/07/strategy-analytics-q2-2014-smartphone-sales-1.jpg>

Πιο πάνω παρουσιάζεται η κατανομή των λειτουργικών συστημάτων κινητών τηλεφώνων στην παγκόσμια αγορά. Παρατηρώντας την τεραστία αύξηση ποσοστού κατοχής τηλεφώνων Android συγκρίνοντας το δεύτερο τρίμηνο του 2013 με το δεύτερο τρίμηνο του 2014, συμπεραίνουμε πως οι πώλησης Android συσκευών

βρίσκονται στις πρώτες προτιμήσεις των καταναλωτών, ξεπερνώντας κατά πολύ τις πωλήσεις των γνωστών Iphone της Apple. Επομένως, με ασφάλεια μπορούμε να πούμε ότι αποτελεί το πιο δημοφιλές λειτουργικό σύστημα κινητών συσκευών σήμερα. Αξίζει να σημειωθεί ότι καθημερινά ενεργοποιούνται περίπου εννιακόσιες χιλιάδες συσκευές Android παγκοσμίως.

Σύμφωνα με επίσημα στοιχεία η ιστοσελίδα της Google, Android Market (πλέον Google Play) περιλαμβάνει αυτή την στιγμή περισσότερες από 1,460,800 εφαρμογές, ενώ ο συνολικός αριθμός κινητών συσκευών που πωλήθηκαν συνολικά ξεπερνά τις 1,175,000 πωλήσεις.

2.2.1 Ιστορική Ανάδρομη

Το λειτουργικό σύστημα Android αναπτύχθηκε από την εταιρεία Android Inc. που εξαγοράστηκε πλήρως από την Google τον Αύγουστο του 2005. Επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με την χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας την συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google. Δύο χρόνια αργότερα, τον Νοέμβριο του 2007, έκανε την εμφάνισή της, η Open Handset Alliance, μια κοινοπραξία 48 τηλεπικοινωνιακών εταιρειών, εταιρειών λογισμικού και κατασκευής hardware, ανάμεσα τους ονόματα όπως LG, Intel, HTC, Motorola και NVidia. Στόχος της ήταν η δημιουργία ανοικτών προτύπων για κινητές συσκευές. Παράλληλα, δημοσίευσαν το πρώτο τους προϊόν, το Android, μια πλατφόρμα βασισμένη στον πυρήνα του Linux (Linux kernel) 2.6. Η Google έδωσε στην δημοσιότητα και το μεγαλύτερο μέρος του πηγαίου κώδικα του Android, υπό τους όρους της Apache License. Η πρώτη 'early look' έκδοση του Android SDK δημοσιεύτηκε το Νοέμβριο του 2007, ενώ το πρώτο smart phone που έκανε χρήση λειτουργικού Android ήταν το G1 της T-Mobile.

Οι συσκευές Android άρχισαν να διαδίδονται με γρήγορο ρυθμό κυρίως λόγω της δυνατότητας της πλατφόρμας να εκμεταλλεύεται το μοντέλο cloud computing αλλά και της έμφυτης υποστήριξης για συνεργασία με μία σχεσιακή βάση δεδομένων (SQLite). Ένα από τα πιο δυνατά σημεία του Android είναι πως η ανάπτυξη εφαρμογών γίνεται σχεδόν αποκλειστικά στη Java, ίσως την πιο δημοφιλή, πιο ολοκληρωμένη και καλύτερα δομημένη γλώσσα που υπάρχει σήμερα. Επιπλέον, η υλοποίηση Android εφαρμογών μπορεί να γίνει με μικρό κόστος από την πλευρά του προγραμματιστή. Ο υποψήφιος Android developer θα πρέπει να διαθέτει έναν προσωπικό υπολογιστή με

οποιοδήποτε λειτουργικό σύστημα και ένα κινητό Android για το σωστό έλεγχο των εφαρμογών του.

2.2.2 Εκδόσεις Google Android

Η ιστορία του λειτουργικού συστήματος Android ξεκινά τον Νοέμβριο του 2007 με την πρώτη δοκιμαστική (beta) έκδοσή του. Η πρώτη εμπορική έκδοση του Android κυκλοφόρησε έναν χρόνο και έναν μήνα αργότερα, τον Σεπτέμβριο του 2009.

Ακολούθησαν αρκετές αναβαθμισμένες εκδόσεις του Android, κάθε μία προσθέτοντας νέα χαρακτηριστικά και λειτουργίες.

Για κάποιον άγνωστο λόγο, κάθε έκδοση του Android φέρει και μια κωδική ονομασία ενός γλυκού edέσματος.



Εικόνα 2.2: Εκδόσεις Android

<http://teks.co.in/site/blog/wp-content/uploads/2014/03/Android.jpg>

Android 1.5 – Cupcake

Αξιοσημείωτες αλλαγές στο λειτουργικό Android θα παρουσιαστούν στο Cupcake και περιλαμβάνουν αλλαγές στο σύστημα διαχείρισης των μεταφορτώσεων (download manager), το framework, Bluetooth, το λογισμικό συστήματος, το ραδιόφωνο και το σύστημα τηλεφωνίας, εργαλεία προγραμματισμού, το κυρίως σύστημα και διάφορες εφαρμογές, καθώς και πληθώρα διορθώσεις σφαλμάτων.

Στις 30 Απριλίου 2009, κυκλοφόρησε η επίσημη ενημέρωση έκδοσης 1.5 για το Android.



Εικόνα 2.3: Έκδοση Cupcake

<http://theinspirationroom.com/daily/design/2013/9/android-cupcake-logo.jpg>

Αποτελείται από πολλά νέα χαρακτηριστικά και βελτιώσεις στο γραφικό περιβάλλον:

- Δυνατότητα καταγραφής κινούμενης εικόνας με την χρήση της αντίστοιχης λειτουργίας του τηλεφώνου
- Μεταφόρτωση αρχείων βίντεο στο YouTube και εικόνων στο Picasa κατευθείαν από το τηλέφωνο
- Επανασχεδιασμένο λογισμικό πληκτρολογίου με λειτουργία αυτόματης συμπλήρωσης κειμένου
- Δυνατότητα αυτόματης σύνδεσης ασύρματης συσκευής ακουστικού Bluetooth εφόσον εντοπιστεί σε μια συγκεκριμένη απόσταση
- Νέα widgets και φάκελοι που μπορούν να τοποθετηθούν στην επιφάνεια εργασίας
- Διευρυμένη λειτουργία αντιγραφής/επικόλλησης

Android 1.6 – Donut

Η έκδοση Donut κυκλοφόρησε τον Σεπτέμβριο του 2009 και στηρίχτηκε στα χαρακτηριστικά της έκδοσης 1.5 που επέκτεινε κάποια από αυτά, παρόλο που δεν έγιναν σημαντικές αλλαγές οι οποίες ήταν ορατές στον χρήστη, οι αλλαγές στην βάση του λειτουργικού συστήματος προετοίμασαν το έδαφος για μελλοντικές εντυπωσιακές αλλαγές. Αλλαγές που πραγματοποιήθηκαν είναι:

- Βελτίωση της φωνητικής και μη αναζήτησης ώστε να περιλαμβάνει το ιστορικό των επισκέψεων στο διαδίκτυο, τις επαφές και το διαδίκτυο (με χρήση του Google.com)
- Δυνατότητα σύνθεσης ομιλίας σε διαφορετικές γλώσσες
- Ευκολότερη αναζήτηση εφαρμογών στην αγορά της Google και προβολή στιγμιότυπων από την εφαρμογή
- Δυνατότητα επιλογής περισσότερων τις μιας φωτογραφιών για διαγραφή



Εικόνα 2.4: Έκδοση Donut
http://www.androidguys.com/wp-content/uploads/2009/09/android_donut_logo.jpg

- Αναβάθμιση των τεχνολογιών που χρησιμοποιούνταν για CDMA/EVDO, 802.1x και VPN
- Υποστήριξη για οθόνες ανάλυσης WVGA

Android 2.0/2.0.1/2.1 – Éclair

Η έκδοση Éclair ήταν ένα αρκετά σημαντικό βήμα για την εξέλιξη του λειτουργικού συστήματος σε σχέση με τις προηγούμενες εκδόσεις. Δημιουργήθηκε στα τέλη του 2009, βελτιώνοντας χαρακτηριστικά όπως την πλοήγηση, την υπηρεσία Google Maps και τις διεπιφάνειες χρήστη.



Εικόνα 2.5: Έκδοση Éclair

http://img2.wikia.nocookie.net/__cb20130520200626/logo/pedia/images/2/25/Eclair_2009.png

Επίσης δημιουργήθηκε η υπηρεσία Google Maps Navigation η οποία έδινε την δυνατότητα στις κινητές συσκευές να συγκριθούν με συσκευές πλοήγησης GPS. Η έκδοση 2.0 αντικαταστήθηκε γρήγορα από την 2.0.1 η οποία βγήκε λίγους μήνες αργότερα για να διορθώσει τα προβλήματα της έκδοσης 2.0. στην συνέχεια τον Ιούνιο του 2010 δημοσιοποιήθηκε η έκδοση 2.1 και προσέφερε καλύτερες διεπιφάνειες χρήστη με καλύτερα γραφικά.

Κύριες αλλαγές με βάση προηγούμενες εκδόσεις είναι:

- Πλέον ο χρήστης μπορεί να εισάγει πολλαπλούς λογαριασμούς από διάφορες υπηρεσίες και οι αντίστοιχες εφαρμογές μπορούν να χρησιμοποιούν τα στοιχεία αυτά για να συγχρονίζουν το περιεχόμενό τους
- Αναβάθμιση Bluetooth Υποστήριξη του πρωτοκόλλου Bluetooth 2.0
- Δυνατότητα αναζήτησης στα μηνύματα του χρήστη και αυτόματη διαγραφή μηνυμάτων που χρονολογικά περνούν κάποιο καθορισμένο όριο
- Υποστήριξη flash, ψηφιακό ζουμ, scene mode λειτουργίας, ρύθμιση ισορροπίας λευκού, εισαγωγή εφέ χρώματος και δυνατότητα macro focus
- Βελτίωση της ταχύτητας δακτυλογράφησης στο πληκτρολόγιο με χρήση έξυπνου λεξικού που μαθαίνει από τις πληκτρολογήσεις του χρήστη
- Βελτιώσεις του λειτουργικού για επίτευξη καλύτερων επιδόσεων και ανανέωση της διεπαφής χρήστη

- Αναβάθμιση του Google Maps στην έκδοση 3.1.2
- Κίνηση του φόντου της επιφάνειας εργασίας καθώς ο χρήστης αλλάζει οθόνες

Android 2.2 – Froyo

Το Android 2.2 ανακοινώθηκε τον Μάιο του 2010 από την Google στο Σαν Φρανσίσκο. Η μεγαλύτερη αλλαγή που εισήγαγε ήταν η σημαντική αύξηση της επεξεργαστικής ισχύς της κινητής συσκευής, που το είχε εγκατεστημένο.

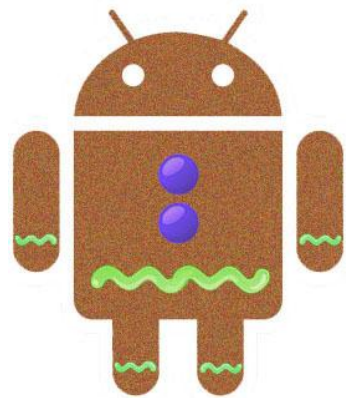
- Ο διερμηνευτής Dalvik έγινε πέντε φορές πιο γρήγορος συγκρινόμενος με τον διερμηνευτή της έκδοσης 2.1 του Android, η μηχανή V8 του Chrome βοηθάει τον browser να φορτώνει γρηγορότερα σελίδες με JavaScript περιεχόμενο και αλλαγές στην διαχείριση μνήμης στο επίπεδο του πυρήνα αύξησαν ακόμη περισσότερο τις επιδόσεις
- Επίσης παρείχε υποστήριξη για το Adobe Flash που έδινε την δυνατότητα στις κινητές συσκευές να παρέχουν σύνδεση στο Διαδίκτυο σε όποια συσκευή επιθυμούσαν (Wi-Fi spot).
- Επιπρόσθετα, δύναται η δυνατότητα αποθήκευσης δεδομένων στο cloud και ανάκτησή τους από τους χρήστες ανεξάρτητα από την συσκευή στην οποία βρίσκονται.
- Τέλος, πλέον οι εφαρμογές μπορούν να εγκατασταθούν και σε άλλες μνήμες της συσκευής πέραν της εσωτερικής της μνήμης

Εικόνα 2.6: Έκδοση Froyo
http://wccftech.com/images/news/Android-2.2-Froyo-Build-FRF91/android_froyo.jpg



Android 2.3/2.4 – Gingerbread

Το Android 2.3 δημοσιεύθηκε τον Δεκέμβριο του 2010. Έφερε βελτιώσεις στις διεπιφάνειες χρήστη δίνοντας καλύτερη αίσθηση στον χειρισμό τους. Στην έκδοση αυτή σχεδιάστηκε από την αρχή το εικονικό πληκτρολόγιο, βελτιώθηκαν οι δυνατότητες πλοήγησης και έγιναν βήματα προς την καλύτερη διαχείριση ενέργειας. Παρόλα αυτά το android δεν άλλαξε ουσιαστικά από την προηγούμενη του έκδοση.



Εικόνα 2.7: Έκδοση Gingerbread
<http://img.talkandroid.com/uploads/2010/10/Android-Gingerbread.jpg>

- Το Android έχει το δικαίωμα να τερματίσει οποιαδήποτε εφαρμογή τρέχει στο παρασκήνιο και καταναλώνει πολύ ενέργεια ή τρέχει στο προσκήνιο για περισσότερο χρόνο του κανονικού (συνήθως πέντε δευτερόλεπτα) προκειμένου διασφαλίσει την μέγιστη διάρκεια λειτουργίας
- Παρέχει υποστήριξη για NFC(Near Field Communication) τεχνολογία. Πρόκειται για μια τεχνολογία ανταλλαγής πληροφοριών ανάμεσα σε κινητές συσκευές φέρνοντας την μια δίπλα στην άλλη.
- Η υποστήριξη του πρωτοκόλλου SIP δίνει την δυνατότητα ενσωμάτωσης, στις εφαρμογές, δυνατοτήτων τηλεφωνίας

Android 3.0/3.1/3.2 – Honeycomb

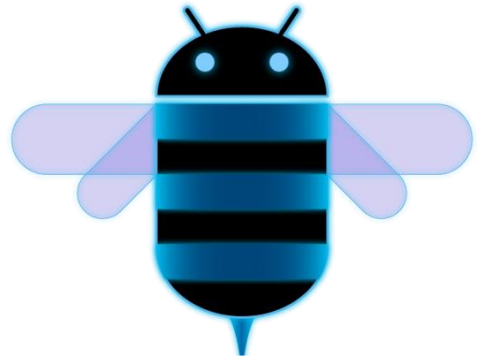
Το Android 3.0 κυκλοφόρησε τον Φεβρουάριο του 2011. Ήταν η πρώτη έκδοση του λειτουργικού συστήματος που δημιουργήθηκε αποκλειστικά για κινητές συσκευές τύπου tablet που έφερε σημαντικές αλλαγές στις διεπιφάνειες χρήστη. Επίσης αναβαθμίστηκαν εφαρμογές που παρέχει η Google όπως το Gmail και το Google talk. Σημαντική ήταν η βελτίωση του υλικού των

συσκευών μέσω των χαρακτηριστικών του Honeycomb.

Αξίζει να αναφερθούμε στο γεγονός ότι η Google εφάρμοσε ένα διαφορετικό τρόπο στην διανομή του κώδικα του λειτουργικού συστήματος της στους κατασκευαστές κινητών συσκευών. Αυτό είχε ως αποτέλεσμα να περιοριστεί η ελευθερία του λογισμικού το οποίο θεωρείται ανοικτό και χρησιμοποιούμενο από όλους. Βελτιώσεις στο Android 3.0 ανακοινώθηκαν τον Μάιο του 2011 (Android 3.1 και Android 3.2).

Σημαντικές αλλαγές είναι:

- Ελκυστική 3D-looking διεπαφή χρήστη πλήρως ρυθμιζόμενη ως προς το περιεχόμενό της



Εικόνα 2.8: Έκδοση Honeycomb
<https://tctechcrunch2011.files.wordpress.com/2011/02/honeycombbbee.png>

- Σχεδιασμός εκ νέου του πληκτρολογίου λόγω τις μεγαλύτερης διαθέσιμης επιφάνειας στα tablets
- Δυνατότητα επισκόπησης των ταυτόχρονα εκτελούμενων εφαρμογών και μετάβαση σε οποιαδήποτε από αυτές ο χρήστης επιθυμεί
- Πλήρως συμβατό με εφαρμογές που έχουν γραφτεί για παλαιότερες εκδόσεις Android
- Δυνατότητα αναγνώρισης κινήσεων ακόμη και στα widgets ώστε ο χρήστης να μπορεί να πλοηγείται στο περιεχόμενο τους
- Υποστήριξη επεξεργαστών με περισσότερους του ενός επεξεργαστικούς πυρήνες
- Υποστήριξη οθονών μεγαλύτερου μεγέθους

Android 4.0 – Ice Cream Sandwich

Η συνέχεια στις εκδόσεις του λειτουργικού συστήματος Android ανακοινώθηκε από την Google τον Μάιο του 2011 και κυκλοφόρησε στις 19 Οκτωβρίου 2011, βασισμένη στον πυρήνα του Linux 3.0.1. Έφερε χαρακτηριστικά του Android 3.0 ,που όπως προαναφέραμε ήταν αποκλειστικά για tablets, στα smart phones και ήρθε για να ενώσει του δυο κόσμους συσκευών, αυτόν των tablets και αυτόν των smart phones.



Εικόνα 2.9: Έκδοση Ice Cream Sandwich
http://cdn.redmondpie.com/wp-content/uploads/2011/10/ICS_Android.jpg

Σε αυτήν την έκδοση έγιναν πολλές βελτιώσεις σε είδη υπάρχουσες εφαρμογές και χαρακτηριστικά του συστήματος όπως ολοκλήρωση των κοινωνικών δικτύων στην εφαρμογή των επαφών, επιτάχυνση του UI από το υλικό και εγγραφή 1080p βίντεο. Επιπλέον διορθώθηκαν αρκετά σφάλματα στον κώδικα και έγιναν πολλές βελτιώσεις που διασφάλισαν την σταθερότητα της έκδοσης αυτής.

Ακολούθησαν βελτιωμένες εκδόσεις όπως η 4.0.2, 4.0.3 και 4.0.4 που κυκλοφόρησε τον Μάρτιο του 2012.

Κύριες αλλαγές είναι:

- Βελτιωμένο πληκτρολόγιο, διόρθωση από λάθη σε προηγούμενες εκδόσεις

- Βελτιωμένη λειτουργία αντιγραφής/επικόλλησης
- Ξεκλείδωμα του κινητού με είσοδο το πρόσωπο του χρήστη (Face Unlock)
- Επιλογή που προειδοποιεί τους χρηστές εάν έφτασαν σε κάποιο κρίσιμο σημείο χρησιμοποίησης δεδομένων υπηρεσιών (Data usage)
- Android Beam αποτελεί NFC λειτουργία, που επιτρέπει στους χρηστές να ανταλλάσσουν δεδομένα που έχουν να κάνουν με σελιδοδείκτες ιστοσελίδων, πληροφορίες επαφών, YouTube videos κ.α.
- Αλλαγές στο user interface για επιτάχυνση του υλικού του κινητού.

Android 4.1 – Jelly Bean

Το Android 4.1 είναι η πιο γρήγορη και πιο απλή έκδοση και ανακοινώθηκε στις 27 Ιουνίου 2012. Το Jelly Bean βελτιώνει την απλότητα και την ομορφιά του Android 4.0 εισάγοντας μια νέα εμπειρία του Google search. Η έκδοση αυτή είναι βασισμένη στον πυρήνα του Linux 3.1.10 και κύριος σκοπός ήταν η βελτίωση της διεπαφής χρήστη.



Εικόνα 2.10: Έκδοση Jelly Bean
<http://support1.alcatelonetouch.com/images/b58a4409c00590944fae9358464342fd1377706002.jpg>

Βελτιώσεις που έγιναν περιλαμβάνουν την σημαντική βελτίωση των ηχητικών επιδόσεων, της μπάρας ενημερώσεων, την προσβασιμότητα των λειτουργιών και άλλα

Στις 24 Ιουλίου 2013 δημοσιεύτηκε η τελευταία έκδοση , η Android 4.3. Προσφέρει καλύτερες διεπαφές του χρήστη, το Photo Sphere, επανασχεδιασμένη και βελτιωμένη εφαρμογή για κάμερα, πιο καλό πληκτρολόγιο και το Google Now.

Κύριες αλλαγές είναι:

- Ανανεωμένο και ομαλότερο περιβάλλον χρηστή
- Μεταφορά δεδομένων από το Android Beam μέσω Bluetooth
- Ενίσχυση στην προσβασιμότητα συνδέσεων Διαδικτύου και μεταφοράς δεδομένων

- Μειωμένης ενεργειακής λειτουργίας Bluetooth
- Βελτιωμένα γραφικά παιχνιδιών με υποστήριξη OpenGL ES 3.0
- Υποστήριξη ανάλυσης 4K (για την έκδοση 4.3)
- Ενίσχυση ασφάλειας δεδομένων και απόδοσης χρήσης

Android 4.4 – 4.4.4 – KitKat

Το Android 4.4 αποτελεί την προσπάθεια της Google να βελτιστοποιήσει την χρήση του λογισμικού Android σε μεγαλύτερο εύρος κινητών συσκευών, με την μονή προϋπόθεση, η μνήμη RAM του κινητού να διατίθεται από 512 MB και άνω. Δημοσιεύθηκε στις 31 Οκτώβριου 2013 και η σχεδίαση της

νέας έκδοσης προσδίδει περισσότερη ταχύτητα και ομαλότητα μεταξύ των διεπιφανειών και με καλύτερη διαχείριση της μνήμης και της ενεργειακής παροχής του κινητού τηλέφωνα. Η τελική έκδοση Android 4.4 .4 δημοσιεύτηκε στις 19 Ιουνίου 2014, με έμφαση περισσότερο στην ενίσχυση της ασφάλειας.

Εικόνα 2.11: Έκδοση Kit Kat
<http://media.engadget.com/img/products/488/agzq/agzq-800.jpg>



Κύριες αλλαγές είναι:

- Βελτιστοποιημένη απόδοση με λιγότερες προδιαγραφές περιλαμβανομένου της υποστήριξης zRam
- Υποστήριξη απομακρυσμένης εκτύπωσης (Wireless Printing)
- Εισαγωγή του Android Runtime (ART) ως εναλλακτικό περιβάλλον εκτέλεσης εφαρμογών, αντικαθιστώντας την εικονική μηχανή Dalvik
- Περαιτέρω ενίσχυση σε θέματα ασφάλειας και απόδοσης

Android 5.0 – Lollipop

Η νεότερη έκδοση του λογισμικού Android δημοσιεύτηκε στις 12 Νοέμβριου 2014 και χαρακτηρίστηκε ως το πιο «ώριμο» λογισμικό που δημιούργησε μέχρι τώρα η Google. Παρατηρήθηκαν αλλαγές ως προς την χρηστικότητα των διεπιφανειών, όπου η κάθε αλληλεπίδραση με στοιχεία της οθόνης προσθέτει μια ομαλή κίνηση των διεπιφανειών στο χώρο, μαζί με έντονες αποχρώσεις και χρήση γραφικών σκιών. Όλα

αυτά συνθέτουν ένα πιο φιλικό προς τον χρήστη περιβάλλον, με βελτιώσεις στις ειδοποιήσεις, απόδοσης επεξεργασίας CPU και απόδοσης διαχείρισης της κατανάλωσης μπαταρίας. Ακόμη, γίνεται λόγος χρησιμοποίησης του λογισμικού στις μελλοντικές 64-bit κινητές συσκευές που θα κατακλείσουν την παγκόσμια αγορά σε λίγα χρόνια.



Εικόνα 2.12: Έκδοση Lollipop

[http://i.kinja-img.com/gawker-](http://i.kinja-img.com/gawker-media/image/upload/eanj3pdvkrspfsanjhns.jpg)

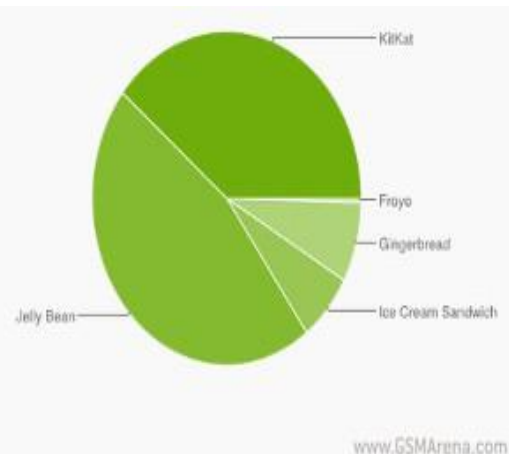
[media/image/upload/eanj3pdvkrspfsanjhns.jpg](http://i.kinja-img.com/gawker-media/image/upload/eanj3pdvkrspfsanjhns.jpg)

Νεότερες εκδόσεις αρχίζουν να κατακλύζουν την αγορά όπως η έκδοση 5.0.1 στις 2

Δεκεμβρίου 2014 και η έκδοση 5.0.2 στις

19 Δεκεμβρίου 2014. Το SDK για την ανάπτυξη εφαρμογών για το Android Lollipop είναι ήδη διαθέσιμο σε οποιονδήποτε επιθυμεί να αναπτύξει εφαρμογές για τις τελευταίας τεχνολογίας κινητές συσκευές και ολοένα περισσότεροι κατασκευαστές το υιοθετούν για τις κινητές συσκευές τους.

Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	7.8%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	6.7%
4.1.x	Jelly Bean	16	19.2%
4.2.x		17	20.3%
4.3		18	6.5%
4.4	KitKat	19	39.1%



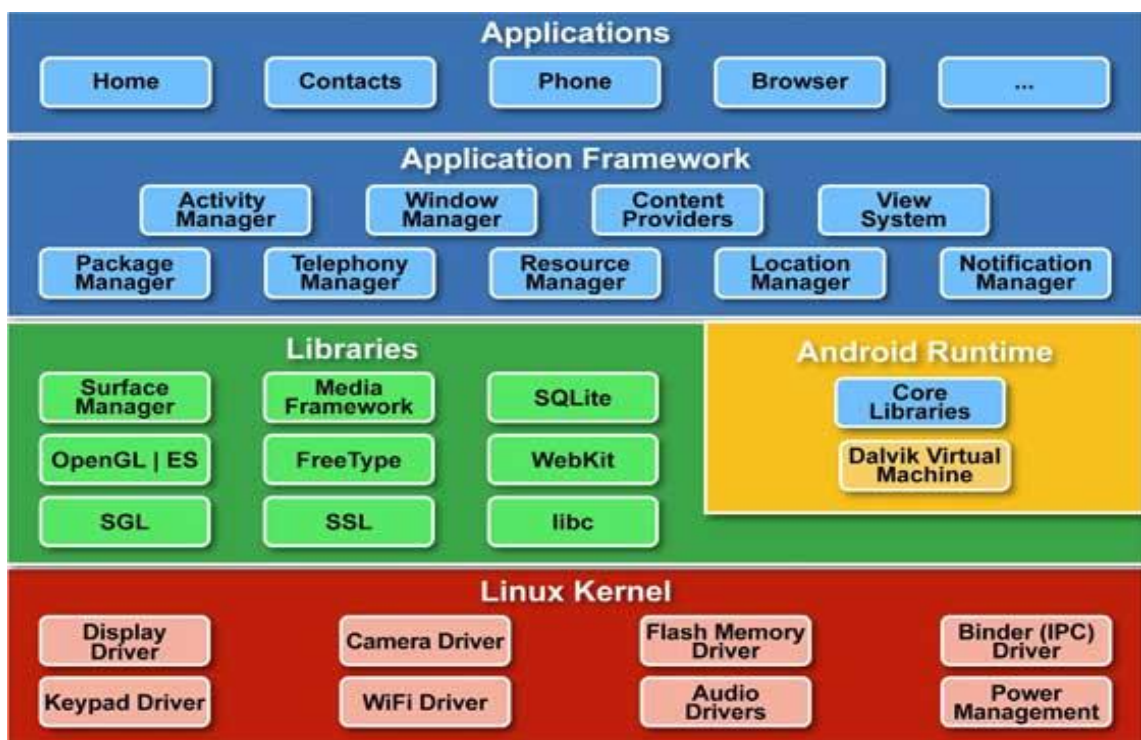
Με βάση τα στατιστικά στοιχεία, οι εκδόσεις Android που βρίσκονται στο παγκόσμιο προσκήνιο είναι το νεότερο λογισμικό **Kit Kat**, ακολουθημένο από το διάσημο λογισμικό **Jelly Bean**. (Ιανουάριος 2015- Πηγή: gsmarena.com)

2.2.3. Αρχιτεκτονική Android

Το Android είναι ένα λειτουργικό σύστημα και σαν λειτουργικό σύστημα σκοπός του είναι να παρέχει ένα επίπεδο αφαιρετικότητας ανάμεσα στο υλικό και τον χρήστη. Για να το καταφέρει αυτό, το Android, αποτελείται από μία στοίβα λογισμικών τμημάτων (software stack) με ξεκάθαρους και αυστηρά καθορισμένους ρόλους

Στον πυρήνα της πλατφόρμας Android βρίσκεται ένα Linux Kernel, το οποίο είναι υπεύθυνο για τη διαχείριση των device drivers, τον έλεγχο πρόσβασης στους πόρους του συστήματος, τη διαχείριση μνήμης και τις λοιπές υπηρεσίες που παρέχει ένα λειτουργικό σύστημα. Στους device drivers συγκαταλέγονται αυτοί της οθόνης, του Wi-Fi, της κάμερας, του ήχου κ.α.

Ένα επίπεδο επάνω βρίσκονται οι native βιβλιοθήκες του συστήματος που είναι γραμμένες σε C++ και περιλαμβάνουν το OpenGL, την SQLite, την Media library κ.α. Οι εφαρμογές που τρέχουν στο κινητό μπορούν να έχουν πρόσβαση στις βιβλιοθήκες αυτές μέσω της Dalvik JVM. Όπως έχει ήδη αναφερθεί, οι εφαρμογές Android είναι γραμμένες σε Java και άρα για να τρέξουν χρειάζονται το αντίστοιχο περιβάλλον. Όπως λοιπόν για να εκτελέσουμε μία εφαρμογή σε ένα PC είναι απαραίτητο να είναι εγκατεστημένο το κατάλληλο JRE (Java Runtime Environment), για τις εφαρμογές Android τον ρόλο του JRE παίζει η Dalvik VM.



Εικόνα 2.13: Αρχιτεκτονική Android
(http://www.webgeometrics.com/wp-content/uploads/2014/02/android_architecture.jpg)

Τα επίπεδα αρχιτεκτονικής από τα οποία αποτελείται το Android είναι αυτό των εφαρμογών (applications), το περιβάλλον των εφαρμογών (application framework), οι βιβλιοθήκες (libraries), ο κώδικας του Android (Android Runtime) και στο τελευταίο επίπεδο είναι ο πυρήνας του Linux.

2.2.3.1 Τα επίπεδα και το περιεχόμενο της αρχιτεκτονικής Android

Εφαρμογές (Applications)

Περιλαμβάνει μια σειρά από βασικές εφαρμογές όπως εξυπηρετητή ηλεκτρονικού ταχυδρομείου, εφαρμογή για την αποστολή SMS, εφαρμογή ημερολογίου, εφαρμογή πλοήγησης διαδικτύου, εφαρμογή επαφών κ.α. Όλες οι εφαρμογές είναι γραμμένες στην γλώσσα προγραμματισμού JAVA.

Περιβάλλον εφαρμογών (Application Framework)

Είναι η βάση πάνω στην οποία αναπτύσσονται όλες οι εφαρμογές Android, οι προγραμματιστές ξοδεύουν τις περισσότερες ώρες δουλεύοντας με αυτό το επίπεδο. Τα κυριότερα μέρη του περιβάλλοντος εφαρμογών είναι :

1. Διαχειριστής Δραστηριοτήτων (Activity Manager)

Αλληλεπιδρά με το σύνολο των δραστηριοτήτων που εκτελούνται στο σύστημα. Δίνει πληροφορίες σχετικά, με την διαθέσιμη μνήμη του συστήματος, με διεργασίες οι οποίες είναι σε λάθος κατάσταση, με εργασίες που ο χρήστης έχει πρόσφατα ξεκινήσει ή επισκεφθεί, με τρέχουσες διεργασίες, με μία συγκεκριμένη υπηρεσία που εκτελείται αυτή τη στιγμή στο σύστημα και με μια συγκεκριμένη εργασία που αυτή την στιγμή τρέχει στο σύστημα.

2. Διαχειριστής Πόρων (Resource Manager)

Παρέχει πρόσβαση σε πόρους, που δεν έχουν σχέση με τον κώδικα του λειτουργικού, όπως συμβολοσειρές, γραφικά, αρχεία διάταξης κ.α., αποθηκεύοντας τα εκτός της εφαρμογής, για γενίκευση της εφαρμογής από συσκευές με διαφορετικά μεγέθη οθόνης και διαφορετικά χαρακτηριστικά, και για ανεξάρτητη συντήρηση των στοιχείων της εφαρμογής.

3. Διαχειριστής Παραθύρων (Window Manager)

Η διεπαφή που χρησιμοποιούν οι εφαρμογές για να «μιλήσουν» με τον διαχειριστή παραθύρων.

4. Πάροχος Περιεχομένου (Content Provider)

Οι πάροχοι περιεχομένου αποθηκεύουν και ανακτούν δεδομένα έτσι ώστε να γίνονται προσβάσιμα σε όλες τις εφαρμογές. Για κάθε πάροχο περιεχομένου αντιστοιχεί ένας συγκεκριμένος τύπος δεδομένου όπως εικόνα, ήχος κ.α., ενώ κάθε πάροχος αποθηκεύεται στο android.provider πακέτο της Java.

5. Διαχειριστής Τηλεφώνου (Telephony Manager)

Παρέχει πληροφορίες για τις υπηρεσίες τηλεφωνίας στη συσκευή. Με κάθε νέα αλλαγή στην τηλεφωνική κατάσταση της συσκευής, οι εφαρμογές μπορούν να λαμβάνουν ειδοποιήσεις, καθορίζοντας τις απαιτούμενες ενέργειες για την υλοποίηση μιας εφαρμογής. Έτσι, περιορίζονται ανάλογα με τον τύπο κινητής που διαθέτει η συγκεκριμένη συσκευή.

6. Προβολή Συστήματος (View System)

Αυτή η κλάση αποτελεί το βασικό δομικό στοιχείο για την δημιουργία μιας διεπαφής χρήστη και είναι υπεύθυνη για την κατάρτιση και την διαχείριση συμβάντων. Η κλάση View είναι η βασική κλάση η οποία περιέχει τα εργαλεία διάδρασης (widgets) τα οποία χρησιμοποιούνται για την δημιουργία διαδραστικών στοιχείων διεπιφάνειας χρήστη (κουμπιά, πεδία κειμένου, πεδία επιλογής κ.τ.λ.), ενώ αντίστοιχα η υποκλάση ViewGroup περιέχει τα ειδή εμφάνισης (layouts).

7. Διαχειριστής Κοινοποιήσεων (Notification Manager)

Η κλάση αυτή μέσω κοινοποιήσεων ενημερώνει τον χρήστη για τα γεγονότα τα οποία συμβαίνουν στο σύστημα.

Μορφές Κοινοποιήσεων είναι: εικονίδια που εμφανίζονται στην γραμμή κατάστασης, μέσω LED λυχνιών, μέσω δόνησης ή μέσω αναπαραγωγής ενός συγκεκριμένου ήχου.

8. Διαχειριστής Πακέτων (Package Manager)

Κλάση η οποία ανακτά πληροφορίες οι οποίες συνδέονται με τα πακέτα εφαρμογών που υπάρχουν εγκατεστημένα στην κινητή συσκευή.

2.2.3.2 Βιβλιοθήκες

Το Android περιλαμβάνει βιβλιοθήκες της C και C++ που χρησιμοποιούνται από διάφορα συστατικά του λειτουργικού συστήματος. Αυτές οι βιβλιοθήκες γίνονται διαθέσιμες μέσω του περιβάλλοντος εφαρμογών. Οι βασικές βιβλιοθήκες είναι:

1. Βιβλιοθήκη της C – πρόκειται για τροποποιημένη βιβλιοθήκη σε σχέση με την στάνταρ βιβλιοθήκη της C (libc) η οποία είναι προσαρμοσμένη για να λειτουργεί σε ενσωματωμένα συστήματα τα οποία βασίζονται σε Linux.
2. Βιβλιοθήκες Μέσων (Media Libraries) – βιβλιοθήκες οι οποίες υποστηρίζουν την αναπαραγωγή και την καταγραφή πολλών γνωστών προτύπων εικόνας, βίντεο και ήχου.
3. Βιβλιοθήκη Διαχείρισης Επιφανειών (Surface Manager) – διαχειρίζεται την εμφάνιση στην οθόνη απεικονίσεων 2D και 3D για τις εφαρμογές.
4. Βιβλιοθήκη LibWebCore – βιβλιοθήκη η οποία παρέχει την δυνατότητα δημιουργίας ενός στοιχείου προβολής Web(WebView) ή επεκτείνει την εξορισμού εφαρμογή πλοήγησης του Android.
5. Βιβλιοθήκη SGL – παρέχει μηχανή για την αναπαράσταση γραφικών 2D
6. Βιβλιοθήκες 3D – η χρήση τους βασίζεται στο OpenGL ES 1.0 API, οι βιβλιοθήκες αυτές χρησιμοποιούν είτε επιταχυντή υλικού 3D (όπου είναι αυτός διαθέσιμος) είτε επιταχυντή λογισμικού 3D.
7. Βιβλιοθήκη FreeType – βιβλιοθήκη που κάνει διαθέσιμα αρχεία εικονοστοιχείων (bitmap) και διανύσματα απόδοσης γραμματοσειρών(fonts).
8. Βιβλιοθήκη SQLite – μέσω της βιβλιοθήκης αυτής γίνεται διαθέσιμο ένα σχεσιακό σχήμα βάσης δεδομένων σε όλες τις εφαρμογές που την χρησιμοποιούν. Το σχεσιακό σχήμα βάσης δεδομένων έχει όλες τις λειτουργίες μια βάσης δεδομένων αλλά είναι πιο ελαφριά στην λειτουργία της.

2.2.3.3 Το επίπεδο του Android Runtime

Οι εφαρμογές του περιβάλλοντος Android καθώς και πολλές από τις βιβλιοθήκες υψηλού επιπέδου (στο επίπεδο του Application Framework) είναι γραμμένες στην γλώσσα προγραμματισμού Java και γι' αυτόν τον λόγο η ύπαρξη ενός διερμηνέα είναι αναγκαία για την εκτέλεση αυτών των προγραμμάτων. Στο Android υλοποιήθηκε ένας διερμηνέας που είναι βελτιστοποιημένος για μικρά ενσωματωμένα συστήματα με περιορισμένους πόρους (απαιτεί ένα ελάχιστο 64MB μνήμης).

Το όνομα του διερμηνέα του Android είναι Dalvik και ονομάστηκε έτσι από τον προγραμματιστή που τον ανέπτυξε. Κάθε εφαρμογή στο Android εκτελείτε σε ένα δικό της στιγμιότυπο του Dalvik απομονώνοντας την από τις υπόλοιπες εφαρμογές διασφαλίζοντας έτσι την ευστάθεια και την ασφάλεια του λειτουργικού συστήματος.

Τα αρχεία με τα byte codes που παράγονται για εκτέλεση από τον Dalvik είναι αρκετά μικρότερα από τα αντίστοιχα Java byte codes που παράγονται για εκτέλεση στον JVM (Java Virtual Machine) μειώνοντας έτσι σημαντικά τον χώρο που απαιτεί κάθε εφαρμογή για την εγκατάστασή της.

Ένας άλλος μηχανισμός του Android για σωστή διαχείριση μνήμης είναι ο συλλέκτης σκουπιδιών (garbage collector). Κάθε στιγμιότυπο του διερμηνευτή έχει τον δικό του συλλέκτη και άρα η περιοχή μνήμης που έχει στην διάθεσή του καθαρίζεται ανεξάρτητα από τους άλλους διερμηνείς. Κύριος ρόλος του συλλέκτη είναι να ελευθερώνει περιοχές μνήμης που καταλαμβάνονται από δεδομένα που δεν θα χρειαστούν ξανά κατά την πορεία εκτέλεσης του της εφαρμογής.

Τέλος, ένα σημαντικό χαρακτηριστικό του Android Runtime είναι η ύπαρξη του JIT (Just in Time) μεταφραστή. Ο JIT έχει σαν σκοπό την μετάφραση από byte codes σε κώδικα μηχανής προκειμένου να αυξηθεί η ταχύτητα εκτέλεσης των αντίστοιχων τμημάτων της εφαρμογής. Ο μεταφραστής και ο διερμηνέας λειτουργούν αρμονικά σε ένα αυστηρά καθορισμένο περιβάλλον συνεργασίας.

2.2.3.4 Πυρήνας Linux

Ο πυρήνας λειτουργεί ως ένα επίπεδο αφαίρεσης μεταξύ του υλικού μια συσκευής και της στοίβας λογισμικού. Το λειτουργικό σύστημα Android στηρίζεται στην έκδοση 2.6 του Linux το οποίο είναι υπεύθυνο για τη διαχείριση των device drivers, τον έλεγχο πρόσβασης στους πόρους του συστήματος, τη διαχείριση μνήμης και τις λοιπές υπηρεσίες που παρέχει ένα λειτουργικό σύστημα. Η επιλογή της αυτή

είχε σαν αποτέλεσμα κάθε καινούρια έκδοση του πυρήνα που κυκλοφορούσε να πρέπει να προσαρμοστεί, αντίστοιχα με την έκδοση 2.6, πριν μπορέσουν οι χρήστες του Android να επωφεληθούν από τα νέα χαρακτηριστικά που τυχόν θα έχουν εισαχθεί. Με την έκδοση 3.3 του πυρήνα του Linux, έχει γίνει η πολυπόθητη από πολλούς συγχώνευση του κώδικα του Android

2.2.3.5. Συστατικά στοιχεία εφαρμογής

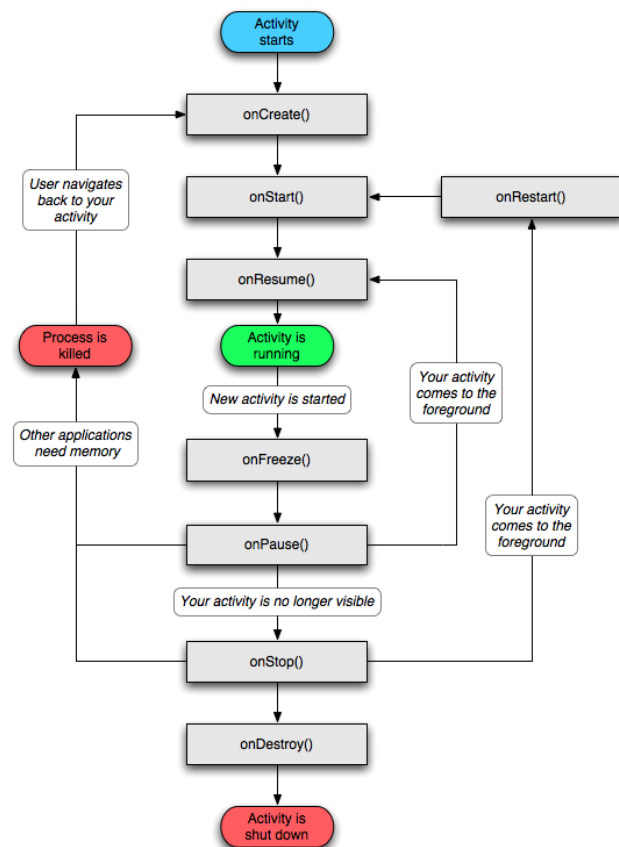
Με τον όρο αυτό εννοούμε, τα απαραίτητα δομικά στοιχεία μιας εφαρμογής Android, τα οποία έχουν πρόσβαση στο λειτουργικό σύστημα της εφαρμογής. Μπορούμε να τα διακρίνουμε σε τέσσερα βασικά στοιχεία : τις δραστηριότητες (activities), τις υπηρεσίες (services), τους παρόχους περιεχομένου (content providers), και τους καθολικούς παραλήπτες μηνυμάτων (broadcast receivers).

Δραστηριότητες

Αποτελούν το βασικότερο στοιχείο οποιασδήποτε εφαρμογής. Η κάθε δραστηριότητα (activity) αποτελεί μια διαφορετική οθόνη (παράθυρο) της εφαρμογής, με την οποία ο χρήστης αλληλεπιδρά. Σε αυτήν δηλαδή φορτώνεται το γραφικό περιβάλλον το οποίο βλέπει τελικά το χρήστης.

Η διεπαφή χρήστη (user interface) της εφαρμογής εμφανίζεται σε μια συσκευή μέσω μιας activity, συνήθως με μια activity που δημιουργήθηκε για κάθε μοναδική οθόνη. Αυτή την πρώτη οθόνη που βλέπει ο χρήστης την ονομάζουμε αρχική (main) activity. Εσωτερικά υπάρχει μια στοίβα των δραστηριοτήτων, κατά τη μετακίνηση από τη μία οθόνη στην άλλη. Συγκεκριμένα, η επόμενη activity που είναι να εμφανιστεί στέλνεται στην

Εικόνα 2.14: Κύκλος Ζωής Δραστηριότητας
<http://i.stack.imgur.com/88cpr.jpg>



κορυφή της στοίβας - με άλλα λόγια, η activity στην κορυφή της στοίβας είναι αυτή που είναι ορατή στην οθόνη. Οι activities βγαίνουν από τη στοίβα πατώντας το πλήκτρο επιστροφής, ενέργεια η οποία συνεχίζει την προηγούμενη activity. Αν πατήσουμε το πλήκτρο επιστροφής στην main activity τότε βγαίνουμε από την εφαρμογή μας.

Οι πιθανές καταστάσεις στις οποίες μπορεί να βρεθεί ένα activity είναι η Running, Paused και Stopped. Με τις μεθόδους αυτές μπορούμε να λάβουμε δράση όταν το Android αλλάζει κατάσταση στα activity της εφαρμογής μας. Όλες οι διαθέσιμες καταστάσεις ενός activity και οι πιθανές μεταβάσεις μεταξύ τους καθορίζουν τον κύκλο ζωής ενός activity.

Κάθε activity πρέπει να υλοποιεί την μέθοδο `onCreate()`, η οποία καλείται την στιγμή που ξεκινά η activity. Η μέθοδος αυτή καλείται μόνο όταν τρέξουμε μια νέα activity είτε από ένα κουμπί είτε από κάποια επιλογή, δηλαδή δεν καλείται αν επιστρέψουμε σε αυτή με το πλήκτρο επιστροφής. Αυτό συμβαίνει γιατί αν συμβεί το παραπάνω σενάριο θα δούμε ένα στιγμιότυπο της activity και αυτή δεν θα δημιουργηθεί ξανά. Επιπρόσθετα καλείται και η μεγέθους `onPause()`, η οποία ενδείκνυται πως ο χρήστης εγκαταλείπει την συγκεκριμένη οθόνη της εφαρμογής.

Επίσης, μέσα σε αυτήν την μέθοδο πρέπει να δημιουργηθεί και το user interface το οποίο δηλώνεται με την εντολή `setContentView()`, που περιέχει το XML αρχείο με τις διατάξεις των γραφικών οθόνης και των εργαλείων διάταξης.

Επιπρόσθετες μέθοδοι που παρουσιάζονται από το πιο πάνω σχεδιάγραμμα:

- **onStart ()**: Η μέθοδος αυτή καλείται ακριβώς πριν η διεπαφή εμφανιστεί στον χρήστη. Την κλήση αυτή ακολουθεί η κλήση της `onResume()` όταν η διεπαφή βρεθεί στο προσκήνιο ή η κλήση της `onStop()` όταν η διεπαφή βρεθεί στο παρασκήνιο.
- **onResume ()** Η μέθοδος αυτή καλείται ακριβώς πριν η διεπαφή γίνει διαθέσιμη στον χρήστη για αλληλεπίδραση.
- **onPause ()** Η μέθοδος αυτή καλείται όταν το σύστημα ετοιμάζεται να καλέσει την `onResume()` κάποιου άλλου activity. Εδώ πρέπει να γίνουν τυχών αποθηκεύσεις δεδομένων σε μόνιμα μέσα αποθήκευσης καθώς και να σταματήσουν διεργασίες που καταναλώνουν χρόνο στον επεξεργαστή

- **onStop ()** Η μέθοδος αυτή καλείται όταν το activity δεν είναι εμφανές στην οθόνη. Αυτό μπορεί να συμβαίνει γιατί μπορεί το λειτουργικό σύστημα να τερματίζει το activity αυτό ή γιατί κάποιο άλλο activity το καλύπτει.
- **onDestroy ()** Η μέθοδος αυτή καλείται όταν ένα activity καταστρέφεται. Η κλήση αυτή είναι η τελευταία που λαμβάνει το activity και μπορεί να σημαίνει ότι απλά τερματίζει, με κλήση της *finish()*.

Υπηρεσίες

Μια υπηρεσία (service) αποτελεί ένα στοιχείο της εφαρμογής, το οποίο τρέχει στο παρασκήνιο και μπορεί να εκτελέσει αρκετές και χρονοβόρες διαδικασίες, δίνοντας την δυνατότητα να εκτελούνται πλήθος εφαρμογών ταυτόχρονα, χωρίς να διακόπτει η μια την άλλη.

Βασικές λειτουργίες που σχετίζονται με τις υπηρεσίες είναι

- **startService()**: καλείται από κάποια activity για να ξεκινήσει η υπηρεσία και θα συνεχίσει να εκτελείται ακόμα και όταν η activity που την κάλεσε τερματιστεί.
- **bindService()**: ενώνει την υπηρεσία με κάποιο activity, έτσι ώστε να υπάρχει αλληλεπίδραση μεταξύ αυτών και αποστολή ή λήψη αποτελεσμάτων. Στην περίπτωση αυτή, αν τερματιστεί το στοιχείο που έχει ενωθεί με τη υπηρεσία, τότε τερματίζεται και αυτή.

Η εντολή **stopSelf()** τερματίζει τις συγκεκριμένες υπηρεσίες που επιθυμεί ο χρήστης, ενώ το λειτουργικό σύστημα μπορεί να δουλέψει αυτόνομα και να τις τερματίσει αντίστοιχα, απελευθερώνοντας μνήμη και υπολογιστική ισχύ.

Πάροχοι Περιεχομένου

Οι πάροχοι περιεχομένου (content providers) διαχειρίζονται αποθηκευτικούς χώρους για δεδομένα, οι οποίοι είναι προσπελάσιμοι από οποιαδήποτε εφαρμογή. Αποτελούν τον μοναδικό τρόπο για αποθήκευση δεδομένων, στα οποία θέλουμε πρόσβαση και από άλλες εφαρμογές. (π.χ SQL Lite for Android)

Broadcast Receivers

Οι broadcast receivers είναι διεπαφές που ενημερώνονται για ένα συγκεκριμένο γεγονός από το λειτουργικό σύστημα και τότε ενεργοποιούνται. Οι broadcast receivers δεν έχουν user interface, ωστόσο μπορούν να ενημερώσουν τον χρήστη μέσω των ειδοποιήσεων status bar.

2.2.3.6 Διεπαφή χρήστη(User Interface)

Η διεπαφή χρήστη (user interface) αποτελεί την τελική εικόνα που βλέπει ο χρήστης, το γραφικό περιβάλλον στο οποίο θα περιηγηθεί και ενεργοποιεί όλες τις λειτουργίες της εφαρμογής. Πολλές φορές είναι πιο δύσκολο για τον προγραμματιστή να κατασκευάσει ένα εύχρηστο και όμορφο περιβάλλον εργασίας παρά η ίδια η εφαρμογή. Καθίσταται σαφές λοιπόν, ότι απαιτεί μεγάλη προσπάθεια και προσοχή η δημιουργία ενός γραφικού περιβάλλοντος που θα προσελκύει τους χρήστες και θα τους ωθεί να χρησιμοποιούν μια συγκεκριμένη εφαρμογή έναντι μιας άλλης με τις ίδιες λειτουργίες.

Διατάξεις οθόνης (Layouts)

Ο πιο συνηθισμένος τρόπος για τον καθορισμό μιας διάταξης οθόνης η οποία θα εκφράζει μια ιεραρχία κόμβων που περιγράψαμε παραπάνω είναι με την δημιουργία ενός αρχείου XML (XML Layout). Η XML προσφέρει μια δομή που είναι πιο εύκολα κατανοητή από τον άνθρωπο όπως η HTML. Κάθε στοιχείο σε ένα αρχείο XML είναι ένα αντικείμενο View ή View Group. Το όνομα ενός XML στοιχείου είναι αντίστοιχο με την κλάση Java που αντιπροσωπεύει. Σε κάθε οθόνη της εφαρμογής πρωταρχικό στοιχείο του γραφικού περιβάλλοντος αποτελεί η διάταξη των γραφικών στοιχείων (layout). Το layout περιλαμβάνει όλα τα γραφικά στοιχεία της οθόνης τα οποία μπορεί να είναι διατεταγμένα σε επιμέρους layouts.

Υπάρχουν πέντε είδη διάταξης:

- **Γραμμική διάταξη (Linear Layout):** την διάταξη στοιχείων σε οριζόντια ή κατακόρυφη σειρά. Ένα Linear Layout λαμβάνει υπόψη του τα περιθώρια (margins) μεταξύ των στοιχείων του καθώς και την στοίχιση (gravity) τους.
- **Διάταξη πλαισίου (Frame Layout):** αποτελεί την πιο απλή διάταξη στοιχείων, καθώς δημιουργεί ένα κενό χώρο όπου μπορεί ο προγραμματιστής να βάλει ένα αντικείμενο, για παράδειγμα μια εικόνα, και αυτό το στοιχείο θα έχει τον ρόλο της ρίζας (root) όπου όλα τα υπόλοιπα θα διατάσσονται σύμφωνα με αυτό

- **Σχετική διάταξη (Relative Layout):** δίνει στον προγραμματιστή μια μεγαλύτερη ευχέρεια και ελευθερία με την έννοια ότι μπορούμε να εμφανίσουμε οποιοδήποτε αντικείμενο σε όποια θέση θέλουμε (βασιζόμενο στο ID του κάθε στοιχείου)
- **Διάταξη πίνακα (Table Layout):** , είναι η διάταξη πίνακα, μπορεί δηλαδή να διατάσσει τα παιδιά του (children) σε σειρές και στήλες.
- **Absolute Layout:** ορίζει την διάταξη των αντικείμενων με βάση την απόλυτη θέση τους σε σχέση με την οθόνη, η οποία όμως διαφέρει ανάλογα με την συσκευή που έχει ο χρήστης
- **List View:** ο χρήστης έχει στην οθόνη του μία λίστα από αντικείμενα και μπορεί να περιηγηθεί σε αυτά. Η πλατφόρμα Android παρέχει την κλάση List View η οποία είναι ικανή για την προβολή μιας λίστας στοιχείων κύλισης. Αυτά τα στοιχεία μπορεί να είναι οποιουδήποτε τύπου.

2.2.3.7 Εργαλεία διάδρασης (Basic widgets)

Κάθε διεπιφάνεια χρήστη χρησιμοποιεί κάποια βασικά εργαλεία διάδρασης όπως ετικέτες, πεδία κειμένου, κουμπιά κ.α. Έτσι και οι διεπιφάνειες χρήστη του Android δεν διαφοροποιούνται. Παρακάτω θα δούμε πως λειτουργούν τα βασικά εργαλεία διάδρασης στις Android εφαρμογές.

Ετικέτες (Labels)

Το απλούστερο εργαλείο διάδρασης είναι η ετικέτα. Στην πλατφόρμα Android το εργαλείο αυτό ονομάζεται TextView. Όπως και σε άλλες πλατφόρμες, στις περισσότερες διεπιφάνειες χρήστη οι ετικέτες είναι κομμάτια κειμένου τα οποία δεν μπορούν να τροποποιηθούν από τους χρήστες. Συνήθως οι ετικέτες χρησιμοποιούνται για να προσδιορίσουν άλλα εργαλεία διάδρασης. Στην πλατφόρμα Android πιο συχνά δημιουργούνται ετικέτες μέσω του XML layout που παρουσιάσαμε παραπάνω προσθέτοντας ένα στοιχείο TextView και καθορίζοντας την ιδιότητα text του ανάλογα με το κείμενο που πρέπει να εμφανίζεται στην οθόνη του χρήστη, μαζί με επιπρόσθετες ιδιότητες.

Κουμπιά (Buttons)

Το Button είναι ένα εργαλείο διάδρασης που αντιπροσωπεύει ένα κουμπί. Το κουμπί αυτό μπορεί να πιεστεί από τον χρήστη προκειμένου να εκτελεστεί μια ενέργεια. Κάθε

Button ορίζει το στυλ του χρησιμοποιώντας τις προεπιλεγμένες επιλογές του συστήματος οι οποίες συχνά είναι διαφορετικές από τη μία συσκευή στην άλλη.

Εικόνες (Images)

Το Android έχει δύο εργαλεία διάδρασης τα οποία βοηθάνε στην ενσωμάτωση εικόνων σε μια εφαρμογή. Το Image View και το Image Button. Κάθε εργαλείο από τα δύο που προαναφέραμε έχει μια ιδιότητα `res` (σε ένα XML layout) με την οποία καθορίζει ποια εικόνα χρησιμοποιεί. Συνήθως αυτή η εικόνα αναφέρεται σε μια εξωτερική πηγή εικόνων (εικόνες τύπου .png γίνονται δέκτες από την πλατφόρμα Android).

Πεδία (Fields)

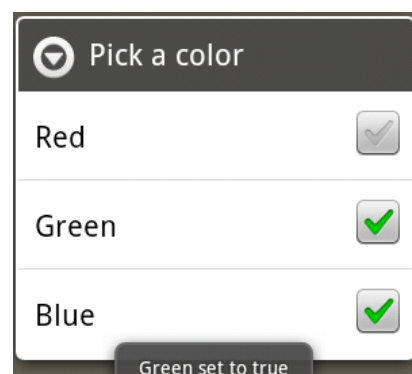
Μαζί με τα κουμπιά και τις ετικέτες, τα πεδία είναι τα επόμενα πιο βασικά στοιχεία μιας διεπιφάνειας χρήστη. Ο χρήστης καλείται να συμπληρώσει τα κενά πεδία, ώστε να αλληλεπιδράσει όσο πιο αποδοτικά με την εφαρμογή. Η στοίχιση των πεδίων, καθώς και οι περιορισμοί που πιθανόν να υπάρχουν (π.χ. αλφαριθμητικά – αριθμητικά δεδομένα μόνο), πρέπει να είναι όσο το δυνατόν πιο εύχρηστα προς τον χρήστη για παροχή υποκειμενικής ικανοποίησης.

Πλαίσιο ελέγχου (check box)

Το κλασικά πλαίσια ελέγχου έχουν δύο καταστάσεις: επιλεγμένο (checked) και μη επιλεγμένο (unchecked). Επιλέγοντας ένα πλαίσιο ελέγχου επιλέγονται μια από αυτές τις δύο καταστάσεις. Στο Android, υπάρχει το εργαλείο διάδρασης Checkbox το οποίο εξυπηρετεί αυτές τις ανάγκες.

Στην Java για το checkbox υπάρχουν οι μέθοδοι :

- ***isChecked()***: Καθορίζει αν το πλαίσιο ελέγχου έχει επιλεγθεί
- ***setChecked()***: Αναγκάζει το πλαίσιο ελέγχου να είναι σε δύο από τις πιθανές καταστάσεις (checked/unchecked)
- ***toggle()***: εναλλάσσετε το πλαίσιο ελέγχου ανάμεσα στις πιθανές καταστάσεις αν ο χρήστης το επιλέξει



Εικόνα 2.15: Πλαίσιο Ελέγχου
<http://malsandroid.blogspot.com/2010/04/checkbox-dialog.html>

Κουμπιά επιλογής (radio buttons)

Όπως και σε άλλα γραφικά περιβάλλοντα έτσι και στο Android τα κουμπιά επιλογής είναι δύο καταστάσεων σαν τα πλαίσια ελέγχου αλλά μπορούν να ομαδοποιηθούν έτσι ώστε μόνο ένα κουμπί επιλογής από μια ομάδα να είναι επιλεγμένο ανά πάσα

στιγμή. Η κλάση που αναπαριστά τα κουμπιά επιλογής στο Android ονομάζεται Radio Button.



Εικόνα 2.16: Κουμπιά Επιλογής
https://cdn.tutsplus.com/mobile/uploads/legacy/Android-SDK_Radio-Buttons/Fig2.png

Πρόσβαση σε αυτό μέσω της γλώσσα Java από τις παρακάτω μεθόδους:

- **check():** ελέγχει ένα συγκεκριμένο κουμπί επιλογής μέσω του ID του
- **clearCheck():** καθαρίζει τα κουμπιά επιλογής, έτσι ώστε κανένα στην ομάδα να μην είναι επιλεγμένο

Επιλογή ημέρας και ώρας (Date Picker & Time Picker)

Τα εργαλεία Date Picker & Time Picker χρησιμοποιούνται από τους σχεδιαστές, ώστε να ελαχιστοποιούνται τα σφάλματα κατά την εισαγωγή δεδομένων από τον χρήστη. (π.χ. κατά την εισαγωγή ημερομηνίας σε ένα πεδίο (Field), ισχύουν πάμπολλοι περιορισμοί όσον αφορά το format της ημερομηνίας). Άρα μειώνονται οι πιθανότητες στο να κάνει ο χρήστης λάθη, διευκολύνοντας την αλληλεπίδραση του με την εφαρμογή. Το Date Picker και το Time Picker είναι τα εργαλεία διάδρασης και το DatePickerDialog και TimePickerDialog είναι τα αντίστοιχα πλαίσια

διαλόγου με τα οποία ο χρήστης αλληλεπιδρά για να καθορίσει την ημερομηνία και την ώρα.



Εικόνα 2.17: Επιλογή Ημέρας Ώρας
<http://cfile29.uf.tistory.com/image/131B064F4E51DB9A0B05B6>

2.3. Πλατφόρμα FI-STAR (Future Internet Social Technological Alignment in HealthCare)

2.3.1 Εισαγωγή

Η πλατφόρμα FI-STAR (Future Internet – Social Technological Alignment in Healthcare) που αρχικά ξεκίνησε με τη συνεργασία 7 χωρών της Ευρωπαϊκής Ένωσης (Νορβηγία, Ηνωμένο Βασίλειο, Ισπανία, Ιταλία, Γερμανία, Πολωνία), τελικά συγκλίνει στο να γίνει αυτόνομη και να συνεχίσει ως ένα βιώσιμο οικοσύστημα για όλες τις ομάδες χρηστών στην παγκόσμια υγειονομική περίθαλψη βελτιώνοντας έτσι την ποιότητα ζωής των ασθενών. Προκειμένου να ανταποκριθεί στις απαιτήσεις που προκύπτουν από το τομέα της υγείας, η πλατφόρμα FI-STAR κτίζεται πάνω στο μοντέλο του cloud computing. Σκοπός της πλατφόρμας αυτής είναι να φέρει το λογισμικό στα δεδομένα και όχι τα δεδομένα στο λογισμικό.

Η πλατφόρμα FI-STAR στηρίζεται στις προδιαγραφές των πλατφόρμων FI-PPP και FI-WARE οι οποίες υποστηρίζουν αποτελεσματικές και χρήσιμες υπηρεσίες που αφορούν και τον τομέα της υγειονομικής περίθαλψης παρέχοντας κάποια generic enablers στα οποία στηρίζονται οι developers της πλατφόρμας FI-STAR για τη δημιουργία specific enablers ούτως ώστε να καλύψουν τις ανάγκες που δημιουργούνται για μια καλύτερη υγειονομική περίθαλψη.

Τα specific enablers της πλατφόρμας FI-STAR παρέχουν πολλές δυνατότητες όπως η διαχείριση ηλεκτρονικών ιατρικών εγγραφών (EHR specific enabler), η ενσωμάτωση και αποθήκευση ηλεκτρονικών ιατρικών αρχείων (storage se), η επικοινωνία μεταξύ διαφορετικών πόλεων, χωρών μέσω του NCP (National Contact Point) του eP-SOS specific enabler για τη διαχείριση των πληροφοριών των ασθενών, παρέχει ασφάλεια των πληροφοριών που διακινούνται και πιστοποίηση της ταυτότητας των χρηστών (security και management/administration enablers) καθώς επίσης γίνεται και διαχείριση των events που τυγχάνουν (timinig service και event manager enablers). Παρέχοντας τις πιο πάνω δυνατότητες και αξιοποιώντας πλήρως όλα τα πλεονεκτήματα του cloud computing και τη διασφάλιση της προστασίας των προσωπικών δεδομένων των χρηστών, η πλατφόρμα fi-star αποτελεί μια μοναδική ευκαιρία εφαρμογής του future internet.

Η πλατφόρμα FI-WARE στην οποία στηρίζεται η πλατφόρμα FI-STAR παρέχει open standard APIs (Application Programming Interfaces) τα οποία βοηθούν στην ευκολότερη σύνδεση με Internet of Things (IoT) devices, επεξεργάζεται δεδομένα και

media σε πραγματικό χρόνο και με μεγάλη κλίμακα, εκτελεί ανάλυση μεγάλων δεδομένων (Big Data) ή ενσωματώνει χαρακτηριστικά για αλληλεπίδραση με το χρήστη. Η FI-STAR πλατφόρμα στηρίζεται στη FI-WARE πλατφόρμα γιατί οι ειδικότητες της τελευταίας συνοψίζονται ως εξής:

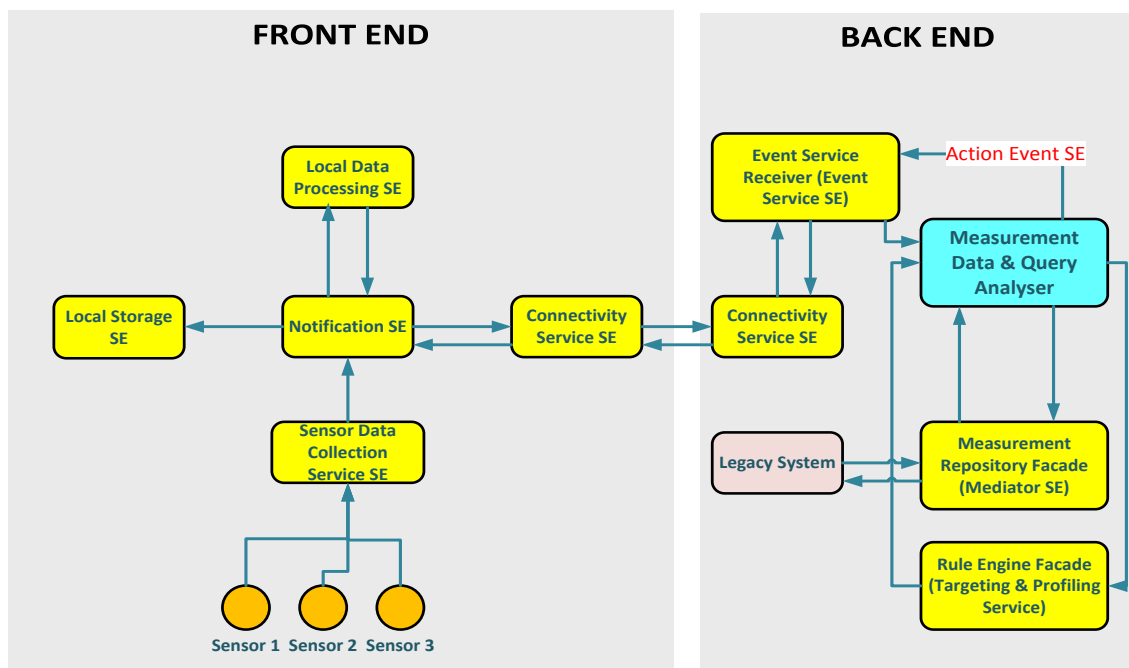
- Cloud
- open source
- Big Data
- Internet of Things
- Smart Cities
- Opens APIs
- Open Data

καλύπτοντας πλήρως τους στόχους της πλατφόρμας FI-STAR

Σημείωση: Όλα τα παρακάτω σχεδιαγράμματα πάρθηκαν από το site:
Fi-Star Catalogue, 2015. [<http://catalogue.fi-star.eu/> Fi-star eHealthCare]

2.3.2 Αρχιτεκτονική πλατφόρμας FI-STAR

Η αρχιτεκτονική της πλατφόρμας FI-STAR για υλοποίηση στην πλατφόρμα Google Android χωρίζεται σε δύο μέρη: Front End και Back End αρχιτεκτονικές λογισμικού (Πιο κάτω σχεδιάγραμμα). Το κομμάτι που θα αφορά το μέρος της



Σχεδιάγραμμα 2.1: Επικοινωνία Front και Back End

διπλωματικής μου εργασίας είναι η Back End αρχιτεκτονική, όπου θα υλοποιηθεί η εφαρμογή αλληλεπίδρασης του ιατρού με το σύστημα (ο ιατρός παίρνει τα δεδομένα από την βάση δεδομένων, και έπειτα παρουσιάζονται από την εφαρμογή σε μορφή ψηφιακού καρδιογραφήματος).

Front End Αρχιτεκτονική:

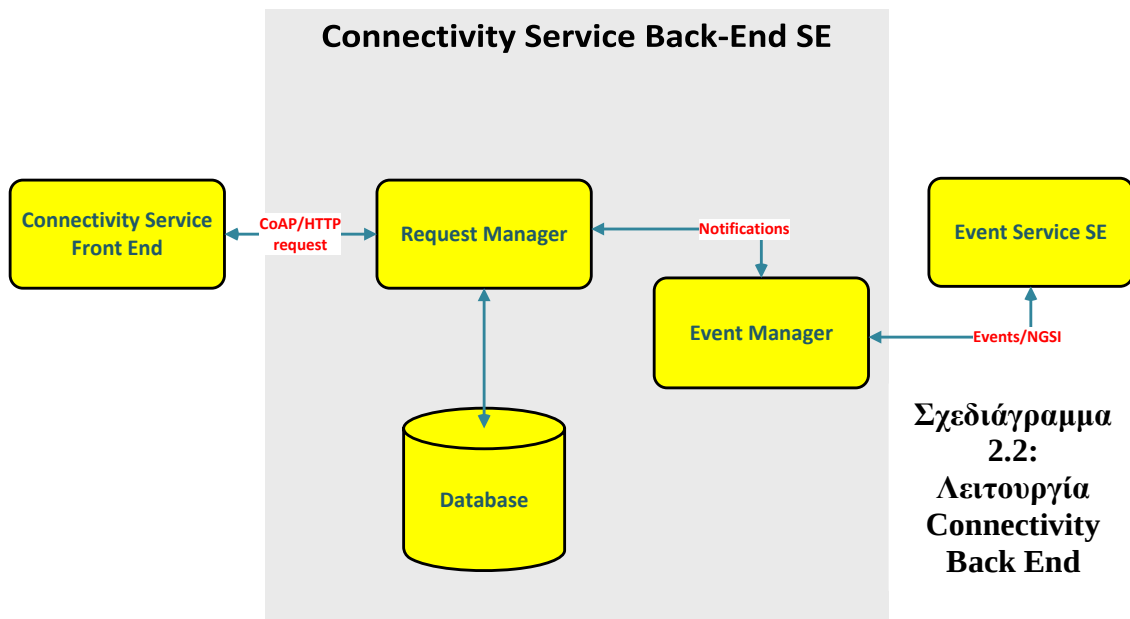
- **Local Data Processing SE:** στα δεδομένα, τα οποία συλλέγονται από τους αισθητήρες, θα πρέπει να υπάρξει κάποια προ-επεξεργασία, ώστε να μετατραπούν σε συμβατά δεδομένα αποθήκευσης στην βάση δεδομένων (δημιουργία του καταλληλότερου ψηφιακού ηλεκτροκαρδιογραφήματος).
- **Local Storage SE:** η βάση δεδομένων, όπου θα αποθηκεύονται τα δεδομένα συλλογής από τους αισθητήρες. Τα δεδομένα θα αποθηκεύονται σε 3 στήλες, ως ακολούθως:
 1. Η πρώτη στήλη θα περιέχει τις τιμές του καρδιογραφήματος για τις απαγωγές LA-LL
 2. Η δεύτερη στήλη θα περιέχει τις τιμές του καρδιογραφήματος για τις απαγωγές RA-LL
 3. Η τρίτη στήλη θα περιέχει τις χρονικές τιμές του καρδιογραφήματος (timestamp).
- **Sensor Data Collection Service SE:** οι αισθητήρες καταγραφής του καρδιογραφήματος
- **Notification SE:** Το μέρος της εφαρμογής, που ενημερώνει τον χρήστη σχετικά με τις μετρήσεις που πάρθηκαν από τον καρδιογράφο, καθώς και για υπενθυμίσεις σχετικά με τα δεδομένα αποθήκευσης στην βάση δεδομένων.
- **Connectivity Service SE:** Η συνδεσιμότητα στην Front End αρχιτεκτονική αλληλεπιδρά αμφίδρομα με την συνδεσιμότητα στην Back End αρχιτεκτονική (πιο κάτω λεπτομερείς επεξήγηση σχετικά με την Back End συνδεσιμότητα).

Back End Αρχιτεκτονική:

Connectivity Service SE:

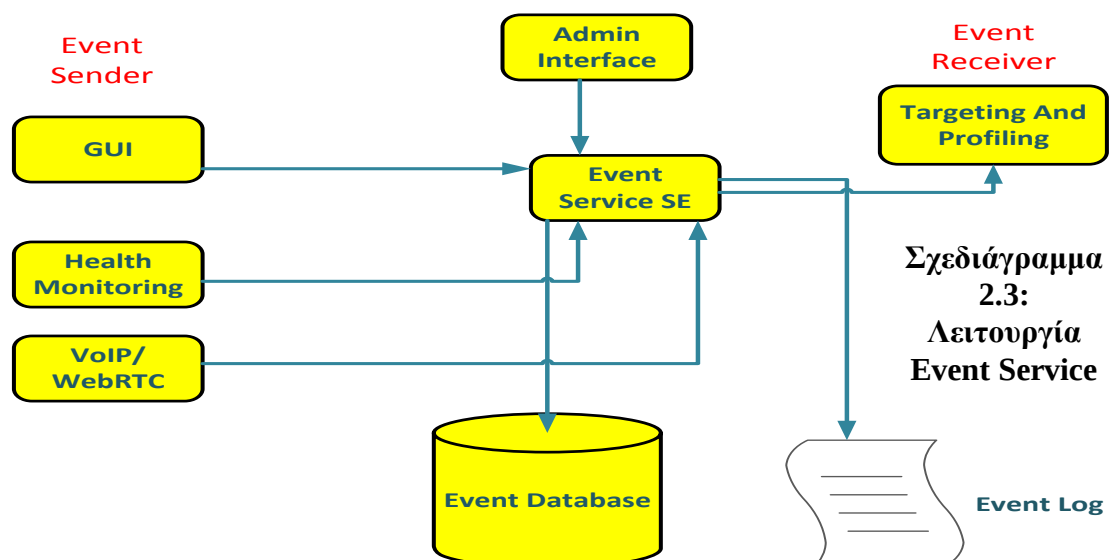
Η συνδεσιμότητα στην Back End αρχιτεκτονική χρησιμοποιείται από την πλατφόρμα Fi-Star στο να στέλλονται τα δεδομένα στην Front End SE ή να λαμβάνονται τα δεδομένα από την Front End SE. Επιτρέπει, έτσι, την δημιουργία

πληροφοριών της συσκευής στο Back End και έπειτα, την προώθηση των πληροφοριών



στο Event Service, χρησιμοποιώντας Events. Η μετάδοση των δεδομένων υποστηρίζεται είτε από HTTP είτε από CoAP.

Η υπηρεσία συνδεσιμότητας του Back End χωρίζεται σε 2 μέρη, όπου κάθε μέρος αλληλεπιδρά με ένα SE. Το πρώτο μέρος είναι υπεύθυνο για να διαχειρίζεται τις πολλαπλές αιτήσεις που έρχονται από την υπηρεσία συνδεσιμότητας του Front End. Επιπρόσθετα αποθηκεύει τις πληροφορίες της συσκευής, εάν επιθυμεί ο διαχειριστής. Το δεύτερο μέρος ειδοποιεί, σε περίπτωση ενημερώσεων στις τιμές δεδομένων, και δημιουργεί Event, τα οποία στέλλονται στο Event Service SE. **Event Service Receiver (Event Service SE):** παρέχει το γραφικό περιβάλλον χρήστη (του διαχειριστή-ιατρού),



όπου ο υπεύθυνος ιατρός μόνο μπορεί να πιστοποιηθεί για είσοδο στο σύστημα (ασφάλεια στο σύστημα).

Το Event Service SE είναι μια διαδικτυακή υπηρεσία, που παρέχει την διεπιφάνεια πρόσβασης. Προωθεί τα events στην κατακευμαμένη βάση δεδομένων-events, στα log files και σε όλα τα εγγεγραμμένα events. Ακολουθώς, δίνεται η δυνατότητα ταυτοποίησης των events από την βάση δεδομένων, ώστε ο διαχειριστής να πραγματοποιήσει εγγραφή/διαμόρφωση των events.

- **Measurement Repository Facade (Mediator SE):**

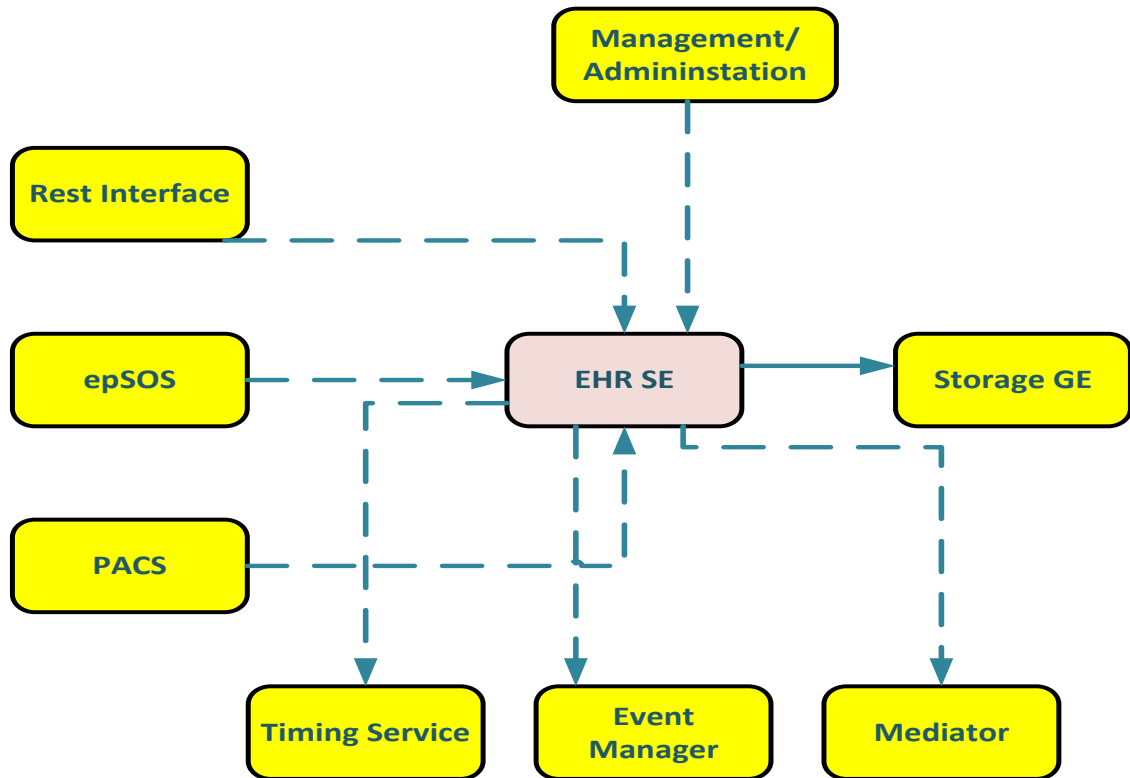
παρέχει την δυνατότητα σε άλλα Fi-Star SE να αλληλεπιδράσουν με το προϋπάρχον σύστημα στον τομέα της υγείας (legacy system). Στην πρώτη φάση, οι καλούντες διαμορφώσεις γίνονται με κλήσεις SOAP και παρέχονται κάποιες ταμπλέτες και οδηγίες για τον τρόπο διαμόρφωσης των εργασιών.

- **EHR SE Legacy System:**

παρέχει τα απαραίτητα components διασύνδεσης και επικοινωνίας. Η αρχιτεκτονική αυτή είναι βασισμένη στα ακόλουθα components:

1. **Management/Administration component:** είναι υπεύθυνο για την διαμόρφωση και παρακολούθηση του EHR SE.
2. **Rest Interface:** υπεύθυνο για την αποκατάσταση επικοινωνίας με τρίτους
3. **Storage component:** υπεύθυνο για επικοινωνία με την βάση για αποθήκευση και ανάκτηση δεδομένων.
4. **Timing Service component:** υπεύθυνο για επικοινωνία με την timing service SE, έχοντας κοινό timestamp.
5. **Security component:** υπεύθυνο για εξουσιοδότηση και πιστοποίηση της αίτησης πρόσβασης στο EHR SE.
6. **Event Management:** υπεύθυνο για επικοινωνία με το Event Management SE
7. **EpSOS SE:** υπεύθυνο για παροχή πληροφοριών του ασθενή
8. **PACS SE:** υπεύθυνο για παροχή της διαπροσωπείας εικόνας

9. **EHR SE:** υπεύθυνο για την θέσπιση των κατάλληλων components διασύνδεσης και επικοινωνίας



Σχεδιάγραμμα 2.4: Λειτουργία EHR

- **Rule Engine Facade (Targeting & Profiling Service):** παρέχει τις διαπροσωπείες, από τις οποίες θα ανακτώνται τα προφίλ των ασθενών, από τα δεδομένα αποθήκευσης στην βάση
- **Measurement Data & Query Analyser:** η εφαρμογή που αναφέρεται στο ιατρικό κομμάτι του συστήματος, όπου ο ιατρός αφού ζητήσει εξουσιοδότηση από το σύστημα για σύνδεση, μπορεί να ανακτήσει τις πληροφορίες των ασθενών από την βάση δεδομένων και να τις παρουσιάσει στην οθόνη της συσκευής

Κεφάλαιο 3

Ανάλυση Συστήματος

3.1 Αποτελέσματα Συζητήσεων	45
3.2 Απαιτήσεις Ασθενών-Ιατρού	46
3.2.1 Απαιτήσεις Ιατρού	46
3.2.2 Απαιτήσεις Ασθενών	47
3.3 Συσκευή Arduino - Shimmer και Αισθητήρων	47
3.3.1 Εισαγωγή πλατφόρμα Arduino	47
3.3.2 Λειτουργία Arduino	50
3.3.3 Ονομασία και μονάδες μετρήσεις των αισθητήρων	50
3.3.4 Λειτουργία Shimmer	59
3.4 Ανάλυση δραστηριοτήτων του συστήματος	60

3.1 Αποτελέσματα Συζητήσεων

Μετά από συζήτηση με τον κ. Ανδρέα Παναγίδη, προχώρησα στην αναζήτηση των απαιτήσεων που θα χρειάζονται στην καταλληλότερη παρακολούθηση των βιομετρικών δεδομένων των ασθενών.

Μετά από πολύωρη συζήτηση, καταλήξαμε στα εξής συμπεράσματα:

- Ο ασθενής θα έχει την δυνατότητα να καταγράφει σε πραγματικό χρόνο τα εξής ιατρικά δεδομένα:

Ιατρικά Δεδομένα Μετρήσεων
Οξυγόνωση του Αίματος
Καρδιακούς Παλμούς
Θερμοκρασία Σώματος
Ροή Αέρα (Αναπνευστική Παροχή)
Συστολική Πίεση Αίματος
Διαστολική Πίεση Αίματος
Ηλεκτροκαρδιογράφημα
Ηλεκτρομυογράφημα
Διαγωγιμότητα Δέρματος

- Η συσκευή που θα χρησιμοποιεί ο ασθενής για την καταγραφή των δεδομένων θα είναι ένα κινητό τηλέφωνο λογισμικού Google Android
- Ο ιατρός θα παρακολουθούσε τα συμβάντα του ασθενή μέσω του ηλεκτρονικού υπολογιστή (Web App)

3.2 Απαιτήσεις Ασθενών-Ιατρού

3.2.1 Απαιτήσεις Ιατρού

Με βάση τις συζητήσεις που είχαμε τόσο με το προσωπικό του κ. Παττίχη όσο και με τον ιατρό, φτάσαμε στο συμπέρασμα πως οι ασθενείς θα πρέπει να ελέγχονται όχι μόνο στο θέμα του ηλεκτροκαρδιογραφήματος, αλλά και σε επιπρόσθετες βιομετρήσεις (Όπως αναφέρονται και πιο πάνω):

- Η οξυγόνωση του αίματος, σε συνδυασμό με την αναπνευστική παροχή θα ελέγχει εάν ο ασθενής υποφέρει από κάποιο αναπνευστικό πρόβλημα ή υπάρχει κίνδυνος άσθματος και επιδείνωσης της δυσλειτουργίας των αεραγωγών του αναπνευστικού συστήματος.
- Η θερμοκρασία του σώματος καθορίζει εάν υπάρχει κάποια περίπτωση πυρετού ή υποθερμίας στον ασθενή.
- Συστολική και Διαστολική πίεση πρέπει να πραγματοποιούνται συχνά για τον σωστό έλεγχο του καρδιακού κύκλου και παροχής.
- Ηλεκτροκαρδιογράφημα θα πραγματοποιείται όταν ο ασθενής νιώσει κάποια καρδιακή ενόχληση, και έτσι ο ιατρός να μπορεί να διαπιστώσει εάν υπάρχει κάποιο πρόβλημα ή όχι.
- Ηλεκτρομυογράφημα θα πραγματοποιείται για εξακρίβωση εάν τα δυναμικά ενεργείας που παράγονται από την μυϊκή δραστηριότητα είναι στα κανονικά επίπεδα και ο ασθενής δεν υποφέρει από μυϊκή ατροφία ή οποιαδήποτε άλλη μυϊκή πάθηση.
- Η διαγωγιμότητα του δέρματος δείχνει εάν ο ασθενής βρίσκεται σε κατάσταση σοκ ή είναι αγχώδης, πράγμα που θα ειδοποιά τον ιατρό να λάβει μέτρα διαχείρισης της ψυχολογίας του ασθενή.

3.2.2 Απαιτήσεις ασθενών

Οι ασθενείς θα είναι άτομα όλων των ηλικιών, με ή χωρίς κάποια τεχνολογική γνώση. Γι' αυτόν τον λόγο, κρίνεται σωστό η κατάλληλη σχεδίαση και ευχρηστία της εφαρμογής, ώστε οποιοσδήποτε και με εύκολο τρόπο να μπορεί να διαχειριστεί και να κατανόη σωστά τα δεδομένα που θα εμφανίζονται στην εφαρμογή. Η χρήση οδηγιών και κατάλληλων μηνυμάτων θα ενισχύσει αυτήν την προσπάθεια.

Επιπρόσθετα, ο ασθενής θα μπορεί να παρακολουθούσε το ιστορικό των βιομετρήσεων του, παρακολουθώντας την πορεία της μέτρησης ανά μέρα / εβδομάδα ή μήνα.

3.3 Συσκευή Arduino / Shimmer και Αισθητήρων

Σημείωση: Όλες οι παρακάτω φωτογραφίες των αισθητήρων και συσκευής Arduino
πάρθηκαν από το site

Cooking Hacks, July 2014.

[/http://www.cooking-hacks.com/documentation/tutorials/ehealth-biometric-sensor-platform-arduino-raspberry-pi-medical](http://www.cooking-hacks.com/documentation/tutorials/ehealth-biometric-sensor-platform-arduino-raspberry-pi-medical)

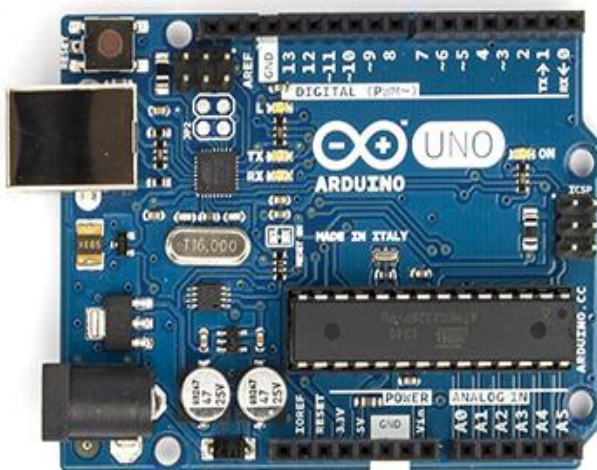
3.3.1 Εισαγωγή: Πλατφόρμα Arduino

Το Arduino είναι μια απλή μητρική πλακέτα ανοικτού κώδικα, με ενσωματωμένο μικροελεγκτή και εισόδους/εξόδους, και η οποία μπορεί να προγραμματιστεί με τη γλώσσα Wiring (σύνταξη και βιβλιοθήκες), παρόμοια με την C++ με απλοποιήσεις και αλλαγές. Το Arduino μπορεί να χρησιμοποιηθεί για την ανάπτυξη ανεξάρτητων διαδραστικών αντικειμένων, κάνοντας την συσκευή να ξεχωρίζει λόγω του ότι είναι πιο φθηνή από άλλα πρωτότυπα συστήματα. Η έκδοση Arduino Uno είναι η πλατφόρμα, στην οποία υλοποίησα την εφαρμογή Android. Πιο κάτω παρατίθενται τα τεχνητά χαρακτηριστικά της πλατφόρμας αυτής:

Τεχνητά Χαρακτηριστικά Arduino Uno	
Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)

Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Τεχνητά Χαρακτηριστικά από site
<http://www.arduino.cc/en/Main/ArduinoBoardUno>



Η φόρτιση καθώς και ο προγραμματισμός της συσκευής πραγματοποιείται από ένα USB καλώδιο, το οποίο συνδέεται στον υπολογιστή. Για σκοπούς επικοινωνίας, καθορίσαμε την χρήση του Wi-Fi Shield (θα εξηγήσω παρακάτω την χρήση του). Τα Arduino και τα Arduino συμβατά boards χρησιμοποιούν

την τεχνολογία των shields, τυπωμένων boards επεκτάσεων κυκλωμάτων που συνδέονται στα κανονικά παρεχόμενα Arduino pin-headers. Για τον προγραμματισμό του Arduino υπάρχει το ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) του Arduino, γραμμένο σε Java, που λειτουργεί σε πολλές πλατφόρμες, και προέρχεται από το IDE για τη γλώσσα προγραμματισμού Processing και το σχέδιο Wiring. Έχει σχεδιαστεί για να εισαγάγει τον προγραμματισμό στους καλλιτέχνες και τους νέους που δεν είναι εξοικειωμένοι με την ανάπτυξη λογισμικού. Περιλαμβάνει ένα πρόγραμμα επεξεργασίας κώδικα με χαρακτηριστικά όπως είναι η επισήμανση σύνταξης και ο συνδυασμός αγκύλων και είναι επίσης σε θέση να μεταγλωττίζει και να φορτώνει

προγράμματα στην πλακέτα με ένα μόνο κλικ. Ένα πρόγραμμα ή κώδικας που γράφτηκε για Arduino ονομάζεται σκίτσο (sketch).

Τα Arduino προγράμματα είναι γραμμένα σε C ή C++. Το Arduino IDE έρχεται με μια βιβλιοθήκη λογισμικού που ονομάζεται "Wiring" από το πρωτότυπο σχέδιο Wiring γεγονός που καθιστά πολλές κοινές λειτουργίες εισόδου/εξόδου πολύ πιο εύκολες.

Οι χρήστες πρέπει μόνο να ορίσουν δύο λειτουργίες για να κάνουν ένα πρόγραμμα κυκλικής εκτέλεσης:

- **setup():** μία συνάρτηση που τρέχει μία φορά στην αρχή του προγράμματος η οποία αρχικοποιεί τις ρυθμίσεις
- **loop():** μία συνάρτηση η οποία καλείται συνέχεια μέχρι η πλακέτα να απενεργοποιηθεί

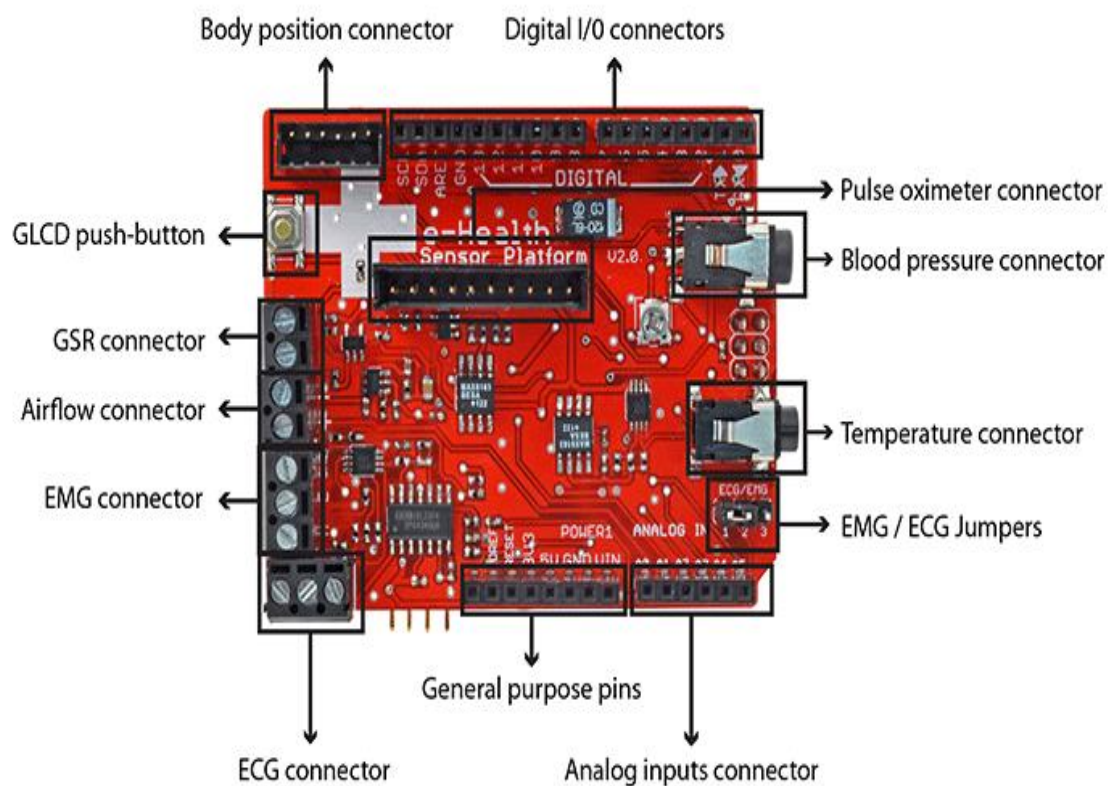
Ένα τυπικό πρώτο πρόγραμμα για έναν μικροελεγκτή αναβοσβήνει απλά ένα LED. Απόσπασμα κώδικα γραμμένο σε γλωσσά Arduino φαίνεται πιο κάτω:

```
/*  
  
int pushButton = 2;  
// the setup routine runs once when you press reset:  
void setup() {  
  // initialize serial communication at 9600 bits per second:  
  Serial.begin(9600);  
  // make the pushbutton's pin an input:  
  pinMode(pushButton, INPUT);  
}  
// the loop routine runs over and over again forever:  
void loop() {  
  // read the input pin:  
  int buttonState = digitalRead(pushButton);  
  // print out the state of the button:  
  Serial.println(buttonState);  
  delay(1);    // delay in between reads for stability
```

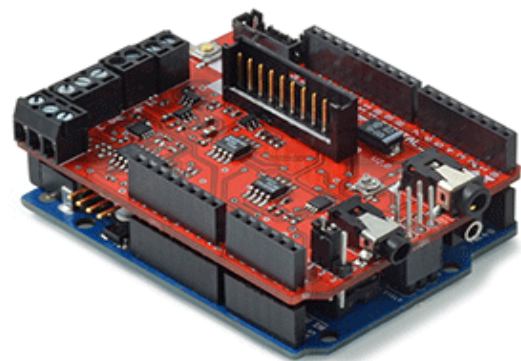
Κώδικας Arduino από site
<https://codebender.cc/sketch:57119>

3.3.2 Λειτουργιά Arduino (μαζί με την πλατφόρμα eHealth)

Η πλατφόρμα e-Health Sensor Shield V2.0 επιτρέπει στο Arduino να πραγματοποιεί και να ενσωματώνει βιομετρικές σε ιατρικές εφαρμογές με την χρήση 10 και πλέον διαφορετικών αισθητήρων (sensors). Η πληροφορία των βιομετρήσεων γίνεται σε πραγματικό χρόνο, όπου ο ασθενής άμεσα αντιλαμβάνεται εάν κάτι δεν πάει καλά με την υγεία του. Η βιομετρική πληροφορία συλλέγεται ασύρματα με την χρήση ενός Wi-Fi Shield και τα δεδομένα αναλύονται και παρουσιάζονται από την αντίστοιχη εφαρμογή χρήσης (στην περίπτωση μας μιλάμε για την πλατφόρμα Android). Η πλακέτα καθώς και οι θέσεις των αισθητήρων παρουσιάζονται πιο κάτω:



Η πλακέτα e-Health τροφοδοτείται από την θύρα USB ενός PC, ενώ για τον σωστό προγραμματισμό της πλακέτας, ο προγραμματιστής πρέπει να κατεβάσει την βιβλιοθήκη της e-Health, η οποία είναι γραμμένη σε C++ . Η βιβλιοθήκη αφήνει εύκολα να καταμετρηθούν όλες οι πληροφορίες από τους αισθητήρες και είναι ανοικτού πηγαίου



κώδικα. Η πλακέτα e-Health και η πλατφόρμα Arduino συνδέονται όπως παρουσιάζεται στην δεξιά εικόνα:

3.3.3 Ονομασία και μονάδες μετρήσεις των αισθητήρων Για τους σκοπούς της διπλωματικής θα χρησιμοποιήσω τις πιο κάτω βιομετρήσεις με τον αντίστοιχο αισθητήρα που αντιπροσωπεύουν. Η ανάλυση των αισθητήρων πραγματοποιείται πιο κάτω:

Βιομετρήσεις Αισθητήρων
Οξυγόνωση του Αίματος
Καρδιακούς Παλμούς
Θερμοκρασία Σώματος
Ροή Αέρα (Αναπνευστική Παροχή)
Συστολική Πίεση Αίματος
Διαστολική Πίεση Αίματος
Ηλεκτροκαρδιογράφημα
Ηλεκτρομυογράφημα
Διαγωγιμότητα Δέρματος

1) Αισθητήρας Οξυγόνωσης του Αίματος (SpO₂), μαζί με μετρητή Καρδιακών Παλμών (Παλμικό Οξύμετρο)



Ο αισθητήρας παλμικής οξυμετρίας, μαζί με τον αισθητήρα οξυγόνωσης του αίματος είναι μια μη-επεμβατική μέθοδος για την ένδειξη του κορεσμού του οξυγόνου στην λειτουργία της αιμοσφαιρίνης. Ο κορεσμός οξυγόνου ορίζεται ως η μέτρηση της ποσότητας του οξυγόνου που διαλύεται στο αίμα, με βάση την ανίχνευση της αιμοσφαιρίνης και των δεοξυαιμοσφαιρίνης. Ο τρόπος εντοπισμού αυτής της μέτρησης πραγματοποιείται από δύο διαφορετικά μήκη κύματος φωτός που χρησιμοποιούνται για τη μέτρηση της πραγματικής διαφοράς στα φάσματα απορρόφησης HbO₂ και Hb. Η κυκλοφορία του αίματος επηρεάζεται από την συγκέντρωση του HbO₂ και Hb. Το ένα μήκος κύματος εκπέμπεται στα 660 nm (κόκκινο φως φάσματα) ενώ το άλλο μήκος κύματος στα 940 nm (υπέρυθρο φάσμα φωτός). Διαφορετικά μήκη κύματος απορροφώνται από διαφορετικό τύπο αιμοσφαιρίνης (Οξυγονωμένη και μη - Οξυγονωμένη Αιμοσφαιρίνη).

Η μη-οξυγονωμένη αιμοσφαιρίνη (Hb) έχει υψηλότερη απορρόφηση στα 660 nm και η οξυγονωμένη αιμοσφαιρίνη (HbO₂) έχει υψηλότερη απορρόφηση στα 940 nm. Στη συνέχεια, μια φωτογραφία-ανιχνευτής αντιλαμβάνεται τη μη-απορρόφηση του φωτός από τα LEDs, υπολογίζοντας έτσι τον κορεσμό αρτηριακού αίματος σε οξυγόνο. Ένας αισθητήρας παλμικής οξυμετρίας είναι χρήσιμος σε περιβάλλον όπου η οξυγόνωση του ασθενούς είναι ασταθής. Ο αισθητήρας θα πρέπει να συνδέεται με το Arduino και τροφοδοτείται άμεσα μέσω της πλατφόρμα.

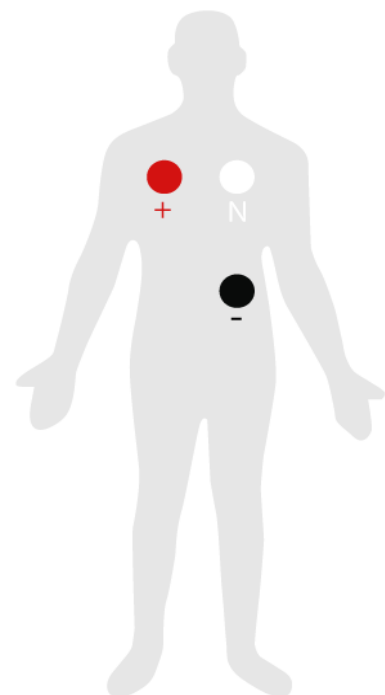
Κατώφλια (Thresholds) Μονάδα Μέτρησης = %	
88 – 94	Υποξαιμικό Πρόβλημα Κίνησης
95 – 99	Κανονικά Επίπεδα
100	Δηλητηρίαση από Μονοξείδιο του Άνθρακα

2) Ηλεκτροκαρδιογράφημα



Το ηλεκτροκαρδιογράφημα (ΗΚΓ) είναι ένα διαγνωστικό εργαλείο που χρησιμοποιείται συνήθως για την αξιολόγηση των ηλεκτρικών και μυϊκών λειτουργιών της καρδιάς. Ο αισθητήρας του ηλεκτροκαρδιογραφήματος (ΗΚΓ), έχει αναπτυχθεί για να είναι ένα από τα συνηθέστερα χρησιμοποιούμενα εργαλεία στη σύγχρονη ιατρική. Χρησιμοποιείται σε πλήθος καρδιακών παθολογικών καταστάσεων, που κυμαίνονται από ισχαιμία του μυοκαρδίου και έμφραγμα προς συγκοπή. Η ακρίβεια του ΗΚΓ εξαρτάται από την κατάσταση που δοκιμάζεται. Ένα πρόβλημα με την καρδιά μπορεί να μην παρουσιάζεται πάντα στο ΗΚΓ. Οι απαγωγές του ΗΚΓ συνδέονται με το σώμα, ενώ ο ασθενής βρίσκεται σε επίπεδη κατάσταση.

Από το ΗΚΓ μπορεί να μετρήσει κανείς πλήθος παθολογικών καταστάσεων όπως: ο προσανατολισμός της καρδιάς στην κοιλότητα του θώρακα, υποτροφία του καρδιακού μυ, απόδειξη βλάβης στα διάφορα μέρη του μυοκαρδίου, εάν υπάρχει μειωμένη ροή αίματος στην καρδιά, ανώμαλη ηλεκτρική δραστηριότητα που αλλοιώνουν τον καρδιακό ρυθμό κτλ.



Η τοποθέτηση των ηλεκτροδίων παρουσιάζεται στην δεξιά εικόνα, όπως παρουσιάστηκε στην πρώτη ενότητα με το τρίγωνο Eindhoven και τις τρεις απαγωγές:

3) Αισθητήρας Ροής Αέρα

Η παρακολούθηση του αναπνευστικού ρυθμού κρίνεται σημαντική, αφού οι μεταβολές του αναπνευστικού ρυθμού είναι από τις πρώτες ενδείξεις αστάθειας του σώματος. Ο αισθητήρας ροής αέρα μπορεί να παρέχει έγκαιρη προειδοποίηση της υποξυγοναιμίας και της άπνοιας. Από τον



αισθητήρα μετράται ο ρυθμός αναπνοής σε έναν ασθενή, τοποθετώντας κάποιες προεξοχές στα ρουθούνια της μύτης. Ένα κανονικό ενήλικο άνθρωπο έχει ένα αναπνευστικό ρυθμό 15-30 αναπνοών ανά λεπτό.

Κατώφλια (Thresholds)	
Μονάδα Μέτρησης = breath per minute	
< 30	Χαμηλά Επίπεδα
15 – 30	Κανονικά Επίπεδα
> 30	Ψηλά Επίπεδα

4) Θερμοκρασία Σώματος



Η θερμοκρασία του σώματος εξαρτάται από την θέση του σώματος στο οποίο γίνεται η μέτρηση, την ώρα της ημέρας και το επίπεδο δραστηριότητας του ατόμου. Διάφορα μέρη του σώματος έχουν διαφορετικές θερμοκρασίες.

Η κοινώς αποδεκτή μέση θερμοκρασία του πυρήνα του σώματος (που λαμβάνονται εσωτερικά) είναι 37,0 ° C (98,6 ° F). Αυξομειώσεις στην θερμοκρασία παρατηρούνται καθ' όλη την διάρκεια της ημέρας με χαμηλότερες θερμοκρασίες το πρωί και υψηλότερες θερμοκρασίες αργά το απόγευμα και το βράδυ.

Η θερμοκρασία του σώματος έχει υψίστη ιατρική σημασία, αφού ένας αριθμός ασθενειών συνοδεύονται από χαρακτηριστικές αλλαγές στη θερμοκρασία του σώματος. Άρα η αποτελεσματικότητα μιας θεραπείας και η παρακολούθηση ορισμένων ασθενειών μπορεί να παρακολουθούνται με την μέτρηση της θερμοκρασίας.

Κατώφλια (Thresholds) Μονάδα Μέτρησης = °C	
Υποθερμία	<35.0 °C (95.0 °F)
Κανονική Θερμοκρασία	36.5–37.5 °C (97.7–99.5 °F)
Πυρετός ή Υπερθερμία	>37.5–38.3 °C (99.5–100.9 °F)
Υπερπυρεξία	>40.0–41.5 °C (104–106.7 °F)

5) Πίεση Αίματος



Η αρτηριακή πίεση είναι η πίεση του αίματος στις αρτηρίες, καθώς αντλείται σε όλο το σώμα από την καρδιά. Όταν η καρδιά χτυπάει, συστέλλεται και ωθεί το αίμα μέσω των αρτηριών προς το υπόλοιπο σώμα. Η δύναμη αυτή δημιουργεί πίεση στις αρτηρίες. Πίεση του αίματος καταγράφεται ως δύο αριθμοί,

της συστολικής πίεσης (όπως η καρδιά χτυπά) πάνω από την διαστολική πίεση (όπως η καρδιά χαλαρώνει ανάμεσα στους χτύπους).

Η παρακολούθηση της αρτηριακής πίεσης στο σπίτι είναι σημαντική για πολλούς ανθρώπους, ειδικά αν έχετε υψηλή αρτηριακή πίεση. Η πίεση του αίματος αλλάζει καθ' όλη την διάρκεια της ημέρας. Επηρεάζεται από διάφορους παράγοντες, όπως η θέση του σώματος, η αναπνοή ή συναισθηματική κατάσταση, η άσκηση και ο ύπνος. Για την μέτρηση της αρτηριακής πίεσης συνίσταται να είσαστε χαλαροί και ξαπλωμένοι ή καθιστοί.

Κατώφλια (Thresholds)		
Μονάδα Μέτρησης = (mm Hg)		
	Συστολική Πίεση	Διαστολική Πίεση
Υπόταση	< 90	< 60
Επιτρεπτά Όρια	90–119	60–79
Προϋπέρταση	120–139	80–89
Στάδιο 1 Υπέρτασης	140–159	90–99
Στάδιο 2 Υπέρτασης	160–179	100–109
Υπερτασική Κρίση	≥ 180	≥ 110

6) Διαγωγιμότητα Δέρματος



Η διαγωγιμότητα του δέρματος, επίσης γνωστή ως γαλβανική αντίδραση του δέρματος (GSR) είναι μια μέθοδος μέτρησης της ηλεκτρικής διαγωγιμότητας του δέρματος, η οποία ποικίλλει ανάλογα με το επίπεδο υγρασίας του. Αυτό είναι ενδιαφέρον, επειδή οι ιδρωτοποιοί αδένες που ελέγχονται από το συμπαθητικό νευρικό σύστημα, σε έντονες συναισθηματικές φορτίσεις, μεταβάλλουν την ηλεκτρική αντίσταση του δέρματος. Έτσι, η διαγωγιμότητα του δέρματος χρησιμοποιείται ως ένδειξη της ψυχολογικής ή φυσιολογικής διέγερσης.

Η αρχή ή θεωρία πίσω από τη λειτουργία του αισθητήρα γαλβανικής απόκρισης είναι η μέτρηση της ηλεκτρικής αντίστασης του δέρματος με βάση τον ιδρώτα που παράγεται από τον οργανισμό. Σε υψηλό επίπεδο εφίδρωσης, η ηλεκτρική αντίσταση του δέρματος μειώνεται, ενώ ένα στεγνό δέρμα έχει μεγαλύτερη αντίσταση. Συναισθήματα όπως ενθουσιασμός, άγχος, σοκ, κλπ μπορεί να οδηγήσουν στην διακύμανση της αγωγιμότητας του δέρματος. Μέτρηση της αγωγιμότητας του δέρματος είναι ένα από τα στοιχεία των συσκευών ανιχνευτή ψεύδους και χρησιμοποιούνται στην επιστημονική έρευνα της συναισθηματικής ή φυσιολογική διέγερση.

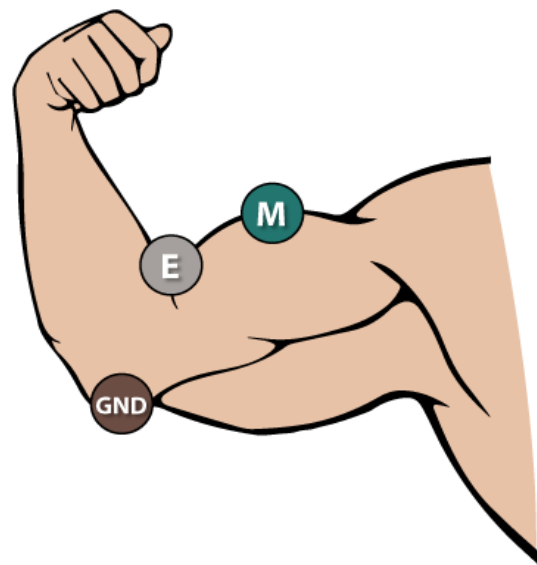
7) Ηλεκτρομυογράφημα



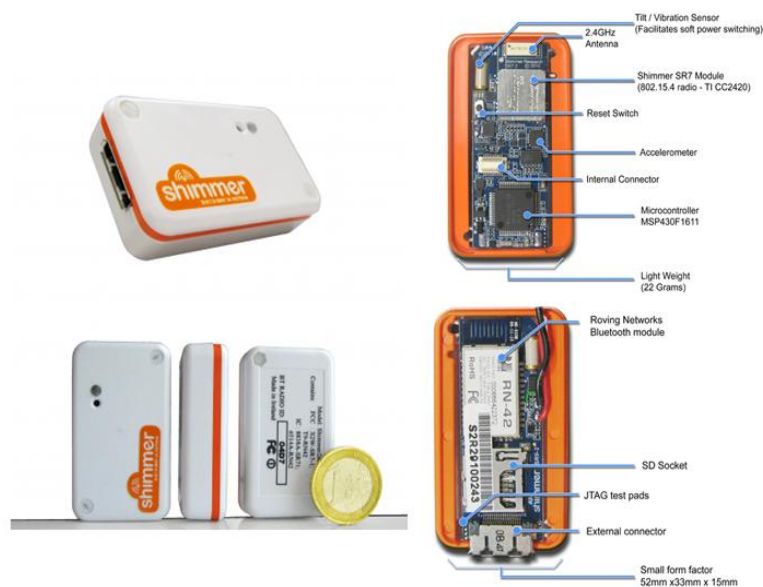
Το ηλεκτρομυογράφημα (EMG) μετρά την ηλεκτρική δραστηριότητα των μυών σε ηρεμία και κατά τη διάρκεια της συστολής. Το EMG είναι μια τεχνική για την αξιολόγηση και την καταγραφή της ηλεκτρικής δραστηριότητας που παράγεται από τους σκελετικούς μύες. Ένας ηλεκτρομυογράφος ανιχνεύει το ηλεκτρικό δυναμικό που παράγεται από τα μυϊκά κύτταρα όταν αυτά τα κύτταρα ενεργοποιηθούν και τα σήματα μπορούν να αναλυθούν για την ανίχνευση μυϊκών ανωμαλιών.

Τα σήματα EMG χρησιμοποιούνται σε πολλές κλινικές και βιοϊατρικές εφαρμογές. Το ηλεκτρομυογράφημα χρησιμοποιείται ως διαγνωστικό εργαλείο για τον εντοπισμό νευρομυϊκών παθήσεων, την αξιολόγηση χαμηλής οσφυαλγίας, κινησιολογίας, και διαταραχές του κινητικού ελέγχου.

Η τοποθέτηση των ηλεκτροδίων παρουσιάζεται δεξιά:



3.3.3 Εισαγωγή: Λειτουργιά Shimmer Sensor



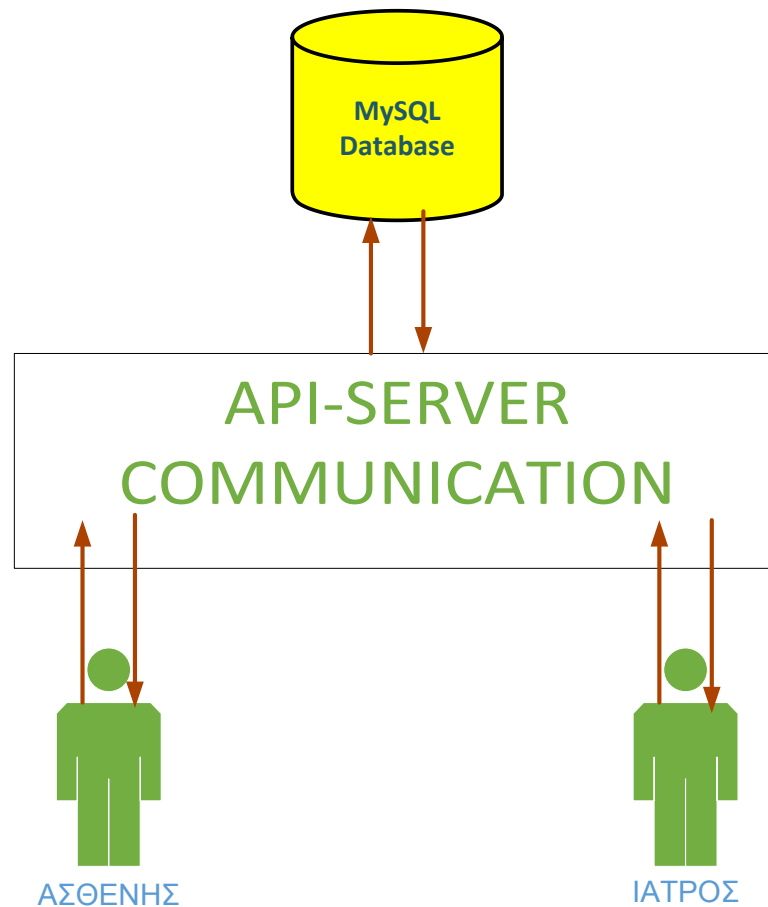
Εικόνα 3.1: Shimmer Sensor Engineering
http://zeitgeistlab.ca/doc/doc_images/wban05.png

Η εταιρεία Shimmer είχε σχεδιαστεί το 2006 αλλά εμφανίστηκε στην αγορά το 2008. Η πρωτοποριακή ιδέα της εταιρείας ήταν η κατασκευή ασύρματων αισθητήρων και άλλων ασύρματων προϊόντων τα οποία σήμερα πωλούνται σε πάνω από 60 χώρες ανά το παγκόσμιο. Κατά το πέρασμα του χρόνου υπήρξαν πολλές αναβαθμίσεις και καινοτομίες στα προϊόντα της εταιρείας, ορισμένα από τα προϊόντα της οποίας χρησιμοποιήσαμε για τους σκοπούς της δικής μας διπλωματικής εργασίας

Η πλατφόρμα της Shimmer συνδυάζει τηλεσκοπική υγεία με κινητικότητα και πειραματική επαναχρησιμοποίηση. Η Shimmer είναι μια μικρή ασύρματη αισθητήρια πλατφόρμα, που υποστηρίζει πλήθος εφαρμογών και δουλεύει με την τεχνολογία Bluetooth. Σε πραγματικό χρόνο παρέχει άμεσες πληροφορίες που αφορούν όχι μόνο το ηλεκτροκαρδιογράφημα (που αφορά την διπλωματική), αλλά επιπλέον πληροφορίες όπως επιταχυνσιογράφο, γυροσκόπιο, καρδιακούς παλμούς, πίεση αίματος, χρονική στιγμή λήψης δεδομένων κ.α. Τοποθετούνται τρία ηλεκτρόδια, που αντιπροσωπεύουν τις τρεις απαγωγές του ηλεκτροκαρδιογραφήματος, και έπειτα οι πληροφορίες παρουσιάζονται σε πραγματικό χρόνο στην εφαρμογή του κινητού τηλέφωνου.

Η Shimmer σχεδιάστηκε για να βοηθήσει στην δημιουργία ενός οικοσυστήματος των τεχνολογιών που αφορούν την υγεία, παρέχοντας φορητότητα και εύκολη ενσωμάτωση των υπάρχουσών τεχνολογιών και υποδομών.

3.4 Ανάλυση δραστηριοτήτων του συστήματος

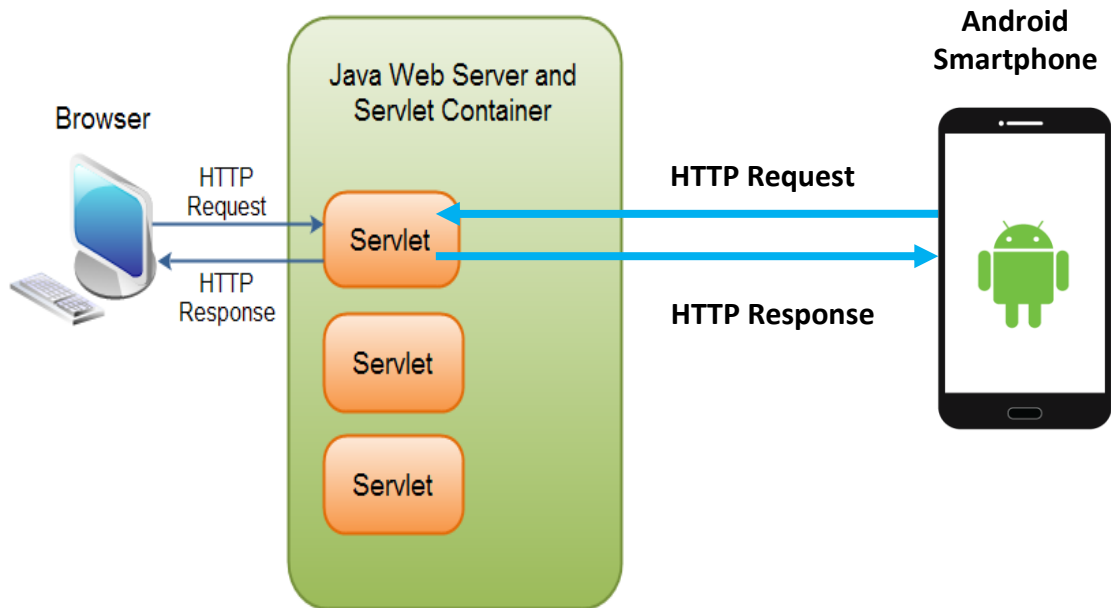


Ο ασθενής θα χρησιμοποιεί τα παραπάνω μέσα (πλατφόρμα Shimmer – Arduino) για συλλογή των ιατρικών δεδομένων που επιθυμεί και ανάλογα με την περίπτωση. Έπειτα, το σύστημα θα επικοινωνεί με μια βάση δεδομένων και τα δεδομένα θα αποστέλλονται για μακρόχρονη αποθήκευση χρησιμοποιώντας ένα ενδιάμεσο API και με την τεχνολογία Wi-Fi. Επίσης, ο ασθενής θα έχει την δυνατότητα να ανακτήσει τα δεδομένα, βλέποντας το ιστορικό υγείας του ανά χρονική περίοδο επιθυμίας. Και όλα αυτά, με την χρήση του κινητού τηλεφώνου (Android smart phone).

Ο ιατρός θα μπορεί σε οποιαδήποτε χρονική στιγμή να παρακολουθεί τις πληροφορίες υγείας του ασθενούς, χωρίς την άμεση παρέμβαση του ασθενούς. Οι πληροφορίες θα ανακτώνται και στις δυο περιπτώσεις με την τεχνολογία Wi-Fi.

Ο τρόπος επικοινωνίας της βάσης δεδομένων με το κινητό τηλέφωνο πραγματοποιείται με την αποστολή HTTP πρωτοκόλλου. Αρχικά, στέλλεται ένα αίτημα

στο server, το οποίο μπορεί να είναι αίτημα Εισαγωγής, Επιλογής και Ενημέρωσης κάποιων εγγράφων στην βάση. Η εφαρμογή Android πρέπει να επικοινωνεί με την ίδια διεύθυνση IP (Internet Protocol Address) και το ίδιο port. Τα δεδομένα αποστολής και λήψης θα είναι της μορφής JSON.



Εικόνα 3.2: Servlet Communication

- 1) <http://tutorials.jenkov.com/images/java-servlets/overview.png>
- 2) <https://cdn4.iconfinder.com/data/icons/smart-phones-technologies/128/android-phone-color.png>

Κεφάλαιο 4

Ανάλυση Προδιαγραφών

4.1 Χαρακτηριστικά συστήματος	62
4.2 Λειτουργίες Συστήματος	63
4.2.1 Χρήστης / Ασθενής	63
4.2.1.1 Επιλογή Γλώσσας	63
4.2.1.2 Είσοδος στο Σύστημα	64
4.2.1.3 Εγγραφή Χρήστη	64
4.2.1.4 Επιλογή πλατφόρμας Shimmer / Arduino	64
4.2.1.5 Επιλογή συσκευής Shimmer	64
4.2.1.6 Επιλογή πλατφόρμας Arduino	64
4.2.1.7 Παρουσίαση βιομετρήσεων και καρδιογραφήματος ...	65
4.2.1.8 Παρουσίαση ιστορικού μετρήσεων ανά αισθητήρα	65
4.2.1.9 Αναφορά Περίληψης Μετρήσεων Ασθενή	65
4.2.1.10 Οδηγοί Χρήσεως / Βοήθεια	65
4.2.2 Χρήστης / Ιατρός	65
4.2.2.1 Εγγραφή Ιατρού στο σύστημα	65
4.2.2.2 Είσοδος στο σύστημα	66
4.2.2.3 Παρακολούθηση δεδομένων Ασθενούς	66
4.2.2.4 Επιλογή Χρήστη	66
4.2.2.5 Απεικόνιση Βιομετρήσεων	66
4.2.2.6 Εισαγωγή Σχόλιου	66
4.2.2.7 Περίληψη μετρήσεων ασθενούς	66

4.1 Χαρακτηριστικά συστήματος

Λόγω της αποστολής / ανάκτησης απόρρητων πληροφοριών υγείας του ασθενούς προς τον ιατρό, καθώς και η χρήση του συστήματος τόσο από τον ιατρό όσο και από τον ασθενή, τότε θα πρέπει να προϋπάρχουν κάποια σημαντικά χαρακτηριστικά, που να παρέχουν πλήρης ευχρηστία, αξιοπιστία και ασφάλεια στο σύστημα. Ειδικότερα, στον τομέα της ασφάλειας θα εξασφαλίζεται ότι τα δεδομένα θα αποστέλλονται εάν και μόνο εάν ο χρήστης είναι εξουσιοδοτημένος από το σύστημα,

δηλαδή να πραγματοποιήσει από πριν την απαραίτητη σύνδεση με την χρήση κωδικού πρόσβασης. Για άτομα που δεν είναι εξουσιοδοτημένα, το σύστημα δεν θα τους αφήσει να προχωρήσουν σε μετέπειτα ενέργειες. Επίσης, με κάθε νέα σύνδεση, το σύστημα θα κατοχυρώνει την ώρα και ημερομηνία εισόδου, πράγμα εξασφαλίζει την ακεραιότητα και διαθεσιμότητα του συστήματος προς τον χρήστη.

Στον τομέα της ευχρηστίας και αξιοπιστίας, θα πρέπει το σύστημα να μπορεί να χρησιμοποιείται τόσο από τον ιατρό, ο οποίος θα κατέχει κάποιες τεχνολογικές γνώσεις, όσο και από τον ασθενή, ο οποίος θα μπορεί να είναι οποιασδήποτε ηλικίας και κοινωνικής τάξης. Γι' αυτόν τον λόγο, η αλληλεπίδραση χρήστη - συστήματος πρέπει να γίνεται με τέτοιο τρόπο, ώστε να βοηθά τον χρήστη και να επικεντρώνεται στην υποκειμενική του ικανοποίηση. Η ενσωμάτωση ελληνικών και αγγλικών ορολογιών θα βοηθήσει στην σωστή επικοινωνία του συστήματος, καθώς και η χρήση γραφικών περιβάλλοντος και βοηθών εκμάθησης θα ενισχύσει αυτήν την προσπάθεια. Η χρήση κινητού τηλεφώνου για τον οποιοδήποτε ασθενή αποτελεί τεράστιο πλεονέκτημα, λόγω της μεγάλης απήχησης και συμβατότητας που θα έχει στο κοινό. Σε περίπτωση σφάλματος του συστήματος, θα πρέπει ο χρήστης να προχωρήσει στις απαραίτητες ενέργειες ανάκτησης του συστήματος, χωρίς την ψυχολογική κόπωση του χρήστη και πάντα με τα κατάλληλα μηνύματα επαναφοράς του συστήματος.

4.2 Λειτουργίες Συστήματος

Οι λειτουργίες του συστήματος καθορίζονται από τα άτομα που θα το χρησιμοποιούν. Άρα μας αφορούν οι διεπιφάνειες χρήσης των ασθενών και ιατρών.

Ειδικότερα:

4.2.1 Χρήστης / Ασθενής:

4.2.1.1 Επιλογή Γλώσσας

Ο χρήστης αρχικά επιλέγει την γλώσσά επιλογής του (ελληνικά ή αγγλικά), καλύπτοντας και έτσι άτομα που υστερούν σε μια από τις δυο γλώσσες. Έτσι, ο χρήστης ξεφεύγει από δύσκολες ορολογίες και ορισμούς ιατρικών λέξεων.

4.2.1.2 Είσοδος στο Σύστημα

Η είσοδος στο σύστημα πραγματοποιείται με την χρήση εισαγωγής κωδικού πρόσβασης και όνομα χρηστή (username). Για την σύνδεση στο σύστημα απαιτείται η χρήση της ασύρματης τεχνολογίας Wi-Fi (σε περίπτωση που δεν υπάρχει σύνδεση Wi-Fi, το σύστημα θα ενημερώνει τον χρηστή να συνδεθεί).

4.2.1.3 Εγγραφή Χρήστη

Ο χρήστης θα πρέπει να πραγματοποιήσει, πριν από την είσοδο στο σύστημα, την απαραίτητη συμπλήρωση των προσωπικών του πληροφοριών. Οι πληροφορίες του χρήστη αποθηκεύονται μόνιμα στην βάση δεδομένων, όπου στην είσοδο στο σύστημα ανακτώνται το username και password για σωστή ταυτοποίηση.

4.2.1.4 Επιλογή πλατφόρμας Shimmer / Arduino

Με την είσοδο στο σύστημα, ο χρήστης καλείται να επιλέξει με πια συσκευή επιθυμεί να συλλέξει τα ιατρικά δεδομένα. Έχει την δυνατότητα να επιλέξει είτε την συσκευή της Shimmer για την καταγραφή ηλεκτροκαρδιογραφήματος, είτε να επιλέξει την πλατφόρμα Arduino για καταγραφή ηλεκτροκαρδιογραφήματος και άλλων βιομετρήσεων, σύμφωνα με την ενσωμάτωση των κατάλληλων αισθητήρων.

4.2.1.5 Επιλογή συσκευής Shimmer

Επιλέγοντας την καταγραφή ηλεκτροκαρδιογραφήματος με την συσκευή Shimmer, ο χρήστης αρχικά καλείται να επιλέξει μια συσκευή, η οποία λειτουργεί με την τεχνολογία Bluetooth και επίσης προϋποθέτει πως είναι ήδη συσκευή σύζευξης με το κινητό τηλέφωνο (Bluetooth Pair Device). Εάν το Bluetooth του κινητού είναι ενεργοποιημένο, τότε ο χρήστης καλείται να πραγματοποιήσει σύνδεση με την συσκευή της Shimmer. Όταν πραγματοποιηθεί η σύνδεση, στην συνέχεια θα εκτελέσει την συλλογή δεδομένων και το καρδιογράφημα θα εμφανίζεται στην οθόνη. Τα δεδομένα που θα καταγράφονται θα στέλνονται απευθείας στην βάση δεδομένων.

4.2.1.6 Επιλογή πλατφόρμας Arduino

Επιλέγοντας την πλατφόρμα Arduino για καταγραφής βιομετρήσεων και ηλεκτροκαρδιογραφήματος, αρχικά ενεργοποιείται αυτόματα η σύνδεση με Wi-Fi hotspot και έπειτα τα δεδομένα συλλέγονται αυτόματα από την συσκευή προς το

κινητό. Καθώς τα δεδομένα θα στέλνονται, θα συναθροίζονται και μετά από ένα λεπτό συλλογής δεδομένων, θα υπολογίζεται ο μέσος όρος κάθε κατηγορίας δεδομένου και θα στέλνεται απευθείας στην βάση δεδομένων. Έτσι, θα ελαχιστοποιούμε τα σφάλματα θορύβου ή αλλοιωμένων μεμονωμένων μετρήσεων.

4.2.1.7 Παρουσίαση των βιομετρήσεων και καρδιογραφήματος

Καθώς τα δεδομένα θα συλλέγονται, θα εμφανίζεται σε πραγματικό χρόνο και σε ξεχωριστές γραφικές παραστάσεις, η διακύμανση των τιμών που λαμβάνομαι ανά πάσα χρονική στιγμή.

4.2.1.8 Παρουσίαση του ιστορικού μετρήσεων ανά αισθητήρα

Ο χρήστης έχει την δυνατότητα να επιλέξει ένα συγκεκριμένο τύπο μέτρησης, που τον αφορά περισσότερο, και να εμφανίσει ολόκληρο το ιστορικό των μετρήσεων, το οποίο ανακτάται από την βάση δεδομένων. Η απεικόνιση θα γίνεται σε ειδικά διαμορφωμένη γραφική παράσταση και με την χρήση την τεχνολογίας Wi-Fi.

4.2.1.9 Αναφορά Περίληψης Μετρήσεων Ασθενή

Ο μέσος όρος των μετρήσεων που θα υπολογίζεται ανά λεπτό, θα παρουσιάζεται στην οθόνη του κινητού, και με μια επιπλέον λειτουργία, θα υπάρχει αυτόματη φωνητική ενημέρωση για την τρέχων κατάσταση του ασθενούς και τι πρέπει να προσέξει.

4.2.1.10 Οδηγοί Χρήσεως / Βοήθεια

Σε κάθε δραστηριότητα της εφαρμογής θα υπάρχουν τα κατάλληλα μηνύματα λειτουργίας και χρησιμοποίησης της εφαρμογής καθώς και οδηγοί χρήσεως, ώστε να αποφευχθούν τυχόν σφάλματα και για σωστή καθοδήγηση του χρήστη.

4.2.2 Χρήστης / Ιατρός

4.2.2.1 Εγγραφή Ιατρού στο σύστημα

Ο ιατρός θα πρέπει να πραγματοποιήσει, πριν από την είσοδο στο σύστημα, την απαραίτητη συμπλήρωση των προσωπικών του πληροφοριών. Οι πληροφορίες του χρήστη αποθηκεύονται μόνιμα στην βάση δεδομένων, όπου στην είσοδο στο σύστημα ανακτώνται το username και password για σωστή ταυτοποίηση.

4.2.2.2 Είσοδος στο σύστημα

Η είσοδος στο σύστημα πραγματοποιείται με την χρήση εισαγωγής κωδικού πρόσβασης και όνομα χρηστή (username). Έπειτα ο ιατρός θα έχει την δυνατότητα επιλογής ασθενούς (με βάση το username του) και να παρακολουθούσα την πορεία των δεδομένων του ασθενούς. Θα υπάρχει και η επιλογή της αποσύνδεσης.

4.2.2.3 Παρακολούθηση δεδομένων Ασθενούς

Ο ιατρός θα μπορεί να παρακολουθούσα όλες τις μετρήσεις του ασθενούς, όπως ηλεκτροκαρδιογράφημα, καρδιακούς παλμούς, θερμοκρασία σώματος, οξυγόνωσης αίματος, συστολικής και διαστολικής πίεσης, παροχής αέρα, ηλεκτρομυογραφήματος και διαγωγιμότητας δέρματος.

4.2.2.4 Επιλογή Χρήστη

Ο ιατρός θα έχει μπροστά του μια λίστα με τους ασθενείς που επιβλέπονται από αυτόν. Θα επιλέγει από αυτή τη λίστα για ποιόν θέλει να δει τα δεδομένα του, ακόμα και να πραγματοποιήσει αναζήτηση για κάποιον ασθενή.

4.2.2.5 Απεικόνιση Βιομετρήσεων

Ο ιατρός θα μπορεί να παρακολουθούσα ξεχωριστά κάθε βιομέτρηση που καταγράφηκε από τον συγκεκριμένο ασθενή και να προβάλλεται με την μορφή γραφικής παράστασης στον χρόνο. Τα δεδομένα θα ανακτώνται από την βάση δεδομένων, όπου πρώτα καταγράφηκαν οι βιομετρήσεις του ασθενούς.

4.2.2.6 Εισαγωγή Σχόλιου

Ο ιατρός θα μπορεί να εισάγει ένα σχόλιο στον χρήστη είτε με κάποιες οδηγίες ή κάποιες διορθώσεις, τα όποια θα ενημερώνουν τον ασθενή για το πώς προχώρα η κατάσταση υγείας του.

4.2.2.7 Περίληψη μετρήσεων ασθενούς

Ο ιατρός θα έχει την δυνατότητα να δημιουργήσει ένα αρχείο το οποίο θα έχει όλες τις βιομετρήσεις και τα προσωπικά του στοιχεία του ασθενή. Έπειτα, θα μπορεί να στέλνει το αρχείο αυτό στον ασθενή για την άμεση ανταπόκριση σε κάποιο ιατρικό πρόβλημα.

Κεφάλαιο 5

Σχεδίαση Συστήματος

5.1 Εισαγωγή	67
5.2 Αναλυτική Σχεδίαση Εφαρμογής Android	68
5.3 Σχεδίαση Βάσης Δεδομένων – Server	74
5.4 Σχεδίαση Web App	77

5.1 Εισαγωγή

Κατά την φάση σχεδίασης και ανάπτυξης, ορίζονται επιτυχώς οι απαιτήσεις των χρηστών και οι προδιαγραφές, που καθορίστηκαν στα πιο πάνω κεφάλαια. Σ ‘ αυτήν την φάση, ο προγραμματιστής θα προσαρμόσει τις απαιτήσεις και προδιαγραφές, σκιαγραφώντας πως οι χρήστες θα αλληλεπιδρούν σωστά με την εφαρμογή. Αρχικά, μιλώντας για το μέρος των ασθενών, θα παρουσιάσω μια αναλυτική σχεδίαση της εφαρμογής Android, όπου με αρκετή λεπτομέρεια θα προσπαθήσω να δώσω τα κατάλληλα συστατικά που θα απαρτίζουν τα επί μέρους τμήματα των δραστηριοτήτων. Κάθε δραστηριότητα (activity) θα πρέπει να σχεδιαστεί με ιδιαίτερη προσοχή, ώστε να προσφέρει στον χρήστη την λειτουργικότητα και την υποκειμενική ικανοποίηση, χωρίς να τον κουράζει στην χρήση και στα σφάλματα. Αν πραγματοποιηθεί σωστά η φάση της σχεδίασης, τότε σχεδόν άψογα θα πραγματοποιηθεί και η φάση της υλοποίησης, όπως θα δούμε και στο επόμενο κεφάλαιο.

Έπειτα, θα αναλυθεί η σχεδίαση της βάσης δεδομένων, όπου τα δεδομένα από τις βιομετρήσεις θα αποθηκεύονται αυτόματα, καθώς και προσωπικές πληροφορίες των χρηστών και ταυτόχρονα θα πρέπει να υπάρξει και η κατάλληλη ανάκτηση των δεδομένων, ανάλογα με την περίπτωση. Όλες αυτές οι λειτουργίες δεν θα γίνονταν δυνατές, εάν δεν υπήρχε η δυνατότητα σχεδίασης του server με την χρήση Java servlet API, που δέχεται αιτήματα σε μορφή JSON και απαντά πίσω με την ίδια μορφή.

Τέλος, θα αναλυθεί και το τμήμα σχεδίασης του Web App, που θα αφορά τον χρήστη / ιατρό, ο οποίος θα παρακολουθούσα και θα ενημερώνεται για τις ιατρικές εξελίξεις των ασθενών, άμεσα, εύκολα και χωρίς ιδιαίτερες χρονοτριβές.

5.2 Αναλυτική Σχεδίαση Εφαρμογής Android

❖ Κλάση Language.java

Η κλάση Language είναι υπεύθυνη για την επιλογή της γλώσσας από τον χρήστη. Ο χρήστης θα μπορεί να επιλέξει ανάμεσα στις γλώσσες Ελληνικά και Αγγλικά. Με την επιλογή της κατάλληλης γλώσσας, τα μηνύματα βοήθειας, καθώς και οι οδηγοί χρήσεως και τα γραφικά στο περιβάλλον της εφαρμογής θα προσαρμόζονται στην κατάλληλη γλώσσά. Επίσης, αυτή η κλάση θα καλωσορίζει τον χρήστη στην εφαρμογή.

❖ Κλάση Login.java

Η κλάση Login είναι η σημαντικότερη κλάση, γιατί ο χρήστης δεν θα έχει την δυνατότητα να προχωρήσει στην υπόλοιπη εφαρμογή, εάν πρώτα δεν πραγματοποιήσει σύνδεση με τον Server, και ταυτοποιηθεί εάν ο χρήστης είναι εξουσιοδοτημένος να προχωρήσει. Αρχικά, ο χρήστης εισάγει στα κελιά συμπλήρωσης το username και password, και με το πάτημα ενός κουμπιού, θα προσπαθήσει να ενωθεί με το σύστημα. Στην περίπτωση που η σύνδεση Wi-Fi είναι απενεργοποιημένη, τότε η εφαρμογή θα ενημερώνει τον χρήστη να πραγματοποιήσει πρώτα σύνδεση Wi-Fi. Επίσης, θα υπάρχει μήνυμα σε περίπτωση που ο χρήστης πραγματοποιήσει σύνδεση και τα κελιά συμπλήρωσης είναι κενά. Εάν όλα είναι συμπληρωμένα σωστά, τότε ελέγχεται η εγκυρότητα των δεδομένων εισαγωγής και γίνεται επικύρωση από την βάση δεδομένων του συστήματος. Αν η επικύρωση πραγματοποιηθεί με επιτυχία, τότε γίνεται σύνδεση στο σύστημα και παρουσιάζεται κατάλληλο μήνυμα επιτυχίας, ενώ αν υπάρξει μη - εξουσιοδοτημένη πρόσβαση ή πρόβλημα με τον server, τότε η εφαρμογή θα ενημερώνει με κατάλληλα μηνύματα τον χρήστη. Επίσης, θα υπάρχει ένα ακόμα κουμπί, όπου τα δεδομένα συμπλήρωσης θα καθαρίζονται αυτόματα.

Με την σωστή εξουσιοδότηση από το σύστημα, ο χρήστης καλείται να επιλέξει με ποια συσκευή επιθυμεί να συνεχίσει: Shimmer ή Arduino, εμφανίζοντας κατάλληλο μήνυμα.

❖ Κλάση Register.java

Η κλάση Register είναι η κλάση όπου ο ασθενής μπορεί να εισάγει προσωπικές πληροφορίες, οι οποίες θα αποθηκεύονται και θα ανακτώνται από την βάση δεδομένων. Οι πληροφορίες αυτές απαρτίζονται από: όνομα , επίθετο , αριθμό ταυτότητας, φύλο, ημερομηνία γεννήσεως, διεύθυνση σπιτιού, αριθμό σπιτιού, ταχυδρομικό κώδικα, πόλη διαμονής, χώρα διαμονής, αριθμό τηλεφώνου, username, password και διεύθυνση ηλεκτρονικού ταχυδρομείου. Θα υπάρχει ειδικό κουμπί, όπου τα δεδομένα συμπλήρωσης θα καθαρίζονται αυτόματα. Μόλις ο χρήστης έχει συμπληρώσει όλα τα στοιχεία του, θα γίνεται έλεγχος εάν υπάρχει κάποιο κελί συμπλήρωσης που είναι κενό. Εάν ναι, τότε θα εμφανίζεται κατάλληλο μήνυμα. Εάν όχι, τότε με το πάτημα ενός κουμπιού, οι πληροφορίες θα στέλνονται στην βάση για μόνιμη αποθήκευση. Σε περίπτωση, που τα δεδομένα δεν αποθηκευτήκαν ή υπήρξε κάποιο πρόβλημα με τον server, τότε θα εμφανίζεται κατάλληλο μήνυμα σφάλματος. Αλλιώς, εάν τα δεδομένα αποθηκευτήκαν σωστά, τότε ένα μήνυμα επιτυχίας θα εμφανίζεται στην οθόνη.

❖ Κλάση Shimmer.java

Η κλάση της Shimmer καθορίζει την καταγραφή ηλεκτροκαρδιογραφήματος από τις συσκευές της εταιρείας Shimmer. Μόλις εκκινήσει αυτή η κλάση, θα εμφανίζεται ένας σύντομος οδηγός χρήσεως, στην γλωσσά επιλογής του χρήστη, ώστε ο χρήστης να μπορεί να αλληλεπιδράσει σωστά με το σύστημα. Θα υπάρχουν 7 διαφορετικά κουμπιά στην κλάση αυτή, τα όποια είναι: εμφάνιση συσκευής σύζευξης, αποθήκευση δεδομένων, σύνδεση, αποσύνδεση, συλλογή δεδομένων, τερματισμός συλλογής δεδομένων και εμφάνιση ηλεκτροκαρδιογραφήματος.

1. Κουμπί Εμφάνισης Συσκευής Σύζευξης:

Το κουμπί αυτό θα εμφανίζει όλες τις συσκευές που λειτουργούν με τεχνολογία Bluetooth, σαν λίστα, με το δικαίωμα επιλογής με ποια συσκευή ο χρήστης επιθυμεί να συνεχίσει. Για να εμφανιστούν οι συσκευές σύζευξης, πρέπει να ενεργοποιηθεί το

Bluetooth του κινητού τηλεφώνου. Εάν είναι απενεργοποιημένο, τότε θα εμφανίζεται κατάλληλο μήνυμα ενεργοποίησής του.

Σημείωση: Για το πιο πάνω, προϋποθέτεται πως ο χρήστης πραγματοποίησε σύζευξη του κινητού τηλεφώνου με την συσκευή της Shimmer με την χρήση τεχνολογίας Bluetooth, πρώτου ξεκινήσει η εφαρμογή.

2. Κουμπί Αποθήκευσης Δεδομένων:

Τα δεδομένα που θα καταγράφονται από την συσκευή θα μπορούν είτε να αποθηκεύονται μόνιμα στην κάρτα μνήμης της συσκευής για μελλοντική χρήση με την μορφή .data αρχείου, είτε να εμφανίζονται μονό για προσωρινή χρήση.

3. Κουμπί Σύνδεσης με την συσκευή

Ο χρήστης, αφού πρώτα έχει επιλέξει μια ποια συσκευή σύζευξης επιθυμεί να συνεχίσει, τότε με το πάτημα του κουμπιού σύνδεσης, πραγματοποιείται η αναμονή της σύνδεσης και στην συνέχεια η σύνδεση στην συσκευή. Επειδή, υπάρχει η περίπτωση η συσκευή να χάσει κάποια σύνδεση, τότε θα εμφανίζεται ένα γραφικό περιβάλλον με χρώματα, όπου το κίτρινο χρώμα θα αναφέρεται πως αναμένεται κάποια σύνδεση, το πράσινο χρώμα πως η σύνδεση πραγματοποιήθηκε με επιτυχία, το μπλε χρώμα πως τα δεδομένα διαβάζονται με επιτυχία, και το κόκκινο χρώμα πως η σύνδεση ήταν αποτυχημένη ή ο χρήστης επιθυμεί να αποσυνδεθεί. Στην περίπτωση της σύνδεσης, θα εμφανιστεί το πράσινο χρώμα, με το κατάλληλο μήνυμα επιτυχίας.

4. Κουμπί Αποσύνδεσης από την συσκευή

Ο χρήστης με το πάτημα του κουμπιού θα μπορεί να αποσυνδεθεί από την συσκευή, εμφανίζοντας το κόκκινο χρώμα με το κατάλληλο μήνυμα αποσύνδεσης

5. Κουμπί Συλλογής Δεδομένων

Με το κουμπί συλλογής δεδομένων, δημιουργείται μια συνεχής ροή δεδομένων τιμών ηλεκτροκαρδιογραφήματος, όπου το γραφικό περιβάλλον παίρνει χρώμα μπλε και εμφανίζεται το κατάλληλο μήνυμα ροής δεδομένων.

6. Κουμπί Τερματισμού Συλλογής Δεδομένων

Με το κουμπί τερματισμού συλλογής δεδομένων, σταματά η συνεχής ροή δεδομένων τιμών ηλεκτροκαρδιογραφήματος, όπου το γραφικό περιβάλλον παίρνει πράσινο πάλι (η συσκευή παραμένει συνδεδεμένη με το κινητό) και εμφανίζεται το κατάλληλο μήνυμα τερματισμού δεδομένων.

7. Εμφάνιση Ηλεκτροκαρδιογραφήματος

Με το κουμπί αυτό εμφανίζεται η γραφική παράσταση του ηλεκτροκαρδιογραφήματος με τις αντίστοιχες τιμές να ενημερώνονται συνεχώς. Για να προχωρήσουμε στην εμφάνιση του ηλεκτροκαρδιογραφήματος πρέπει να υπάρχει πρώτα συνεχής ροή δεδομένων συλλογής. Οι τιμές που συλλέγονται στην συνεχεία, στέλνονται αυτόματα στην βάση δεδομένων, για μόνιμη αποθήκευση, μαζί με το αριθμό ταυτότητας του χρήστη και την ημερομηνία / ώρα που πραγματοποιήθηκε η μέτρηση. Εάν υπάρξει κάποιο πρόβλημα στην αποθήκευση, τότε θα εμφανίζεται κατάλληλο μήνυμα ειδοποίησης προς τον χρήστη.

❖ Κλάση Arduino.java

Η κλάση Arduino χρησιμοποιείται για την καταγραφή, όχι μόνο του ηλεκτροκαρδιογραφήματος, αλλά και άλλων επιπλέον βιομετρήσεων, όπως αυτές καθορίζονται από τους αισθητήρες της πλατφόρμας e-Health Sensor. Για σκοπούς διευκόλυνσης του χρήστη, θα ενεργοποιείται αυτόματα η σύνδεση Wi-Fi hotspot και θα απενεργοποιείται αυτόματα η σύνδεση Wi-Fi.

Σημείωση: Για την ένωση του Wi-Fi hotspot της συσκευής Arduino με το Wi-Fi hotspot της κινητής συσκευής, προϋποθέεται να υπάρχει το ίδιο αναγνωριστικό όνομα κατοχυρωμένο και στις 2 συσκευές.

Μόλις εκκινήσει αυτή η κλάση, θα εμφανίζεται ένας σύντομος οδηγός χρήσεως, στην γλωσσά επιλογής του χρήστη, ώστε ο χρήστης να μπορεί να αλληλεπιδράσει σωστά με το σύστημα.

Η κλάση Arduino θα απαρτίζεται από άλλες επιμέρους κλάσεις, όπου όλες μαζί θα συναθροίζουν την τελική σχεδίαση για την καταγραφή δεδομένων από την πλατφόρμα Arduino.

➤ **Κλάση Real-Time Data.java**

Σ ‘ αυτήν την κλάση καταγράφονται οι τιμές των εξής αισθητήρων: θερμοκρασίας σώματος, καρδιακοί παλμοί, οξυγόνωσης αίματος, παροχής αέρα από αναπνευστικές οδούς, συστολικής και διαστολικής πίεσης. Λόγω του ότι δεν είναι αναγκαίο να φαίνεται γραφικά η απεικόνιση των τιμών, θα πραγματοποιείται μόνο ενημέρωση των τιμών ανά δευτερόλεπτο.

➤ **Κλάση HistoryData.java**

Σ ‘ αυτήν την κλάση θα υπάρχει μια γραφική παράσταση, στην οποία θα απεικονίζονται οι τιμές ενός συγκεκριμένου αισθητήρα, που αποθηκευτήκαν στην βάση δεδομένων σε κάποια χρονική στιγμή. Λόγω του ότι θα πρέπει να ενεργοποιηθεί η σύνδεση Wi-Fi, ώστε να γίνει η αίτηση για ανάκτηση πληροφοριών από την βάση, τότε με κάποιο κατάλληλο μήνυμα θα ενημερώνεται ο χρήστης ότι θα πρέπει να απενεργοποιηθεί το Wi-Fi hotspot και να ενεργοποιηθεί η σύνδεση Wi-Fi. Μόλις πραγματοποιηθεί η σύνδεση Wi-Fi και αφού ο χρήστης επιλέξει πιο τύπο δεδομένων αισθητήρα επιθυμεί να δει και για ποια χρονική περίοδο, τότε με το πάτημα ενός κουμπιού, στέλλεται το αίτημα στην βάση και από την βάση τα δεδομένα ανακτώνται και παρουσιάζονται στην γραφική παράσταση. Με το κλείσιμο της κλάσης αυτής, θα πρέπει να επαναφέρεται αυτόματα η σύνδεση Wi-Fi hotspot, για

την συνέχιση της λήψης δεδομένων από την πλατφόρμα Arduino. Σε περίπτωση που δεν υπάρχουν δεδομένα ανάκτησης, θα εμφανίζεται κατάλληλο μήνυμα σφάλματος ανάκτησης.

➤ **Κλάση MuscleIntensity.java**

Σ' αυτήν την κλάση θα παρουσιάζονται οι τιμές του ηλεκτρομυογραφήματος σε μια γραφική παράσταση, μαζί με μια ειδικά ζωγραφισμένη γραμμή ζωνών, που αναφέρεται σε πιο σημείο βρίσκεται η κατάσταση του μυ και θα υπάρχει συνεχής ενημέρωση με τις τρέχων τιμές.

➤ **Κλάση ECG.java**

Σ' αυτήν την κλάση θα παρουσιάζονται οι τιμές του ηλεκτροκαρδιογραφήματος σε μια γραφική παράσταση, μαζί με την τιμή που θα λαμβάνεται από την πλατφόρμα Arduino την δεδομένη στιγμή.

➤ **Κλάση Conductivity.java**

Σ' αυτήν την κλάση θα παρουσιάζονται οι τιμές της διαγωγιμότητας του δέρματος σε μια γραφική παράσταση, μαζί με μια ειδικά ζωγραφισμένη γραμμή ζωνών, που αναφέρεται σε πιο σημείο βρίσκεται η κατάσταση της διαγωγιμότητας, η οποία θα ενημερώνεται συνεχώς με τις τρέχων τιμές.

➤ **Κλάση PatientSummary.java**

Σ' αυτήν την κλάση θα παρουσιάζεται ο μέσος όρος των τιμών που λαμβάνονται από την πλατφόρμα Arduino ανά λεπτό, και έπειτα θα υπάρχει άμεση ενημέρωση του χρήστη για τις νέες τιμές, καθώς και με το πάτημα ενός κουμπιού το σύστημα θα ενημερώνει φωνητικά (με λειτουργία text-to-speech) τον χρήστη για την τρέχων κατάσταση της υγείας του, με βάση κάποια κατώφλια που έθεσα στο σύστημα. Η κλάση θα διαθέτει ράβδο

προόδου, που θα απεικονίζει πόσα δεδομένα χρειάζεται ακόμα στο να συναθροιστούν τα δεδομένα και να βγει ο μέσος όρος.

Λόγω της λειτουργίας του Wi-Fi hotspot στην πλατφόρμα Arduino και μετά από συζητήσεις με την ομάδα του Δρ. Παττίχη, αποφάσισα τα δεδομένα που θα στέλνονται στην βάση για αποθήκευση να είναι στην ουσία ο μέσος όρος των τιμών που συναθροιστήκαν ανά λεπτό. Έτσι, για κάθε λεπτό, η εφαρμογή θα καταμετρά τα δεδομένα που λαμβάνει και έπειτα, μόλις το ένα λεπτό περάσει αυτόματα θα απενεργοποιείται το Wi-Fi hotspot και θα ενεργοποιείται το Wi-Fi, ώστε να σταλθούν τα δεδομένα στην βάση. Εάν υπάρξει κάποιο πρόβλημα με τον server ή κάποιο σφάλμα αποθήκευσης θα ενημερώνεται ο χρήστης με κατάλληλο μήνυμα αποτυχίας. Αλλιώς, εάν αποθηκευτούν, τότε θα εμφανίζεται μήνυμα επιτυχίας.

Μετά από αυτήν την φάση, θα ενεργοποιείται και πάλι αυτόματα το Wi-Fi hotspot, για να συνεχίσει η φάση της συλλογής δεδομένων, με νέες τιμές και νέους μέσους όρους. Μαζί με τους μέσους όρους, θα στέλνω στην βάση τον αριθμό ταυτότητας του χρήστη, καθώς και την ημερομηνία / ώρα καταγραφής των δεδομένων.

5.3 Σχεδίαση Βάσης Δεδομένων – Server

Για την σχεδίαση της βάσης δεδομένων, λαμβάνω υπόψη τον τρόπο που ο χρήστης θα αλληλεπιδρά με το σύστημα και ποιες λειτουργίες επιθυμεί να προσφέρει το σύστημα. Έτσι, για αυτόν τον σκοπό θα δημιουργήσω 4 πίνακες ως εξής:

Register User			
Column	Data Type	Primary Key	Not Null
User ID	INT(11)	✓	✓
Name	VARCHAR(45)		✓
Surname	VARCHAR(45)		✓
Gender	VARCHAR(6)		✓
Date of Birth	DATE		✓
Street	VARCHAR(45)		✓
Number of Street	INT(2)		✓

City	VARCHAR(45)		✓
Country	VARCHAR(45)		✓
Postal Code	INT(5)		✓
Telephone Number	INT(8)		✓
Email	VARCHAR(45)		✓
Username	VARCHAR(45)		✓
Password	VARCHAR(45)		✓

Στον πίνακα Register αποθηκεύονται οι προσωπικές πληροφορίες του χρήστη, όπως παρουσιάζονται πιο πάνω. Ο πίνακας αυτός θα έχει την χρησιμότητα του στο γεγονός πως στην περίπτωση που ο χρήστης πραγματοποιήσει σύνδεση με το σύστημα (login), τότε πρέπει το username και password του χρήστη να ταυτοποιηθούν πως υπάρχουν σαν εγγραφή, και εάν ναι, τότε να επιστραφεί ο αριθμός ταυτότητας του χρήστη, για να συνεχίσει ο χρήστης να χρησιμοποιεί την εφαρμογή.

User Login			
Column	Data Type	Primary Key	Not Null
User ID	INT(11)	✓	✓
Start Session	DATETIME	✓	✓
End Session	DATETIME		✓
State	VARCHAR(10)		✓

Στον πίνακα User Login, μόλις ο χρήστης πραγματοποιήσει εξουσιοδοτημένη πρόσβαση από το σύστημα, τότε ο αριθμός ταυτότητας του (που επιστρέφεται από την σωστή ταυτοποίηση username και password) μαζί με την ημερομηνία / ώρα εισόδου και εξόδου, θα αποθηκεύονται μέσα στην βάση. Αρχικά, η ημερομηνία / ώρα εισόδου και εξόδου θα είναι οι ίδιες τιμές, αλλά μόλις ο χρήστης επιθυμεί να πραγματοποιήσει αποσύνδεση, τότε το σύστημα θα ενημερώνει την συγκεκριμένη εγγραφή, ότι η ημερομηνία / ώρα εξόδου έχει αλλάξει. Επίσης, η στήλη State θα παίρνει 2 τιμές: Sign In (αν ο χρήστης πραγματοποίησε είσοδο), το οποίο στην συνέχεια θα ενημερωθεί και θα μετατραπεί σε Log Out, αν ο χρήστης πραγματοποίησε σωστή αποσύνδεση. Σε

περίπτωση, που ο χρήστης δεν πραγματοποιήσει σωστή αποσύνδεση από το σύστημα, θα παρατηρηθεί από την βάση δεδομένων.

Shimmer Measurements			
Column	Data Type	Primary Key	Not Null
User ID	INT(11)	✓	✓
Date Taken	DATETIME	✓	✓
ECG 1	FLOAT		✓
ECG 2	FLOAT		✓

Στον πίνακα Shimmer Measurements θα αποθηκεύονται οι τιμές, που θα συλλέγονται από την συσκευή ηλεκτροκαρδιογραφήματος της Shimmer. Κάθε εγγραφή θα έχει δυο πρωτεύοντα κλειδιά (που αποτελούνται από τον αριθμό ταυτότητας του χρήστη, σε συνδυασμό με την ημερομηνία/ώρα που πάρθηκε η μέτρηση), μαζί με τις 2 τιμές που λαμβάνονται από την συσκευή.

Arduino Measurements			
Column	Data Type	Primary Key	Not Null
User ID	INT(11)	✓	✓
Date Taken	DATETIME	✓	✓
Mean Value Temperature	FLOAT		✓
Mean Value Airflow	FLOAT		✓
Mean Value Blood Oxygen	FLOAT		✓
Mean Value Pulse	FLOAT		✓
Mean Value Systolic Pressure	FLOAT		✓
Mean Value Diastolic Pressure	FLOAT		✓
Mean Value Electrocardiograph	FLOAT		✓
Mean Value Electromyography	FLOAT		✓
Mean Value Conductivity	FLOAT		✓

Στον πίνακα Arduino Measurements θα αποθηκεύονται οι τιμές, που θα συλλέγονται από την πλατφόρμα Arduino . Κάθε εγγραφή θα έχει δυο πρωτεύοντα κλειδιά (που αποτελούνται από τον αριθμό ταυτότητας του χρήστη, σε συνδυασμό με την ημερομηνία/ώρα που πάρθηκε η μέτρηση), μαζί με τους 9 μέσους όρους τιμών που θα αντιστοιχούν στις μετρήσεις κάθε αισθητήρα ξεχωριστά.

Τόσο ο πίνακας Shimmer Measurements, όσο και ο πίνακας Arduino Measurements θα παρακολουθείται συχνά από τον ιατρό για εξακρίβωση κάποιου προβλήματος του ασθενούς, αλλά επίσης και ο ασθενής θα έχει την δυνατότητα να παρακολουθεί την πορεία των μετρήσεων του.

Σχεδίαση Server

Για την δημιουργία του server θα χρησιμοποιήσω την υπηρεσία servlets της Java. Τα servlets είναι κλάσεις της Java, τα οποία υλοποιούν HTTP αιτήματα. Για την υλοποίηση του server, θα χρησιμοποιήσω τις JAR βιβλιοθήκες, που δημιουργήθηκαν για τους σκοπούς του EHR. Οι κυρίες λειτουργίες που θα πραγματοποιούν οι κλάσεις του server είναι: INSERT, UPDATE και SELECT.

Με τις κλάσεις της INSERT, για όλους τους πίνακες της βάσης, θα αποθηκεύονται οι εγγραφές. Αντιστοίχως, για τις κλάσεις της UPDATE θα πραγματοποιείται ενημέρωση εγγράφων σε κάποια δεδομένα των στηλών και για τις κλάσεις της SELECT θα επιστρέφονται κάποια δεδομένα, που θα αφορούν συγκεκριμένες λειτουργίες της εφαρμογής. Στην συνέχεια, μέσω του Apache Tomcat, θα τρέξουμε τα servlets και Java Server Pages (JSP).

5.4 Σχεδίαση Web App

Για την σχεδίαση του Web App, καθόρισα τις απαιτήσεις και τις προδιαγραφές που απευθύνονται προς τον ιατρό. Όπως ανέφερα στα προηγούμενα κεφάλαια, ο ιατρός θα έχει την γενική εποπτεία του ασθενούς. Θα μπορεί να παρακολουθεί τον ασθενή ανά πάσα χρονική στιγμή και να επιβλέπει τις λεπτομέρειες υγείας του ασθενούς, δίνοντας του τις απαραίτητες ιατρικές συμβουλές.

Από αυτά προκύπτει πως ο ιατρός θα πρέπει να πραγματοποιεί εξουσιοδοτημένη είσοδο στο σύστημα. Θα πρέπει να υπάρχουν και τα στοιχεία του ιατρού στην βάση, ώστε να μπορεί να υπάρχει ανάκτηση των προσωπικών και ιατρικών πληροφοριών.

Ιδιαίτερα:

- ❖ Ο ιατρός θα πρέπει να εισέρχεται εξουσιοδοτημένα στο σύστημα ή να πραγματοποιεί εγγραφή προσωπικών δεδομένων στην βάση.
- ❖ Να μπορεί να παρακολουθούτα τα προφίλ των ασθενών του, παρουσιάζοντας τις προσωπικές πληροφορίες (εμπιστευτικά).
- ❖ Να παρακολουθούτα τις γραφικές παραστάσεις των καταγεγραμμένων μέσων τιμών από τους αισθητήρες ανά διαφορετική χρονική στιγμή και ανά αισθητήρα.
- ❖ Να μπορεί να συμβουλεύει τον ασθενή για τυχόν προβλήματα υγείας, μέσω περίληψης των βιομετρήσεων του ασθενούς.
- ❖ Να υπάρχει σωστός διαχωρισμός μεταξύ διαφορετικών ασθενών, ώστε να μην εμπλέκονται τα προσωπικά στοιχεία των ασθενών μεταξύ τους.
- ❖ Να υπάρχει η δυνατότητα να επιλέξει από πλήθος ασθενών, και να πραγματοποιεί εύρεση από μια λίστα ασθενών.

Κεφάλαιο 6

Υλοποίηση και Αξιολόγηση Συστήματος

6.1 Εισαγωγή	79
6.2 Βήματα υλοποίησης συστήματος	80
6.3 Έλεγχοι συστήματος	81
6.3.1 Ανάνηψη από σφάλματα	81
6.3.2 Έλεγχος Ασφάλειας	82
6.3.3 Έλεγχος Ορθότητας Δεδομένων	82
6.4 Παρουσίαση εφαρμογής με Screenshots του τελικού προϊόντος	82
6.4.1 Έναρξη Εφαρμογής	82
6.4.2 Επιλογή Γλώσσας	82
6.4.3 Σύνδεση Χρήστη	83
6.4.4 Εγγραφή Χρήστη	83
6.4.5 Ηλεκτροκαρδιογράφημα από συσκευή Shimmer	86
6.4.6 Παρουσίαση Ηλεκτροκαρδιογραφήματος Shimmer	87
6.4.7 Ηλεκτροκαρδιογράφημα-βιομετρήσεις από Arduino	92
6.4.8 Παρουσίαση βιομετρήσεων από πλατφόρμα Arduino	95
6.5 Αξιολόγηση Συστήματος από χρήστες	99

6.1 Εισαγωγή

Κατά την φάση υλοποίησης και αξιολόγησης του συστήματος, ορίζονται επιτυχώς οι απαιτήσεις των χρηστών, οι προδιαγραφές, καθώς και ο σχεδιασμός του συγκεκριμένου συστήματος, όπως καθορίστηκαν όλα στα πιο πάνω κεφάλαια. Σ' αυτήν την φάση, ο προγραμματιστής, έχοντας στην διάθεση του κάποια εργαλεία υλοποίησης (όπως ECLIPSE IDE , MySQL κτλ), θα δώσει πνοή στην εφαρμογή, κτίζοντας τα επί μέρους τμήματα για τον σχηματισμό της. Καθώς το πρόγραμμα θα υλοποιείται, τεραστία έμφαση θα δοθεί στους ελέγχους του συστήματος, ώστε οποιοσδήποτε χρήστης να μπορεί να αντεπεξέλθει σε τεχνητά σφάλματα και να μην ενισχύεται η ψυχολογική κόπωση από την χρήση της εφαρμογής.

Ιδιαίτερα, θα δοθούν οι ελέγχοι ανάνηψης από σφάλματα, δηλαδή να μπορεί ο χρήστης να «ξεφεύγει» εάν κάτι δεν πάει καλά με τρέξιμο της εφαρμογής χωρίς να χρειάζεται να τερματίζει η εφαρμογή αυτόματα, οι ελέγχοι ασφάλειας, όπου πρόσβαση και αποθήκευση των δεδομένων θα έχουν μόνο εξουσιοδοτημένα άτομα, και οι ελέγχοι ορθότητας δεδομένων, για την σωστή εισαγωγή και καταγραφή των δεδομένων εισόδου και βιομετρήσεων.

Έπειτα, θα υπάρχουν τα screenshots της εφαρμογής, καθώς και λειτουργικές επεξηγήσεις για το πώς ο κάθε χρήστης θα μπορεί να αλληλεπιδρά με το γραφικό περιβάλλον. Τέλος, ο χρήστης (ασθενής και ιατρός) θα προβεί σε μια γενική αξιολόγηση για την εικόνα της εφαρμογής, εάν οι απαιτήσεις και οι προδιαγραφές καλυφτήκαν πλήρως και εάν, το προϊόν μπορεί να συνεισφέρει στον τομέα της ηλεκτρονικής υγείας.

6.2 Βήματα υλοποίησης συστήματος

Για την υλοποίηση ολοκλήρου του συστήματος, χρειάστηκε να κατανοήσω την χρήση κάποιων εξειδικευμένων εργαλείων και έπειτα προχώρησα βήμα προς βήμα στο κτίσιμο του συστήματος.

➤ Βήμα 1:

Εκμάθηση και κατανόηση της γλώσσας Java, για την σύνταξη κώδικα Android.

Για την υλοποίηση χρησιμοποίησα το εργαλείο Eclipse και συγκεκριμένα την έκδοση Eclipse for mobile developers και του plug-in Android SDK.

➤ Βήμα 2:

Σχεδίαση και ανάπτυξη της βάσης δεδομένων του συστήματος με την βοήθεια του εργαλείου MySQL

➤ Βήμα 3:

Υλοποίηση του server, με την σύνταξη κώδικα για HTTP αιτήματα για INSERT, UPDATE και SELECT.

➤ **Βήμα 4:**

Ένωση της βάσης δεδομένων με server, ώστε τα αιτήματα της μορφής JSON να στέλλονται και να επεξεργάζονται από τον server, ο οποίος server θα εκτελεί τις εντολές και θα προβαίνει στις απαραίτητες ενέργειες αποθήκευσης, ανάκτησης, και ενημέρωσης δεδομένων στην βάση. Οι ενέργειες λαμβάνονται από τον Apache Tomcat και τη χρήση servlet.

➤ **Βήμα 5:**

Η εφαρμογή Android θα αποστέλλει μέσω της σύνδεσης Wi-Fi, τα απαραίτητα HTTP αιτήματα, ενώνοντας έτσι την εφαρμογή με τον server.

6.3 Έλεγχοι συστήματος

6.3.1 Ανάληψη από σφάλματα

Μεγάλο μέρος της υλοποίησης της εφαρμογής, το αφιέρωσα στην υποκειμενική ικανοποίηση του χρήστη. Κάθε δράση και πλοήγηση του χρήστη, ίσως προκαλέσει την εμφάνιση κάποιου σφάλματος, γι ' αυτό ως προγραμματιστής μεσολάβησα στο να αποφεύγει ο χρήστης τυχόν σφάλματα αλληλεπίδρασης και δικτύων. Οι ασύρματες συνδέσεις Wi-Fi , Wi-Fi hotspot και Bluetooth συνοδεύονται με τα κατάλληλα μηνύματα ελέγχου και σε παρουσίαση κάποιου σφάλματος, υπάρχει έλεγχος για μη-τερματισμό της εφαρμογής.

Επίσης, στην ένωση της εφαρμογής με τον server (που συνοδεύεται και με την αποστολή HTTP αιτημάτων σε μορφή JSON), πραγματοποιείται σημαντικός έλεγχος, σε περίπτωση που κάτι δεν πάει καλά με την αποστολή των αιτημάτων ή τα δεδομένα στην βάση δεν μπορούν να ανακτηθούν ή να αποθηκευτούν. Έπειτα μετά από αμέτρητους ελέγχους, παρατηρήσαμε πως οι συσκευές, τόσο της Shimmer όσο και της πλατφόρμας Arduino συμπεριφέρονταν σωστά στις καταγραφές των τιμών, με ακρίβεια και ευκρίνεια. Τα δεδομένα που συλλέγονταν, μπορούσαν να αποθηκευτούν στην βάση, χωρίς προβλήματα, και το σημαντικό να μπορούν να ανακτηθούν για την σωστή λειτουργικότητα της εφαρμογής. Σε περίπτωση σφάλματος, ο χρήστης ενημερώνεται άμεσα για να προβεί σε νέες ενέργειες.

6.3.2 Έλεγχος Ασφάλειας

Ο κύριος σκοπός της εφαρμογής είναι η συλλογή προσωπικών βιομετρήσεων από τις συσκευές. Γι αυτόν τον λόγο, η ασφάλεια και η εμπιστευτικότητα των δεδομένων παραμένει ένα μείζων θέμα. Όπως εξήγησα και στο κεφάλαιο της σχεδίασης, ο χρήστης θα πραγματοποιεί σύνδεση στο σύστημα (login) για να μπορεί να προχωρήσει στα επόμενα στάδια. Η χρήση προσωπικού username και password ενισχύει αυτόν τον σκοπό. Επίσης, τα δεδομένα που θα αποθηκεύονται στην βάση θα είναι εμπιστευτικά μόνο μεταξύ των ασθενών και του ιατρού.

6.3.3 Έλεγχος Ορθότητας Δεδομένων

Ο χρήστης θα καλείται πολλές φορές να συμπληρώσει κάποιες προσωπικές πληροφορίες που χρειάζεται το σύστημα. Έτσι, θα πρέπει με κατάλληλα μέσα να μειώνω την πιθανότητα του χρήστη να πραγματοποιεί σφάλματα εισαγωγής δεδομένων.

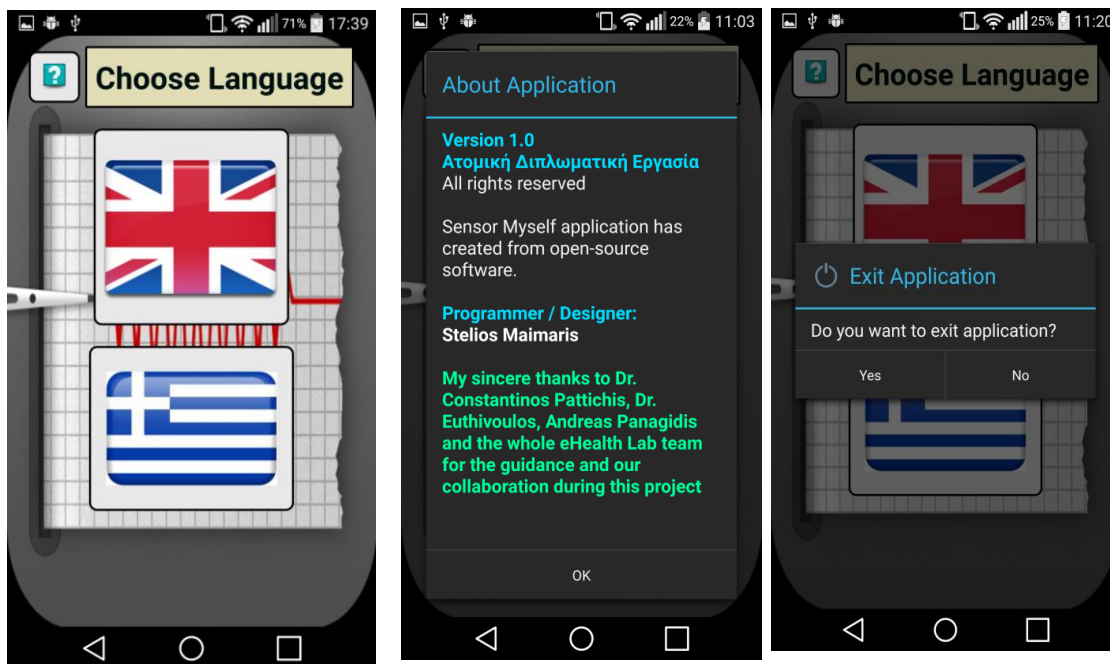
6.4 Παρουσίαση εφαρμογής με Screenshots του τελικού προϊόντος

6.4.1 Έναρξη Εφαρμογής



Με την έναρξη της εφαρμογής εμφανίζεται η αρχική εικόνα, που παρουσιάζει το ιερό φίδι του Ασκληπίου, μαζί με την ιατρική ατάκα “Your health, Our Mission”. Η εμφάνιση της αρχικής εικόνας διαρκεί μερικά δευτερόλεπτα μόνο.

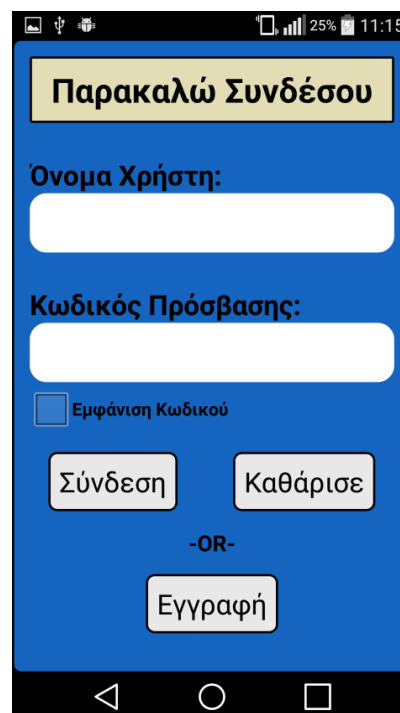
6.4.2 Επιλογή Γλώσσας

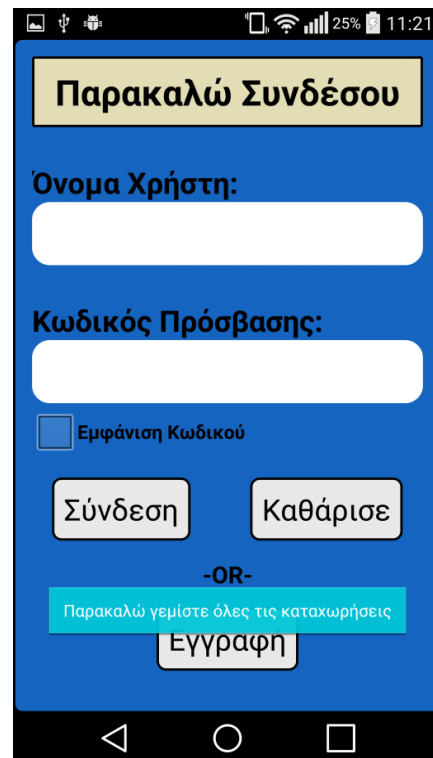
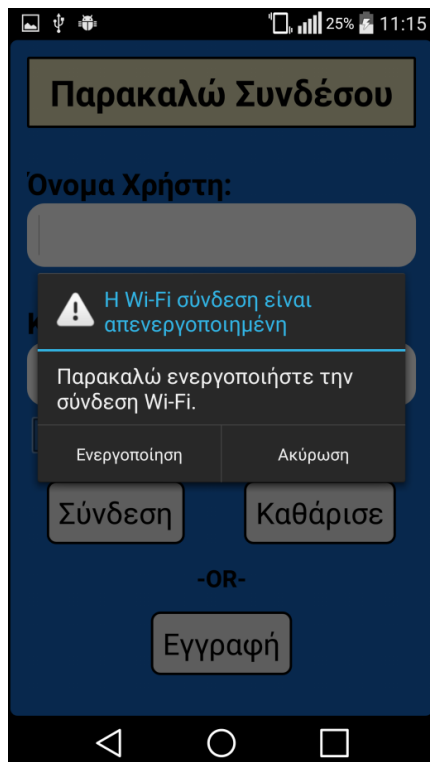


Ο χρήστης έχει την δυνατότητα να επιλέξει ανάμεσα σε δυο διαφορετικές γλώσσες: Ελληνικά και Αγγλικά. Αριστερά από τον τίτλο της διεπαφής υπάρχει ένα εικονικό κουμπί, όπου εμφανίζονται οι ευχαριστίες, το όνομα του προγραμματιστή και της εφαρμογής. Με το πλήκτρο επιστροφής της κινητής συσκευής εμφανίζεται το κατάλληλο μήνυμα που ενημερώνει τον χρήστη, αν επιθυμεί να εξέλθει από την εφαρμογή.

6.4.3 Σύνδεση Χρήστη

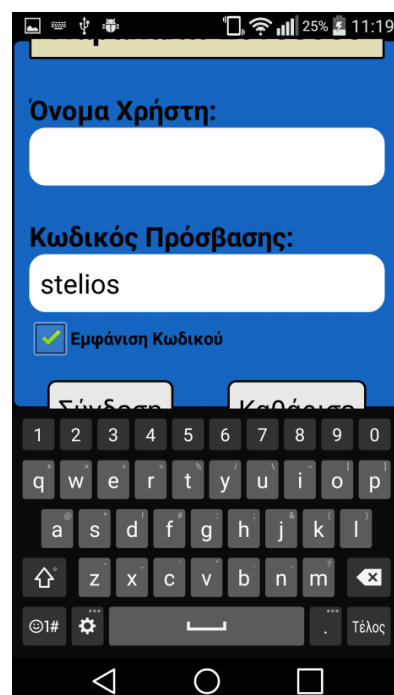
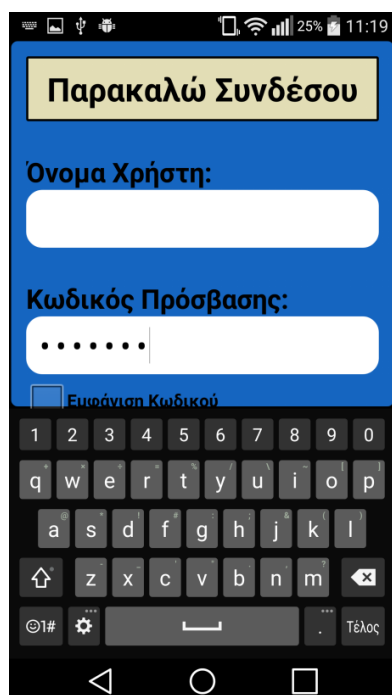
Όπως παρατηρούμε στα δεξιά, εμφανίζεται η δραστηριότητα εισόδου, όπου ο χρήστης καλείται να συμπληρώσει το username και password. Με το κουμπί «Σύνδεση», ο χρήστης μπορεί είτε να προχωρήσει στα επόμενα μέρη της εφαρμογής, είτε το σύστημα να μην τον αφήσει να συνεχίσει (μη-εξουσιοδοτημένη πρόσβαση). Το κουμπί «Καθάρισε» διαγραφεί τυχόν χαρακτήρες που υπάρχουν στα κελιά συμπλήρωσης.

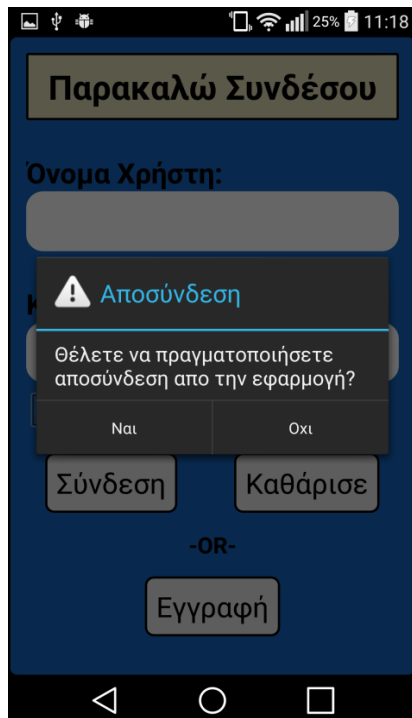




Αξίζει να σημειωθεί ο έλεγχος που πραγματοποιείται με το κουμπί «Σύνδεση». Για τον λόγο, πως η σύνδεση του χρήστη στον server προϋποθέτει και σύνδεση στο Wi-Fi, τότε όταν το Wi-Fi της συσκευής είναι απενεργοποιημένο, εμφανίζεται αμέσως ένα μήνυμα που ζητά εξουσιοδότηση για ενεργοποίηση του Wi-Fi.

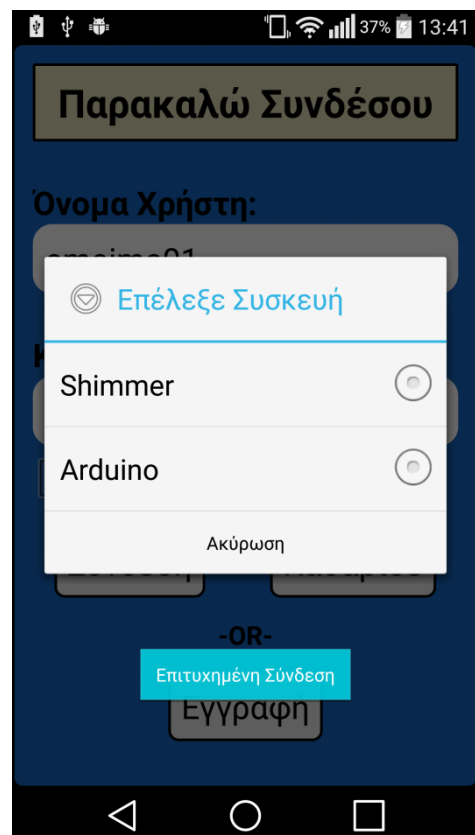
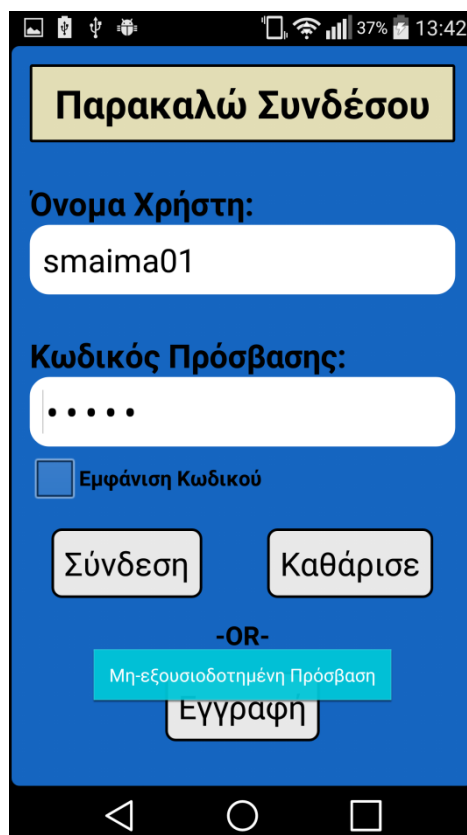
Ακόμη, όταν κάποιο από τα κελιά συμπλήρωσης είναι κενό, τότε κατάλληλο μήνυμα ενημερώνει τον χρήστη να συμπληρώσει όλα τα κελιά..





Το κουμπί ελέγχου «Εμφάνιση Κωδικού» εμφανίζει τον κρυμμένο κωδικό πρόσβασης

Ο χρήστης μπορεί να πραγματοποιήσει αποσύνδεση από το σύστημα, πατώντας το πλήκτρο επιστροφής της κινητής συσκευής. Όταν ο χρήστης βρίσκεται σε κατάσταση σύνδεσης, τότε με την αποσύνδεση του ενημερώνεται αυτόματα η βάση δεδομένων για την ημερομηνία / ώρα εξόδου και η κατάσταση αλλάζει από "Sign In" σε "Log Out".



Σε περίπτωση που τα στοιχεία εισαγωγής του χρήστη είναι εσφαλμένα, τότε εμφανίζεται κατάλληλο μήνυμα λάθους, ενώ σε αντίθετη περίπτωση, όταν η σύνδεση είναι επιτυχημένη, τότε εμφανίζεται κατάλληλο μήνυμα μαζί με το παράθυρο επιλογής συσκευών Shimmer ή Arduino.

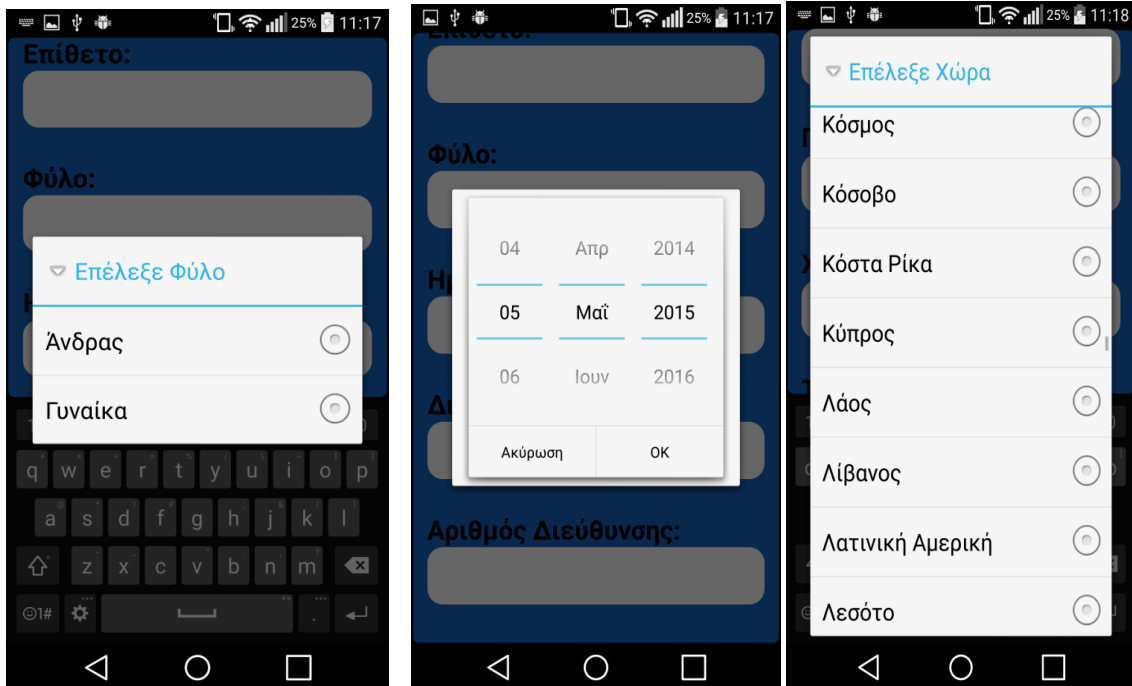
6.4.4 Εγγραφή Χρήστη

The image displays two screenshots of a mobile application interface for user registration. The left screenshot shows the 'Εγγραφή Χρήστη' (User Registration) screen with input fields for 'Ταυτότητα:', 'Όνομα:', 'Επίθετο:', and 'Φύλο:'. The right screenshot shows the same screen with additional fields for 'Αριθμός Τηλεφώνου:', 'Διεύθυνση Email:', 'Όνομα Χρήστη:', and 'Κωδικός Πρόσβασης:'. It also includes a checkbox for 'Εμφάνιση Κωδικού' and buttons for 'Εγγραφή' and 'Καθάρισε'.

Στην δραστηριότητα «Εγγραφή Χρήστη», ο χρήστης συμπληρώνει τις προσωπικές πληροφορίες, οι οποίες με το κουμπί «Εγγραφή» αποθηκεύονται στην βάση δεδομένων. Το κουμπί «Καθάρισε» διαγραφεί τυχόν χαρακτήρες που υπάρχουν στα κελιά συμπλήρωσης. Το κουμπί ελέγχου «Εμφάνιση Κωδικού» εμφανίζει τον κρυμμένο κωδικό πρόσβασης.

The image displays two screenshots of a mobile application interface for user registration. The left screenshot shows the 'Εγγραφή Χρήστη' (User Registration) screen with a numeric keypad for the password field. The right screenshot shows the same screen with the password field filled with '6767' and a confirmation message 'Παρακαλώ γεμίστε όλες τις καταχωρήσεις'.

Κάθε κελί συμπλήρωσης δίνει ένα διαφορετικό τρόπο εισαγωγής δεδομένων. Για παράδειγμα πατώντας το κελί συμπλήρωσης για αριθμό ταυτότητας εμφανίζεται το αριθμητικό πληκτρολόγιο, το κελί συμπλήρωσης για όνομα εμφανίζεται το αλφαβητικό πληκτρολόγιο κτλ. Επίσης, εάν κάποιο κελί είναι κενό, τότε θα εμφανίζεται κατάλληλο μήνυμα που θα ενημερώνει τον χρήστη.

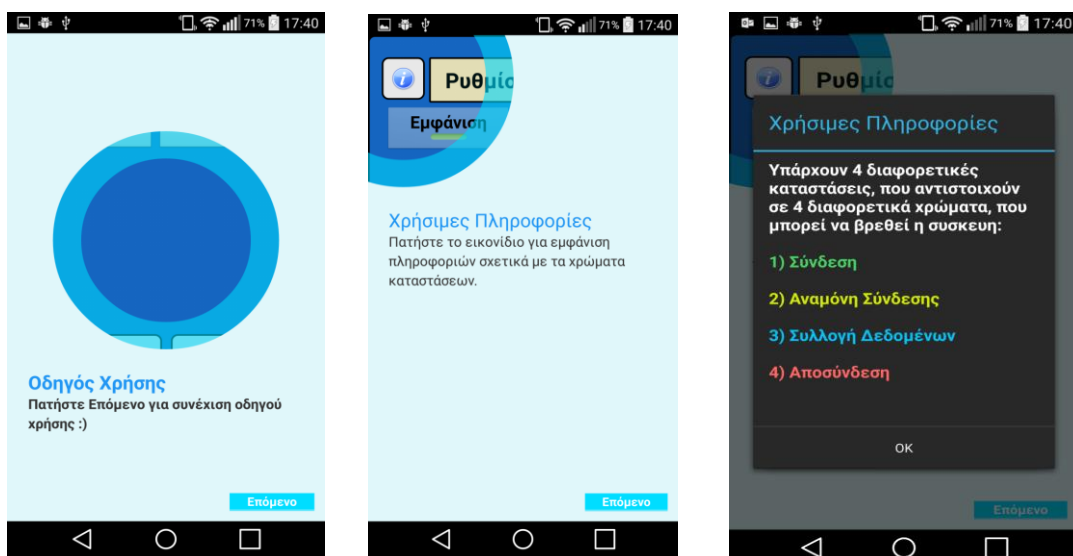


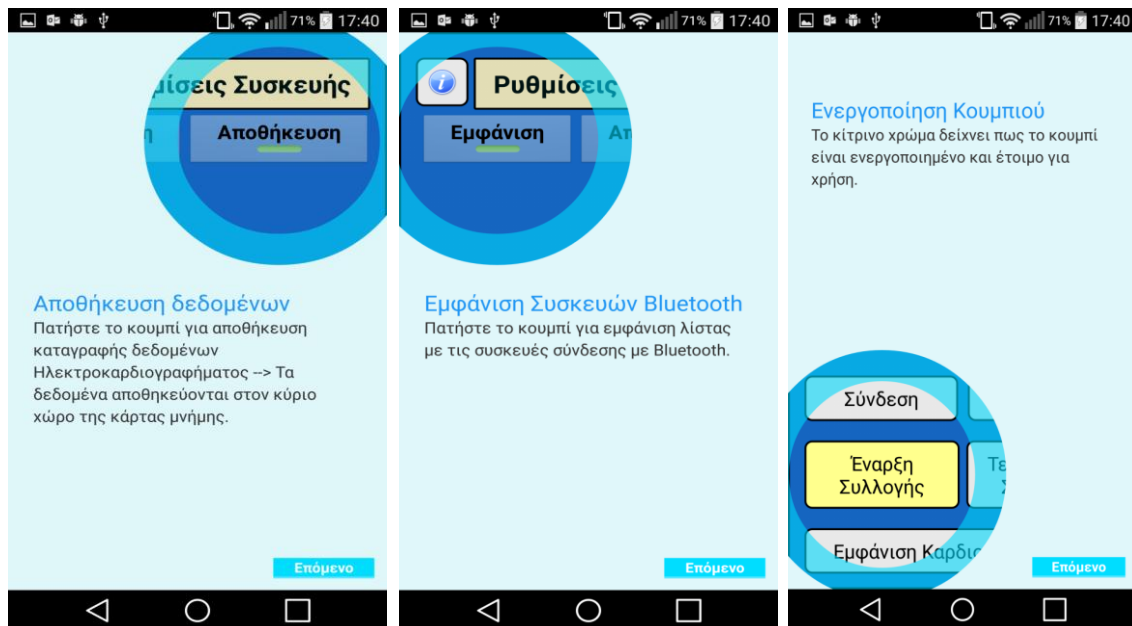
Επιλογή Φύλου Χρήστη

Επιλογή Ημερομηνίας Γεννήσεως

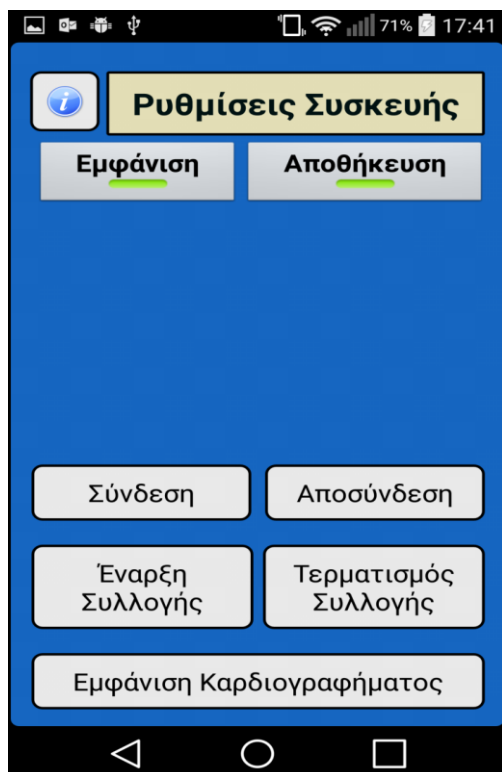
Επιλογή Χώρα

6.4.5 Ηλεκτροκαρδιογράφημα από συσκευή Shimmer

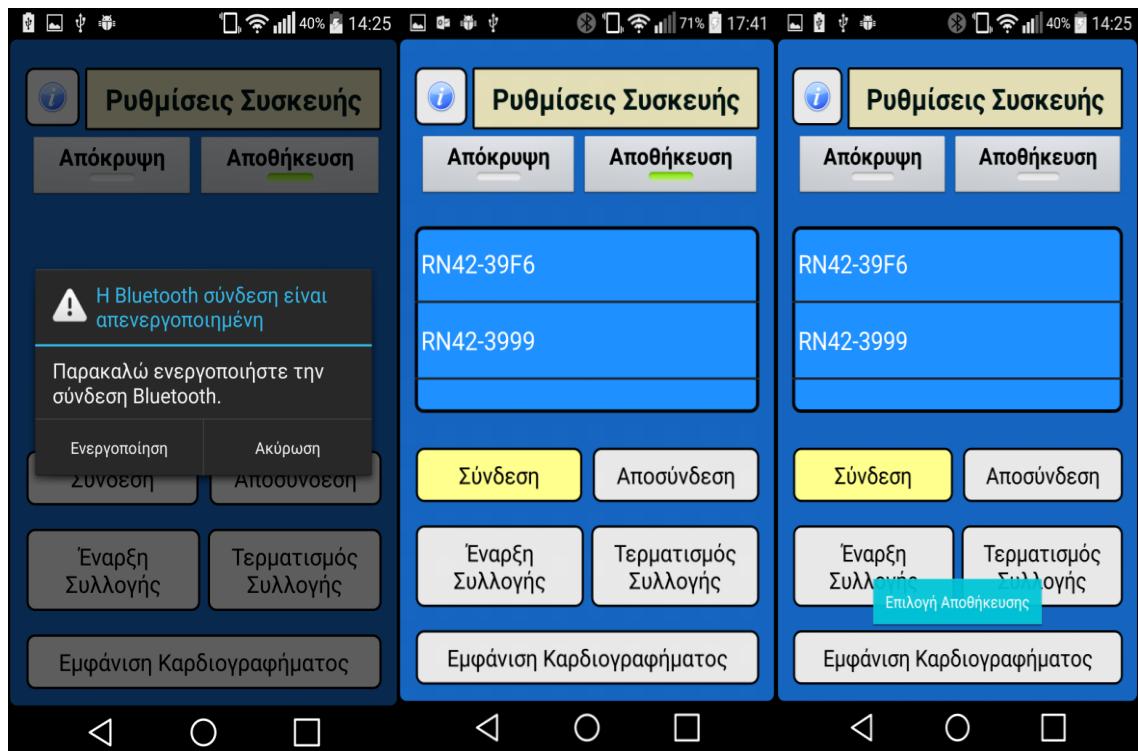




Με την έναρξη του τμήματος της συσκευής Shimmer, εμφανίζεται ένας σύντομος οδηγός χρήσης, με οδηγίες είτε στα Ελληνικά είτε στα Αγγλικά (εξαρτάται από την γλωσσά επιλογής στην αρχή της εφαρμογής). Στα αριστερά του τίτλου υπάρχει ένα εικονικό κουμπί που εξηγά τις διάφορες φάσεις χρωμάτων, που μπορεί να βρεθεί μια συσκευή της Shimmer(που θα εξηγηθούν παρακάτω).



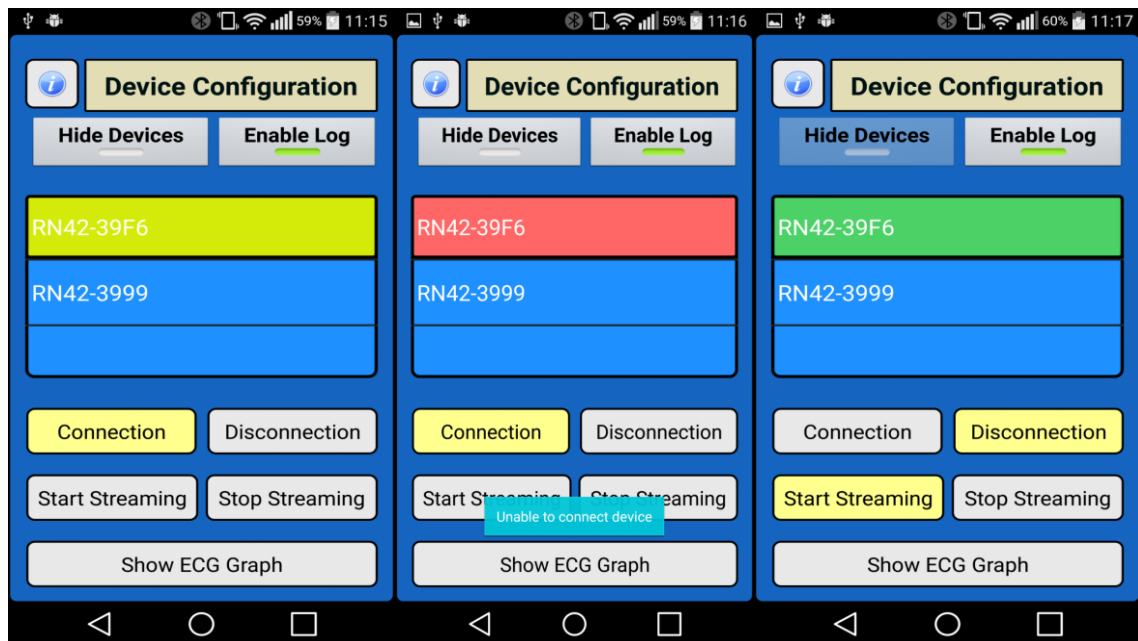
Με τον τερματισμό του οδηγού χρήσεως εμφανίζεται η αρχική οθόνη της συσκευής Shimmer. Παρατηρούμε 7 κουμπιά, τα οποία επιτελούν όλες τις κυρίες λειτουργίες που θα πραγματοποιούνται από την δραστηριότητα. Τα 5 κουμπιά Σύνδεση, Αποσύνδεση, Έναρξη Συλλογής, Τερματισμός Συλλογής και Εμφάνιση Καρδιογραφήματος παραμένουν απενεργοποιημένα, ενώ τα κουμπιά Εμφάνιση και Αποθήκευση είναι ενεργοποιημένα.



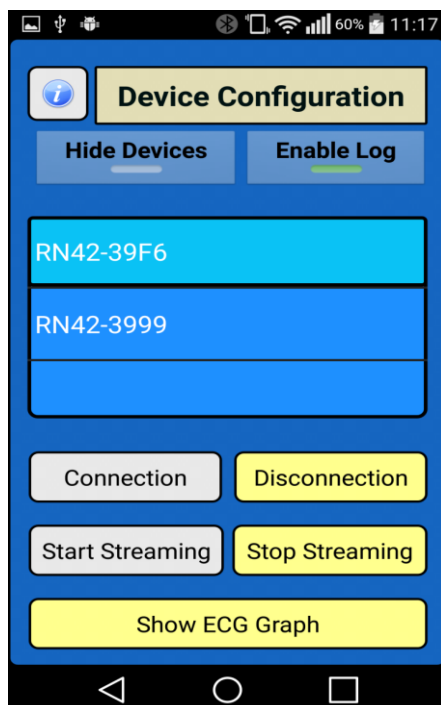
Όταν ο χρήστης πατήσει στο κουμπί Εμφάνιση, τότε εάν το Bluetooth είναι απενεργοποιημένο, η εφαρμογή ζητά να ενεργοποιηθεί το Bluetooth. Με την ενεργοποίηση του, τότε εμφανίζεται ο πίνακας με τα ονόματα των συσκευών Bluetooth που βρίσκονται σε σύζευξη με το κινητό τηλέφωνο. Παρατηρούμε πως το κουμπί Σύνδεση χρωματίστηκε κίτρινο, που σημαίνει πως είναι έτοιμο για χρήση. Επίσης, πατώντας το κουμπί της Αποθήκευσης εμφανίζεται κατάλληλο μήνυμα στην οθόνη.

Μόλις επιλέγει κάποια συσκευή από τον πίνακα, τότε στο κάτω μέρος, με κατάλληλο μήνυμα, εμφανίζεται η διεύθυνση της συσκευής.



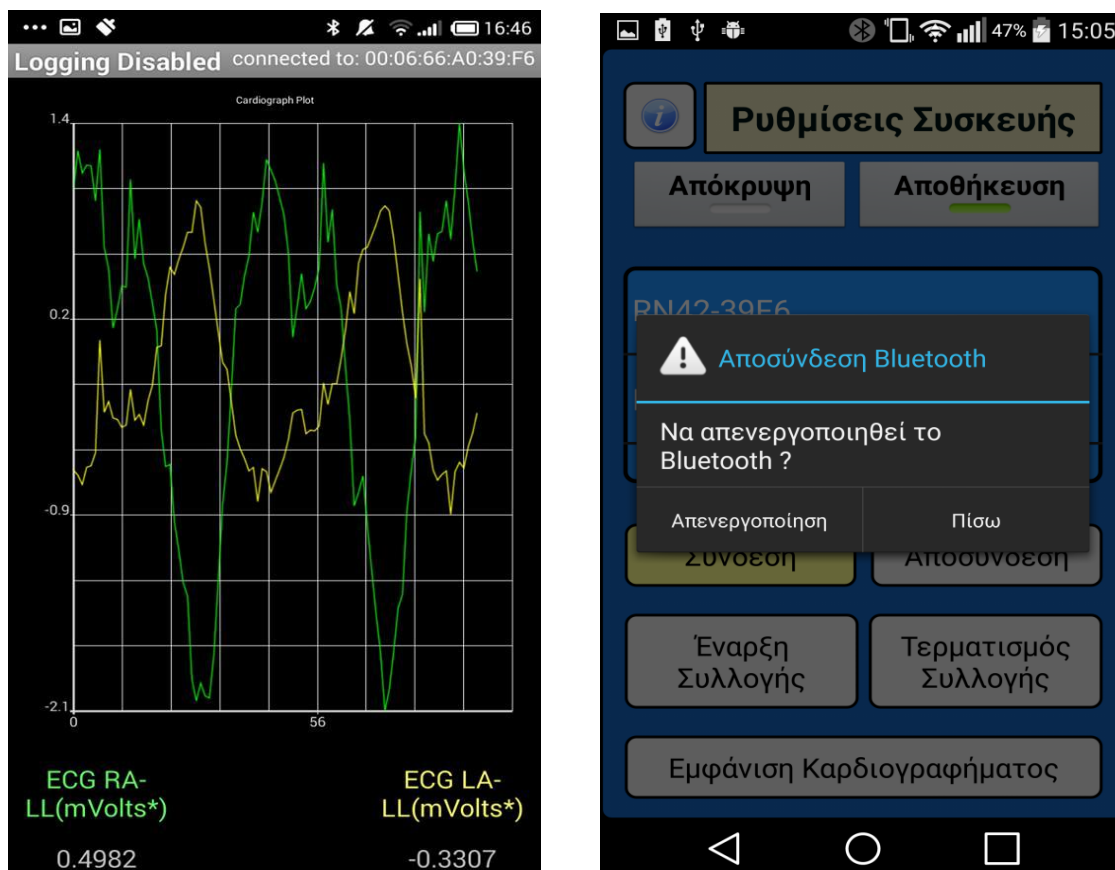


Μετά την επιλογή μιας εκ των συσκευών, θα προχωρήσουμε στην φάση της Σύνδεσης. Πατώντας το κουμπί Σύνδεση, αυτόματα η συσκευή που επιλέγηκε χρωματίζεται κίτρινο, που σημαίνει ότι αναμένεται η σύνδεση. Όταν μετά από κάποιο χρονικό διάστημα, το χρώμα μετατραπεί σε κόκκινο σημαίνει ότι απέτυχε η εφαρμογή να συνδεθεί με την συσκευή. Σε αντίθετη περίπτωση, εάν το χρώμα γίνεται πράσινο, τότε η εφαρμογή συνδέθηκε επιτυχώς και ταυτόχρονα ενεργοποιούνται τα κουμπιά Έναρξη Συλλογής και Αποσύνδεσης και απενεργοποιείται το κουμπί Σύνδεση.



Με το κουμπί Έναρξη Συλλογής να ενεργοποιείται, τότε το χρώμα της συσκευής γίνεται μπλε και ταυτόχρονα ενεργοποιούνται και τα κουμπιά Τερματισμός Συλλογής και Εμφάνιση Καρδιογραφήματος και απενεργοποιείται η Έναρξη Συλλογής.

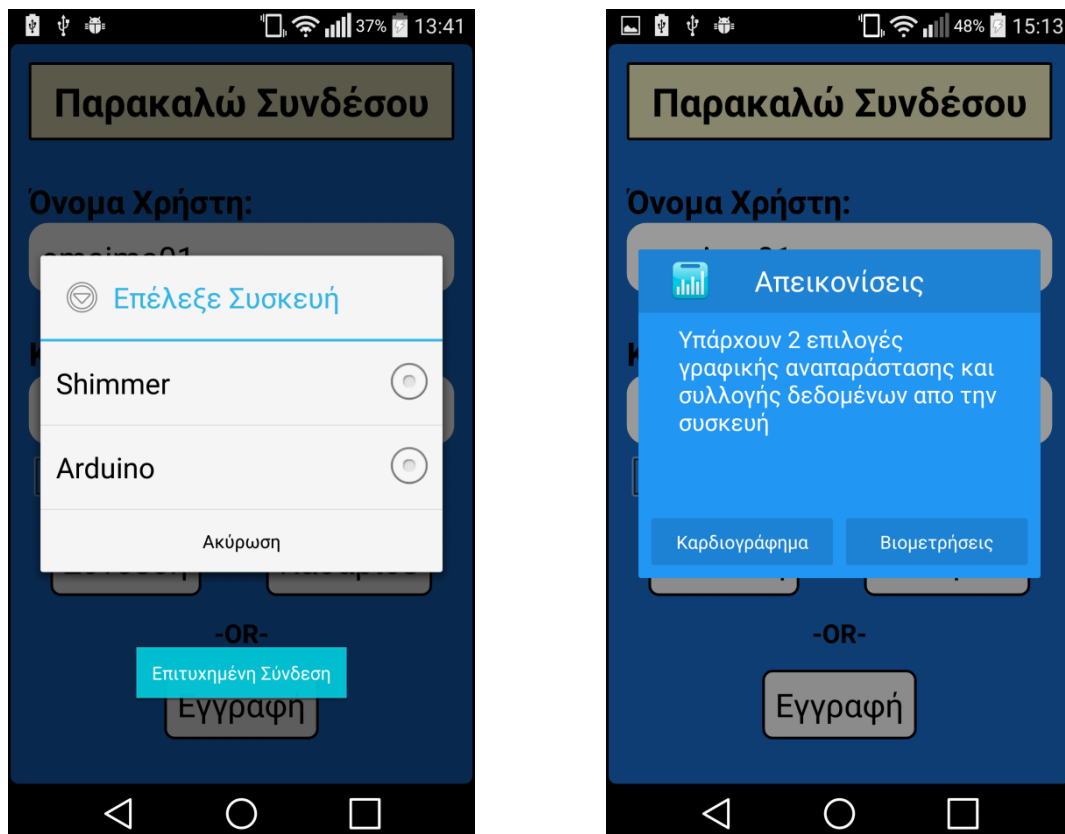
6.4.6 Παρουσίαση Ηλεκτροκαρδιογραφήματος Shimmer



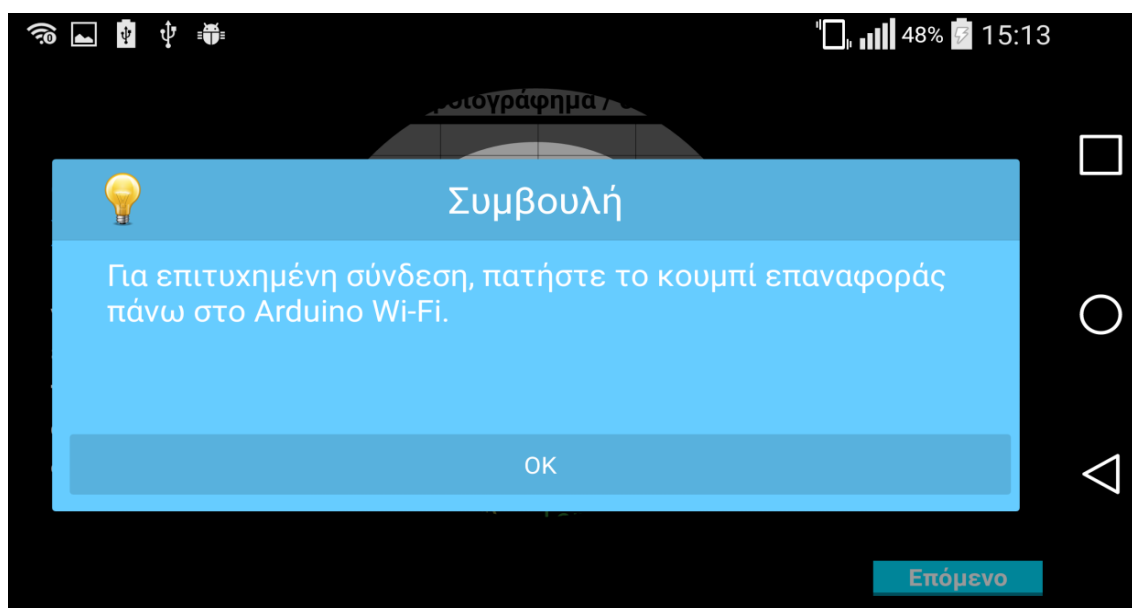
Με την ενεργοποίηση του κουμπιού Εμφάνιση Καρδιογραφήματος, τότε εμφανίζεται η γραφική απεικόνιση του καρδιογραφήματος της Shimmer, με ταυτόχρονη ενημέρωση των τιμών των 2 απαγωγών, που εμφανίζονται στο κάτω μέρος της οθόνης. Πάνω στο τίτλο της οθόνης εμφανίζεται εάν πραγματοποιείται αποθήκευση της καταγραφής ηλεκτροκαρδιογραφήματος και σε ποια διεύθυνση είναι ενωμένο το κινητό τηλέφωνο.

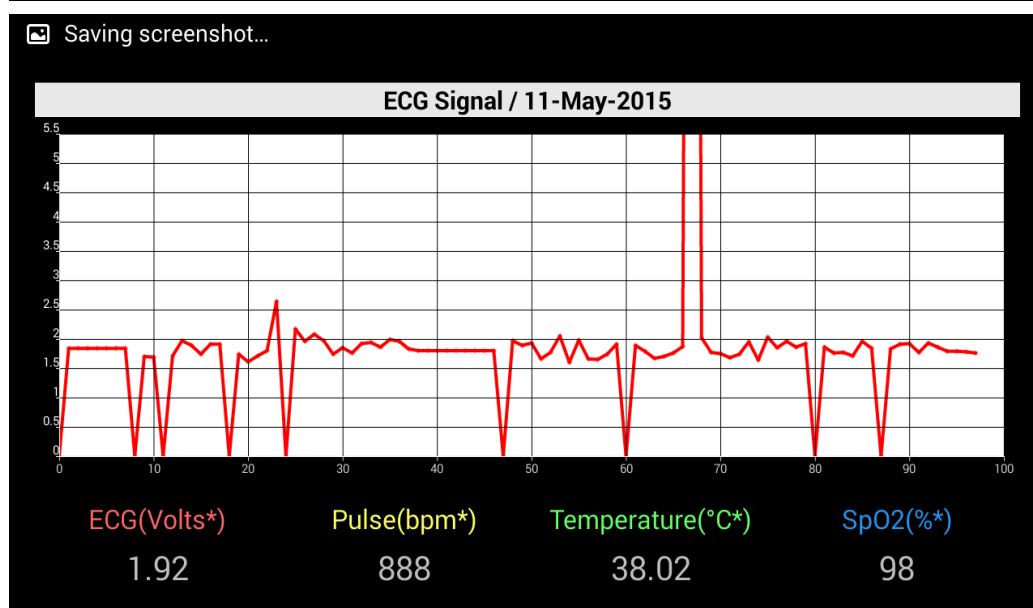
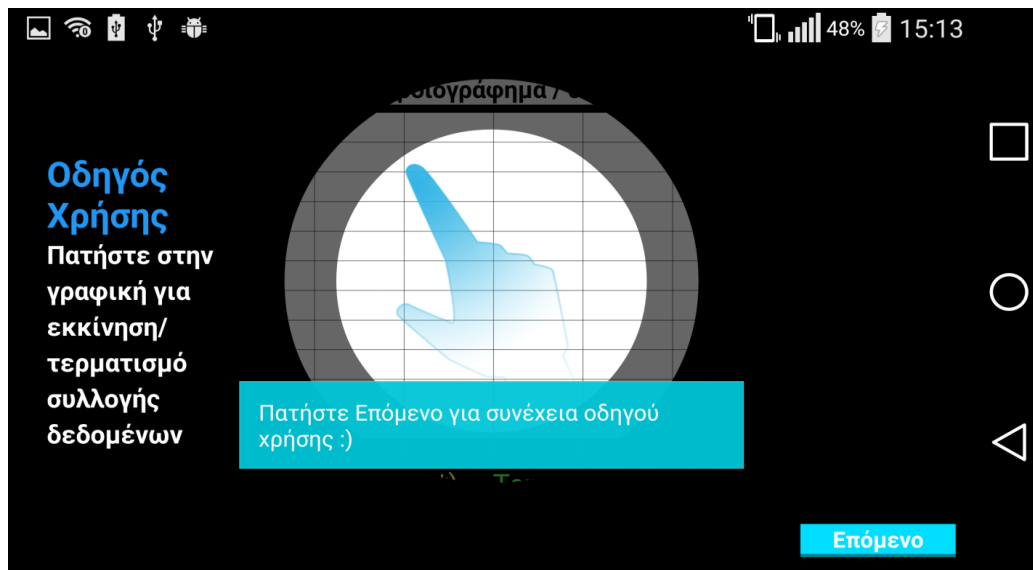
Με το πλήκτρο επιστροφής της κινητής συσκευής να ενεργοποιείται, τότε εμφανίζεται στην οθόνη μήνυμα που ενημερώνει τον χρήστη εάν επιθυμεί να απενεργοποίηση την σύνδεση Bluetooth, για εξοικονόμηση ενέργειας του κινητού.

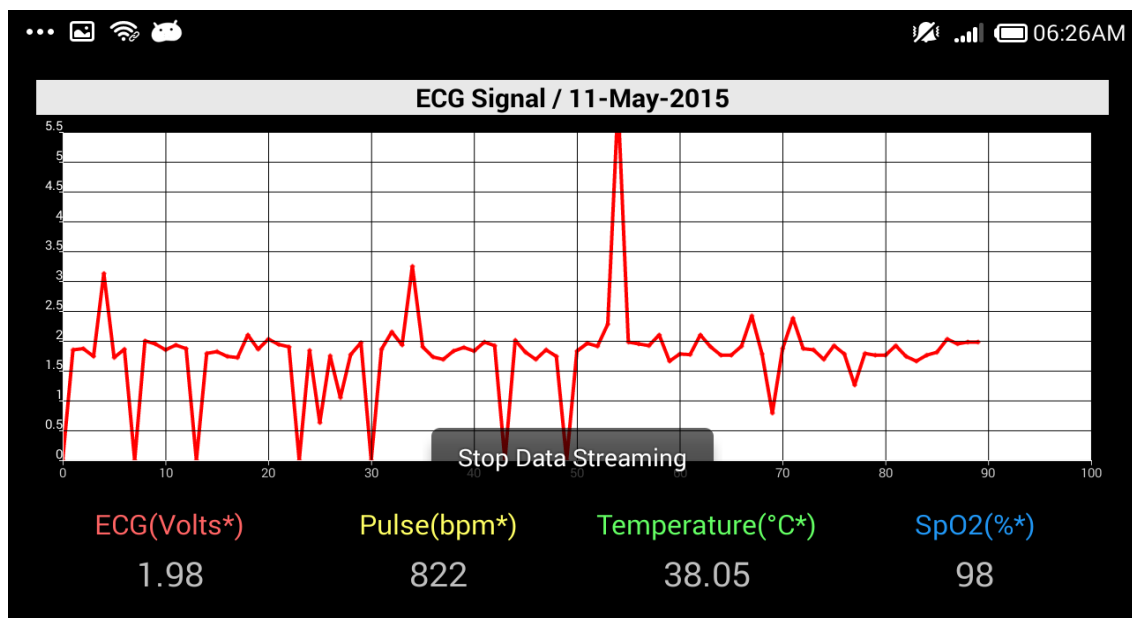
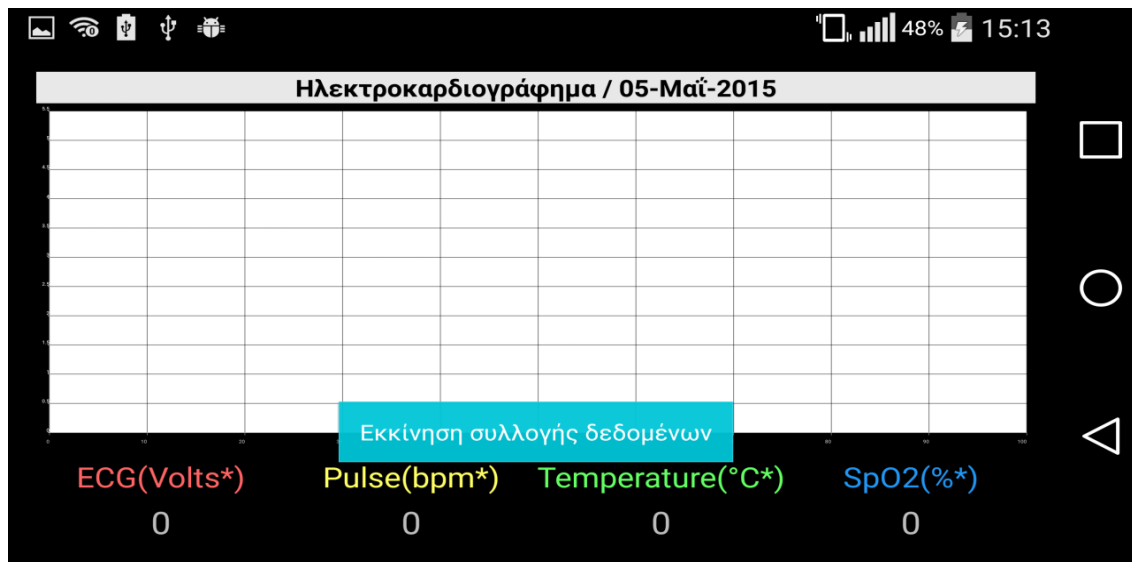
6.4.7 Ηλεκτροκαρδιογράφημα και βιομετρήσεις από πλατφόρμα Arduino



Μετά από μια άλλη επιτυχημένη σύνδεση, εμφανίζεται και πάλι ο πίνακας με τις επιλογές για Shimmer ή Arduino. Με την επιλογή για Arduino, τότε εμφανίζεται γραφικό παράθυρο για επιλογή απεικόνισης δεδομένων.

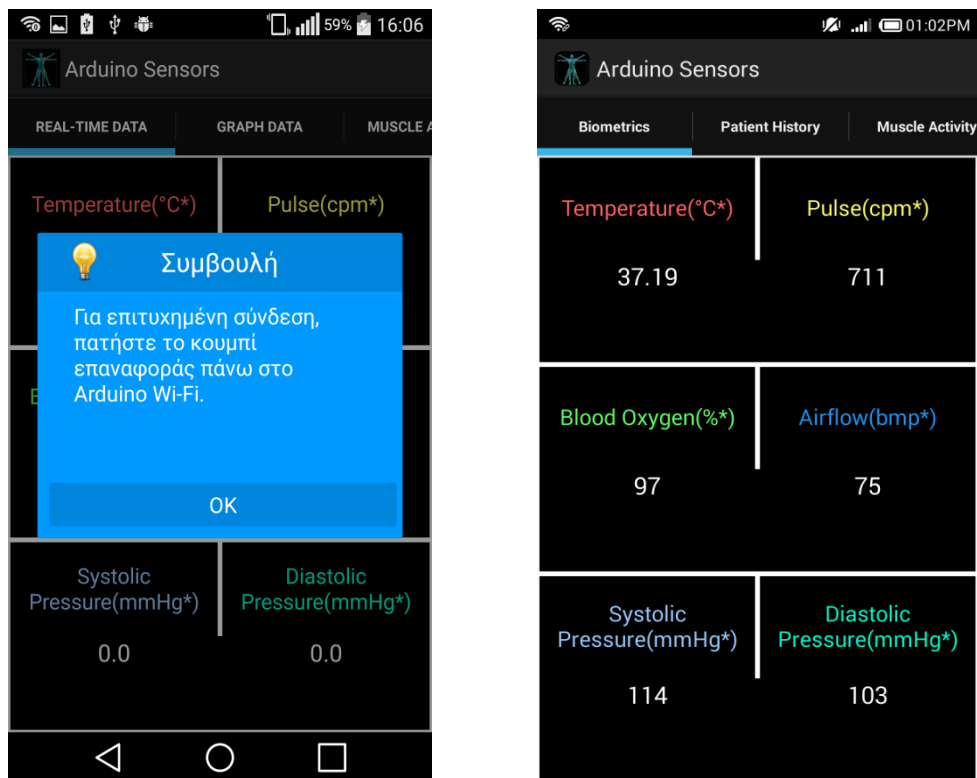






Ο χρήστης μπορεί, με το να πατήσει πάνω στην γραφική παράσταση του ηλεκτροκαρδιογραφήματος, να επιλέξει εάν θα εκκινήσει την συλλογή δεδομένων ή να την τερματίσει για όσο χρειάζεται.

6.4.8 Παρουσίαση βιομετρήσεων από πλατφόρμα Arduino



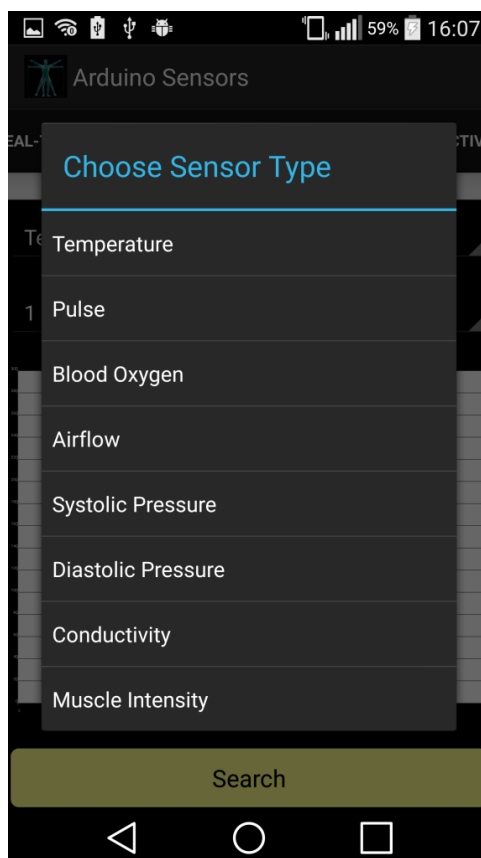
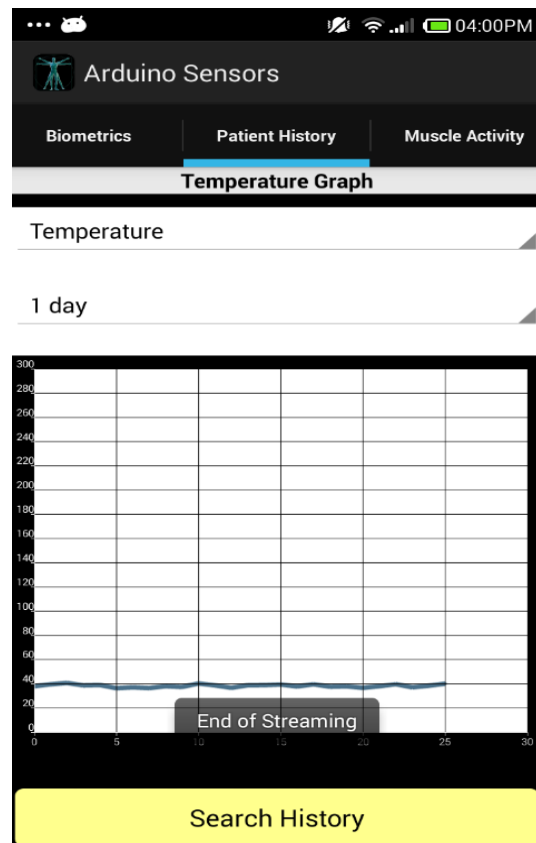
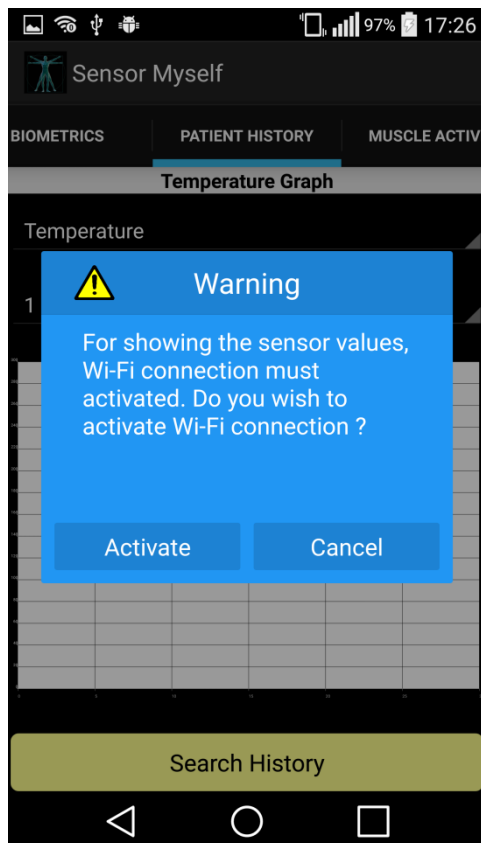
Με την εκκίνηση της δραστηριότητας για καταγραφή βιομετρήσεων, τότε εμφανίζεται κατάλληλο μήνυμα που ειδοποιά τον χρήστη πως ξεκίνησε η σύνδεση με Wi-Fi hotspot. Έπειτα, θα εμφανίζονται 6 διαφορετικές καρτέλες για την εξυπηρέτηση διαφορετικών σκοπών. Τα δεδομένα θα διαβάζονται από την πλατφόρμα Arduino και θα ενημερώνονται ταυτόχρονα και οι 6 καρτέλες.

➤ **Καρτέλα Real-Time Data:**

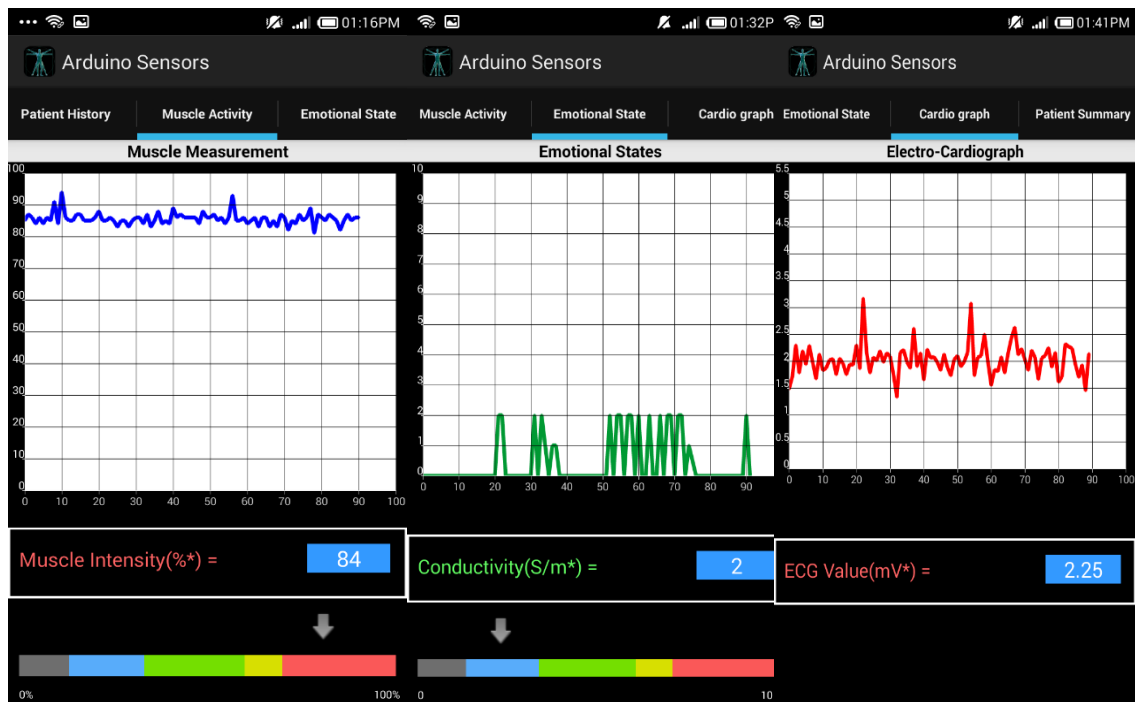
Εδώ εμφανίζονται και ενημερώνονται οι τιμές που έχουν να κάνουν με θερμοκρασία σώματος, καρδιακών παλμών, οξυγόνωσης αίματος, παροχής αέρα, συστολικής και διαστολικής πίεσης.

➤ **Καρτέλα Graph Data:**

Εδώ εμφανίζονται τα ιστορικά των μετρήσεων, όπως αυτά διαβάζονται από την βάση. Λόγω του ότι, χρειάζεται σύνδεση στο Wi-Fi για την ανάκτηση πληροφοριών, τότε με κατάλληλο μήνυμα θα ενημερώνεται ο χρήστης να ενεργοποιήσει την σύνδεση Wi-Fi.



Έπειτα επιλέγει τύπο δεδομένων αισθητήρα που επιθυμεί να παρακολουθήσει και πατώντας το κουμπί Search παρακολουθεί την αναπαράσταση των συγκεκριμένων δεδομένων. Σε περίπτωση που δεν υπάρχουν δεδομένα για την συγκεκριμένη χρονική στιγμή, τότε εμφανίζεται κατάλληλο μήνυμα ειδοποίησης.



➤ **Καρτέλα Muscle Activity:**

Εδώ θα παρουσιάζονται οι τιμές του ηλεκτρομυογραφήματος σε μια γραφική παράσταση, μαζί με μια ειδικά ζωγραφισμένη γραμμή ζωνών, που αναφέρεται σε πιο σημείο βρίσκεται η κατάσταση του μυ και θα υπάρχει συνεχής ενημέρωση με τις τρέχων τιμές.

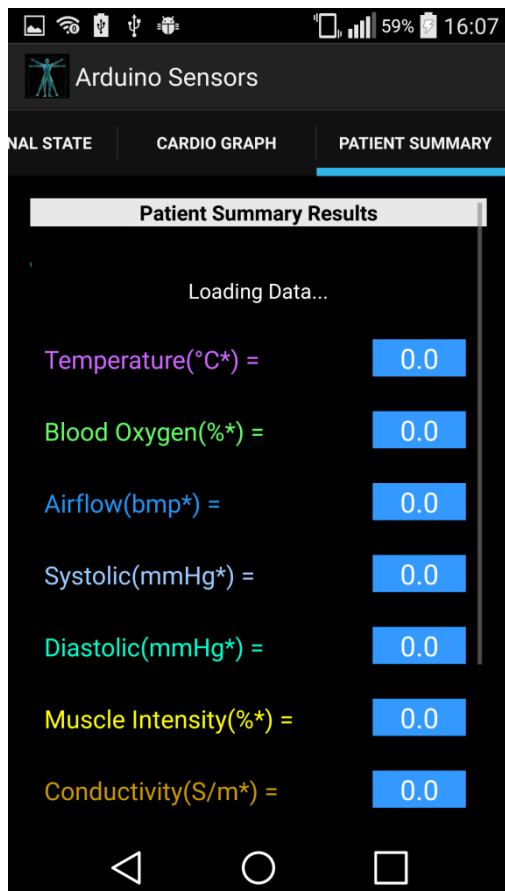
➤ **Καρτέλα Emotional State:**

Εδώ θα παρουσιάζονται οι τιμές της διαγωγιμότητας του δέρματος σε μια γραφική παράσταση, μαζί με μια ειδικά ζωγραφισμένη γραμμή ζωνών, που αναφέρεται σε πιο σημείο βρίσκεται η κατάσταση της διαγωγιμότητας, η οποία θα ενημερώνεται συνεχώς με τις τρέχων τιμές.

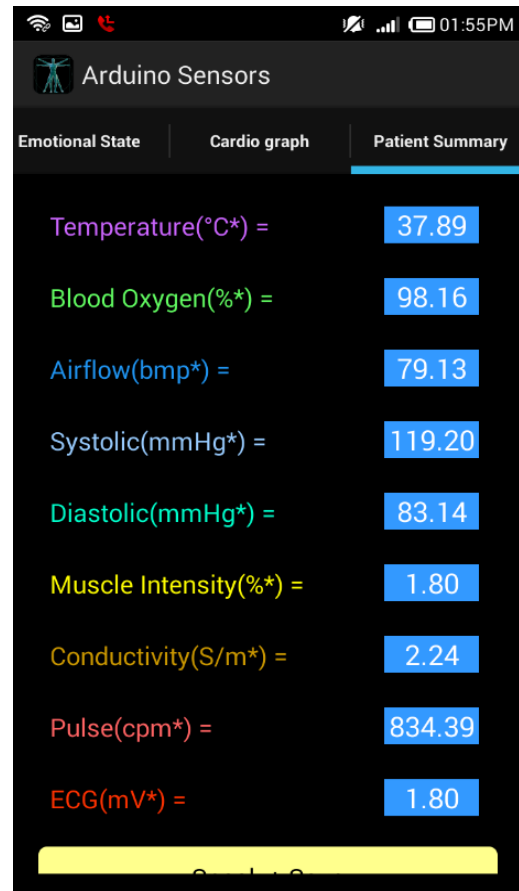
➤ **Καρτέλα Electro-Cardiograph:**

Εδώ θα παρουσιάζονται οι τιμές του ηλεκτροκαρδιογραφήματος σε μια γραφική παράσταση, μαζί με την τιμή που θα λαμβάνεται από την πλατφόρμα Arduino την δεδομένη στιγμή.

Αναμονή Συνάθροισης Δεδομένων

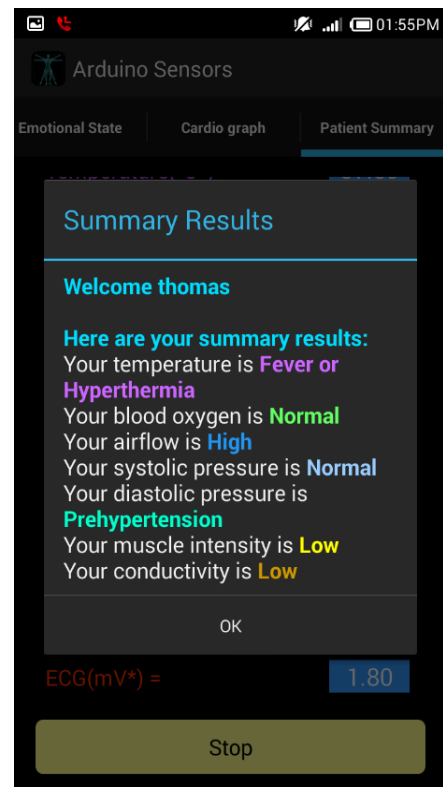


Αποτελέσματα Μέσων Όρων



➤ Καρτέλα Patient Summary Results:

Εδώ εμφανίζονται οι μεσοί όροι των τιμών των αισθητήρων που καταγράφονται. Μετά από ένα 1 λεπτό καταγραφής δεδομένων, θα πραγματοποιηθεί ενημέρωση των τιμών και θα μπορεί το σύστημα να πραγματοποιήσει την εκτέλεση των αποτελεσμάτων της περίληψης, φωνητικά και γραπτά.



6.5 Αξιολόγηση Συστήματος από χρήστες

Η εφαρμογή αξιολογήθηκε από το πανεπιστημιακό προσωπικό του Δρ. Παττίχη, καθώς και από συμφοιτητές μου. Η γενική απήχηση που είχε ήταν πάρα πολύ θετική, ειδικά με την χρήση γραφικών μέσων για προβολή μηνυμάτων και παραθύρων. Γενικά, η εφαρμογή πέτυχε τον στόχο της: να είναι εύχρηστη και να ενισχύει την υποκειμενική ικανοποίηση του χρήστη.

Τα δεδομένα που καταγράφονται τόσο στην κινητή συσκευή όσο και στην βάση δεδομένων, παρουσιάζονται με αρκετή ακρίβεια. Ολόκληρο το σύστημα φαίνεται να λειτουργεί χωρίς ιδιαίτερα προβλήματα και υπάρχει μια ομοιομορφία στις μεταβάσεις μεταξύ των δραστηριοτήτων.

Κεφάλαιο 7

Συμπεράσματα

7.1 Συμπεράσματα	100
7.2 Μελλοντικά Σχέδια	101

7.1 Συμπεράσματα

Ζούμε σε μια εποχή τεράστιων τεχνολογικών επιτεύξεων και ιδιαίτερα μεγάλη ανάπτυξη επιδέχθηκε ο τομέας της υγείας. Ο μέσος όρος προσδόκιμου ορίου ζωής βρίσκεται σε ολοένα και μεγαλύτερη κλίμακα αύξησης και γενικότερα η ποιότητα ζωής βελτιώθηκε κατά πολύ, αν αναλογιστεί κανείς το προσδόκιμο όριο ζωής πριν μερικά δεκάδες χρόνια. Η τεχνολογία χάρισε στην ανθρωπότητα πράγματα που ονειρευόμασταν μόνο ότι θα είχαμε και έδωσε τα θεμέλια για την συνεργασία και συναναστροφή ευφυών ανθρώπων, για την συγκρότηση ενός καλύτερου κόσμου. Ειδικότερα, παρατηρούμε πως και η πληροφορική ενσωματώθηκε σε κάθε τομέα της ανθρώπινης δημιουργικότητας και σκέψης. Παρατηρώντας την διπλωματική μου εργασία, βλέπουμε πως η πληροφορική κατάφερε να εισχωρήσει και στον τομέα της ανθρώπινης υγείας, και αν σκεφτεί κανείς πως η πληροφορική ξεκίνησε σαν ένα μαθηματικό μοντέλο αλγορίθμων και υπολογισμών, φτάνει σήμερα να σημαίνει κάτι περισσότερο στον άνθρωπο. Δεν θα είναι υπερβολή να εκφράσω την άποψη πως πολλά τεχνολογικά επιτεύγματα, πραγματοποιήθηκαν με την άνθηση της πληροφορικής.

Φτάνουμε τώρα στον γενικό σκοπό της διπλωματικής μου εργασίας. Σαν θέμα η διπλωματική μου είχε την ανάπτυξη μιας εφαρμογής στην πλατφόρμα Google Android για ψηφιακή ανάλυση ηλεκτροκαρδιογραφήματος, αλλά στην συνέχεια επεκτάθηκε με την καταγραφή επιπρόσθετων βιομετρήσεων από τις συσκευές Shimmer και Arduino. Η χρησιμοποίηση τόσο της συσκευής Shimmer όσο και της πλατφόρμας Arduino, μου πρόσφεραν επιπλέον τεχνολογική γνώση. Ιδιαίτερα, έμαθα να χειρίζομαι και κομμάτια υλικού (hardware), που δεν είχα την τύχη να μάθω κατά την διάρκεια των σπουδών μου. Ο συνδυασμός τόσο του υλικού (hardware), με τη χρήση αισθητήρων και διαφόρων οργάνων καταγραφής δεδομένων, όσο και του λογισμικού (software), για την επεξεργασία, αποθήκευση και παρουσίαση των δεδομένων υπήρξε μια πολύ καλή εμπειρία.

Ο ασθενής θα αντιμετωπίζει εξατομικευμένα κάποια προβλήματα υγείας που προκύπτουν, θα έχει την δυνατότητα να γνωρίζει κάποιο πρόβλημα υγείας που προκύπτει, χωρίς χρονοτριβές, και με άμεση ανταπόκριση από τον θεράποντα ιατρό. Σημαντικό κομμάτι είναι η χρήση των αισθητήρων για οικιακή χρήση, πράγμα που δείχνει την κατεύθυνση που θα έχει ο τομέας της ιατρικής περίθαλψης στο μέλλον.

7.2 Μελλοντικά Σχέδια

Βλέποντας μια ματιά στο μέλλον, παρατηρώ πως υπάρχουν πλήθος βελτιώσεων που μπορούν να πραγματοποιηθούν, ώστε η εφαρμογή να αντεπεξέλθει στον σύγχρονο κόσμο. Έχοντας στην διάθεση μου τις συσκευές Shimmer και Arduino, διαπίστωσα πως οι αισθητήρες του μέλλοντος θα πρέπει να λειτουργούν ασύρματα, διευκολύνοντας την μετακίνηση του ασθενούς, χωρίς να τον περιορίζουν συνεχώς. Επιπρόσθετα, η εφαρμογή θα μπορούσε να επεκταθεί και με την χρήση νευρωνικών δικτύων, ώστε να σύστημα να μαθαίνει από τις μετρήσεις του ασθενούς, καθιστώντας το σύστημα αυτόνομο και με μαθησιακή μνήμη.

Με την χρήση φωνητικής λειτουργιάς και εκμάθησης μπορεί να μετατραπεί σε σύστημα τεχνητής νοημοσύνης, όπου τα δεδομένα υγείας θα λαμβάνονται καθημερινά, χωρίς την μεσολάβηση του ασθενούς πάντα, και να υπάρχει συνεχής ανατροφοδότηση προς τον χρήστη. Επίσης, θα μπορούσε να υπάρχει άμεση επικοινωνία του ιατρού με τον ασθενή, ώστε τα σχόλια και οι ιατρικές συμβουλές να παρουσιάζονται αμέσως με λειτουργία εικονικής γνωστοποίησης (notification). Επιπρόσθετα, η χρήση τοπικής βάσης δεδομένων θα βοηθούσε στην μαζική προώθηση μέσω των δεδομένων στην βάση δεδομένων. Έτσι, δεν θα χρειάζεται το ανοιγοκλείσιμο της σύνδεσης Wi-Fi για την αποστολή και ανάκτηση δεδομένων.

Επιπρόσθετα, το σύστημα θα μπορούσε να είναι συμβατό και με περισσότερες εφαρμογές συλλογής και καταγραφής ηλεκτροκαρδιογραφήματος και βιομετρήσεων, καθιστώντας την καθολική χρήση και ευελιξία. Σημαντικό κομμάτι θα είναι εάν ο κάθε άνθρωπος διαθέτει σπίτι του, το δικό του ευφυή σύστημα ιατροφαρμακευτικής ανάδρασης.

Τέλος, το σύστημα θα ήθελα να ενσωματωθεί και υλοποιηθεί σύμφωνα με τα πρότυπα της πλατφόρμας Fi-Star, καθώς και να αναλυθεί σύμφωνα με το ECG Analysis (που υλοποιεί ο Κωνσταντίνου Ιωάννης, ΑΔΕ 2015), για πλήρης γενίκευση και χρησιμοποίησης όλων των μεμονωμένων λειτουργιών του συστήματος.

Βιβλιογραφία

- [1] Fi-Star Catalogue, 2015. [<http://catalogue.fi-star.eu/> Fi-star eHealthCare]
- [2] Fi-star eHealthCare, 2015. [<https://www.fi-star.eu/fi-star.html>]
- [3] Android Developers, 2015. [<http://developer.android.com/about/versions>]
- [4] Fi-Ware, 12 May 2015. [<http://forge.fiware.org/plugins/mediawiki/wiki/fiware>]
- [5] Eu Medline, 2010 [<http://www.eumedline.eu>]
- [6] Fi-Ware Catalogue, 2015.[<http://catalogue.fiware.org/enablers/publishsubscribe-context-broker-orion-context-broker>]
- [7] Android, 2015 [<http://www.android.com/history>]
- [8] Wikipedia, 24 May 2015 [http://en.wikipedia.org/wiki/Android_version_history]
- [9] TutorialsPoint, Android Architecture, 2014
[http://www.tutorialspoint.com/android/android_architecture.htm]
- [10] Google Store, 2015. [<https://play.google.com/store/apps?hl=el>]
- [11] Android Developers Widgets, 2015.
[<https://developer.android.com/guide/topics/appwidgets/index.html>]
- [12] Android, Android Services, 2015.
[<http://developer.android.com/guide/components/services.html>]
- [13] Android Developers, Android Training, 2015.
[<https://developer.android.com/training/index.html>]

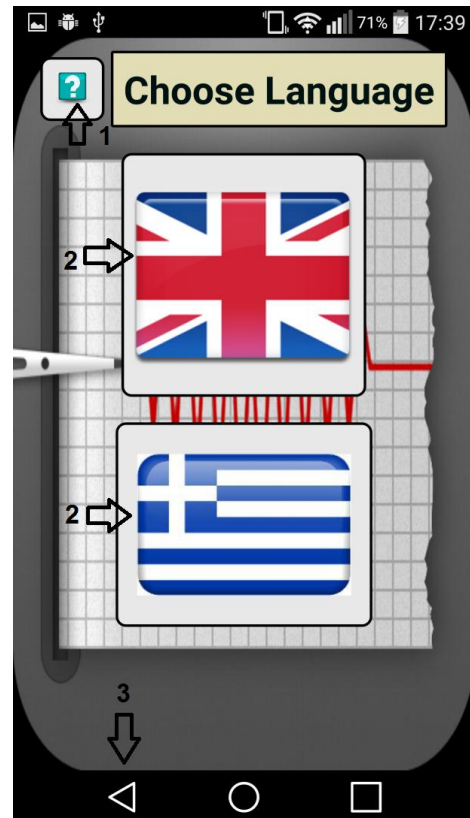
- [14] MySQL Workbench, 2015 [<https://www.mysql.com/products/workbench>]
- [15] Vogella, 20 August 2014.
[<http://www.vogella.com/tutorials/ApacheTomcat/article.html>]
- [16] Cooking Hacks, July 2014.
[<http://www.cooking-hacks.com/documentation/tutorials/ehealth-biometric-sensor-platform-arduino-raspberry-pi-medical>]
- [17] Shimmer, 2015. [<http://www.shimmersensing.com>]
- [18] Arduino, 2015. [<http://www.arduino.cc/en/Main/ArduinoBoardUno>]
- [19] σημειώσεις μαθήματος ΕΠΛ241 Μάθημα Πανεπιστημίου Κύπρου: Ανάλυση και σχεδίαση συστημάτων
- [20] σημειώσεις μαθήματος ΕΠΛ361 Μάθημα Πανεπιστημίου Κύπρου: Τεχνολογία Λογισμικού
- [21] σημειώσεις μαθήματος ΕΠΛ435 Μάθημα Πανεπιστημίου Κύπρου: Αλληλεπίδραση Ανθρώπου - Υπολογιστή
- [22] σημειώσεις μαθήματος ΗΜΜΥ370 Μάθημα Πανεπιστημίου Κύπρου: Βιοϊατρική Μηχανική

Παράρτημα Α

Εγχειρίδιο χρήσης ασθενή και ιατρού

➤ Εγχειρίδιο χρήσης ασθενή

ΕΝΑΡΞΗ ΕΦΑΡΜΟΓΗΣ ΚΑΙ ΕΠΙΛΟΓΗ ΓΛΩΣΣΑΣ

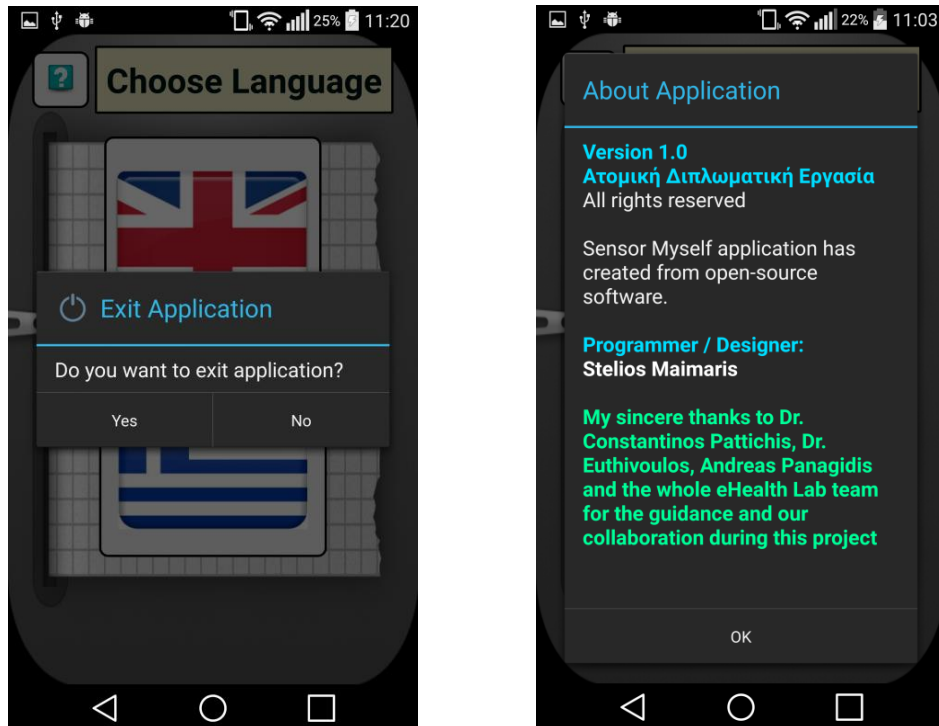


Με την έναρξη της εφαρμογής εμφανίζεται η αρχική εικόνα, που παρουσιάζει το ιερό φίδι του Ασκληπίου, μαζί με την ιατρική ατάκα “Your health, Our Mission”. Η εμφάνιση της αρχικής εικόνας διαρκεί μερικά δευτερόλεπτα μόνο.

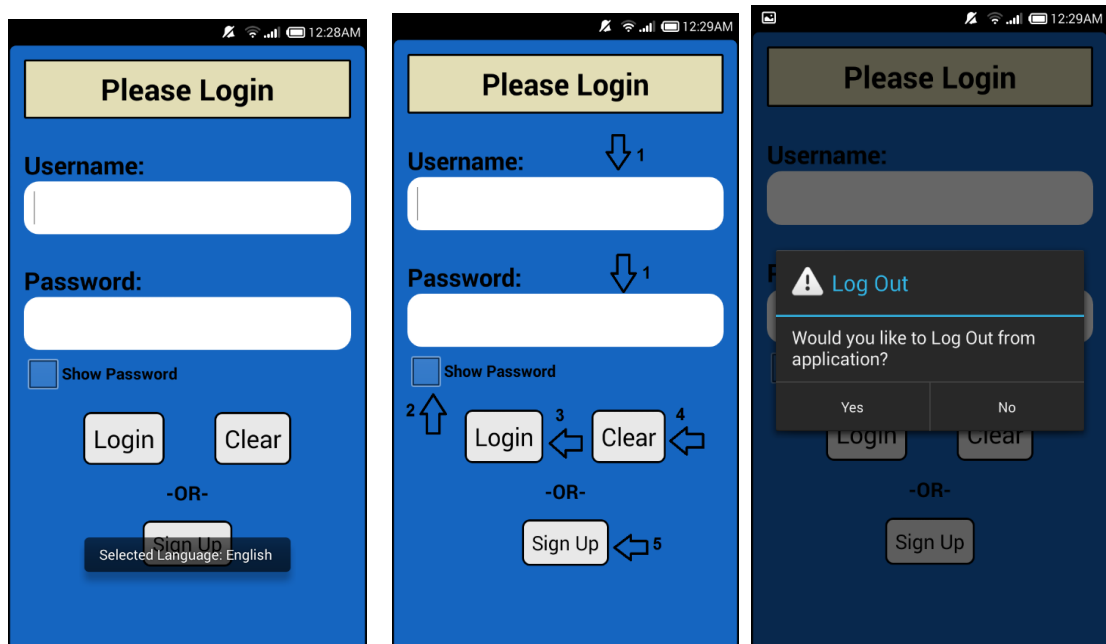
Έπειτα, παρουσιάζεται η εικόνα που έχουμε πάνω δεξιά, στην οποία έχουμε τις παρακάτω επιλογές:

1. **ΕΜΦΑΝΙΣΗ ΠΛΗΡΟΦΟΡΙΩΝ ΕΦΑΡΜΟΓΗΣ:** Πατώντας το συγκεκριμένο κουμπί εμφανίζονται οι ευχαριστίες, το όνομα του προγραμματιστή και της εφαρμογής, όπως φαίνεται στην εικόνα κάτω δεξιά.
2. **ΕΠΙΛΟΓΗ ΓΛΩΣΣΑΣ:** Πατώντας σε μια από τις σημαίες μπορούμε να επιλέξουμε την ανάλογη γλώσσα, η οποία θα μας συνδέσει στο σύστημα. Οι επιλογές γλώσσας είναι Αγγλικά και Ελληνικά.

3. **ΕΠΙΛΟΓΗ ΕΞΟΔΟΥ:** Πατώντας στο κουμπί αυτό εμφανίζεται το κατάλληλο μήνυμα που ενημερώνει τον χρήστη, αν επιθυμεί να εξέλθει από την εφαρμογή, όπως βλέπουμε και στην εικόνα κάτω αριστερά.



ΕΠΙΛΟΓΕΣ ΧΡΗΣΤΗ ΚΑΙ ΕΙΣΟΔΟΣ ΣΤΟ ΣΥΣΤΗΜΑ

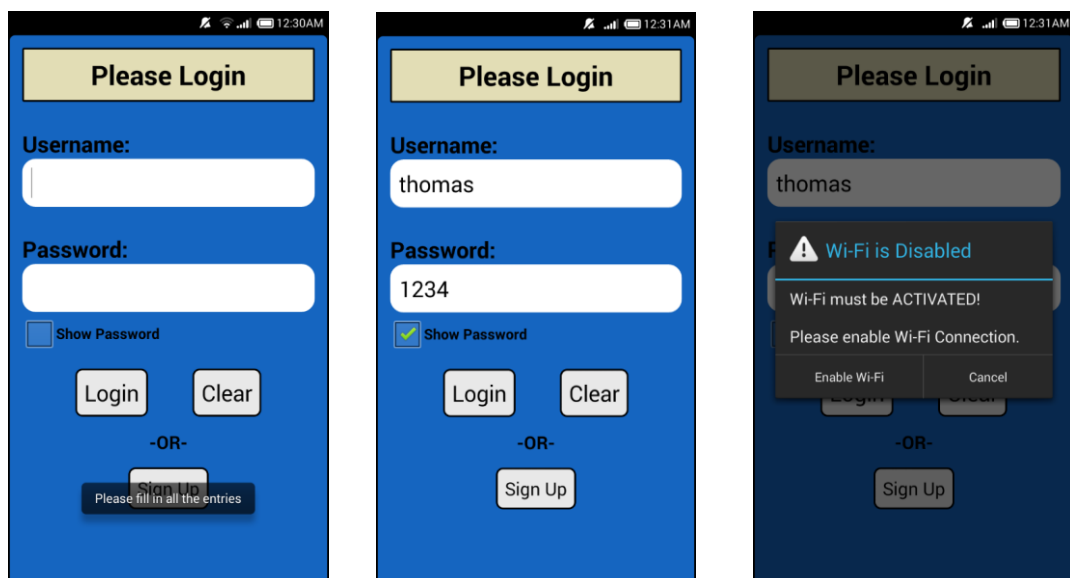


Εφόσον έχουμε επιλέξει μια γλώσσα, στην περίπτωση μας τα αγγλικά, εμφανίζεται η οθόνη που βλέπουμε στην εικόνα πάνω αριστερά, η οποία μας εμφανίζει και ένα μήνυμα που εμφανίζει τη γλώσσα που έχουμε επιλέξει, το οποίο διαρκεί λίγα δευτερόλεπτα. Ακολουθώντας, έχουμε τις επιλογές που φαίνονται στην εικόνα πάνω στο κέντρο:

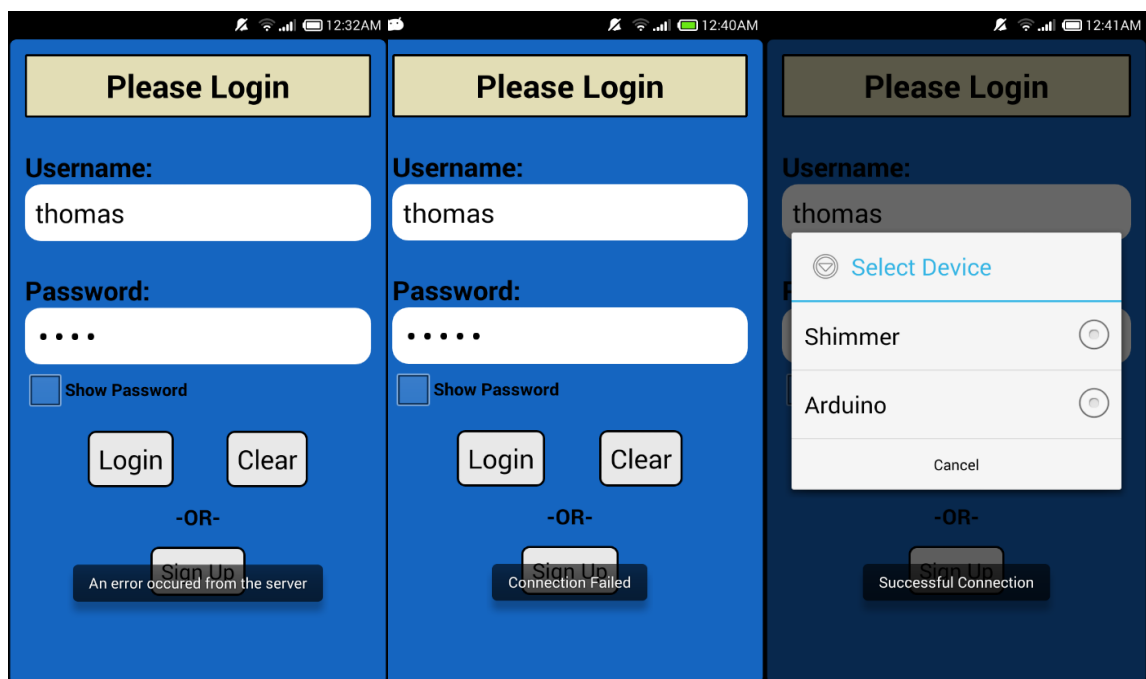
1. **ΕΙΣΑΓΩΓΗ USERNAME ΚΑΙ PASSWORD:** Στα 2 συγκεκριμένα πεδία εισάγουμε το username και το password που έχουμε επιλέξει, αφού έχουμε πρώτα κάνει εγγραφή με την επιλογή 5, το Sign Up.
2. **ΕΠΙΛΟΓΗ ΕΜΦΑΝΙΣΗΣ PASSWORD:** Πατώντας σε αυτό το check box, εμφανίζεται ο κρυμμένος κωδικός πρόσβασης.
3. **ΕΠΙΛΟΓΗ ΕΙΣΟΔΟΥ ΣΤΟ ΣΥΣΤΗΜΑ:** Πατώντας στο κουμπί αυτό πραγματοποιούμε είσοδο στην εφαρμογή, εφόσον τα πεδία που έχουμε εισάγει στην επιλογή 1 δεν είναι κενά.
4. **ΕΠΙΛΟΓΗ ΚΑΘΑΡΙΣΜΟΥ ΠΕΔΙΩΝ:** Πατώντας στο κουμπί αυτό γίνεται εκκαθάριση των στοιχείων που εισαγάγαμε με την επιλογή 1.
5. **ΕΠΙΛΟΓΗ ΔΗΜΙΟΥΡΓΙΑΣ ΛΟΓΑΡΙΑΣΜΟΥ:** Πατώντας σε αυτό το κουμπί μπορούμε να δημιουργήσουμε λογαριασμό ως ασθενής για να μπορούμε να χρησιμοποιούμε την εφαρμογή.

Να σημειωθεί ότι πατώντας το πλήκτρο επιστροφής της κινητής συσκευής εμφανίζεται η οθόνη που βλέπουμε πάνω δεξιά, με την οποία μπορούμε να εξέλθουμε από την εφαρμογή.

ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ



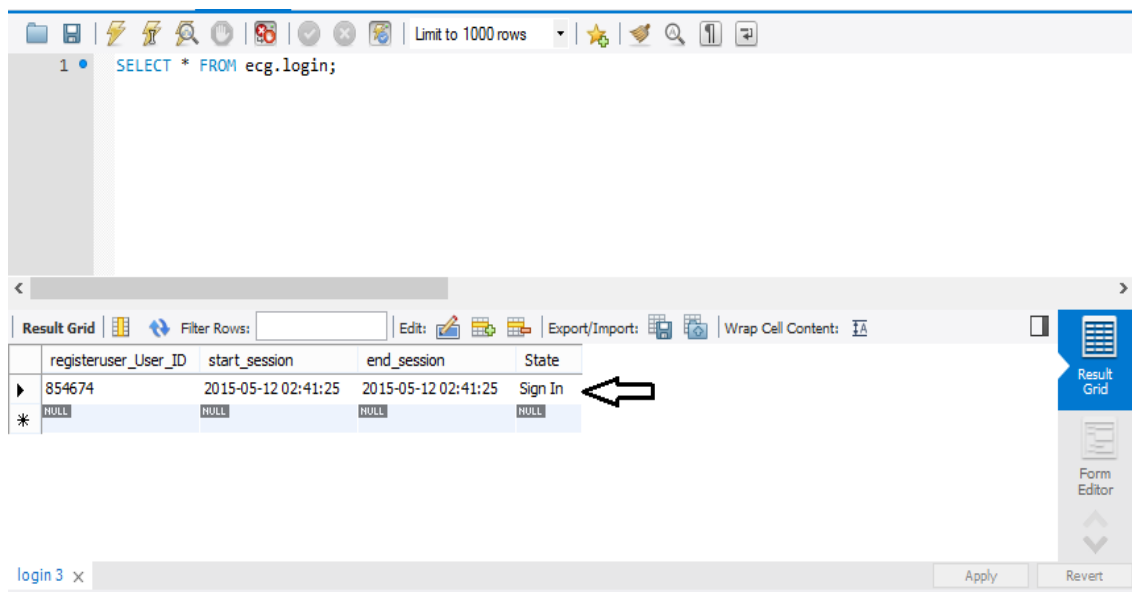
- Αν πατήσουμε στο πλήκτρο Login χωρίς να εισάγουμε και Username και Password, τότε εμφανίζεται μήνυμα για να συμπληρώσουμε όλα τα πεδία προτού επιλέξουμε να συνδεθούμε, όπως φαίνεται στην εικόνα πάνω αριστερά.
- Αφού προσθέσαμε τα στοιχεία μας στα πεδία, έχουμε την επιλογή για εμφάνιση κωδικού όπως φαίνεται στην εικόνα πάνω στο κέντρο.
- Αν δεν είμαστε συνδεδεμένοι με το δίκτυο Wi-Fi της κινητής συσκευής, εμφανίζεται το μήνυμα που βλέπουμε στην εικόνα πάνω δεξιά, που μας δίνει τη δυνατότητα να ενεργοποιήσουμε τα Wi-Fi. Το μήνυμα αυτό εμφανίζεται όταν πατήσουμε στο κουμπί Login.



- Για την επικοινωνία της κινητής μας συσκευής με τη βάση δεδομένων που χρησιμοποιήσαμε, τη MySQL, απαιτείται η εγκατάσταση ενός server που θα επικοινωνεί με τη βάση δεδομένων. Εμείς χρησιμοποιήσαμε τον Apache Tomcat 7, τον οποίο ενσωματώσαμε και στο Eclipse IDE. Αν κατά την προσπάθεια μας να ενωθούμε με τη βάση δεδομένων για την επαλήθευση των στοιχείων που εισαγάγαμε, δεν έχουμε ξεκινήσει τον server, τότε εμφανίζεται το μήνυμα που βλέπουμε στην εικόνα πάνω αριστερά.
- Αν κατά την διαδικασία εξακρίβωσης της ορθότητας των στοιχείων μας εισαγάγαμε λάθος στοιχεία στο username και password, τότε εμφανίζεται το

μήνυμα που βλέπουμε στην εικόνα πάνω στο κέντρο, που μας ενημερώνει για αποτυχημένη σύνδεση.

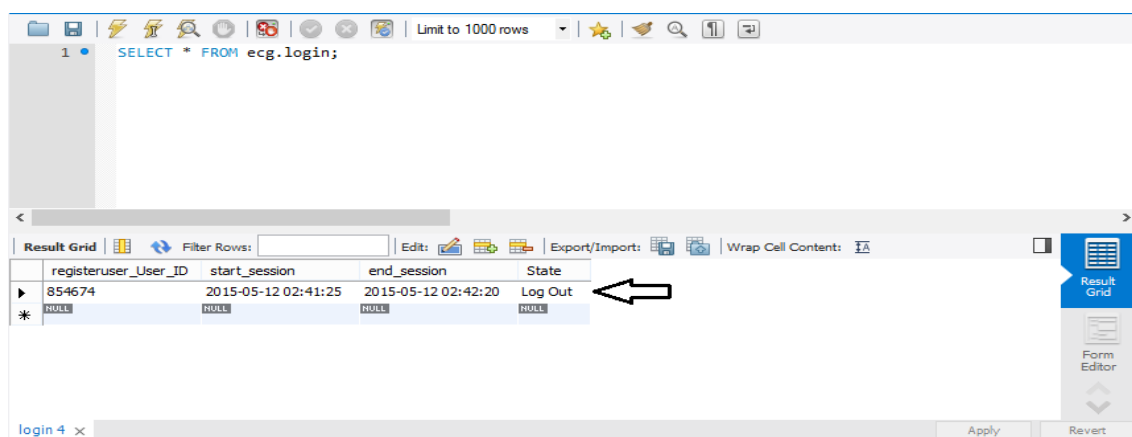
- Αν καταφέραμε να εισάγουμε σωστά τα στοιχεία μας όπως τα εισαγάγαμε στην βάση δεδομένων, εμφανίζεται η οθόνη που φαίνεται στην εικόνα πάνω δεξιά, η οποία μας ενημερώνει ότι καταφέραμε επιτυχημένη σύνδεση με τη βάση δεδομένων και μας δίνει την επιλογή να διαλέξουμε μια από τις 2 επιλογές καταγραφής ηλεκτροκαρδιογραφήματος, Shimmer ή Arduino. Αυτόματα καταγράφεται και στη βάση δεδομένων η ταυτότητα του ατόμου που έκανε Login, με την ημερομηνία και την ώρα να αποθηκεύονται μαζί και την κατάσταση να εμφανίζεται ως Sign In όπως φαίνεται πιο κάτω:



The screenshot shows a database application interface. At the top, there is a toolbar with various icons and a text input field containing the SQL query: `SELECT * FROM ecg.login;`. Below the toolbar, there is a table with the following columns: `registeruser_User_ID`, `start_session`, `end_session`, and `State`. The table contains two rows. The first row has the values: 854674, 2015-05-12 02:41:25, 2015-05-12 02:41:25, and Sign In. The second row has the values: NULL, NULL, NULL, and NULL. A black arrow points to the 'Sign In' text in the first row. On the right side of the table, there is a vertical toolbar with buttons for 'Result Grid', 'Form Editor', and 'Apply/Revert'.

registeruser_User_ID	start_session	end_session	State
854674	2015-05-12 02:41:25	2015-05-12 02:41:25	Sign In
NULL	NULL	NULL	NULL

- Πατώντας το πλήκτρο επιστροφής της κινητής συσκευής εμφανίζεται η οθόνη που είδαμε πιο πάνω, με την οποία μπορούμε να κάνουμε Log Out από την εφαρμογή. Αν επιλέξουμε να αποσυνδεθούμε από την εφαρμογή, τότε αυτόματα



The screenshot shows the same database application interface as the previous one. The SQL query in the text input field is still `SELECT * FROM ecg.login;`. The table now shows a different record. The first row has the values: 854674, 2015-05-12 02:41:25, 2015-05-12 02:42:20, and Log Out. The second row has the values: NULL, NULL, NULL, and NULL. A black arrow points to the 'Log Out' text in the first row. The right side of the interface remains the same.

registeruser_User_ID	start_session	end_session	State
854674	2015-05-12 02:41:25	2015-05-12 02:42:20	Log Out
NULL	NULL	NULL	NULL

αλλάζει η κατάσταση στη βάση δεδομένων και από Sign In γίνεται Log Out, αλλά και το end_session παίρνει την ανανεωμένη ώρα αποσύνδεσης σύμφωνα με την οποία κάναμε Log Out, όπως βλέπουμε στην εικόνα πιο πάνω:

- Για μερικά δευτερόλεπτα, παρουσιάζεται στο κάτω μέρος της οθόνης το μήνυμα Successful Log Out, για να ενημερώσει τον χρήστη ότι η ενέργεια του για αποσύνδεση από την εφαρμογή ήταν επιτυχής.

ΔΥΝΑΤΟΤΗΤΑ ΔΗΜΙΟΥΡΓΙΑΣ ΛΟΓΑΡΙΑΣΜΟΥ ΑΣΘΕΝΗ

Sign Up

ID: 1

Name: 1

Surname: 1

Gender: 1

Date of Birth:

Date of Birth: 1

Street: 1

Number of Street: 1

City: 1

Country: 1

Postal Code: 1

Telephone Number: 1

Email: 1

Username: 1

Password: 1

Show Password 2

1. ΕΙΣΑΓΩΓΗ ΠΡΟΣΩΠΙΚΩΝ ΣΤΟΙΧΕΙΩΝ:

Στα συγκεκριμένα πεδία εισάγουμε τα προσωπικά στοιχεία του ασθενή, επιλέγοντας στο τέλος ένα μοναδικό username με ένα password που θα χρησιμοποιεί ο ασθενής κάθε φορά που κάνει Login στην εφαρμογή.

2. ΕΠΙΛΟΓΗ ΕΜΦΑΝΙΣΗΣ PASSWORD:

Πατώντας σε αυτό το check box, εμφανίζεται ο κωδικός που έχουμε πληκτρολογήσει από πριν.

Telephone Number: 1

Email: 1

Username: 1

Password: 1

Show Password 2

Please fill in all the entries

3 Register 4 Clear

3. ΕΠΙΛΟΓΗ ΚΑΤΑΓΡΑΦΗΣ ΣΤΟΙΧΕΙΩΝ: Πατώντας στο κουμπί αυτό πραγματοποιούμε καταγραφή των στοιχείων στη βάση δεδομένων.

4. ΕΠΙΛΟΓΗ ΚΑΘΑΡΙΣΜΟΥ ΠΕΔΙΩΝ: Πατώντας στο κουμπί αυτό γίνεται εκκαθάριση των στοιχείων που εισαγάγαμε με την επιλογή 1.

Να σημειωθεί ότι αν δεν εισαχθούν στοιχεία σε κάποιο πεδίο, εμφανίζεται το μήνυμα που μας προτρέπει να συμπληρώσουμε όλα τα πεδία.

ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ

The screenshot displays a mobile application interface with a blue background and white text. At the top, there is a yellow button labeled 'Sign Up'. Below it, the form is organized into three columns. The first column contains fields for ID (854674), Name (Thomas), Surname (Jonathan), Gender (Male), and Date of Birth. The second column contains fields for Date of Birth (1985-07-04), Street (Nelson), Number of Street (15), City (New York), and Country (United States). The third column contains fields for Postal Code (2526), Telephone Number (99123456), Email (thomas@hotmail.com), Username (thomas), and Password (1234). A 'Show Password' checkbox is located at the bottom right of the form.

Field	Value
ID	854674
Name	Thomas
Surname	Jonathan
Gender	Male
Date of Birth	1985-07-04
Street	Nelson
Number of Street	15
City	New York
Country	United States
Postal Code	2526
Telephone Number	99123456
Email	thomas@hotmail.com
Username	thomas
Password	1234

Αφού έχουμε εισάγει τα στοιχεία του ασθενή όπως φαίνεται και πιο πάνω, πατούμε στο κουμπί register και στέλνονται τα στοιχεία στη βάση δεδομένων. Όπως βλέπουμε πιο κάτω, τα στοιχεία εισάχθηκαν κανονικά στη βάση δεδομένων, και την επόμενη φορά που ο χρήστης θελήσει να χρησιμοποιήσει την εφαρμογή, θα μπορεί να το πράξει εισάγοντας το username και password που έχει επιλέξει.

1 • `SELECT * FROM ecg.registeruser;`

Result Grid

	User_ID	Name	Surname	Gender	Date_of_Birth	Street	Number_of_Street	City	Country	Postal_Code	Telephone_Number
▶	785463	Jennifer	Robinson	Female	1981-06-25	Tiffany's	23	California	USA	3537	99654321
	854674	Thomas	Jonathan	Male	1985-07-04	Nelson	15	New York	USA	2526	99123456
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

registeruser 1 x Apply

Τέλος, να αναφέρουμε ότι κατά τη διαδικασία δημιουργίας λογαριασμού, αυτοματοποιήσαμε 3 πεδία για την ευκολότερη εισαγωγή στοιχείων από τον χρήστη.

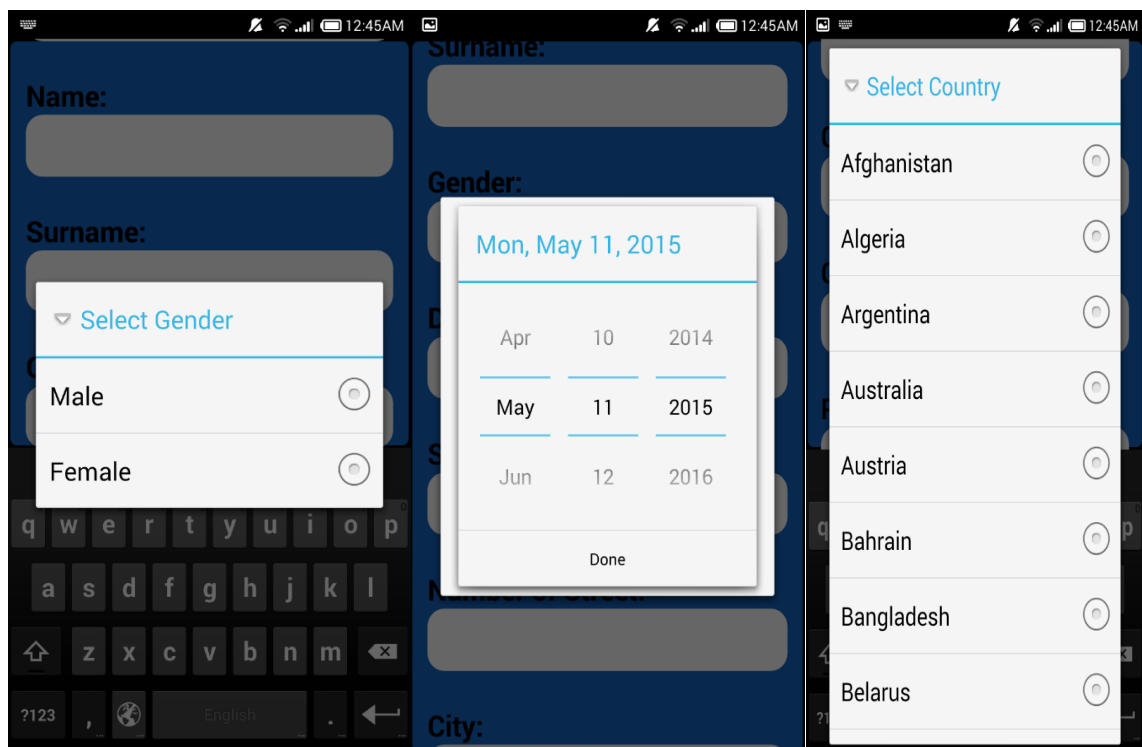
1 • `SELECT * FROM ecg.registeruser;`

Result Grid

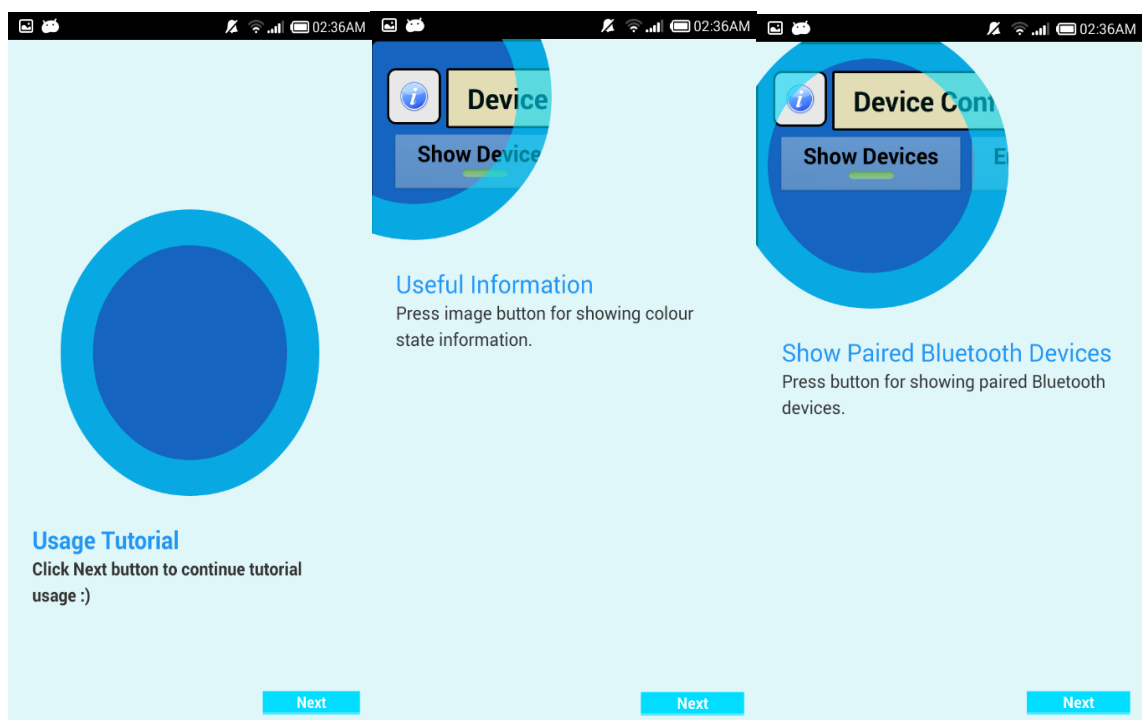
	Date_of_Birth	Street	Number_of_Street	City	Country	Postal_Code	Telephone_Number	Email	Username	UserPassword
▶	981-06-25	Tiffany's	23	California	USA	3537	99654321	jennifer@gmail.com	jennifer	1234
	985-07-04	Nelson	15	New York	USA	2526	99123456	thomas@hotmail.com	thomas	1234
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

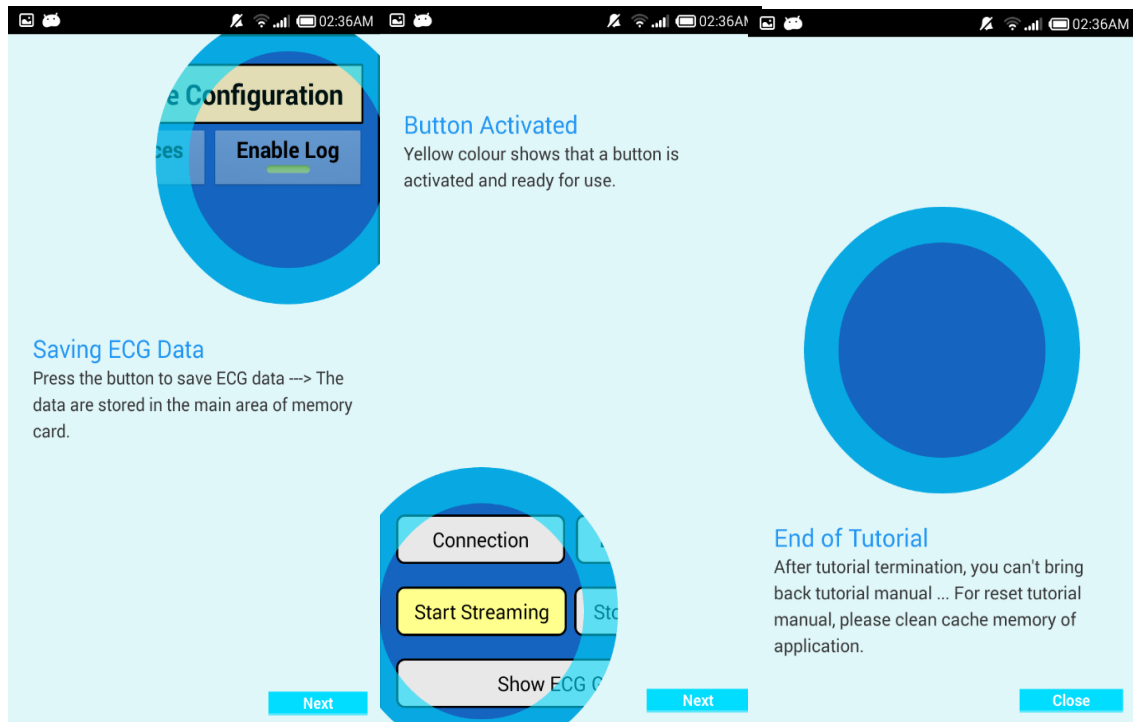
registeruser 1 x Apply Revert

Αυτά τα πεδία είναι το Gender, το Date of Birth και το Country, όπως φαίνονται και στις οθόνες πιο κάτω:



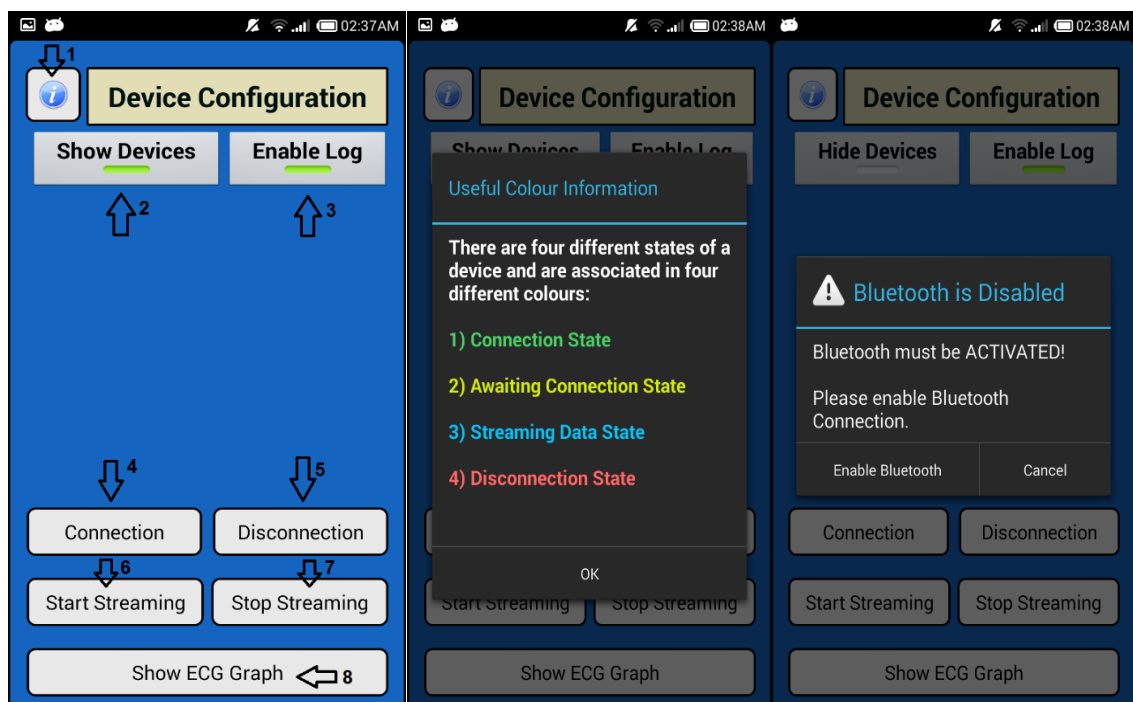
ΗΛΕΚΤΡΟΚΑΡΔΙΟΓΡΑΦΗΜΑ ΑΠΟ ΣΥΣΚΕΥΗ SHIMMER





Με την έναρξη του τμήματος της συσκευής Shimmer, εμφανίζεται ένας σύντομος οδηγός χρήσης, με οδηγίες είτε στα ελληνικά είτε στα αγγλικά (εξαρτάται από την γλωσσά επιλογής στην αρχή της εφαρμογής). Πιο κάτω θα αναφέρουμε πιο συγκεκριμένα τι ακριβώς κάνει το κάθε κουμπί στην αρχική οθόνη της συσκευής Shimmer, καθώς και τη διαδικασία καταγραφής του ηλεκτροκαρδιογραφήματος.

Με τον τερματισμό του οδηγού χρήσεως εμφανίζεται η αρχική οθόνη της συσκευής



Shimmer. Παρατηρούμε 8 κουμπιά, τις λειτουργίες των οποίων θα αναφέρουμε πιο κάτω:

1. ΠΛΗΡΟΦΟΡΙΕΣ ΚΑΤΑΣΤΑΣΗΣ SHIMMER:

Οι πληροφορίες που εμφανίζονται πατώντας αυτό το κουμπί, φαίνονται στην εικόνα πιο πάνω στο κέντρο. Συγκεκριμένα, μια Shimmer συσκευή μπορεί να βρίσκεται σε 4 καταστάσεις, κάθε μια από τις οποίες αναγνωρίζεται από ένα μοναδικό χρώμα. Με πράσινο η συσκευή είναι συνδεδεμένη με την κινητή συσκευή, με κίτρινο χρώμα αναμένουμε σύνδεση της συσκευής Shimmer, με μπλε χρώμα υποδεικνύεται η μεταφορά δεδομένων από τη συσκευή Shimmer στη κινητή συσκευή μας και με κόκκινο χρώμα η συσκευή Shimmer αποσυνδέθηκε.

2. ΕΜΦΑΝΙΣΗ PAIRED ΣΥΣΚΕΥΩΝ SHIMMER:

Πατώντας σε αυτό το κουμπί, εμφανίζονται οι συσκευές Shimmer με τις οποίες έχουμε, από προηγουμένως, κάνει pair με την κινητή συσκευή, χρησιμοποιώντας την τεχνολογία Bluetooth.

3. ΕΠΙΛΟΓΗ ΚΑΤΑΓΡΑΦΗΣ ΗΛΕΚΤΡΟΚΑΡΔΙΟΓΡΑΦΗΜΑΤΟΣ:

Πατώντας στο κουμπί αυτό πραγματοποιούμε αποθήκευση των τιμών του ηλεκτροκαρδιογραφήματος τοπικά στην κινητή μας συσκευή.

4. ΣΥΝΔΕΣΗ ΣΥΣΚΕΥΗΣ SHIMMER:

Πατώντας στο κουμπί αυτό γίνεται σύνδεση της συσκευής Shimmer με την κινητή μας συσκευή. Η επιτυχής σύνδεση εμφανίζεται με πράσινο χρώμα, ενώ η αναμενόμενη σύνδεση με κίτρινο χρώμα.

5. ΑΠΟΣΥΝΔΕΣΗ ΣΥΣΚΕΥΗΣ SHIMMER:

Πατώντας στο κουμπί αυτό γίνεται αποσύνδεση της συσκευής Shimmer με την κινητή μας συσκευή, η οποία εμφανίζεται για μερικά δευτερόλεπτα με κόκκινο χρώμα.

6. ΕΝΑΡΞΗ ΣΥΛΛΟΓΗΣ ΔΕΔΟΜΕΝΩΝ ΚΑΡΔΙΟΓΡΑΦΗΜΑΤΟΣ:

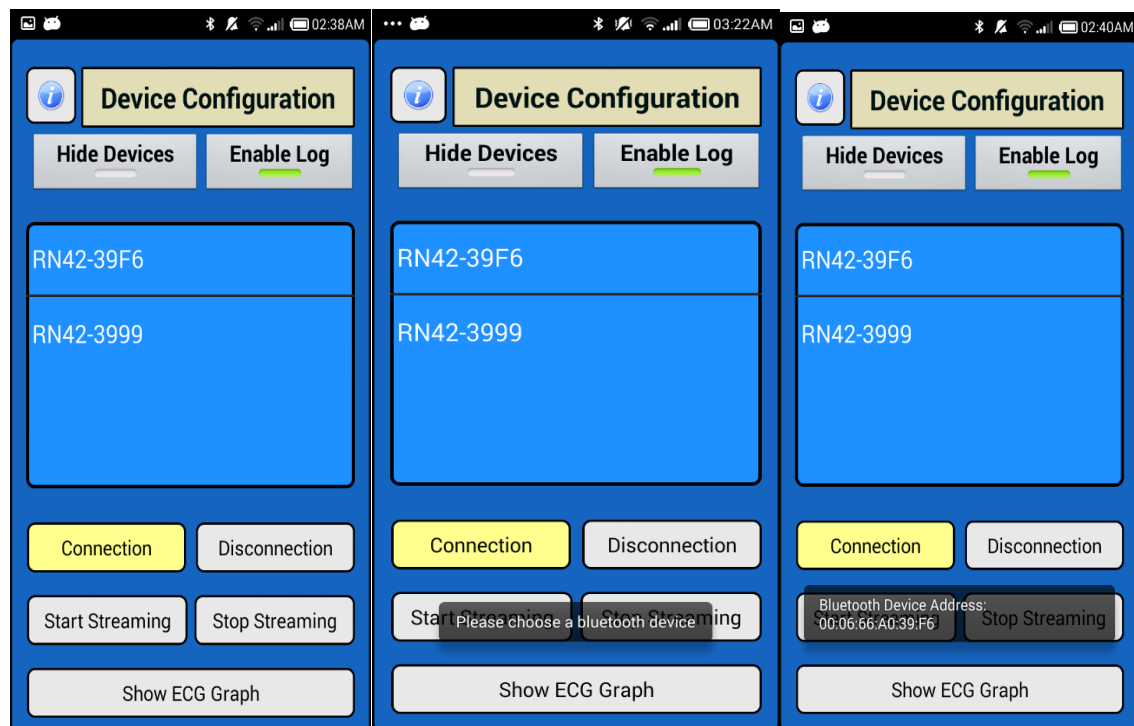
Πατώντας στο κουμπί αυτό γίνεται αποστολή των δεδομένων του ηλεκτροκαρδιογραφήματος, όπως αυτά λαμβάνονται από τους sensors της συσκευής Shimmer και αποστέλλονται στην κινητή μας συσκευή. Κατά τη διαδικασία αυτή η συσκευή Shimmer παρουσιάζεται στην εφαρμογή με μπλε χρώμα.

7. ΤΕΡΜΑΤΙΣΜΟΣ ΑΠΟΣΤΟΛΗΣ ΔΕΔΟΜΕΝΩΝ: Πατώντας στο κουμπί αυτό σταματά η αποστολή των δεδομένων από τη συσκευή Shimmer στην κινητή μας συσκευή. Η συσκευή Shimmer επιστρέφει στην κατάσταση Σύνδεσης και έχει πράσινο χρώμα.

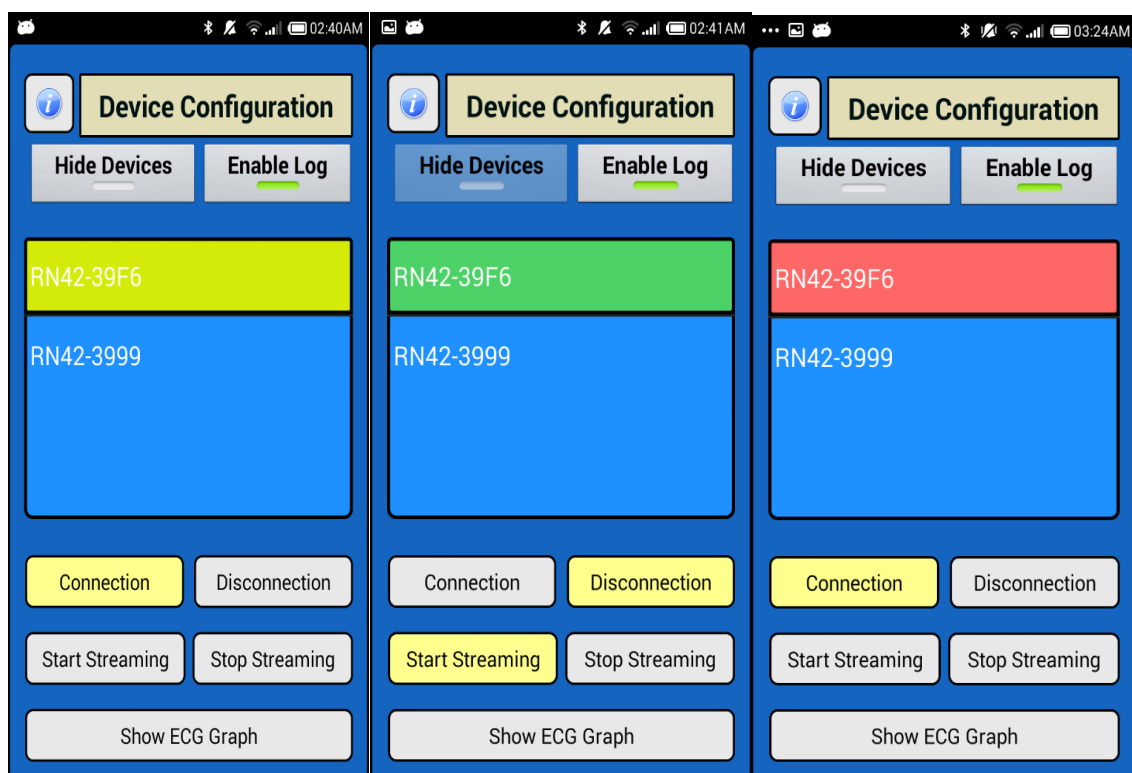
8. ΠΑΡΟΥΣΙΑΣΗ ΗΛΕΚΤΡΟΚΑΡΔΙΟΓΡΑΦΗΜΑΤΟΣ: Πατώντας στο κουμπί αυτό παρουσιάζεται το ηλεκτροκαρδιογράφημα σύμφωνα με τα δεδομένα που έλαβε η κινητή συσκευή από τους sensors της συσκευής Shimmer. Πιο συγκεκριμένα, παρουσιάζονται τα leads RA-LL και LA-LL.

- Αν πατήσουμε στο κουμπί Show Devices χωρίς να είναι ενεργοποιημένο το Bluetooth της κινητής συσκευής, εμφανίζεται το μήνυμα που βλέπουμε στο σχήμα πάνω δεξιά, το οποίο μας επιτρέπει ταυτόχρονα να ανοίξουμε το Bluetooth της συσκευής μας.
- Τα 5 κουμπιά Connection, Disconnection, Start Streaming, Stop Streaming και Show ECG Graph παραμένουν απενεργοποιημένα με την εμφάνιση της αρχικής οθόνης της Shimmer, ενώ τα κουμπιά Show Devices και Enable Log είναι ενεργοποιημένα.

ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ

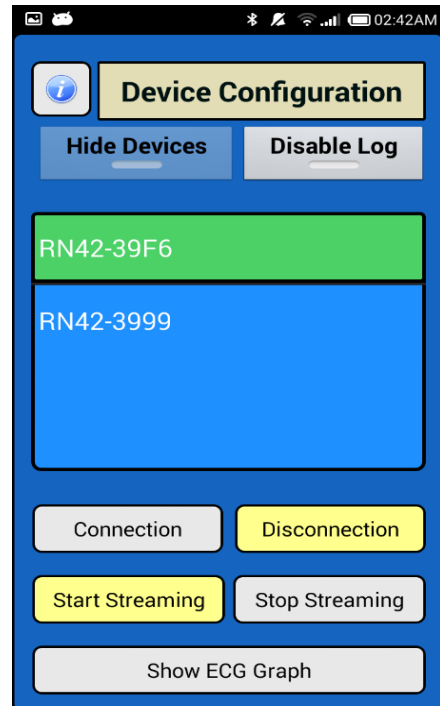
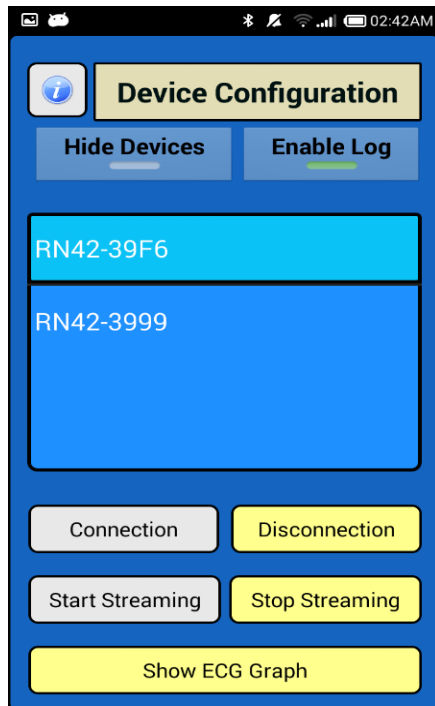


- Αφού έχουμε ενεργοποιήσει το Bluetooth, πατώντας στο κουμπί Show Devices εμφανίζονται οι paired συσκευές Shimmer, όπως φαίνεται στην εικόνα πάνω αριστερά.
- Αν πατήσουμε στο κουμπί Connection, μας εμφανίζεται το μήνυμα να επιλέξουμε πρώτα μια συσκευή Shimmer προτού την ενώσουμε με την κινητή μας συσκευή, όπως φαίνεται στην εικόνα πιο πάνω στο κέντρο.
- Ακολούθως, αν πατήσουμε επάνω σε μια συσκευή Shimmer, μας εμφανίζεται η διεύθυνσή της, γεγονός που αναφέρει ότι έχουμε επιλεγμένη μια Shimmer συσκευή, όπως φαίνεται στο σχήμα πάνω δεξιά.

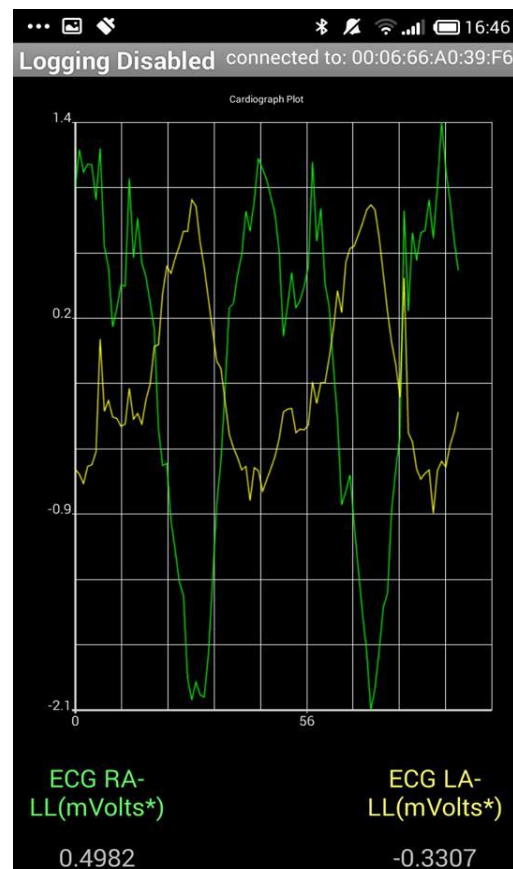


- Μετά την επιλογή μιας συσκευής Shimmer, πατούμε στο κουμπί Connection για να ενωθούμε με την συσκευή αυτή. Αυτόματα η συσκευή που επιλέγηκε χρωματίζεται με κίτρινο, που σημαίνει ότι αναμένεται η σύνδεση, όπως φαίνεται στην εικόνα πάνω αριστερά.
- Όταν μετά από κάποιο χρονικό διάστημα, το χρώμα μετατραπεί σε πράσινο, σημαίνει ότι πέτυχε η σύνδεση της εφαρμογής με την κινητή συσκευή, όπως φαίνεται στην οθόνη πιο πάνω στο κέντρο. Βλέπουμε ακόμη ότι έχουμε διαθέσιμες επιλογές το Disconnection και το Start Streaming.

- Αν πατήσουμε στο κουμπί Disconnection, η συσκευή θα αποσυνδεθεί από την εφαρμογή και για ένα μικρό χρονικό διάστημα θα χρωματιστεί κόκκινη, όπως φαίνεται στην εικόνα πάνω δεξιά.

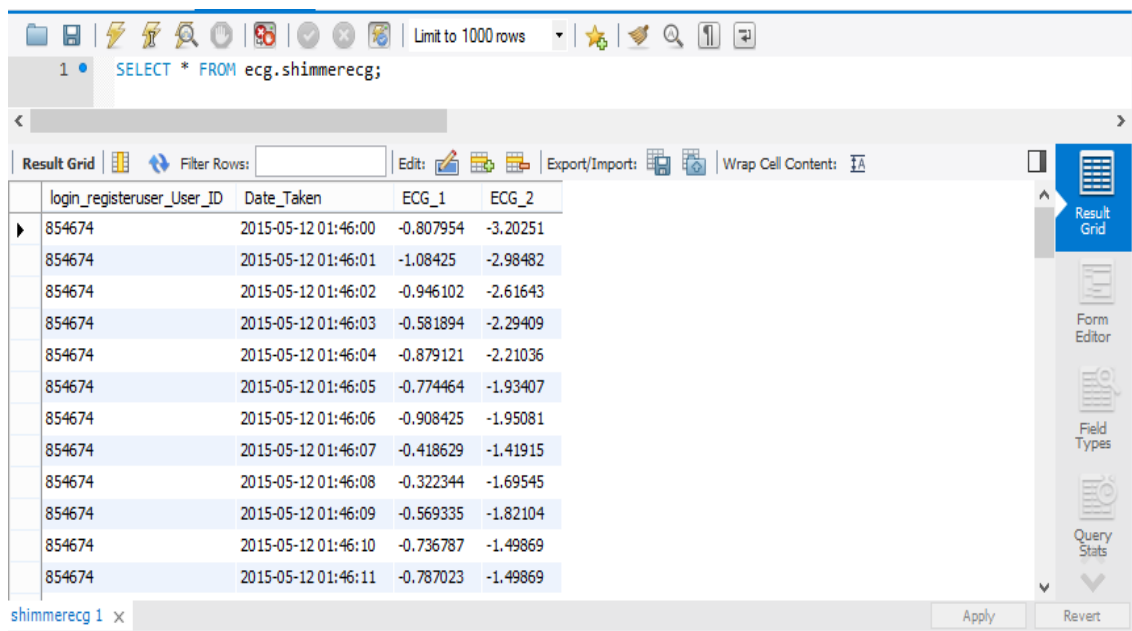


- Πατώντας στο κουμπί Start Streaming, η συσκευή Shimmer παίρνει χρώμα μπλε και αρχίζει η αποστολή δεδομένων στην κινητή συσκευή, όπως φαίνεται στο σχήμα πάνω αριστερά.
- Ακολούθως, μπορούμε να επιλέξουμε το Disconnection που και πάλι θα μας αποσυνδέσει από την συσκευή Shimmer ή την επιλογή Stop Streaming με την οποία θα επιστρέψουμε στην κατάσταση Connected. Εδώ μετά την αποστολή των δεδομένων μπορούμε να ενεργοποιήσουμε την



καταγραφή του ηλεκτροκαρδιογραφήματος, επιλέγοντας το Enable Log, όπως φαίνεται στην οθόνη πάνω στο κέντρο.

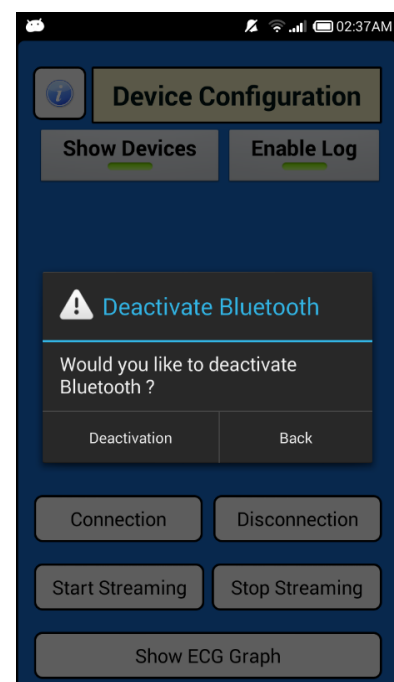
- Μια ακόμη επιλογή εκτός από τις 2 που αναφέραμε προηγουμένως, είναι και η επιλογή εμφάνισης του ηλεκτροκαρδιογραφήματος πατώντας το κουμπί Show ECG Graph. Με αυτήν την επιλογή θα εμφανιστεί το ηλεκτροκαρδιογράφημα του ασθενή με βάση τις τιμές που κατέγραψαν οι sensors της συσκευής Shimmer, όπως φαίνεται στο σχήμα πάνω δεξιά. Γίνεται ταυτόχρονη ενημέρωση των τιμών των 2 απαγωγών, που εμφανίζονται στο κάτω μέρος της οθόνης.



The screenshot shows a database application interface. At the top, there's a toolbar with various icons and a dropdown menu set to 'Limit to 1000 rows'. Below the toolbar, a SQL query is entered: 'SELECT * FROM ecg.shimmerrecg;'. The main area displays a 'Result Grid' with a table of data. The table has five columns: 'login_registeruser_User_ID', 'Date_Taken', 'ECG_1', and 'ECG_2'. There are 12 rows of data, all with the same 'login_registeruser_User_ID' (854674) and 'Date_Taken' (2015-05-12 01:46:00 to 2015-05-12 01:46:11). The 'ECG_1' and 'ECG_2' columns contain numerical values. On the right side, there's a sidebar with icons for 'Result Grid', 'Form Editor', 'Field Types', and 'Query Stats'. At the bottom, there are 'Apply' and 'Revert' buttons.

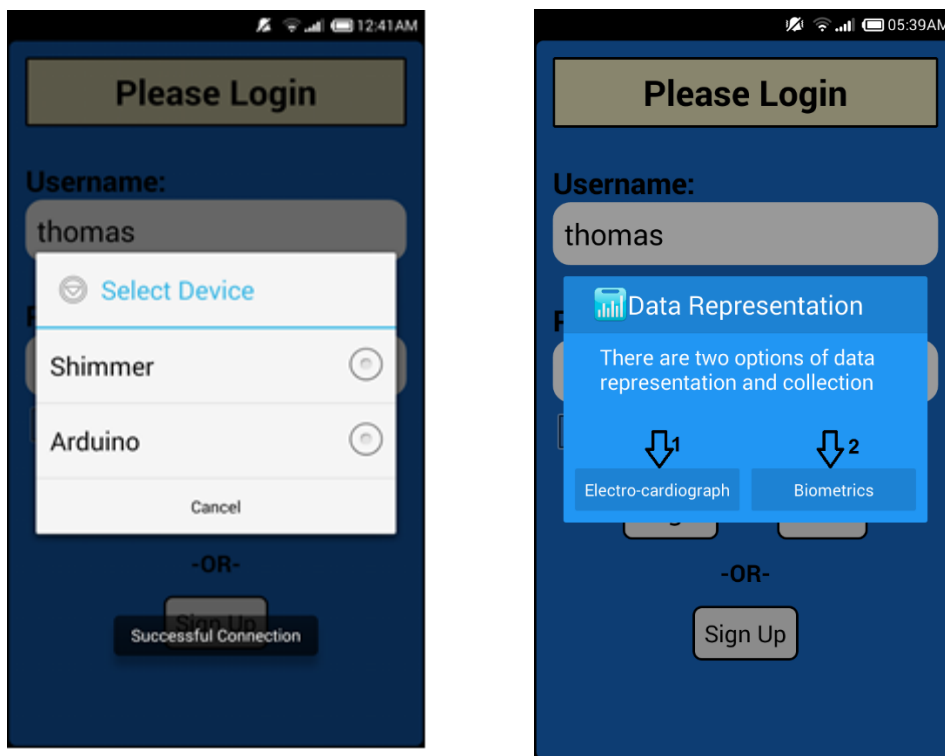
login_registeruser_User_ID	Date_Taken	ECG_1	ECG_2
854674	2015-05-12 01:46:00	-0.807954	-3.20251
854674	2015-05-12 01:46:01	-1.08425	-2.98482
854674	2015-05-12 01:46:02	-0.946102	-2.61643
854674	2015-05-12 01:46:03	-0.581894	-2.29409
854674	2015-05-12 01:46:04	-0.879121	-2.21036
854674	2015-05-12 01:46:05	-0.774464	-1.93407
854674	2015-05-12 01:46:06	-0.908425	-1.95081
854674	2015-05-12 01:46:07	-0.418629	-1.41915
854674	2015-05-12 01:46:08	-0.322344	-1.69545
854674	2015-05-12 01:46:09	-0.569335	-1.82104
854674	2015-05-12 01:46:10	-0.736787	-1.49869
854674	2015-05-12 01:46:11	-0.787023	-1.49869

- Πάνω στο τίτλο της οθόνης εμφανίζεται εάν πραγματοποιείται αποθήκευση της καταγραφής ηλεκτροκαρδιογραφήματος και σε ποια διεύθυνση είναι ενωμένο το κινητό τηλέφωνο. Στην εικόνα πιο πάνω βλέπουμε ότι όταν αρχίζει να σχηματίζεται η γραφική παράσταση, αποστέλλονται στη βάση δεδομένων η ταυτότητα του χρήστη και σε κάθε δευτερόλεπτο οι τιμές των 2 απαγωγών
- Να σημειωθεί ότι πατώντας το πλήκτρο επιστροφής εμφανίζεται η οθόνη που βλέπουμε αριστερά, με την οποία μπορούμε να



απενεργοποιήσουμε το Bluetooth για εξοικονόμηση ενέργειας.

ΗΛΕΚΤΡΟΚΑΡΔΙΟΓΡΑΦΗΜΑ ΜΕ ΤΟ ARDUINO ΚΑΙ ΤΗΝ ΠΛΑΚΕΤΑ ΤΗΣ LIBELIUM



Με τον έξοδο μας από την οθόνη της συσκευής Shimmer, εμφανίζεται η οθόνη που μπορούμε να επιλέξουμε μεταξύ της Shimmer και του Arduino. Αυτή τη φορά επιλέγουμε το Arduino και εμφανίζεται η οθόνη που βλέπουμε πιο πάνω δεξιά, με τις πιο κάτω επιλογές:

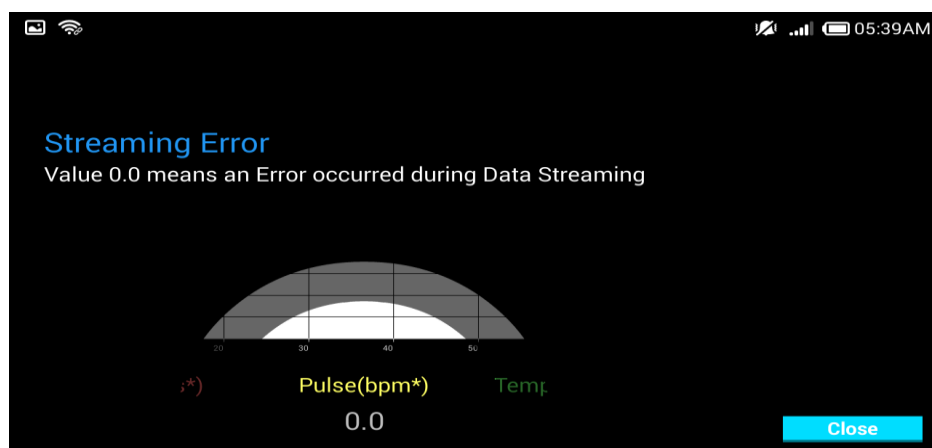
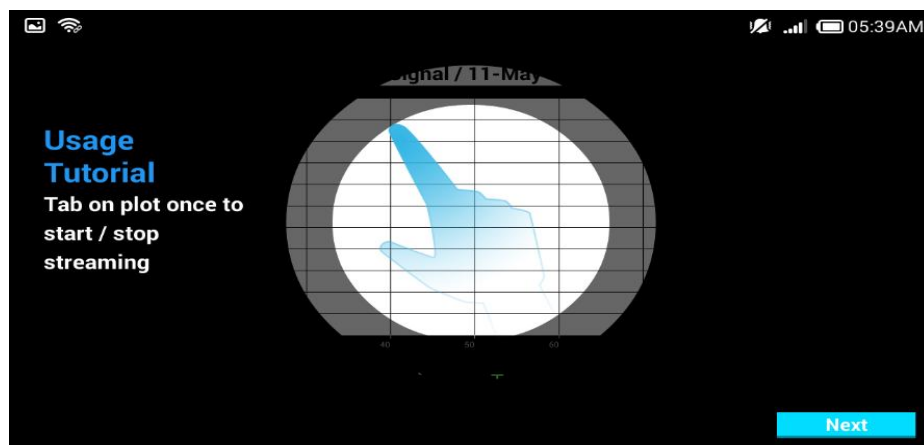
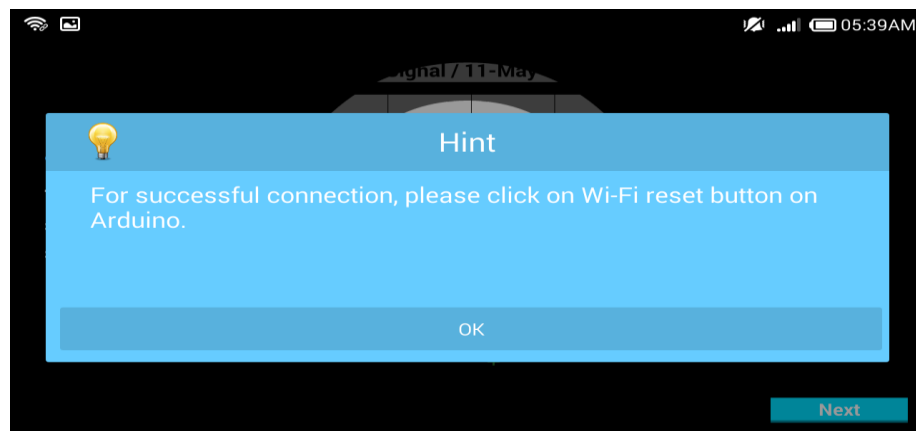
1. ΠΑΡΟΥΣΙΑΣΗ ΗΛΕΚΤΡΟΚΑΡΔΙΟΓΡΑΦΗΜΑΤΟΣ LIBELIUM:

Πατώντας σε αυτό το κουμπί, παρουσιάζεται το ηλεκτροκαρδιογράφημα όπως το λαμβάνουμε από τους sensors της πλακέτας e-Health της Libelium.

2. ΠΑΡΟΥΣΙΑΣΗ ΒΙΟΜΕΤΡΙΚΩΝ ΠΛΗΡΟΦΟΡΙΩΝ:

Πατώντας σε αυτό το κουμπί, εμφανίζονται οι βιομετρικές πληροφορίες του ασθενή, όπως η θερμοκρασία, η οξυγόνωση στο αίμα, συστολική και διαστολική πίεση, ηλεκτρομυογράφημα, διαγωγιμότητα, παλμοί, ροή αέρα συμπεριλαμβανόμενου και του ηλεκτροκαρδιογραφήματος.

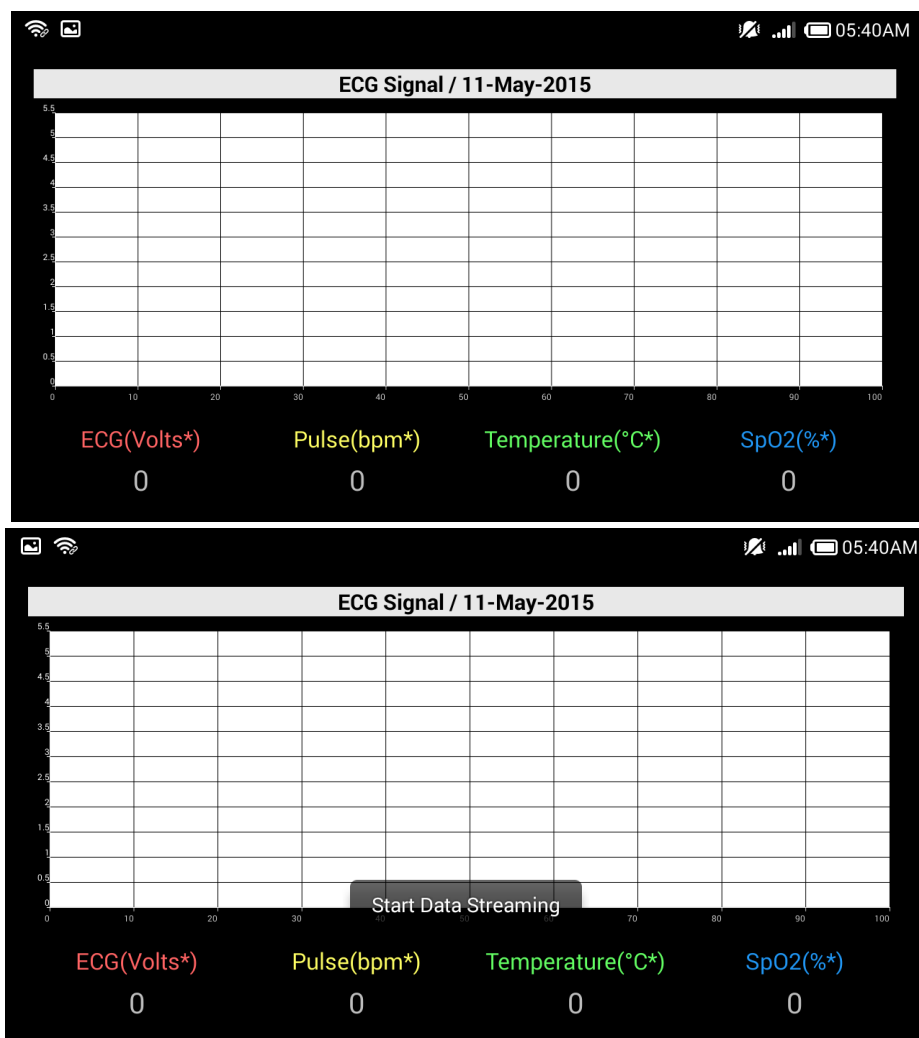
ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ



- Στο πιο πάνω παράδειγμα, πατήσαμε στο κουμπί Electro-Cardiograph για να παρουσιαστεί η γραφική παράσταση του ηλεκτροκαρδιογραφήματος όπως αποστέλλεται στην κινητή μας συσκευή από το Arduino. Βλέπουμε στην πρώτη

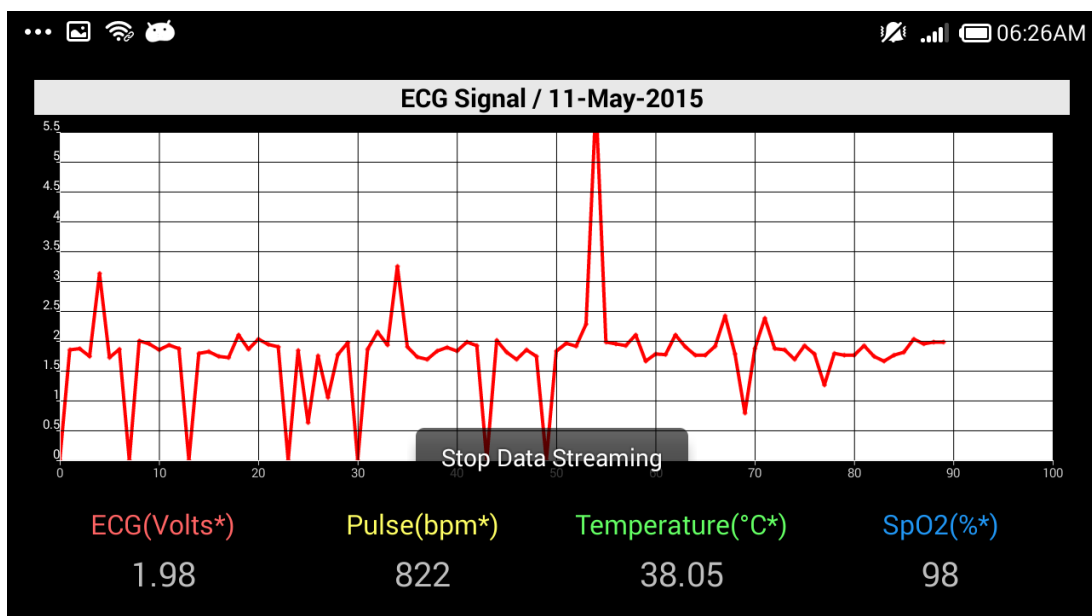
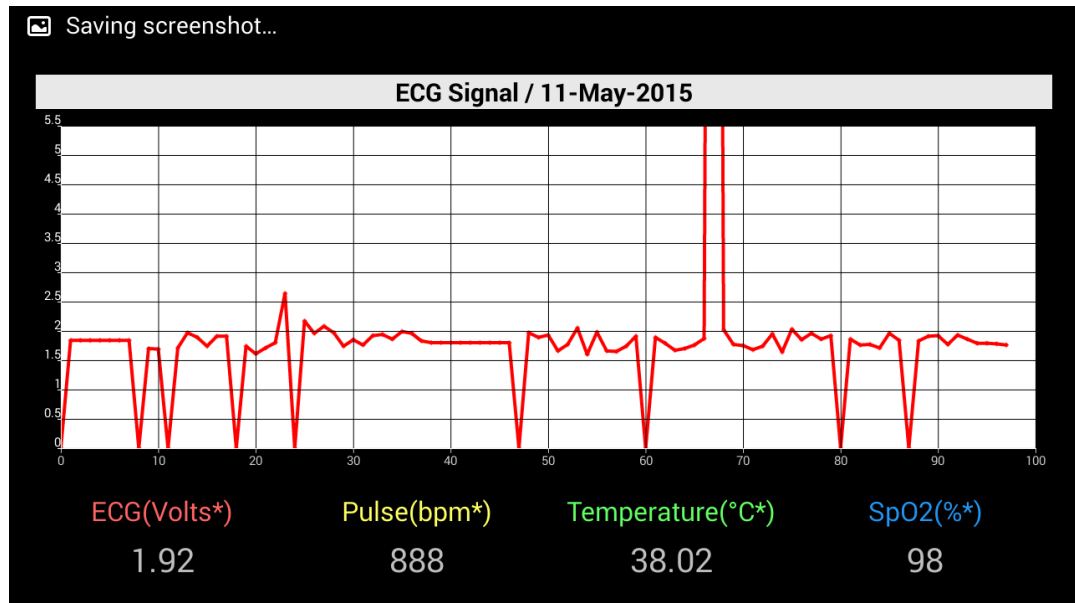
εικόνα ότι μας δίνεται μια συμβουλή, η οποία αναφέρει ότι πατώντας στο κουμπί reset στο Arduino έχουμε περισσότερες πιθανότητες για μια επιτυχημένη σύνδεση του Arduino με την εφαρμογή μας.

- Ακολουθεί και πάλι ένας οδηγός χρήσης όπως και στη συσκευή Shimmer. Ο συγκεκριμένος οδηγός τονίζει ότι πατώντας μια φορά στη γραφική παράσταση αρχίζει η μεταφορά δεδομένων από το Arduino στην εφαρμογή μας, ενώ πατώντας για δεύτερη φορά σταματά η μεταφορά αυτή.
- Στη συνέχεια επισυνάπτει ότι η παρουσία μηδενικής τιμής σε κάποια από τις τιμές που θα σταλούν, υποδεικνύει ότι προέκυψε κάποιο σφάλμα στην μεταφορά δεδομένων.



- Παρατηρούμε ότι εκτός από την γραφική παράσταση του ηλεκτροκαρδιογραφήματος, στο κάτω μέρος της οθόνης θα φαίνεται οι τιμές του ΗΚΓ, των παλμών, της θερμοκρασίας του ασθενή αλλά και του ποσοστού οξυγόνου που υπάρχει μέσα στο αίμα του ασθενή.

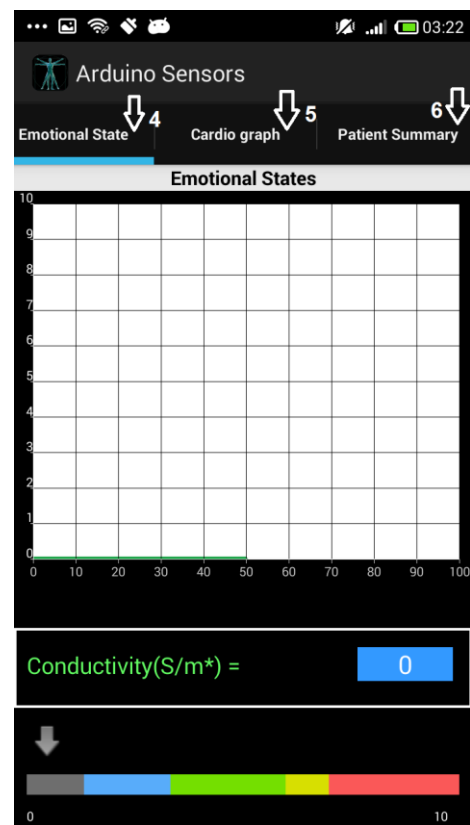
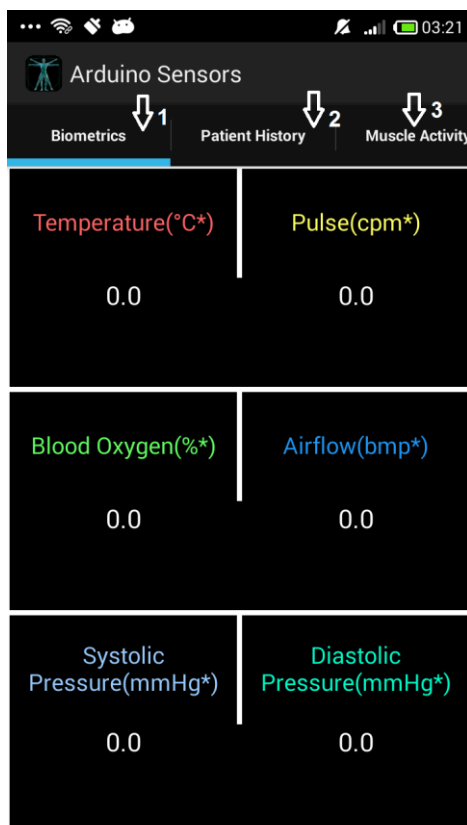
- Στο επάνω μέρος της οθόνης αυτής, παρουσιάζεται η ημερομηνία που λήφθηκαν οι μετρήσεις αυτές, έτσι ώστε να αποθηκεύονται οργανωμένα στη βάση δεδομένων οι πληροφορίες.
- Πατώντας μια φορά επάνω στη γραφική παράσταση, βλέπουμε ότι εμφανίζεται μήνυμα προς τον χρήστη, το οποίο αναφέρει ότι η μεταφορά των δεδομένων από το Arduino έχει εκκινήσει.



- Εφαρμόζοντας επάνω στον ασθενή τους αισθητήρες του Arduino, βλέπουμε ότι παρουσιάζονται τα δεδομένα στο κάτω μέρος της οθόνης, τα οποία αλλάζουν συνεχώς τιμές λόγω της real-time μεταφοράς δεδομένων στην εφαρμογή.

- Βλέπουμε επίσης ότι δημιουργείται επιτυχώς η γραφική παράσταση του ηλεκτροκαρδιογραφήματος του ασθενή, παίρνοντας τιμές από 0 μέχρι 5.5 Volts.
- Τέλος, πατώντας για δεύτερη φορά επάνω στη γραφική παράσταση, εμφανίζεται μήνυμα που ενημερώνει τον ασθενή ότι η μεταφορά των δεδομένων έχει διακοπεί, σταματώντας ταυτόχρονα την αλλαγή στις τιμές των υπολοίπων βιομετρικών πληροφοριών του ασθενή. Αν ξαναπατήσουμε επάνω στη γραφική παράσταση, τότε συνεχίζεται το ηλεκτροκαρδιογράφημα από το σημείο που έμεινε την τελευταία φορά που το σταματήσαμε και οι τιμές στο κάτω μέρος της οθόνης αρχίζουν και πάλι να αλλάζουν μέχρι την επανεκκίνηση.

ΒΙΟΜΕΤΡΙΚΕΣ ΠΛΗΡΟΦΟΡΙΕΣ ΑΣΘΕΝΗ ΜΕ ΤΟ ARDUINO ΚΑΙ ΤΗΝ ΠΛΑΚΕΤΑ ΤΗΣ LIBELIUM



Από πριν έχουμε αναφέρει ότι στην οθόνη επιλογής Shimmer ή Arduino, όταν επιλέξουμε το Arduino έχουμε 2 επιλογές, την παρουσίαση του ηλεκτροκαρδιογραφήματος ή των βιομετρικών στοιχείων. Σε αυτό το σημείο θα

αναφέρουμε ποιες επιλογές εμφανίζονται όταν επιλέξουμε τα βιομετρικά στοιχεία του ασθενή:

1. ΠΛΗΡΟΦΟΡΙΕΣ ΒΙΟΜΕΤΡΙΚΩΝ ΠΛΗΡΟΦΟΡΙΩΝ:

Οι πληροφορίες που εμφανίζονται πατώντας αυτό το κουμπί, φαίνονται στην εικόνα πιο πάνω αριστερά. Συγκεκριμένα, με μια συσκευή Arduino παίρνουμε τον παλμό του ασθενή, τη θερμοκρασία του, το ποσοστό οξυγόνου στο αίμα του, την ροή του αέρα που εισπνέει και εκπνέει, την συστολική και την διαστολική πίεση.

2. ΕΜΦΑΝΙΣΗ ΙΣΤΟΡΙΚΟΥ ΑΣΘΕΝΗ:

Πατώντας σε αυτό το κουμπί, εμφανίζονται οι πληροφορίες για το ιστορικό του ασθενή όσον αφορά τα βιομετρικά του στοιχεία και η δυνατότητα επιλογής ενός χρονικού διαστήματος (μέρα, μήνας, χρόνος κτλ.) για την προβολή των πληροφοριών.

3. ΕΜΦΑΝΙΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ ΜΥΟΣ:

Πατώντας στο κουμπί αυτό πραγματοποιούμε παρουσίαση μιας γραφικής παράστασης, η οποία περιλαμβάνει τις τιμές μεταβολής της έντασης του μυός του ασθενή.

4. ΕΜΦΑΝΙΣΗ ΣΥΝΑΙΣΘΗΜΑΤΙΚΗΣ ΚΑΤΑΣΤΑΣΗΣ:

Πατώντας στο κουμπί αυτό γίνεται παρουσίαση μιας γραφικής παράστασης, η οποία περιλαμβάνει τις τιμές διαγωγιμότητας στο δέρμα του ασθενή, κάνοντας και μια τυπική διάγνωση της συναισθηματικής του κατάστασης, όπως φαίνεται στην εικόνα πάνω δεξιά.

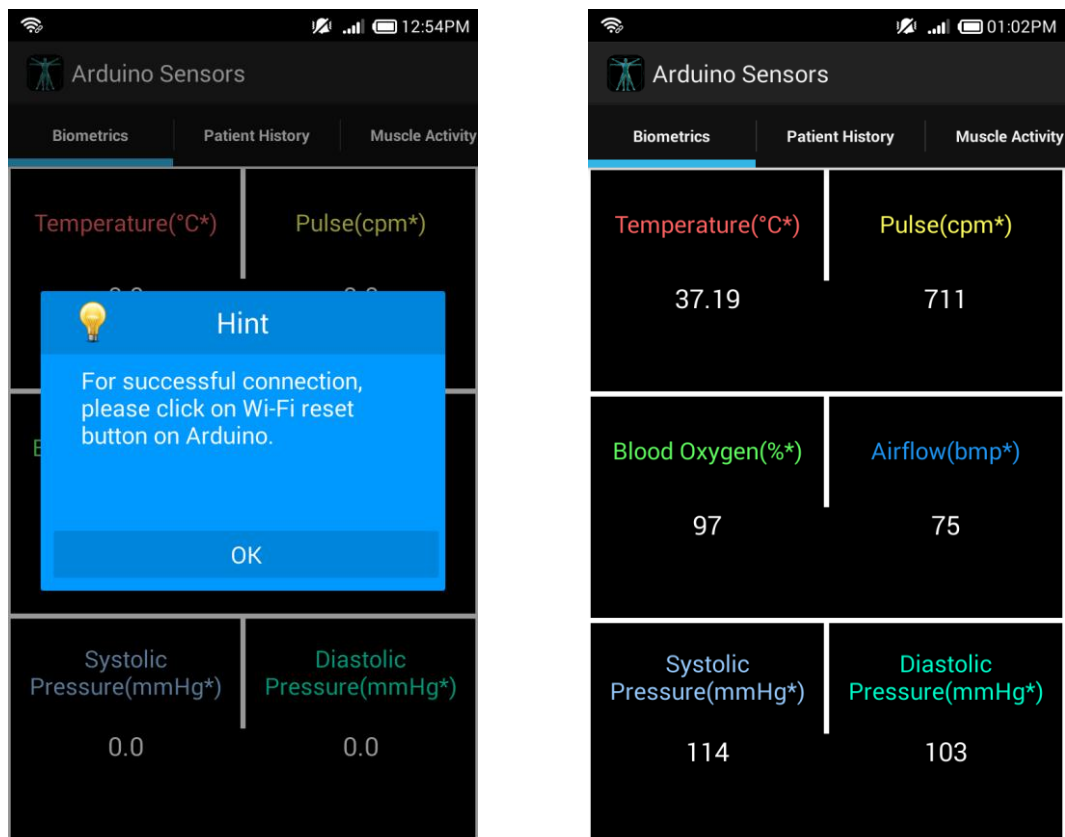
5. ΕΜΦΑΝΙΣΗ ΗΛΕΚΤΡΟΚΑΡΔΙΟΓΡΑΦΗΜΑΤΟΣ:

Πατώντας στο κουμπί αυτό γίνεται παρουσίαση και πάλι της γραφικής παράστασης του ηλεκτροκαρδιογραφήματος του ασθενή.

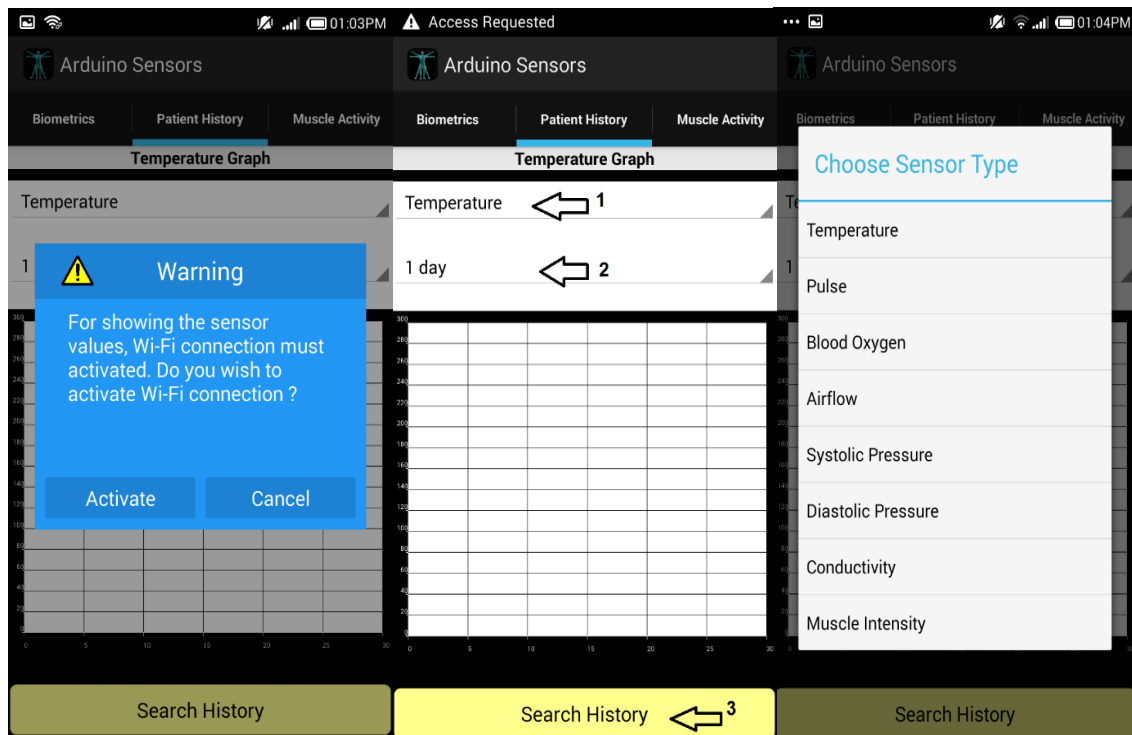
6. ΠΕΡΙΛΗΨΗ ΒΙΟΜΕΤΡΙΚΩΝ ΣΤΟΙΧΕΙΩΝ:

Πατώντας στο κουμπί αυτό γίνεται μια γενική συλλογή τιμών για όλα τα βιομετρικά στοιχεία του ασθενή και η αποστολή τους στην βάση δεδομένων. Ακόμη, εξάγουμε και μια περίληψη της κατάστασης του ασθενούς τόσο φωνητικά όσο και γραπτά.

ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ



- Εφόσον έχουμε συνδέσει τα κατάλληλα καλώδια με τους sensors επάνω στον ασθενή παρατηρούμε ότι εμφανίζονται τιμές για τη θερμοκρασία του σώματός του, του παλμού του, του ποσοστού οξυγόνου στο αίμα του, της ροής του αέρα κατά την εισπνοή και εκπνοή και για τη συστολική και διαστολική πίεση, όπως φαίνεται στο σχήμα πάνω δεξιά. Οι τιμές αυτές αλλάζουν συνεχώς μέχρι να αποσυνδέσουμε τους sensors.
- Να αναφέρουμε ότι για αυτήν την οθόνη, παρουσιάζεται στην αρχή μια συμβουλή, όπως φαίνεται στην εικόνα πάνω αριστερά, η οποία αναφέρει ότι για επιτυχημένη σύνδεση πατούμε στο κουμπί reset για το Wi-Fi του Arduino.



- Πατώντας στο δεύτερο κουμπί της καρτέλας, εμφανίζεται ένα μήνυμα το οποίο μας προτρέπει να ανοίξουμε τα Wi-Fi της κινητής συσκευής προκειμένου να παρουσιαστούν τα ιστορικά δεδομένα του ασθενή, όπως βλέπουμε στην εικόνα πάνω αριστερά. Αυτό συμβαίνει διότι για την παρουσίαση αυτών των πληροφοριών απαιτείται πρώτα σύνδεση με την βάση δεδομένων.
- Αφού έχουμε ενεργοποιήσει τα Wi-Fi, παίρνουμε την εικόνα που βλέπουμε επάνω στο κέντρο, στην οποία έχουμε 3 επιλογές:

1. ΕΠΙΛΟΓΗ ΒΙΟΜΕΤΡΙΚΟΥ ΣΤΟΙΧΕΙΟΥ:

Πατώντας αυτό το κουμπί, μας παρουσιάζονται όλα τα βιομετρικά στοιχεία από τη βάση δεδομένων και πρέπει να επιλέξουμε ένα για την παρουσίασή του, όπως βλέπουμε και στην εικόνα πάνω δεξιά.

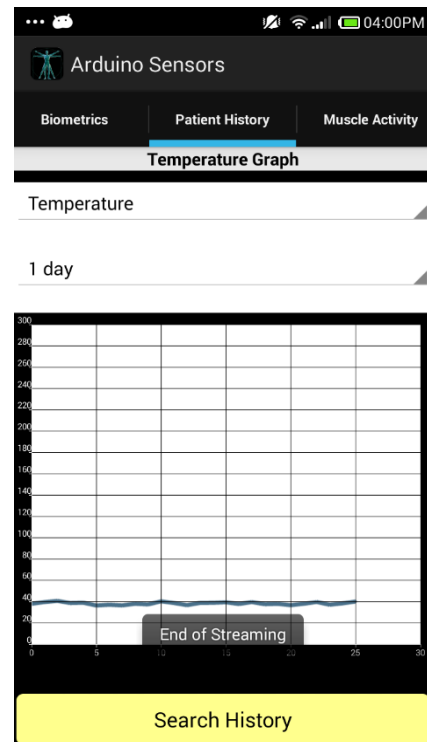
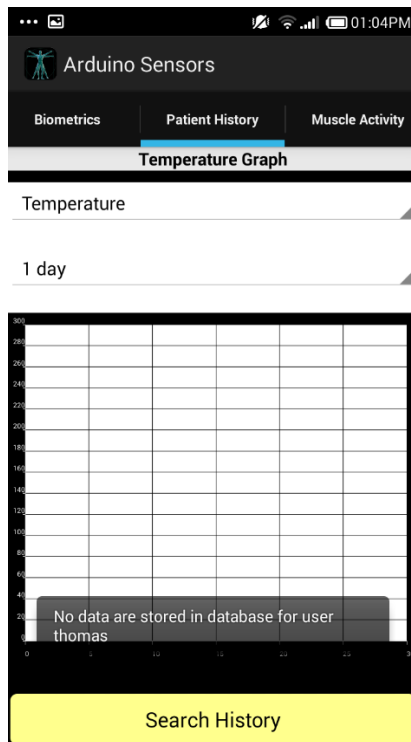
2. ΕΠΙΛΟΓΗ ΠΕΡΙΟΔΟΥ ΚΑΤΑΓΡΑΦΗΣ:

Πατώντας σε αυτό το κουμπί, εμφανίζονται οι τιμές που καταγράφηκαν από τον ασθενή, όπως φαίνεται στην εικόνα κάτω αριστερά. Και εδώ πρέπει να επιλέξουμε μια χρονική στιγμή που επιθυμούμε.

3. ΠΑΡΟΥΣΙΑΣΗ ΓΡΑΦΙΚΗΣ ΠΑΡΑΣΤΑΣΗΣ:

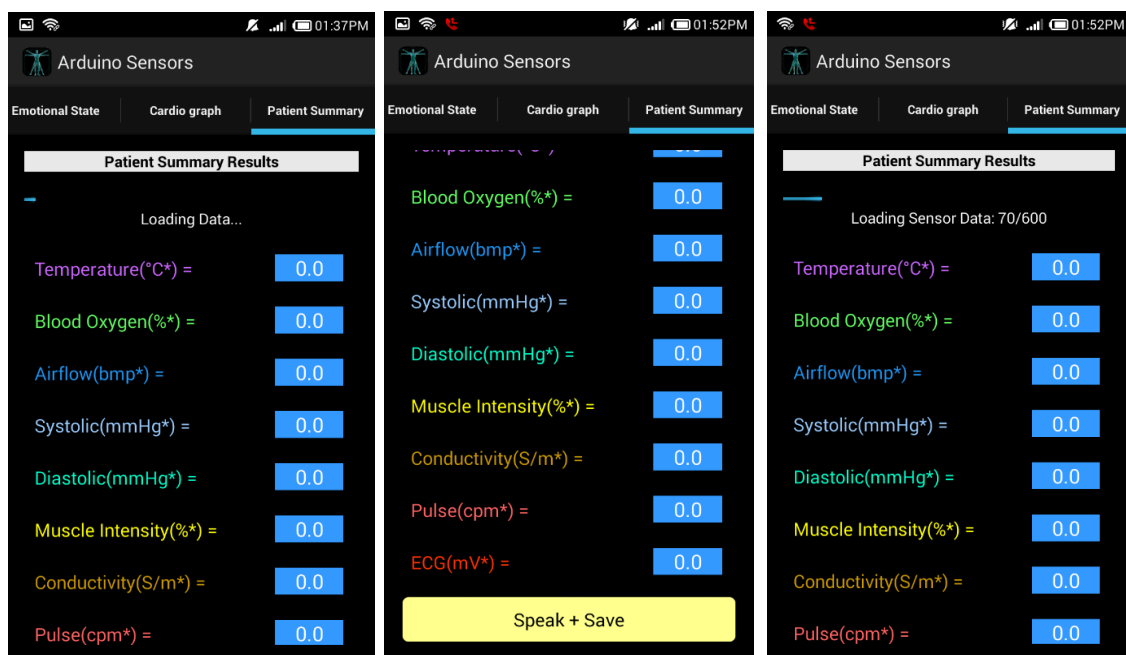
Πατώντας στο κουμπί αυτό πραγματοποιούμε παρουσίαση της γραφικής παράστασης του βιομετρικού στοιχείου που έχουμε επιλέξει για την συγκεκριμένη χρονική περίοδο. Αν δεν υπάρχουν δεδομένα στη βάση

δεδομένων για τον ασθενή, τότε παρουσιάζεται το μήνυμα που βλέπουμε στην οθόνη κάτω στο κέντρο. Αν έχουμε την περίπτωση που υπάρχουν κανονικά τα δεδομένα, τότε παρουσιάζεται η γραφική παράσταση του βιομετρικού στοιχείου, στην περίπτωση μας της θερμοκρασίας του ασθενή μέχρι να τελειώσουν τα δεδομένα από τη βάση δεδομένων (End of Streaming).

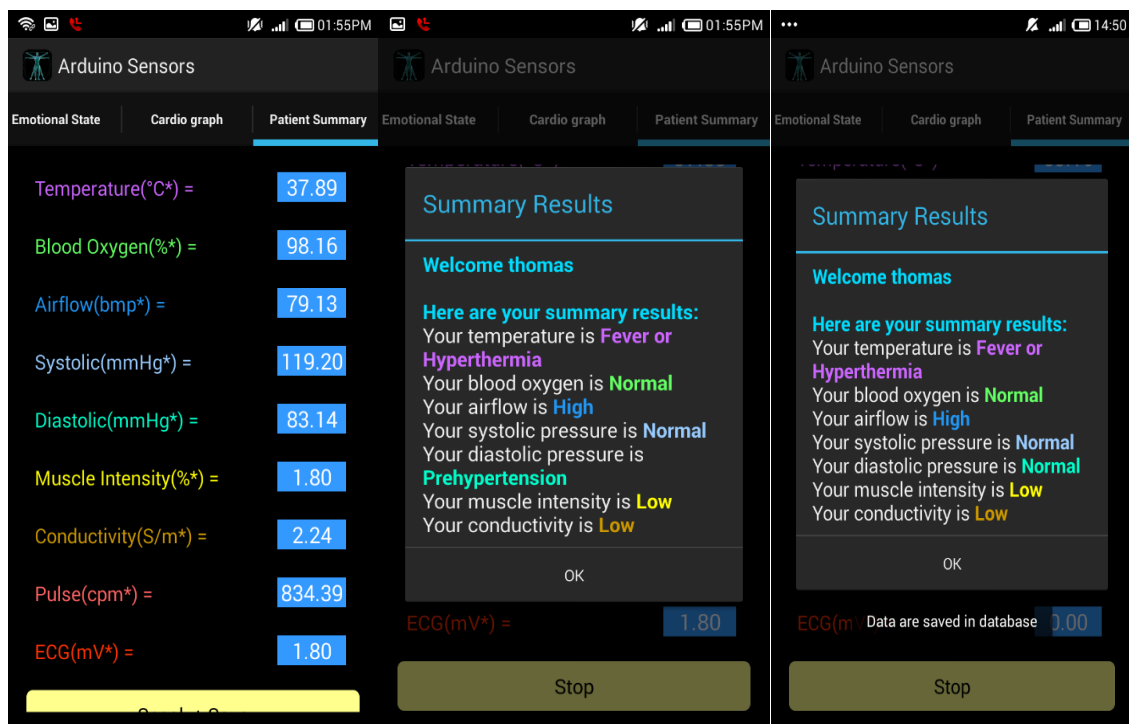




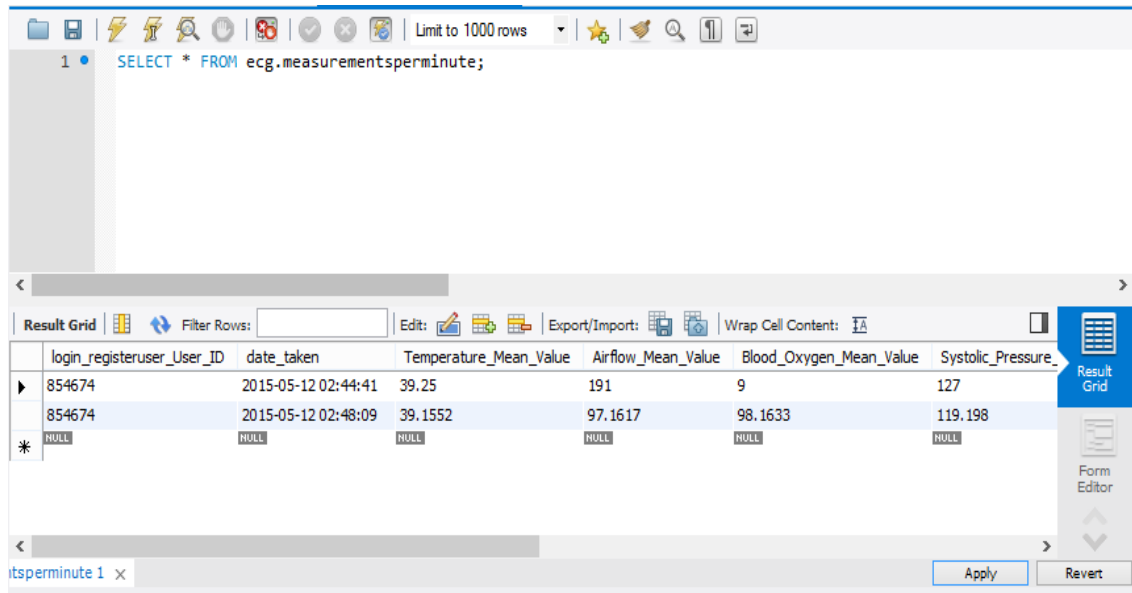
- Στις επόμενες 3 οθόνες που βλέπουμε πιο πάνω, παρουσιάζονται οι γραφικές παραστάσεις της έντασης του μυός, της διαγωγιμότητας του δέρματος και του ηλεκτροκαρδιογραφήματος του ασθενή, εφόσον έχουμε πρώτα συνδέσει ορθά τους αισθητήρες επάνω στον ασθενή. Παρατηρούμε ότι και στις 3 οθόνες παρουσιάζεται η τιμή κάτω δεξιά στην οθόνη για το κάθε βιομετρικό στοιχείο του ασθενή, η οποία αλλάζει συνεχώς μέχρι να αποσυνδέσουμε το Arduino από την κινητή μας συσκευή ή αφαιρέσουμε τους αισθητήρες.



- Στην τελευταία οθόνη, το Patient Summary, γίνεται πρώτα ένα loading των δεδομένων όπως βλέπουμε στην εικόνα πάνω αριστερά, εφόσον πρώτα έχουμε συνδέσει όλα τα βιομετρικά στοιχεία πάνω στον ασθενή.
- Ακολούθως, μετά από λίγα δευτερόλεπτα, γίνεται η μεταφορά των δεδομένων από το Arduino στην κινητή μας συσκευή, διαδικασία η οποία διαρκεί 1 λεπτό όπως φαίνεται στο σχήμα πάνω στο κέντρο.
- Παρατηρούμε ότι στο κάτω μέρος της οθόνης έχουμε ένα κουμπί, το Speak + Save, όπως βλέπουμε στο σχήμα επάνω δεξιά, το οποίο θα μεταφέρει τις τιμές που λάβουμε για τα βιομετρικά στοιχεία στη βάση δεδομένων.
- Αφού έχει περάσει το λεπτό και πήραμε όλες τις τιμές για τα βιομετρικά στοιχεία του ασθενή, όπως φαίνεται στο σχήμα κάτω αριστερά, πατούμε στο κουμπί Speak + Save για την αποστολή των πληροφοριών στη βάση δεδομένων. Βλέπουμε ότι με βάση τις τιμές που πήραμε από το Arduino, η εφαρμογή μας παρουσιάζει διάφορες διαγνώσεις για τον ασθενή, όπως παρατηρούμε και στην εικόνα κάτω στο κέντρο. Τέλος, οι πληροφορίες που εμφανίζονται σε αυτήν την εικόνα, αναγγέλλονται και από την εφαρμογή ταυτόχρονα φωνητικά, ενώ παράλληλα εμφανίζεται και ένα μήνυμα όπως βλέπουμε στην εικόνα κάτω δεξιά ότι τα δεδομένα αποθηκεύτηκαν στη βάση δεδομένων.

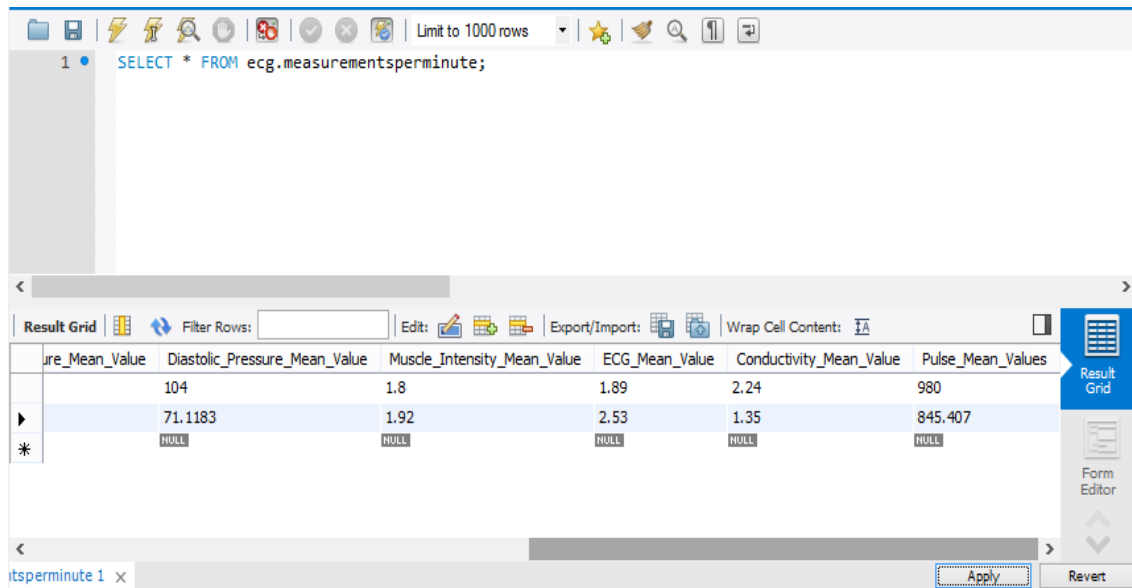


- Πιο κάτω βλέπουμε την αναπαράσταση των πληροφοριών που στάλθηκαν στη βάση δεδομένων.



The screenshot shows a database query tool interface. At the top, there is a toolbar with various icons and a dropdown menu set to "Limit to 1000 rows". Below the toolbar, a SQL query is entered in a text area: `SELECT * FROM ecg.measurementsperminute;`. The results are displayed in a "Result Grid" below the query. The grid has a header row with the following columns: `login_registeruser_User_ID`, `date_taken`, `Temperature_Mean_Value`, `Airflow_Mean_Value`, `Blood_Oxygen_Mean_Value`, and `Systolic_Pressure`. The first two rows of data are visible, followed by a row of NULL values. The first row has values: 854674, 2015-05-12 02:44:41, 39.25, 191, 9, 127. The second row has values: 854674, 2015-05-12 02:48:09, 39.1552, 97.1617, 98.1633, 119.198. The third row has NULL values for all columns. The grid is scrollable, and there are buttons for "Apply" and "Revert" at the bottom right.

login_registeruser_User_ID	date_taken	Temperature_Mean_Value	Airflow_Mean_Value	Blood_Oxygen_Mean_Value	Systolic_Pressure
854674	2015-05-12 02:44:41	39.25	191	9	127
854674	2015-05-12 02:48:09	39.1552	97.1617	98.1633	119.198
NULL	NULL	NULL	NULL	NULL	NULL



The screenshot shows the same database query tool interface as above, but with a different SQL query: `SELECT * FROM ecg.measurementsperminute;`. The results are displayed in a "Result Grid" below the query. The grid has a header row with the following columns: `ure_Mean_Value`, `Diastolic_Pressure_Mean_Value`, `Muscle_Intensity_Mean_Value`, `ECG_Mean_Value`, `Conductivity_Mean_Value`, and `Pulse_Mean_Value`. The first two rows of data are visible, followed by a row of NULL values. The first row has values: 104, 71.1183, 1.8, 1.89, 2.24, 980. The second row has values: 71.1183, 1.92, 2.53, 1.35, 845.407. The third row has NULL values for all columns. The grid is scrollable, and there are buttons for "Apply" and "Revert" at the bottom right.

ure_Mean_Value	Diastolic_Pressure_Mean_Value	Muscle_Intensity_Mean_Value	ECG_Mean_Value	Conductivity_Mean_Value	Pulse_Mean_Value
104	71.1183	1.8	1.89	2.24	980
71.1183	1.92	2.53	1.35	845.407	
NULL	NULL	NULL	NULL	NULL	NULL

➤ **Εγχειρίδιο χρήσης ιατρού**

ΕΝΑΡΞΗ ΕΦΑΡΜΟΓΗΣ ΚΑΙ ΣΥΝΔΕΣΗ ΣΤΟ ΣΥΣΤΗΜΑ

Με την έναρξη της εφαρμογής εμφανίζεται η αρχική οθόνη που φαίνεται πιο πάνω. Ο γιατρός αφού έχει πρώτα δημιουργήσει λογαριασμό και υπάρχουν οι πληροφορίες του στη βάση δεδομένων, κάνει Login χρησιμοποιώντας τις 3 πιο κάτω επιλογές:

1. ΕΙΣΑΓΩΓΗ USERNAME:

Ο γιατρός εισάγει το username του όπως το όρισε στη βάση δεδομένων.

2. ΕΙΣΑΓΩΓΗ PASSWORD:

Ο γιατρός εισάγει τον αντίστοιχο κωδικό του, όπως τον όρισε στη βάση δεδομένων.

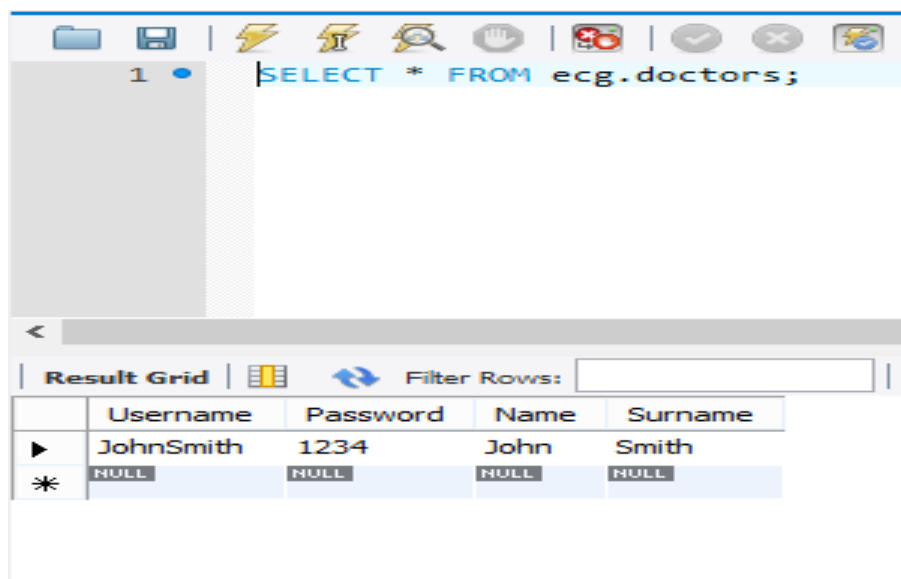
3. ΕΠΙΛΟΓΗ ΕΙΣΟΔΟΥ ΣΤΗΝ ΕΦΑΡΜΟΓΗ:

Πατώντας στο κουμπί αυτό ο γιατρός εισέρχεται στην εφαρμογή.

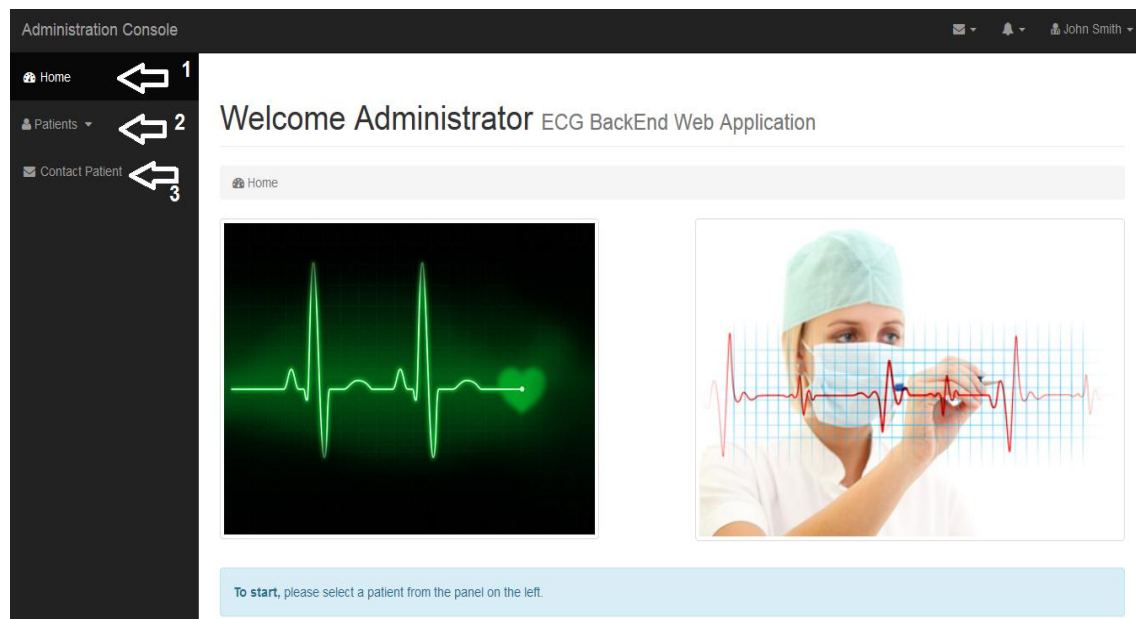
ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ



- Στην περίπτωση που ο ιατρός εισήγαγε λάθος username ή password, ή δεν συμπλήρωσε και τα 2 αυτά πεδία και πάτησε στο κουμπί Login, μας εμφανίζεται το μήνυμα λάθος όπως φαίνεται στην εικόνα πάνω αριστερά.
- Αν ο ιατρός εισήγαγε τα σωστά στοιχεία, στην περίπτωση μας το username JohnSmith και το password 1234, όπως φαίνονται στην εικόνα πάνω δεξιά και στη βάση δεδομένων στην εικόνα πιο κάτω, τότε ο ιατρός καταφέρνει επιτυχημένη σύνδεση στην εφαρμογή.



ΠΑΡΟΥΣΙΑΣΗ ΒΙΟΜΕΤΡΙΚΩΝ ΣΤΟΙΧΕΙΩΝ ΚΑΙ ΚΑΡΔΙΟΓΡΑΦΗΜΑΤΟΣ ΑΣΘΕΝΩΝ



Με την είσοδο μας στην εφαρμογή, έχουμε την αρχική οθόνη του συστήματος όπως φαίνεται στην εικόνα πιο πάνω. Παρατηρούμε στο κάτω μέρος της οθόνης ότι υπάρχει ένα μήνυμα προς τον ιατρό, που του υποδεικνύει ότι για να παρουσιαστούν τα δεδομένα πρέπει να επιλέξει ένα ασθενή από την λίστα στα αριστερά. Όπως βλέπουμε από τις 2 εικόνες πιο κάτω και την εικόνα πιο πάνω, έχουμε τις παρακάτω 6 επιλογές:

1. ΕΠΙΛΟΓΗ ΕΠΙΣΤΡΟΦΗΣ ΣΤΗΝ ΑΡΧΙΚΗ ΣΕΛΙΔΑ:

Πατώντας στο πεδίο αυτό έχουμε τη δυνατότητα να επιστρέψουμε στην αρχική οθόνη του συστήματος, όπως τη βλέπουμε στην εικόνα πιο πάνω.

2. ΕΠΙΛΟΓΗ ΕΜΦΑΝΙΣΗΣ ΑΣΘΕΝΩΝ:

Πατώντας στο πεδίο αυτό εμφανίζεται μια λίστα με τους ασθενείς που επιβλέπει ο γιατρός που εισήχθη στην εφαρμογή. Αυτή η επιλογή μας εμφανίζει τις επιλογές 5 και 6.

3. ΕΠΙΛΟΓΗ ΕΠΙΚΟΙΝΩΝΙΑΣ ΜΕ ΑΣΘΕΝΗ:

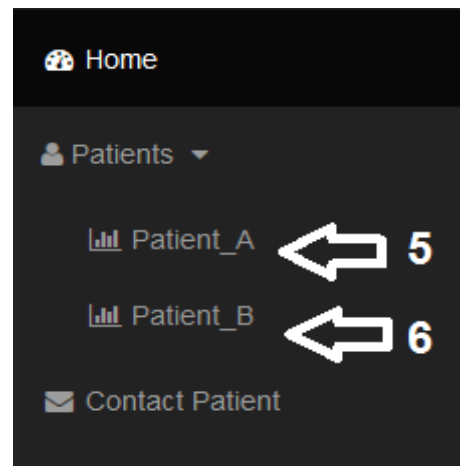
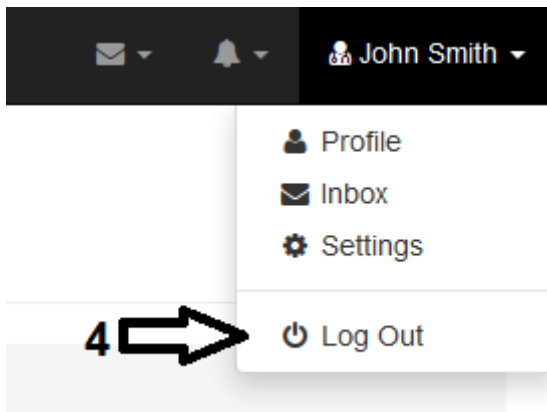
Πατώντας στο πεδίο αυτό, έχουμε τη δυνατότητα να στείλουμε με e-mail στον ασθενή τις διαγνώσεις που συμπέρανε ο ιατρός, ή να έχει οποιουδήποτε είδους επικοινωνία ο ιατρός με τον ασθενή μέσω e-mail.

4. ΕΠΙΛΟΓΗ ΑΠΟΣΥΝΔΕΣΗΣ:

Πατώντας στο κουμπί αποσυνδεόμαστε από την εφαρμογή και επιστρέφουμε στην οθόνη Login.

5+ 6. ΕΜΦΑΝΙΣΗ ΣΤΟΙΧΕΙΩΝ ΑΣΘΕΝΗ:

Πατώντας στο πεδίο αυτό εμφανίζονται όλες οι πληροφορίες για ένα ασθενή.

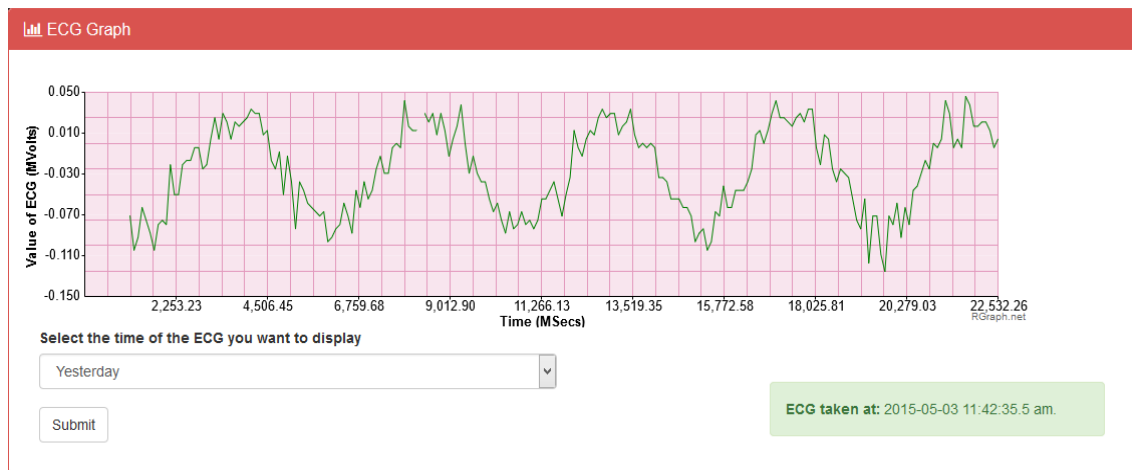


ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ

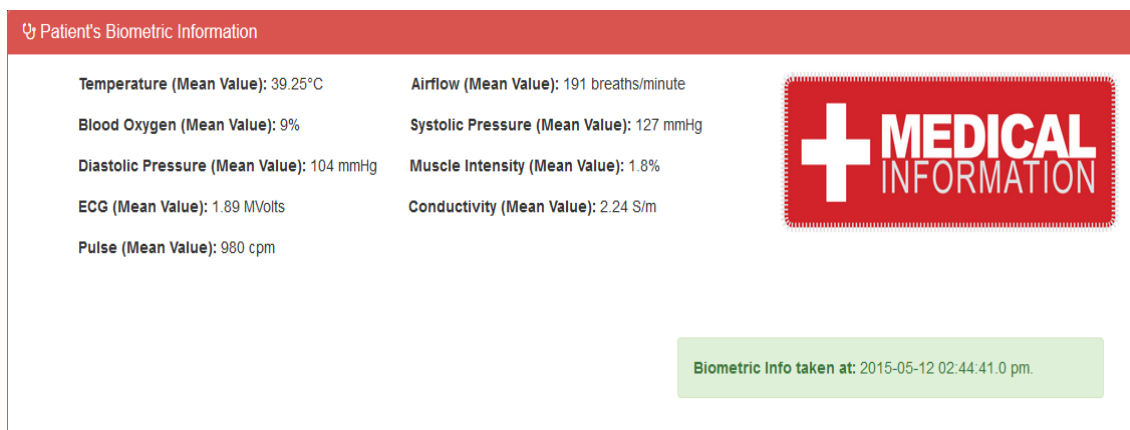
Patient Jennifer Robinson

Home / Patients			
Patient's Personal Information			
User ID: 785463	Name: Jennifer	Last Name: Robinson	Gender: Female
Date of Birth: 1981-06-25	Street: Tiffany's	Number of Street: 23	City: California
Country: USA	Postal Code: 3537	Telephone Number: 99654321	E-mail: jennifer@gmail.com

- Στην πιο πάνω εικόνα, εμφανίζονται όλα τα προσωπικά στοιχεία του ασθενή, όπως ο ίδιος ο ασθενής τα καταχώρησε στη βάση δεδομένων κάνοντας λογαριασμό για το android σύστημα που περιγράψαμε προηγουμένως.



- Στην πιο πάνω οθόνη, παρουσιάζεται το ηλεκτροκαρδιογράφημα όπως το λαμβάνουμε από τον πίνακα ηλεκτροκαρδιογραφήματος της Shimmer στη βάση δεδομένων. Ο ιατρός μπορεί να εμφανίζει τα ηλεκτροκαρδιογραφήματα του ασθενή χρονικά, δηλαδή της εβδομάδας, του μήνα, του χρόνου κτλ. Κάτω δεξιά στην οθόνη παρουσιάζεται ένα μήνυμα που αναφέρει την ημερομηνία και την ώρα που ο ασθενής κατέγραψε το ηλεκτροκαρδιογράφημα του.



- Στην πιο πάνω οθόνη, παρουσιάζονται όλα τα βιομετρικά στοιχεία όπως καταγράφηκαν από τη συσκευή Arduino και την πλακέτα e-Health της Libelium στη βάση δεδομένων. Κάτω δεξιά όπως και στην προηγούμενη περίπτωση με το ηλεκτροκαρδιογράφημα πιο πάνω, παρουσιάζεται η ημερομηνία και η ώρα καταγραφής όλων των βιομετρικών πληροφοριών του ασθενή.

Contact a Patient Send your Medical Advice/Information

[Home](#) / [Contact Patient](#)

Enter the E-mail of the patient: ↩ 1 Enter the subject of the E-mail: ↩ 2

E-mail Address Subject of E-mail

Enter the message which you want to send: ↩ 3

Body of E-mail

Submit ↩ 4

A green banner with the text "Doctor Advice" in a large, serif font on the left. On the right, there is a photograph of an elderly male patient sitting and talking to a male doctor in a white coat who is leaning over him.

- Στην πιο πάνω οθόνη, ο ιατρός μπορεί να στείλει e-mail στον ασθενή για να του αναφέρει τις διαγνώσεις του με βάση τα ιατρικά στοιχεία που του έστειλε ο ασθενής στη βάση δεδομένων, ή απλά για να έχει οποιαδήποτε επικοινωνία με τον ασθενή. Βλέπουμε στην πιο πάνω εικόνα ότι έχουμε 4 επιλογές προτού έχουμε επικοινωνία:

1. ΚΑΤΑΧΩΡΗΣΗ E-MAIL ADDRESS:

Πατώντας στο πεδίο αυτό έχουμε τη δυνατότητα να πληκτρολογήσουμε το E-mail Address του ασθενή.

2. ΚΑΤΑΧΩΡΗΣΗ ΘΕΜΑΤΟΣ E-MAIL:

Πατώντας στο πεδίο αυτό μπορούμε να εισάγουμε το θέμα του E-mail που θα αποσταλεί.

3. ΚΑΤΑΧΩΡΗΣΗ ΚΕΙΜΕΝΟΥ E-MAIL:

Πατώντας στο πεδίο αυτό έχουμε τη δυνατότητα να γράψουμε τις διαγνώσεις ή ό,τι άλλο επιθυμούμε για να το αποστείλουμε στον ασθενή.

4. ΕΠΙΛΟΓΗ ΑΠΟΣΤΟΛΗΣ E-MAIL:

Πατώντας στο κουμπί αυτό κάνουμε προώθηση του E-mail που συντάξαμε στον ασθενή για την έναρξη της επικοινωνίας.

Παράρτημα Β

Εγχειρίδιο χρήσης Τεχνικού της Πληροφορικής

➤ Εγχειρίδιο Χρήσης Front End

Κλάση Language Activity

Function Display Metrics:

```
DisplayMetrics metrics = this.getResources().getDisplayMetrics();  
switch (metrics.densityDpi) {  
case DisplayMetrics.DENSITY_LOW:  
title.setTextSize(15);  
about.setLayoutParams(new LinearLayout.LayoutParams(32, 32));  
  
break;  
case DisplayMetrics.DENSITY_MEDIUM:  
title.setTextSize(20);  
about.setLayoutParams(new LinearLayout.LayoutParams(45, 45));  
  
break;  
case DisplayMetrics.DENSITY_HIGH:  
title.setTextSize(25);  
about.setLayoutParams(new LinearLayout.LayoutParams(70, 70));  
  
break;  
case DisplayMetrics.DENSITY_XHIGH:  
title.setTextSize(30);  
about.setLayoutParams(new LinearLayout.LayoutParams(100, 100));  
  
break;  
}
```

Η συνάρτηση αυτή προσαρμόζει τα αντικείμενα του γραφικού περιβάλλοντος σε διαφορετικές οθόνες ανάλογα με την πυκνότητα της κάθε οθόνης, δηλαδή 720ρ, 1080ρ κτλ. Έτσι, κάθε περιβάλλον στην οθόνη θα εφαρμόζεται σωστά και ανάλογα.

Παρ' όλα αυτά, θα μπορούσε η συνάρτηση αυτή να πραγματοποιηθεί με μετρικές που αφορούν τις διαστάσεις της οθόνης (ύψος και πλάτος), παρά την πυκνότητα της, αφού υπάρχουν κινητές συσκευές με μεγάλη πυκνότητα αλλά χαμηλότερες διαστάσεις οθόνης. Άρα κρίνεται σημαντικό η μετατροπή των μετρικών από πυκνότητα σε ύψος ή πλάτος οθόνης (Προτιμητέο το ύψος οθόνης (height) , επειδή η εφαρμογή βρίσκεται σε portrait mode τις περισσότερες φορές).

Σημείωση: Αυτό μπορεί να πραγματοποιηθεί για κάθε συνάρτηση της κάθε κλάσης, ώστε να έχουμε ομοιομορφία στην στοίχιση των αντικειμένων διάδρασης στον χώρο της οθόνης.

Κλάση Login Activity

```
private static final String SERVICE_URL = "http://10.16.4.119:8080/ECG/SelectLogin";
private static final String SERVICE_INSERT = "http://10.16.4.119:8080/ECG/InsertLogin";
private static final String SERVICE_UPDATE = "http://10.16.4.119:8080/ECG/UpdateLogin";
```

Function Find Identical Username & Password:

```
private class WebServiceTask extends AsyncTask<String, Integer, String> {
```

```
    // connection timeout, in milliseconds (waiting to connect)
```

```
    private static final int CONN_TIMEOUT = 5000;
```

```
    // socket timeout, in milliseconds (waiting for data)
```

```
    private static final int SOCKET_TIMEOUT = 5000;
```

```
    protected String doInBackground(String... urls) {
```

```
        String result = "null";
```

```
        HttpResponse response = null;
```

```
        try {
```

```
            response = doResponse(SERVICE_URL);
```

```
        } catch (JSONException e1) {
```

```
            // TODO Auto-generated catch block
```

```
            e1.printStackTrace();
```

```
        }
```

```
        if (response == null) {
```

```
            Log.d("result1", result);
```

```
            return result;
```

```
        } else {
```

```
        try {
```

```
            result = JsonAnswer;
```

```
            String getResponse = result.toString();
```

```
            Log.d("response", getResponse);
```

```
        } catch (IllegalStateException e) {
```

```
            // TODO Auto-generated catch block
```

```
            e.printStackTrace();
```

```
        }
```

```
        return result;
```

```
    }
```

```
}
```

```
@Override
```

```
protected void onPostExecute(String result) {
```

```

InsertDataLogin webInsert = new InsertDataLogin();
webInsert.execute(new String[] { SERVICE_INSERT });

}

```

Αυτή η ασύγχρονη διεργασία (Async Task) χρησιμοποιείται στο να γίνεται σωστά η ταυτοποίηση του username και password του χρήστη, με το να στέλνονται αιτήματα της μορφής JSON στον server με την ανάλογη επεξεργασία. Στον πιο πάνω κώδικα, στέλνομαι το αίτημα JSON, εάν το username και password του χρήστη υπάρχουν στην βάση δεδομένων και είναι μοναδικά, και εάν υπάρχουν τότε επιστρέφεται ο μοναδικός αριθμός ταυτότητας του χρήστη. Εάν όχι, τότε επιστρέφουμε μήνυμα που δεν αφήνει τον χρήστη να εξουσιοδοτηθεί για είσοδο στο σύστημα.

Οι επεκτάσεις που μπορούν να γίνουν είναι η ενσωμάτωση της πλατφόρμας Fi-Star στο τμήμα αποστολής αιτημάτων JSON, καθώς και η αναγνώριση όλων των χρηστών σε περίπτωση που το username ή password δεν είναι μοναδικά, δηλαδή να βγάξει λίστα με ταυτότητες χρηστών που στην χειρίστη περίπτωση μπορούν να έχουν τα ίδια στοιχεία ταυτοποίησης.

Function Update Data to Database:

```

private class UpdateTaskLogin extends AsyncTask<String, Integer, String> {

    // connection timeout, in milliseconds (waiting to connect)
    private static final int CONN_TIMEOUT = 5000;

    // socket timeout, in milliseconds (waiting for data)
    private static final int SOCKET_TIMEOUT = 5000;

    protected String doInBackground(String... urls) {
        String result = "null";
        HttpResponse response = null;

        JsonUpdate = new JSONObject();
        try {
            JsonUpdate.put("registeruserUserId", idJSON);
            JsonUpdate.put("startSession", datetime);
            JsonUpdate.put("endSession", Newdatetime);
            JsonUpdate.put("state", "Log Out");
        } catch (JSONException e2) {
            // TODO Auto-generated catch block
            e2.printStackTrace();
        }

        try {
            response = doResponse(SERVICE_UPDATE);
        } catch (JSONException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
    }
}

```



```

    }

    if (response == null) {
        Log.d("result1", result);
        return result;
    } else {

        try {

            result = JsonAnswer;
            String getResponse = result.toString();

            Log.d("response", getResponse);

        } catch (IllegalStateException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return result;
    }
}

```

Αυτή η ασύγχρονη διεργασία (Async Task) χρησιμοποιείται στην ενημέρωση των πεδίων end_session και state, αφού ο χρήστης πραγματοποιήσει πρώτα αποσύνδεση από το σύστημα. Δηλαδή με την αποσύνδεση, θα ενημερώνεται η ώρα και ημερομηνία που ο χρήστης εξήλθε από το σύστημα, καθώς και ότι ο χρήστης όντως εξήλθε με τον σωστό τρόπο (κατάσταση Log Out) .

Επεκτάσεις που μπορούν να γίνουν είναι στην ενημέρωση περαιτέρω πεδίων, με παρόμοιο τρόπο, όπως στην δραστηριότητα Εγγραφής του χρήστη και σε άλλες βιομετρήσεις.

Function Insert Data to Database:

```

private class InsertDataLogin extends AsyncTask<String, Integer, String> {

    // connection timeout, in milliseconds (waiting to connect)
    private static final int CONN_TIMEOUT = 5000;

    // socket timeout, in milliseconds (waiting for data)
    private static final int SOCKET_TIMEOUT = 5000;

    protected String doInBackground(String... urls) {
        String result = "null";
        HttpResponse response = null;

        JsonInsert = new JSONObject();
        idJSON = null;
    }
}

```

```

if (JsonAnswer.equals("") == false) {
    String temp[] = JsonAnswer.split(":");
    String temp1[] = temp[2].split("]");
    idJSON = temp1[0].replace("}", "");
}

try {
    datetime = (String) DateFormat.format("yyyy-MM-dd
HH:mm:ss",
        new java.util.Date());
    JsonInsert.put("registeruserUserId", idJSON);
    JsonInsert.put("startSession", datetime);
    JsonInsert.put("endSession", datetime);
    JsonInsert.put("state", "Sign In");
} catch (JSONException e2) {
    // TODO Auto-generated catch block
    e2.printStackTrace();
}

try {
    response = doResponse(SERVICE_INSERT);
} catch (JSONException e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
}

if (response == null) {
    Log.d("result1", result);
    return result;
} else {

    try {

        result = JsonAnswer;
        String getResponse = result.toString();

        Log.d("response", getResponse);

    } catch (IllegalStateException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return result;
}
}

```

Αυτή η ασύγχρονη διεργασία (Async Task) χρησιμοποιείται στην εισαγωγή των δεδομένων εισόδου στο σύστημα, δηλαδή ότι η σύνδεση του χρήστη ήταν επιτυχημένη. Τα δεδομένα που εισάγονται στην βάση είναι ο αριθμός ταυτότητας, ώρα και ημερομηνία εισόδου και εξόδου (στην αρχή οι 2 τιμές είναι οι ίδιες, ωστόσο ο χρήστης πραγματοποιήσει αποσύνδεση από το σύστημα) και το πεδίο Κατάστασης, δηλαδή εάν ο χρήστης συνδέθηκε με το σύστημα (Log In) ή αποσυνδέθηκε (Log Out).

Επεκτάσεις που μπορούν να γίνουν είναι ως προς την χρήση των IP numbers. Για κάθε φορά που τρέχουμε το πρόγραμμα, λόγω αλλαγής περιβάλλοντος και διαφορετικής σύνδεσης Wi-Fi, θα πρέπει να αλλάζουμε την διεύθυνση IP με την καινούργια, πράγμα χρονοβόρο. Άρα θα ήταν καλύτερα, αυτό να γίνεται με αυτοματοποιημένο τρόπο ή απλά ο server και η βάση δεδομένων να βρίσκονται σε ένα σταθερό σημείο πρόσβασης (γι' αυτό κρίνεται σημαντική η χρήση της πλατφόρμας Fi-Star).

Κλάση Bluetooth Shimmer

```
private static Button connect, disconnect, collect, stopcollect, showgraph;
private BluetoothSocket mmSocket;
private BluetoothDevice mmDevice;
private static ListView lv;
private BluetoothAdapter BA;
private Set<BluetoothDevice> pairedDevices;
private static ToggleButton check, logfile;
private static ArrayAdapter<String> adapter;
private static String devicename = null, titlelogging = null, mBluetoothAddress = null;
private ArrayList<String> devices;
private static View viewall;
private TextView title;
private ImageButton information;
private static String mFileName = "Default", IDuser = null;
static Logging log = new Logging(mFileName, "\t"); // insert file name
private static boolean mEnableLogging = true;
static Context context;
private final static int interval = 7500, timeinterval2 = 1000;
private static boolean exitenabler = true;
private ShowcaseView sv;
private int counter = 0;
```

Function Get Item from List

```
lv.setOnItemClickListener(new OnItemClickListener() {

    @Override
    public void onItemClick(AdapterView<?> parent, View view,
        int position, long id) {
        // TODO Auto-generated method stub
        devicename = (String) devices.get(position);
```

```

vibe.vibrate(20);

if (language == 1)
    Toast.makeText(getApplicationContext(),
        "Bluetooth Device Address: " + devicename,
        Toast.LENGTH_SHORT).show();

viewall = view;

    }
});

```

Η συνάρτηση αυτή παίρνει το όνομα της συσκευής σύζευξης από την λίστα συσκευών, εμφανίζοντας κατάλληλο μήνυμα.

Function Connect Shimmer

```

/**
 * Connection button
 */
connect.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        vibe.vibrate(20);
        if (devicename != null && BA.isEnabled()) {
            Thread connectionThread = new Thread(new Runnable() {

                @Override
                public void run() {

                    BA.cancelDiscovery();
                    mmDevice = BA.getRemoteDevice(devicename);
                    // try {
                    Method m = null;
                    try {
                        m = mmDevice.getClass().getMethod(
                            "createRfcommSocket",
                            new Class[] { int.class });
                    } catch (NoSuchMethodException e) {
                        // TODO Auto-generated catch
                        // block
                        e.printStackTrace();
                    }
                    try {
                        mmSocket = (BluetoothSocket) m.invoke(mmDevice,
                            Integer.valueOf(1));
                    } catch (IllegalAccessException e) {

```

```

        // TODO Auto-generated catch
        // block
        e.printStackTrace();
    } catch (IllegalArgumentException e) {
        // TODO Auto-generated catch
        // block
        e.printStackTrace();
    } catch (InvocationTargetException e) {
        // TODO Auto-generated catch
        // block
        e.printStackTrace();
    }
    mService.connectShimmer(devicename, "Device");
    mBluetoothAddress = devicename;
    mService.setGraphHandler(mHandler);
}

});

        connectionThread.start();

    } else {
        Toast.makeText(getApplicationContext(),
            "Please choose a bluetooth device",
            Toast.LENGTH_SHORT).show();

        }

    }

});

```

Το κουμπί Σύνδεσης ενώνει την κινητή συσκευή με την συσκευή Shimmer, μέσω τεχνολογίας Bluetooth. Τα μαυρισμένα κομμάτια κώδικα (bold) είναι αυτά που πραγματοποιούν την σύνδεση. Και όλα αυτά τρέχουν με την χρήση ενός thread.

Function Collect Data from Shimmer

```

collect.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        vibe.vibrate(20);
        mService.startStreaming(devicename);
        log = new Logging(mFileName, "\t");
        Log.d("StartStreaming", "stream");
    }
});

```

```
    }
});
```

Αυτή η συνάρτηση συλλέγει τα δεδομένα από την συσκευή ηλεκτροκαρδιογραφήματος της Shimmer (και μπορεί να τα αποθηκεύσει επίσης).

Function Stop Data Collection from Shimmer

```
stopcollect.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        vibe.vibrate(20);
```

```
        mService.stopStreaming(mBluetoothAddress);
        Log.d("StartStreaming", "no streaming");
```

```
    }
});
```

Αυτή η συνάρτηση τερματίζει την συλλογή δεδομένων από την συσκευή ηλεκτροκαρδιογραφήματος της Shimmer.

Function Disconnection from Shimmer

```
disconnect.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        vibe.vibrate(20);
        if (mmSocket != null && BA.isEnabled()) {
```

```
            mService.disconnectAllDevices();
            logfile.setEnabled(true);
            devicename = null;
            exitenabler = true;
```

```
        /**
```

```
         * Update the gui of screen
```

```
        */
```

```
        Thread t = new Thread(new Runnable() {
            @Override
            public void run() {
                // TODO Auto-generated method stub
                runOnUiThread(new Runnable() {
```

```

        @Override
        public void run() {
            lv.setEnabled(true);
            viewall.setBackgrou
            ndResource(R.drawable.red
            );

            adapter.notifyDataSetChanged();
        }
    });
    t.start();

```

Αυτή η συνάρτηση αποσυνδέει την συσκευή Shimmer από την κινητή συσκευή (bold line), με τις απαραίτητες μετατροπές στην παρουσίαση του γραφικού περιβάλλοντος.

Function Save File

```

/**
 * Toggle Button Save file
 * @param view
 */
public void onToggleLogging(View view) {
    if (((ToggleButton) view).isChecked() == false) {
        mEnableLogging = true;
        mService.setEnableLogging(mEnableLogging);
        mService.setLoggingName(mFileName);

        if (language == 1) {
            titlelogging = "Logging Enabled";

        } else {
            titlelogging = "Αποθήκευση";
            Toast.makeText(getApplicationContext(), "Επιλογή
Αποθήκευσης",
                                Toast.LENGTH_SHORT).show();
        }

    } else {
        mEnableLogging = false;
        mService.setEnableLogging(mEnableLogging);
        // mService.setLoggingName(mFileName);
        if (language == 1)
            titlelogging = "Logging Disabled";
        else {
            titlelogging = "Μη-Αποθήκευση";
            Toast.makeText(getApplicationContext(),

```

```

Toast.LENGTH_SHORT).show();
    }
}

protected ServiceConnection mTestServiceConnection = new
ServiceConnection() {

    public void onServiceConnected(ComponentName arg0, IBinder
service) {

        // TODO Auto-generated method stub
        Log.d("ShimmerService", "service connected from main
activity");

        LocalBinder binder = (LocalBinder) service;
        mService = binder.getService();
        mServiceBind = true;
        mService.setGraphHandler(mHandler);

    }

    public void onServiceDisconnected(ComponentName arg0) {
        // TODO Auto-generated method stub
        mServiceBind = false;
    }

};

```

Η συνάρτηση αυτή πραγματοποιεί την αποθήκευση των δεδομένων του ηλεκτροκαρδιογραφήματος στην κάρτα επέκτασης μνήμης της κινητής συσκευής.

Function Handler States

```

private static Handler mHandler = new Handler() {

    public void handleMessage(Message msg) {
        switch (msg.what) {

            case Shimmer.MESSAGE_STATE_CHANGE:
                switch (msg.arg1) {
                    case Shimmer.STATE_CONNECTED:
                        handchangecolor.postDelayed(runnable, interval);
                        exitenabler = false;
                        lv.setEnabled(false);
                        break;
                    case Shimmer.MSG_STATE_FULLY_INITIALIZED:
                        Log.d("ShimmerActivity",
                            "Message Fully Initialized
                            Received from Shimmer driver");

```



```

        mBluetoothAddress = ((ObjectCluster)
        msg.obj).mBluetoothAddress;
        mService.enableGraphingHandler(true);
        break;
    case Shimmer.STATE_CONNECTING:
        Log.d("ShimmerActivity",
            "Driver is attempting to establish
            connection with Shimmer device");
        lv.setEnabled(false);

viewall.setBackgroundResource(R.drawable.yellow);
        adapter.notifyDataSetChanged();
        break;
    case Shimmer.STATE_NONE:
        Log.d("ShimmerActivity", "Shimmer No State");
        mBluetoothAddress = null;
        exitenabler = true;
        lv.setEnabled(true);
        collect.setEnabled(false);
        collect.setBackgroundResource(R.drawable.disable_colour);
        viewall.setBackgroundResource(R.drawable.red);
        adapter.notifyDataSetChanged();
        disconnecthandler.postDelayed(runDisconnect, timeinterval2);
    }
    break;
    case Shimmer.MESSAGE_READ:

        break;
    case Shimmer.MESSAGE_ACK_RECEIVED:

        break;
    case Shimmer.MESSAGE_DEVICE_NAME:
        Toast.makeText(BluetoothActivity.context,
            "Connected to " + mBluetoothAddress,
Toast.LENGTH_SHORT)
            .show();
        break;

    case Shimmer.MESSAGE_TOAST:
        Toast.makeText(BluetoothActivity.context,

msg.getData().getString(Shimmer.TOAST),
            Toast.LENGTH_SHORT).show();
        break;

    }
}
};

```

Αυτός ο χειρίστης καταστάσεων δείχνει επακριβώς σε ποιες καταστάσεις μπορεί να βρεθεί η συσκευή (bold lines), δίνοντας το απαραίτητο γραφικό περιβάλλον και μηνύματα. Επεκτάσεις μπορούν να γίνουν ανάλογα με τις επιθυμίες του προγραμματιστή και ανάλογα με την κατάσταση που επιθυμεί.

Function Show Electrocardiograph

```
private static Handler mHandler = new Handler() {

    public void handleMessage(Message msg) {
        switch (msg.what) {

            case Shimmer.MESSAGE_STATE_CHANGE:
                switch (msg.arg1) {
                    case Shimmer.STATE_CONNECTED:
                        break;
                    case Shimmer.MSG_STATE_FULLY_INITIALIZED:
                        break;
                    case Shimmer.STATE_CONNECTING:
                        break;
                    case Shimmer.STATE_NONE:
                }
                break;
            case Shimmer.MESSAGE_READ:

                if ((msg.obj instanceof ObjectCluster)) {
                    ObjectCluster objectCluster =
                        (ObjectCluster) msg.obj;

                    calibrateddataArray = new double[0];
                    String[] sensorName = new String[0];
                    String calibratedUnits = "";
                    String calibratedUnits2 = "";
                    // mSensorView determines which sensor to graph

                    sensorName = new String[2];
                    calibrateddataArray = new double[2];
                    sensorName[0] = "ECG RA-LL";
                    sensorName[1] = "ECG LA-LL";

                    if (sensorName.length != 0) { // Device 1 is
                        the assigned
                        // user id, see constructor
                        // of the Shimmer
                        if (sensorName.length > 0) {
                            //
                            Collection<FormatCluster> ofFormats =
                                objectCluster.mPropertyCluster.get(sensorName[0]);
```

```

        FormatCluster formatCluster = ((FormatCluster) ObjectCluster
        .returnFormatCluster(ofFormats, "CAL"));
        if (formatCluster != null) {
            // //Obtain data for text view
            calibratedDataArray[0] = formatCluster.mData;
            calibratedUnits = formatCluster.mUnits;

            // first check if there is data
            List<Number> data;
            if (mPlotDataMap.get("serie 1") != null) {
                data = mPlotDataMap.get("serie 1");
            } else {
                data = new ArrayList<Number>();
            }
            if (data.size() > X_AXIS_LENGTH) {
                data.clear();
            }
            data.add(formatCluster.mData);
            mPlotDataMap.put("serie 1", data);

            // next check if the series exist
            LineAndPointFormatter lapf = new LineAndPointFormatter(
            Color.rgb(0, 255, 0), null, null);
            if (mPlotSeriesMap.get("serie 1") != null) {
                // if the series exist get the line format
                mPlotSeriesMap.get("serie 1").updateData(
                data);
            } else {
                XYSeriesShimmer series = new XYSeriesShimmer(
                data, 0, "serie 1");
                mPlotSeriesMap.put("serie 1", series);
                dynamicPlot
                .addSeries(mPlotSeriesMap
                .get("serie 1"), lapf);
            }
        }
        //
    }
}

if (sensorName.length > 1) {
    Collection<FormatCluster> ofFormats = objectCluster.mPropertyCluster
    .get(sensorName[1]); // first retrieve all the
    // possible formats for
    // the current sensor
    // device

```

```

FormatCluster formatCluster = ((FormatCluster) ObjectCluster
    .returnFormatCluster(ofFormats, "CAL"));
if (formatCluster != null) {
    calibratedDataArray[1] = formatCluster.mData;
    // Obtain data for text view
    calibratedUnits2 = formatCluster.mUnits;

    List<Number> data;
    if (mPlotDataMap.get("serie 2") != null) {
        data = mPlotDataMap.get("serie 2");
    } else {
        data = new ArrayList<Number>();
    }
    if (data.size() > X_AXIS_LENGTH) {
        data.clear();
    }
    data.add(formatCluster.mData);
    mPlotDataMap.put("serie 2", data);

    // next check if the series exist
    LineAndPointFormatter lapf = new LineAndPointFormatter(
        Color.rgb(255, 255, 0), null, null);
    if (mPlotSeriesMap.get("serie 2") != null) {
        // if the series exist get the line format
        mPlotSeriesMap.get("serie 2").updateData(data);
    } else {
        XYSeriesShimmer series = new XYSeriesShimmer(
            data, 0, "serie 2");
        mPlotSeriesMap.put("serie 2", series);
        dynamicPlot.addSeries(
            mPlotSeriesMap.get("serie 2"), lapf);
    }

}

}

}

if (sensorName.length > 2) {

    Collection<FormatCluster> ofFormats = objectCluster.mPropertyCluster
        .get(sensorName[2]); // first retrieve all the
    // possible formats for
    // the current sensor
    // device
    FormatCluster formatCluster = ((FormatCluster) ObjectCluster
        .returnFormatCluster(ofFormats, "CAL"));
    if (formatCluster != null) {
        calibratedDataArray[2] = formatCluster.mData;
    }
}

```

```

List<Number> data;
if (mPlotDataMap.get("serie 3") != null) {
data = mPlotDataMap.get("serie 3");
} else {
data = new ArrayList<Number>();
}
if (data.size() > X_AXIS_LENGTH) {
data.clear();
}
data.add(formatCluster.mData);
mPlotDataMap.put("serie 3", data);

// next check if the series exist
LineAndPointFormatter lapf = new LineAndPointFormatter(
Color.rgb(255, 0, 0), null, null);
if (mPlotSeriesMap.get("serie 3") != null) {
// if the series exist get the line format
mPlotSeriesMap.get("serie 3").updateData(data);
} else {
XYSeriesShimmer series = new XYSeriesShimmer(
data, 0, "serie 3");
mPlotSeriesMap.put("serie 3", series);
dynamicPlot.addSeries(
mPlotSeriesMap.get("serie 3"), lapf);

}
}

}
dynamicPlot.redraw();

if (calibratedDataArray.length > 0) {
mValueSensor1.setText(String.format("%.4f",
calibratedDataArray[0]));
mTextSensor1.setText(sensorName[0] + "("
+ calibratedUnits + ")");
}
if (calibratedDataArray.length > 1) {
mValueSensor2.setText(String.format("%.4f",
calibratedDataArray[1]));
mTextSensor2.setText(sensorName[1] + "("
+ calibratedUnits2 + ")");
}

datetime = (String) DateFormat.format(
"yyyy-MM-dd HH:mm:ss", new java.util.Date());

ShimmerWebTask wst = new ShimmerWebTask();
wst.execute(new String[] { SERVICE_SHIMMER });

```

```

}

break;
case Shimmer.MESSAGE_ACK_RECEIVED:

break;
case Shimmer.MESSAGE_DEVICE_NAME:
// save the connected device's name

break;

case Shimmer.MESSAGE_TOAST:

break;

}
}
};

```

Η συνάρτηση αυτή λαμβάνει τα δεδομένα από την συσκευή Shimmer και τα επεξεργάζεται με τέτοιο τρόπο, ώστε να έχουμε συνεχείς ενημέρωση των τιμών δηλαδή να εμφανίζονται οι καταγεγραμμένες τιμές ανά δευτερόλεπτο και έπειτα παρουσιάζεται και η γραφική παράσταση του ηλεκτροκαρδιογραφήματος, ανάλογα με τις 2 τιμές επαγωγών που εμφανίζονται. Έπειτα, οι τιμές αποθηκεύονται αυτόματα στην βάση δεδομένων.

Function Insert Electrocardiograph Data to database

```

private static class ShimmerWebTask extends
    AsyncTask<String, Integer, String> {

    // connection timeout, in milliseconds (waiting to connect)
    private static final int CONN_TIMEOUT = 5000;

    // socket timeout, in milliseconds (waiting for data)
    private static final int SOCKET_TIMEOUT = 5000;

    protected String doInBackground(String... urls) {
        String result = "null";
        HttpResponse response = null;

        JsonInsertShimmer = new JSONObject();
        try {
            JsonInsertShimmer.put("registeruserUserId",
idUser);
JsonInsertShimmer.put("dateTaken", datetime);
JsonInsertShimmer.put("ecg1",

```

```

Float.parseFloat(calibratedDataArray[0] + "" + "");
    JsonInsertShimmer.put("ecg2",

Float.parseFloat(calibratedDataArray[1] + "" + "");
    } catch (JSONException e2) {
        // TODO Auto-generated catch block
        e2.printStackTrace();
    }

    try {
        response = doResponse(SERVICE_SHIMMER);
    } catch (JSONException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }

    if (response == null) {
        Log.d("result1", result);
        return result;
    } else {

        try {

            result = JsonAnswer;
            String getResponse = result.toString();

            Log.d("response", getResponse);

        } catch (IllegalStateException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return result;
    }
}

```

Αυτή η ασύγχρονη διεργασία (Async Task) χρησιμοποιείται στην εισαγωγή των δεδομένων ηλεκτροκαρδιογραφήματος Shimmer στην βάση δεδομένων. Τα δεδομένα που εισάγονται στην βάση είναι ο αριθμός ταυτότητας, ώρα και ημερομηνία εισόδου και το 2 πεδία τιμών επαγωγών

Επεκτάσεις που μπορούν να γίνουν είναι ως προς την χρήση των IP numbers. Για κάθε φορά που τρέχουμε το πρόγραμμα, λόγω αλλαγής περιβάλλοντος και διαφορετικής σύνδεσης Wi-Fi, θα πρέπει να αλλάζουμε την διεύθυνση IP με την καινούργια, πράγμα χρονοβόρο. Άρα θα ήταν καλύτερα, αυτό να γίνεται με αυτοματοποιημένο τρόπο ή απλά ο server και η βάση δεδομένων να βρίσκονται σε ένα σταθερό σημείο πρόσβασης (γι' αυτό κρίνεται σημαντική η χρήση της πλατφόρμας Fi-Star).

Function Arduino Streaming

```
public void mainThread() {
    Thread t = new Thread(new Runnable() {
        public void run() {
            flag = true;
            int randomvalue = 0;
            try {
                while (flag == true) {
                    try {
localDatagramPacket = new DatagramPacket(new byte[1024], 1024);
socket.receive(localDatagramPacket);
arrayOfString = new String(localDatagramPacket.getData()).split("#");
                        Random pick = new Random();
                        randomvalue = pick.nextInt(15);

                        if (arrayOfString.length < 10) {
                            Log.d("error streaming", arrayOfString.length
                                + "");

                            arrayOfString = new String[] { "20", "0",
                                "127", "70", "0", "37", "75", "98",
                                "2", "0", "20", "0", "50", "0" };
                        }

                        else {
                            arrayOfString[2] = systolicarray[randomvalue]
                                + "";
                            arrayOfString[3] = diastolicarray[randomvalue]
                                + "";
                        }
                    }
                }
            } catch (Exception e) {
                Log.e("Error", e.getMessage());
            }
        }
    });
    t.start();
}
```

Η συνάρτηση αυτή λαμβάνει μέσω της σύνδεσης Wi-Fi hotspot, τα δεδομένα λήψης από την πλατφόρμα Arduino. Με βάση τα **bold lines** παρατηρούμε πως τα δεδομένα έρχονται σε πακέτο (τύπου Datagram Packet), έπειτα ανοίγουμε μια σύνδεση (socket) με βάση τον αριθμό θύρας (port) της συσκευής.

Έπειτα, γίνεται το απαραίτητο parsing (αφού τα δεδομένα έρχονται με την μορφή String) και εισάγονται σε έναν πίνακα, όπου ανάλογα κάθε πληροφορία επεξεργάζεται.

Επεκτάσεις μπορούν να γίνουν με την προσθήκη περισσότερων δεδομένων αισθητήρων, καθώς και με την προσθήκη της καταμέτρησης της αρτηριακής πίεσης με δεδομένα σε πραγματικό χρόνο.

Κώδικας Arduino

```
#include <PinChangeInt.h>
#include <eHealth.h>

char recv[128];
uint8_t cont = 0;

void setup()
{
  Serial.begin(9600);

  eHealth.initPulsioximeter();
  eHealth.initPositionSensor();

  //Attach the interruptions for using the pulsioximeter.
  PCintPort::attachInterrupt(6, readPulsioximeter, RISING);
  //delay(1000);

}

void loop()
{
  //while (Serial.available()>0) {}
  // Enters in command mode
  Serial.print("$$$"); check();
  // Sets DHCP and TCP protocol
  Serial.print("set ip dhcp 1\r"); check();
  Serial.print("set ip protocol 1\r"); check();
  // Configures the way to join the network AP
  Serial.print("set wlan join 0\r"); check();
  Serial.print("join ANDROID\r"); check();
  Serial.print("set i h 255.255.255.255\r"); delay(1000);
  Serial.print("set i r 12345\r"); check();
  Serial.print("set i l 2000\r"); check();
  Serial.print("exit\r"); check();

  while(1){

    //1. Read from eHealth.
    int airFlow = eHealth.getAirFlow();
    float temperature = eHealth.getTemperature();
```

```

float conductance = eHealth.getSkinConductance();
float resistance = eHealth.getSkinResistance();
float conductanceVol = eHealth.getSkinConductanceVoltage();
int BPM = eHealth.getBPM();
int SPO2 = eHealth.getOxygenSaturation();
uint8_t pos = eHealth.getBodyPosition();
//int syst = eHealth.getSystolicPressure(1);
//int diast = eHealth.getDiastolicPressure(1);
float ECG = eHealth.getECG();
int EMG = eHealth.getEMG();
EMG = map (EMG, 200,300,0,100); // Use this function to adapt your muscle
level to 0-100%
//uint8_t glucose = eHealth.glucoseDataVector[0].glucose;

//Data sensor must be sent in this order to mobile android application
Serial.print(int(airFlow)); Serial.print("#");
Serial.print(ECG); Serial.print("#");
Serial.print(int(0)); Serial.print("#"); //Systolic is not implemented
Serial.print(int(0)); Serial.print("#"); //Diastolic is not implemented
Serial.print(int(0)); Serial.print("#"); //Glucose is not implemented
Serial.print(temperature); Serial.print("#");
Serial.print(int(BPM)); Serial.print("#");
Serial.print(int(SPO2)); Serial.print("#");
Serial.print(conductance); Serial.print("#");
Serial.print(int(resistance)); Serial.print("#");
Serial.print(int(airFlow)); Serial.print("#");
Serial.print(int(pos)); Serial.print("#");
Serial.print(int(EMG)); Serial.print("#");
Serial.print("\n");

// Reduce this delay for more data rate
delay(250);
}
}

void check(){
    cont=0; delay(500);
    while (Serial.available()>0)
    {
        recv[cont]=Serial.read(); delay(10);
        cont++;
    }
    recv[cont]='\0';
    Serial.println(recv);
    Serial.flush(); delay(100);
}

```

Στον κώδικα Arduino σε bold lines, βλέπουμε τις τιμές που μπορούμε να λάβουμε πληροφορίες από τους αισθητήρες. Έπειτα, οι τιμές αποστέλλονται μέσω του Serial.print προς την κινητή συσκευή.

Function Fetch Data from Database

```
private static class FetchDataTask extends
    AsyncTask<String, Integer, String> {

    // connection timeout, in milliseconds (waiting to connect)
    private static final int CONN_TIMEOUT = 5000;

    // socket timeout, in milliseconds (waiting for data)
    private static final int SOCKET_TIMEOUT = 5000;

    protected String doInBackground(String... urls) {
        String result = "null";
        HttpResponse response = null;
        JSONObject getJSONid = new JSONObject();

        jsonFetch = new JSONObject();

        try {

            getJSONid.put("loginRegisteruserUserId", idUser);
            jsonFetch.put("=", getJSONid);
        } catch (JSONException e2) {
            // TODO Auto-generated catch block
            e2.printStackTrace();
        }

        try {
            response =
doResponse(SERVICE_FETCH_DATA);
        } catch (JSONException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }

        if (response == null) {
            Log.d("result1", result);
            return result;
        } else {

            try {

                result = JsonAnswer;
                String getResponse = result.toString();
```

```

        Log.d("response", getResponse);

    } catch (IllegalStateException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return result;
}

}

@Override
protected void onPostExecute(String result) {

    if (result.startsWith("{\"measurements\"")) {

        JSONObject jObject;
        temperatureArray = new ArrayList<Double>();
        airflowArray = new ArrayList<Double>();
        systolicArray = new ArrayList<Double>();
        diastolicArray = new ArrayList<Double>();
        oxygenArray = new ArrayList<Double>();
        pulseArray = new ArrayList<Double>();
        ecgArray = new ArrayList<Double>();
        muscleArray = new ArrayList<Double>();
        conductivityArray = new ArrayList<Double>();
        try {
            jObject = new JSONObject(result);
            JSONArray data =
jObject.getJSONArray("measurements");

            for (int i = 0; i < data.length(); i++) {
                JSONObject rec =
data.getJSONObject(i);

                temperatureArray.add(rec

                .getDouble("temperatureMeanValue"));

                airflowArray.add(rec.getDouble("airflowMeanValue"));
                systolicArray.add(rec

                .getDouble("systolicPressureMeanValue"));
                diastolicArray.add(rec

                .getDouble("diastolicPressureMeanValue"));

                oxygenArray.add(rec.getDouble("bloodOxygenMeanValue"));

```

```

pulseArray.add(rec.getDouble("pulseMeanValues"));

ecgArray.add(rec.getDouble("ecgMeanValue"));
                                muscleArray.add(rec

.getDouble("muscleIntensityMeanValue"));
                                conductivityArray.add(rec

.getDouble("conductivityMeanValue"));
                                }

                                new
ChartTaskShowHistoryMeasurements()

.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR);

                                } catch (JSONException e) {
                                    // TODO Auto-generated catch block
                                    e.printStackTrace();
                                }

                                }

                                else if (result.equals("{}")) {

                                    if (language == 1)
                                        Toast.makeText(
                                            context,
                                            "No data are stored in
database for user "
                                                                    + username,
Toast.LENGTH_LONG).show();

                                    else if (language == 0)
                                        Toast.makeText(
                                            context,
                                            "Δεν υπάρχουν
αποθηκευμένα δεδομένα για τον χρήστη "
                                                                    + username,
Toast.LENGTH_LONG).show();

                                }

```

Αυτή η συνάρτηση μαζεύει τα δεδομένα ιστορικού ενός χρήστη, και ανάλογα με την επιλογή αισθητήρα, παρουσιάζονται στην γραφική παράσταση δεδομένων.

Επεκτάσεις μπορούν να γίνουν ως προς την εισαγωγή χρόνου των δεδομένων, δηλαδή να εμφανίζονται δεδομένα ανάλογα με την χρονική στιγμή που ορίζει ο

χρηστής. Στην διπλωματική μου, εμφανίζονται τα δεδομένα ενός αισθητήρα, αλλά χωρίς να λαμβάνεται υπόψη ο χρόνος (ημερομηνία και ώρα) των δεδομένων και αυτό θα ήταν καλό να συμπεριλήφθη στις μετέπειτα εργασίες.

➤ Εγχειρίδιο Χρήσης Back End

Κλάση Insert Login

```
protected void doPost(HttpServletRequest request,
                        HttpServletResponse response) throws ServletException,
IOException {
    HashMap<String, String> jsonData;
    // If there was a problem parsing the JSON data
    if ((jsonData = parseSTJSON(getJSONString(request), response)) ==
null) {
        return;
    }

    // Open the session
    Session session = HibernateUtil.getSessionFactory().openSession();
    session.beginTransaction();

    // Create new object
    Login log = new Login();
    LoginId logid = new LoginId();

    logid.setStartSession(toDateTime(jsonData.get("startSession")));
    logid.setRegisteruserUserId(toInteger(jsonData
        .get("registeruserUserId")));
    log.setId(logid);

    // Set the values of the new admission
    log.setEndSession(toDateTime(jsonData.get("endSession")));
    log.setState(jsonData.get("state"));

    // Save the new data in the database
    try {
        session.save(log);
        // Commit the transaction
        session.getTransaction().commit();
        session.close();
    } catch (Exception e) {
        session.close();
        setErrorMessage(response, 400, "Bad Request",
            "Could not insert data. Check the data given as
input.");
        return;
    }
}
```

```

        JSONObject jsonObj = new JSONObject();
        jsonObj.put("userId", log.getId().getRegisteruserUserId());
        jsonObj.put("endSession", log.getEndSession());
        jsonObj.put("startSession", log.getId().getStartSession());
        // Set the success message with the results
        setSuccessMessage(response, jsonObj);
    }

```

Αυτή η κλάση προσθέτει την ταυτότητα χρήστη, ημερομηνία ώρα εισόδου /εξόδου καθώς και την κατάσταση χρήστη (Σύνδεση ή Αποσύνδεση). Τα αιτήματα σε μορφή JSON αναλύονται συντακτικά και έπειτα στέλνονται στην βάση δεδομένων για αποθήκευση.

Κλάση Insert Register

```

protected void doPost(HttpServletRequest request,
                        HttpServletResponse response) throws ServletException,
                        IOException {
    HashMap<String, String> jsonData;
    // If there was a problem parsing the JSON data
    if ((jsonData = parseSTJSON(getJSONString(request), response)) ==
null) {
        return;
    }

    Session session = HibernateUtil.getSessionFactory().openSession();
    session.beginTransaction();

    // Create new object
    Registeruser register = new Registeruser();

    // Set the values of the new admission

    register.setCity(jsonData.get("city"));
    register.setName(jsonData.get("name"));
    register.setSurname(jsonData.get("surname"));
    register.setGender(jsonData.get("gender"));
    register.setStreet(jsonData.get("street"));
    register.setCountry(jsonData.get("country"));
    register.setEmail(jsonData.get("email"));
    register.setDateOfBirth(toDate(jsonData.get("dateOfBirth")));
    register.setUserId(toInteger(jsonData.get("userId")));
    register.setNumberOfStreet(toInteger(jsonData.get("numberOfStreet")));
    register.setPostalCode(toInteger(jsonData.get("postalCode")));

    register.setTelephoneNumber(toInteger(jsonData.get("telephoneNumber")));
    register.setUsername(jsonData.get("username"));

```

```

register.setUserPassword(jsonData.get("password"));

// Save the new data in the database
try {
    session.save(register);
    // Commit the transaction
    session.getTransaction().commit();
    session.close();
} catch (Exception e) {
    session.close();
    setErrorMessage(response, 400, "Bad Request",
        "Could not insert data. Check the data given as
input.");
    return;
}

JSONObject jsonObj = new JSONObject();
jsonObj.put("registerUser", register.getUserId());
// Set the success message with the results
setSuccessMessage(response, jsonObj);
}

```

Αυτή η κλάση προσθέτει τα πεδία προσωπικών πληροφοριών του ασθενή/χρήστη, με βάση τα αιτήματα σε μορφή JSON.

Κλάση Insert Shimmer

```

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    HashMap<String, String> jsonData;
    // If there was a problem parsing the JSON data
    if((jsonData = parseSTJSON(getJSONString(request), response)) ==
null){
        return;
    }

    //Open the session
    Session session = HibernateUtil.getSessionFactory().openSession();
    session.beginTransaction();

    //Create new object
    Shimmerecg shimmer = new Shimmerecg();

    ShimmerecgId logid = new ShimmerecgId();
    logid.setDateTaken(toDateTime(jsonData.get("dateTaken")));
    logid.setLoginRegisteruserUserId(toInteger(jsonData.get("registeruserUserId")));

    shimmer.setId(logid);

```



```

//Set the values of the new admission
// shimmer.setLoginId(toInteger(jsonData.get("loginId")));
shimmer.setEcg1(toFloat(jsonData.get("ecg1")));
shimmer.setEcg2(toFloat(jsonData.get("ecg2")));

//Save the new data in the database
try{
    session.save(shimmer);
    //Commit the transaction
    session.getTransaction().commit();
    session.close();
}catch(Exception e){
    session.close();
    setErrorMessage(response, 400, "Bad Request", "Could not insert data. Check
the data given as input.");
    return;
}

```

```

JSONObject jsonObj = new JSONObject();
jsonObj.put("registeruserUserId",
shimmer.getId().getLoginRegisteruserUserId());
jsonObj.put("dateTaken", shimmer.getId().getDateTaken());
//Set the success message with the results
setSuccessMessage(response,jsonObj);
}

```

Αυτή η κλάση προσθέτει τα πεδία καταγραφής ηλεκτροκαρδιογραφήματος από την συσκευή της Shimmer, με βάση τα αιτήματα σε μορφή JSON.

Κλάση Insert Measurementsperminute

```

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    HashMap<String, String> jsonData;
    // If there was a problem parsing the JSON data
    if((jsonData = parseSTJSON(getJSONString(request), response)) ==
null){
        return;
    }
}

```

```

//Open the session
Session session = HibernateUtil.getSessionFactory().openSession();
session.beginTransaction();

//Create new object

```

```

        Measurementsperminute log = new Measurementsperminute();
        MeasurementsperminuteId logid = new MeasurementsperminuteId();
        logid.setDateTaken(toDateTime(jsonData.get("dateTaken")));

logid.setLoginRegisteruserUserId(toInteger(jsonData.get("loginRegisteruserUserId")));

        log.setId(logid);
        //Set the values of the new admission
        //log.setLoginId(toInteger(jsonData.get("loginId")));
        log.setTemperatureMeanValue(toFloat(jsonData.get("temperatureMeanValue")));

log.setBloodOxygenMeanValue(toFloat(jsonData.get("bloodOxygenMeanValue")));
        log.setAirflowMeanValue(toFloat(jsonData.get("airflowMeanValue")));

log.setSystolicPressureMeanValue(toFloat(jsonData.get("systolicPressureMeanValue")))
);

log.setDiastolicPressureMeanValue(toFloat(jsonData.get("diastolicPressureMeanValue"
)));

log.setMuscleIntensityMeanValue(toFloat(jsonData.get("muscleIntensityMeanValue")))
;
        log.setEcgMeanValue(toFloat(jsonData.get("ecgMeanValue")));
        log.setConductivityMeanValue(toFloat(jsonData.get("conductivityMeanValue")));
        log.setPulseMeanValues(toFloat(jsonData.get("pulseMeanValues")));

        //Save the new data in the database
        try{
            session.save(log);
            //Commit the transaction
            session.getTransaction().commit();
            session.close();
        }catch(Exception e){
            session.close();
            setErrorMessage(response, 400, "Bad Request", "Could not insert data. Check
the data given as input.");
            return;
        }

        JSONObject jsonObj = new JSONObject();
        jsonObj.put("registeruserUserId",
log.getId().getLoginRegisteruserUserId());
        jsonObj.put("dateTaken", log.getId().getDateTaken());
        //Set the success message with the results
        setSuccessMessage(response,jsonObj);
    }

```

Κλάση Select User by Password / Username

```
// Create the string of the query
String queryStr = "FROM " + fromTable + " t WHERE";
boolean firstKey = true;
for (String keyOp : whereFields.keySet()) {
    for (String key : whereFields.get(keyOp).keySet()) {
        String whereStatement;
        // set the correct path for the key
        if (key.equals("username") &&
key.equals("userPassword")) {
            whereStatement = " t.registeruser.userID " +
keyOp + " "
                                + "?";
        } else {
            whereStatement = " t." + key + " " + keyOp + " " +
"?";
        }
        // if this is the first
        if (firstKey) {
            queryStr += whereStatement;
            firstKey = false;
        } else {
            queryStr += " AND" + whereStatement;
        }
    }
}

Session session = HibernateUtil.getSessionFactory().openSession();
// Create the query
Query query = null;
try {
    query = createSelectQuery(session, Registeruser.class.getName(),
                                jsonData);
} catch (Exception e) {
    session.close();
    setErrorMessage(response, 400, "Bad Request",
        "Could not select data. Check the data given as
input.");
}

// Get the results
@SuppressWarnings("unchecked")
Iterator<Registeruser> iter = query.list().iterator();
JSONObject jsonObj = new JSONObject();

while (iter.hasNext()) {
```

```

        Registeruser log = iter.next();
        JSONObject tempJObj = new JSONObject();

        tempJObj.put("registeruserUserId", log.getUserId());
        iter.remove();
        jObj.append("login", tempJObj);

    }

```

Ταυτοποίηση πως υπάρχει ο χρήστης στην βάση δεδομένων με βάση κωδικό πρόσβασης και ψευδώνυμο.

Κλάση Update Login

```

protected void doPost(HttpServletRequest request,
                        HttpServletResponse response) throws ServletException,
                        IOException {
    HashMap<String, String> jsonData;
    // If there was a problem parsing the JSON data
    if ((jsonData = parseSTJSON(getJSONString(request), response)) ==
null) {
        return;
    }

    // Open the session
    Session session = HibernateUtil.getSessionFactory().openSession();
    session.beginTransaction();

    // Create new object
    Login log = new Login();
    LoginId logid = new LoginId();

    logid.setRegisteruserUserId(toInteger(jsonData
        .get("registeruserUserId")));
    logid.setStartSession(toDateTime(jsonData.get("startSession")));
    log.setEndSession(toDateTime(jsonData.get("endSession")));
    log.setId(logid);
    log.setState(jsonData.get("state"));

    try {
        session.update(log);
        // Commit the transaction
        session.getTransaction().commit();
        session.close();
    } catch (Exception e) {
        session.close();
        setErrorMessage(response, 400, "Bad Request",

```

```

        "Could not update data. Check the data given as
input.");
        return;
    }

    JSONObject jsonObj = new JSONObject();
    jsonObj.put("userId", log.getId().getRegisteruserUserId());
    jsonObj.put("endSession", log.getEndSession());
    jsonObj.put("startSession", log.getId().getStartSession());
    jsonObj.put("state", log.getState());
    // Set the success message with the results
    setSuccessMessage(response, jsonObj);
}

```

Επεκτάσεις των πιο πάνω κλάσεων μπορούν να γίνουν με την ενημέρωση και των προσωπικών στοιχείων του χρήστη, εκτός του login. Επίσης δεδομένα θα μπορούσε να ανακτηθούν και από όλες τις προαναφερθείσες κλάσεις, παρά μόνο από την Login.

Βάση Δεδομένων Συστήματος

-- MySQL Workbench Forward Engineering

```

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

```

```

-----
-- Schema mydb
-----

```

```

-----
-- Schema ecg
-----

```

```

-----
-- Schema ecg
-----

```

```

CREATE SCHEMA IF NOT EXISTS `ecg` DEFAULT CHARACTER SET utf8 ;
USE `ecg` ;

```

```

-----
-- Table `ecg`.`login`
-----

```

```

CREATE TABLE IF NOT EXISTS `ecg`.`login` (
  `registeruser_User_ID` INT(11) NOT NULL,
  `start_session` DATETIME NOT NULL,
  `end_session` DATETIME NOT NULL,

```

```

`State` VARCHAR(45) NOT NULL,
PRIMARY KEY (`registeruser_User_ID`, `start_session`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

```

```

-----
-- Table `ecg`.`doctors`
-----

```

```

CREATE TABLE IF NOT EXISTS `ecg`.`doctors` (
  `Username` VARCHAR(45) NOT NULL,
  `Password` VARCHAR(45) NOT NULL,
  `Name` VARCHAR(45) NOT NULL,
  `Surname` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Username`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

```

```

-----
-- Table `ecg`.`measurementsperminute`
-----

```

```

CREATE TABLE IF NOT EXISTS `ecg`.`measurementsperminute` (
  `login_registeruser_User_ID` INT(11) NOT NULL,
  `date_taken` DATETIME NOT NULL,
  `Temperature_Mean_Value` FLOAT NOT NULL,
  `Airflow_Mean_Value` FLOAT NOT NULL,
  `Blood_Oxygen_Mean_Value` FLOAT NOT NULL,
  `Systolic_Pressure_Mean_Value` FLOAT NOT NULL,
  `Diastolic_Pressure_Mean_Value` FLOAT NOT NULL,
  `Muscle_Intensity_Mean_Value` FLOAT NOT NULL,
  `ECG_Mean_Value` FLOAT NOT NULL,
  `Conductivity_Mean_Value` FLOAT NOT NULL,
  `Pulse_Mean_Values` FLOAT NOT NULL,
  PRIMARY KEY (`login_registeruser_User_ID`, `date_taken`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

```

```

-----
-- Table `ecg`.`registeruser`
-----

```

```

CREATE TABLE IF NOT EXISTS `ecg`.`registeruser` (
  `User_ID` INT(11) NOT NULL,
  `Name` VARCHAR(45) NOT NULL,
  `Surname` VARCHAR(45) NOT NULL,
  `Gender` VARCHAR(6) NOT NULL,
  `Date_of_Birth` DATE NOT NULL,
  `Street` VARCHAR(45) NOT NULL,

```

```

`Number_of_Street` INT(2) NOT NULL,
`City` VARCHAR(45) NOT NULL,
`Country` VARCHAR(45) NOT NULL,
`Postal_Code` INT(5) NOT NULL,
`Telephone_Number` INT(8) NOT NULL,
`Email` VARCHAR(45) NOT NULL,
`Username` VARCHAR(45) NOT NULL,
`UserPassword` VARCHAR(45) NOT NULL,
PRIMARY KEY (`User_ID`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

```

```

-----
-- Table `ecg`.`shimmerecg`
-----
CREATE TABLE IF NOT EXISTS `ecg`.`shimmerecg` (
  `login_registeruser_User_ID` INT(11) NOT NULL,
  `Date_Taken` DATETIME NOT NULL,
  `ECG_1` FLOAT NOT NULL,
  `ECG_2` FLOAT NOT NULL,
  PRIMARY KEY (`login_registeruser_User_ID`, `Date_Taken`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

```

```

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```

Επεκτάσεις ως προς την βάση δεδομένων θα μπορούσαν να πραγματοποιηθούν με σκοπό την αποφυγή πλεονασμού δεδομένων στα πεδία. Ακόμη, με τα ίδια attributes, θα μπορούσε να δημιουργηθεί μια τοπική βάση δεδομένων, για ευκολότερη διαχείριση των μαζικών δεδομένων καταγραφής βιομετρήσεων.