



**ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΩΝ**

ΚΑΤΑΣΚΕΥΗ ΦΟΡΗΤΟΥ ΚΑΡΔΙΟΓΡΑΦΟΥ ΣΥΝΕΧΟΥΣ

ΚΑΤΑΓΡΑΦΗΣ (HOLTER MONITOR)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

**ΚΩΝΣΤΑΝΤΙΝΟΣ ΚΝΑΗΣ
ΑΕΜ 8967**

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΑΛΚΙΒΙΑΔΗΣ ΧΑΤΖΟΠΟΥΛΟΣ

ΘΕΣΣΑΛΟΝΙΚΗ, 5 Απριλίου 2021

Περίληψη

Η παρούσα εργασία αποτελεί προσπάθεια ανάλυσης και κατασκευής φορητής συσκευής καρδιογράφου (Holter Monitor) με προϊόντα του εμπορίου. Συγκεχριμένα, δημιουργήθηκε κύκλωμα ενίσχυσης και αναλογικού φιλτραρίσματος του σήματος εισόδου, το οποίο έπειτα μετατρέπεται σε ψηφιακό με συσκευή Arduino nano. Μετέπειτα πραγματοποιείται ψηφιακό φιλτράρισμα του σήματος, αποθήκευση και αποστολή του με Bluetooth σε συσκευή Android. Για τη συσκευή δημιουργήσαμε εφαρμογή για αυτόματη σύνδεση και παρουσίαση των δεδομένων σε μορφή γραφικής παράστασης. Το όλο σύστημα μπήκε σε ειδικά σχεδιασμένο κουτί εκτυπωμένο από 3D printer.

Χρήσιμοι όροι: ηλεκτροκαρδιογράφημα, φορητή ηλεκτρονική διάταξη, ηλεκτρόδια, 7 απαγωγοί, Arduino nano, εφαρμογή Android.

Abstract

This work is an attempt to analyze and manufacture a portable cardiograph also known as Holter Monitor with commercial products. Specifically, an amplification and analog filtering circuit was created, conversion of the analog signal to digital was done with an Arduino nano device, with subsequent digital signal filtering, storage and sending with Bluetooth to an Android device. For the Android device we created an application for automatic connection and presentation of data in the form of chart. The whole circuit was put in a specially designed 3D printed enclosure.

Useful terms: electrocardiogram, ECG, mobile electronic device, electrodes, 7 leads, Arduino nano, Android app.

Περιεχόμενα

1 Εισαγωγή	6
1.1 Περιγραφή	6
1.2 Σκοπός διπλωματικής	7
1.3 Διάρθρωση εργασίας	8
2 Θεωρητικό Υπόβαθρο	9
2.1 Η καρδιά	9
2.1.1 Μηχανική Δραστηριότητα	9
2.1.2 Ηλεκτρική δραστηριότητα	11
2.2 Ηλεκτροκαρδιογράφημα	14
2.2.1 Οι απαγωγές του ηλεκτροκαρδιογραφήματος	14
2.2.2 Διπολικές απαγωγές	15
2.2.3 Μονοπολικές απαγωγές	16
2.2.4 Ηλεκτρικό δυναμικό στο ΗΚΓ	17
2.3 Όργανα μέτρησης	18
2.3.1 Ηλεκτροκαρδιογράφος	18
2.3.2 Holter	19
2.3.3 Πηγές θορύβου	19
2.3.4 Κύκλωμα απαλοιφής θορύβου	20
3 Ανάλυση Υλικού	21
3.1 Αισθητήρες Ηλεκτροκαρδιογράφου	21
3.1.1 Συσκευή Bitalino	21
3.1.2 Ανάλυση κυκλώματος	22
3.1.3 Σχεδίαση κυκλώματος	24
3.2 Arduino nano	25
3.2.1 Micro SD Module	26
3.2.2 Bluetooth module	28
3.2.3 Τροφοδοσία	30

4 Ανάλυση Λογισμικού	31
4.1 Λογισμικό Ενσωματωμένου	31
4.1.1 Εισαγωγή Βιβλιοθηκών	31
4.1.2 Αρχικοποίηση Μεταβλητών	32
4.1.3 Συνάρτηση εκκίνησης	33
4.1.4 Συνάρτηση λειτουργίας	34
4.2 Λογισμικό εφαρμογής	37
4.2.1 Λειτουργία Bluetooth	37
4.2.2 Αρχικοποίηση	37
4.2.3 Εύρεση συσκευών	38
4.2.4 Σύνδεση σε συσκευή	38
4.2.5 Παραλαβή δεδομένων	39
4.2.6 Παρουσίαση δεδομένων	41
5 Συναρμολόγηση	43
5.1 Υλικά κατασκευής	43
5.2 Συναρμολόγηση χυκλώματος	44
5.3 Δημιουργία κουτιού	49
5.4 Φόρτωση λογισμικού Arduino	50
5.5 Φόρτωση λογισμικού Android	51
6 Αποτελέσματα και παρατηρήσεις	52
6.1 Τελικό αποτέλεσμα	52
6.2 Μετρήσεις	52
6.3 Απόδοση κώδικα	55
7 Επίλογος	56
7.1 Συμπεράσματα	56
7.2 Περαιτέρω έρευνα	56
8 Παράρτημα	58
8.1 Εγχειρίδιο χρήσης	58
8.1.1 Επισκόπηση	58
8.1.2 Οδηγίες χρήσης	60
8.2 Κώδικας	62

Κατάλογος Σχημάτων

1.1	Block diagram holter monitor	8
2.1	Διάγραμμα καρδιάς	10
2.2	Δραστηριότητα ενός απλού κολπικού βηματοδότη και ενός απλού κυττάρου κολπικού ιστού.	11
2.3	Διάγραμμα ηλεκτρικής δραστηριότητας	12
2.4	Πορεία ηλεκτρικής δραστηριότητας	13
2.5	Θέση απαγωγών	15
2.6	Θέση απαγωγών βάσει Eindhoven	15
2.7	Θέση υωρακιών απαγωγών	16
2.8	Χαρακτηριστικά ενός κανονικού ηλεκτροκαρδιογραφήματος.	17
2.9	Ηλεκτρικό δυναμικό της καρδιάς	18
2.10	ηλεκτροκαρδιογράφος	19
2.11	To Holter monitor	19
2.12	Block diagram	20
3.1	Bitalino ECG	21
3.2	Block διαγραμμα σήματος	22
3.3	Εσωτερικό instrumentation amplifier	23
3.4	Instrumentation amplifier με ανατροφοδότηση high pass filter	23
3.5	Low pass filter	24
3.6	Κύκλωμα συσκευής Bitalino	24
3.7	Τελική σχεδίαση κυκλώματος	25
3.8	Arduino nano pinout	25
3.9	Μονάδα χρήσης MicroSD	26
3.10	Λειτουργία σημάτων SPI	28
3.11	ΣύνδεσηSD module με Arduino nano	28
3.12	ΣύνδεσηUART	29
3.13	Πακέτο UART	29
3.14	Αποστολή δεδομένων UART	30
3.15	Σύνδεση bluetooth module	30

4.1	Τελική εφαρμογή	42
5.1	Σύνδεση Arduino για τροφοδοσία κυκλώματος. VCC - κόκκινο, Virtual ground - πορτοκαλί, Γείωση - Μαύρο	44
5.2	INA333	45
5.3	Σύνδεση INA333 (μπλέ)	45
5.4	AD8694	45
5.5	Σύνδεση με υψηπερατό φίλτρο (πράσινο) και κύκλωμα απομόνωσης (γαλάζιο)	46
5.6	Σύνδεση για χαμηλοπερατό φίλτρο	46
5.7	Σύνδεση SD module με Arduino nano	47
5.8	Σύνδεση bluetooth module με Arduino nano	47
5.9	Σύνδεση κουμπιού με Arduino nano	47
5.10	Τελική συνδεσμολογία	48
5.11	Τελική πειραματική συνδεσμολογία	48
5.12	Τελική συνδεσμολογία	49
5.13	Σχεδίαση κουτιού	49
5.14	Τοποθέτηση κυκλώματος στο κουτί	50
5.15	Φόρτωση λογισμικού Arduino	50
5.16	Τελική εφαρμογή	51
6.1	Σύγκριση μετρήσεων	52
6.2	Σύγκριση μετρήσεων	53
6.3	Τελικό αποτέλεσμα	54
8.1	Κουτί holter monitor	58
8.2	Ακροδέκτες holter monitor	59
8.3	Εφαρμογή Android	59
8.4	Τοποθέτηση ακροδεκτών στο ασθενή	60

Κομμάτια κώδικα

4.1	Εισαγωγή Βιβλιοθηκών	31
4.2	Αρχικοποίηση Bluetooth	32
4.3	Αρχικοποίηση διάφορων μεταβλητών	32
4.4	Συνάρτηση εκκίνησης	33
4.5	Συνάρτηση αρχικοποίησης κάρτας SD	33
4.6	Τιμές μετρήσεων και central	34
4.7	Έλεγχος παραλαβής εντολών	35
4.8	Αποστολή δεδομένων με bluetooth και εγγραφή σε κάρτα SD	35
4.9	Συνάρτηση αποστολής δεδομένων	36
4.10	Άδειες χρήσης Bluetooth και τοποθεσίας συσκευής	38
4.11	Πρόσβαση σε bluetooth adapter ενεργοποίηση του..	38
4.12	Δημιουργία socket για παραλαβή δεδομένων.	38
4.13	Διάβασμα δεδομένων	39
4.14	Παρουσίαση δεδομένων	41
4.15	Παρουσίαση δεδομένων	42
8.1	Κώδικας Arduino	62
8.2	Κλάση MainActivity.java	65
8.3	Κλάση ConnectedThread.java	65
8.4	Κλάση Ardconn.java	68

Κεφάλαιο 1

Εισαγωγή

1.1 Περιγραφή

Η ζωή σήμερα είναι γεμάτη με νέες τεχνολογίες και πληροφορίες, έτοιμες να διευκολύνουν τη καθημερινότητα μας. Με τη χρήση διάφορων έξυπνων συσκευών και τη εξάπλωση του διαδικτύου, μπορούμε να έχουμε πρόσβαση σε οποιαδήποτε πληροφορία κάθε χρονική στιγμή. Παραδείγματος χάριν, με ένα έξυπνο ρολόι μπορούμε να κάνουμε αγορές χωρίς και να χρειαστεί να πάρουμε πορτοφόλι, ενώ ένα μικρό raspberryPi είναι ικανό να ανιχνεύει και να ειδοποιεί για σεισμούς σε πραγματικό χρόνο.

Ένας από τους πιο σημαντικούς τομείς που έχουν εκμεταλλευτεί τη ραγδαία αυτή πρόοδο της τεχνολογίας είναι αυτός της ιατρικής. Ειδικότερα, ανάμεσα στους πολυάριθμους τομείς της έχει δημιουργηθεί ένας καινούργιος, αυτός της βιοϊατρικής. Η βιοϊατρική έχει σκοπό την ανάπτυξη, εκμετάλλευση και εφαρμογή νέων τεχνολογιών στους υπόλοιπους τομείς της ιατρικής με σκοπό την πιο εύκολη και σωστή διάγνωση ασθενεών. Για τον λόγο αυτό, αναπτύχθηκαν απλά εργαλεία που επιτρέπουν την καλύτερη παρακολούθηση του ασθενή, χωρίς να είναι απαραίτητη η παρουσία γιατρού στον ίδιο χώρο.

Ο καρδιογράφος είναι ένα από αυτά τα παραδείγματα που κάνουν τη ζωή του γιατρού πιο εύκολη, καθώς με την χρήση του είναι σε θέση να αναγνωρίσει διάφορα προβλήματα της καρδιάς, όπως είναι οι αρρυθμίες ή άλλες ανωμαλίες. Αυτό μπορεί να πάει ένα βήμα παραπέρα με τον Holter monitor, ένα φορητό καρδιογράφο. Ο ασθενής δεν χρειάζεται να είναι περιορισμένος στο κρεβάτι του νοσοκομείου και ο γιατρός μπορεί να παρατηρήσει τη λειτουργία της καρδιάς μέσα σε μια φυσιολογική μέρα.

Η χρήση τέτοιων έξυπνων συσκευών είναι πλέον αναγκαία στον κλάδο της ιατρικής, ειδικότερα στην εποχή του κορονοϊού, όπου ο φόρτος εργασίας στα νοσοκομεία είναι αυξημένος και η επικοινωνία πρόσωπο με πρόσωπο πρέπει να είναι περιορισμένη. Μικρές, έξυπνες και αξιόπιστες συσκευές είναι ένα πολύ καλό εργαλείο για τους γιατρούς και τους λύνει τα χέρια.

1.2 Σκοπός διπλωματικής

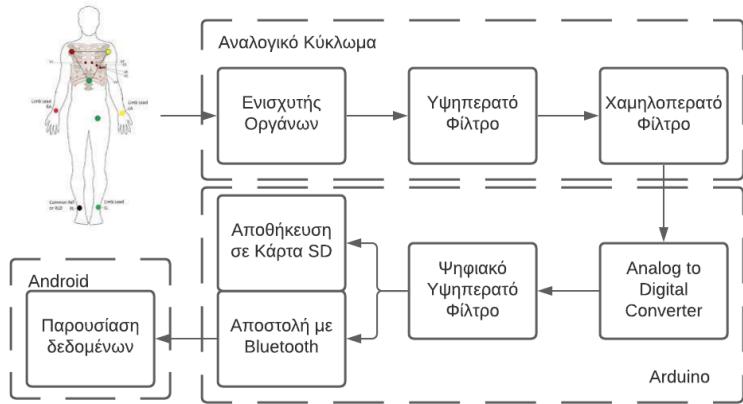
Έμπνευση αποτελεί προσωπική εμπειρία, στη οποία ο καρδιολόγος μου πρότεινε να κάνω ένα καρδιογράφημα για ανίχνευση ανωμαλιών. Επίσκεψη ρουτίνας, απλά όμως τοποθετούσαν τα ηλεκτρόδια και όμως έφευγα. Στο νοσοκομείο η πραγματικότητα ήταν διαφορετική. Το holter ήταν απλά ένα μαύρο κουτί, χωρίς κάποια ένδειξη ότι λειτουργεί εκτός από ένα led το οποίο είχαν καλύψει με ταινία γιατί ήταν ενοχλητικό το βράδυ. Έτσι οι νοσοκόμοι τοποθέτησαν τα ηλεκτρόδια και μου είπαν να φύγω νομίζοντας ότι όλα λειτουργούν, χωρίς να συνειδητοποιήσουν πως δεν ξεκίνησαν τη μέτρηση. Το συνειδητοποίησαν μόνο τη επόμενη μέρα όταν πήγα να το βγάλω και είδαν ότι δεν καταγράφηκε κάτι. Μου ξανασύνδεσαν το holter ελέγχοντας αυτή τη φορά ότι λειτουργεί.

Προφανώς το μεγάλο ελάττωμα της συσκευής είναι ότι δεν είχε κάποια επικοινωνία με το περιβάλλον παρά μόνο το led. Μια ουδόνη όμως ήταν καλή ιδέα αλλά έχει προτεραιότητα η μεγαλύτερη λειτουργία της συσκευής. Μια καλή εναλλακτική είναι μια bluetooth εφαρμογή, αφού πλέον τα κινητά βρίσκονται στη τσέπη όλων. Με αυτό το τρόπο όμως μπορεί να παρουσιάζονται δεδομένα χωρίς να χρειάζεται να μαντεύει ο χρήστης αν η συσκευή λειτουργεί.

Στη διπλωματική αυτή όμως προσπαθήσουμε να μελετήσουμε και να αναπτύξουμε μια τέτοια συσκευή, ικανή για 24ωρη παρακολούθηση της ηλεκτρικής δραστηριότητας της καρδιάς με προϊόντα του εμπορίου και ζωντανή παρακολούθηση της μέσω εφαρμογής Android. Έχουμε ως στόχο τη ακριβή μέτρηση της ηλεκτρικής δραστηριότητας καρδιάς και η καθαρή παρουσίαση τους από τον τελικό χρήστη που προορίζεται να είναι ένας γιατρός ή ασθενής.

Η γενική μορφή του holter monitor χωρίζεται σε τρία κύρια μέρη: το αναλογικό κύκλωμα, το κύκλωμα του arduino και τη εφαρμογή android. Συγκεκριμένα αυτά χωρίζονται σε:

- Αναλογικό κύκλωμα
 - Ενυσχυτής οργάνων
 - Ψηφιακό υψηλερατό φίλτρο
 - Χαμηλοπερατό φίλτρο
- Arduino
 - Analog to digital converter
 - Ψηφιακό υψηλερατό φίλτρο
 - Αποθήκευση σε κάρτα sd
 - Αποστολή με bluetooth
- Android
 - Παρουσίαση δεδομένων.



Σχήμα 1.1: Block diagram holter monitor

1.3 Διάρθρωση εργασίας

Η διάρθρωση της εργασίας είναι ως εξής:

- Θεωρητικό Υπόβαθρο:** θεωρητική ανάλυση της λειτουργίας της καρδιάς και του καρδιογράφου. Εξηγούμε τη φυσιολογία της καρδιάς και τη ηλεκτρική της δραστηριότητα τη οποία παρακολουθούμε. Παρουσιάζουμε τη σωστή τοποθέτηση των ηλεκτροδίων, πως παρουσιάζεται το ηλεκτρικό δυναμικό παραδοσιακά σε χαρτί και η λειτουργία των συνηθισμένων οργάνων μέτρησης.
- Ανάλυση Τλικού:** παρουσίαση του υλικού που θα χρησιμοποιηθεί και σωστή σύνδεση του. Συγκεκριμένα αναλύουμε το κύκλωμα της συσκευής Bitalino, παρουσιάζουμε τη συσκευή Arduino nano και τα περιφερικά που θα τοποθετηθούν σε αυτό.
- Ανάλυση Λογισμικού:** παρουσίαση του κώδικα που θα χρησιμοποιηθεί στο Arduino και τη συσκευή Android. Εξηγούμε πως φιλτράρουμε, αποθηκεύουμε και στέλνουμε δεδομένα από πλευράς Arduino. Από πλευράς Android εξηγούμε πως λειτουργεί το bluetooth στο κινητό, πως βρίσκουμε συσκευές, παραλαμβάνουμε και παρουσιάζουμε δεδομένα σε μορφή γραφικής.
- Συναρμολόγιση:** οδηγίες συναρμολόγησης όσων ειπώθηκαν σε ένα κεφάλαιο, για εύκολη αναπαραγωγή της πτυχιακής. Αναλυτικά βήματα για το τι και πόσα υλικά χρειάζονται, πως να δημιουργήσουμε το κύκλωμα σε breadboard και σύνδεση του με το Arduino και των περιφερικών. Επίσης πως φορτώνουμε το λογισμικό σε Arduino και Android αντίστοιχα.
- Παράρτημα:** κώδικας που χρησιμοποιήθηκε στο Arduino και το Android και γενικές οδηγίες χρήσης.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

Για να κατανοήσουμε τη λειτουργία του συστήματος μας, αρχικά πρέπει να εξηγήσουμε το πώς λειτουργεί η καρδιά, τι ακριβώς μετράμε και πως δουλεύει μια συσκευή Holter.

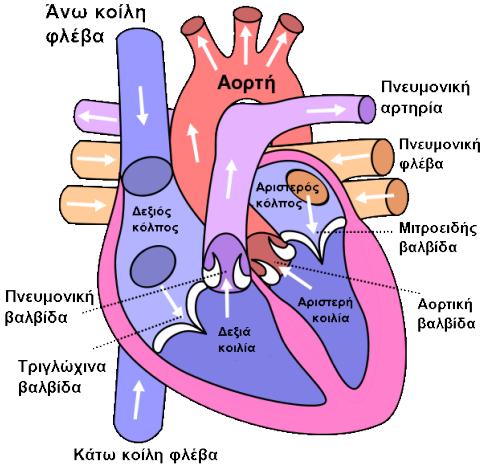
2.1 Η καρδιά

2.1.1 Μηχανική Δραστηριότητα

Η καρδιά είναι ένα κοίλο μυώδες όργανο του σώματος που μετατρέπει την μεταβολική ενέργεια σε μηχανική, παρέχοντας με τον τρόπο αυτό τη δύναμη που απαιτείται για τη συνεχή κυκλοφορία του αίματος μέσα στο κυκλοφοριακό σύστημα. Λειτουργεί δηλαδή ως αντλία. Η καρδιά είναι εφοδιασμένη με 4 ξεχωριστές αντλίες : δύο προαντλίες, τους κόλπους και δύο προωθητικές αντλίες, τις κοιλίες.

Τύπο φυσιολογικές συνθήκες, το αίμα ρέει συνεχώς από τις μεγάλες φλέβες στους κόλπους. Το 70% περίπου από αυτό ρέει κατευθείαν στους κόλπους προς τις κοιλίες πριν ακόμα οι κόλποι συσταλούν. Η κολπική συστολή προσθέτει το υπόλοιπο 30% για τη πλήρωση των κοιλιών. Κατά συνέπεια οι κόλποι λειτουργούν ως προαντλίες που αυξάνουν την αποτελεσματικότητα των κοίλων ως αντλιών κατά 30%. Με τη βοήθεια των κόλπων, των κοιλιών και των διαφόρων άλλων τμημάτων της καρδιάς γίνεται εφικτή η πραγματοποίηση ενός καρδιακού κύκλου. Ο καρδιακός κύκλος πάει ως εξής: Το αίμα αφού διαβιβασθεί στα διάφορα μέρη του σώματος επιστρέφει στη καρδιά μέσω των κοίλων (άνω και κάτω) φλεβών και εισέρχεται στο δεξιό κόλπο (RA). Κατόπιν το αίμα ρέει στη δεξιά κοιλία (RV) από όπου διοχετεύεται στα πνευμόνια, οξυγονώνεται και επιστρέφει στο αριστερό κόλπο (LA). Από εδώ εισέρχεται στη αριστερή κοιλία (LV), από τη οποία καθαρό πλέον διοχετεύεται στα άλλα μέρη του σώματος μέσω της αορτής. Στη συνέχεια το αίμα επιστρέφει και πάλι στη καρδιά μέσω των κοίλων φλεβών για τη έναρξη ενός νέου κύκλου λειτουργίας. Η ρυθμική δραστηριότητα της καρδιάς αποτελείται από τα εξής διαστήματα:

1. Γέμισμα της κοιλίας
2. Ισομετρική συστολή της κοιλίας
3. Κοιλιακή εξώθηση
4. Ισομετρική χαλάρωση της κοιλίας



Σχήμα 2.1: Διάγραμμα καρδιάς

Γέμισμα της κοιλίας

Όταν αρχίζει η διαστολή των κοιλιών οι κοιλιακές πιέσεις γίνονται μικρότερες από τις κολπικές πιέσεις γίνονται μικρότερες από τις κολπικές με αποτέλεσμα να έχουμε το άνοιγμα των κολποκοιλιακών βαλβίδων και εισροή του αίματος με μεγάλη ταχύτητα στις κοιλίες. Αυτή η περίοδος διαρκεί περίπου το πρώτο τρίτο της διαστολής. Κατά το δεύτερο τρίτο εισέρχεται στις κοιλίες μια μικρή ποσότητα αίματος. Πρόκειται για αίμα που εξακολουθούν να αδειάζουν οι φλέβες στους κόλπους από τους οποίους μεταβιβάζεται κατευθείαν στις κοιλίες. Κατά το τελευταίο τρίτο της κοιλιακής διαστολής, οι κόλποι συστέλλονται και ωθούν προς τις κοιλίες το 30% περίπου της ποσότητας του αίματος που υπολείπεται για το γέμισμα τους. Δηλαδή στα δύο τελευταία δύο τρίτα της κοιλιακής διαστολής παρατηρείται μια επιβράδυνση του γεμίσματος των κοιλιών. Η φάση αυτή είναι γνωστή ως περίοδος διάστασης.

Ισομετρική συστολή της κοιλίας

Όταν αρχίσει η κοιλιακή συστολή η πίεση στις κοιλίες αυξάνει απότομα (ο όγκος τους παραμένει σταθερός). Η αύξηση αυτή έχει σαν αποτέλεσμα το κλείσιμο των κολποκοιλιακών βαλβίδων. Οι βαλβίδες της αορτής και της πνευμονικής αρτηρίας παραμένουν κλειστές μέχρι να αναπτυχθεί αρχετή πίεση στις κοιλίες για να προκαλέσει τη διάνοιξη τους. Επομένως κατά τη διάρκεια της χρονικής αυτής περιόδου οι κοιλίες συστέλλονται χωρίς όμως να αδειάζουν

Κοιλιακή εξώθηση

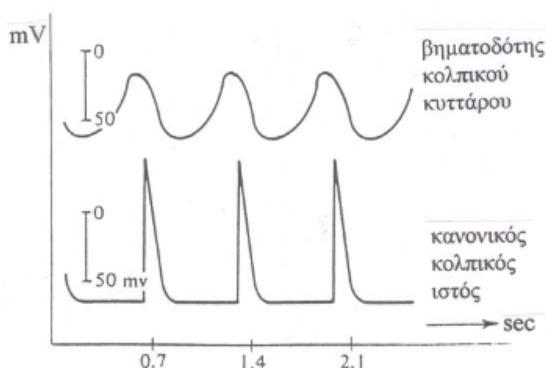
Όταν οι πιέσεις στις κοιλίες γίνουν λίγο μεγαλύτερες από 80 mm Hg στη αριστερή και 8mm Hg στη δεξιά ανοίγουν μηνοειδείς βαλβίδες. Αμέσως αρχίζει η έξοδος του αίματος από τις κοιλίες που εκκενώνονται κατά 60% στο πρώτο τέταρτο της συστολής και κατά το μεγαλύτερο μέρος από τα υπόλοιπα 40% στα δύο επόμενα τέταρτα της. Αυτά τα τρία τέταρτα χρόνου συστολής ονομάζονται περίοδος εξώθησης.

Ισομετρική χαλάρωση της κοιλίας

Στο τέλος της συστολής αρχίζει απότομα η διαστολή των κοιλιών με συνέπεια τη γρήγορη πτώση των ενδοκοιλιακών πιέσεων. Οι αυξημένες πιέσεις των μεγάλων αρτηριών προκαλούν παλινδρόμηση του αίματος προς τις κοιλίες με αποτέλεσμα το απότομο κλείσιμο της αορτικής και πνευμονικής αρτηρίας. Η χαλάρωση του μυοκαρδίου των κοιλιών συνεχίζεται για λίγο ακόμη και οι ενδοκοιλιακές πιέσεις επιστρέφουν γρήγορα στις πολύ χαμηλές διαστολικές τους τιμές. Η περίοδος χαλάρωσης του μυοκαρδίου των κοιλιών λέγεται και περίοδος ισομετρικής χαλάρωσης. Στη συνέχεια ανοίγουν οι κολποκοιλιακές βαλβίδες και αρχίζει νέος κύκλος της λειτουργίας των κοιλιών ως αντλιών.

2.1.2 Ηλεκτρική δραστηριότητα

Η ηλεκτρική δραστηριότητα της καρδιάς αρχίζει από το φλεβόκομβο, γνωστός και ως βηματοδότης. Τα κύτταρα που βρίσκονται στη περιοχή του φλεβόκομβου έχουν το χαρακτηριστικό της αυτοδιέγερσης. Το δυναμικό της μεμβράνης του κυττάρου μετά από ένα ηλεκτρικό παλμό αντί να διατηρηθεί σε ηρεμία αυξάνει απότομα μέχρι τη τιμή κατωφλιού με αποτέλεσμα τη δημιουργία ενός ηλεκτρικού παλμού. Αυτοί οι παλμοί οδηγούν στη δημιουργία μιας φυσιολογικής σειράς καρδιακών παλμών

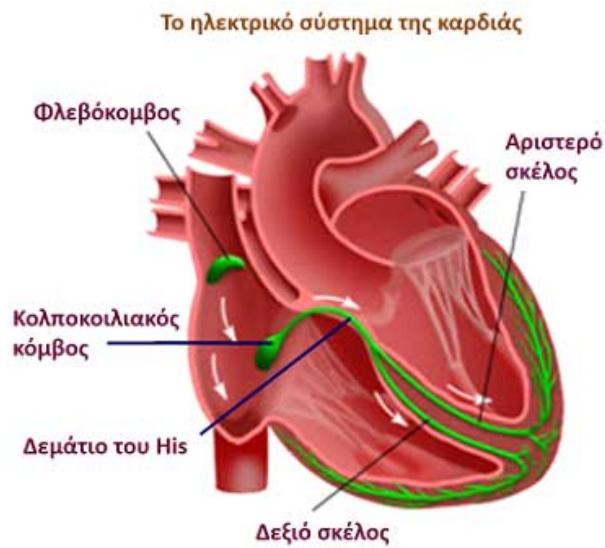


Σχήμα 2.2: Δραστηριότητα ενός απλού κολπικού βηματοδότη και ενός απλού κυττάρου κολπικού ιστού.

Ο ρυθμός της καρδιάς καθορίζεται από τη δραστηριότητα του βηματοδότη, ο οποίος ενεργεί σαν ελεύθερος πολυταλαντοτής. Ο ρυθμός τροποποιείται από τις ανταγωνίστηκες επιδράσεις των παρασυμπαθητικών και συμπαθητικών νεύρων. Η επίδραση των παρασυμπαθητικών ελαττώνει το ρυθμό της καρδιάς ενώ η επίδραση των συμπαθητικών αυξάνει το ρυθμό.

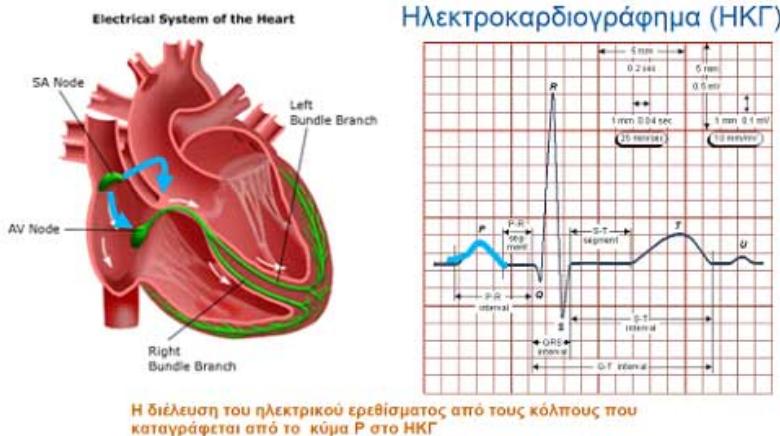
Η δραστηριότητα που αρχίζει στο κόμβο του φλεβόκομβου (SA) μεταδίδεται μέσω του κολπικού μυ με ταχύτητα 1m/sec και χρειάζονται 80ms για πλήρη ενεργοποίηση του κόλπου. Προς το τέλος αυτής της περιόδου η ηλεκτρική δραστηριότητα φυλάνει κολποκυλιακό κόμβο (AV). Μέσω του κόμβου αυτού ειδικευμένα κύτταρα αγωγής μεταφέρουν τη διέγερση στις κοιλίες. Η διάδοση της διέγερσης στο κολποκυλιακό κόμβο γίνεται πάρα πολύ αργά με ταχύτητα 0.1m/sec.

Έπειτα το σήμα περνάει από το δεμάτιο του His και κατόπιν μέσω των ειδικευμένων αριστερών και δεξιών δεσμών του δεματίου η διέγερση μεταδίδεται σε όλα τα σημεία των κοιλιών. Δηλαδή αφού φτάσει πρώτα στη εσωτερική επιφάνεια των κοιλιών σχεδόν αμέσως εξαπλώνεται και στη εξωτερική επιφάνεια μέσω αυτών των ινών. Το σύστημα αυτό βοηθάει στη έναρξη της ηλεκτρικής διέγερσης στη μυϊκή περιοχή. Το μυοκάρδιο συμπεριφέρεται σαν ένα απλό κύτταρο. Η διεγερσιμότητα μεταδίδεται με ταχύτητα 0,5m/sec και προκαλεί μηχανικές συστολές με αποτελεσματικό και συγχρονισμένο τρόπο.



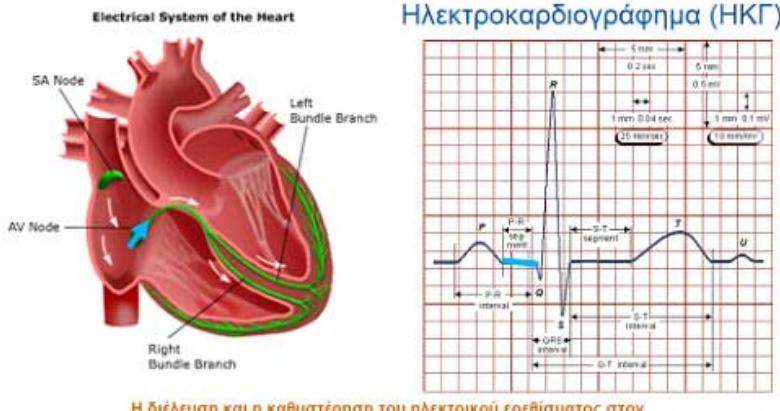
Σχήμα 2.3: Διάγραμμα ηλεκτρικής δραστηριότητας

Η πορεία του ηλεκτρικού ερεθίσματος της καρδιάς και το ηλεκτροκαρδιογράφημα



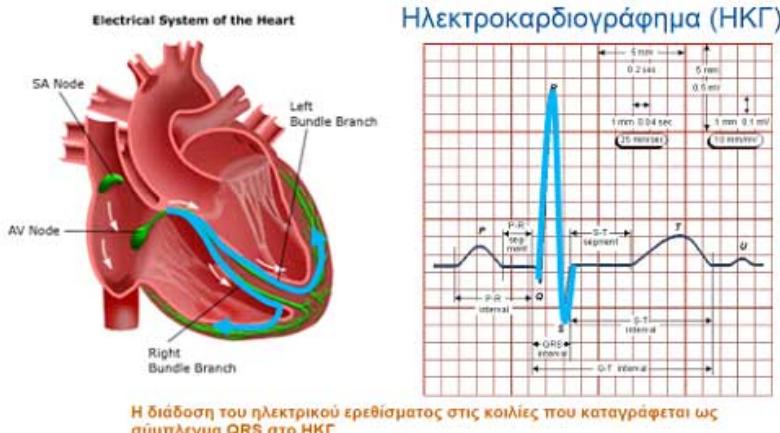
Η διέλευση του ηλεκτρικού ερεθίσματος από τους κόλπους που καταγράφεται από το κύμα P στο ΗΚΓ

Η πορεία του ηλεκτρικού ερεθίσματος της καρδιάς και το ηλεκτροκαρδιογράφημα



Η διέλευση και η καθυστέρηση του ηλεκτρικού ερεθίσματος στον κόλποιοιλιακό κόλπο που καταγράφεται ως «τμήμα PR» στο ΗΚΓ

Η πορεία του ηλεκτρικού ερεθίσματος της καρδιάς και το ηλεκτροκαρδιογράφημα



Η διάδοση του ηλεκτρικού ερεθίσματος στις κοιλίες που καταγράφεται ως σύμπλεγμα QRS στο ΗΚΓ

Σχήμα 2.4: Πορεία ηλεκτρικής δραστηριότητας

2.2 Ηλεκτροκαρδιογράφημα

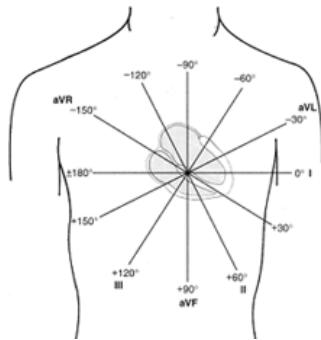
Λόγω της ηλεκτρικής δραστηριότητας του καρδιακού μυ έχουμε τη εμφάνιση ενός ηλεκτρικού πεδίου στη αγώγιμη περιοχή του σώματος που περιβάλλει τη καρδιά. Τα δυναμικά που εμφανίζονται στη επιφάνεια του σώματος σαν επακόλουθα τη καρδιακής δραστηριότητας είναι γνωστά σαν ηλεκτροκαρδιογράφημα (ECG). Η δημιουργία του ECG μελετάται σε δύο μέρη.

Στο πρώτο μέρος γίνεται κατά κάποιο τρόπο ο καθορισμός των πηγών των ρευμάτων στη καρδιά. Επιπλέον μπορούμε να καθορίσουμε τη κατανομή της διπολικής ροπής ανά μονάδα όγκου \vec{J}_i . Μια απλή προσέγγιση στο πρόβλημα μας είναι να υπολογίσουμε το άθροισμα όλων των διανυσμάτων των διπολικών στοιχείων $\vec{J}_i dV$. Δηλαδή σχηματίζουμε το διάνυσμα $\vec{p} = \int_v \vec{J}_i dV$ και θεωρούμε ότι αυτό το απλό δίπολο παριστάνει τη ηλεκτρική δραστηριότητα της καρδιάς. Σχεδόν όλη η κλινική ηλεκτροκαρδιογραφία βασίζεται σε αυτό το γεγονός. Στο δεύτερο μέρος της μελέτης του ECG γίνεται ο καθορισμός των επιφανειακών δυναμικών που οφείλονται στο δίπολο της καρδιάς. Στη ανάλυση αυτή το σώμα θεωρείται γραμμικό, ομοιογενές και ισοτοπικό.^[1]

2.2.1 Οι απαγωγές του ηλεκτροκαρδιογραφήματος

Οι ΗΚΓ φικές απαγωγές των άκρων καταγράφουν τα δυναμικά της καρδιάς στο μετωπιαίο επίπεδο. Η κάθε μία από αυτές τις απαγωγές καταγράφει τα ηλεκτρικά δυναμικά από μία ορισμένη κατεύθυνση, η οποία αναφέρεται με βάση τη γωνία σε μοίρες που σχηματίζει ο θετικός πόλος της με την απαγωγή I. (Η απαγωγή I καταγράφει τα ηλεκτρικά δυναμικά από την πλευρά του αριστερού βραχίονα, από το αριστερό άκρο μίας οριζόντιας γραμμής που περνά από το κέντρο της καρδιάς). Έτσι, ο θετικός πόλος της απαγωγής I είναι στις 0° ,

της II στις $+60^\circ$, της III στις $+120^\circ$, της aVR στις -150° , της avL στις -30° και της avF στις $+90^\circ$, όπως φαίνεται στο σχήμα:



Σχήμα 2.5: Θέση απαγωγών

2.2.2 Διπολικές απαγωγές

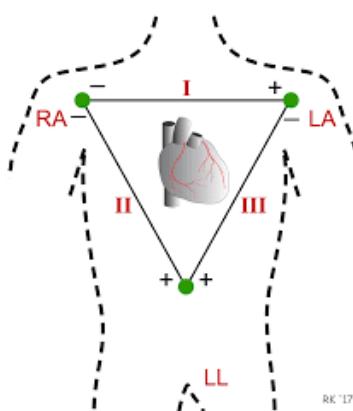
Στις διπολικές απαγωγές δύο ενεργά ηλεκτρόδια για καταγραφή διαφοράς δυναμικού μεταξύ δύο ενεργών σημείων που τοποθετούνται. Συνήθεις θέσεις των ηλεκτροδίων είναι το δεξί χέρι, αριστερό χέρι και αριστερό πόδι. Έτσι προκύπτουν τρείς συνδυασμοί:

Lead I = LA-RA

Lead II = LL-RA

Lead III = LL-LA

Τα ανύσματα μεταξύ τους μπορούν να προσεγγισθούν σε ένα ισόπλευρο τρίγωνο, γνωστό ως το τρίγωνο του Eindhoven. Σύμφωνα με τον κανόνα του Eindhoven το δυναμικό του II είναι ισοδύναμο με το άθρισμα των I + III, δηλαδή: Lead II = Lead I + Lead III ή LL-RA = LA-RA + LL-LA.



Σχήμα 2.6: Θέση απαγογών βάσει Eindhoven

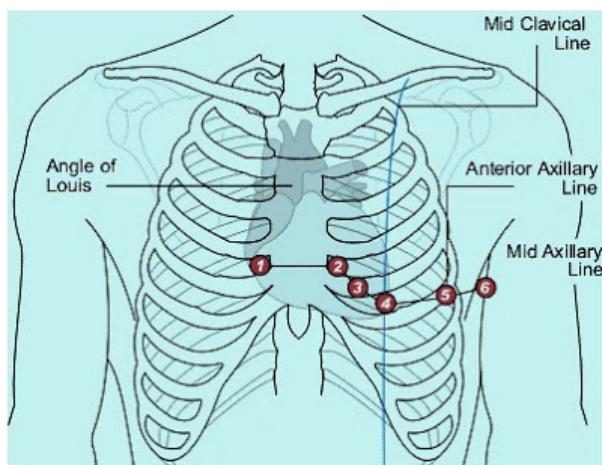
2.2.3 Μονοπολικές απαγωγές

Σε αντίθεση με τις διπολικές οι μονοπολικές απαγωγές δημιουργούν ένα τεχνητό σημείο αναφοράς στο καρδιογράφο και μετράμε διαφορές δυναμικού μεταξύ αυτού και ενός ενεργού ηλεκτροδίου. Έτσι προκύπτουν οι θωρακικές και προσαυξημένες επαγωγές.

Θωρακικές απαγωγές

Οι προκάρδιες απαγωγές καταγράφουν ηλεκτρικά δυναμικά που προέρχονται από την καρδιά, από συγκεκριμένες θέσεις του θωρακικού τοιχώματος που είναι οι εξής:

- Η V1, στο τέταρτο μεσοπλεύριο διάστημα ακριβώς δεξιά του στέρνου,
- η V2, στο τέταρτο μεσοπλεύριο διάστημα ακριβώς αριστερά του στέρνου,
- η V3 στο μέσον μεταξύ V2 και V4,
- η V4 στο πέμπτο μεσοπλεύριο διάστημα στη μεσοκλειδική γραμμή,
- η V5 στην πρόσθια μασχαλιά γραμμή, στο ίδιο ύψος με τη V4, και
- η V6, στη μέση μασχαλιά γραμμή, στο ίδιο ύψος με τις V4 και V5.

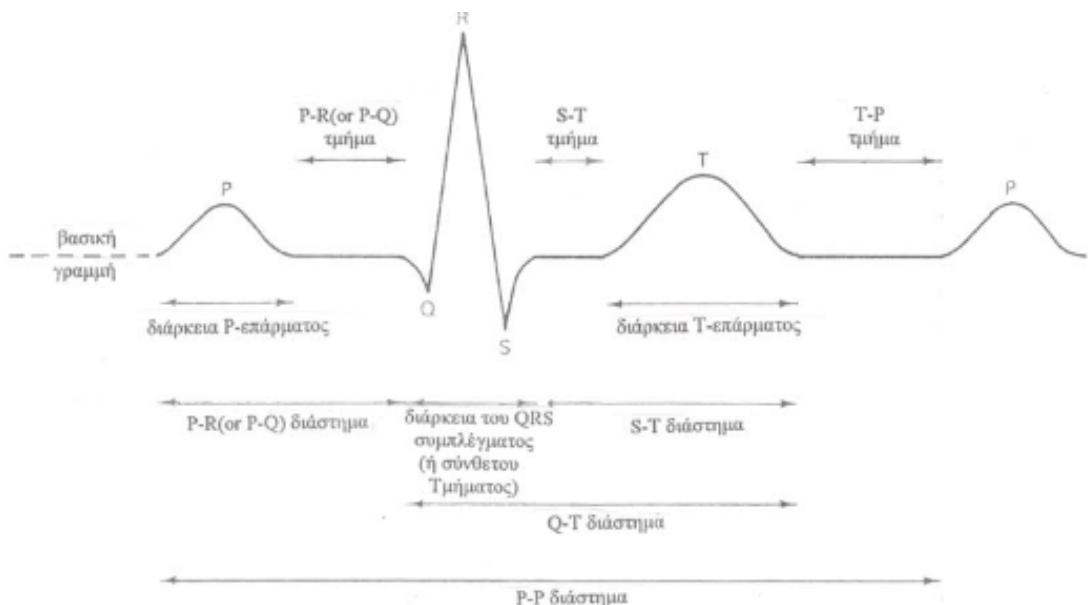


Σχήμα 2.7: Θέση θωρακικών απαγωγών

Όλες μαζί οι απαγωγές (οι 6 των άκρων που βρίσκονται στο μετωπιαίο επίπεδο και οι 6 προκάρδιες που είναι διατεταγμένες κατά το οριζόντιο επίπεδο), παρέχουν μια τρισδιάστατη καταγραφή της καρδιακής ηλεκτρικής δραστηριότητας. Η κάθε απαγωγή μπορεί να παρομοιασθεί σαν ένα 'μάτι' που βλέπει τα ίδια γεγονότα (στην προκειμένη περίπτωση την εκπόλωση και επαναπόλωση του μυοκαρδίου) από διαφορετική γωνία ή κατεύθυνση. Στο ηλεκτροκαρδιογράφημα οι σημαντικές ποσότητες είναι τα P,Q,R,S και T των επαρμάτων, οι διάρκειες τους και τα χρονικά διαστήματα μεταξύ των επαρμάτων. Το χρονικό σημείο αναφοράς εκλέγεται συνήθως η κορυφή R και η διάρκεια του καριδακού κύκλου είναι το διάστημα RR.

1. Το έπαρμα P προκαλείται από ηλεκτρικά ρεύματα που παράγονται καθώς οι κόλποι εκπολώνονται πριν από τη συστολή τους.

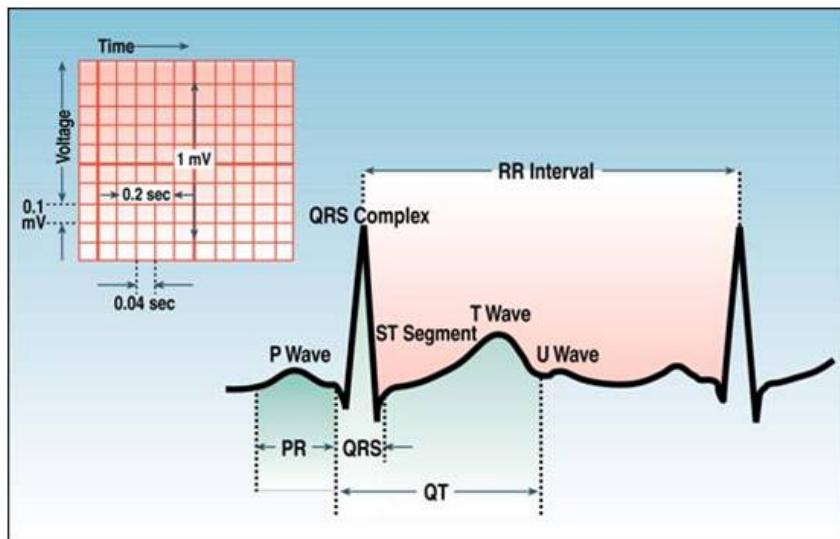
2. Το QRS προκαλείται από ηλεκτρικά ρεύματα που παρέχονται όταν οι κοιλίες εκπολώνονται πρίν από τη συστολή τους
3. Το έπαρμα T προκαλείται από ρεύματα που παράγονται καθώς οι κοιλίες επαναπολώνονται πριν από τη διαστολή τους.
4. Το διάστημα PR είναι ένα μέτρο του AV χρόνου αγωγιμότητας και παραμορφώσεις με τη αγωγιμότητα σχετίζονται με το διάστημα αυτό
5. Η βασική γραμμή δίνεται από το τμήμα PT το σχετικό επίπεδο του τμήματος ST είναι ένα σπουδαίο διαγνωστικό μέτρο
6. Το διάστημα QT δίνει τη ολική διάρκεια της κοιλιακής συστολής



Σχήμα 2.8: Χαρακτηριστικά ενός κανονικού ηλεκτροκαρδιογραφήματος.

2.2.4 Ηλεκτρικό δυναμικό στο ΗΚΓ

Το ΗΚΓ αποτελεί μία καμπύλη ηλεκτρικού δυναμικού σε συνάρτηση με το χρόνο. Στον οριζόντιο άξονα μετράται ο χρόνος: με τη συνήθη ταχύτητα του ΗΚΓφικού χαρτιού που είναι 25 mm/sec, 1mm = 40 msec (0,04 sec). Στον κάθετο άξονα μετράται το ηλεκτρικό δυναμικό: 10mm = 1mV. Η οριζόντια γραμμή που αντιστοιχεί σε μηδενική τιμή του ηλεκτρικού δυναμικού λέγεται ισοηλεκτρική γραμμή. Αποτελεί τη γραμμή από την οποία μετράται το ύψος των διαφόρων κυμάτων του ΗΚΓ. Η παραγωγή του αρχικού ηλεκτρικού σήματος από το φλεβόκομβο δεν παράγει εμφανές κύμα στο ΗΚΓ. Τα θετικά κύματα είναι όσα βρίσκονται πάνω από την ισοηλεκτρική γραμμή, ενώ τα αρνητικά προεξέχουν κάτω από την ισοηλεκτρική γραμμή. [2]



Σχήμα 2.9: Ηλεκτρικό δυναμικό της καρδιάς

Στη συνήθη ταχύτητα του ΗΚΓφικού χαρτιού (25 mm/s), μία απόσταση 25 mm στον οριζόντιο άξονα αντιπροσωπεύει 1 δευτερόλεπτο . Αφού $1 \text{ λεπτό} = 60 \text{ δευτερόλεπτα}$, στο ΗΚΓ το ένα λεπτό είναι μία απόσταση $60 \times 25 \text{ mm} = 1500 \text{ mm}$. Όταν ο καρδιακός ρυθμός είναι κανονικός, τότε ένας καρδιακός παλμός εμφανίζεται σε μία σχεδόν σταθερή απόσταση (D) από τον προηγούμενο. Συνεπώς, κάθε φορά που το ΗΚΓφικό χαρτί θα διανύσει αυτή την απόσταση D, θα εμφανισθεί ένας καινούργιος καρδιακός παλμός. Αν ο ρυθμός είναι κανονικός, διαιρώντας την απόσταση που διανύει το χαρτί σε 1 λεπτό δια την απόσταση D (σε mm) που μεσολαβεί μεταξύ δύο διαδοχικών παλμών, βρίσκουμε πόσοι παλμοί εμφανίζονται ανά λεπτό

2.3 Όργανα μέτρησης

2.3.1 Ηλεκτροκαρδιογράφος

Οι απαγωγές συνδέονται σε ένα ηλεκτροκαρδιογράφο, ρόλος του οποίου είναι να ενισχύει το αδύναμο σήμα που έρχεται από τους απαγωγούς (τάξεως $3-5 \text{ mV}$), να το περνάει από ένα ζωνοπερατό φίλτρο, να καταγράφει το σήμα και να το εμφανίζει. Πιο αναλυτικά θα το δούμε παρακάτω στην ανάλυση υλικού.



Σχήμα 2.10: ηλεκτροκαρδιογράφος

2.3.2 Holter

To Holter monitor είναι ένας φορητός ηλεκτροκαρδιογράφος, εφεύρεση του Norman J. Holter το 1962. To holter είναι σχεδιασμένο για παρακολούθηση του ασθενή σε καθημερινές ασχολίες, χωρίς να χρειάζεται να μείνει στο νοσοκομείο. Τροφοδοτείται με μπαταρία και σκοπός τους είναι η καταγραφή 24,48 εώς και 72 ωρών.



Σχήμα 2.11: Το Holter monitor

2.3.3 Πηγές θορύβου

Το ηλεκτροκαρδιογράφημα είναι σήμα χαμηλών συχνοτήτων μεταξύ 0,5-150Hz. Συνήθη προβλήματα που συναντούμε είναι η ύπαρξη θορύβου κατά την καταγραφή. Για αποφυγή

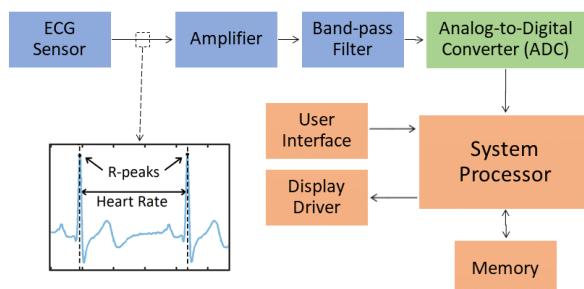
σφαλμάτων κατά τη διάγνωση, πρέπει να εντοπίσουμε τις πηγές θορύβου και να τις απαλείψουμε. Κύριες πηγές θορύβου μπορεί να είναι

- Κίνηση ασθενή
- Παρεμβολές από γραμμή ηλεκτρικού ρεύματος
- Θόρυβος οργάνων

2.3.4 Κύκλωμα απαλοιφής θορύβου

Για την απαλοιφή του θορύβου, το σήμα μας περνάει από έναν ενισχυτή οργάνων (instrumentation amplifier) ο οποίος λειτουργεί σαν προενοισχητής (preamplifier), για ενίσχυση του ασθενούς σήματος που έρχεται από τις απαγωγές με κέρδος 10. Έπειτα, το σήμα περνάει από ένα υψηλοπερατό φίλτρο (high pass filter) για αποκοπή συχνοτήτων στα 0.5Hz που μπορεί να συμβαίνουν από κίνηση ασθενή η από τα καλώδια. Το σήμα αυτό ενισχύεται ξανά (κέρδος 1000) και περνάει από ένα τελευταίο φίλτρο χαμηλοπεράτο (low pass filter) αυτή τη φορά αποκόπτοντας συχνότητες πάνω από 40Hz.

Με τη μέθοδο αυτή, απαλείφουμε τόσο τους περισσότερους θορύβους, όσο και την βελτίωση του σήματος, μέσω καλής ψωράκισης των καλωδίων και καλής επαφής των ηλεκτροδίων με το δέρμα.



Σχήμα 2.12: Block diagram

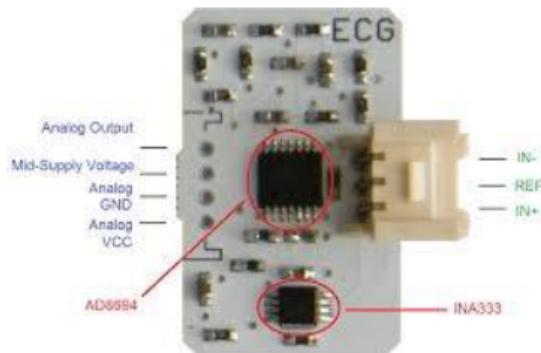
Κεφάλαιο 3

Ανάλυση Υλικού

3.1 Αισθητήρες Ηλεκτροκαρδιογράφου

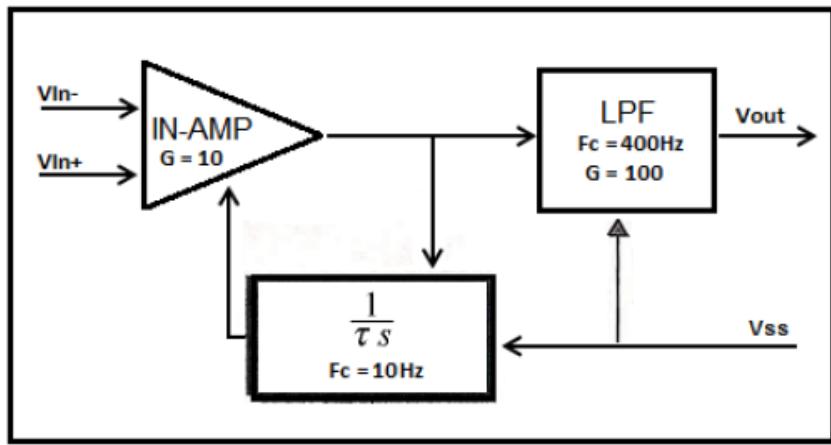
3.1.1 Συσκευή Bitalino

Η συνδεσμολογία των αισθητήρων βασίζεται στη συσκευή Bitalino . Παρατηρώντας το κύκλωμα έχει τάση λειτουργίας analog VCC 3.3V που τροφοδοτείται από το pin τροφοδοσίας 3.3V του Arduino, γείωση analog GND που συνδέεται στη γείωση του Arduino και εικονική γείωση VSS 1.65V. Έχει εισόδους IN-, REF, IN+ που παίρνω από τα ηλεκτρόδια λήψης σήματος και έξοδο analog output η οποία συνδέεται στα pin A1, A2, A3 του Arduino.



Σχήμα 3.1: Bitalino ECG

Συνοπτικά, το σήμα εισόδων ενισχύεται με κέρδος 10 από ένα instrumentation amplifier INA333 και μετά περνάει από υψηλερατό φίλτρο αποκοπής συχνοτήτων 0.05Hz - 0.67Hz και χαμηλοπερατό φίλτρο συχνοτήτων 40Hz - 150Hz



Σχήμα 3.2: Block διαγραμμα σήματος

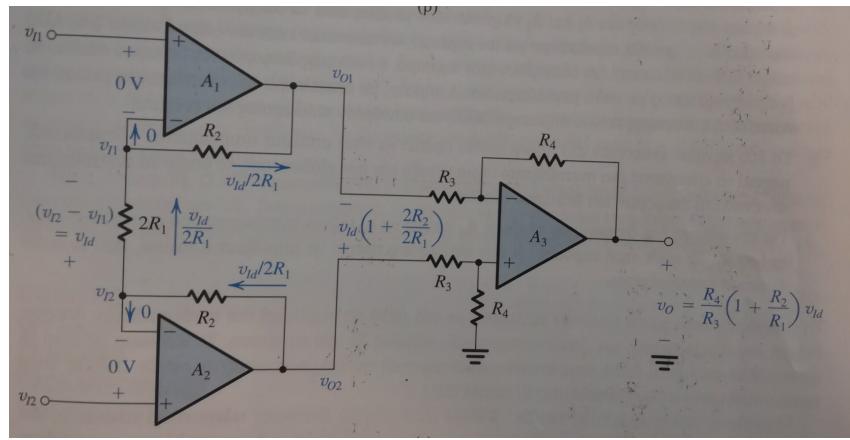
3.1.2 Ανάλυση κυκλώματος

Instrumentation Amplifier

Το σήμα που λαμβάνεται από τα ηλεκτρόδια είναι της τάξεως των μερικών mV (<5mV). Για να μπορέσουμε να μετρήσουμε σωστά τη διαφορά τους (θυμίζω Lead I = LA - RA) χρησιμοποιύμε instrumentation amplifier (INA).

Το κύκλωμα του INA αποτελείται από δύο στάδια. Το πρώτο σχηματίζεται από τους τελεστικούς ενισχυτές A_1 και A_2 και τις σχετικές αντιστάσεις, ενώ το δεύτερο είναι διαφορικός ενισχυτής που σχηματίζεται από τον τελεστικό A_3 και τις τέσσερις επιπλέον αντιστάσεις. Παρατηρήστε πως οι A_1 και A_2 χρησιμοποιούν μη αναστρέψουσα συνδεσμολογία και κατά συνέπεια επιτυγχάνουν κέρδος $1 + \frac{R_2}{R_1}$. Ο διαφορικός ενισχυτής στο δεύτερο στάδιο λειτουργεί με το σήμα διαφοράς $(1 + \frac{R_2}{R_1})(v_{I2} - v_{I1}) = (1 + \frac{R_2}{R_1})v_{Id}$ και έχω έξοδο $v_o = \frac{R_4}{R_3}(1 + \frac{R_2}{R_1})v_{Id}$. Προφανώς το κέρδος είναι $A_d = \frac{R_4}{R_3}(1 + \frac{R_2}{R_1})$. [3]

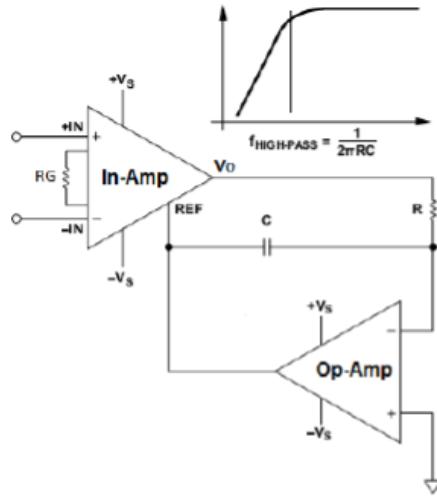
Πρακτικά το INA έρχεται σε μορφή έτοιμου chip και ο χρήστης μπορεί να επιλέξει το κέρδος βάζοντας την αντίστοιχη R_1 , με όλες τις υπόλοιπες αντιστάσεις σταθερές, έχοντας κέρδος $G = 1 + \frac{100k}{R_1}$.



Σχήμα 3.3: Εσωτερικό instrumentation amplifier

Τυπικερατό φίλτρο

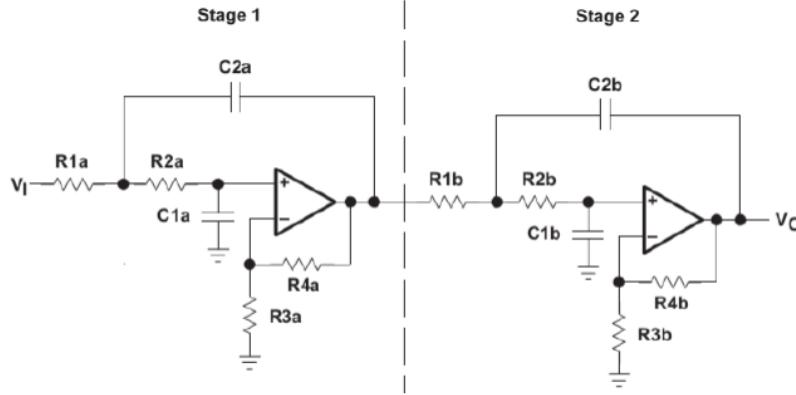
Το σήμα εξόδου από το INA περνάει από το υψηλερατό φίλτρο και η έξοδος του ανατροφοδοτείται στο pin REF, επιτρέποντας έτσι τον ακριβή έλεγχο των επιπέδων και απλοποιώντας τον σχεδιασμό κυκλωμάτων γρήγορης ανάκαμψης.



Σχήμα 3.4: Instrumentation amplifier με ανατροφοδότηση high pass filter

Χαμηλοπερατό φίλτρο

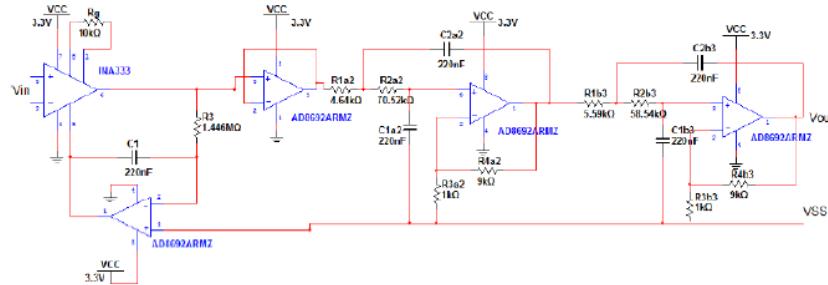
Για χαμηλοπερατό φίλτρο χρησιμοποιήθηκε φίλτρο Butterworth 4^{ης} τάξης με αρχιτεκτονική Sallen-Key. Επιλέχθηκε μια ενεργή τοπολογία για καλύτερη απόδοση και μειωμένη πολυπλοκότητα σε σύγκριση με μια παυθητική. [4]



Σχήμα 3.5: Low pass filter

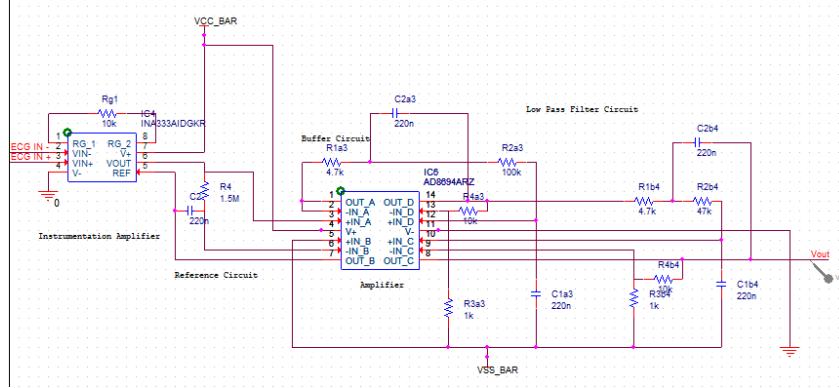
3.1.3 Σχεδίαση κυκλωμάτος

Αφού έχουμε αναλύσει τη λειτουργία του αισθητήρα μπορούμε να σχεδιάσουμε το κύκλωμα στη Cadence. Προσθέτουμε το κύκλωμα του INA με το υψηπερατό φίλτρο, προσθέτουμε ένα buffer για απομόνωση των δύο κυκλωμάτων και τέλος το χαμηλοπερατό φίλτρο. Το κύκλωμα της συσκευής Bitalino μοιάζει κάπως έτσι:



Σχήμα 3.6: Κύκλωμα συσκευής Bitalino

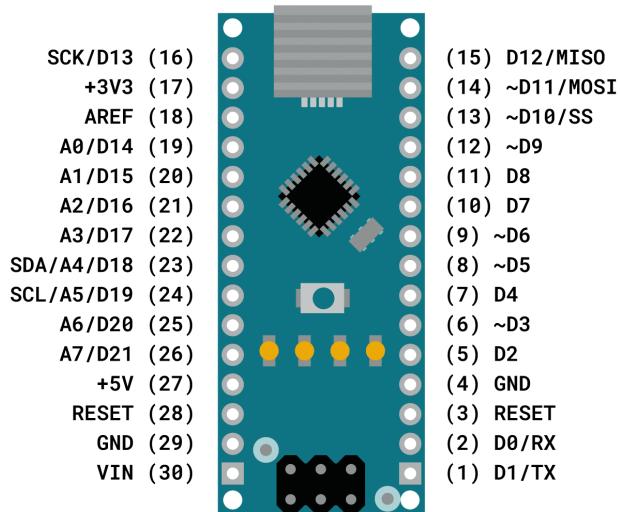
Για απλοποίηση της κατασκευής και πιο εύκολη αντιστοίχηση στο πραγματικό κύκλωμα, οι τιμές των αντιστάσεων στρογγυλοποιήθηκαν και εισήγαγα στο Cadence την πραγματική μορφή των τελεστικών.



Σχήμα 3.7: Τελική σχεδίαση κυκλώματος

3.2 Arduino nano

Για την επεξεργασία των μετρήσεων χρησιμοποιήθηκε το Arduino nano. Χρησιμοποιεί για κύριο επεξεργαστή τον ATmega328 στα 16Mhz.



Σχήμα 3.8: Arduino nano pinout

Χρειάζεται μόλις 5V για τη λειτουργία του και 40mA για κάθε pin που χρησιμοποιείται. Η πλακέτα χρειάζεται 19mA στη λειτουργία της. [5]

Πίνακας 3.1: Χαρακτηριστικά Arduino nano

Microcontroller	ATmega328
Architecture	AVR
Operating Voltage	5V
Flash Memory	32 KB of which 2 KB used by bootloader
SRAM	2KB
Clock Speed	16MHz
Analog I/O Pins	8
EEPROM	1KB
DC Current per I/O Pin	40mA (IO pins)
Digital Input / Output Pins	22
PWM Pins	6
Power Consumption	19 mA
PCB Size	18 x 45 mm
Weight	7 g
Product Code	A000005

3.2.1 Micro SD Module

Για τη αποθήκευση των τιμών χρησιμοποιούμε τη πλακέτα επέκτασης DFRobot MicroSD card. Η συγκεκριμένη χρησιμοποιεί πρωτόκολλο επικοινωνίας SPI το οποίο ενδείκνυται για τη μεταφορά δεδομένων.



Σχήμα 3.9: Μονάδα χρήσης MicroSD

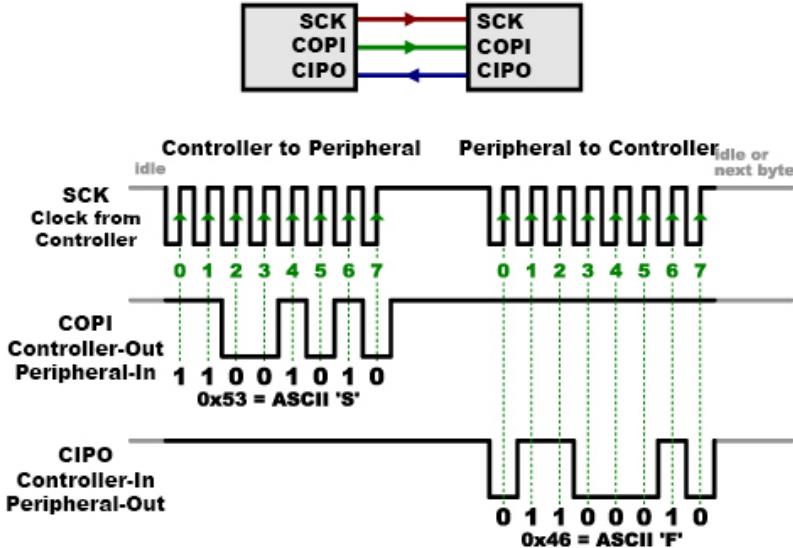
Πρωτόκολλο επικοινωνίας SPI

Το καλοκαίρι του 2020 πάρθηκε πρωτοβουλία από πολλούς οργανισμούς για τη αλλαγή των ονομασιών Master/Slave σε Controller/Peripheral μαζί με πολλά από τα ακρωνύμια τους. Ακολουθεί μια λίστα με τις αλλαγές για εύκολη αντιστοίχιση. [6]

Παλιά Ονομασία	Καινούργια Ονομασία	Περιγραφή
MOSI – Master Out Slave In	SDO – Serial Data Out	Σήμα εξόδου όπου το σήμα στέλνεται σε μια άλλη συσκευή
MISO – Master In Slave Out	SDI – Serial Data In	Σήμα εισόδου όπου το σήμα λαμβάνεται από μια άλλη συσκευή
SS – Slave Select	CS – Chip Select	Ενεργοποιείται από τον Controller για να ξεκινήσει τη επικοινωνία με το Peripheral
MOMI – Master Out Master In	COPI - Controller Out Peripheral In	Για συσκευές που μπορούν να είναι είτε Controller είτε Peripheral. Στέλνει δεδομένα αν λειτουργεί σαν Controller και λαμβάνει αν λειτουργεί σαν Peripheral
SOSI – Slave Out Slave In	CIPO - Controller In Peripheral Out	Για συσκευές που μπορούν να είναι είτε Controller είτε Peripheral. Στέλνει δεδομένα αν λειτουργεί σαν Peripheral και λαμβάνει αν λειτουργεί σαν Controller
SCK ή CLK – Serial Clock	SCK ή CLK – Serial Clock	Σήμα χρονισμού

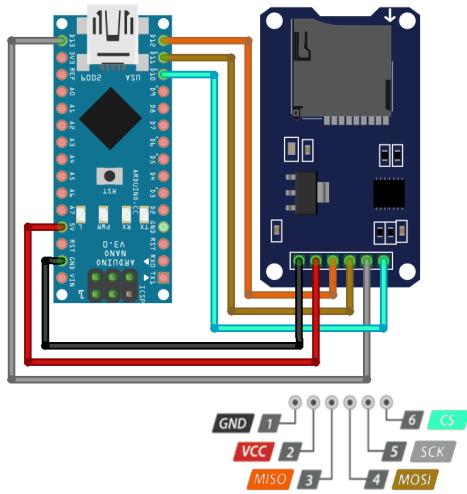
Πίνακας 3.2: Αντιστοίχιση ονομασιών σημάτων SPI

Το πρωτόκολλο SPI είναι σύγχρονο. Αυτό σημαίνει πως έχει ξεχωριστή γραμμή για ρολόι και ξεχωριστή για τα δεδομένα, τα οποία διαβάζονται κάθε φορά το ρολόι βρίσκεται στη θέση προς τα πάνω. Η συσκευή που ορίζει το ρολόι ονομάζεται Controller και αυτή που δέχεται Peripheral. Όταν δεδομένα στέλνονται από το Controller τα στέλνει στη γραμμή COPI και αντίστοιχα όταν λαμβάνει από τη γραμμή CIPO. Για να μπορώ να επικοινωνήσω το CS πρέπει να είναι στη θέση 0.[7]



Σχήμα 3.10: Λειτουργία σημάτων SPI

Πέρα από τους ακροδέκτες που περιγράφηκαν παραπάνω, υπάρχουν και οι ακροδέκτες τροφοδοσίας 5V και GND. Αυτούς τους συνδέουμε με τους αντίστοιχους του Arduino. Το Arduino έχει προκαθορισμένους ακροδέκτες για χρήση σαν MOSI, MISO, SCK και SS ανάλογα την έκδοση του. Για την έκδοση Nano αυτοί είναι οι D11, D12, D13 και D10 αντίστοιχα. Η τελική συνδεσμολογία παρουσιάζεται πιο κάτω.



Σχήμα 3.11: Σύνδεση SD module με Arduino nano

3.2.2 Bluetooth module

Για τη μετάδοση δεδομένων χρησιμοποιήθηκε η τεχνολογία Bluetooth. Έγινε μια προσπάθεια για χρήση τεχνολογίας Bluetooth Low Energy (BLE) αλλά απορρίφθηκε λόγω της

αργής μετάδοσης των δεδομένων.

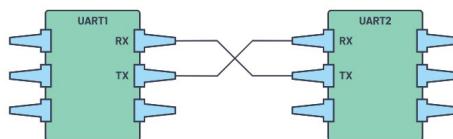
Στη συσκευή μας συνδέσαμε πλακέτα επέκτασης HC-05, δίνοντας στο Arduino ικανότητα Bluetooth 4.0. Το HC-05 υποστηρίζει προτόκολλο UART για επικοινωνία και έχει τους εξής ακροδέκτες:

- **Enable/Key:** Χρησιμοποιείται για τη επιλογή λειτουργίας της πλακέτας. Σε επιλογή LOW είναι σε λειτουργία αποστολής δεδομένων και HIGH σε λειτουργία AT Command Mode. Μπορεί να έχω επιλογή HIGH πατώντας το κουμπί
- **Vcc:** Ακροδέκτης τροφοδοσίας 5V
- **Ground:** Ακροδέκτης γείωσης
- **TXD:** Εκπομπός σειριακών δεδομένων. Δεδομένα που λαμβάνονται μέσω Bluetooth στέλνονται μέσω αυτού του ακροδέκτη στο Arduino.
- **RXD:** Παραλήπτης σειριακών δεδομένων. Δεδομένα στέλνονται από το Arduino λαμβάνονται σε αυτόν τον ακροδέκτη και αποστέλνονται μέσω Bluetooth.

Πρωτόκολλο επικοινωνίας UART

Το UART (Universal Asynchronous Receiver/Transmitter), είναι ένα από τα πιο διαδεδομένα πρωτόκολλα επικοινωνίας συσκευών.

Για επικοινωνία UART χρειάζονται δύο ακροδέκτες, RX και TX. Το TX εκπέμπει δεδομένα και το RX λαμβάνει. Έτσι για σωστή επικοινωνία συνδέουμε το RX με το TX. [8]

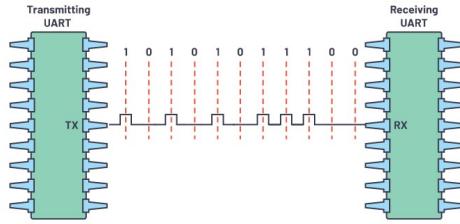


Σχήμα 3.12: Σύνδεση UART

Το κάθε bit στέλνεται ξεχωριστά μέσω σειριακής μετάδοσης και τα πακέτα ξεχωρίζουν από τα start και end bits.

Start Bit (1 bit)	Data Frame (5 to 9 Data Bits)	Parity Bits (0 to 1 bit)	Stop Bits (1 to 2 bits)
----------------------	----------------------------------	-----------------------------	----------------------------

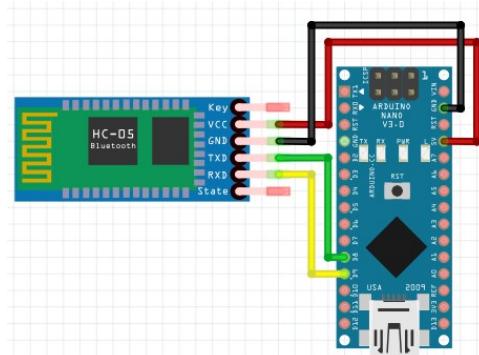
Σχήμα 3.13: Πακέτο UART



Σχήμα 3.14: Αποστολή δεδομένων UART

Όπως λέει και το όνομα το UART δεν έχει κάποιο ρολόι αλλά τα δεδομένα στέλνονται σύμφωνα με το εσωτερικό ρολόι της συσκευής.

Για σύνδεση του Bluetooth module με το Arduino, συνδέουμε τη τροφοδοσία και τη γείωση στους αντίστοιχους ακροδέκτες. Τα TX και RX συνδέονται με τα αντίστροφα του Arduino. Όμως επειδή οι ακροδέκτες TX και RX χρησιμοποιούνται για επικοινωνία του Arduino για φόρτωση λογισμικού και αποστολή δεδομένων τα τοποθετούμε στους ακροδέκτες D8,9 αντίστοιχα. Πρακτικά μετά από πειράματα η συσκευή σε κατάσταση λειτουργίας διάρκεσε 23 ώρες.



Σχήμα 3.15: Σύνδεση bluetooth module

3.2.3 Τροφοδοσία

Για τροφοδοσία του arduino χρησιμοποιήθηκε power bank 2700mAh εμπορίου.

Περιλαμβάνεται από τη μπαταρία και voltage regulator εξόδου 5V, τροφοδοτώντας έτσι το arduino με σταθερή τάση. Επίσης παίρνουμε τη άκρη της μπαταρίας σε pin για να υπολογίσουμε ποσοστό μπαταρίας.

Για να δούμε αν αρκεί πρέπει να υπολογίσουμε πόση κατανάλωση έχει το κάθε περιφερικό. Το arduino έχει από μόνο του 19mA, το bluetooth module 40mA, το micro SD στη χειρότερη περίπτωση καταναλώνει συνολικά 200mA και τέλος τα pin από 20mA. Έτσι συνολικά στη χειρότερη περίπτωση έχουμε $19 + 40 + 200 + 3 * 20 = 320$ mA. Άρα μια μπαταρία με 2700mAh θα διαρκέσει $2700 \text{ mAh} \div 320 \text{ mA} = 8.4$ ώρες.

Κεφάλαιο 4

Ανάλυση Λογισμικού

4.1 Λογισμικό Ενσωματωμένου

Όλες οι πλακέτες Arduino nano που χρησιμοποιήθηκαν είναι open source και το IDE(Integrated Development Environment) τους είναι βασισμένο στις γλώσσες C και C++. [9] Οι βιβλιοθήκες τους είναι βασισμένες στο the wire project, [10] το οποίο αντίστοιχα είναι ένα open source πλαίσιο που προσθέτει ένα επιπρόσθετο επίπεδο C++ για πιο απλό προγραμματισμό ενσωματωμένων.

4.1.1 Εισαγωγή Βιβλιοθηκών

Βιβλιοθήκες για εισαγωγή κάρτας

- **SPI.h:** Βιβλιοθήκη για τη επικοινωνία μεταξύ συσκευών με το πρωτόκολλο SPI.
- **SD.h:** Βιβλιοθήκη για δημιουργία/ανάγνωση/εγγραφή αρχείων στη κάρτα SD.

Βιβλιοθήκη για επεξεργασία σήματος

- **Filters.h:** Βιβλιοθήκη για ψηφιακή επεξεργασία σήματος σε πραγματικό χρόνο

Βιβλιοθήκη για Bluetooth

- **SoftwareSerial.h:** Βιβλιοθήκη για επικοινωνία μεταξύ συσκευών με το πρωτόκολλο UART

Listing 4.1: Εισαγωγή Βιβλιοθηκών

```
1 //Sd libraries-----
2 #include <SPI.h>
3 #include <SD.h>
4 //-----
5 //Filter libraries-----
```

```

6   #include <Filters.h>
7   //-----
8   //Bluetooth libraries-----
9   #include <SoftwareSerial.h>
10  //-----

```

[11]

4.1.2 Αρχικοποίηση Μεταβλητών

Ορίζουμε ακροδέκτες για UART επικοινωνία με το Bluetooth module. Χρειάζεται να ορίσουμε άλλους από τους ακροδέκτες TX, RX του Arduino για να μπορούμε να φορτώνουμε λογισμικό και να επικοινωνούμε με αυτό.

Listing 4.2: Αρχικοποίηση Bluetooth

```
1   SoftwareSerial mySerial(8, 9); // RX, TX
```

Για να ορίζουμε μεταβλητές για ευκολία εκτέλεσης των εντολών:

int flag=0: Σημαία για έλεγχο παραλαβής δεδομένων

int startFlag=0: Σημαία για αρχή εγγραφής και αποστολής δεδομένων

Float input,2,3: Αποτέλεσμα μέτρησης εξόδου του κυκλώματος ECG

AnalogPin1,2,3: Ορισμός των pin εισόδου.

buttonPin: Ορισμός pin κουμπιού

LowFrequency: Ορίζουμε συχνότητα αποκοπής 0,5Hz για πιο ομαλή κυματομορφή

highpassFilter: Μεταβλητή τύπου FilterOnePole η οποία είναι φίλτρο RC με συχνότητα αποκοπής που ορίζεται από την μεταβλητή LowFrequency.

myFile: Μεταβλητή τύπου File όπου δημιουργούμε το αρχείο και γράφουμε τις τιμές

filename: Μεταβλητή στη οποία θα ορίσουμε το όνομα του myFile

currentMillis: χρόνος λειτουργίας Arduino από τη εκκίνηση της πλακέτας

values: String στο οποίο τοποθετούνται οι τιμές μέτρησης για αποστολή με Bluetooth.

Listing 4.3: Αρχικοποίηση διάφορων μεταβλητών

```

1  int flag=0;
2  int startFlag=0;
3
4  float input1=0;
5  float input2=0;
6  float input3=0;
7  long sendTime=0;
8  float batteryLvl=0;
9  const int analogPin1 = A1;
10 const int analogPin2 = A2;
11 const int analogPin3 = A3;
12 const int batteryPin = A5;
13
14 float LowFrequency = 0.5;
15 FilterOnePole highPassFilter(HIGHPASS,LowFrequency);
16

```

```

17 File myFile;//SD card file
18 String fileName;
19 int num=0;
20 unsigned long currentMillis=0;
21 String values;

```

4.1.3 Συνάρτηση εκκίνησης

Η συνάρτηση `setup()` εκτελείται πάντα μια φορά κατά την εκκίνηση του μικροεπεξεργαστή. Όλες οι αρχικοποιήσεις γίνονται εδώ.

Serial.begin(38400): Θέτουμε το ρυθμό επικοινωνίας (Baud rate) του Arduino στα 9600 bits per second.

mySerial.begin(38400): Θέτουμε το ρυθμό επικοινωνίας (Baud rate) του Bluetooth στα 9600 bits per second. Δεν έχει νόημα να αυξήσουμε το ρυθμό επικοινωνίας λόγω του ότι το bluetooth στα κινητά είναι σταθερό στα 9600.

setupSD(): Η αρχικοποίηση του SD module έγινε σε ξεχωριστή συνάρτηση για πιο καθαρό κώδικα και πιο εύκολο εντοπισμό σφαλμάτων.

Listing 4.4: Συνάρτηση εκκίνησης

```

1 void setup() {
2     // Open serial communications and wait for port to open:
3     Serial.begin(38400);
4
5     // set the data rate for the SoftwareSerial port
6     mySerial.begin(38400);
7
8     if(setupSD()) {
9         Serial.println("SD setup success");
10    }else{
11        Serial.println("SD setup failed");
12    }
13    analogReference(DEFAULT);
14 }

```

setupSD()

Η `SD.begin()` αρχικοποιεί τη βιβλιοθήκη και τη κάρτα και επιστρέφει `true` αν είναι επιτυχής. Προφανώς αν επιστρέψει `false` έχουμε κάνει λάθος στη σύνδεση του module.

Ορίζω το όνομα `ECG_αριθμός.txt` για να μπορώ να έχω μετρήσεις πολλών ασθενών χωρίς να αντικαθίσταται το αρχείο, και δημιουργώ αρχείο με αυτό το όνομα με τη εντολή `SD.open`. Ελέγχω αν μπορώ να το ανοίξω και αν δημιουργήθηκε σωστά, γράφω σε αυτό τη πρώτη γραμμή που είναι `Time, ECG1, ECG2, ECG3`.

Listing 4.5: Συνάρτηση αρχικοποίησης κάρτας SD

```

1 bool setupSD() {

```

```

2 //Setup SD start
3 Serial.print("Initializing SD card... ");
4 if(SD.begin()) {
5   fileName = "ECG_";
6   fileName += num;
7   fileName += ".txt";
8   Serial.print("Creating file with name");
9   Serial.println(fileName);
10  myFile = SD.open(fileName, FILE_WRITE);
11  // if the file opened okay, write to it:
12  if (myFile) {
13    //Serial.print("Writing to test.txt...");
14    myFile.println("Time , ECG1, ECG2, ECG3");
15  }
16  return true;
17 }else{
18  return false;
19 }
20 }
```

4.1.4 Συνάρτηση λειτουργίας

Η συνάρτηση `loop()` εκτελείται συνεχώς όσο λειτουργεί ο μικροεπεξεργαστής. Η γενική λογική της συνάρτησης είναι παίρνω τιμές, τις στέλνω με Bluetooth και τις γράφω στη κάρτα SD. Επίσης κάνω και έλεγχο αν πατήθηκε το κουμπί

Listing 4.6: Τιμές μετρήσεων και central

```

1 void loop()
2 {
3   input1 = highPassFilter.input(analogRead(analogPin1))*5/1023; // read the
               input pin1
4   input2 = highPassFilter.input(analogRead(analogPin2))*5/1023; // read the
               input pin2
5   input3 = highPassFilter.input(analogRead(analogPin3))*5/1023; // read the
               input pin3
6
7   //Write values to each place
8   values="";
9   values+=String(input1);
10  values+=", ";
11  values+=String(input2);
12  values+=", ";
13  values+=String(input3);
```

Διαβάζω αν έχει έρθει κάποια εντολή από το android με τη εντολή `read()`. Αν έχω παραλάβει το χαρακτήρα 1 τότε σταματάω τη λειτουργία και αποθηκεύω όλα τα δεδομένα που έχω πάρει μέχρι τώρα με τη εντολή `flush()` και κλείνω το αρχείο. Αν έχω παραλάβει το χαρακτήρα 2 τότε ξεκινάω τη λειτουργία θέτοντας το `startFlag=1` και ελέγχουμε αν υπάρχει κάρτα SD. Η συνάρτηση `setupSD` αρχικοποιεί και ανοίγει ένα καινούργιο αρχείο και αν επιστρέψει `true` μπορούμε να αρχίσουμε τη κανονική λειτουργία.

Listing 4.7: Έλεγχος παραλαβής εντολών

```

1  if(mySerial){//Receive commands from phone
2    flag=mySerial.read();
3    if(flag==49){//Ascii code 49 = number 1
4      //Stop recording
5      startFlag=0;
6      mySerial.println("Please wait, saving");
7
8      myFile.flush(); //5ms
9      myFile.close();
10
11     mySerial.println("Saved, you can eject the SD card");
12   }
13   if(flag == 50){
14     //Start recording
15     startFlag=1;
16     //Make sure sd card is inserted
17     while(!setupSD()){
18       mySerial.println("Insert SD Card");
19       delay(2000);
20     }
21     //SetupSD initializes SD card if true
22     mySerial.println("Starting...");
23   }
24 }updateCounter();
25 }
```

Αν `startFlag=1` είναι ασφαλές να αρχίσουμε να γράφουμε και να στέλνουμε μετρήσεις. Αρχικά στέλνουμε τις μετρήσεις με τη συνάρτηση `sendBlue(2,input1,input2,input3,dif)` και μετά τις αποθηκεύω με τη `myFile.println()`. Στέλνω τον χρόνο ανάμεσα σε δύο διαδοχικές μετρήσεις και τις προσθέτω στο android, αποφεύγοντας να στείλω τιμή long που είναι 32bits. Επίσης κάθε ένα λεπτό (ή 60000 milliseconds) διαβάζω τη τάση στα άκρα της μπαταρίας που τροφοδοτεί το arduino και αν είναι μικρότερη από 3V στέλνω ειδοποίηση για χαμηλή μπαταρία.

Listing 4.8: Αποστολή δεδομένων με bluetooth και εγγραφή σε κάρτα SD

```

1
2  if(startFlag){// If start flag = 1 sd card is inserted and its safe to
3    start writing
4
5    dif = micros()-tic;
6    sendBlue(2,input1,input2,input3,dif); //7450us ->7.45ms
7    tic = micros();
8
9    values = String(input1);
10   values+= ", ";
11   values+= String(input2);
12   values+= ", ";
13   values+= String(input3);
14   values+=", ";
15   values+=sendTime;
16   myFile.println(values);
17
18   i++;
```

```

18     if(i==1000) {
19         i=0;
20         myFile.flush(); //5ms
21     }
22     if((sendTime-currentMillis)>60000){//1 minuite
23     currentMillis=millis();
24     if((analogRead(batteryPin)*5/1023)<3){
25         sendBlue(5,5,5,5,5);
26     }
27 }
28 }
```

Για αποστολή δεδομένων δημιουργήθηκε η συνάρτηση sednBlue. Το πρώτο byte είναι χαρακτηρισμός των δεδομένων (2: Μετρήσεις, 3: Ασφαλής αποθήκευση, 4: Μη ύπαρξη κάρτας, 5: Χαμηλή μπαταρία) που ακολουθούν και για κάθισ μέτρηση τοποθετούμε τα most signifigant bites στο πρώτο σημείο και στο δεύτερο τα υπόλοιπα. Τα δεδομένα θα τα ξανασυνδιάσουμε μετά, στη εφαρμογή android

Listing 4.9: Συνάρτηση αποστολής δεδομένων

```

1
2 void sendBlue(int what, int x, int y, int z, long t){
3     measurementsArray[0] = what;    //
4     measurementsArray[1] = x >> 8;//MSB x
5     measurementsArray[2] = x;      //LSB x
6     measurementsArray[3] = y >> 8;//MSB y
7     measurementsArray[4] = y;      //LSB y
8     measurementsArray[5] = z >> 8;//MSB z
9     measurementsArray[6] = z;      //LSB z
10    measurementsArray[7] = t >> 8;//MSB z
11    measurementsArray[8] = t;      //LSB z
12    mySerial.write(measurementsArray, sizeof(measurementsArray));
13 }
```

4.2 Λογισμικό εφαρμογής

Η εφαρμογή Android σχεδιάστηκε με σκοπό να είναι όσο πιο απλή στη σύνδεση, να δείχνει καθαρά το καρδιογράφημα και με αρκετές πληροφορίες για αναγνώριση από το γιατρό. Γιαυτό η εφαρμογή συνδέεται αυτόματα με τη συσκευή, χωρίς να χρειαστεί να τη ψάξει και να γίνει η σύνδεση καθώς έχει τη φυσική διεύθυνση του bluetooth module και τα δεδομένα εμφανίζονται σε πραγματικό χρόνο σε μορφή γραφικής με τη βοήθεια της βιβλιοθήκης MPAndroidChart.

4.2.1 Λειτουργία Bluetooth

Για δύο συσκευές να μπορούν να επικοινωνούν μεταξύ τους πρέπει να δημιουργήσουν ένα κανάλι επικοινωνίας. Η μια συσκευή κάνει το εαυτό της διαθέσιμη για εισερχόμενες αιτήσεις. Μια άλλη βρίσκει τη διαθέσιμη συσκευή και όταν δεχθεί τη αίτηση οι δύο συσκευές έχουν γίνει paired. Έπειτα ξεκινάει μια διαδικασία όπου οι δύο συσκευές ανταλλάζουν κλειδιά (bonding process) και πληροφορίες.

Οι κλάσεις που θα χρησιμοποιηθούν είναι:

- **MainActivity** δημιουργεί τη επιφάνεια που εμφανίζονται τα δεδομένα.
- **Connected Thread** όταν γίνει η σύνδεση λαμβάνει και αποστέλλει τα δεδομένα.
- **Ardconn** διαχειρίζεται τη σύνδεση με τη συσκευή.

Και η γενική λειτουργία της εφαρμογής είναι ως εξής:

1. Ψάξε για συσκευές Bluetooth
2. Δημιουργία RFCOMM καναλιού
3. Σύνδεση σε συσκευή
4. Παραλαβή δεδομένων
5. Παρουσίαση δεδομένων

4.2.2 Αρχικοποίηση

Πριν καν ξεκινήσουμε το κώδικα μας πρέπει να πάρουμε ορίσουμε τι δικαιώματα θα έχει η εφαρμογή από τη συσκευή μας. Στο AndroidManifest.xml δίνουμε άδεια για χρήση Bluetooth

Listing 4.10: Άδειες χρήσης Bluetooth και τοποθεσίας συσκευής

```
1 <uses-permission android:name="android.permission.BLUETOOTH"/>
2 <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
3 <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
4 <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

Πρίν αρχίσουμε τη σύνδεση πρέπει πρώτα να έχουμε πρόσβαση στο bluetooth του κινητού. Αυτό γίνεται μέσω ενός bluetooth adapter το οποίο μας δίνει έλεγχο του. Αν επιστρέψει null η συσκευή δεν υποστηρίζει bluetooth. Έπειτα ζητάμε να ενεργοποιήσουμε το bluetooth της συσκευής. Αν είναι κλειστό θα εμφανιστεί παράθυρο που ζητάει τη ενεργοποίηση του.

Listing 4.11: Πρόσβαση σε bluetooth adapter ενεργοποίηση του..

```
1     BluetoothAdapter bta;
2     bta = BluetoothAdapter.getDefaultAdapter();
3     if(!bta.isEnabled()){
4         setRequestEnableBt();
5     }else{
6         initiateBluetoothProcess();
7     }
8
9     private void setRequestEnableBt(){
10    Intent enableBTIntent= new
11        Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
12    startActivityForResult(enableBTIntent,REQUEST_ENABLE_BT);
13 }
```

4.2.3 Εύρεση συσκευών

Αφού έχουμε ενεργοποιήσει το bluetooth στη συσκευή, μπορούμε να βρούμε συσκευές, χρησιμοποιώντας τον BluetoothAdapter.

Σε αυτή τη φάση η συσκευή μας ψάχνει για συσκευές και ζητάει πληροφορίες από αυτές. Αν μια συσκευή είναι διαθέσιμη απαντάει με τα στοιχεία της όπως το όνομα της συσκευής, τη τάξη της και τη φυσική της διεύθυνση. Στη περίπτωση μας αφού θέλουμε να συνδεθούμε μόνο σε μία συγκεκριμένη συσκευή της οποίας γνωρίζουμε τη φυσική της διεύθυνση, μπορούμε να αγνοήσουμε αυτό το βήμα.

4.2.4 Σύνδεση σε συσκευή

Για σύνδεση με τη συσκευή ανοίγουμε ένα BluetoothSocket, το οποίο ακούει για εισερχόμενα δεδομένα. Δημιουργούμε socket με toCreateInsecureRfcommSocketToServiceRecord και προσπαθούμε να συνδεθούμε με το connect και αν έρχονται δεδομένα παραμένει ανοικτή. Αν για κάποιο χρονικό διάστημα δεν έρθουν δεδομένα αυτή κλείνει.

Listing 4.12: Δημιουργία socket για παραλαβή δεδομένων.

```
1 //Attempt to connect to bluetooth module
```

```

2  BluetoothSocket mmSocket = null;
3  BluetoothSocket tmp = null;
4  mmDevice = bta.getRemoteDevice(MODULE_MAC);
5  toastMessage("Searching...");
6  //Create socket
7  try {
8      tmp=mmDevice.createInsecureRfcommSocketToServiceRecord(MY_UUID);
9      mmSocket=tmp;
10     mmSocket.connect();
11     Log.i("[BLUETOOTH]","Connected to: "+mmDevice.getName());
12     toastMessage("Connected to: "+mmDevice.getName());
13 } catch (IOException e) {
14     try {
15         mmSocket.close();
16     }catch (IOException i){
17         i.printStackTrace();
18     }
19 }

```

4.2.5 Παραλαβή δεδομένων

Αφού έχουμε συνδεθεί, μπορούμε να περιμένουμε για παραλαβή δεδομένων. Η κλάση InputStream διαχειρίζεται τις παραλαβές στο socket και μπορούμε να τις χρατάμε με τη εντολή read(curBuf).

Τα δεδομένα ξεχωρίζουν μεταξύ τους με το αρχικό byte σαν χαρακτηριστικό των δεδομένων που ακολουθούν. Προσθέτω κάθε byte που έρχεται σε πίνακα bytes μεγέθους 9 στοιχείων. Αν το πρώτο στοιχείο είναι 2, δηλαδή μετρήσεις, μετατρέπω κάθε δύο bytes 8 bit σε αριθμούς short 16bit. Αφού μαζέψω αρκετά στοιχεία, μέσω ενός handler στέλνω τους αριθμούς για να παρουσιαστούν.

Αν το πρώτο στοιχείο δεν είναι 2 έχουμε χάσει κάποιο στοιχείο και πρέπει να περιμένουμε το επόμενο 2 για να ξαναμπούν τα στοιχεία σε σωστή σειρά.

Listing 4.13: Διάβασμα δεδομένων

```

1  mmSocket=socket;
2  InputStream tmpIn = null;
3  OutputStream tmpOut = null;
4  this.uih=uih;
5  Log.i("[THREAD-CT]","Creating thread");
6  try {
7      tmpIn=socket.getInputStream();
8      tmpOut=socket.getOutputStream();
9  } catch (IOException e) {
10     Log.e("[THREAD-CT]","Error:"+ e.getMessage());
11 }
12 mmInStream=tmpIn;
13 mmOutStream=tmpOut;
14 }
15

```

```

16 int bytesAvailable = mmInStream.available();
17 if (bytesAvailable > 0) {
18     byte[] curBuf = new byte[bytesAvailable];
19     mmInStream.read(curBuf);
20     for (byte b : curBuf) {
21         if (b == 2 && iterator == num_of_packets) { //start of message
22             short[] shorts = new short[bytes.length/2];
23             // to turn bytes to shorts as either big endian or little endian.
24             byte[] modifiedBytes = Arrays.copyOfRange(bytes, 1, bytes.length);
25             //Discard first byte
26             ByteBuffer.wrap(modifiedBytes).order(ByteOrder.BIG_ENDIAN).asShortBuffer().get(shorts);
27             //After 100 packets send to ardconn to view
28             if(measurements_num==max_measurments*4){
29                 measurements_num=0;
30                 //Send measurements to Ardconn
31                 Message msg = new Message();
32                 msg.what = RESPONSE_MESSAGE;
33                 msg.obj =measurements;
34                 uih.sendMessage(msg);
35             }
36             //Add shorts to a bigger array
37             System.arraycopy(shorts, 0, measurements, measurements_num,
38                             shorts.length);
39             //Added measurements, increase point on array
40             measurements_num=measurements_num+4;
41             iterator = 0;
42             next2Flag=false;
43             bytes[iterator] = b;
44         } else if(iterator<num_of_packets) {
45             bytes[iterator] = b;
46         }else{ //if im here I lost the 2 for the start and i have to wait for
47             the next
48             Log.d("[THREAD-CT]","next2Flag enabled");
49             //Should have a flag to wait for the next 2
50             next2Flag=true;
51         }
52         if (next2Flag){
53             //wait untill next 2
54             Log.d("[THREAD-CT]","next2Flag enabled");
55             iterator=num_of_packets;
56         }else {
57             iterator++;
58         }
59     } catch (IOException e) {
60         break;
61     }

```

4.2.6 Παρουσίαση δεδομένων

Αφού πήραμε τα δεδομένα, τώρα μπορούμε να τα παρουσιάσουμε. Μέσω του handler δημιουργείται function το οποίο καλείται αυτόματα κάθε φορά που έρχονται καινούργια δεδομένα. Διαβάζοντας το χαρακτηριστικό του μηνύματος ή παρουσιάζουμε το μήνυμα σε μορφή toast message (σύντομο μήνυμα που εμφανίζεται στη οθόνη) ή παρουσιάζουμε τα δεδομένα σε μορφή γραφικής.

Τα δεδομένα έρχονται σε ένα μεγάλο array σε σειρά, τα ξεχωρίζουμε σε δισδιάστατο πίνακα, μετατρέπουμε τις τιμές σε Volt και προσθέτουμε τις χρονικές τιμές αφού πρώτα τις μετατρέψουμε σε δευτερόλεπτα. Τέλος πριν παρουσιάσουμε τις τιμές τις περνάμε από ένα χαμηλοπερατό φίλτρο για να απαλείψουμε τυχών θορύβους που δημιουργήθηκαν κατά τη μετάδοση.

Listing 4.14: Παρουσίαση δεδομένων

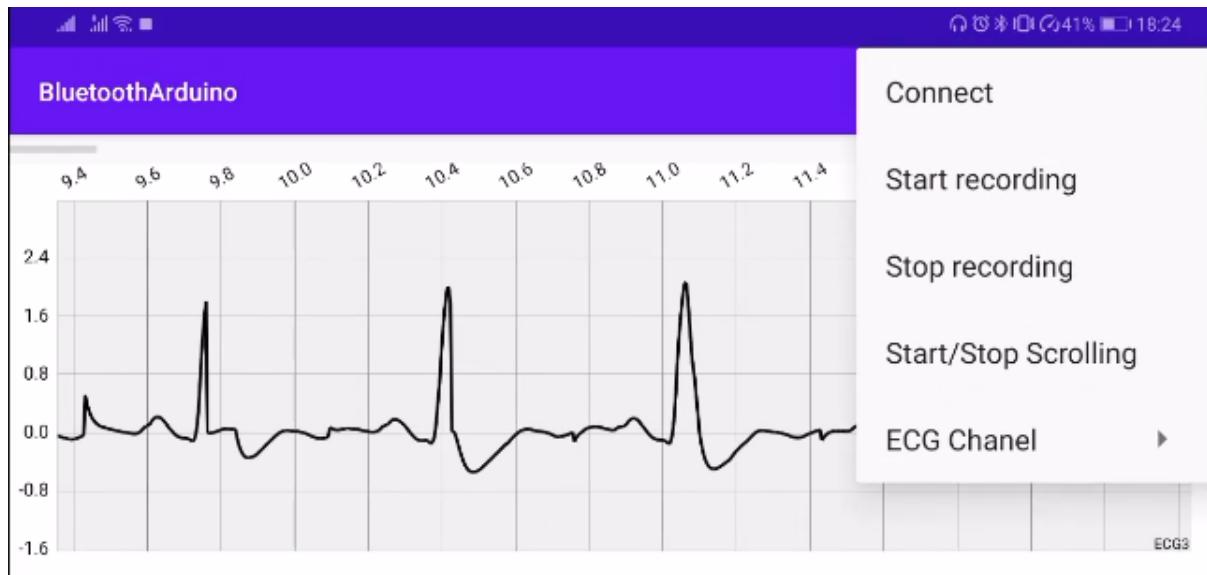
```
1 public void handleMessage(Message msg) {
2     if(msg.what == ConnectedThread.RESPONSE_MESSAGE) {
3         //Get message from connected thread
4         short[] txt = (short[]) msg.obj;
5         //Log.d("[Ardconn]","txt.length: "+txt.length);
6         //Separate to each 1,2,3 ecg
7         //Log.d("[Ardconn]","txt.length: "+txt.length);
8         for(int i=0;i<txt.length;i=i+4){
9             int j = i/4;
10            //Log.d("[THREAD-CT]","i: "+i+" j: "+j);
11            values[0][j] = (float)txt[i]*5/1023; //ECG1
12            values[1][j] = (float)txt[i+1]*5/1023; //ECG2
13            values[2][j] = (float)txt[i+2]*5/1023; //ECG3
14            if(j==0){
15                values[3][j]=(float)
16                    (values[3][values[3].length-1]+txt[i+3]/1000000.0); //Convert to s
17            }else{
18                values[3][j]=(float) (values[3][j-1]+txt[i+3]/1000000.0);
19                    //Convert to s
20            }
21            for (int i=0;i<3;i++){
22                values[i] = lowPassFilter(values[i]);
23            }
24            previewData(values);
25        }
26        if(msg.what == ConnectedThread.MESSAGE_MESSAGE) {
27            //Get battery lvl
28            String txt = (String) msg.obj;
29            toastMessage(txt);
30        }
31    }
32 }
```

Για να μπορούμε να παρουσιάσουμε τα δεδομένα χρειάζεται να τα βάλουμε σε ειδικά δημιουργημένες arrays από το library, LineDataSet και LineData. To LineDataSet δέχεται δεδομένα σε μορφή ArrayList και το LineData σε μορφή LineDataSet. Έτσι τα δεδομένα που έχουμε παραλάβει, τοποθετούνται σε ArrayList, αυτό σε ένα LineDataSet και τέλος

αυτό σε LineData. Με τη εντολή setData(LineData) μπορούμε να παρουσιάσουμε τα δεδομένα. Για να μην έχουμε πρόβλημα με μεγάλο όγκο δεδομένων κάθε 1000 μηδενίζουμε το κάθε ArrayList

Listing 4.15: Παρουσίαση δεδομένων

```
1 valuesECG1.add(new Entry(values[3][i],data[0][i]));
2 LineDataSet set1 = new LineDataSet(valuesECG1,"ECG1");
3 LineData lineData1 = new LineData(set1);
4 chart.setData(lineData1);
```



Σχήμα 4.1: Τελική εφαρμογή

Κεφάλαιο 5

Συναρμολόγηση

Αφού έχουμε αναλύσει πως λειτουργούν όλα τα κομμάτια τώρα πρέπει να τα βάλουμε μαζί. Σκοπός της πτυχιακής αυτής ήταν να τοποθετήσουμε το κύκλωμα σε διάτρητη πλακέτα και να το τοποθετήσουμε σε ειδικά σχεδιασμένο κουτί από 3D printer.

Δυστυχώς δεν κατάφερα να ολοκληρώσω το τελευταίο αυτό κομμάτι, γιατί τα εργαστήρια ήταν κλειστά λόγω της καραντίνας που έγινε για τον κορονοϊό το 2020. Περιορίστηκα στη κατασκευή του κυκλώματος σε δοκιμαστική πλακέτα για ένα αισθητήρα και ένα δεύτερο για δοκιμή της διάταξης σε διατρητική πλακέτα. Ακολουθεί μια περιγραφή συναρμολόγησης του κυκλώματος στη δοκιμαστική πλακέτα, φόρτωση λογισμικού στο Arduino και της εφαρμογής στο κινητό για εύκολη αναπαραγωγή (και βελτίωση) της πτυχιακής.

5.1 Υλικά κατασκευής

Για το κύκλωμα όμως χρειαστούμε:

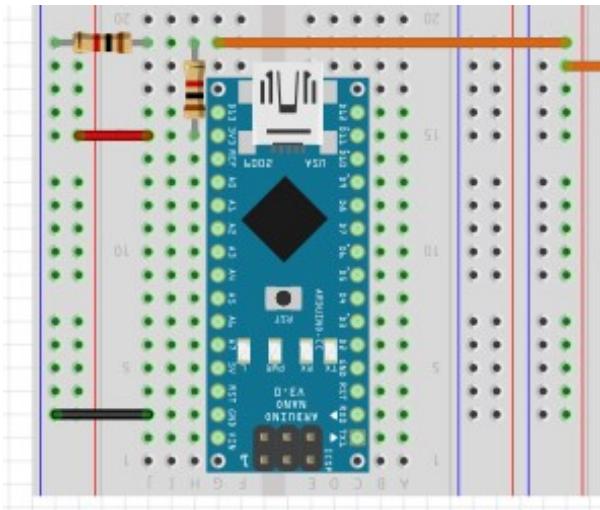
Εξάρτημα	Όνομα εξαρτήματος	Ποσότητα
Instrumentation Amplifier	INA333	3
Τελεστικός ενισχυτής	AD8694	3
VSSOP-8 to DIP-8 SMT Adapter	-	3
SOIC-14 to DIP-14 SMT Adapter	-	3
Αντίσταση 1K	-	6
Αντίσταση 4.7K	-	6
Αντίσταση 10K	-	9
Αντίσταση 47K	-	3
Αντίσταση 100K	-	3
Αντίσταση 1.5M	-	3
Πυκνωτής 220nF	-	15
Microcontroller	Arduino nano	1

SD Card expansion	DFRobot MicroSD card module	1
Bluuetooth module	HM-05	1
Αισθητήρες Ag/AgCl	-	7
Ακροδέκτες snap	-	7
Powerbank	-	1

Ο τελεστικός ενισχυτής και ο instrumentation amplifier έρχονται σε μορφή VSSOP και SOIC, προορισμένοι για χρήση σε PCB. Με τους adapters VSSOP-8 και SOIC-14 to DIP-8 μπορούμε να τα τοποθετήσουμε σε διάτρητη πλακέτα. Επίσης ωστε χρειαστούν breadboard και καλώδια σύνδεσης.

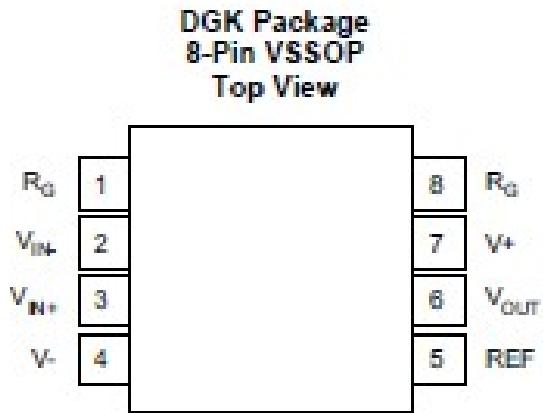
5.2 Συναρμολόγηση κυκλώματος

Τροφοδοτούμε το κύκλωμα από το Arduino. Αφού το τοποθετήσουμε στο breadboard πάρινουμε το pin 3.3V, και τροφοδοτούμε τη γραμμή +, δημιουργούμε ένα διαιρέτη τάσης για το virtual ground και η γραμμή - με το pin GND.

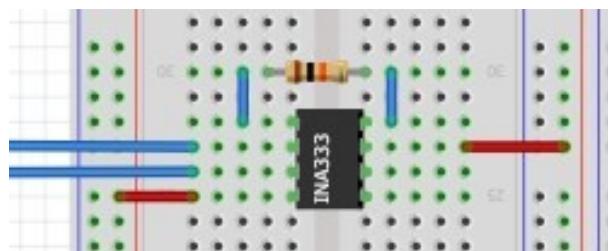


Σχήμα 5.1: Σύνδεση Arduino για τροφοδοσία κυκλώματος.
VCC - κόκκινο, Virtual ground - πορτοκαλί, Γείωση - Μαύρο

Τοποθετούμε το INA333, στο pin 7 τροφοδοτούμε το ενισχυτή και στο pin 4 τη γείωση. Στα pins 1 και 8 ελέγχουμε το κέρδος $G = 1 + \frac{100k}{R_1}$ με τη αντίστοιχη αντίσταση. Στη περίπτωση μας βάζουμε αντίσταση 10K για κέρδος 10. Στα pins 2, 3 είναι η είσοδος των ακροδεκτών και στο 6 η έξοδος.

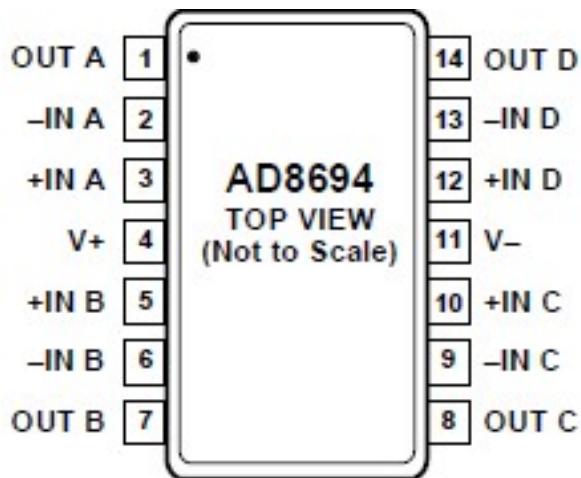


Σχήμα 5.2: INA333



Σχήμα 5.3: Σύνδεση INA333 (μπλέ)

Tα pins του AD8694 είναι ως εξής:



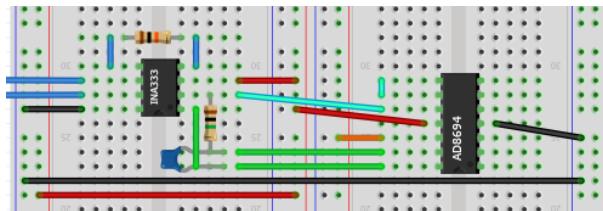
Σχήμα 5.4: AD8694

Το A χρησιμοποιείται για το κύκλωμα απομόνωσης, το B για το υψηπερατό φίλτρο και τα C, D στο χαμηλοπερατό φίλτρο.

Τώρα όταν δημιουργήσουμε το υψηπερατό φίλτρο ανατροφοδότησης στο ina και το κύκλωμα

απομόνωσης.

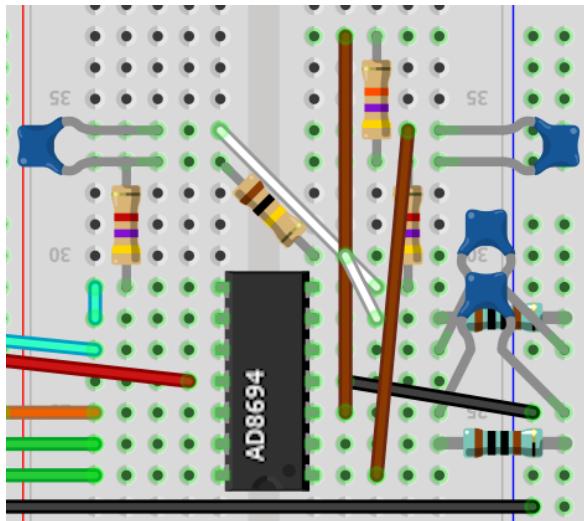
Αρχικά το pin 5 (IN+ B) πάει στο virtual ground. Τοποθετούμε ένα πυκνωτή στα pins 6(IN- B) και 7 (OUT B) του τελεστικού ενισχυτή και το πόδι του πυκνωτή που συνδέεται με το pin 7 με το pin 5 (REF) του INA. Τέλος τη έξοδο του INA, pin 6 με το πόδι του πυκνωτή που είναι ελεύθερο με αντίσταση 1.5M. Για το κύκλωμα απομόνωσης παίρνω τη έξοδο του INA pin 6, τη συνδέω με το IN+ A pin 3 και το pin 2 με 3. Το κύκλωμα πρέπει να μοιάζει κάπως έτσι:



Σχήμα 5.5: Σύνδεση με υψηπερατό φίλτρο (πράσινο) και κύκλωμα απομόνωσης (γαλάζιο)

Τελευταίο κομμάτι είναι το χαμηλοπερατό φίλτρο στα C και D. Πρώτα βάζουμε τις αντιστάσεις και πυκνωτές με το virtual ground. Αντίσταση 1K μεταξύ virtual ground, IN- και πυκνωτής virtual ground με IN+ και αντίσταση 10K ανάμεσα σε IN- και OUT.

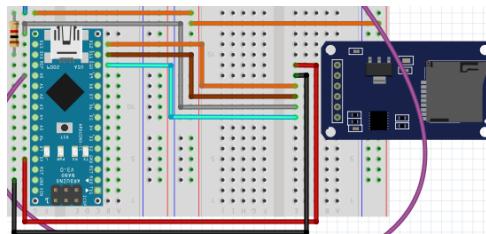
Αντίσταση 4.7K από OUT A,D και από αυτή 100K για IN+ D και 47K για IN+ C. Ανάμεσα σε αυτές ένας πυκνωτής στο OUT.



Σχήμα 5.6: Σύνδεση για χαμηλοπερατό φίλτρο

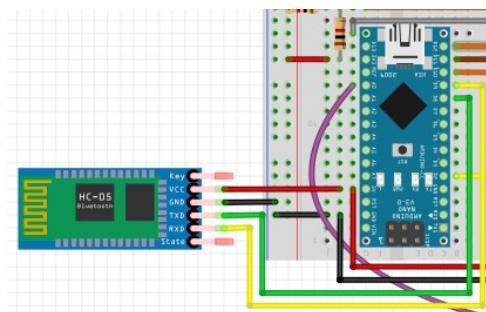
Το τελικό σήμα εξόδου είναι στο OUT C και το συνδέουμε με κάποιο analog pin του Arduino. Προφανώς αυτό το κύκλωμα πρέπει να γίνει άλλες 2 φορές για ολοκληρωμένο holter, με 3 εξόδους να πηγαίνουν στο Arduino

Αφού ολοκληρώσαμε το χύκλωμα προσθέτουμε το micro SD module. Όπως έχουμε πει στη ανάλυση υλικού η σύνδεση με το Arduino είναι η εικόνα 3.11. Αν το module έχει διαφορετικά ονόματα δείτε το πίνακα αντιστοίχησης 3.2.



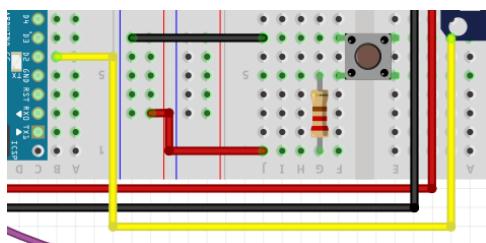
Σχήμα 5.7: Σύνδεση SD module με Arduino nano

Προσθέτουμε το bluetooth module. Τροφοδοσία στον ακροδέκτη 5V, γείωση στο GND. TXD, RXD σε D8,9 αντίστοιχα. Χρησιμοποιούμε αυτούς τους ακροδέκτες γιατί το Arduino χρησιμοποιεί αυτούς για φόρτωση λογισμικού και επικοινωνία.



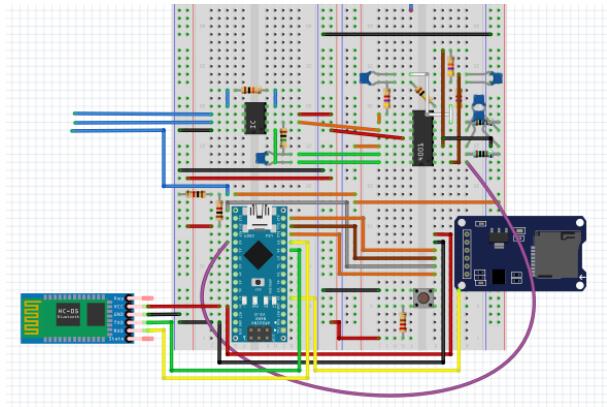
Σχήμα 5.8: Σύνδεση bluetooth module με Arduino nano

Τέλος προσθέτουμε ένα κουμπί για δημιουργία νέου αρχείου.

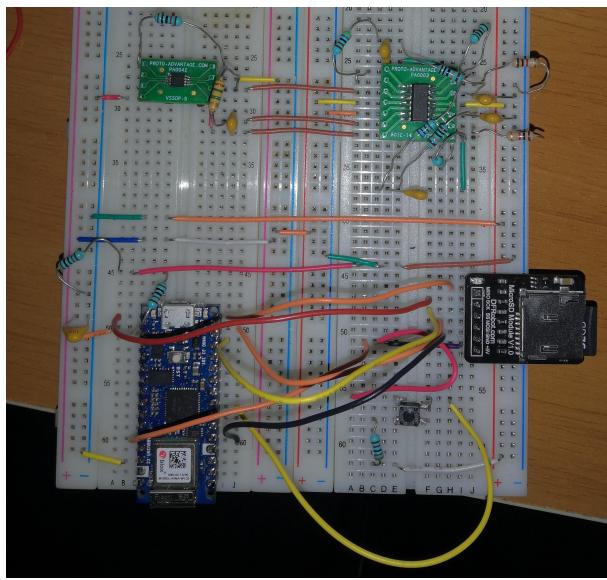


Σχήμα 5.9: Σύνδεση κουμπιού με Arduino nano

Το τελικό κύκλωμα μοιάζει κάπως έτσι:

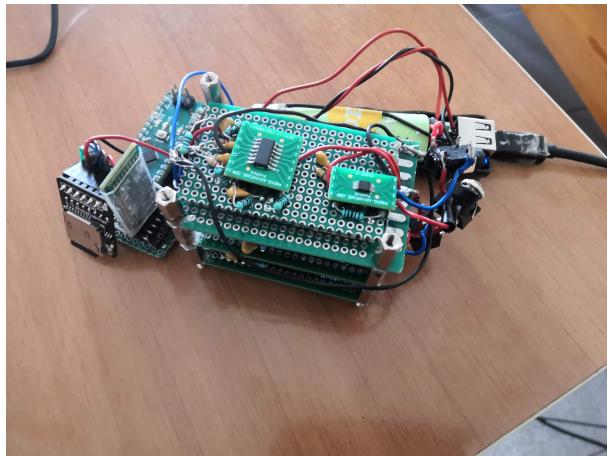


Σχήμα 5.10: Τελική συνδεσμολογία



Σχήμα 5.11: Τελική πειραματική συνδεσμολογία

Το όλο κύκλωμα γίνεται πιο μόνιμο κολλώντας το σε διάτρητη πλακέτα, ακολουθώντας τις ίδιες οδηγίες. Προφανώς το κύκλωμα ενίσχυσης πρέπει να γίνει 3 φορές, μια για κάθε κανάλι μέτρησης. Προσπαθούμε να κάνουμε το κύκλωμα όσο πιο μικρό γίνεται γιατί πρέπει να μπει σε κουτί. Επίσης το Arduino τοποθετείται και αυτό σε πλακέτα για να έχουμε πιο σταθερές συνδέσεις με το κύκλωμα ενισχυτή και τα περιφερικά του. Οι είσοδοι των ακροδεκτών γίνονται με εισόδους 3.5mm και τροφοδοτείται με powerbank του εμπορίου από το οποίο πήραμε έξοδο 5V, γείωση GND και τάση της μπαταρίας. Το τελικό κύκλωμα μοιάζει κάπως έτσι:



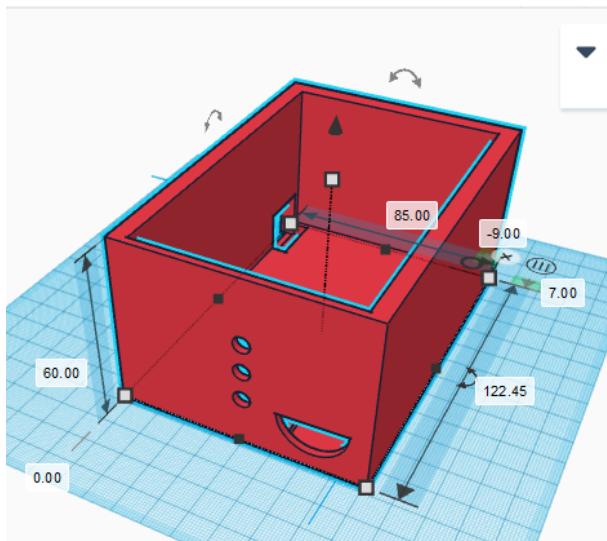
Σχήμα 5.12: Τελική συνδεσμολογία

5.3 Δημιουργία κουτιού

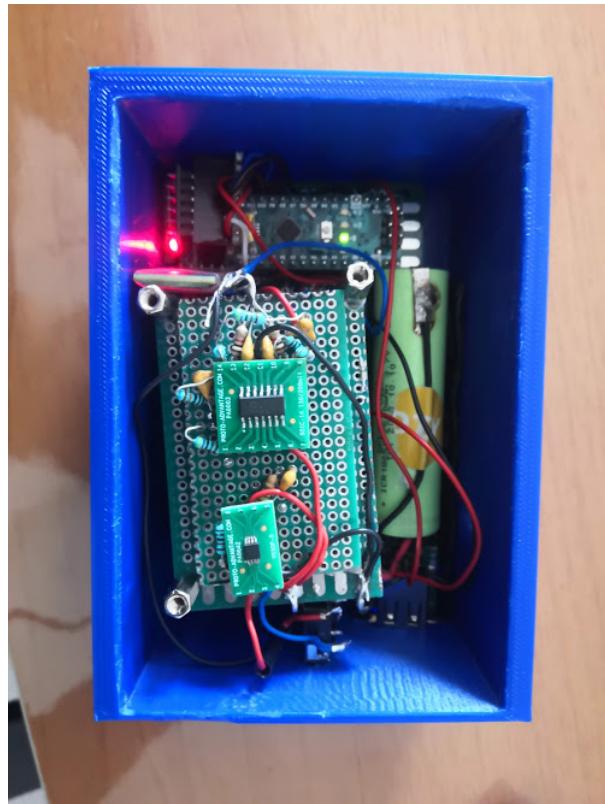
Αφού ολοκληρώσαμε τη τελική συνδεσμολογία, μπορούμε να πάρουμε μετρήσεις για τις διαστάσεις του κουτιού. Συγκεκριμένα έχουμε 12x8x6 εκατοστά. Για τη δημιουργία του κουτιού χρησιμοποιήσαμε 3D εκτυπωτή και δημιουργήσαμε το σχέδιο με τη ιστοσελίδα tinkerCAD.com.

Από τη τελική συνδεσμολογία παρατηρούμε πως χρειαζόμαστε 3 εισόδους για 3.5mm, μια είσοδο για φόρτιση μπαταρία και μια έξοδο για τη κάρτα sd. Τα τοιχώματα είναι 5mm εκτός από το τοίχωμα με τις εισόδους 3.5mm.

Μετά από τη εκτύπωση παρατηρήθηκε πως το τοίχωμα στα 5mm ήταν υπεραρκετό και θα μπορούσε να είναι το μισό, στα 2.5mm. Το τελικό προϊόν είναι:



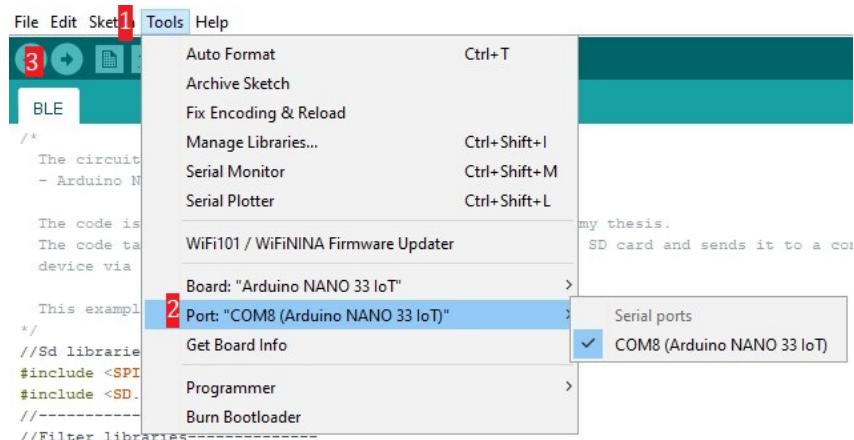
Σχήμα 5.13: Σχεδίαση κουτιού



Σχήμα 5.14: Τοποθέτηση κυκλώματος στο κουτί

5.4 Φόρτωση λογισμικού Arduino

Αφού συνδέσουμε το Arduino στο υπολογιστή και ανοίξουμε το IDE Arduino σιγουρεύομαστε ότι ο υπολογιστής ανίχνευσε τη συσκευή από τη επιλογή Tools Port. Αν η συσκευή δεν ανιχνεύτηκε την επιλέγουμε από το Tools Boards Arduino SAMD Boards και ξανασυνδέουμε. Πατάμε upload για να ανεβάσουμε τον κώδικα στο Arduino.

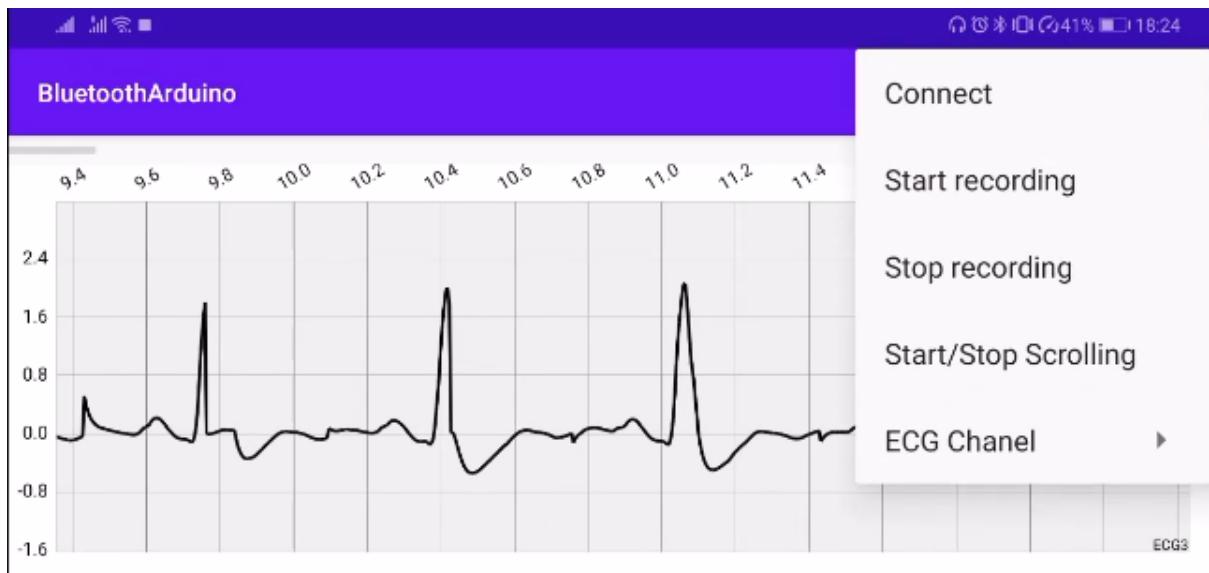


Σχήμα 5.15: Φόρτωση λογισμικού Arduino

5.5 Φόρτωση λογισμικού Android

Για να αναγνωρίσει το Android studio IDE τη συσκευή μας πρέπει να έχουμε ενεργοποιήσει τα developer options. Στις ρυθμίσεις πάμε στο about phone και πατάμε επτά φορές το build number και έτσι ενεργοποιούμε το developer options. Στα settings θα εμφανιστεί καινούργια επιλογή developer options και εκεί ενεργοποιούμε το USB debugging. Τη επόμενη φορά που θα ανοίξουμε το android studio θα αναγνωρίσει τη συσκευή και θα τρέξει το πρόγραμμα σε αυτή.

Αφού φορτώσουμε τη εφαρμογή το πρόγραμμα θα πρέπει να μοιάζει κάπως έτσι:



Σχήμα 5.16: Τελική εφαρμογή

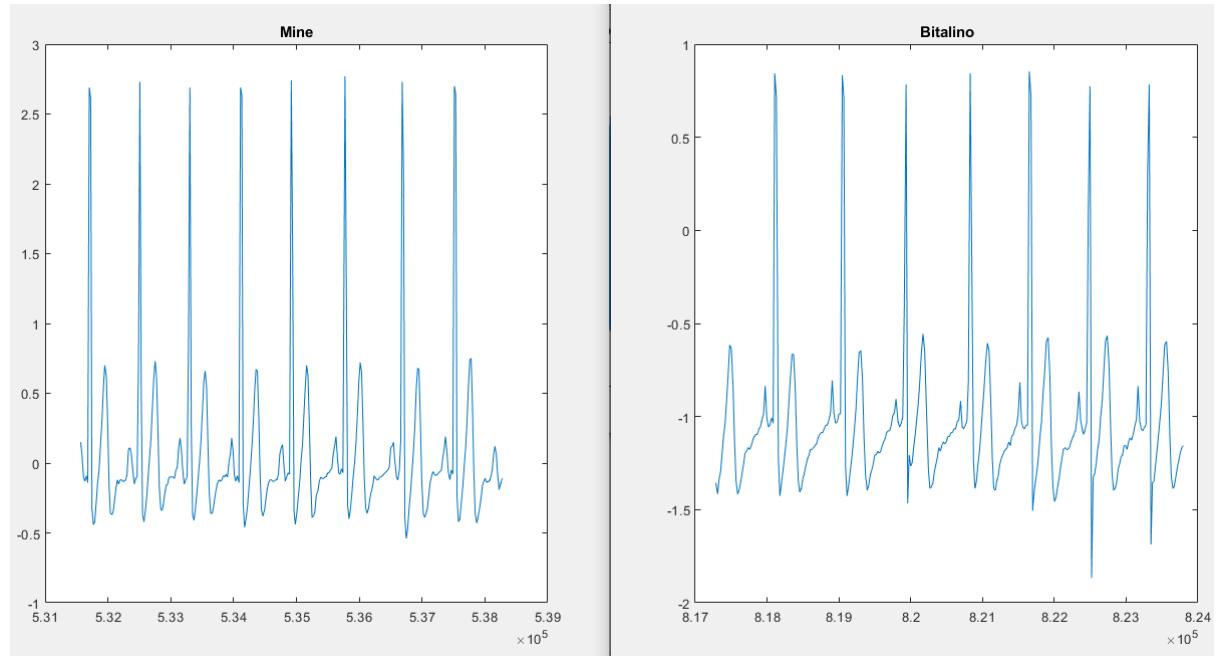
Κεφάλαιο 6

Αποτελέσματα και παρατηρήσεις

6.1 Τελικό αποτέλεσμα

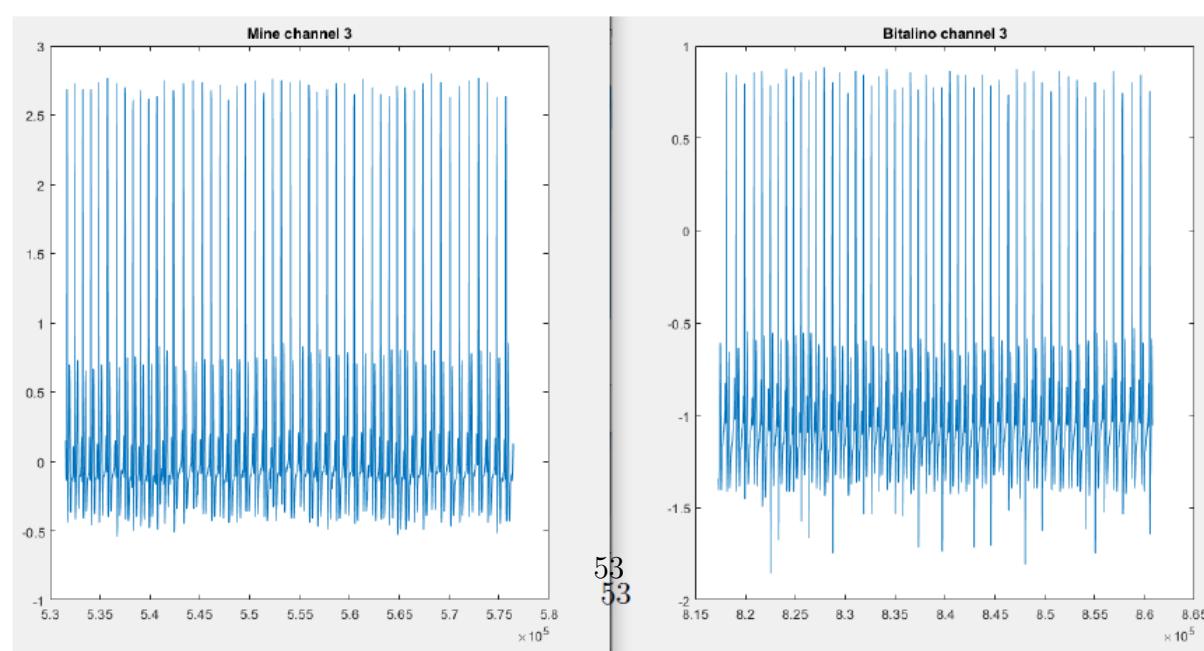
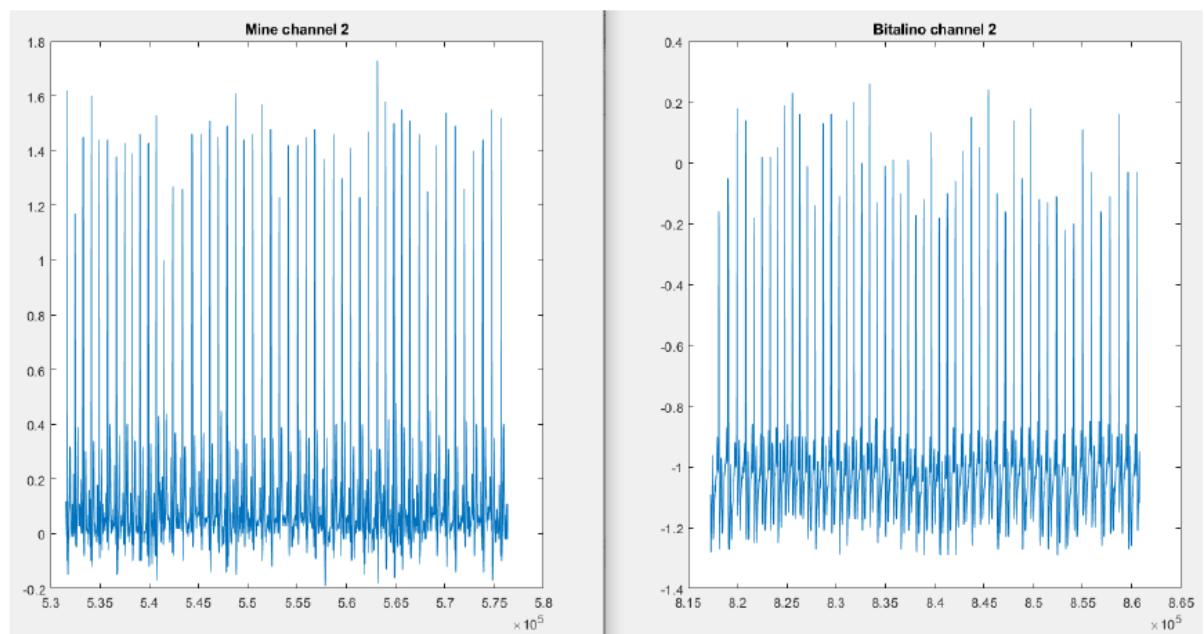
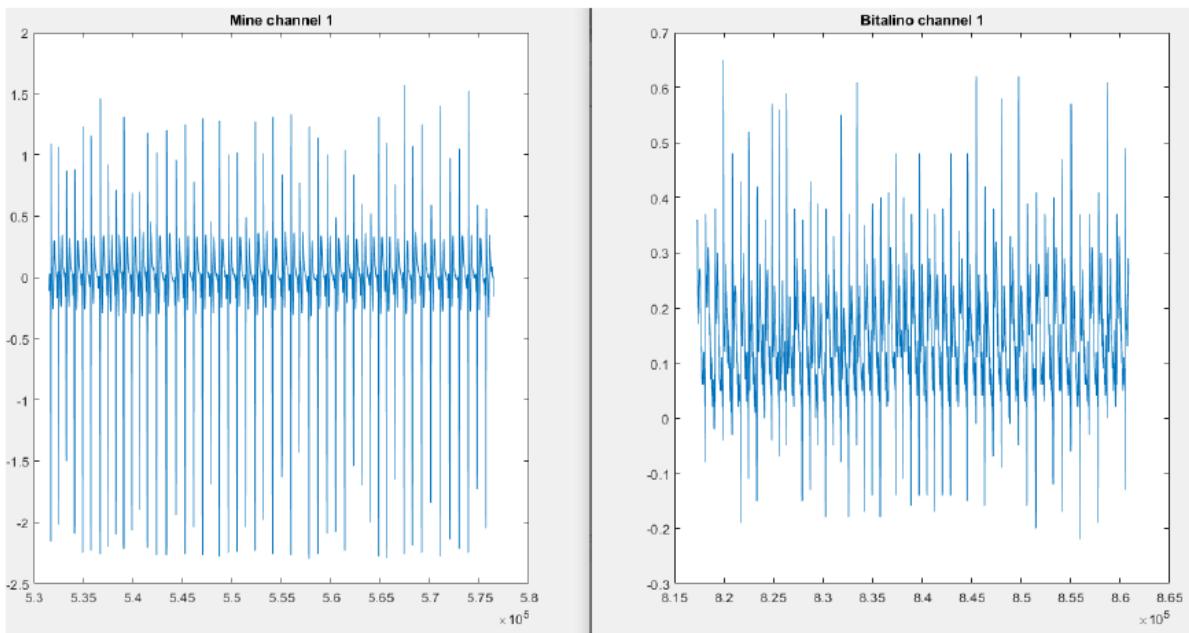
6.2 Μετρήσεις

Για μετρήσεις σύνδεσα το holter monitor όπως το σχήμα 8.4 και πήρα μετρήσεις για μισή ώρα. Τα αποτελέσματα σε ένα τυχαίο κομμάτι 300 σημείων:



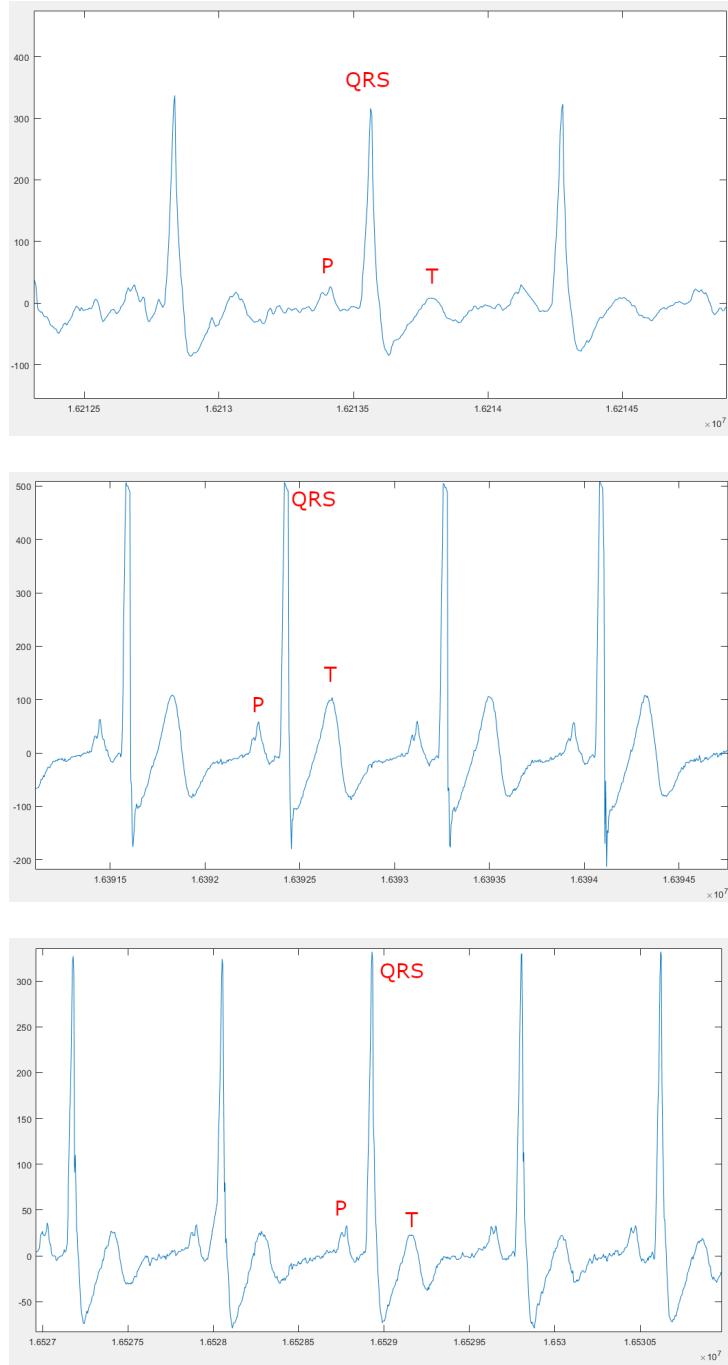
Σχήμα 6.1: Σύγχριση μετρήσεων

και σε άλλο δύο χιλιάδων σημείων:



Η κύρια παρατήρηση είναι πως στη δική μου υλοποίηση σε σύγκριση με το bitalino είναι πως το τελικό αποτέλεσμα είναι πιο ομαλό. Κατά τα άλλα οι δύο μετρήσεις είναι αρκετά όμοιες κάτι το οποίο είχαμε και στόχο.

Επίσης από χοντά μπορούμε να παρατηρήσουμε τις χαρακτηριστικές καμπύλες του καρδιογραφήματος



Σχήμα 6.3: Τελικό αποτέλεσμα

6.3 Απόδοση κώδικα

Μεγάλο μέρος της απόδοσης οφείλεται στο κώδικα του arduino, συγκεκριμένα στους χρόνους εγγραφής και αποστολής με bluetooth. Καταφέραμε να μειώσουμε το χρόνο αποστολής από σημείο σε σημείο περίπου στα 4ms (250Hz) αποθηκεύοντας τα δεδομένα κάθε 100 στοιχεία, ξεκινώντας και σταματώντας τις μετρήσεις με εντολή του χρήστη και αποστέλλοντας τα δεδομένα σε μορφή bytes. Αυτό έχει ως μειονέκτημα αν ο χρήστης βγάλει τη κάρτα από τη συσκευή, μερικά δεδομένα δεν θα εγγραφούν χωρίς εντολή από τη εφαρμογή αλλά καταφέραμε να μειώσουμε έτσι το χρόνο αποστολής κατά 40ms.

Επιπλέον η εντολή millis() που χρησιμοποιούμε για τη χρονική στιγμή των μετρήσεων εξαρτάται από τη συχνότητα του arduino nano η οποία δεν είναι πάντα σταθερή. Προτείνεται να προστεθεί ένα real time clock module το οποίο λειτουργεί με κρύσταλλο για πιο ακριβή χρονική στιγμή αλλά και για αναγνώριση ώρας και ημέρας, όχι μόνο χρόνο από τη αρχή της λειτουργίας του arduino

Τέλος, ιδανικά ο κώδικας πρέπει να γραφτεί σε καθαρή C για να μπορέσουμε να φτάσουμε κοντά στις θεωρητικές ταχύτητες bluetooth εγγραφής δεδομένων στη κάρτα SD.

Κεφάλαιο 7

Επίλογος

7.1 Συμπεράσματα

Η παρούσα διπλωματική εργασία αποτελεί ένα πλήρες εγχειρίδιο για την κατασκευή και τον προγραμματισμό ενός Holter monitor. Αρχικά αναλύσαμε τη ηλεκτρική λειτουργία της καρδιάς και τη συσκευή Bitalino. Με εξαρτήματα του εμπορίου καταφέραμε να δημιουργήσουμε ένα λειτουργικό καρδιογράφο με δικό μας κώδικα ο οποίος αποθηκεύει τις μετρήσεις σε κάρτα sd και συνδέεται μέσω bluetooth σε συσκευή android σε δική μας εφαρμογή. Το όλο κύκλωμα μπήκε σε ειδικά διαμορφωμένο χουτί για πιο όμορφη και πρακτική παρουσίαση.

Το κύκλωμα αυτό είναι καλό για πειραματικές μετρήσεις σε ακαδημαϊκό περιβάλλον αλλά πιστεύω χρειάζεται περισσότερη δουλειά και έλεγχοι για να είναι αποδεκτή σε ιατρικό περιβάλλον.

7.2 Περαιτέρω έρευνα

Η εργασία αυτή αποτελεί ένα καλό βήμα προς τη υλοποίηση μιας συσκευής έτοιμη για ιατρική χρήση. Κρίνεται όμως απαραίτητη η δημιουργία της συσκευής σε μικρότερη κλίμακα. Ιδανικά το όλο σχέδιο μπορεί να γίνει σε τυπωμένο κύκλωμα pcb με πολύ πιο μικρά εξαρτήματα. Επίσης αν γίνει σε pcb είναι ευκαιρία να αντικατασταθεί το arduino με σχεδίαση μικροεπεξεργαστή βασισμένου σε αυτό, μιας και τα σχέδια και κατάλογος υλικών είναι open source και εύκολα διαθέσιμα. Έτσι θα μπορεί να γίνει μια συσκευή πολύ μικρότερη από αυτή που έχουμε παρουσιάσει.

Επίσης σημαντική είναι και η καλή απόδοση του κώδικα. Καλό θα ήταν καλό να γραφτεί εξ ολοκλήρου σε C χωρίς τη βοήθεια βιβλιοθηκών του arduino. Ιδανικά να φτάσουμε ταχύτητες αποστολής κάθε 1ms. Τέλος μια μεγαλύτερη μπαταρία για πολλαπλές μέρες λειτουργίας θα ήταν καλό.

Βιβλιογραφία

- [1] π. Γεώργιος Αναγνωστόπουλος. Εμβιοηλεκτρομαγνητισμός, Κεφάλαιο 4. [course] Ανακτήθηκε την Κυριακή, 03 Ιανουαρίου 2021 από <https://eclass.duth.gr/courses/TMA267/>
- [2] Γεώργιος Χατζηαθανασίου, Σύγχρονο διαδικτυακό βιβλίο καρδιολογίας, Το ηλεκτροκαρδιογράφημα (ΗΚΓ)- ενηλίκου και παιδικό. [blog]
- [3] Sendra/Smith Μικροηλεκτρονικά Κυκλώματα - 7η έκδοση, Αθήνα, Εκδώσεις Παπασωτηρίου, 2015, pp. 82-85
- [4] Sendra/Smith Μικροηλεκτρονικά Κυκλώματα - 7η έκδοση, Αθήνα, Εκδώσεις Παπασωτηρίου, 2015, pp. 1342-1344
- [5] Επίσημη ιστοσελίδα arduino
- [6] "A resolution to redefine SPI signal names", Open Source Hardware Association
- [7] "Serial Peripheral Interface (SPI) ", Επίσημη ιστοσελίδα sparkfun
- [8] Eric Peña and Mary Grace Legaspi, "UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter", AnalogDialogue
- [9] Επίσημο git repository, Arduino
- [10] Επίσημο git repository, The wiring project
- [11] Επίσημο reference manual βιβλιοθηκών, Arduino
Όλα τα links είναι προσβάσιμα στις 5 Απριλίου 2021

Κεφάλαιο 8

Παράρτημα

8.1 Εγχειρίδιο χρήσης

8.1.1 Επισκόπηση

Το holter monitor μας είναι φορητός καρδιογράφος καταγραφής καρδιακής λειτουργίας σε κάρτα sd και παρουσίαση της μέσω bluetooth σε εφαρμογή Android.



Σχήμα 8.1: Κουτί holter monitor

Όπως φαίνεται στο πιο πάνω σχήμα το κουτί αποτελείται από 5 κύρια μέρη:

1. Ακροδέκτης 1
2. Ακροδέκτης 2

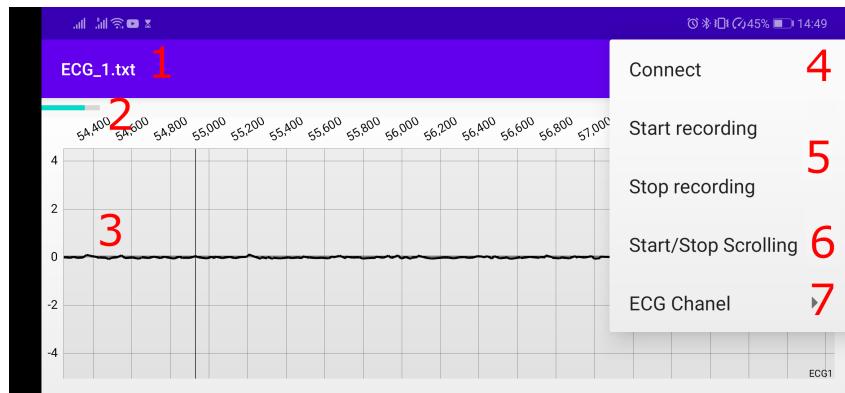
3. Ακροδέκτης 3
4. Είσοδος φόρτησης
5. Είσοδος κάρτας sd

Μαζί έρχονται και οι αντίστοιχοι ακροδέκτες, επισημασμένοι για τοποθέτηση τους στις ανάλογες εισόδους.



Σχήμα 8.2: Ακροδέκτες holter monitor

Ανοίγοντας τη εφαρμογή έχουμε τα εξής:



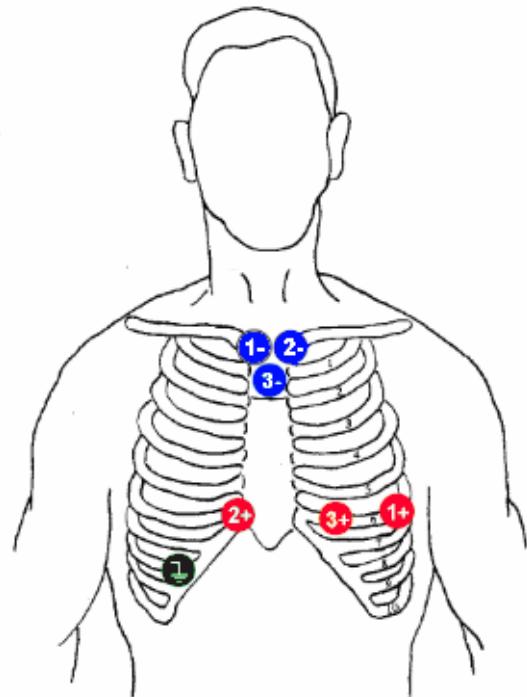
Σχήμα 8.3: Εφαρμογή Android

1. Όνομα αρχείου στο οποίο καταγράφει η συσκευή στη κάρτα sd.
2. Ποσοστό μπαταρίας συσκευής.
3. Παρουσίαση καρδιακού παλμού.
4. Επανασύνδεση στην εφαρμογή.

5. Εκκίνηση καταγραφής.
6. Παύση καταγραφής.
7. Επιλογή καναλιού.

8.1.2 Οδηγίες χρήσης

1. Τοποθετούμε τους ακροδέκτες στον ασθενή όπως φαίνεται στο παρακάτω σχήμα, και τους συνδέουμε στην αντίστοιχη είσοδο:



Σχήμα 8.4: Τοποθέτηση ακροδεκτών στο ασθενή

Οποιαδήποτε τυπική τοποθέτηση διπολικού καρδιογράφου είναι αποδεκτή.

2. Τοποθετούμε κάρτα sd. Η συσκευή αρχικοποιεί τη κάρτα και είναι έτοιμη για καταγραφή
3. Στην android συσκευή ανοίγουμε την εφαρμογή. Συνδεόμαστε αυτόματα με το μήνυμα α Connected to: holter monitor. Αν όχι, στο μενού επιλογών (τρεις τελείες πάνω δεξιά), επιλέγουμε connect και μετά start recording.

Ο παλμός εμφανίζεται σε μορφή γραφικής, με τον άξονα Y να είναι η τάση σε V και ο άξονας X ο χρόνος σε ms.

ΠΡΟΣΟΧΗ: Για να αρχίσει η καταγραφή πρέπει να σταλθεί εντολή start recording.

4. Όταν τελειώσουμε τις μετρήσεις και θέλουμε να βγάλουμε τη κάρτα sd, επιλέγουμε Stop recording. Όταν είναι έτοιμη, εμφανίζεται το μήνυμα Saved, you can eject the SD card.

ΠΡΟΣΟΧΗ: Αν η κάρτα βγεί χωρίς να θέσουμε Stop recording δεν θα αποθηκευτούν τα δεδομένα στη κάρτα.

8.2 Κώδικας

Github repository με κώδικα android, arduino και αρχεία για 3D εκτύπωση

Arduino

Listing 8.1: Κώδικας Arduino

```
1
2
3  /*
4  Konstantinos Knais - 8967
5  This code is part of my univercity thesis.
6  To be used along with an ecg amplifier.
7  Reads inputs at pins, passes them through low pass filter, writes them at
8  sd card and sends them with bluetooth.
9  This code is in the public domain.
10 */
11 //Sd libraries-----
12 #include <SPI.h>
13 #include <SD.h>
14 //-----
15 //Filter libraries-----
16 #include <Filters.h>
17 //-----
18 #include <SoftwareSerial.h>
19 //-----
20
21 SoftwareSerial mySerial(8, 9); // RX, TX
22
23 int flag=0;
24 int startFlag=0;
25
26 int input1=0;
27 int input2=0;
28 int input3=0;
29
30 long sendTime=0;
31 int batteryLvl=0;
32 const int analogPin1 = A1;
33 const int analogPin2 = A2;
34 const int analogPin3 = A3;
35 const int batteryPin = A5;
36
37 byte measurementsArray[9];
38
39 float LowFrequency = 0.5;
40 FilterOnePole highPassFilter(HIGHPASS,LowFrequency);
41
42 File myFile;//SD card file
43 String fileName;
44 int num=0;
45 int i=0;
46 int j=0;
47 unsigned long currentMillis=0;
48 String values;
```

```

49
50 long start;
51 long res;
52 long dif=0;
53 long tic=0;
54
55 void setup() {
56     // Open serial communications and wait for port to open:
57     Serial.begin(38400);
58
59     // set the data rate for the SoftwareSerial port
60     mySerial.begin(38400);
61
62     //Make sure sd card is inserted
63     while(!setupSD()){
64         sendBlue(4,4,4,4); //mySerial.println("Insert SD Card");
65         delay(2000);
66     }
67     analogReference(DEFAULT);
68
69 }
70
71 void loop() { // run over and over
72     //Take measurements
73     //start = micros();
74     startTime=millis();
75     input1 = highPassFilter.input(analogRead(analogPin3)); // read the input
76     pin1
77     //I wired the input in reverse.
78     input2 = highPassFilter.input(analogRead(analogPin2)); // read the input
79     pin2
80     input3 = highPassFilter.input(analogRead(analogPin1)); // read the input
81     pin3
82     //1120us -> 1.12ms
83
84     if(mySerial){//Receive commands from phone
85         flag=mySerial.read();
86         if(flag==49){//Ascii code 49 = number 1
87             //Stop recording
88             startFlag=0;
89             myFile.flush(); //5ms
90             myFile.close();
91             sendBlue(3,3,3,3); //mySerial.println("Saved, you can eject the SD
92             card");
93         }
94         if(flag == 50 && startFlag==0){
95             //Start recording
96             startFlag=1;
97             //Make sure sd card is inserted
98             while(!setupSD()){
99                 sendBlue(4,4,4,4); //mySerial.println("Insert SD Card");
100                delay(2000);
101            }
102        }

```

```

103  if(startFlag){// If start flag = 1 sd card is inserted and its safe to
     start writing
104
105  dif = micros()-tic;
106  tic = micros();
107  sendBlue(2,input1,input2,input3,dif);//7450us ->7.45ms
108
109  values = input1*5/1023.0;
110  values+= " , ";
111  values+= input2*5/1023.0;
112  values+= " , ";
113  values+= input3*5/1023.0;
114  values+=" , ";
115  values+=sendTime;
116  myFile.println(values); //450us normal -> 3900us if flush
117
118  i++;
119  if(i==1000){
120      i=0;
121      myFile.flush(); //5ms
122  }
123  if((sendTime-currentMillis)>60000){//1 minuite
124      currentMillis=millis();
125      if((analogRead(batteryPin)*5/1023)<3){
126          sendBlue(5,5,5,5,5);
127      }
128  }
129 }
130 //Tottal time: 3500us->3.5ms -> 285.71Hz
131 }
132
133 void sendBlue(int what, int x, int y, int z, long t){
134     measurementsArray[0] = what; //2- Measurements, 3 - Eject sd card, 4 -
        Insert SD, 5 - Low battery
135     measurementsArray[1] = x >> 8;//MSB x
136     measurementsArray[2] = x; //LSB x
137     measurementsArray[3] = y >> 8;//MSB y
138     measurementsArray[4] = y; //LSB y
139     measurementsArray[5] = z >> 8;//MSB z
140     measurementsArray[6] = z; //LSB z
141     measurementsArray[7] = t >> 8;//MSB z
142     measurementsArray[8] = t; //LSB z
143     mySerial.write(measurementsArray, sizeof(measurementsArray));
144 }
145
146
147 bool setupSD(){
148     //Setup SD start
149     if(SD.begin()) {
150         fileName = "ECG_";
151         fileName += num;
152         fileName += ".txt";
153         num++;
154         myFile = SD.open(fileName, FILE_WRITE);
155         // if the file opened okay, write to it:
156         if (myFile) {
157             myFile.println("ECG1, ECG2, ECG3, Time");
158         }

```

```

159      return true;
160  }else{
161      return false;
162  }
163 }
```

Android

Listing 8.2: Κλάση MainActivity.java

```

1 package com.example.bluetootharduino;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.os.Bundle;
6 import android.view.Menu;
7 import android.view.MenuInflater;
8 /*
9 * Konstantinos Knais 8967
10 * Android application used in my thesis - 7 lead holter monitor
11 * To be used along with the designed holter monitor
12 * Main Activity class
13 */
14 public class MainActivity extends AppCompatActivity {
15
16     @Override
17     protected void onCreate(Bundle savedInstanceState) {
18         super.onCreate(savedInstanceState);
19         setContentView(R.layout.activity_main);
20     }
21 }
```

Listing 8.3: Κλάση ConnectedThread.java

```

1 package com.example.bluetootharduino;
2
3 import android.bluetooth.BluetoothSocket;
4 import android.content.Intent;
5 import android.os.Build;
6 import android.os.Message;
7 import android.util.Log;
8
9 import java.io.BufferedReader;
10 import java.io.BufferedInputStream;
11 import java.io.ByteArrayInputStream;
12 import java.io.IOException;
13 import java.io.InputStream;
14 import java.io.InputStreamReader;
15 import java.io.OutputStream;
16 import java.nio.ByteBuffer;
17 import java.nio.ByteOrder;
18 import java.time.Instant;
19 import java.util.Arrays;
20
21 import android.os.Handler;
22
23 import androidx.annotation.RequiresApi;
```

```

24
25 import static java.lang.Math.abs;
26
27 /**
28 * Konstantinos Knais 8967
29 * Android application used in my thesis - 7 lead holter monitor
30 * To be used along with the designed holter monitor
31 * Manages incoming and outcomming data to arduino
32 */
33 public class ConnectedThread extends Thread {
34     private final BluetoothSocket mmSocket;
35     private final InputStream mmInStream;
36     private final OutputStream mmOutStream;
37     public static final int RESPONSE_MESSAGE = 10;
38     public static final int MESSAGE_MESSAGE = 12;
39     public static final int max_measurments = 80;
40
41     Handler uih;
42
43     public Handler mHandler;
44
45     public ConnectedThread(BluetoothSocket socket,Handler uih){
46         mmSocket=socket;
47         InputStream tmpIn = null;
48         OutputStream tmpOut = null;
49         this.uih=uih;
50         Log.i("[THREAD-CT]","Creating thread");
51         try {
52             tmpIn=socket.getInputStream();
53             tmpOut=socket.getOutputStream();
54         } catch (IOException e) {
55             Log.e("[THREAD-CT]","Error:"+ e.getMessage());
56         }
57         mmInStream=tmpIn;
58         mmOutStream=tmpOut;
59         try{
60             mmOutStream.flush();
61         } catch (IOException e) {
62             return;
63         }
64         Log.i("[THREAD-CT]","IO's obtained");
65     }
66     @RequiresApi(api = Build.VERSION_CODES.O)
67     public void run(){
68         int num_of_packets = 9;
69         byte[] bytes = new byte[num_of_packets];
70         int iterator = 0;
71         boolean next2Flag=false;
72         short[] measurements = new short[max_measurments*4];
73         int measurements_num=0;
74         BufferedReader br = new BufferedReader(new
75             InputStreamReader(mmInStream));
76         BufferedInputStream bis = new BufferedInputStream(mmInStream);
77
78         Log.i("[THREAD-CT]","Starting thread");
79         while (true){
80             try{
81                 int bytesAvailable = mmInStream.available();

```

```

81     if (bytesAvailable > 0) {
82         byte[] curBuf = new byte[bytesAvailable];
83         mmInStream.read(curBuf);
84         for (byte b : curBuf) {
85             if (b == 2 && iterator == num_of_packets) { //start of message,
86                 translate byte array collected and send
87                 short[] shorts = new short[bytes.length/2];
88                 // to turn bytes to shorts as either big endian or little
89                 // endian.
90                 byte[] modifiedBytes = Arrays.copyOfRange(bytes, 1,
91                     bytes.length); //Discard first byte
92                 ByteBuffer.wrap(modifiedBytes).order(ByteOrder.BIG_ENDIAN).asShortBuffer().get(s
93                     numbers from the other bytes
94                 if(shorts[0]>900){ //
95                     shorts[0]=0;
96                     shorts[1]=0;
97                     shorts[2]=0;
98                 }
99                 //After 100 packets send to ardconn to view
100                if(measurements_num==max_measurments*4){
101                    measurements_num=0;
102                    //Send measurements to Ardconn
103                    Message msg = new Message();
104                    msg.what = RESPONSE_MESSAGE;
105                    msg.obj =measurements;
106                    uih.sendMessage(msg);
107                }
108                //Add shorts to a bigger array
109                System.arraycopy(shorts, 0, measurements, measurements_num,
110                    shorts.length);
111                //Added measurements, increase point on array
112                measurements_num=measurements_num+4;
113                /*
114                for(short s:shorts){
115                    Log.d("[THREAD-CT]","shorts: "+s);
116                }
117                Log.d("[THREAD-CT]","");
118                */
119                iterator = 0;
120                next2Flag=false;
121                bytes[iterator] = b;
122                //Log.d("[THREAD-CT]","bytes["+iterator+"]: "+bytes[iterator]);
123                } else if(iterator<num_of_packets) { //Save next byte
124                    bytes[iterator] = b;
125                    //Log.d("[THREAD-CT]","bytes["+iterator+"]: "+bytes[iterator]);
126                }else{ //if im here I lost the 2 for the start and i have to wait
127                    for the next
128                    Log.d("[THREAD-CT]","next2Flag enabled");
129                    //Should have a flag to wait for the next 2
130                    next2Flag=true;
131                }
132                if(b==3 && iterator==num_of_packets){
133                    //Code to send "Saved, you can eject the SD card"
134                    Message msg = new Message();
135                    msg.what = MESSAGE_MESSAGE;
136                    msg.obj ="Saved, you can eject the SD card";
137                    uih.sendMessage(msg);
138                }else if(b==4 && iterator==num_of_packets){

```

```

133         //Code to send "Insert SD Card"
134         Message msg = new Message();
135         msg.what = MESSAGE_MESSAGE;
136         msg.obj ="Insert SD card";
137         uih.sendMessage(msg);
138     }else if (b==5 && iterator==num_of_packets){
139         //Code to send Battery low"
140         Message msg = new Message();
141         msg.what = MESSAGE_MESSAGE;
142         msg.obj ="Battery low";
143         uih.sendMessage(msg);
144     }
145     if (next2Flag){
146         //wait untill next 2
147         Log.d("[THREAD-CT]","next2Flag enabled");
148         iterator=num_of_packets;
149     }else {
150         iterator++;
151     }
152 }
153 }
154 } catch (IOException e) {
155     break;
156 }
157 }
158 }
159 Log.i("[THREAD-CT]","While loop ended");
160 }
161
162 public void write(byte[] bytes){
163     try{
164         Log.i("[THREAD-CT]","Writting bytes");
165         mmOutStream.write(bytes);
166     }catch(IOException e){}
167 }
168
169 public void cancel(){
170     try{
171         mmSocket.close();
172     }catch(IOException e){}
173 }
174
175
176 }

```

Listing 8.4: Κλάση Ardconn.java

```

1 package com.example.bluetootharduino;
2
3 import android.bluetooth.BluetoothAdapter;
4 import android.bluetooth.BluetoothDevice;
5 import android.bluetooth.BluetoothSocket;
6 import android.content.Intent;
7 import android.graphics.Color;
8 import android.os.Build;
9 import android.os.Bundle;
10 import android.os.Handler;
11 import android.os.Looper;
12 import android.os.Message;

```

```

13 import android.os.SystemClock;
14 import android.text.method.ScrollingMovementMethod;
15 import android.util.Log;
16 import android.view.GestureDetector;
17 import android.view.Menu;
18 import android.view.MenuInflater;
19 import android.view.MenuItem;
20 import android.view MotionEvent;
21 import android.view.View;
22 import android.widget.Button;
23 import android.widget.ProgressBar;
24 import android.widget.TextView;
25 import android.widget.Toast;
26
27 import androidx.annotation.NonNull;
28 import androidx.annotation.RequiresApi;
29 import androidx.appcompat.app.AppCompatActivity;
30 import androidx.core.view.GestureDetectorCompat;
31
32 import com.github.mikephil.charting.charts.LineChart;
33 import com.github.mikephil.charting.components.Legend;
34 import com.github.mikephil.charting.data.Entry;
35 import com.github.mikephil.charting.data.LineData;
36 import com.github.mikephil.charting.data.LineDataSet;
37
38 import java.io.IOException;
39 import java.time.LocalDateTime;
40 import java.time.temporal.ChronoField;
41 import java.util.ArrayList;
42 import java.util.UUID;
43
44 import static android.os.SystemClock.elapsedRealtime;
45 /**
46 * Konstantinos Knais 8967
47 * Android application used in my thesis - 7 lead holter monitor
48 * To be used along with the designed holter monitor
49 * Creates connection to arduino and shows incoming data with
50 * mpandroidchart
51 */
52 //Initialize Bluetooth and make a connection
53 public class Ardconn extends AppCompatActivity {
54     public final static String MODULE_MAC = "98:D3:31:FD:78:D9";//Change
55     mac address if different arduino
56     public final static int REQUEST_ENABLE_BT=1;
57     private static final UUID MY_UUID =
58         UUID.fromString("00001101-0000-1000-8000-00805f9b34fb");
59
60     private GestureDetectorCompat mDetector;
61
62     BluetoothAdapter bta;
63     BluetoothDevice mmDevice;
64     ConnectedThread btt=null;
65
66     TextView response;
67     ProgressBar batteryLvl;
68     public Handler mHandler;
69
70     public LineChart chart;

```

```

68     ArrayList<Entry> valuesECG1 = new ArrayList<>();
69     long time_ecg;
70     long start;
71     float[] lastVal = new float[3];
72     ArrayList<Entry> valuesECG2 = new ArrayList<>();
73     ArrayList<Entry> valuesECG3 = new ArrayList<>();
74     //int max_measurments = 80;//
75     float[][] values = new float[4][ConnectedThread.max_measurments];
76     int showChart=1;
77     boolean scrolling = true;
78     long sum;
79     int num;
80     final int ARRAY_LIMIT = 100000; //Change if i want more
81
82     BluetoothSocket mmSocket = null;
83     @Override
84     protected void onCreate(Bundle savedInstanceState) {
85         super.onCreate(savedInstanceState);
86         setContentView(R.layout.activity_main);
87
88         Log.i("[BLUETOOTH]", "Creating listeners");
89         //    response = (TextView) findViewById(R.id.textView);
90         //    response.setMovementMethod(new ScrollingMovementMethod());
91
92         bta = BluetoothAdapter.getDefaultAdapter();
93         //If bluetooth is not enabled create intent for user to turn it on
94         if(!bta.isEnabled()){
95             setRequestEnableBt();
96         }else{
97             initiateBluetoothProcess();
98         }
99
100        chart=findViewById(R.id.chart);
101        setupChart(chart, Color.rgb(255,255,255));
102        //start = System.currentTimeMillis();
103
104    }
105    @Override
106    public boolean onCreateOptionsMenu(Menu menu) {
107        MenuInflater inflater = getMenuInflater();
108        inflater.inflate(R.menu.my_menu, menu);
109        return true;
110    }
111
112    @Override
113    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
114        switch (item.getItemId()){
115            case R.id.connectOption:
116                if(mmSocket!=null){
117                    try {
118                        mmSocket.close();
119                    } catch (IOException e) {
120                        e.printStackTrace();
121                    }
122                }
123                initiateBluetoothProcess();
124                return true;
125            case R.id.startOption:

```

```

126         //Send the number 2 to arduino
127         Log.d("[Ardconn]","Sending 2");
128         btt.write("2".getBytes());
129         return true;
130     case R.id.stopOption:
131         //Send the number 1 to arduino
132         Log.d("[Ardconn]","Sending 1");
133         btt.write("1".getBytes());
134         return true;
135     case R.id.scrollOption:
136         scrolling = !scrolling;
137         return true;
138     case R.id.chanel1:
139         //Show ecg1
140         showChart=1;
141         return true;
142     case R.id.chanel2:
143         //Show ecg2
144         showChart=2;
145         return true;
146     case R.id.chanel3:
147         //Show ecg3
148         showChart=3;
149         return true;
150     default:
151         return super.onOptionsItemSelected(item);
152     }
153 }
154
155 private void setRequestEnableBt() {
156     Intent enableBTIntent= new
157             Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
158     startActivityForResult(enableBTIntent,REQUEST_ENABLE_BT);
159 }
160 //Catch result of on off dialog
161 @Override
162 protected void onActivityResult(int requestCode, int resultCode,
163         Intent data) {
164     super.onActivityResult(requestCode, resultCode, data);
165
166     if(resultCode==RESULT_OK && requestCode==REQUEST_ENABLE_BT) {
167         initiateBluetoothProcess();
168     } else{
169         toastMessage("Please Enable Bluetooth");
170     }
171
172     private void initiateBluetoothProcess() {
173         Log.d("[Ardconn]","initiateBluetoothProcess");
174         if(bta.isEnabled()){
175             //Attempt to connect to bluetooth module
176             final BluetoothSocket[] tmp = new BluetoothSocket[1];
177             mmDevice = bta.getRemoteDevice(MODULE_MAC);
178             toastMessage("Searching...");
179             //Create socket
180             try {
181                 tmp[0]
182                     =mmDevice.createInsecureRfcommSocketToServiceRecord(MY_UUID);
183             mmSocket= tmp[0];

```

```

181         mmSocket.connect();
182         Log.i("[BLUETOOTH]", "Connected to: "+mmDevice.getName());
183         toastMessage("Connected to: "+mmDevice.getName());
184     } catch (IOException e) {
185         try {
186             mmSocket.close();
187         }catch (IOException i){
188             i.printStackTrace();
189         }
190     }
191     Log.i("[BLUETOOTH]", "Creating handler");
192     mHandler= new Handler(Looper.getMainLooper()) {
193         @RequiresApi(api = Build.VERSION_CODES.O)
194         @Override
195         public void handleMessage(Message msg){
196             //super.handleMessage(msg);
197             if(msg.what == ConnectedThread.RESPONSE_MESSAGE) {
198                 //Get message from connected thread
199                 short[] txt = (short[]) msg.obj;
200                 //Log.d("[Ardconn]", "txt.length: "+txt.length);
201                 //Seperate to each 1,2,3 ecg
202                 //Log.d("[Ardconn]", "txt.length: "+txt.length);
203                 for(int i=0;i<txt.length;i=i+4){
204                     int j = i/4;
205                     //Log.d("[THREAD-CT]", "i: "+i+" j: "+j);
206                     values[0][j] = (float)txt[i]*5/1023; //ECG1 I wired them
207                     weong
208                     values[1][j] = (float)txt[i+1]*5/1023; //ECG2
209                     values[2][j] = (float)-txt[i+2]*5/1023; //ECG3
210                     if(j==0){
211                         values[3][j]= (float)
212                             (values[3][values[3].length-1]+txt[i+3]/1000000.0);
213                             //Convert to s
214                     }else{
215                         values[3][j]= (float)
216                             (values[3][j-1]+txt[i+3]/1000000.0); //Convert
217                             to s
218                     }
219                 }
220                 for (int i=0;i<3;i++){
221                     values[i][0] = lastVal[i];
222                     values[i] = lowPassFilter(values[i]);
223                     lastVal[i] = values[i][values.length - 1];
224                 }
225                 previewData(values);
226             }
227             if(msg.what == ConnectedThread.MESSAGE_MESSAGE) {
228                 //Get battery lvl
229                 String txt = (String) msg.obj;
230                 toastMessage(txt);
231             }
232         };
233     }
234     Log.i("[BLUETOOTH]", "Creating and running Thread");
235     btt = new ConnectedThread(mmSocket,mHandler);
236     btt.start();

```

```

234         }else{
235             setRequestEnableBt();
236         }
237     }
238
239     private float[] lowPassFilter(float[] input) {
240         float[] output = new float[input.length];
241         float cutoffPoint = (float) 0.87;
242         output[0] = cutoffPoint * input[0];
243         for (int i = 1; i < input.length; i++) {
244             output[i] = output[i - 1] + cutoffPoint * (input[i] - output[i - 1]);
245         }
246         return output;
247     }
248     private void setupChart(LineChart chart,int color){
249         Log.d("[LedControl]","setupChart called");
250         // no description text
251         chart.getDescription().setEnabled(true);
252         chart.getDescription().setTextColor(Color.BLACK);
253
254         // chart.setDrawHorizontalGrid(false);
255         //
256         // enable / disable grid background
257         chart.setDrawGridBackground(true);
258
259         // enable touch gestures
260         chart.setTouchEnabled(true);
261
262         // enable scaling and dragging
263         chart.setDragEnabled(true);
264         chart.setScaleEnabled(false);
265
266         // if disabled, scaling can be done on x- and y-axis separately
267         chart.setPinchZoom(false);
268         chart.setScaleYEnabled(false);
269         chart.setBackgroundColor(color);
270
271         // set custom chart offsets (automatic offset calculation is
272         // hereby disabled)
273         //chart.setViewPortOffsets(10, 0, 10, 0);
274         chart.getAxisX().setDrawLimitLinesBehindData(true);
275         chart.getAxisLeft().setEnabled(true);
276         chart.getAxisLeft().setSpaceTop(40);
277         //chart.getAxisLeft().setLabelCount(50,true);
278         chart.getAxisLeft().setSpaceBottom(40);
279         chart.getAxisRight().setEnabled(false);
280         //chart.getAxisX().setEnabled(true);get
281         chart.getAxisX().setLabelCount(20,false);
282
283         chart.getAxisLeft().setDrawGridLines(true);
284         chart.getAxisX().setDrawGridLines(true);
285         chart.getAxisX().setLabelRotationAngle(-30);
286
287         // animate calls invalidate()...
288         //chart.animateX(2500);
289         chart.invalidate();
}

```

```

290     private void toastMessage(String message) {
291         Toast.makeText(this, message, Toast.LENGTH_LONG).show();
292     }
293
294     @RequiresApi(api = Build.VERSION_CODES.O)
295     private void previewData(float[][][] data) {
296         //Log.d("[LedControl]", "previewCharts called");
297         //Log.d("[previewData]", "Previewing data[0]: "+data[0]);
298         if(valuesECG1.size()==ARRAY_LIMIT){
299             Log.d("[Ardconn]", "Clearing arrays");
300             valuesECG1.clear();
301             valuesECG2.clear();
302             valuesECG3.clear();
303         }
304
305         for (int i=0;i<data[0].length;i++){
306             //Log.d("[Ardconn]", "ECG1: "+values[3][i]+", "+data[0][i]);
307             valuesECG1.add(new Entry(values[3][i],data[0][i]));
308             //Log.d("[Ardconn]", "ECG2: "+times[i]+", "+data[1][i]);
309             valuesECG2.add(new Entry(values[3][i],data[1][i]));
310             //Log.d("[Ardconn]", "ECG3: "+times[i]+", "+data[2][i]);
311             valuesECG3.add(new Entry(values[3][i],data[2][i]));
312             //Log.d("[Ardconn]", " ");
313         }
314
315
316
317         LineDataSet set1 = new LineDataSet(valuesECG1, "ECG1");
318         LineDataSet set2 = new LineDataSet(valuesECG2, "ECG2");
319         LineDataSet set3 = new LineDataSet(valuesECG3, "ECG3");
320         LineData lineData1 = new LineData(set1);
321         LineData lineData2 = new LineData(set2);
322         LineData lineData3 = new LineData(set3);
323
324         switch(showChart){
325             case 1:
326                 chart.getDescription().setText("ECG1");
327                 updateChart(chart,lineData1);
328                 break;
329             case 2:
330                 chart.getDescription().setText("ECG2");
331                 updateChart(chart,lineData2);
332                 break;
333             case 3:
334                 chart.getDescription().setText("ECG3");
335                 updateChart(chart,lineData3);
336                 break;
337         }
338     }
339
340     private void updateChart(LineChart chart, LineData data) {
341         //Log.d("[LedControl]", "updateChart called");
342         ((LineDataSet) data.getDataSetByIndex(0)).setLineWidth(1.75f);
343         ((LineDataSet) data.getDataSetByIndex(0)).setColor(Color.BLACK);
344         ((LineDataSet)
345             data.getDataSetByIndex(0)).setHighLightColor(Color.BLACK);
346         ((LineDataSet) data.getDataSetByIndex(0)).setDrawValues(true);
347         ((LineDataSet) data.getDataSetByIndex(0)).setDrawCircles(false);

```

```
347     // add data
348     chart.setVisibleXRangeMaximum(3);
349     //chart.setVisibleYRangeMaximum(2,);
350     if(scrolling){
351         chart.moveViewToX(data.getXMax());
352     }
353
354     // get the legend (only possible after setting data)
355     Legend l = chart.getLegend();
356     l.setEnabled(false);
357     //Refresh
358     chart.setData(data);
359     chart.notifyDataSetChanged();
360     chart.invalidate();
361
362 }
363 }
```