Τεχνική έκθεση

Σύστημα χαμηλής πολυπλοκότητας με Αισθητήρα αναπνευστικού κινδύνου και σύστημα συναγερμού

Illinois RapidVent

Πανεπιστήμιο Δυτικής Αττικής

Students: Κολίτση Φωτεινή 48038
Παπαδόπουλος Χρήστος 6927

Email: ele48038@uniwa.gr
ee6927@uniwa.gr

Web : https://www.uniwa.gr/

Αγίου Σπυρίδωνος 28, Αιγάλεω 122 43

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΕΡΙΛΗΨΗ 3

1. ΕΙΣΑΓΩΓΗ 4

2. ΠΑΡΟΥΣΙΑΣΗ ΕΡΓΟΥ RapidVent 5

2.1 ΛΕΙΤΟΥΡΓΙΑ ΑΝΑΠΝΕΥΣΤΗΡΑ 5

2.2 ΣΥΣΤΗΜΑ ΑΙΣΘΗΤΗΡΩΝ 6

2.3 ΜΕΤΑΦΟΡΑ ΔΕΔΟΜΕΝΩΝ ΠΙΕΣΗΣ 7

2.4 ΕΛΕΓΧΟΣ ΚΥΚΛΟΥ ΑΝΑΠΝΟΗΣ 8

2.5 ΠΡΟΥΠΟΘΕΣΕΙΣ ΣΥΝΑΓΕΡΜΟΥ 9

2.6 ΙΔΙΟΤΗΤΕΣ ΑΛΓΟΡΙΘΜΟΥ 10

2.7 ΠΕΙΡΑΜΑΤΙΚΕΣ ΔΟΚΙΜΕΣ ΑΝΑΠΝΕΥΣΤΗΡΑ 11

3. ΥΛΟΠΟΙΗΣΗ ΚΑΤΑΣΚΕΥΗΣ 12

3.1 ΠΡΟΔΙΑΓΡΑΦΕΣ 12

3.2 ΑΠΑΙΤΟΥΜΕΝΑ ΥΛΙΚΑ 13

3.3 ΕΡΓΑΛΕΙΑ 15

3.4 ΣΥΝΔΕΣΜΟΛΟΓΙΑ ΚΥΚΛΩΜΑΤΟΣ 16

3.5 ΠΛΑΚΕΤΑ ΤΥΠΩΜΕΝΟΥ ΚΥΚΛΩΜΑΤΟΣ 17

3.6 ΠΕΡΙΒΛΗΜΑ ΣΥΣΤΗΜΑΤΟΣ 18

4. ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΤΟΥ ΜΙΚΡΟΕΛΕΓΚΤΗ 19

4.1 ΟΔΗΓΙΕΣ ΕΓΚΑΤΑΣΤΑΣΗΣ ΛΟΓΙΣΜΙΚΟΥ 19

4.2 ΚΩΔΙΚΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ 22

4.2.1 ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ 22

4.2.2 ΕΞΑΓΩΓΗ ΔΕΔΟΜΕΝΩΝ 27

5. ΠΑΡΑΤΗΡΗΣΕΙΣ-ΣΥΜΠΕΡΑΣΜΑΤΑ 28

6. ΕΓΧΕΙΡΙΔΙΟ ΧΡΗΣΗΣ RAPIDVENT 29

6.1.1 ΕΓΚΑΤΑΣΤΑΣΗ 29

6.1.2 ΕΝΔΕΙΞΕΙΣ ΟΘΟΝΗΣ 30

6.1.3 ΛΕΙΤΟΥΡΓΙΕΣ ΟΘΟΝΗΣ 31

6.1.4 ΣΥΝΘΗΚΕΣ ΣΥΝΑΓΕΡΜΟΥ 32

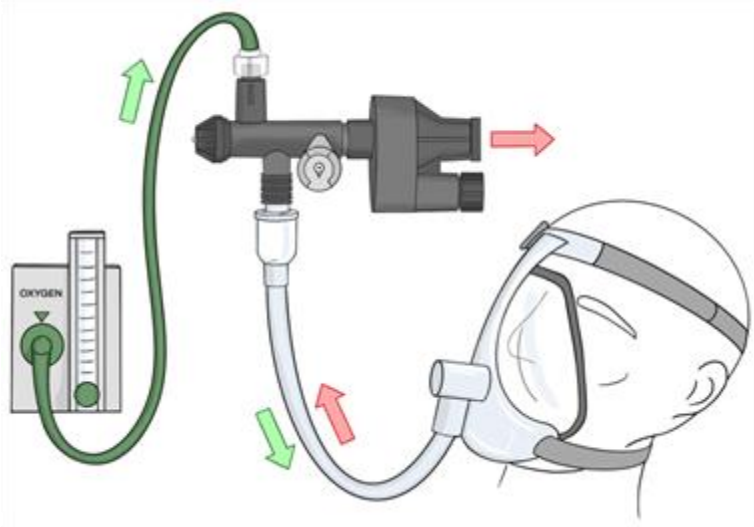
7. ΠΑΡΑΡΤΗΜΑ 33

ΠΕΡΙΛΗΨΗ

Η επέλαση της πανδημίας COVID-19, δημιούργησε έντονα προβλήματα στον εξοπλισμό των δομών υγείας ανά το κόσμο. Αρκετές εταιρείες προέβησαν σε παραγωγή αναπνευστήρων χαμηλού κόστους, οι οποίοι είναι εφικτό να παραχθούν, ωστόσο υστερούν σε εργονομία και έλλειψη συναγερμού για περιπτώσεις έκτακτης ανάγκης. Σε αυτή την εργασία παρουσιάζεται η μελέτη και κατασκευή ενός χαμηλού κόστους συστήματος συναγερμού σε περιπτώσεις δυσλειτουργίας του αναπνευστήρα. Χρησιμοποιώντας αισθητήρες, συλλέγονται μετρήσεις για την πίεση και τον αναπνευστικό ρυθμό του ανθρώπου, ενεργοποιώντας συναγερμό σε κατάσταση έκτακτης ανάγκης. Ο αλγόριθμος που χρησιμοποιείται απαιτεί λίγη μνήμη και εκτελεί μικρό αριθμό υπολογισμών για κάθε δείγμα, ώστε να είναι συμβατός με σχεδόν οποιοδήποτε μικροελεγκτή.

This is Apollo 13...
We have a team of brilliant and dedicated people that made something that actually works in less than one week. It's very inspiring. We hope that we can engage even more people to work on the global response to COVID-19 as we continue to develop the prototype."

William King, professor in The Grainger College of Engineering and the Carle Illinois College of Medicine



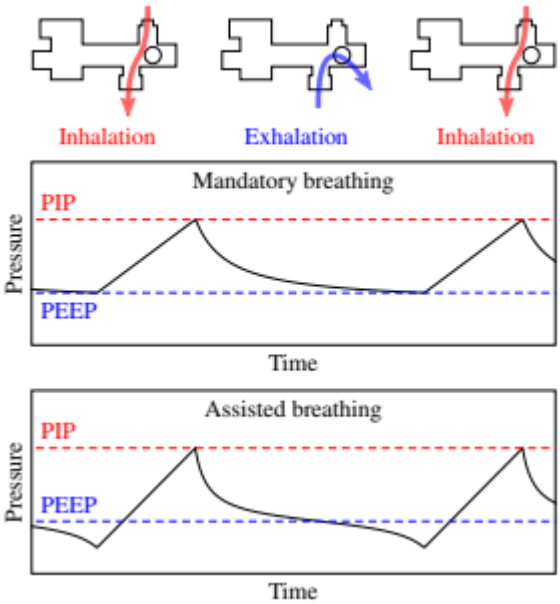
Εικόνα 1 Σχηματική αναπαράσταση αναπνευστήρα

1. ΕΙΣΑΓΩΓΗ

Ο ιός COVID-19, σε κάποιους ασθενείς μπορεί να προκαλέσει πολύ σοβαρά αναπνευστικά προβλήματα, συνήθως είναι η οξεία αναπνευστική δυσχέρεια, το σύνδρομο (ARDS), το οποίο προκαλεί υπερβολική δυσκολία στην αναπνοή λόγω διαρροής υγρού στους πνεύμονες. Η θεραπεία αυτής της κρίσης μπορεί να επιτευχθεί με έναν αναπνευστήρα, παρέχοντας οξυγόνο σε καίριες στιγμές. Η δημιουργία εξοπλισμού με επιπλέον αισθητήρες και συστήματα συναγερμού, αποδεσμεύει το ιατρικό προσωπικό από την συνεχή παρακολούθηση του ασθενή για την εξέλιξη της θεραπείας του, παρέχοντας τη δυνατότητα για επίβλεψη περισσότερων ασθενών ταυτοχρόνως.

Η λειτουργία του αναπνευστήρα που παρουσιάζεται στην εργασία, βασίζεται στον κύκλο λειτουργίας της αναπνοής, χρησιμοποιώντας αισθητήρες πίεσης και ειδοποιώντας ηχητικά σε περίπτωση δυσλειτουργίας της αναπνοής. Η παρακολούθηση της αναπνοής πραγματοποιείται βασιζόμενη στην εναλλαγή της πίεσης που δημιουργείται μεταξύ της εισπνοής και της εκπνοής. Οι δυσλειτουργίες εντοπίζονται όταν υπάρχουν μεταβολές στο σήμα της πίεσης. Οι κλινικές παράμετροι που είναι ιδιαίτερα χρήσιμες για απόδοση της περιγραφής είναι η μέγιστη πίεση εισπνοής (peak inspiratory pressure (PIP)),η θετική πίεση στο τέλος της εκπνοής (Positive end-expiratory pressure(PEEP))και ο αναπνευστικός ρυθμός (respiratory rate(RR)).

Οι μετρήσεις που λαμβάνει το σύστημα, αφορούν επίσης και τον όγκο ή την ποσότητα του αέρα που παρέχεται σε κάθε αναπνοή καθώς και την συγκέντρωση του οξυγόνου στον αέρα, δημιουργώντας τα κατάλληλα σήματα συναγερμού σε περιπτώσεις δυσλειτουργίας. Ωστόσο με τον χαμηλού κόστους εξοπλισμό, δεν είναι δυνατό να αντληθούν πληροφορίες για τη συγκέντρωση του οξυγόνου στον αέρα από τις κυματομορφές πίεσης. Επομένως για την επεξεργασία πληροφοριών της συγκέντρωσης οξυγόνου στον αέρα, χρησιμοποιείται ένας αλγόριθμος χαμηλής πολυπλοκότητας για την επεξεργασία σήματος, βασισμένο στην λειτουργία των ακουστικών βαρηκοΐας. Η επιλογή ενός αλγορίθμου με μικρές απαιτήσεις μνήμης, οφείλεται στο γεγονός ότι μπορεί να χρησιμοποιηθεί από σχεδόν οποιοδήποτε μικροελεγκτή.



Εικόνα 2 Ένας αναπνευστήρας χρησιμοποιεί θετική πίεση για να παράσχει οξυγόνο. Κατά τη διάρκεια της κανονικής λειτουργίας, παράγει μια ξεχωριστή κυματομορφή πίεσης.

2. ΠΑΡΟΥΣΙΑΣΗ ΕΡΓΟΥ RAPIDVENT

2.1 ΛΕΙΤΟΥΡΓΙΑ ΑΝΑΠΝΕΥΣΤΗΡΑ

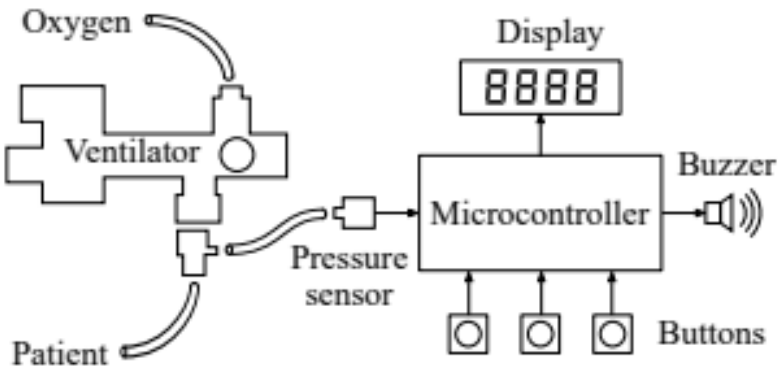
Οι αναπνευστήρες που τροφοδοτούνται από πεπιεσμένο αέρα είναι ιδιαιτέρως χρήσιμοι, επειδή έχουν χαμηλό κόστος κατασκευής και δεν απαιτούνται ηλεκτρονικά εξαρτήματα για βασικές λειτουργίες. Παρέχουν συμπιεσμένο αέρα στον αεραγωγό του ασθενούς και κάνουν κύκλο μεταξύ των τρόπων εισπνοής και εκπνοής βάσει πνευματικής λογικής, δηλαδή ο πεπιεσμένος αέρας είναι το μέσο ελέγχου αντί του ηλεκτρικού ρεύματος. Κατά την εισπνοή, η πίεση στον αεραγωγό αυξάνεται μέχρι να φτάσει σε ένα μέγιστο όριο πίεσης που ρυθμίζεται από τον χρήστη. Μόλις φτάσει στο μέγιστο όριο πίεσης, απελευθερώνεται ο αέρας στο περιβάλλον. Όταν η πίεση πέσει κάτω από συγκεκριμένο όριο, ένα ελατήριο κλείνει το μονοπάτι προς την ατμόσφαιρα για να αρχίσει η εισπνοή. Εάν το κύκλωμα αερίου εμποδιστεί ή αποσυνδεθεί, ο διαμορφωτής θα σταματήσει την ανακύκλωση μεταξύ των τρόπων εισπνοής και εκπνοής, προκαλώντας το σήμα πίεσης να παραμείνει σταθερό. Το προτεινόμενο σύστημα συναγερμού χρησιμοποιεί απλούς αλγόριθμους επεξεργασίας σήματος για την ανίχνευση αυτής της κατάστασης σταθερής πίεσης.

Κατά τη λειτουργία υποβοηθούμενης αναπνοής, επίσης γνωστή ως λειτουργία υποστήριξης πίεσης, ο ασθενής ξεκινά μια εισπνοή για να τραβήξει την πίεση κάτω από το όριο PEEP. Σε πνευματικούς αναπνευστήρες, το κατώφλι PEEP είναι σταθερό και ρυθμίζεται από τον κατασκευαστή. Εξαιτίας της ιδιομορφίας που παρουσιάζεται σε ασθενείς με COVID-19, τα επίπεδα μέγιστης πίεσης αναπνοής, διαφέρουν από τα όρια που έχουν οι κλασικοί αναπνευστήρες. Επειδή οι αναπνευστικοί κύκλοι πίεσης παράγουν καλά καθορισμένες κυματομορφές πίεσης κατά την κανονική λειτουργία, το σήμα της πίεσης μπορεί επίσης να χρησιμοποιηθεί για τον εντοπισμό δυσλειτουργιών.

2.2 ΣΥΣΤΗΜΑ ΑΙΣΘΗΤΗΡΩΝ

Ένα από τα ιδιαίτερα σημαντικά χαρακτηριστικά του συστήματος είναι πως ο σχεδιασμός του, δεν έχει υλοποιηθεί σε συγκεκριμένο μοντέλο αναπνευστήρα, γεγονός που το καθιστά προσαρμόσιμο σε οποιοδήποτε μηχάνημα υποστήριξης αναπνοής. Λόγω της έκτακτης ανάγκης για παρουσία αναπνευστήρων σε δομές υγείας, ο σχεδιασμός του βασίζεται στο χαμηλό κόστος και στην ευκολία της κατασκευής. Τα ανταλλακτικά του είναι ευρέως διαθέσιμα και μπορεί να υλοποιηθεί σε μια πλακέτα τυπωμένου κυκλώματος. Η συσκευή συνδέεται με τον αεραγωγό του ασθενούς χρησιμοποιώντας τυπικό αναπνευστικό προσαρμογέα σωληνώσεων, συνδεδεμένο στην πλευρά του αναπνευστήρα του ασθενούς.

Το ηλεκτρονικό σύστημα αποτελείται από έναν μικροελεγκτή, μια οθόνη, 3 κουμπιά, έναν βομβητή και έναν αισθητήρα πίεσης. Χρησιμοποιείται 8-bit μικροελεγκτής που περιλαμβάνει αναλογοψηφιακό μετατροπέα και αρκετές ψηφιακές εισόδους και εξόδους. Ο αλγόριθμος παρακολούθησης υλοποιείται σε γλώσσα C, όπου είναι φιλική προς το χρήστη. Τα κουμπιά λειτουργούν ως διεπαφή χρήστη και χρησιμοποιούνται για την ενεργοποίηση και απενεργοποίηση του συναγερμού καθώς και για την προσαρμογή των ρυθμίσεών του. Ο συναγερμός είναι ένας πιεζοηλεκτρικός βομβητής. Το βασικό στοιχείο του κυκλώματος είναι ο αισθητήρας πίεσης, ο οποίος μετατρέπει την πίεση του αέρα σε ηλεκτρικά σήματα, τα οποία μεταδίδονται στον αναλογοψηφιακό μετατροπέα του μικροελεγκτή.

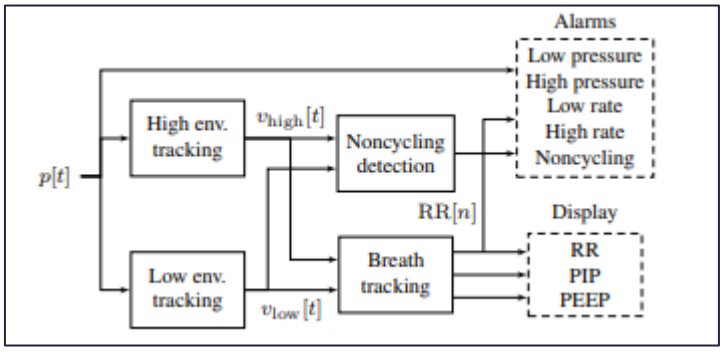


Εικόνα 3 Διάγραμμα κυκλώματος

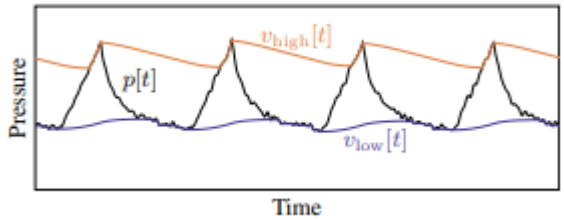
2.3 ΜΕΤΑΦΟΡΑ ΔΕΔΟΜΕΝΩΝ ΠΙΕΣΗΣ

Η συμπεριφορά των αναπνευστικών κύκλων πίεσης χαρακτηρίζεται καλά από το σήμα πίεσης που μετριέται στον αεραγωγό του ασθενούς. Κατά τη διάρκεια της κανονικής λειτουργίας, η πίεση περιστρέφεται μεταξύ PIP και PEEP μία φορά ανά αναπνοή. Σε ιδανικό σύστημα, οι αναπνοές θα μπορούσαν να παρακολουθούνται απλά βρίσκοντας τα μέγιστα και ελάχιστα σημεία της κυματομορφής του σήματος. Ωστόσο, σε πραγματικές συνθήκες υπάρχουν αυξομειώσεις, επομένως ο αλγόριθμος παρακολούθησης πρέπει να είναι ισχυρός και να εκτελεί λίγες πράξεις ανά δείγμα, ώστε να μπορεί να λειτουργεί με φθηνό, χαμηλής ισχύος μικροελεγκτή. Το προτεινόμενο σύστημα επεξεργασίας, που απεικονίζεται στην εικόνα 4, χρησιμοποιεί ένα ζευγάρι μη γραμμικών αναδρομικών φίλτρων για την παρακολούθηση του σήματος πίεσης, τα οποία διαθέτουν ανατροφοδότηση από την έξοδο στην είσοδο του φίλτρου, μπορούν να εκτελέσουν πολλαπλό φιλτράρισμα και εργασίες με λιγότερη μνήμη.

Εάν η μεταφορά των δεδομένων είναι πολύ αργή, μπορεί να χάσει τις αναπνοές όταν οι ρυθμίσεις πίεσης προσαρμόζονται ή, χειρότερα, μπορεί να καθυστερήσει ο συναγερμός όταν ο αναπνευστήρας σταματήσει να λειτουργεί. Κάθε αναδρομικός ανιχνευτής φακέλου αποθηκεύει μια προηγούμενη τιμή φακέλου στη μνήμη.



Εικόνα 5 Οι συνθήκες συναγερμού και οι κλινικές μετρήσεις προέρχονται από τη μέτρηση του σήματος πίεσης



Εικόνα 4 Ένα ζευγάρι αναδρομικών ανιχνευτών κορυφής

2.4 ΕΛΕΓΧΟΣ ΚΥΚΛΟΥ ΑΝΑΠΝΟΗΣ

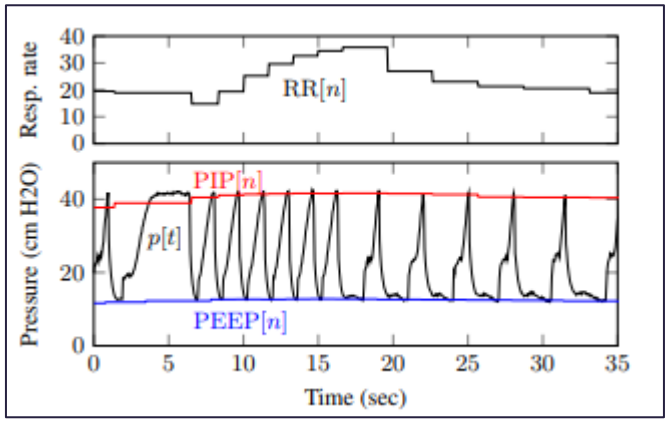
Το σύστημα παρακολούθησης εκτιμά τρεις δείκτες μετρήσεων: PIP, PEEP (ή την ελάχιστη πίεση του κύκλου αναπνοής για υποβοηθούμενη αναπνοή) και RR. Παρακολουθείται ο κύκλος αναπνοής, αναζητώντας συμβάντα χαμηλής πίεσης που ακολουθούν γεγονότα υψηλής πίεσης και το αντίστροφο. Χαμηλή πίεση εννοείται η κατάσταση που προκαλεί το σύστημα να αλλάξει από εισπνοή σε λειτουργία εκπνοής και ένα συμβάν υψηλής πίεσης προκαλεί εναλλαγή από εκπνοή σε λειτουργία εισπνοής. Οι δείκτες κατηγοριοποιούνται για καλύτερη ανάλυση, σύμφωνα με την επεξεργασία και τα δεδομένα που λαμβάνονται από τις μετρήσεις.

Α. Μέγιστη πίεση εισπνοής (peak inspiratory pressure (PIP)& θετική πίεση στο τέλος της εκπνοής (Positive end-expiratory pressure(PEEP))

Όταν εμφανίζεται ένας διακόπτης λειτουργίας, η προηγούμενη τιμή αυξομείωσης πίεσης χρησιμοποιείται για την ενημέρωση της αντίστοιχης εκτίμησης PIP ή PEEP. Σε κάθε συμβάν χαμηλής πίεσης, η οθόνη PIP ενημερώνεται με την πιο πρόσφατη τιμή υψηλής πίεσης.
Όταν εντοπίζεται τιμή υψηλής πίεσης, εμφανίζεται η ένδειξη PEEP και ενημερώνεται με την πιο πρόσφατη τιμή χαμηλής πίεσης. Τόσο το PIP όσο και το PEEP εξομαλύνονται με την πάροδο του χρόνου και χρησιμοποιείται ένας συντελεστής εξομάλυνσης μεταξύ 0 και 1. Αυτό είναι ένα γραμμικό φίλτρο με εκθετική απόκριση ώθησης.

Β. Αναπνευστικός ρυθμός (respiratory rate(RR))

Ένας πλήρης κύκλος αναπνοής μετριέται μεταξύ κορυφών υψηλής πίεσης. Η εικόνα 6 δείχνει τον εκτιμώμενο ρυθμό PIP, PEEP και αναπνευστικό ρυθμό που τοποθετούνται σε κυματομορφή πίεσης, η οποία μετριέται χρησιμοποιώντας τον αναπνευστήρα που συνδέεται με έναν τεχνητό πνεύμονα.



Εικόνα 6 Εμφανιζόμενες τιμές PIP, PEEP και αναπνευστικού ρυθμού για πειραματικά δεδομένα από τεχνητό πνεύμονα.

2.5 ΠΡΟΥΠΟΘΕΣΕΙΣ ΣΥΝΑΓΕΡΜΟΥ

Παρακολουθώντας τα δεδομένα, το σύστημα ενεργοποιεί το συναγερμό σε διάφορες συνθήκες όπου εντοπίζεται δυσλειτουργία του αναπνευστήρα. Τα κατώφλια συναγερμού μπορεί να διαφέρουν μεταξύ των ασθενών και μεταξύ συσκευών αναπνευστήρα και έτσι μπορούν να ρυθμιστούν από τον χρήστη. Οι παράμετροι που οριοθετούν τις ρυθμίσεις του συναγερμού αναλύονται σε υποκατηγορίες.

Α. Πίεση και αναπνευστικός ρυθμός

Οι συναγερμοί υψηλής και χαμηλής πίεσης ενεργοποιούνται αμέσως εάν ο αισθητήρας ανιχνεύει πίεση έξω από το επιτρεπόμενο εύρος. Σε ένα αναπνευστήρα, η πίεση δεν πρέπει ποτέ να υπερβαίνει την τιμή PIP που έχει οριστεί από τον χρήστη. Το κατώφλι χαμηλής πίεσης pmin μπορεί να οριστεί κοντά στο μηδέν, δηλαδή, ατμοσφαιρική πίεση, για ανίχνευση αποσύνδεσης στο κύκλωμα αναπνοής.

Β. Όροι μη ανατροφοδότησης

Η βασική ενεργοποίηση του συναγερμού, είναι η ανίχνευση για την διακοπή του κύκλου αναπνοής. Ενεργοποιείται σε περίπτωση που είναι αρκετή ώρα αμετάβλητα τα δεδομένα πίεσης και ακόμη σε περίπτωση που είναι πολύ κοντά οι τιμές της υψηλής και της χαμηλής πίεσης, επειδή γενικότερα η αναλογία των μεγεθών είναι μια σταθερά.

Γ. Επιλογή παραμέτρων

Στο εύρος των ορίων της πίεσης, ο λόγος του ανώτατου προς το κατώτατο άκρο, είναι ακόμη μια παράμετρος ενεργοποίησης του συναγερμού. Οι μεγάλες τιμές του συντελεστή εξομάλυνσης, μπορούν να επηρεάσουν την ικανότητα του συστήματος να προσαρμόζεται στις αλλαγές της πίεσης. Η ρύθμιση μεταξύ της σχέσης των 2 παραμέτρων, του συντελεστή εξομάλυνσης και της κατώτερης τιμής πίεσης, μπορεί να αποτρέψει επιπλοκές στην ενεργοποίηση του συναγερμού.

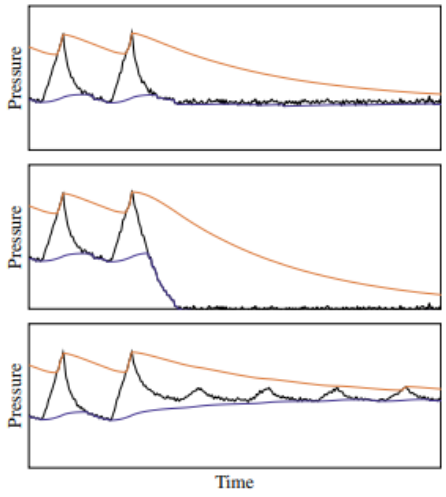


TABLE I
ALARM CONDITIONS

Alarm	Condition	Tunable range on RapidAlarm
High pressure	$p[t] > p_{\max}$	$30 \le p_{\max} \le 90$ cm H_2O
Low pressure	$p[t] < p_{\min}$	$1 \le p_{\min} \le 20$ cm H_2O
High RR	$RR[n] > RR_{\max}$	$15 \le RR_{\max} \le 60$ breath/min
Low RR	$RR[n] < RR_{\min}$	$5 \le RR_{\min} \le 15$ breath/min
Noncycling	$t - T_{\text{high}}[n] > T_{\max}$ $t - T_{\text{low}}[n] > T_{\max}$ $v_{\text{high}}[t]/v_{\text{low}}[t] < r_{\min}$ $v_{\text{high}}[t] - v_{\text{low}}[t] < d_{\min}$	$5 \le \frac{T_{\max}}{f_s} \le 30$ sec

Εικόνα 7 Παραδείγματα ρυθμίσεων ενεργοποίησης συναγερμού

2.6 ΙΔΙΟΤΗΤΕΣ ΑΛΓΟΡΙΘΜΟΥ

Ο αλγόριθμος παρακολούθησης εφαρμόζεται με χαμηλή υπολογιστική πολυπλοκότητα και μικρή κατανάλωση χώρου στην μνήμη. Η λειτουργία για την παρακολούθηση των κύκλων αναπνοής, περιλαμβάνει μια δυαδική μεταβλητή κατάστασης εισπνοής / εκπνοής. Ωστόσο, το πρόγραμμα πρέπει επίσης να αποθηκεύει τα όρια συναγερμού που μπορούν να ρυθμιστούν από το χρήστη στη μνήμη. Επειδή ο αναδρομικός ανιχνευτής φακέλου εκτελεί ένα σταθερό αριθμό υπολογισμών για κάθε δείγμα πίεσης, η συνολική υπολογιστική πολυπλοκότητα του συστήματος εξαρτάται από το ρυθμό δειγματοληψίας. Εάν ο δείκτης είναι πολύ χαμηλός, η ακολουθία δειγματοληψίας ενδέχεται να μην συλλάβει την μικρή κορυφή της κυματομορφής της πίεσης, προκαλώντας σφάλματα στις εκτιμώμενες τιμές PIP και RR. Για να εκτιμηθεί η υπολογιστική πολυπλοκότητα του συστήματος, ο χρόνος εκτέλεσης του αλγορίθμου μετριέται στον μικροελεγκτή με ταχύτητα ρολογιού 8 MHz.

Ο Πίνακας δείχνει την αποθήκευση, τη μνήμη και το χρόνο εκτέλεσης του αλγορίθμου παρακολούθησης και της λογικής διεπαφής χρήστη που ελέγχει τα κουμπιά, το βομβητή και την οθόνη. Ο αλγόριθμος παρακολούθησης απαιτεί λιγότερο από ένα millisecond ανά δείγμα.

RESOURCE UTILIZATION

	Program storage (bytes)	Dynamic memory (bytes)	Execution time per sample (s)
Interface	6847	357	146
Algorithm	4048	93	670
Total	10895	450	816

SAMPLE RATE AND PERFORMANCE

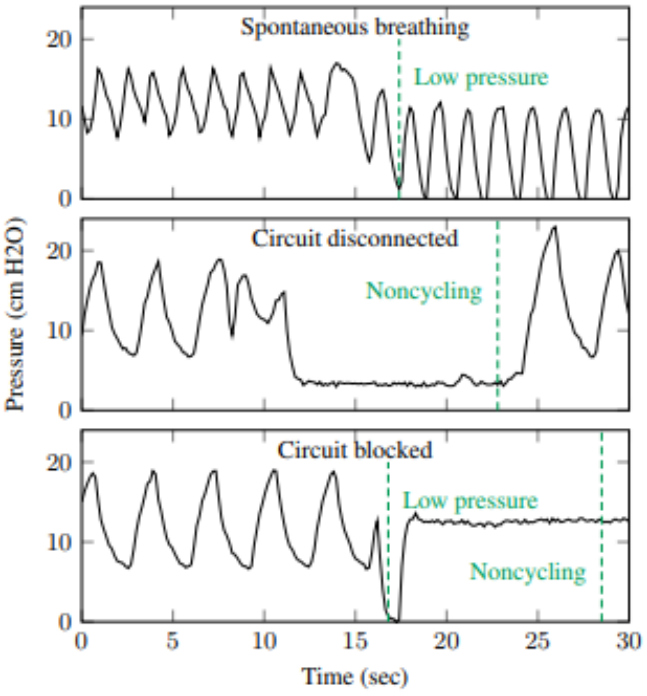
Sample rate (samples/sec)	RMS PIP error (cm H_2O)	RMS RR error (breaths/min)
5	1.7	5.4
10	0.5	0.4
20	0.6	0.2
50	0.3	0.1
100	Baseline	

Πίνακας 1 Χαρακτηριστικά αλγορίθμου

2.7 ΠΕΙΡΑΜΑΤΙΚΕΣ ΔΟΚΙΜΕΣ ΑΝΑΠΝΕΥΣΤΗΡΑ

Ο αλγόριθμος παρακολούθησης επικυρώθηκε χρησιμοποιώντας δεδομένα δοκιμών σε ζώα. Ο πρωτότυπος αναπνευστήρας δοκιμάστηκε σε κατασταλμένους χοίρους, οι οποίοι έχουν πνεύμονες που είναι παρόμοιοι σε μέγεθος με αυτούς των ανθρώπων. Ο πρωταρχικός σκοπός των δοκιμών, ήταν η απόδοση του ίδιου του αναπνευστήρα και το πείραμα δεν περιλάμβανε τη πρωτότυπη συσκευή παρακολούθησης.

Στον πίνακα καταγράφηκαν διάφορες καταστάσεις δοκιμών του αναπνευστήρα. Στο πρώτο ο χοίρος εισπνεύστηκε αρκετά έντονα ενεργοποιώντας τον συναγερμό χαμηλής πίεσης, ο οποίος προειδοποιεί τους κλινικούς ότι ο ασθενής αναπνέει αυθόρμητα. Η μεσαία κυματομορφή απεικονίζει ότι εκούσια το αναπνευστικό κύκλωμα καταστράφηκε στη διάρκεια του πειράματος, επομένως η πίεση μένει σταθερή σε χαμηλό επίπεδο πυροδοτώντας τον συναγερμό διακοπής ανατροφοδότησης. Τέλος, ο αναπνευστήρας μπλοκαρίστηκε σκόπιμα για αρκετά δευτερόλεπτα με αποτέλεσμα το σταθερό επίπεδο υψηλής πίεσης, ενεργοποιώντας τους συναγερμούς χαμηλής πίεσης καθώς επίσης και της διακοπής ανατροφοδότησης. Ο αλγόριθμος παρακολούθησης βρέθηκε να λειτουργεί για όλες τις δοκιμασμένες ρυθμίσεις του PIP και τις κλήσεις ρυθμού, αν και ξαφνικές αλλαγές στις ρυθμίσεις κλήσης μπορεί να προκαλέσουν ψευδείς συναγερμούς και προκαλούν προσωρινές ανακρίβειες στις μετρήσεις.



Εικόνα 8 Κυματορφές καταστάσεων πειράματος. Οι διακεκομμένες γραμμές υποδεικνύουν συναγερμούς που ενεργοποιούνται από τον αλγόριθμο

3. ΥΛΟΠΟΙΗΣΗ ΚΑΤΑΣΚΕΥΗΣ

3.1 ΠΡΟΔΙΑΓΡΑΦΕΣ

Γενικά χαρακτηριστικά

Πληθυσμός ασθενών	*Ασθενείς που χρειάζονται αναπνευστική υποστήριξη*
Συμβατοί αναπνευστήρες	*RapidVent και παρόμοιοι αναπνευστικοί κύκλοι πίεσης*
Διεπαφή ασθενούς	*Συνδέεται μέσω σωλήνα εσωτερικής διαμέτρου 1/8 "(3,2mm) με την εφαρμογή του κυκλώματος αναπνοής*
Περιβάλλον φροντίδας	*Στατική ή φορητή (απαιτείται μπαταρία για φορητή χρήση)*
Μέγεθος	*82 mm x 48 mm x 15,5 mm*
Απαιτείται ισχύς	*5 V DC, 100 mA*

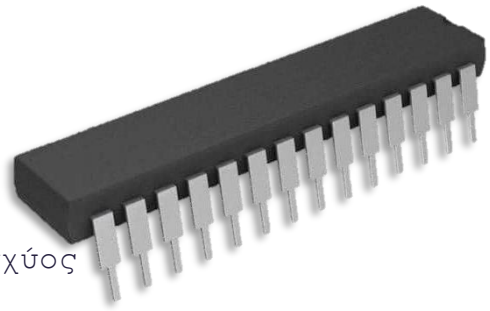
Ενδείξεις

Μετρικός	Εύρος οθόνης	Ανάλυση της οθόνης
PIP (Υψηλή πίεση)	0-99 cm H2O	1 cm H2O
PEEP (Χαμηλή πίεση)	0-99 cm H2O	1 cm H2O
Ρυθμός αναπνοής	0-99 αναπνοές / λεπτό	1 ανάσα / λεπτό

Συνθήκες συναγερμού

Κατάσταση	Προεπιλεγμένη ρύθμιση	Ρυθμιζόμενο εύρος
Μη ανατροφοδότηση	10 δευτερόλεπτα	5-30 δευτ
Χαμηλή πίεση	2 cm H2O	1-20 cm H2O
Υψηλή πίεση	40 cm H2O	30-90 cm H2O
Χαμηλός αναπνευστικός ρυθμός	6 αναπνοές / λεπτό	5-15 αναπνοές / λεπτό
Υψηλός αναπνευστικός ρυθμός	35 αναπνοές / λεπτό	15-60 αναπνοές / λεπτό

3.2 ΑΠΑΙΤΟΥΜΕΝΑ ΥΛΙΚΑ



1. Μικροελεγκτής ATMEGA-328

Είναι ένας 8-bit μικροελεγκτής χαμηλής ισχύος ο οποίος περιέχει:

-Εσωτερικό ρολόι 8MHz

-Αναλογοψηφιακό μετατροπέα

-23 ψηφιακές γραμμές I/O

-Τάση λειτουργίας 1.8 - 5.5V

2. Αισθητήρας πίεσης MPXV5010GC7U

-Μέγιστη πίεση 40 kPa

-Ανοχή θερμοκρασίας 0°C ~ 85°C

-Τάση τροφοδοσίας ~5V

-Ευαισθησία 450~4.413 mV/mm

-Χρόνος απόκρισης 1 ms

3. Βομβητής PKM13EPYH4002-B0

-Επίπεδα λειτουργίας 78dB [±1.5Vo-p,4.0kHz, square wave,10cm]

-Τάση λειτουργίας ±15.0 V

-Χωρητικότητα 5.5 ±30%[1kHz]nF

-Εύρος θερμοκρασιακής λειτουργίας -40 ~ +85 °C

4. Διακόπτης αφής 1825910-6

-Τάση λειτουργίας 1~24 V DC

-Ρεύμα λειτουργίας από 10μA~50mA

5. Πυκνωτής 1uF

-Ηλεκτρολυτικός
-Χωρητικότητας 1uF
-Τάση λειτουργίας 50V
-Υλικό κατασκευής αλουμίνιο

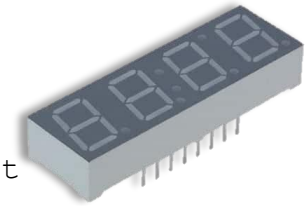
6. Πυκνωτής 0,1uF

-Κεραμικός
-Χωρητικότητας 0,1uF
-Τάση λειτουργίας 50V

7. Συστοιχία αντιστάσεων

-Αντίσταση 680 Ohm
-300mW Ισχύς ανά στοιχείο

8. Οθόνη LED 4 χαρακτήρων



-Μέγιστη απόδοση ισχύος ανά τμήμα 70 mWatt
-Εύρος θερμοκρασίας -35,85
-Μέση τιμή φωτεινής έντασης 200-650 ucd στο 1mA
-Μέγιστή τάση ορθής φοράς ανά segment 2,6V 20mA

9. Αντίσταση 10KΩ

-Ισχύς 1/4 Watt 0,25Watt

10. Υποδοχή βύσματος

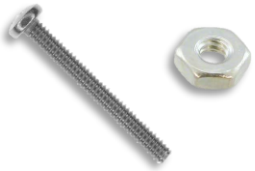
-Μέγιστο ρεύμα 5A
-Μέγιστη Τάση 30VDC

11. Σύνδεση Κεφαλίδας (προαιρετικά)

-Πλήθους 6 Ακίδων

12.Προσαρμογέας τάσης 5V

13.Βίδες 2-56x3 / 4 " & παξιμάδια 2-56

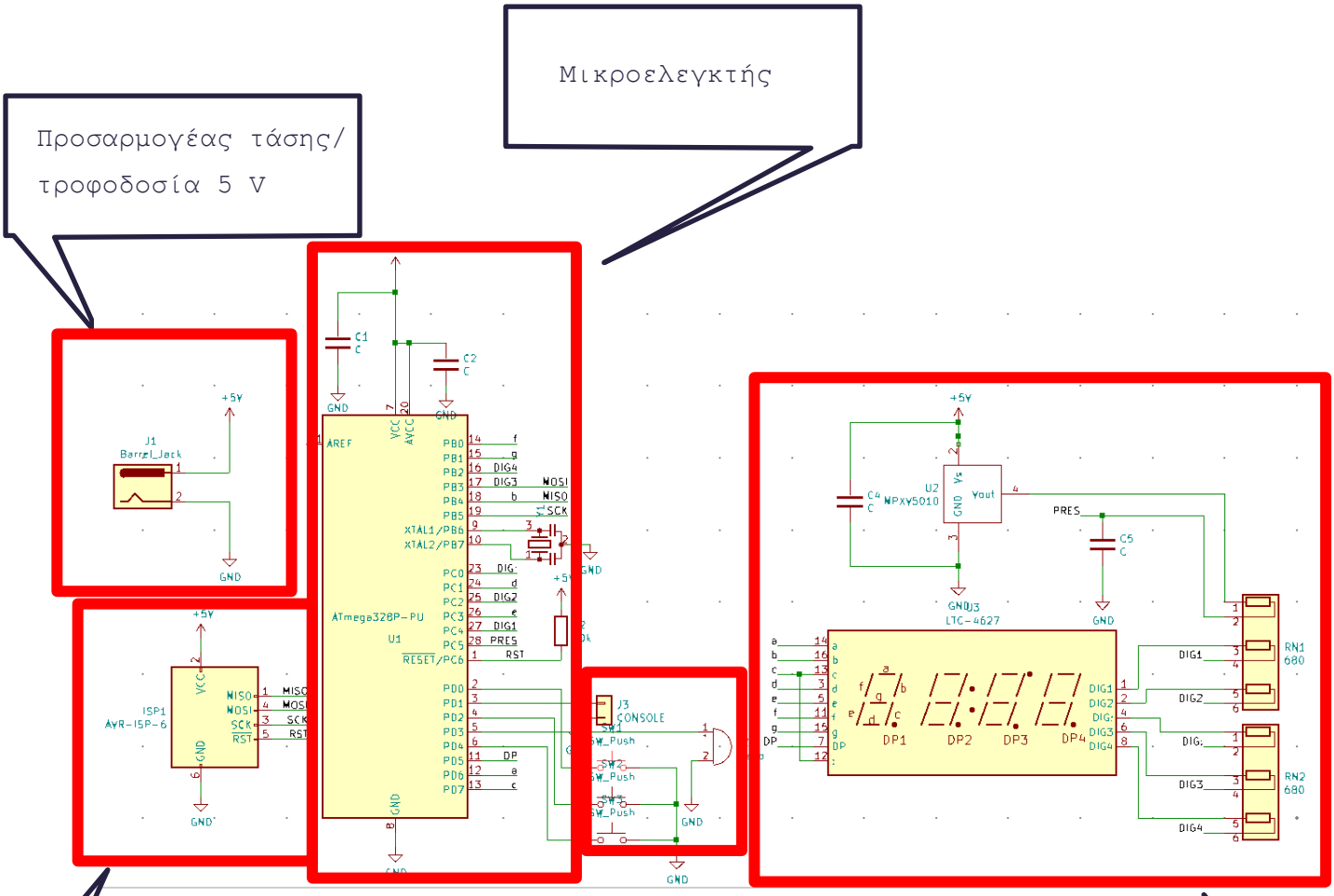


3.3 ΕΡΓΑΛΕΙΑ

Για την κατασκευή του RapidAlarm χρησιμοποιήθηκαν τα ακόλουθα εργαλεία και αναλώσιμα:

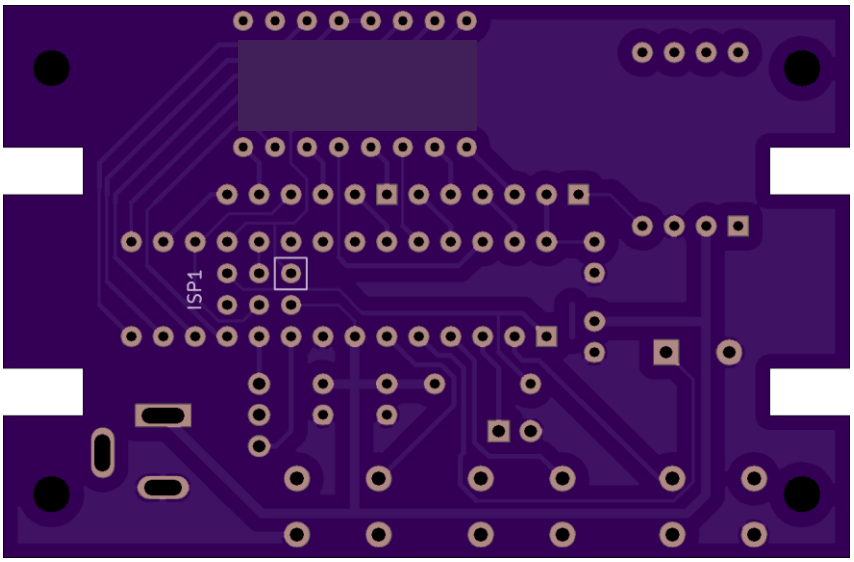
- Ένα συγκολλητικό σίδερο και κολλητήρι για τη συναρμολόγηση της πλακέτας κυκλώματος
- Ένα κατσαβίδι για τη συναρμολόγηση του περιβλήματος
- Ένας κατασκευαστής ετικετών ή ένας μόνιμος δείκτης για την επισήμανση των κουμπιών στο περίβλημα
- Ένας υπολογιστής και προγραμματιστής AVR για φόρτωση του υλικολογισμικού στον μικροελεγκτή.

3.4 ΣΥΝΔΕΣΜΟΛΟΓΙΑ ΚΥΚΛΩΜΑΤΟΣ

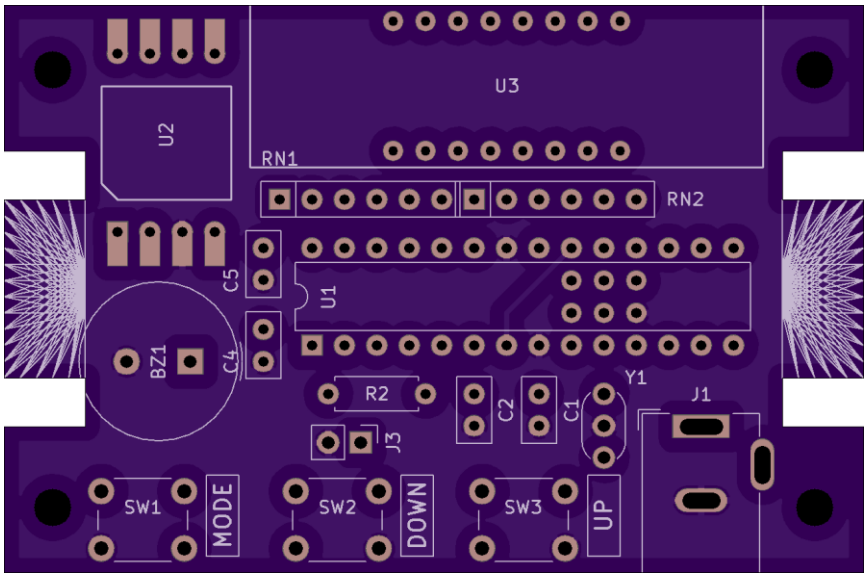


3.5 ΠΛΑΚΕΤΑ ΤΥΠΩΜΕΝΟΥ ΚΥΚΛΩΜΑΤΟΣ

Για την υλοποίηση της συνδεσμολογίας χρησιμοποιείται πλακέτα τυπωμένου κυκλώματος (PCB) 2 επιπέδων , διάτρητη. Το ενδεικτικό σχέδιο φαίνεται στις εικόνες.

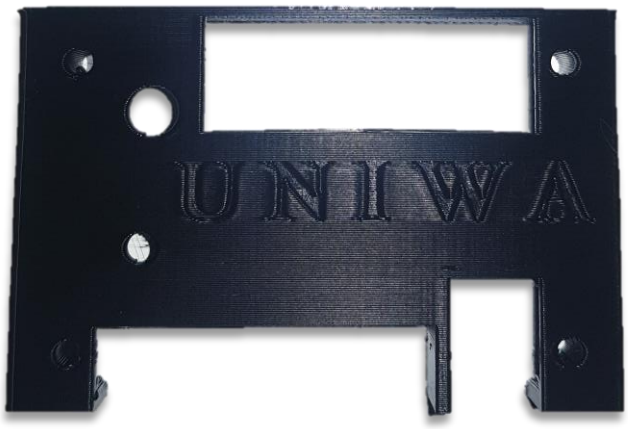


Εικόνα 10 Κάτω όψη PCB



Εικόνα 9 Πάνω όψη PCB

3.6 ΠΕΡΙΒΛΗΜΑ ΣΥΣΤΗΜΑΤΟΣ

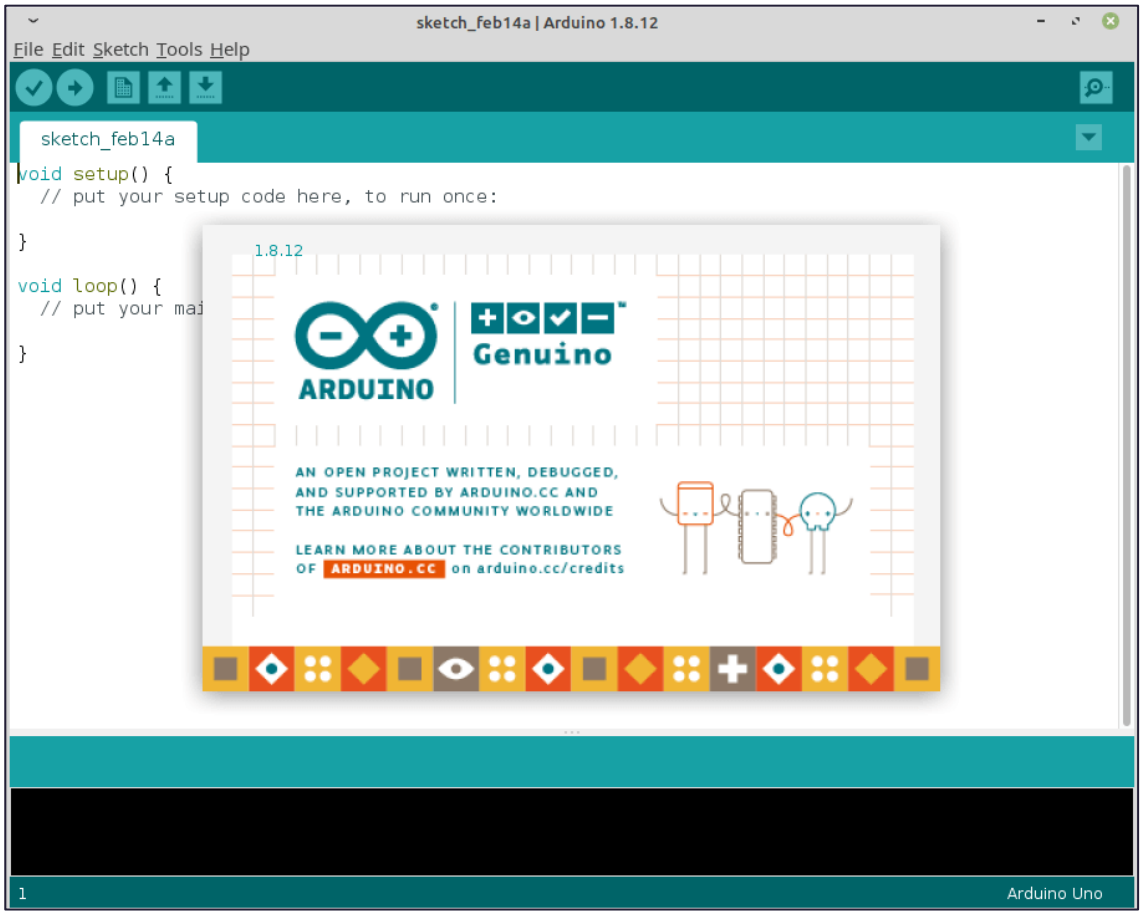


4. ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΤΟΥ ΜΙΚΡΟΕΛΕΓΚΤΗ

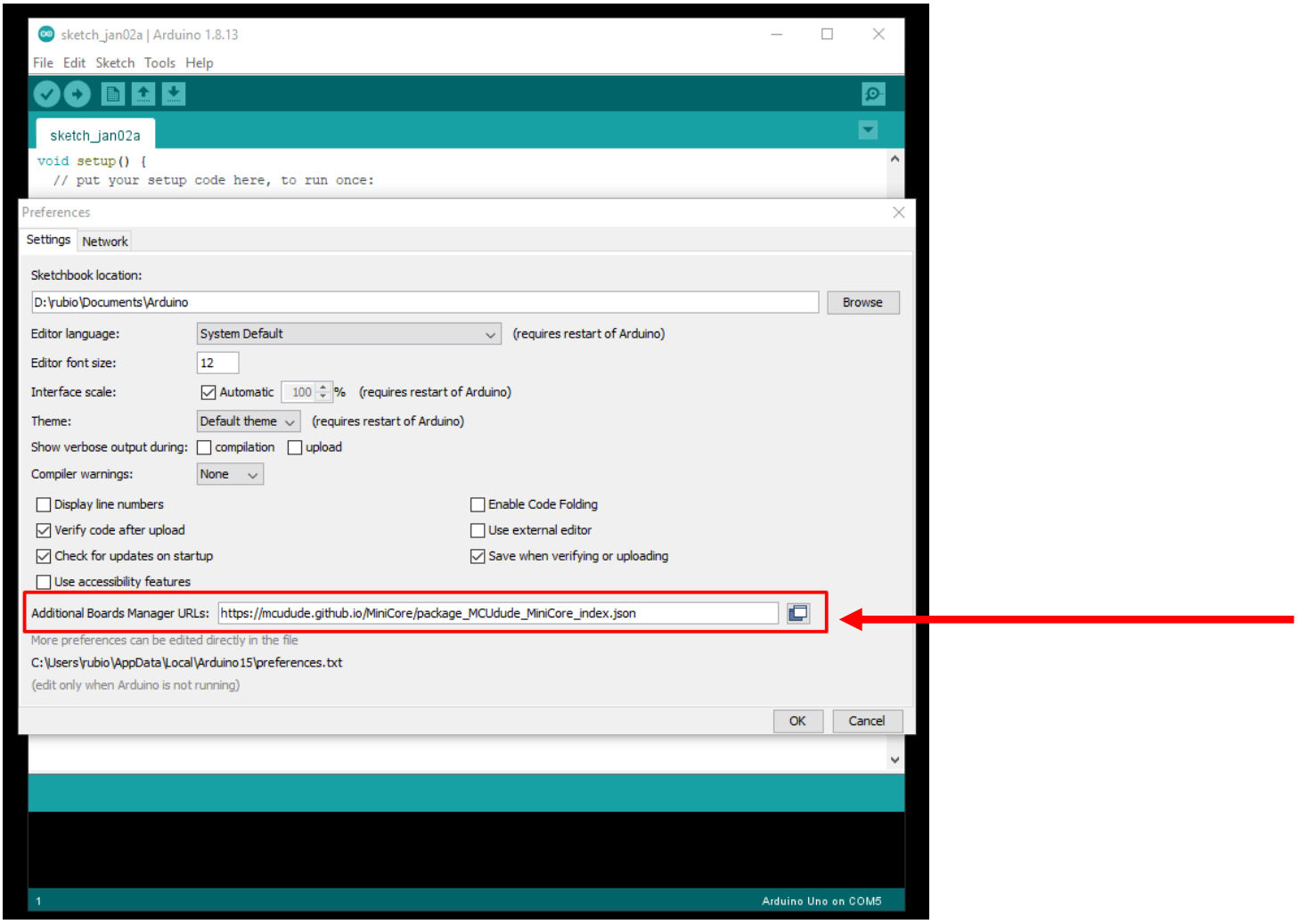
4.1 ΟΔΗΓΙΕΣ ΕΓΚΑΤΑΣΤΑΣΗΣ ΛΟΓΙΣΜΙΚΟΥ

Για τον προγραμματισμό του RapidAlarm χρησιμοποιείται η πλατφόρμα Arduino, για να απλοποιήσει την ανάπτυξη και τον προγραμματισμό του μικροελεγκτή. Αν και δεν υπάρχει υλικό Arduino στην κατασκευή, το ενσωματωμένο περιβάλλον ανάπτυξης Arduino IDE χρησιμοποιείται για τη φόρτωση του υλικολογισμικού στον μικροελεγκτή.

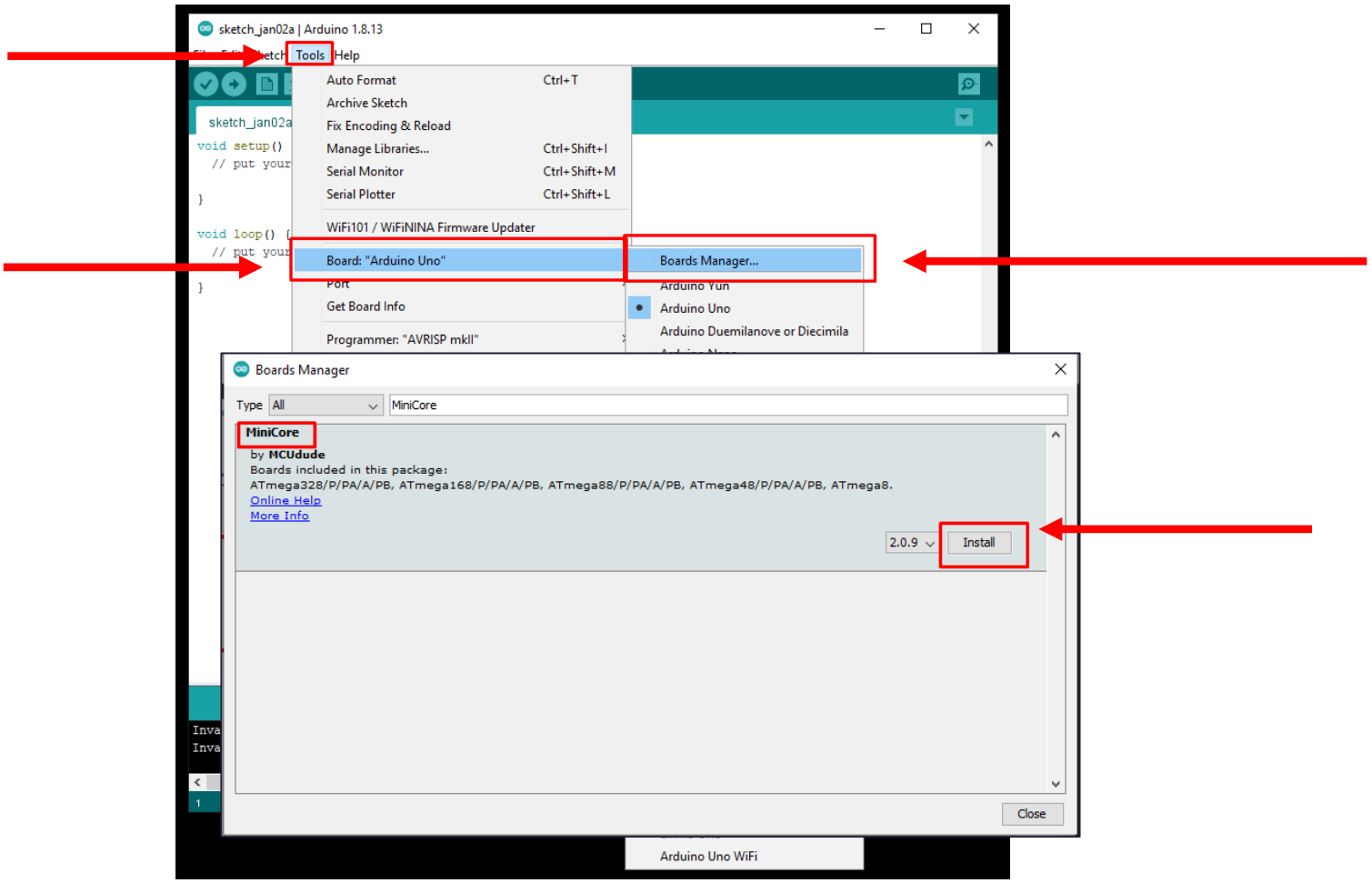
Βήμα 1ο : Εγκατάσταση του Arduino IDE στον υπολογιστή



Βήμα 2° : Προσθήκη βιβλιοθήκης MiniCore



Βήμα 3° : Εγκατάσταση της βιβλιοθήκης MiniCore



Βήμα 4° : Ρύθμιση παραμέτρων στο λογισμικό

Board	ATmega328
Variant	328P / 328PA
Bootloader	No bootloader
BOD	BOD 2.7V
Clock	Internal 8MHz
Compiler LTO	LTO Disabled
Programmer	USBasp

4.2 ΚΩΔΙΚΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

4.2.1 ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ

```
#include <stdlib.h>
#include <stdbool.h>
#include <unistd.h>
#include <stdint.h>
#include <string.h>
```

Συμπερίληψη βιβλιοθηκών στο έργο

```
const uint8_t SAMPLE_RATE = 100;
```

ο ρυθμός δειγματοληψίας του αισθητήρα πίεσης

```
const uint8_t PIN_MODE = 2;
const uint8_t PIN_UP = 0;
const uint8_t PIN_DOWN = 4;
const uint8_t PIN_BUZZ = 3;
const uint8_t PIN_PRES = A5;
const byte digit_pins[] = {A4, A2, A0, 11, 10};
const byte segment_pins[] = {6, 12, 7, A1, A3, 8, 9, 5};
```

Διαμόρφωση εισόδων και εξόδων

```
const uint8_t volume = 5;
```

Ένταση ήχου βομβητή

```
const uint16_t debounce_time = 200;
```

Χρόνος απενεργοποίησης του κουμπιού σε miliseconds

```
#include "algorithm.h"
#include "display.h"
#include "alarm.h"
```

Συμπερίληψη υποροutίνων στο έργο

```
ISR (PCINT2_vect)
{

    if ((PIND & (1 << PD0)) == 0 && debounce()) {
        mode_pressed = true;
    }

    if ((PIND & (1 << PD4)) == 0 && debounce()) {
        up_pressed = true;
    }

    if ((PIND & (1 << PD2)) == 0 && debounce()) {

    }
}
```

Ρύθμιση για χειροκίνητο έλεγχο των κουμπιών.
PD0=λειτουργία διεπαφής
PD1=κάτω κουμπί
PD2=πάνω κουμπί

```
void setup() {
    Serial.begin(115200);
    Serial.println("start");
```

Λειτουργία αισθητήρα πίεσης

```
    pinMode(A6, INPUT);
    pinMode(A7, INPUT);
```

Είσοδος δεδομένων από τον αισθητήρα πίεσης

```
    pinMode(PIN_MODE, INPUT);
    digitalWrite(PIN_MODE, HIGH);
    pci_setup(PIN_MODE);

    pinMode(PIN_UP, INPUT);
    digitalWrite(PIN_UP, HIGH);
    pci_setup(PIN_UP);

    pinMode(PIN_DOWN, INPUT);
    digitalWrite(PIN_DOWN, HIGH);
    pci_setup(PIN_DOWN);
```

Ρυθμίσεις πλήκτρων

```
    pinMode(PIN_BUZZ, OUTPUT);
    analogWrite(PIN_BUZZ, 0);
```

Ρύθμιση βομβητή

```
    TCCR2B = TCCR2B & 0b11111000 | 0b00000010;
```

Εύρος συχνότητας του βομβητή

```
    for(byte i = 0; i < sizeof(digit_pins); i++) {
        pinMode(digit_pins[i], OUTPUT);
    }
    for(byte i = 0; i < sizeof(segment_pins); i++) {
        pinMode(segment_pins[i], OUTPUT);
    }

    init_algorithm(SAMPLE_RATE);

}
```

Αρχικοποίηση των επαφών της οθόνης

```
void loop() {

    if (mode_pressed) {

        Serial.println("MODE pressed");
        mode_pressed = false;
```

Επιλογή συνθήκης για πάτημα του κουμπιού

```
        if (mode == MODE_ALARM) {

            alarm_raised = ALARM_NONE;
            analogWrite(PIN_BUZZ, 0);
            alarm_disabled = true;
            mode = MODE_DISPLAY;
        }
        else {
            mode = (mode + 1) % total_modes;
        }
    }
```

Επιλογή συνθήκης για ενεργοποίηση του συναγερμού και απενεργοποίησή του

```
    if (millis() - last_sample >= (1000 / SAMPLE_RATE)) {
        last_sample = millis();

        if (mode == 0) {
            p = ADC2CM((float) analogRead(PIN_PRES));

            run_algorithm(p, SAMPLE_RATE);

            if(alarm_raised) {
                mode = MODE_ALARM;
            }
```

Έλεγχος τελευταίας ειδοποίησης η οποία ενεργοποίησε το συναγερμό

```
            if (counter < counter_max / 3) {
```

Ενημέρωση της οθόνης κάθε 3 δευτερόλεπτα

```
                sprintf(displayedValue, "PI:%2d", (uint8_t) limit(pip, 0, 99));
            }
            else if (counter < 2 * counter_max / 3) {
```

Δείκτης PIP

**sprintf=String print*
Δεν εκτυπώνει την τιμή, αλλά την αποθηκεύει σε μεταβλητή

```
                sprintf(displayedValue, "PE:%2d", (uint8_t) limit(peep, 0, 99));
            }
            else {
```

Δείκτης PEEP

```
                sprintf(
                    displayedValue,
                    "rr:%2d",
                    (uint8_t) limit(respiration_rate, 0, 99)
                );
            }

        }
```

Δείκτης ρυθμού αναπνοής PR

```
else if (mode == MODE_SET_THRESH_NC) {
```

Ρύθμιση κατάστασης μη ανατροφοδότησης του συναγερμού

```
    if (up_pressed) {
        THRESH_NC = limit(THRESH_NC, 0, 25) + 5;
        up_pressed = false;
    }
    else if (down_pressed) {
        THRESH_NC = limit(THRESH_NC, 5, 35) - 5;
        down_pressed = false;
    }
```

Αλλαγή εύρους τιμών για την ενεργοποίηση του συναγερμού

```
    if (counter % (SAMPLE_RATE / 2) > SAMPLE_RATE / 8) {
        sprintf(displayedValue, "NC:%2d", THRESH_NC);
    } else {
        sprintf(displayedValue, "NC:  ");
    }
}
```

Ένδειξη ενεργοποίησης συναγερμού στην οθόνη

```
else if (mode == MODE_SET_THRESH_LP) {
```

Ρυθμίσεις συναγερμού για την χαμηλή πίεση

```
    if (up_pressed) {
        THRESH_LP = limit(THRESH_LP, 1, 19) + 1;
        up_pressed = false;
    }
    else if (down_pressed) {
        THRESH_LP = limit(THRESH_LP, 3, 21) - 1;
        down_pressed = false;
    }
```

Αλλαγή εύρους τιμών για την χαμηλή πίεση

```
    if (counter % (SAMPLE_RATE / 2) > SAMPLE_RATE / 8) {
        sprintf(displayedValue, "LP:%2d", THRESH_LP);
    } else {
        sprintf(displayedValue, "LP:  ");
    }
}
```

Ένδειξη ενεργοποίησης συναγερμού χαμηλής πίεσης στην οθόνη

```
else if (mode == MODE_SET_THRESH_HP) {
```

Ρυθμίσεις συναγερμού για την υψηλή πίεση

```
    if (up_pressed) {
        THRESH_HP = limit(THRESH_HP, 25, 85) + 5;
        up_pressed = false;
    }
```

Αλλαγή εύρους τιμών για την υψηλή πίεση

```
    if (counter % (SAMPLE_RATE / 2) > SAMPLE_RATE / 8) {
        sprintf(displayedValue, "HP:%2d", THRESH_HP);
    } else {
        sprintf(displayedValue, "HP:  ");
    }
}
```

Ένδειξη ενεργοποίησης συναγερμού υψηλής πίεσης στην οθόνη

```
else if (mode == MODE_SET_THRESH_LR) {
    if (up_pressed) {
        THRESH_LP = limit(THRESH_LR, 4, 14) + 1;
        up_pressed = false;
    }
    else if (down_pressed) {
        THRESH_LP = limit(THRESH_LR, 6, 16) - 1;
        down_pressed = false;
    }
    if (counter % (SAMPLE_RATE / 2) > SAMPLE_RATE / 8) {
        sprintf(displayedValue, "Lr:%2d", THRESH_LR);
    } else {
        sprintf(displayedValue, "Lr:  ");
    }
}
else if (mode == MODE_SET_THRESH_HR) {
    if (up_pressed) {
        THRESH_HR = limit(THRESH_HR, 10, 55) + 5;
        up_pressed = false;
    }
    else if (down_pressed) {
        THRESH_HR = limit(THRESH_HR, 20, 65) - 5;
        down_pressed = false;
    }
    if (counter % (SAMPLE_RATE / 2) > SAMPLE_RATE / 8) {
        sprintf(displayedValue, "Hr:%2d", THRESH_HR);
    } else {
        sprintf(displayedValue, "Hr:  ");
    }
}
else if (mode == MODE_ALARM) {
    if (counter % SAMPLE_RATE > SAMPLE_RATE / 2) {
        analogWrite(PIN_BUZZ, volume);
    }
  else {
      analogWrite(PIN_BUZZ, 0);
  }
  if (counter % (SAMPLE_RATE / 2) > SAMPLE_RATE / 8) {
      sprintf(
          displayedValue,
          "%s:%2d",
          alarm_code,
          (uint8_t) limit(alarm_value, 0, 99)
      );
  }
  else {
      sprintf(displayedValue, "     ");
  }
}
    counter++;
    counter %= counter_max;
  }

  update_display();

}
```

Ρυθμίσεις συναγερμού για χαμηλό ρυθμό αναπνοής

Αλλαγή εύρους τιμών για τον χαμηλό ρυθμό αναπνοής

Ένδειξη ενεργοποίησης συναγερμού για χαμηλό ρυθμό αναπνοής στην οθόνη

Ρυθμίσεις συναγερμού για υψηλό ρυθμό αναπνοής

Αλλαγή εύρους τιμών για τον υψηλό ρυθμό αναπνοής

Ένδειξη ενεργοποίησης συναγερμού για υψηλό ρυθμό αναπνοής στην οθόνη

Ρυθμίσεις συναγερμού

Ενεργοποίηση/απενεργοποίηση συναγερμού

Ρυθμίσεις οθόνης για τις καταστάσεις συναγερμού

Απαριθμητής μετρήσεων για ενημέρωση της ένδειξης της οθόνης

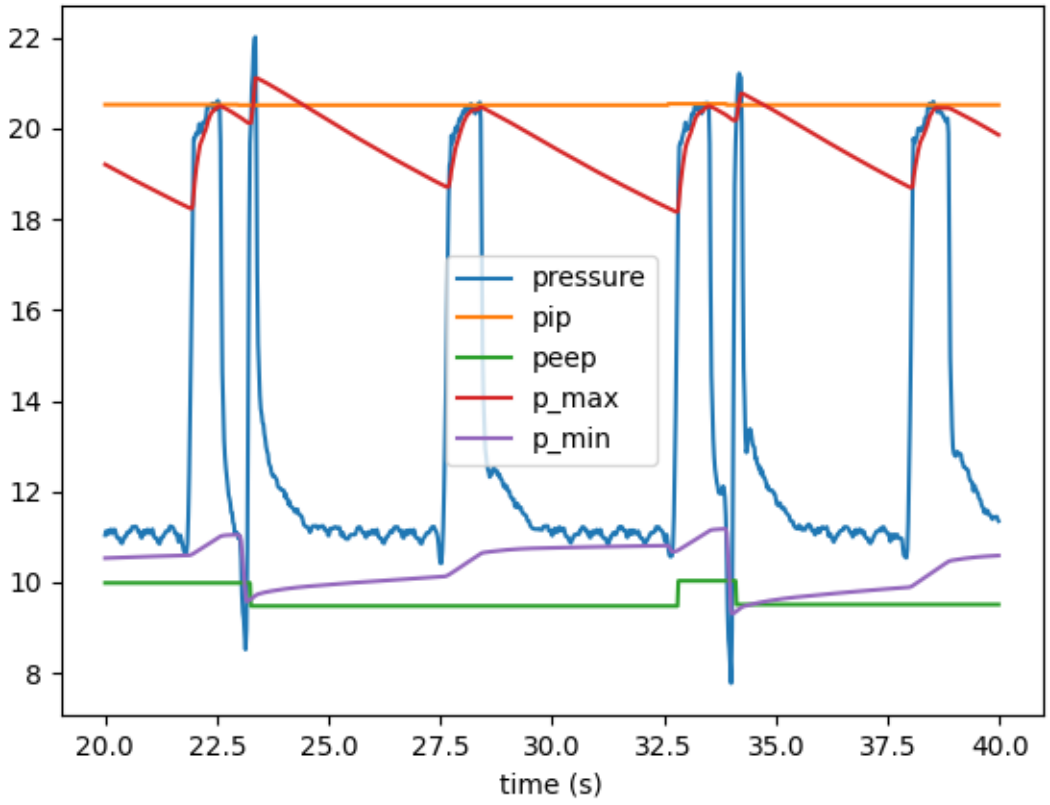
4.2.2 ΕΞΑΓΩΓΗ ΔΕΔΟΜΕΝΩΝ

Οι μετρήσεις που λαμβάνονται από τον αισθητήρα αποθηκεύονται σε αρχείο matlab. Για την δημιουργία των κυματομορφών, τα αρχεία μετατρέπονται σε μορφή csv. Στη συνέχεια φορτώνονται στο πρόγραμμα όλοι οι δείκτες των μετρήσεων και δημιουργείται η αντίστοιχη κυματομορφή.

Οι διαδικασίες που εκτελούνται με python:

-μετατροπή αρχείου μετρήσεων σε csv

-οι βιβλιοθήκες που χρησιμοποιούνται για την κατασκευή γραφημάτων είναι: *matplotlib, numpy, pandas.*



Εικόνα 11 Παράδειγμα κυματομορφών

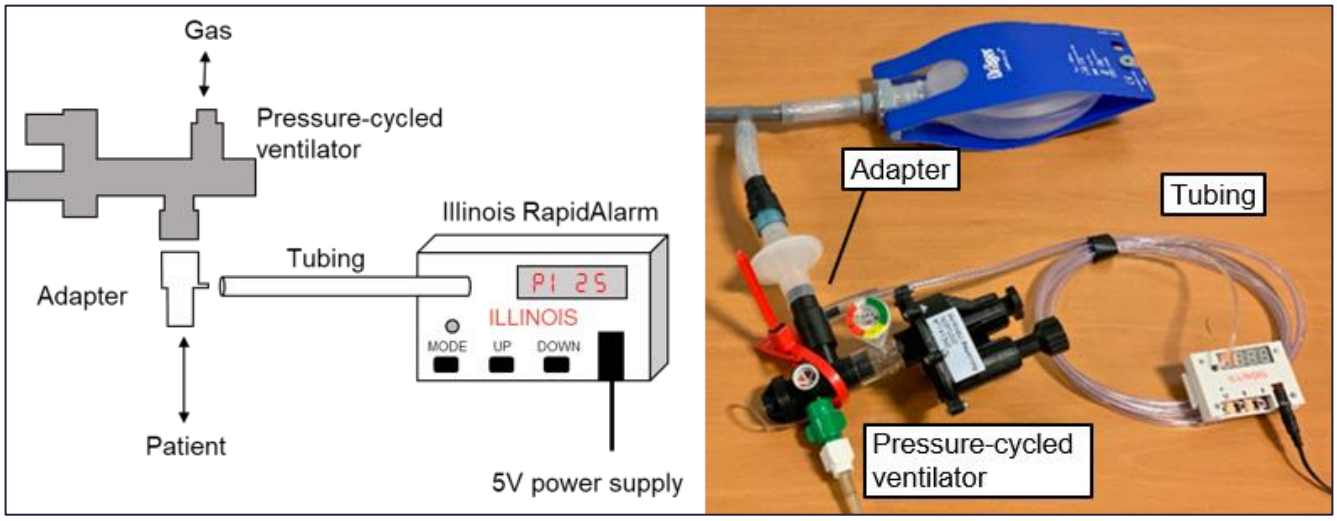
5. ΠΑΡΑΤΗΡΗΣΕΙΣ-ΣΥΜΠΕΡΑΣΜΑΤΑ

Συνοψίζοντας ένα σύνολο παρατηρήσεων που μπορούν να σημειωθούν για την κατασκευή του RapidAlarm είναι πως υπάρχει η δυνατότητα για αλλαγή των ρυθμίσεων συναγερμού , καθώς αυτές επαναφέρονται στις προεπιλογές κάθε φορά που διακόπτεται η τροφοδοσία.
Μόλις συνδεθεί η τροφοδοσία, το RapidAlarm θα ηχήσει μια φορά για να επαληθεύσει ότι ο βομβητής λειτουργεί, και στη συνέχεια θα αρχίσει να κυκλώνει τις μετρημένες παραμέτρους στην οθόνη του. Ακόμη παράγεται ένας ηχητικός συναγερμός εάν εντοπίσει ότι ο αναπνευστήρας δεν λειτουργεί κανονικά. Ο συναγερμός θα συνεχιστεί έως ότου γίνει επαναφορά, πατώντας οποιοδήποτε κουμπί στη συσκευή.
Γενικότερα το σύστημα αισθητήρων και συναγερμών βελτιώνει τη λειτουργικότητα των αναπνευστικών μηχανών. Ο αλγόριθμος αναδρομικής παρακολούθησης φακέλου επιτρέπει στο σύστημα να παρακολουθεί την αναπνοή, να εκτιμά τις μετρήσεις και να εντοπίζει δυσλειτουργίες με μόνο λίγους υπολογισμούς ανά δείγμα και μικρή κατανάλωση στην μνήμη, με αποτέλεσμα να μπορεί να κατασκευαστεί με ένα μικροελεγκτή χαμηλού κόστους και με μερικά άλλα ηλεκτρονικά εξαρτήματα.

Ωστόσο, όπως σε κάθε ηλεκτρονικό κύκλωμα ενδέχεται να υπάρξουν μικρές επιπλοκές, όπως για παράδειγμα, το σύστημα παρόλο που παρακολουθεί την πίεση σε πραγματικό χρόνο για να προκαλέσει συναγερμούς, οι εμφανιζόμενες μετρήσεις υπολογίζονται κατά μέσο όρο σε αρκετούς κύκλους αναπνοής και μπορεί να χρειαστούν έως και 30 δευτερόλεπτα για να αντικατοπτρίζονται οι μεγάλες αλλαγές στις ρυθμίσεις του αναπνευστήρα. Οι μετρήσεις μπορεί επίσης να είναι ανακριβείς όταν η αναπνοή είναι ακανόνιστη, λίγο μετά την αλλαγή των ρυθμίσεων συναγερμού, ή κατά τη διάρκεια και λίγο μετά από μια κατάσταση συναγερμού. Ακόμη, ο συναγερμός μπορεί μερικές φορές να ενεργοποιηθεί λανθασμένα όταν έχουν αλλάξει οι ρυθμίσεις πίεσης του αναπνευστήρα. Αυτοί οι ψευδείς συναγερμοί εμφανίζονται συνήθως εντός 30 δευτερολέπτων από τις ρυθμίσεις. Εάν εμφανίζονται λανθασμένοι συναγερμοί συχνά, συνίσταται να αυξηθεί ο χρόνος συναγερμού μη ανατροφοδότησης.

6. ΕΓΧΕΙΡΙΔΙΟ ΧΡΗΣΗΣ RAPIDVENT

6.1.1 ΕΓΚΑΤΑΣΤΑΣΗ



Εικόνα 12 Συνδεσμολογία αναπνευστήρα

1. Συνδέστε έναν αναπνευστικό προσαρμογέα με ένα στέλεχος στο κύκλωμα αναπνευστήρα στην πλευρά του ασθενούς. Εάν ο αναπνευστήρας είναι εφοδιασμένος με μια θύρα μανόμετρου, αυτή η θύρα μπορεί να χρησιμοποιηθεί αντ 'αυτού.

2. Χρησιμοποιήστε ένα σωλήνα εσωτερικής διαμέτρου 1/8 "(3,2 mm) για να συνδέσετε τη θύρα του αισθητήρα πίεσης στο RapidAlarm στη θύρα του αναπνευστήρα.

3. Συνδέστε το RapidAlarm σε τροφοδοσία 5 volt.

**Οι ρυθμίσεις του συναγερμού επαναφέρονται στις προεπιλογές κάθε φορά που διακόπτεται η τροφοδοσία.*

6.1.2 ΕΝΔΕΙΞΕΙΣ ΟΘΟΝΗΣ

Κωδικός	Δείκτης	Εύρος	Ανάλυση της τιμής
PI	PIP (Υψηλή πίεση)	0-99 cm H2O	1 cm H2O
PE	PEEP (Χαμηλή πίεση)	0-99 cm H2O	1 cm H2O
rr	Ρυθμός αναπνοής	0-99 αναπνοές / λεπτό	1 ανάσα / λεπτό

PIP (PI) : η μέγιστη πίεση εισπνοής σε cm H2O, μετρούμενη ως η μέγιστη πίεση σε έναν κύκλο αναπνοής.

PEEP (PE) : η θετική τελική εκπνευστική πίεση σε cm H_2O, μετρούμενη ως η ελάχιστη πίεση στον κύκλο αναπνοής. Στην υποβοηθούμενη αναπνοή, η ελάχιστη πίεση μπορεί να μην ισούται με τη ρύθμιση PEEP του αναπνευστήρα.

Ρυθμός αναπνοής (rr) : ο αριθμός των πλήρων κύκλων αναπνοής ανά λεπτό, υπολογιζόμενος από το χρόνο μεταξύ των τελευταίων αρκετών αναπνοών.

6.1.3 ΛΕΙΤΟΥΡΓΙΕΣ ΟΘΟΝΗΣ

Το RapidAlarm διαθέτει 3 κουμπιά για την αλλαγή τρόπων και την προσαρμογή των ρυθμίσεων συναγερμού.

- **Λειτουργία** – μετακινείται μέσω τρόπων εμφάνισης ή ρύθμισης
- **Πάνω** – προσαρμόστε τις ρυθμίσεις συναγερμού στις διάφορες λειτουργίες ρυθμίσεων
- **Κάτω** – προσαρμόστε τις ρυθμίσεις συναγερμού στις διάφορες λειτουργίες ρυθμίσεων

Το RapidAlarm έχει τρεις τρόπους λειτουργίας:

- **ΛΕΙΤΟΥΡΓΙΑ ΠΡΟΒΟΛΗΣ**
  - Περιστρέφεται μέσω εμφάνισης PIP, PEEP και αναπνευστικού ρυθμού κάθε 2,5 δευτερόλεπτα
  - Προεπιλεγμένη λειτουργία όταν είναι ενεργοποιημένη.
  - Η οθόνη επιστρέφει αυτόματα στο DISPLAY MODE από SET MODE μετά από 30 δευτερόλεπτα χωρίς είσοδο.
- **ΛΕΙΤΟΥΡΓΙΑ ΣΥΝΑΓΕΡΜΟΥ**
  - Ο συναγερμός ακούγεται συνεχώς.
  - Η οθόνη αναβοσβήνει τον κωδικό συναγερμού.
  - Πατήστε οποιοδήποτε κουμπί για σίγαση του ξυπνητηριού και επιστροφή στο DISPLAY MODE.
- **ΡΥΘΜΙΣΗ ΛΕΙΤΟΥΡΓΙΑΣ**
  - Αλλάξτε τα όρια συναγερμού.
  - Η οθόνη εμφανίζει τον κωδικό συναγερμού και τη ρύθμιση κατωφλίου που αναβοσβήνει.
  - Πατήστε το κουμπί Λειτουργία για να μετακινηθείτε στις ρυθμίσεις:
    1. Χωρίς ανατροφοδότηση (nc) χρόνος συναγερμού (δευτερόλεπτα)
    2. LpΚατώφλι χαμηλής πίεσης ( ) (cm H2O)
    3. HpΌριο υψηλής πίεσης ( ) (cm H2O)
    4. Χαμηλός αναπνευστικός ρυθμός (Lr) κατώφλι (αναπνοές / λεπτό)
    5. Υψηλός αναπνευστικός ρυθμός (Hr) κατώφλι (αναπνοές / λεπτό)
  - Πατήστε το κουμπί Up για να αυξήσετε το όριο
  - Πατήστε το κουμπί Down για να μειώσετε το όριο
  - Όταν τελειώσετε, περιμένετε 30 δευτερόλεπτα ή πατήστε επανειλημμένα το κουμπί Λειτουργία έως ότου η συσκευή επιστρέψει στο DISPLAY MODE.

6.1.4 ΣΥΝΘΗΚΕΣ ΣΥΝΑΓΕΡΜΟΥ

Κωδικός	Κατάσταση	Προεπιλεγμένη ρύθμιση	Ρυθμιζόμενο εύρος	Διάστημα προσαρμογής
nc	Μη ανατροφοδότηση	10 δευτερόλεπτα	5-30 δευτ	5 δευτερόλεπτα
LP	Χαμηλή πίεση	2 cm H2O	1-20 cm H2O	1 cm H2O
HP	Υψηλή πίεση	40 cm H2O	30-90 cm H2O	5 cm H2O
Lr	Χαμηλός αναπνευστικός ρυθμός	6 αναπνοές / λεπτό	5-15 αναπνοές / λεπτό	1 ανάσα / λεπτό
Hr	Υψηλός αναπνευστικός ρυθμός	30 αναπνοές / λεπτό	15-60 αναπνοές / λεπτό	5 αναπνοές / λεπτό

Non-cycling (nc) : Ενεργοποιεί εάν η πίεση δεν έχει αλλάξει σε περισσότερο από τον καθορισμένο αριθμό δευτερολέπτων. Η μη ανατροφοδότηση μπορεί να υποδηλώνει αποσύνδεση, απόφραξη ή άπνοια.

Χαμηλή πίεση (LP) : Ενεργοποιείται αμέσως εάν η πίεση πέσει κάτω από το καθορισμένο κατώφλι. Οι αναπνευστήρες με κύκλο πίεσης έχουν σχεδιαστεί για να διατηρούν θετική πίεση, οπότε μια πτώση στην ατμοσφαιρική πίεση θα μπορούσε να υποδηλώνει αποσύνδεση.

Υψηλή πίεση (HP) : Ενεργοποιείται αμέσως εάν η πίεση υπερβεί το καθορισμένο κατώφλι. Οι αναπνευστήρες με κύκλο πίεσης δεν πρέπει ποτέ να υπερβαίνουν την πίεση που έχει ρυθμιστεί από τη βαλβίδα PIP, επομένως η ένδειξη υψηλής πίεσης μπορεί να υποδηλώνει απόφραξη.

Χαμηλός αναπνευστικός ρυθμός (Lr) : Ενεργοποιεί εάν ο μέσος ρυθμός αναπνοής είναι πολύ χαμηλός. Αυτό θα μπορούσε να δείξει ότι ο αναπνευστήρας δεν έχει ρυθμιστεί σωστά.

Υψηλός αναπνευστικός ρυθμός (Hr) : Ενεργοποιεί εάν ο μέσος ρυθμός αναπνοής είναι πολύ γρήγορος. Αυτό μπορεί να συμβεί εάν ο αναπνευστήρας δεν έχει ρυθμιστεί σωστά ή εάν ο παλιρροιακός όγκος είναι πολύ χαμηλός.

7. ΠΑΡΑΡΤΗΜΑ

Επισυνάπτονται οι υποροutίνες που χρησιμοποιήθηκαν στο κυρίως πρόγραμμα.

1. Συναγερμός (alarm.h)

```
// time of last debounced button press
unsigned long last_debounce_time = 0;

// state for pending button action. reset when handled
bool mode_pressed = false;
bool up_pressed = false;
bool down_pressed = false;

// time of last ADC sample
unsigned long last_sample = millis();
// adc pressure reading
float p;

// counter for cycling through displayed metrics
//and flashing display
uint16_t counter = 0;
// 8 second display cycle
const uint16_t counter_max = 8 * SAMPLE_RATE;

// user interface mode, cycled by mode button
typedef enum {
    // cycle through displayed metrics
    MODE_DISPLAY,
    // set alarm conditions
    MODE_SET_THRESH_NC,
    MODE_SET_THRESH_LP,
    MODE_SET_THRESH_HP,
    MODE_SET_THRESH_LR,
    MODE_SET_THRESH_HR,
    // display raised alarm
    MODE_ALARM
} mode_t;
// button cycles through first 6 modes only
const uint8_t total_modes = 6;
// current mode
uint8_t mode = MODE_DISPLAY;

// convert ADC reading to cm H20 to ADC reading
#define ADC2CM(x) (((x / 1023) - 0.04) / 0.09) * 10.197

// helper function to constrain float to limits
float limit(float x, float lo, float hi) {
    if (x > hi) {
        return hi;
    }
    else if (x < lo) {
        return lo;
    }
    else {
        return x;
    }
}
```

```
// debounce all buttons
bool debounce() {
    if (millis() - last_debounce_time > debounce_time) {
        last_debounce_time = millis();
        return true;
    }
    else {
        return false;
    }
}

// pin interrupt setup
void pci_setup(byte pin)
{

// pin interrupt setup
void pci_setup(byte pin)
{
    *digitalPinToPCMSK(pin) |= bit (digitalPinToPCMSKbit(pin));
    PCIFR  |= bit (digitalPinToPCICRbit(pin));
    PCICR  |= bit (digitalPinToPCICRbit(pin));
}
```

2. Οθόνη (display.h)

```
char displayedValue[] = "--.--";

byte display_index = 0;
byte string_index = 0;
unsigned long nextDisplayUpdate = 0;

// map character to 7 segment
const byte digit_map[] = {
  0b11111100, //0
  0b01100000, //1
  0b11011010, //2
  0b11110010, //3
  0b01100110, //4
  0b10110110, //5
  0b10111110, //6
  0b11100000, //7
  0b11111110, //8
  0b11100110, //9
  0b00000000, //.
  0b00000010, // -
  0b00000000, // space
  0b11001110, // P
  0b10011110, // E
  0b00001100, // I
  0b00001010, // r
  0b00000000, // .
  0b00011100, // L
  0b01101110, // H
  0b11000000, // :
  0b00011010, // c
  0b00101010, // n
};

// cycle through display digits
void update_display() {

  unsigned long now = millis();
  if(now < nextDisplayUpdate) {
    return;

    // FIXME overflow
    nextDisplayUpdate = 4 + now;
    digitalWrite(digit_pins[display_index], LOW);
    display_index++;
    string_index++;
    if(display_index >= sizeof(digit_pins)) {
      display_index = string_index = 0;
    }
    byte dp = (displayedValue[string_index+1] == '.') & 1;
    byte segmentMapIndex;
    switch(displayedValue[string_index]) {
      case ' ':     segmentMapIndex = 12;
                    break;
      case '-':     segmentMapIndex = 11;
                    break;
      case 'P':     segmentMapIndex = 13;
                    break;
      case 'E':     segmentMapIndex = 14;
                    break;
      case 'I':     segmentMapIndex = 15;
                    break;
      case 'r':     segmentMapIndex = 16;
                    break;
      case '.':     segmentMapIndex = 17;
                    break;
      case 'L':     segmentMapIndex = 18;
          break;
      case 'H':     segmentMapIndex = 19;
          break;
      case ':':     segmentMapIndex = 20;
          break;
      case 'c':     segmentMapIndex = 21;
          break;
      case 'n':     segmentMapIndex = 22;
          break;
      default:      segmentMapIndex = displayedValue[string_index] - '0';
    }

    for(byte n = 0; n < sizeof(segment_pins); n++) {
      digitalWrite(segment_pins[n], !((digit_map[segmentMapIndex] | dp) & (128 >> n) ) );
    }
    digitalWrite(digit_pins[display_index], HIGH);
    if(dp) string_index++;
  }
```

3. Αλγόριθμος (algorithm.h)

```
#ifdef __cplusplus
extern "C" {
#endif

#include <stdint.h>
#include <stdbool.h>

// ----- Types -----

// alarm types
typedef enum {
    // no alarm
    ALARM_NONE,
    // ventilator not cycling
    ALARM_NC,
    // PEEP below threshold
    ALARM_LP,
    // PIP above threshold
    ALARM_HP,
    // respiratory rate below threshold
    ALARM_LR,
    // respiratory rate above threshold
    ALARM_HR
} alarm_t;

// ----- Output -----

// whether there is an alarm condition
extern alarm_t alarm_raised;
// display code for raised alarm
extern char *alarm_code;
// disable alarm for N seconds when system turned on
extern bool alarm_disabled;
// value that caused alarm to be raised
extern float alarm_value;
// breath respiration_rate (breaths/min)
extern float respiration_rate;
// peak inspiratory pressure (cm H20)
extern float pip;
// positive end expiratory pressure (cm H20)
extern float peep;

// ----- Config -----

// alarm condition thresholds
extern uint16_t THRESH_NC; // s
extern uint16_t THRESH_LP; // cm H20
extern uint16_t THRESH_HP; // cm H20
extern uint16_t THRESH_LR; // breaths/min
extern uint16_t THRESH_HR; // breaths/min

// ----- Functions -----

// compute samplerate dependent weights
void init_algorithm(uint8_t sample_rate);
// take in single sample at SAMPLE_RATE and update algorithm state
void run_algorithm(float p);

#ifdef __cplusplus
} // extern "C"
#endif
```

Οι κώδικες που χρησιμοποιήθηκαν στην python είναι οι εξής:

1. Μετατροπή αρχείου μετρήσεων σε csv

```
import io
import csv
import argparse

from ventmap.breath_meta import get_production_breath_meta
from ventmap.raw_utils import extract_raw
from ventmap.constants import META_HEADER

parser = argparse.ArgumentParser(description="Convert ventMAP csv to rapidalarm csv")
parser.add_argument('infile', type=str, help="path to input ventMAP csv")
parser.add_argument('outfile', type=str, help="path to output csv")
parser.add_argument('-r', type=str, default=50, help="input file samplerate")
args = parser.parse_args()

generator = extract_raw(io.open(args.infile), False)

# pressure, flow, PIP, PEEP, RR waveforms
pressure = []
flow = []
pip = []
peep = []
rr = []

# read each breath waveform and precomputed scalar metrics
for breath in generator:
    # load single breath
    prod_breath_meta = get_production_breath_meta(breath)

    # get breath pressure and ground truth PIP, PEEP, RR
    pressure += breath['pressure']
    flow += breath['flow']

    # expand scalar PIP/PEEP/RR metric to same length as breath
    breath_samples = len(breath['pressure'])
    pip += breath_samples * [prod_breath_meta[META_HEADER.index('PIP')]]
    peep += breath_samples * [prod_breath_meta[META_HEADER.index('PEEP')]]
    rr += breath_samples * [prod_breath_meta[META_HEADER.index('inst_RR')]]

with open(args.outfile, 'w') as f:
    # write samplerate as metadata header
    f.write("#samplerate,{}\n".format(args.r))
    # write column header
    f.write("pressure,flow,true_pip,true_peep,true_rr\n")

    for x in zip(pressure, flow, pip, peep, rr):
        f.write(",".join(map(str, x)) + "\n")
```

```
from cffi import FFI
from pathlib import Path

ffi = FFI()

code_dir = Path(__file__).parent.parent
source_file = code_dir.joinpath('firmware/algorithm.c').absolute()
include_dir = code_dir.joinpath('firmware/').absolute()
cdef = code_dir.joinpath('rapidalarm/algorithm.cdef').absolute()

# read in variable datatypes
ffi.cdef(open(cdef, 'r').read())

# build shared object
ffi.set_source(
    "rapidalarm.algorithm",
    open(source_file, 'r').read(),
    include_dirs=[include_dir],
)

def build():
    ffi.compile(
        verbose=True,
        tmpdir=code_dir
    )


if __name__ == '__main__':
    build()
```

3. Δημιουργία κυματομορφών (1/4)

```
from cffi import FFI
import argparse
import logging
import numpy as np
import pandas as pd
import sys
from rapidalarm.build_algorithm import build


def load_dataset(datafile):
    """Read a rapidalarm dataset into a Pandas dataframe

    Args:
        datafile (str): path to datafile

    Returns:
        DataFrame: has columns 't', 'pip', 'peep', 'pressure', 'flow'
            and attribute 'sample_rate'
    """

    f = open(datafile, 'r')
    # read sample_rate from first line
    sample_rate = int(f.readline().split(',')[1])

    x = pd.read_csv(f)
    x.sample_rate = sample_rate
    x['t'] = np.linspace(0, len(x) / sample_rate, len(x))

    return x


def load_pandas(dataset, rebuild=False, end=None):
    """Run algorithm with recorded data and return pandas DataFrame with result

    Args:
        data (str): path to dataset

    Returns:
        pandas.DataFrame: dataframe containing alg. input and results
    """
    df = load_dataset(dataset)
```

Δημιουργία κυματομορφών (2/4)

```
    if rebuild:
        build()

    from rapidalarm.algorithm import lib

    lib.init_algorithm(df.sample_rate)

    pip, peep, rr, v_high, v_low, alarm_raised = [], [], [], [], [], []
    debug = []

    # build dict from enums
    alarms = [
        'ALARM_HP', 'ALARM_HR', 'ALARM_LP', 'ALARM_LR', 'ALARM_NC', 'ALARM_NONE'
    ]
    alarms = {getattr(lib, s):s for s in alarms}

    for p in df.pressure[:end]:
        lib.run_algorithm(p)

        pip.append(lib.pip)
        peep.append(lib.peep)
        rr.append(lib.respiration_rate)
        v_high.append(lib.v_high)
        v_low.append(lib.v_low)
        alarm_raised.append(alarms[lib.alarm_raised])
        # alarm_raised.append(lib.alarm_raised)
        debug.append(lib.t_peak)

    df['pip'] = pip
    df['peep'] = peep
    df['rr'] = rr
    df['v_high'] = v_high
    df['v_low'] = v_low

    df['alarm_raised'] = alarm_raised
    df['debug'] = debug

    return df


def csv(args):

    df = load_pandas(args.dataset, args.rebuild)

    df.to_csv(sys.stdout, index=False)
```

Δημιουργία κυματομορφών (3/4)

```
def plot(args):
    """Run prerecorded waveform through algorithm.  Plot time range."""
    import matplotlib.pyplot as plt

    df = load_pandas(args.dataset, args.rebuild)

    fig, [plt1, plt2, plt3, plt4] = plt.subplots(4, 1, sharex=True, figsize=(15, 6))

    plt1.plot(df.t[args.s:args.e], df.pressure[args.s:args.e], 'k', linewidth=0.5)
    plt1.plot(df.t[args.s:args.e], df.v_high[args.s:args.e], 'r', linewidth=0.5)
    plt1.plot(df.t[args.s:args.e], df.v_low[args.s:args.e], 'b', linewidth=0.5)
    plt1.legend(
        ['Pressure', 'High Envelope', 'Low Envelope'],
        loc='center left',
        bbox_to_anchor=[1, .5]
    )
    plt1.grid(True)

    plt2.plot(df.t[args.s:args.e], df.pressure[args.s:args.e], 'k', linewidth=0.5)
    plt2.plot(df.t[args.s:args.e], df.true_pip[args.s:args.e], 'r', linewidth=0.5)
    plt2.plot(df.t[args.s:args.e], df.true_peep[args.s:args.e], 'b', linewidth=0.5)
    plt2.plot(df.t[args.s:args.e], df.pip[args.s:args.e], 'r')
    plt2.plot(df.t[args.s:args.e], df.peep[args.s:args.e], 'b')
    plt2.legend(
        ['Pressure', 'True PIP', 'True PEEP', 'PIP', 'PEEP'],
        loc='center left',
        bbox_to_anchor=[1, .5]
    )
    plt2.grid(True)

    plt3.plot(df.t[args.s:args.e], df.rr[args.s:args.e], 'b')
    plt3.plot(df.t[args.s:args.e], df.true_rr[args.s:args.e], 'b', linewidth=0.5)
    # plt3.plot(df.t[args.s:args.e], df.debug[args.s:args.e])
    plt3.set_ylabel('breaths/min')
    plt3.legend(
        ['RR', 'True RR'],
        loc='center left',
        bbox_to_anchor=[1, .5]
    )
    plt3.grid(True)

    plt4.plot(df.t[args.s:args.e], df.alarm_raised[args.s:args.e])
    plt4.grid(True)
```

Δημιουργία κυματομορφών (4/4)

```
    plt.xlabel('time (s)')
    plt.tight_layout()
    plt.show()

    return plt1, plt2, plt3


def stat(args):
    """Show algorithm accuracy statistics"""

    df = load_pandas(args.dataset, args.rebuild)[args.s:args.e]

    rr_err = df.rr - df.true_rr
    pip_err = df.pip - df.true_pip
    peep_err = df.peep - df.true_peep

    print('RR\n-----')
    print(f'err mean:{rr_err.mean():.2f}')
    print(f'err std:{rr_err.std():.2f}')
    print('\nPIP\n-----')
    print(f'err mean:{pip_err.mean():.2f}')
    print(f'err std:{pip_err.std():.2f}')
    print('\nPEEP\n-----')
    print(f'err mean:{peep_err.mean():.2f}')
    print(f'err std:{peep_err.std():.2f}')
def main():

    parser = argparse.ArgumentParser(description="Algorithm Tester")

    parser.add_argument('--rebuild', action='store_true', default=False, help="rebuild algorithm.c")
    parser.add_argument('--debug', action='store_true', default=False, help="enable debugging")

    subparsers = parser.add_subparsers()
    subparsers.required = True

    csv_parser = subparsers.add_parser('csv', help="generate csv output")
    # csv_parser.add_argument('start', type=int, help="simulation start sample #")
    # csv_parser.add_argument('end', type=int, help="simulation end sample #")
    csv_parser.set_defaults(func=csv)

    plot_parser = subparsers.add_parser('plot', help="plot input waveform along with estimated parameters")
    plot_parser.add_argument('dataset', type=str, help="path to dataset")
    plot_parser.add_argument('-s', metavar='START', type=int, default=0, help="simulation start sample num")
    plot_parser.add_argument('-e', metavar='END', type=int, default=None, help="simulation end sample num")
    plot_parser.set_defaults(func=plot)

    stat_parser = subparsers.add_parser('stat', help="algorithm accuracy statistics for dataset")
    stat_parser.add_argument('dataset', type=str, help="path to dataset")
    stat_parser.add_argument('-s', metavar='START', type=int, default=0, help="simulation start sample num")
    stat_parser.add_argument('-e', metavar='END', type=int, default=None, help="simulation end sample num")
    stat_parser.set_defaults(func=stat)

    args = parser.parse_args()

    if args.debug:
        log.setLevel(logging.DEBUG)
        log.debug("Debugging enabled")

    args.func(args)


if __name__ == '__main__':
    main()
```