ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΧΕΔΙΑΣΗ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΙΑΤΡΙΚΗΣ ΣΥΣΚΕΥΗΣ ΔΙΑΦΡΑΓΜΑΤΙΚΗΣ ΑΝΑΠΝΟΗΣ

Του φοιτητή
Κόλαση Χρυσοβέργη-Γρηγόριου
Αρ. Μητρώου: 51065

Επιβλέπων
Ιορδάνης Κιοσκερίδης
Καθηγητής

Ιούλιος, 2020

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

ΠΡΟΛΟΓΟΣ

Με τον όρο ιατρική συσκευή εννοούμε οποιαδήποτε συσκευή προορίζεται να χρησιμοποιηθεί για ιατρικούς σκοπούς. Οι ιατρικές συσκευές ωφελούν τους ασθενείς βοηθώντας τους παρόχους υγειονομικής περίθαλψης να διαγνώσουν και να θεραπεύσουν ασθενείς, βελτιώνοντας την ποιότητα ζωής τους. Τα προϊόντα αυτά πρέπει να αποδειχθούν ασφαλή και αποτελεσματικά, ώστε να γίνουν βοηθητικά εργαλεία στον ανθρώπους της υγειονομικής περίθαλψης. Η αναπνοή είναι μια απαραίτητη φυσική λειτουργία του οργανισμού των ανθρώπων. Η δε διαφραγματική αναπνοή έχει πολλά οφέλη στην υγεία και χρησιμοποιείται κυρίως σε ανθρώπους με αναπνευστικά προβλήματα είτε σε άτομα που έχουν βγει από το χειρουργείο , αλλά παράλληλα μπορεί να εφαρμοστεί από όλους με την κατάλληλη εκμάθηση της τεχνικής της. Ο φυσικοθεραπευτής προσφέρει υπηρεσίες υγείας με στόχο την ανάπτυξη, διατήρηση και αποκατάσταση της, διά βίου, μέγιστης κινητικής και λειτουργικής ικανότητας, ιδιαίτερα όπου η κίνηση και η λειτουργικότητα απειλούνται από το γήρας, τραυματισμούς, νόσους ή/και περιβαλλοντικούς παράγοντες. Το ζητούμενο είναι η υλοποίηση ενός ικανοποιητικού τρόπου μέτρησης της διαφραγματικής αναπνοής, ώστε να γίνει ένα βοηθητικό εργαλείο στον φυσικοθεραπευτή. Η υλοποίηση της συσκευής αυτής απαίτησε έρευνα τρόπων μέτρησης της αναπνοής σε ένα γενικότερο πλαίσιο με σκοπό την συμβολή τους στην υλοποίηση κατάλληλου τρόπου μέτρησης για την διαφραγματική αναπνοή. Στην συμβολή αυτής της προσπάθειας χρειάστηκαν εμπλουτισμένες γνώσεις σε πολλούς τομείς της τεχνολογίας.

ΠΕΡΙΛΗΨΗ

Στην παρούσα πτυχιακή εργασία γίνεται η μελέτη, η σχεδίαση και η κατασκευή μια ολοκληρωμένης συσκευής που μετράει την διαφραγματική αναπνοή. Η εργασία αποτελείται από την μελέτη και υλοποίηση ενός τρόπου μέτρησης, την κατασκευή των ηλεκτρονικών κυκλωμάτων για την υλοποίηση των λειτουργιών των επιμέρους εξαρτημάτων που την απαρτίζουν, την ασύρματη μεταφορά των δεδομένων και τον απομακρυσμένο έλεγχο από έναν ηλεκτρονικό υπολογιστή με την βοήθεια ενός router. Η καρδία του συστήματος είναι ο μικροελεγκτής ESP32, όπου με βάση αυτού και το μεγάλο πλήθος των λειτουργιών που προσφέρει σε σχετικά μικρό κόστος επιτέλεσε μεγάλη βοήθεια στον κεντρικό στόχο. Τα κύρια δομικά μέρη της συσκευής που διαπραγματεύεται είναι η οθόνη εφτά ιντσών που επιλέχτηκε για την απαιτούμενη οπτική ανατροφοδότηση στον χρήστη της συσκευής βασιζόμενη στο τσιπ ελεγκτή οθονών RA8875, ο αισθητήρας πίεσης MPX5050GP που επιλέχτηκε να συμβάλει στο τρόπο μέτρησης της διαφραγματικής αναπνοής, η υλοποίηση της web εφαρμογής για την αποτύπωση των δεδομένων μέσω του φυλλομετρητή και η δημιουργία βάσης δεδομένων τύπου MySQL για την ανάγκη αποθήκευσης και ανάκτησης των μετρήσεων για την μεταγενέστερη προβολή και σύγκριση αποτελεσμάτων. Παρατίθενται οι μέθοδοι που επιλέχτηκαν για την επίτευξη όλων αυτών τον στόχων αναλύονται τα συμπεράσματα που προέκυψαν καθώς και τρόποι βελτίωσης της συσκευής για περαιτέρω ανάπτυξη.

ABSTRACT

In this thesis the study, design and construction of a complete device that measures diaphragmatic breathing are analyzed. The task consists of studying and implementing a measurement method, constructing electronic circuits to implement the functions of the individual components that make it up, wireless data transfer, and remote control from a computer using a router. The heart of the system is the microcontroller ESP32, where based on this and the large number of functions it offers at a relatively low cost it has been a great help to the central target. The main components of the device being traded are the seven-inch screen selected for the required visual feedback to the user of the device based on the RA8875 screen controller chip, the MPX5050GP pressure sensor selected to analyze the data for the measuring of diaphragmatic breathing, web design application for capturing data through the browser and the creation of a MySQL database for the need to store and retrieve measurements for later viewing and comparison of results. The methods selected to achieve all these objectives are listed, the conclusions drawn and ways to improve the device for further development are analyzed.

ΕΥΡΕΤΗΡΙΟ ΠΕΡΙΟΧΟΜΕΝΩΝ

Πρόλογος 3

Περίληψη....4

Abstract 5

Ευρετήριο Περιεχομένων 6

Ευρετήριο Σχημάτων 7

Εισαγωγή 9

1. **Η ΔΙΑΦΡΑΓΜΑΤΙΚΗ ΑΝΑΠΝΟΗ** 10
    1. Εισαγωγή.... 10
    2. Το διάφραγμα στην αναπνοή 10
    3. Περιγραφή της διαφραγματική αναπνοής.... 11
    4. Μέτρηση διαφραγματικής αναπνοής.... 13
    5. Επιλογή μεθόδου 17
2. **ΜΙΚΡΟΕΛΕΓΚΤΗΣ ESP32** 18
    1. Εισαγωγή.... 18
    2. Το μπλοκ διάγραμμα 19
    3. Αναπτυξιακή πλακέτα.... 21
    4. Προγραμματιστικό περιβάλλον.... 23
3. **ΑΙΣΘΗΤΗΡΙΟ ΠΙΕΣΗΣ** 26
    1. Εισαγωγή.... 26
    2. Φιλτράρισμα.... 27
    3. Είσοδος στον ADC του ESP32.... 29
    4. Αξιολόγηση αποτελεσμάτων 31
4. **ΕΠΟΙΚΟΙΝΩΝΙΑ ΟΘΟΝΗΣ ΚΑΙ ΚΥΡΙΩΣ ΠΡΟΓΡΑΜΑ** 33
    1. Εισαγωγή.... 33
    2. Ελεγκτής οθόνης 34
    3. Γραφικά συσκευής.... 37
    4. Ανάλυση λειτουργίας συσκευής.... 41
5. **WEB ΕΦΑΡΜΟΓΗ** 44
    1. Εισαγωγή.... 44
    2. Διαμόρφωση επικοινωνίας.... 44
    3. Ανάλυση και κατασκευή της Web εφαρμογής και επικοινωνία με τον ESP32 47
    4. Δημιουργία βάσης δεδομένων και επικοινωνία με τις σελίδες.... 55

6. **Η ΟΛΟΚΛΗΡΩΜΕΝΗ ΣΥΣΚΕΥΗ** 62
1. Εισαγωγή 62
2. Διαμόρφωση επικοινωνίας 62
3. Ανάλυση και κατασκευή της Web εφαρμογής και επικοινωνία με τον ESP32 65
Συμπεράσματα 66
Βιβλιογραφία 67
ΠΑΡΑΡΤΗΜΑ 69
1. Κώδικας του ESP32 σε C++ 69
2. Κώδικας Javascript για την σελίδα Home 90
3. Κώδικας Javascript για την σελίδα History 97
4. Κώδικας Javascript για την σελίδα Registration 101
5. Κώδικας PHP για το αρχείο choose_user.php 102
6. Κώδικας PHP για το αρχείο get_data.php 102
7. Κώδικας PHP για το αρχείο get_users.php 103
8. Κώδικας PHP για το αρχείο register_user.php 103
9. Κώδικας PHP για το αρχείο save_data.php 104

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ

Σχήμα 1.1 Εκτέλεση διαφραγματικής αναπνοής σε ξαπλωτή θέση 12
Σχήμα 1.2 Απεικόνιση της συστολής και διαστολής της κοιλίας κατά την αναπνοή και εκπνοή στην διαδικασία της διαφραγματικής αναπνοής 13
Σχήμα 1.3 Απεικόνιση ενός resistive strain gauge 14
Σχήμα 1.4 Μέθοδος μέτρησης αναπνοής με την χρήση τεσσάρων resistive strain gauge τοποθετημένα στο σώμα του ασθενή 15
Σχήμα 1.5 Μέθοδος μέτρησης αναπνοής στην περιοχή της κοιλίας με την χρήση χωρητικού αισθητηρίου 16
Σχήμα 1.6 Εξωτερική εμφάνιση του αισθητήρα πίεσης MPX5050GP 17
Σχήμα 2.1 Μπλοκ διάγραμμα του ESP32 20
Σχήμα 2.2 Η αναπτυξιακή πλακέτα NodeMCU-32S που χρησιμοποιήθηκε στην συσκευή 22
Σχήμα 2.3 Εισαγωγή URL στην καρτέλα του μενού Preferences του Arduino IDE για την προσθήκη του ESP32 core 25
Σχήμα 2.4 Επιλογή Tools – Boards Manager στο Arduino IDE για την κατέβασμα του ESP32 core 25
Σχήμα 2.5 Αναζήτηση ESP32 στο Boards Manager του Arduino IDE 26

Σχήμα 2.6 Επιλογή της πλακέτας NodeMCU-32s για προγραμματισμό μέσω του Arduino IDE 26
Σχήμα 3.1 Σχέση τάσης προς πίεση σε περιβάλλον θερμοκρασίας από 0 ως 85 °C 27
Σχήμα 3.2 Προτεινόμενα στοιχεία φιλτραρίσματος του MPX5050GP από τον κατασκευαστή 28
Σχήμα 3.3 Κύκλωμα ενίσχυσης και χαμηλοπερατού φίλτρου 1ης Τάξης 29
Σχήμα 3.4 Προσαρμογή εύρους του αισθητηρίου MPX5050GP στα 3.3Volt 30
Σχήμα 3.5 Μέτρηση τιμής μέσω του αισθητηρίου MPX505GP στην τιμή αναφοράς 40mmHg σύμφωνα με το μανόμετρο 32
Σχήμα 3.6 Μέτρηση τιμής μέσω του αισθητηρίου MPX505GP στην τιμή αναφοράς 60mmHg σύμφωνα με το μανόμετρο 33
Σχήμα 3.7 Μέτρηση τιμής μέσω του αισθητηρίου MPX505GP στην τιμή αναφοράς 80mmHg σύμφωνα με το μανόμετρο 33
Σχήμα 4.1 Αναλυτική απεικόνιση επικοινωνίας του κεντρικού επεξεργαστή master (ESP32) με τον ελεγκτή slave (RA8875) με σκοπό την εγγραφή εντολής σε έναν καταχωρητή 38
Σχήμα 4.2 Στατικά γραφικά οθόνης (επίπεδο 1) για την μέτρηση της διαφραγματικής αναπνοής σε οθόνη TFT 7 ιντσών με την χρήση του ελεγκτή οδήγησης RA8875 40
Σχήμα 4.3 Η συνολική απεικόνιση των γραφικών της οθόνης για την μέτρηση της διαφραγματικής αναπνοής σε οθόνη TFT 7 ιντσών με την χρήση του ελεγκτή οδήγησης RA8875 41
Σχήμα 5.1 Απεικόνισή του ολοκληρωμένου μοντέλου επικοινωνίας σύμφωνα με την αρχιτεκτονική client-server πάνω στο πρωτόκολλο TCP/IP 46
Σχήμα 5.2 Σελίδα Home (κύρια σελίδα web εφαρμογής) 54
Σχήμα 5.3 Σελίδα Registration (δημιουργίας χρήστη) 57
Σχήμα 5.4 Σελίδα History (ιστορικού χρήστη) 59
Σχήμα 6.1 Σχηματικό διάγραμμα βασικής πλακέτας του συστήματος 63
Σχήμα 6.2 Πλακέτα επικοινωνίας του RA8875 με την οθόνη 7 ιντσών 64
Σχήμα 6.3 Πλακέτα του βασικού κυκλώματος (σχηματικό διάγραμμα στο Σχ. 6.1) 65
Σχήμα 6.4 Πλακέτα για το αισθητήριο πίεσης MPX5050GP (Κύκλωμα ενίσχυσης και χαμηλοπερατού φίλτρου 1ης Τάξης, σχηματικό διάγραμμα στο Σχ. 3.3) 65
Σχήμα 6.5 Πλακέτα φόρτισης-σταθεροποίησης στα 5V και η μπαταρία Samsung INR18650 66

Πίνακας 3.1 Πίνακας 3.1 Δείγματα μετρήσεων του ADC πριν και μετά την βαθμονόμηση 55
Πίνακας 4.1 Οι κύκλοι πρόσβασης καταχωρητή του ελεγκτή RA8875 και τα απαραίτητα bits επιλογής 37
Πίνακας 5.1 Ο MySQL πίνακας users για την αποθήκευση των στοιχείων των χρηστών 55
Πίνακας 5.2 Ο MySQL πίνακας attempts για την αποθήκευση των στοιχείων της μέτρησης 56
Πίνακας 6.1 Οι συνδέσεις της οθόνης με τον ESP32 64

ΕΙΣΑΓΩΓΗ

Η εργασία πλαισιώνεται στην κατασκευή μια συσκευής μέτρησης διαφραγματικής αναπνοής ικανοποιώντας κάποιες βασικές απαιτήσεις που προκύπτουν σύμφωνα με αυτόν τον στόχο. Επιλέχτηκε να αξιοποιηθούν τεχνολογίες, πρωτόκολλα και δίκτυα επικοινωνιών που είναι σε άμεση εφαρμογή στην καθημερινότητα με συνεχόμενη εξέλιξη. Γίνεται εκτενής ανάλυση, χωρισμένη σε κεφάλαια που κάθε ένα με την σειρά του αναλύει διεξοδικά την μέθοδο και τα εξαρτήματα που χρησιμοποιήθηκαν για την υλοποίηση της συσκευής. Στο πρώτο κεφάλαιο γίνεται λόγος στην αναπνευστική φυσικοθεραπεία που είναι η αλληλένδετη επιστήμη με την χρήση αυτής της συσκευής που εξετάζεται, τον ρόλο του διαφράγματος, την ανάλυση της τεχνικής εκτέλεσης της διαφραγματικής αναπνοής, την έρευνα για τρόπους μέτρησης της αναπνοής γενικότερα και την επιλογή της μεθόδου που υλοποιήθηκε στην συσκευή. Στο δεύτερο κεφάλαιο γίνεται αναφορά στον μικροελεγκτή ESP32 που επιλέχτηκε να παίξει τον ρόλο της κεντρικής μονάδας επεξεργασίας στο σύστημα, την αναπτυξιακή πλακέτα που επιλέχτηκε και τον τρόπο προγραμματισμού του. Κάποια από τα βασικά χαρακτηριστικά που ώθησαν στην επιλογή του είναι η δυνατότητα WiFi σύνδεσης στο δίκτυο και η ύπαρξη μεγάλης επεξεργαστικής ισχύς (διπύρηνη αρχιτεκτονική) σε σχέσεις με άλλους μικροελεγκτές στο συγκεκριμένο κόστος. Στο τρίτο κεφάλαιο γίνεται αναφορά στα χαρακτηριστικά του αισθητηρίου πίεσης που επιλέχτηκε και την υλοποίηση του ηλεκτρονικού κυκλώματος για το signal conditioning μέχρι την είσοδο των δεδομένων του στον ADC του ESP32. Στο τέταρτο κεφάλαιο αναλύεται η οθόνη που επιλέχτηκε για να δώσει οπτική ανατροφοδότηση της μέτρησης, με κύρια εμβάθυνση στον τρόπο επικοινωνίας και υλοποίησης των γραφικών μέσω του ελεγκτή RA8875 με το πρωτόκολλο SPI. Το πέμπτο κεφάλαιο ασχολείται με την υλοποίηση της web εφαρμογής και της βάσης δεδομένων που υλοποιήθηκαν. Χρησιμοποιήθηκαν γλώσσες προγραμματισμού όπως η Javascript, η PHP και περιγραφικές γλώσσες όπως η HTML και CSS. Έγινε επίσης χρήση βοηθητικών framework όπως η Jquery και η Bootstrap. Με την χρήση αυτών των εργαλείων έγινε η ανάλυση των λειτουργιών της web εφαρμογής και τον τρόπο μεταφοράς δεδομένων (Websocket, Ajax) καθώς και την δομή τους (JSON). Στο έκτο κεφάλαιο παρουσιάζεται η τελική εμφάνιση της συσκευής μαζί με το σχηματικό διάγραμμα τις συνδέσεις και τα στοιχεία για την φορητότητας της (μπαταρία, φορτιστής). Στο τελευταίο κεφάλαιο γίνονται αναφορά στα συμπεράσματα που επέφερε αυτή η εργασία καθώς και σε κάποιος τρόπους βελτίωσης για μελλοντική αναβάθμιση του συστήματος.

ΚΕΦΑΛΑΙΟ 1

Η ΔΙΑΦΡΑΓΜΑΤΙΚΗ ΑΝΑΠΝΟΗ

1.1 Εισαγωγή

Η αναπνευστική φυσικοθεραπεία αποτελεί μια επιστήμη που ασχολείται με αναπνευστικά προβλήματα εδώ και δεκαετίες και η δράση της θεωρείται ευεργετική όχι μόνο γιατί προλαμβάνει λοιμώξεις αλλά γιατί βοηθά τον ασθενή να βελτιώσει την αναπνευστική του λειτουργία, να μάθει να ζει με το πρόβλημά του χωρίς φόβο και πανικό και φυσικά να έχει μια καλύτερη ποιότητα ζωής. Στα άτομα που μπορεί να επωφεληθούν από την αναπνευστική φυσικοθεραπεία συγκαταλέγονται ασθενείς με χρόνιες αναπνευστικές παθήσεις (πχ. άσθμα, χρόνια αποφρακτική πνευμονοπάθεια, κυστική ίνωση), χρόνια κλινήρεις ασθενείς (πχ. μετά από εγκεφαλικό ή άλλη πάθηση που δεν τους επιτρέπει να σηκωθούν από το κρεβάτι), ασθενείς μετά από οποιοδήποτε χειρουργείο, ιδίως του θώρακα, άτομα με οστεοπόρωση ή σοβαρά προβλήματα κύφωσης και σκολίωσης που παρεμποδίζουν τη σωστή αναπνοή, προκειμένου να διευκολυνθεί η αναπνοή τους και να αποφευχθεί ο κίνδυνος δημιουργίας φλεγμονής και/ή λοίμωξης από τη συγκέντρωση παχύρευστων εκκρίσεων που δεν μπορούν να απομακρυνθούν από τον οργανισμό μέσω του βήχα.

Η εκπαίδευση της σωστής αναπνοής, και δη της διαφραγματικής αναπνοής μέσω της ενδυνάμωσης αυτών των μυών αποτελεί αντικείμενο της αναπνευστικής φυσικοθεραπείας. Πρόκειται για μια τεχνική όπου ο ασθενής διδάσκεται να εκτελεί αναπνοές ήπιες και βαθιές παραγόμενες μόνο από την συστολή του διαφράγματος. Η διδασκαλία της διαφραγματικής αναπνοής μπορεί να γίνει σε πολλές θέσεις, καθώς επίσης και κατά την διάρκεια δραστηριοτήτων, όπως η βάδιση. Στην περίπτωση των δραστηριοτήτων, ο ασθενής διδάσκεται έναν ρυθμό ανάμεσα στις εισπνοές και εκπνοές που πραγματοποιεί.

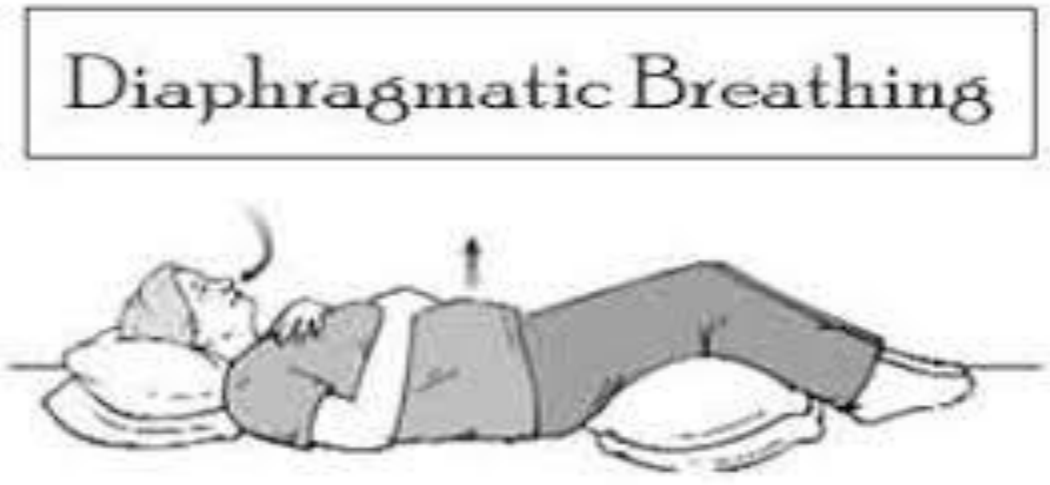
1.2 Το διάφραγμα στην αναπνοή

Το αναπνευστικό σύστημα αναλαμβάνει να διεκπεραιώσει τη διαδικασία διανομής οξυγόνου σε ολόκληρο το σώμα για την διασφάλιση της ζωής και ομαλής λειτουργίας και της περισυλλογής του διοξειδίου του άνθρακα, μεταφοράς του στους πνεύμονες και διοχέτευσής του έξω από το σώμα. Πέρα από τους πνεύμονες που αποτελούν το όργανο της επεξεργασίας του οξυγόνου της ατμόσφαιρας κατά την αναπνοή, ιδιαίτερα σημαντικό ρόλο παίζουν και διάφοροι μυς, με κυριότερο το διάφραγμα. Το διάφραγμα είναι υπεύθυνο για το 80% του συνολικού αναπνευστικού έργου σε κανονική αναπνοή. Βρίσκεται στο κατώτερο μέρος του θώρακα και διαχωρίζει τη θωρακική από την κοιλιακή κοιλότητα του σώματός μας. Υπάρχουν κι άλλοι μικρότεροι μυς που δρουν συνεργικά με το διάφραγμα και συμβάλλουν στην έκπτυξη του θώρακα, ανεβάζοντας τις πλευρές, το στέρνο και την κλείδα. Πιο αναλυτικά, κατά τη διάρκεια της εισπνοής, το διάφραγμα και οι λοιποί μυς συσπώνται και μετακινούνται προς τα κάτω ώστε να αυξηθεί το μέγεθος της θωρακικής κοιλότητας, με αποτέλεσμα τη διεύρυνση των πνευμόνων και πρόσληψη οξυγόνου από αυτούς. Αντίθετα, κατά την εκπνοή, το διάφραγμα και οι λοιποί μυς χαλαρώνουν επιστρέφοντας στην αρχική τους θέση με αποτέλεσμα τη μείωση της χωρητικότητας της θωρακικής κοιλότητας και κατά επέκταση της

χωρητικότητας των πνευμόνων. Έτσι, δημιουργείται μια ελαφριά “πίεση”, που ασκείται στον αέρα «γεμάτο» με διοξείδιο του άνθρακα πλέον και κατά αυτόν τον τρόπο ωθείται προς την έξοδο από τον οργανισμό.

1.3 Περιγραφή της διαφραγματικής αναπνοής

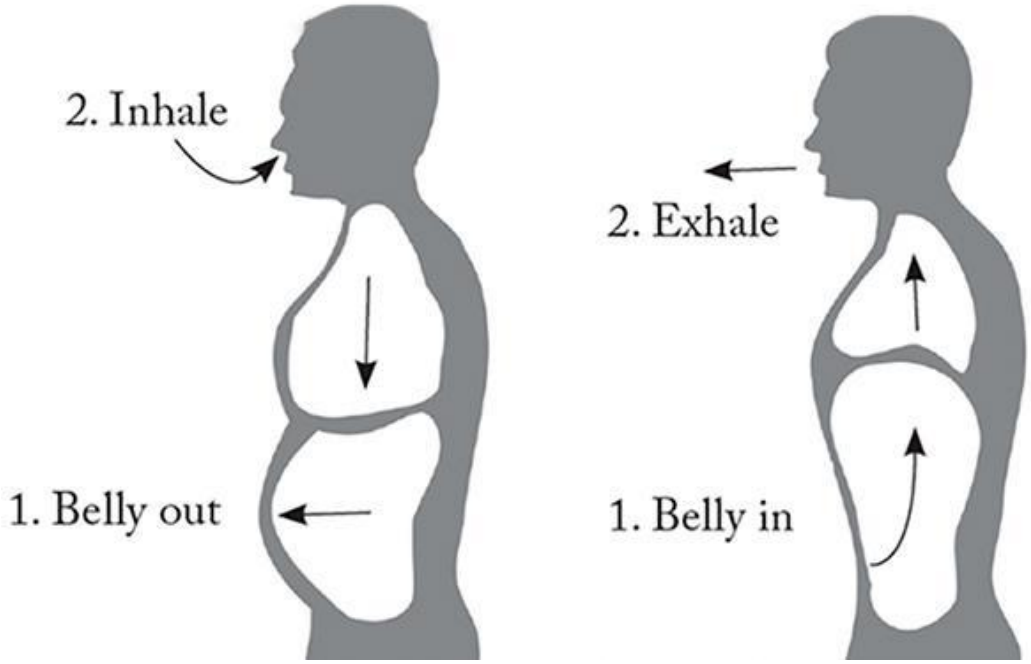
Παρατηρείται μία ασυνέπεια στην ορολογία και στην περιγραφή της διαφραγματικής αναπνοής, η οποία είναι συνώνυμη με τον έλεγχο της αναπνοής, την χαλαρή ελεγχόμενη αναπνοή και την κοιλιακή αναπνοή. Οι ασθενείς ενθαρρύνονται να αναπνέουν κυρίως με το διάφραγμα ελαχιστοποιώντας τη δράση των επικουρικών μυών. Η τεχνική γενικά εκτελείται σε υποστηριζόμενη καθιστή θέση, με χαλαρό τον άνω θωρακικό κλωβό και τους ώμους, παρόλο που αρκετοί ερευνητές έχουν δώσει οδηγίες στους ασθενείς να λάβουν ύπτια θέση. Απτική ανατροφοδότηση δίνεται με το ένα χέρι του θεραπευτή ή του ασθενούς στην κοιλιά, με ή χωρίς το άλλο χέρι στο στήθος. Κατά τη διάρκεια της εισπνοής, οι ασθενείς καλούνται να μετακινήσουν το κοιλιακό τοίχωμα προς τα έξω, προκαλώντας το χέρι που βρίσκεται στην κοιλιά να σηκωθεί προς τα πάνω και έξω ("αναπνεύστε στο χέρι μου / σας"), με ελάχιστη ψηλαφητή κίνηση στο χέρι που βρίσκεται στο στήθος. Η ρινική εισπνοή ενθαρρύνεται με σκοπό να διευκολύνει την συμμετοχή του διαφράγματος καθώς και να ενισχύσει τη φυσική ύγρανση του αέρα. Η εκπνοή είναι χαλαρή και παθητική με το χέρι στην κοιλιά να επιστρέφει ήρεμα στη θέση χαλάρωσης. Η τεχνική κρίνεται αποτελεσματική όταν παρατηρείται μέγιστη κοιλιακή έκπτυξη και μείωση της έκπτυξης του θωρακικού κλωβού ή της κίνησης της άνω θωρακικής μοίρας. Οι παραλλαγές αυτής της τεχνικής περιλαμβάνουν το συνδυασμό της διαφραγματικής αναπνοής στην εισπνοή με την αναπνοή με σφιγμένα χείλη κατά την εκπνοή ή την ενεργητική κοιλιακή εκπνοή. Οι ασθενείς ενθαρρύνονται να εξασκούνται σε αυτή την τεχνική καθημερινά και ο χρόνος κυμαίνεται από 10 λεπτά έως μία ώρα ανά συνεδρία, δύο έως τρεις φορές την ημέρα, αν και η βέλτιστη δοσολογία δεν είναι σαφής (Breslin, 1995).



Σχήμα 1.1 Εκτέλεση διαφραγματικής αναπνοής σε ξαπλωτή θέση

Ο στόχος των αναπνευστικών ασκήσεων διαφραγματικής αναπνοής είναι να διδάξουν στους ασθενείς πώς να ανακουφίσουν και να ελέγξουν τη δύσπνοια μέσω της διόρθωσης των διαταραχών του αναπνευστικού προτύπου, μειώνοντας έτσι το μεταβολικό κόστος της αναπνοής και βελτιώνοντας τη διανομή του αερισμού. Επακόλουθες βελτιώσεις παρατηρούνται στην ανταλλαγή αεριών, στην απόδοση της άσκησης και στα συμπτώματα. Οι ασθενείς είναι σε θέση να μεταβάλλουν εκούσια (προσωρινά) το πρότυπο της αναπνοής τους σε βραδύτερες, βαθύτερες εισπνοές με μεγαλύτερη κοιλιακή και μικρότερη θωρακική έκπτυξη.

Σε μία συστηματική ανασκόπηση των ασκήσεων ελέγχου της αναπνοής, η διαφραγματική αναπνοή αποδείχθηκε ότι έχει στατιστικώς σημαντική ευεργετική επίδραση στην έκπτυξη της κοιλίας και του διαφράγματος, στον αναπνευστικό ρυθμό, στον αναπνεόμενο όγκο, στον κορεσμό του αρτηριακού οξυγόνου και στη διαδερμική μέτρηση του οξυγόνου (Lewis, 2007). Οι περισσότερες από αυτές τις μελέτες περιλάμβαναν ασθενείς με ΧΑΠ και τα ευρήματα ήταν συνεπή όταν αφαιρέθηκαν μελέτες άλλων πληθυσμών. Ωστόσο, η διαφραγματική αναπνοή αποδείχθηκε ότι έχει στατιστικώς σημαντική επιβλαβή επίδραση στο έργο της αναπνοής και στη δύσπνοια σε άτομα με σοβαρή ΧΑΠ (FEV1% 30-50% της προβλεπόμενης τιμής). Δεν υπήρξε επίδραση στις φυσιολογικές εκβάσεις, που σχετίζονται με το ενεργειακό κόστος της αναπνοής, όπως η πρόσληψη οξυγόνου, η αποτελεσματικότητα των αναπνευστικών μυών, ή τα αποτελέσματα που σχετίζονται με την ανταλλαγή αερίων, όπως η κατανομή του αερισμού (Lewis, 2007). Η διαφραγματική αναπνοή έχει συσχετιστεί με την αύξηση της ασύγχρονης και παράδοξης κίνησης του θωρακικού κλωβού σε αρκετές μελέτες (Cahalin, 2002; Fernandez, 2011). Αυτό μπορεί να οφείλεται στην επιβλαβή επίδραση στο έργο της αναπνοής και στη δύσπνοια, παρότι υπήρξαν λίγες μελέτες που διερευνούν την επίδραση της διαφραγματικής αναπνοής στη δύσπνοια ως πρωτεύον καταληκτικό σημείο.



Σχήμα 1.2 Απεικόνιση της συστολής και διαστολής της κοιλίας κατά την αναπνοή και εκπνοή στην διαδικασία της διαφραγματικής αναπνοής

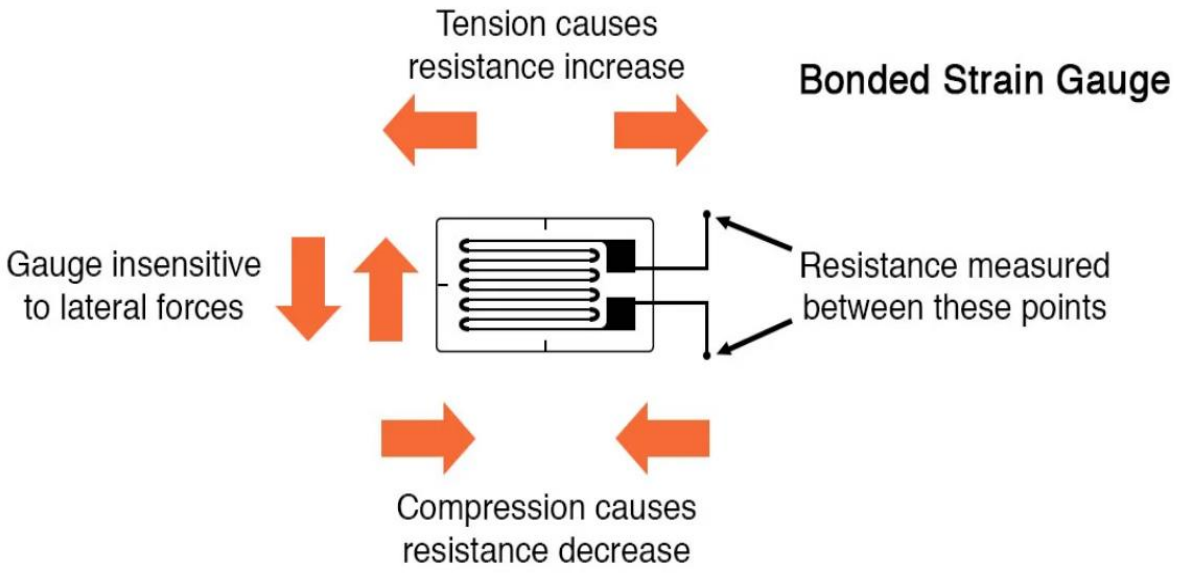
1.4 Μέτρηση διαφραγματικής αναπνοής

Η εκπαίδευση της σωστής αναπνοής απαιτεί αρκετούς μήνες εξάσκησης, γι' αυτό και είναι σημαντικό να εκτιμάται η εξέλιξη του ασθενούς με μετρήσιμο τρόπο. Η λειτουργία της αναπνοής συνεπάγεται με την διαστολή της θωρακικής κοιλότητας και πιο συγκεκριμένα η διαφραγματική αναπνοή με την διαστολή της κοιλιακής χώρας, οπότε η μέτρηση θα πρέπει να γίνει με κάποια μορφή αισθητηρίου που μπορεί να αντιλαμβάνεται τέτοιου είδους αυξομείωση. Αυτό το αισθητήριο μπορεί να τοποθετηθεί στον ενδιάμεσο μια βοηθητικής ζώνης που προσαρμόζεται πάνω στην μετρούμενη περιοχή είτε απευθείας πάνω στην περιοχή η οποία είναι πάνω στο διάφραγμα , στην κοιλιακή χώρα ή στο συνδυασμό και των δύο.

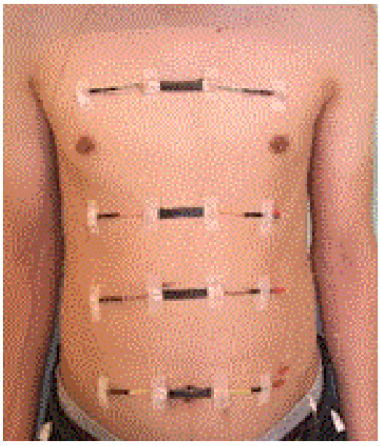
Κάποιες από τις μεθόδους που χρησιμοποιούνται για να επιτύχουν αυτό το αποτέλεσμα είναι οι εξής.

- **Μέθοδος με την χρήση resisistive strain gauge**

Ένα strain gauge εκμεταλλεύεται την φυσική ιδιότητα της ηλεκτρικής αγωγιμότητας και την εξάρτησή της από τη γεωμετρία του αγωγού. Όταν ένας ηλεκτρικός αγωγός τεντώνεται μέσα στα όρια της ελαστικότητάς του έτσι ώστε να μην σπάει ή να παραμορφώνεται μόνιμα, θα γίνει στενότερος και μακρύτερος, γεγονός που αυξάνει την ηλεκτρική αντίστασή του από άκρο σε άκρο. Αντίθετα, όταν ένας αγωγός είναι συμπιεσμένος έτσι ώστε να μην λυγίζει, θα διευρυνθεί και θα μειωθεί, γεγονός που μειώνει την ηλεκτρική αντίσταση του από άκρο σε άκρο. Από τη μετρούμενη ηλεκτρική αντίσταση που παράγεται μέσω της παραμόρφωσης που δέχεται το strain gauge , μπορεί να εξαχθεί η ποσότητα της επαγόμενης τάσης.



Σχήμα 1.3 Απεικόνιση ενός resistive strain gauge



Σχήμα 1.4 : Μέθοδος μέτρησης αναπνοής με την χρήση τεσσάρων resistive strain gauge τοποθετημένα στο σώμα του ασθενή

Σύμφωνα με μια ερεύνα που έγινε σε ένα κέντρο φυσικοθεραπείας στην Κίνα , κατασκεύασαν μια φορητή συσκευή αισθητήρα καταπόνησης όπου χρησιμοποιήθηκε για την ανίχνευση των ινήσεων σύμφωνα με το πρότυπο αναπνοής στο θώρακα και στην κοιλιά. Αποτελείται από τέσσερις αισθητήρες strain gauge μήκους 3,5 εκατοστών.

Ένα ρεύμα χαμηλής έντασης εφαρμόζεται σε κάθε ένα ξεχωριστό strain gauge ώστε σε κάθε παραλλαγή της αντίστασης του να δημιουργείται μια ανάλογη επαγόμενη τάση. Καθώς το άτομο εισπνέει, αντίσταση το κύκλωμα αυξάνεται επειδή η περιοχή διατομής του αγωγού μειώνεται καθώς ο θώρακας διευρύνεται. Τυπικά, η έξοδος του μετρητή τάσης συνδέεται σε μια γέφυρα Wheatstone για να παράγει ένα σήμα εξόδου που είναι άμεσα ανάλογο με τις παραλλαγές αντίστασης που συνοδεύουν κάθε εισπνοή και εκπνοή.

Έχουν τοποθετηθεί σε τέσσερα σημεία πάνω στον σώμα, τα οποία βρίσκονται πάνω στον θώρακα στο ύψος της μασχάλης , στην ξιφοειδής απόφυση που είναι το κατώτερο σημείου του στέρνου , στο ύψος του δέκατου θωρακικού σπόνδυλου και στον ομφαλό με χρήση ιατρικής κόλλας για την βοήθεια της στήριξης τους. Τα σήματα τάσης που παράγονται από την έξοδο των αισθητήρων αποστέλλονται ασύρματα σε μια πλακέτα συλλογής πληροφοριών όπου είναι υπεύθυνη για την επικοινωνία με ένα κεντρικό σύστημα επεξεργασίας που μπορεί να είναι κάποιος ηλεκτρονικός υπολογιστής.

Κατά την διάρκεια της μέτρησης ο ασθενής θα πρέπει να βρίσκεται σε όρθια θέση έχοντας βγάλει την μπλούζα του και χωρίς να κουνάει καθόλου τους ώμους τους και το στήθος του.

Το αποτέλεσμα της μέτρησης βγαίνει από τον μέσο όρο πέντε προσπαθειών βαθιάς εισπνοής και εκπνοής. Μεταξύ των προσπαθειών δίνεται στον ασθενή χρόνος ξεκούρασης και χαλάρωσης ώστε τα αποτελέσματα των μετρήσεων να μην επηρεαστούν λόγω κόπωσης τους ασθενούς.

Σύμφωνα με τα αποτελέσματα αυτής της έρευνας που πραγματοποιήθηκε σε τρεις κατηγορίες ασθενών με ομαδοποίηση κατά την ηλικία τους , διαπιστώθηκε ότι υπάρχουν διαφοροποιήσεις οι οποίες οφείλονται είτε σε λόγους ανατομίας λόγω της γήρανσης η οποία μειώνει την διαφραγματική δύναμη και αποδυναμώνει τον ιστό στήριξης , είτε σε μορφολογικές αλλαγές στην κινητικότητα των σκελετικών μυών και την ελαστικότητα των γειτονικών μυϊκών ιστών που επηρεάζει την επέκταση τους στήθους κατά την αναπνοή.

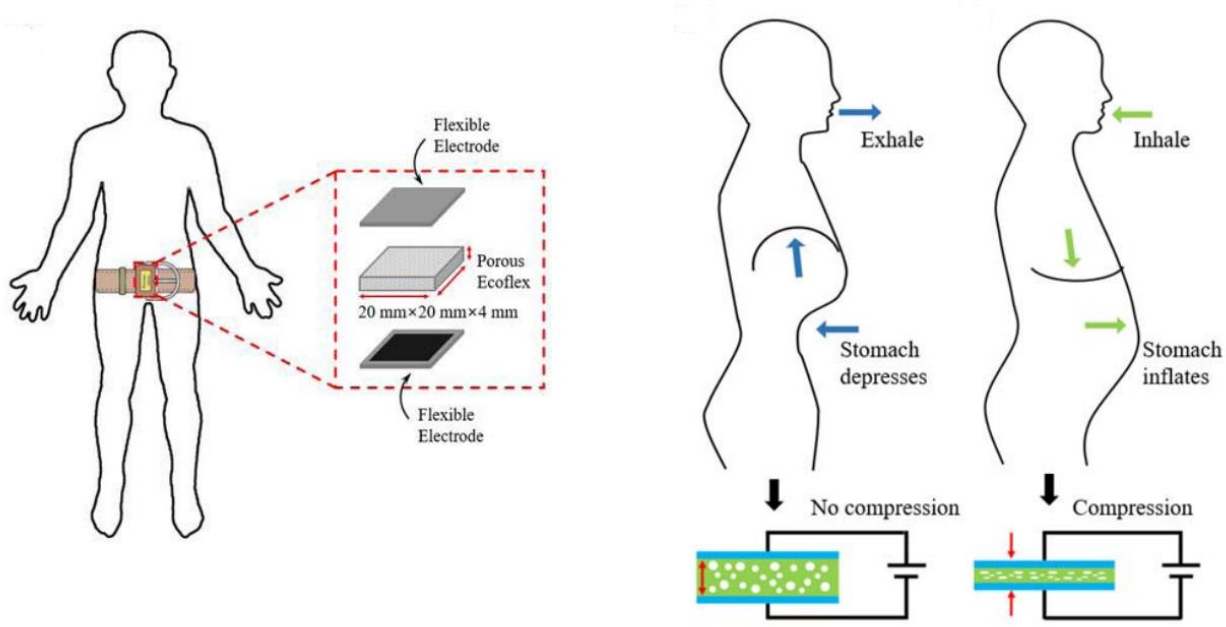
- **Μέθοδος με την χρήση αισθητηρίου χωρητικότητας**

Πρόκειται για έναν αισθητήρα εύκαμπτης χωρητικής πίεσης που έχει κατασκευαστεί χρησιμοποιώντας κατάλληλη χημεία με σκοπό την μέτρηση τέτοιου είδους αυξομείωση. Η γενική εξίσωση που διέπει την χωρητικότητα C ενός πυκνωτή δίνεται από τον τύπο,

$$C = \varepsilon_o \varepsilon_r A/d \quad (1.1)$$

όπου ε_o είναι η διηλεκτρική σταθερά του ελεύθερου χώρου ε_r η σχετική διηλεκτρική σταθερά του υλικού, *A* είναι η περιοχή του ηλεκτροδίου και d είναι ο διαχωρισμός μεταξύ των ηλεκτροδίων. Αυτός ο ειδικός αισθητήρας είναι ενσωματωμένος σε μια ζώνη μέσης μαζί με έναν μετατροπέα χωρητικότητας σε τάση και ανιχνεύει την αναπνοή σε πραγματικό χρόνο.

Για τη μέτρηση του σήματος εξόδου της αναπνοής, ο ασθενής θα πρέπει απλώς να φοράει τη ζώνη μέσης το σώμα του / της πάνω από τα ρούχα του χωρίς να χρειάζεται να η απευθείας του επαφή με το σώμα.



Σχήμα 1.5 Μέθοδος μέτρησης αναπνοής στην περιοχή της κοιλίας με την χρήση χωρητικού αισθητηρίου

Η γενική αρχή λειτουργίας τους αισθητηρίου είναι ότι κατά την διάρκεια της συμπίεσης του, προκαλείται παραμόρφωση του διηλεκτρικού στρώματος όπου μπορεί να προκαλέσει μεταβολή στη διηλεκτρική σταθερά του. Από την άλλη πλευρά, η παραμόρφωση του διηλεκτρικού στρώματος μπορεί επίσης να προκαλέσει την αλλαγή της απόστασης μεταξύ του άνω και του κάτω ηλεκτροδίου. Και οι δύο αυτοί λόγοι θα επηρεάσουν την χωρητικότητα του αισθητήρα χωρητικότητας, επιτρέποντας του να παρέχει μεγάλη ευαισθησία στα αποτελέσματα της μέτρησης.

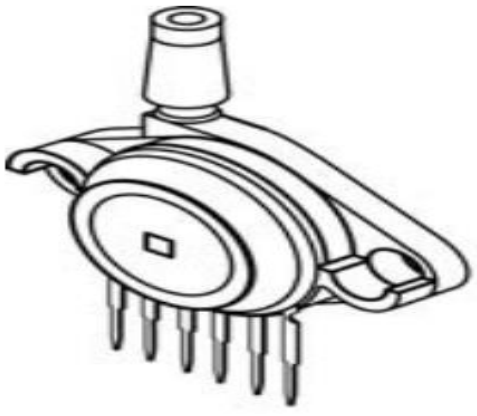
Όταν το άτομο εισπνέει, το διάφραγμα συστέλλεται και το στομάχι φουσκώνει, οπότε ο αισθητήρας είναι συμπιεσμένος. Κατά τη διάρκεια αυτής της κατάστασης, τα ηλεκτρόδια πλησιάζουν πιο κοντά το ένα με το άλλο μειώνοντας έτσι το κλάσμα κενού αναμεσά τους που οδηγεί σε αύξηση της διηλεκτρικής σταθεράς ε_r του χωρητικού αισθητηρίου. Παράλληλα μειώνεται και η απόσταση των οπλισμών d , με επακόλουθο αποτέλεσμα την αύξηση της χωρητικότητας λόγω αυτών των δύο παραγόντων.

Στην περίπτωση της εκπνοής το διάφραγμα επεκτείνεται και το στομάχι καταπιέζεται. Κατά τη διάρκεια αυτής της κατάστασης, ο αισθητήρας δεν συμπιέζεται, έτσι η απόσταση μεταξύ των ηλεκτροδίων παραμένει αμετάβλητη. Λόγω της αποτελεσματικής διηλεκτρικής σταθεράς που είναι ο συνδυασμός του αέρα και του ecoflex , το κλάσμα κενού του πορώδους αυτού στοιχείου είναι πιο κυρίαρχο , οπότε ή διηλεκτρική σταθεράς ε_r δεν επηρεάζεται.

Κατά συνέπεια στην ενεργητική κοιλιακή εκπνοή δηλαδή στην εκπνοή που καταπιέζεται το στομάχι πιο βαθιά σε σχέση με την κατάσταση ισορροπίας ο αισθητήρας δεν συμπιέζεται εφόσον δεν μπορεί να παράγει μέτρηση και εμφανίζει την χωρητική αντίσταση που έχει σε κατάσταση ισορροπίας.

- **Μέθοδος με την χρήση αισθητηρίου πίεσης**

Πρόκειται για έναν αισθητήρα που ανιχνεύει τις μεταβολές της πίεσης του αέρα και στηρίζεται στο πιεζοηλεκτρικό φαινόμενο. Δηλαδή, την ικανότητα ορισμένων υλικών να παράγουν ηλεκτρικό φορτίο ως απόκριση στην εφαρμοζόμενη μηχανική καταπόνηση.
Κατά τη διάρκεια αυτής της κατάστασης, πραγματοποιείται αλλαγή των κέντρων θετικού και αρνητικού φορτίου στο υλικό, το οποίο στη συνέχεια οδηγεί σε εξωτερικό ηλεκτρικό πεδίο.



Σχήμα 1.6 Εξωτερική εμφανίση του αισθητήρα πίεσης MPX5050GP

Για την μέτρηση της διαφραγματικής αναπνοής ο ασθενής χρειάζεται να φοράει μια ειδικά κατασκευασμένη ζώνης μέσης. Πρόκειται για μια ζώνη που βασίζεται στην αρχή λειτουργίας στην οποία βασίζεται και ένα πιεσόμετρο μπράτσου.

Η ζώνη μέσης αποτελείται από μια φουσκωτή περιοχή υπεύθυνη για την μέτρηση της διαφραγματικής αναπνοής, μια μανσέτα για φούσκωμα που συνδέεται μέσω σωλήνα με την ζώνη και έναν σωλήνα που συνδέει την ζώνη με την μονάδα μέτρησης δηλαδή το αισθητήριο πίεσης.

Η μανσέτα περιέχει μία μονόδρομη βαλβίδα για να αποφευχθεί ακούσια διαρροή της

πίεσης ενώ υπάρχει μια ρυθμιζόμενη βαλβίδα με κοχλία για το χειριστή για να επιτρέψει την πίεση στον σύστημα να πέσει με έναν ελεγχόμενο τρόπο.

Για την διεξαγωγή της μέτρησης ο ασθενής χρειάζεται να τοποθετήσει την φουσκωτή περιοχή της ζώνης στο κέντρο της κοιλιάς του και να την φουσκώσει σε μια πίεση αναφοράς. Κατά την αναπνοή το στομάχι διαστέλλεται και σπρώχνει την φουσκωτή περιοχή της ζώνης δημιουργώντας επιπλέον πίεση επάνω της. Κατά την εκπνοή εφαρμόζεται ενεργητική κοιλιακή εκπνοή, το στομάχι καταπιέζεται και η πίεση στη φουσκωτή περιοχή μειώνεται και πέφτει κάτω από την πίεση ισορροπίας.

1.5 Επιλογή μεθόδου

Στην παρούσα εργασία εξετάζεται η μέθοδος με την χρήση αισθητηρίου πίεσης , για λόγους απλότητας μέτρησης και κόστους. Κύριο σκοπός είναι η κατασκευή μια ολοκληρωμένης μετρητικής συσκευής διαφραγματικής αναπνοής σε ένα σενάριο γιατρός – ασθενής . Πιο συγκεκριμένα.

1. Η κατασκευή ενός κυκλώματος προσαρμογής του αισθητηρίου πίεσης και φιλτραρίσματος , με σκοπό την εξαγωγή μια αξιόπιστης μέτρησης σε άμεση σχέση με την διαγραμματική αναπνοή.
2. Η real time απεικόνιση της μέτρησης σε οθόνη , αξιοποιώντας την σημαντική έννοια της οπτικής ανατροφοδότησης ώστε να βοηθήσει τον ασθενή να καταβάλλει όσο το δυνατόν περισσότερη προσπάθεια για το βέλτιστο αποτέλεσμα.
3. Η επικοινωνία της συσκευής με έναν ηλεκτρονικό υπολογιστή για την μεταφορά των δεδομένων μέτρησης , δημιουργία προφίλ ασθενή και διατήρηση ιστορικού με σκοπό την αξιολόγηση της εξέλιξης της διαφραγματικής αναπνοής του ασθενή.
4. Η σωστή και εργονομική σύνδεση όλων των εξαρτημάτων μεταξύ τους και η παροχή φορητότητας στην συσκευή.

Παρακάτω αναλύονται οι μέθοδοι που χρησιμοποιήθηκαν για να επιτύχουν όσο το δυνατόν κοντά σε αυτό το επιθυμητό αποτέλεσμα που περιγράφεται παραπάνω. Σημαντικά σημεία που αξίζει να αναφερθούν κατά την διαδικασία κατασκευής της συσκευής είναι η εν βάθος ενασχόληση με τον μικροεπεξεργαστή αφού αποτελεί τον εγκέφαλο του συστήματος και απαιτείται η κατανόηση της δομής του , των λειτουργιών του αλλά και ο προγραμματισμός του. Η επικοινωνία που εξυπηρετεί τον δίαυλο επικοινωνίας της συσκευής με τον κεντρικό υπολογιστή. Η κατασκευή ιστοσελίδων τοπικού δικτύου και η ανάγκη δημιουργίας βάσης δεδομένων για την αποθήκευση των μετρήσεων ανά ασθενή.

ΚΕΦΑΛΑΙΟ 2

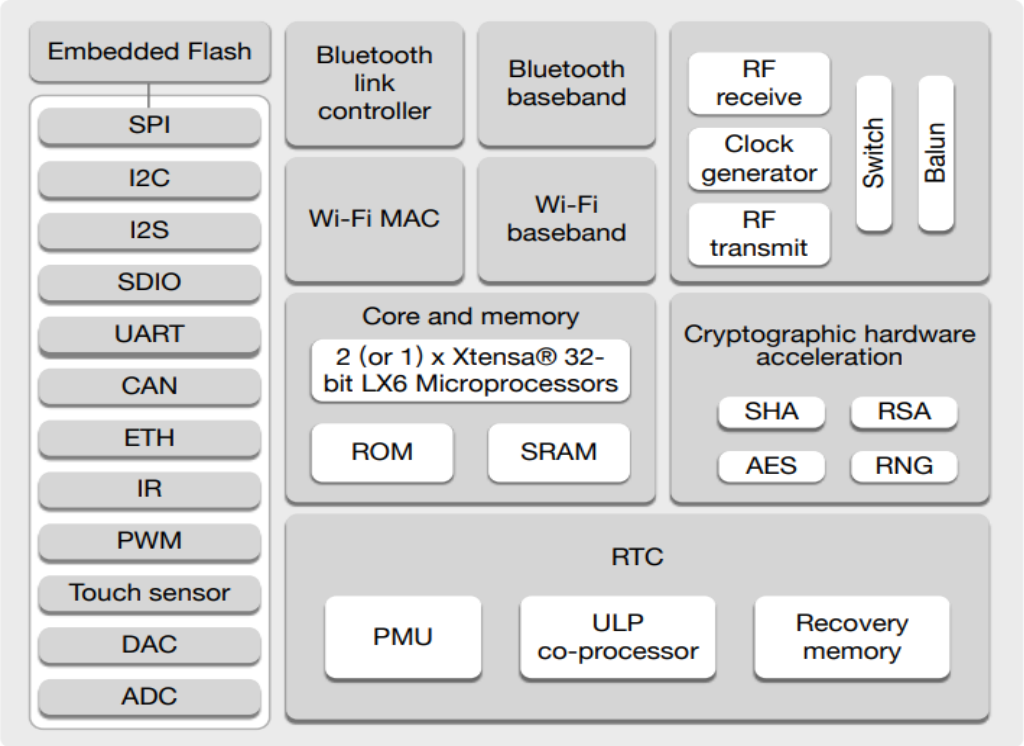
ΜΙΚΡΟΕΛΕΓΚΤΗΣ ESP32

2.1 Εισαγωγή

Μικροελεγκτής είναι ένα ολοκληρωμένο κύκλωμα (Integrated Circuit - IC), το οποίο έχει ενσωματωμένα εκτός από τον μικροεπεξεργαστή και διάφορα περιφερειακά όπως πχ. μνήμες RAM, ROM, πόρτες I/O, χρονιστές/απαριθμητές, σειριακή επικοινωνία, μετατροπέα αναλογικού σήματος σε ψηφιακό, κτλ. Χρησιμοποιούνται σε όλες τις εφαρμογές αυτοματισμού (πχ εργοστάσια, ηλεκτρικές συσκευές, ρομποτική), σε συστήματα ελέγχου (Η/Υ, αυτοκίνητα, κινητά), σε συστήματα ελέγχου προσπέλασης (ασφάλεια), κτλ. Με άλλα λόγια, μπορεί ένας μικροελεγκτής (μέσω των ενσωματωμένων περιφερειακών του) να συνδεθεί με διάφορες εξωτερικές συσκευές, όπως πχ. αισθητήρες, αντιστάσεις, εξωτερικό ρολόι, βηματικούς κινητήρες, διακόπτες, bluetooth, usb, κτλ. και να προκαλέσει ανάλογα με την ρύθμιση του, την επιθυμητή ενέργεια, η να εμφανίσει το αποτέλεσμα στην οθόνη. συμβατή με τα χαρακτηριστικά εισόδου της συσκευής (μορφή τάσης, τάση λειτουργίας, ισχύς εισόδου).

Η αγορά του Διαδικτύου των πραγμάτων (IoT) έχει επεκταθεί ταχέως τα τελευταία χρόνια μετά την αυξημένη ζήτηση επικοινωνίας και ελέγχου για διάφορες συσκευές και συσκευές. Η κύρια απαίτηση που εφαρμόζεται για τις σύγχρονες συσκευές IoT είναι η παροχή αποτελεσματικής συνδεσιμότητας για την εξασφάλιση αξιόπιστης απομακρυσμένης επικοινωνίας και μεταφοράς δεδομένων σε ασύρματο περιβάλλον. Κάθε μονάδα που βασίζεται σε IoT αποτελείται από έναν μικροελεγκτή και μια ασύρματη μονάδα μετατροπής (συνήθως WiFi) ή έναν συνδυασμό και των δύο σε ένα. Το IoT βελτιώνει τον αυτοματισμό του σπιτιού και εισάγει νέες τεχνολογίες που βασίζονται στην επικοινωνία όπως η υποβοηθούμενη διαβίωση, η ηλεκτρονική υγεία και η ηλεκτρονική μάθηση. Προκειμένου να αναπτυχθεί περαιτέρω το IoT και να επεκταθεί ο τομέας των εφαρμογών του, απαιτούνται ισχυρές, χαμηλού κόστους και χαμηλής ισχύος λύσεις για τις συσκευές IoT. Όσο μικρότερο το μέγεθος και το βάρος της συσκευής τόσο ευρύτερη είναι η περιοχή των εφαρμογών της.. Μια μεγάλη ποικιλία μονάδων και μικροελεγκτών είναι ήδη στην αγορά και χρησιμοποιούνται ευρέως για το σχεδιασμό και την ανάπτυξη συσκευών IoT. Αυτά είναι τα Xbee, WhizFi, ορισμένες πλακέτες Arduino. Ωστόσο, οι περισσότερες συσκευές που προσφέρονται σήμερα είναι είτε αρκετά ακριβές είτε μεγάλες όσον αφορά το βάρος και το μέγεθος. Επιπλέον, πολύ λίγες μονάδες είναι συσκευές ανοιχτού κώδικα και δεν έχουν περιορισμό στο σκοπό λειτουργίας.

Η εταιρεία Espressif Systems έρχεται να απαντήσει σε αυτές τις προκλήσεις που απαιτούνται από μια συσκευή στο διαδίκτυο των πραγμάτων κατασκευάζοντας το ESP32. Πρόκειται για ένα System on Chip χαμηλού κόστους – χαμηλής ισχύος με δυνατότητες Wi-Fi και Bluetooth και μια εξαιρετικά ολοκληρωμένη δομή που τροφοδοτείται από έναν μικροεπεξεργαστή διπλού πυρήνα Tensilica Xtensa LX6. Για τις απαιτήσεις της συσκευής που εξετάζεται , το ESP32 κρίνεται το κατάλληλο στοιχείο ώστε να καλύψει τις ανάγκες του μικροελεγκτή ως κεντρικού επεξεργαστή του συστήματος και της επικοινωνίας με τον υπολογιστή αφού καλύπτει και την σύνδεση WiFi. Παρακάτω αναλύεται η εσωτερική δομή του και όλα τα χαρακτηριστικά που το απαρτίζουν.



Σχήμα 2.1 Μπλοκ διάγραμμα του ESP32

2.2 Μπλοκ διάγραμμα

Η δομή του μικροελεγκτή ESP32 έχει σχεδιαστεί για να λειτουργεί στα ακόλουθα πρωτόκολλα - TCP / IP, πλήρες 802.11 b / g / n / e / i WLAN MAC και Wi-Fi Direct. Ο μικροελεγκτής μπορεί να παρέχει βασικό σετ υπηρεσιών (BSS) Station και λειτουργία SoftAP.

- **Station**: Μπορεί να λειτουργήσει ως σταθμός ώστε να συνδεθεί στο διαδίκτυο ή σε ένα τοπικό δίκτυο.
- **SoftAP**: Μπορεί να λειτουργήσει ως σημείο πρόσβασης (access point) προκειμένου να παρέχεται μια διεπαφή χρήστη για παραδείγμα, να υποστηρίξει μια εφαρμογή η σελίδα στο κινητό τηλέφωνο.

Ο μικροελεγκτής υποστηρίζει BLE Bluetooth και είναι ικανό να λειτουργήσει με ταχύτητα έως 4 Mbps. Το ESP32 μπορεί να λειτουργήσει κάτω διάφορες από διάφορες λειτουργίες ισχύος.

- **Ενεργή λειτουργία**: Το τσιπ υπεύθυνο για την επικοινωνία RF είναι ενεργοποιημένο. Το τσιπ μπορεί να λάβει, να μεταδώσει ή να ακούσει.

- Λειτουργία modem-sleep: Η CPU είναι λειτουργική και το ρολόι είναι διαμορφώσιμο. Η λειτουργίες Wi-Fi / Bluetooth και το τσιπ υπεύθυνο για την επικοινωνία RF είναι απενεργοποιημένα.

- **Λειτουργία ελαφρύ ύπνου**: Η CPU έχει τεθεί σε παύση. Η μνήμη RTC και τα περιφερειακά RTC, καθώς ο ULP(Ultra Low Power) βοηθητικός επεξεργαστής. Οποιαδήποτε συμβάντα αφύπνισης (MAC, κεντρικός υπολογιστής, χρονοδιακόπτης RTC ή εξωτερικές διακοπές) θα ξυπνήσουν το τσιπ.

- **Λειτουργία βαθύ ύπνου**: Μόνο η μνήμη RTC και τα περιφερειακά RTC είναι ενεργοποιημένα. Τα δεδομένα σύνδεσης του Wi-Fi και Bluetooth αποθηκεύονται στη μνήμη RTC. Ο ULP(Ultra Low Power) βοηθητικός επεξεργαστής είναι λειτουργικός.

- **Λειτουργία αδρανοποίησης**: Ο εσωτερικός ταλαντωτής 8-MHz και ο βοηθητικός επεξεργαστής ULP είναι απενεργοποιημένοι. Η RTC μνήμη ανάκτησης είναι απενεργοποιημένη. Μόνο ένας χρονοδιακόπτης RTC στο αργό ρολόι και ορισμένα RTC GPIOs είναι ενεργά. Ο χρονοδιακόπτης RTC ή κάποια RTC GPIO μπορούν να ξυπνήσουν το τσιπ από τη λειτουργία αδρανοποίησης.

Τα GPIO περιλαμβάνουν δύο ADC 12-bit με συνολικά 18 κανάλια. Αυτά μπορεί να είναι διαμορφωμένο για αναλύσεις 9-bit, 10-bit και 12-bit με ένα εξασθένητή −0dB, −6dB ή −11dB για διαφορετικό εύρος εισόδου.

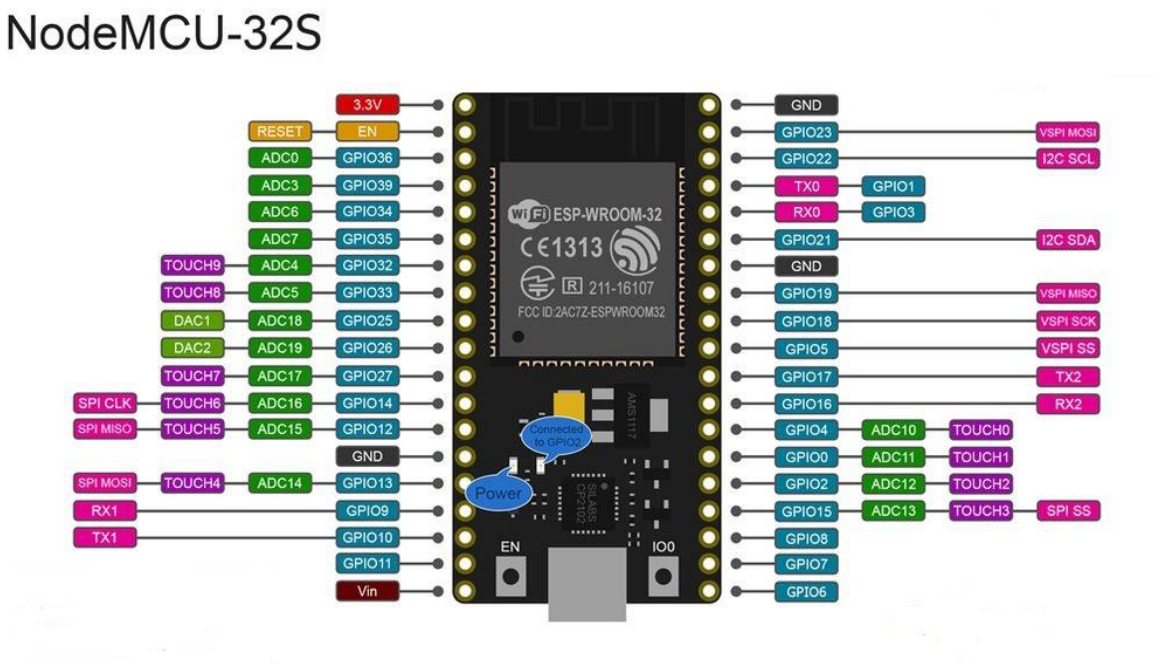
Εκτός από τους ADC υπάρχουν επίσης δύο 8-bit DAC για τη μετατροπή των ψηφιακών σημάτων σε αναλογικές εξόδους σήματος τάσης.

Δέκα από τα GPIO είναι ικανά να αισθανθούν χωρητικές αλλαγές και μπορούν να χρησιμοποιηθούν σαν αισθητήρες αφής.

Επιπλέον, το ESP32 παρέχει μια σειρά διεπαφών.

- Μια διέπαφη Ethernet MAC που είναι υπεύθυνη για την ενσύρματη επικοινωνία σε κάποιο δίκτυο.
- Έναν ελεγκτή SD/SDIO/MMC για την σύνδεση κάποιας SD κάρτας.
- Τρεις διεπαφές UART με ταχύτητα μέχρι τα 5Mbps για επικοινωνία με κάποια συσκευή η υπολογιστή που χρησιμοποιεί αυτό το πρωτόκολλο.
- Δύο I2C κανάλια με δυνατότητα λειτουργείας με κανονικό και γρήγορο ρυθμό με αρχική συχνότητα τα 10kHz μέχρι τα 10MHz
- Έναν τηλεχειριστή υπέρυθρων οκτώ καναλιών.
- Οχτώ μονάδες PCNT (Pulse Counter) μετρητών ανερχόμενων και καθοδικών παλμών ενός σήματος εισόδου.
- Έναν ελεγκτή διαμόρφωσης εύρους παλμών (PWM) ο οποίος μπορεί να οδηγήσει ψηφιακούς κινητήρες είτε να παράγει ψηφιακές κυματομορφές.
- Τέσσερα κανάλια για την επικοινωνία (SPI) σε λειτουργεία master ή slave με ρολόι χρονισμού μέχρι τα 80MHz.

2.3 Αναπτυξιακή πλακέτα



Σχήμα 2.2 Η αναπτυξιακή πλακέτα NodeMCU-32S που χρησιμοποιήθηκε στην συσκευή

Στη συγκεκριμένη συσκευή που εξετάζεται χρησιμοποιείται η αναπτυξιακή πλακέτα NodeMCU-32S η οποία είναι κατασκευασμένη από την κινέζικη εταιρεία Ai-Thinker. Εκτός από τον μικροελεγκτή ESP32 συμπεριλαμβάνεται ένας γραμμικός σταθεροποιητής τάσης AMS1117 από 5 Volt σε 3.3 Volt για την τροφοδοσία του μικροελεγκτή , μια θύρα micro- USB ένα τσιπ μετατροπέα USB σε σειριακή CP2102 και δύο μπουτόν τα οποία είναι υπεύθυνα για τον προγραμματισμό και την επανεκκίνηση (reset). Το τσιπ ESP32 γενικώς διαθέτει 48 pin με πολλαπλές λειτουργίες. Δεν εκτίθενται όλες οι ακίδες σε όλες τις πλακέτες ανάπτυξης και υπάρχουν μερικές ακίδες που δεν μπορούν να χρησιμοποιηθούν διότι είναι δεσμευμένες από το σύστημα.

- Το pin στο οποίο αναγράφεται η ετικέτα Vin είναι συνδεδεμένο απευθείας στην είσοδο του AMS1117 σταθεροποιητή τάσης , χρησιμοποιείται για να τροφοδοτήσει το σύστημα σε περίπτωση που δεν υπάρχει συνδεδεμένο το USB. Η εξωτερική τάση εισόδου προτείνεται να είναι στα 5 Volt για την καλύτερη λειτουργία του σταθεροποιητή
- Το pin στο οποίο αναγράφεται η ετικέτα RESET – EN , είναι υπεύθυνο για την επανεκκίνηση του μικροελεγκτή και ενεργοποιείται αν συνδεθεί στην γείωση GND , είτε αν πατηθεί το μπουτόν EN.
- Τα pin στα οποία αναγράφεται η ετικέτα GPIO , σημαίνει ότι είναι pin γενικού σκοπού και μπορούν να χρησιμοποιηθούν είτε σαν έξοδοι είτε σαν είσοδοι του συστήματος.
- Τα pin στα οποία αναγράφεται η ετικέτα TOUCH , σημαίνει ότι μπορούν να ανιχνεύσουν αλλαγές σε οτιδήποτε έχει ηλεκτρικό φορτίο , όπως το ανθρώπινο δέρμα. Αυτά τα pins μπορούν να χρησιμοποιηθούν επίσης για να βγάλουν τον μικροελεγκτή από τον βαθύ ύπνο.

- Τα pin στα οποία αναγράφεται η ετικέτα DAC , σημαίνει ότι σε αυτά μπορεί να γίνει μετατροπή ψηφιακού σήματος σε αναλογικό.
- Τα pin στα οποία αναγράφεται η ετικέτα ADC , σημαίνει ότι σε αυτά μπορεί να γίνει μετατροπή αναλογικού σήματος σε ψηφιακό.
- Τα pin στα οποία αναγράφεται η ετικέτα TX η RX , είναι υπεύθυνα για την σειριακή επικοινωνία UART.
- Τα pin στα οποία αναγράφεται η ετικέτα SPI και VSPI , είναι υπεύθυνα για την επικοινωνία SPI.

Κάποια pin του ESP32 δεν μπορούν να χρησιμοποιηθούν είτε σε ορισμένες συνθήκες είτε καθόλου. Είναι προβληματικά και μη ασφαλή ακόμα και αν εκτίθενται στην πλακέτα πρέπει να αποφεύγεται η χρήση τους.

❖ Τα pins αυτά είναι συνδεδεμένα με την εσωτερική SPI Flash μνήμη που βρίσκεται στον μικροελεγκτή.

- GPIO 6
- GPIO 7
- GPIO 8
- GPIO 9
- GPIO 10
- GPIO 11

❖ Τα pins αυτά μπορούν να χρησιμοποιηθούν μόνο σαν είσοδοι και δεν έχουνε ψηφιακές pull-up η pull-down αντιστάσεις.

- GPIO 34
- GPIO 35
- GPIO 36
- GPIO 39

❖ Τα pins αυτά (Strapping pins) είναι συνδεδεμένα εσωτερικά σε κατάσταση pull up , ή pull down για κάποιες απαραίτητες ενέργειες του συστήματος και χρησιμοποιούνται για να μπει ο μικροελεγκτής σε κατάσταση προγραμματισμού ή φόρτωσης bootloader.

- GPIO 0
- GPIO 2
- GPIO 4
- GPIO 5
- GPIO 12
- GPIO 15

- ❖ Ο ADC2 του ESP32 χρησιμοποιείται από τον οδηγό του Wi-Fi. Οπότε τα pins αυτά μπορούν να χρησιμοποιηθούν μόνο όταν το Wi-Fi δεν είναι ενεργοποιημένο στο σύστημα.
    - GPIO 4 (ADC2_CHO)
    - GPIO 0 (ADC2_CH1)
    - GPIO 2 (ADC2_CH2)
    - GPIO 15 (ADC2_CH3)
    - GPIO 13 (ADC2_CH4)
    - GPIO 12 (ADC2_CH5)
    - GPIO 14 (ADC2_CH6)
    - GPIO 27 (ADC2_CH7)
    - GPIO 25 (ADC2_CH8)
    - GPIO 26 (ADC2_CH9)

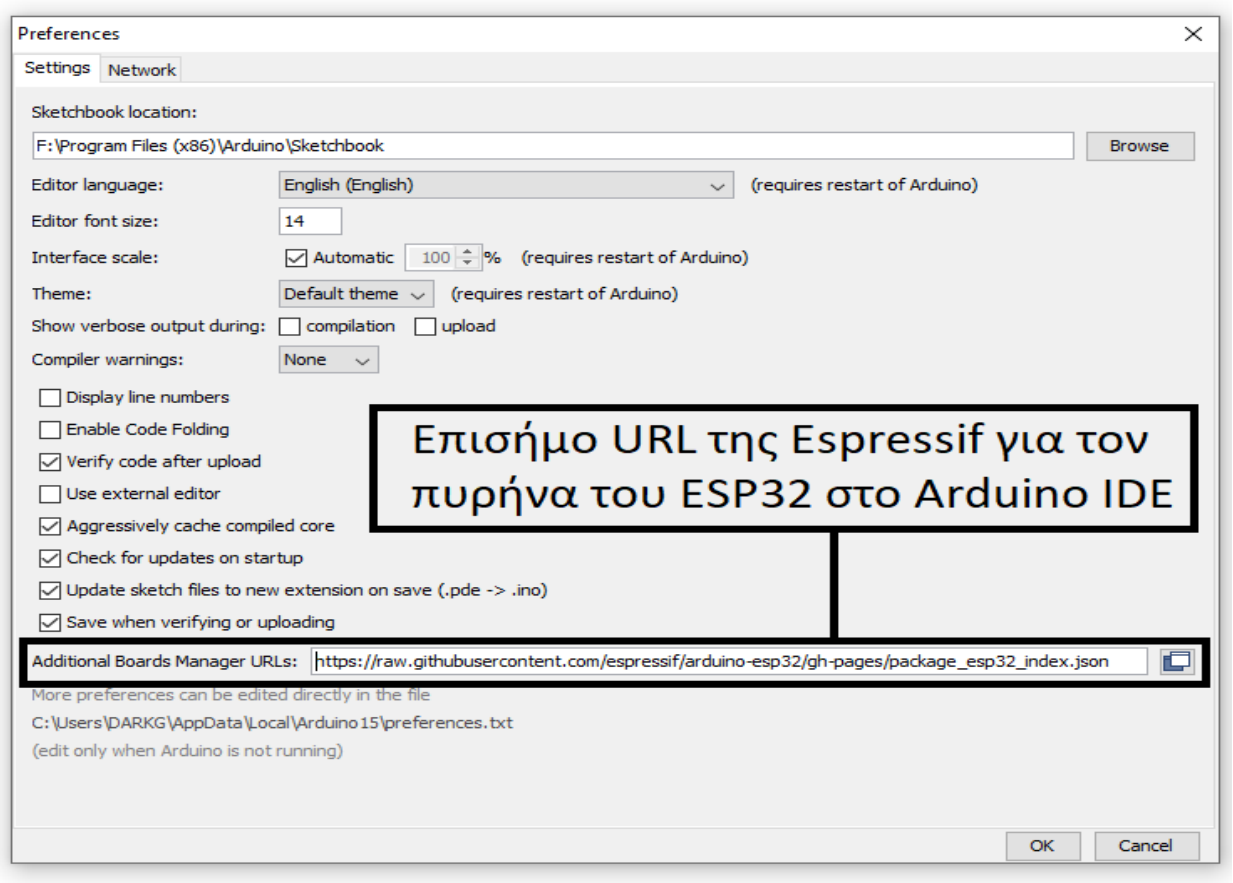
2.4 Προγραμματιστικό περιβάλλον

Ο βασικός τρόπος προγραμματισμού του ESP32 γίνεται με την χρήση του Espressif IoT Development Framework , όπου είναι διαθέσιμο στο επίσημο repository της Espressif systems στο github. Ο χρήστης θα πρέπει να είναι εξοικειωμένος με την χρήση του τερματικού όπου θα χρειαστεί να γραφούν οι κατάλληλες εντολές για να γίνει το compile του προγράμματος και η εισαγωγή του μέσα στον μικροελεγκτή. Αξίζει να αναφερθεί ότι το ESP32 μπορεί να προγραμματιστεί και με την χρήση του Eclipse IDE , όπου το περιβάλλον προγραμματισμού του είναι παρόμοιο με αυτό που προαναφέρθηκε.

Δυνατή είναι και η χρήση του Arduino IDE , όπου είναι ένα εργαλείο προγραμματισμού πιο φιλικό προς τον χρήστη , με την πολύ δημοφιλή χρήση στον χώρο του προγραμματισμού αναπτυξιακών πλακετών και της σειράς Arduino όπως αναγράφεται και στο όνομα του.

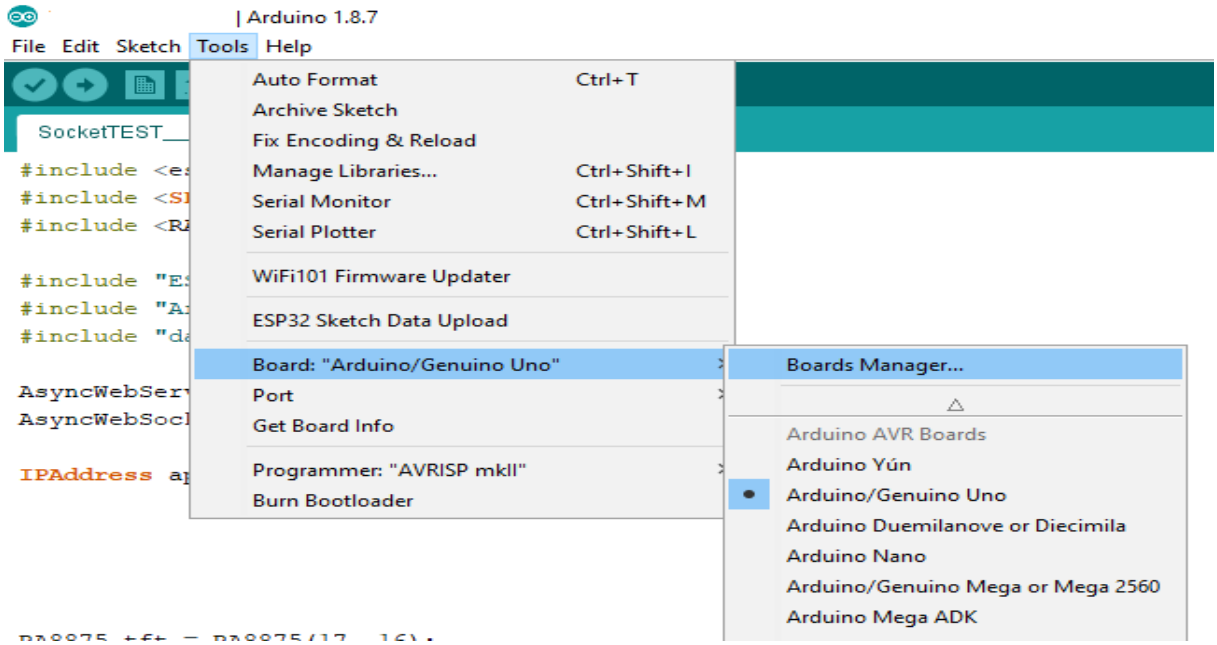
Το Arduino IDE , περιέχει ένα πρόγραμμα επεξεργασίας κειμένου για τη σύνταξη κώδικα σε γλώσσα C/C++, μια περιοχή μηνυμάτων, μια κονσόλα κειμένου, μια γραμμή εργαλείων με κουμπιά για κοινές λειτουργίες και μια σειρά μενού. Επίσης περιλαμβάνει έναν διαχειριστή board (board manager) , όπου επιλέγεις τον μικροελεγκτή που θες να προγραμματίσεις σε συνδυασμό με την αναπτυξιακή πλακέτα. Στην συγκεκριμένη εφαρμογή θα πρέπει να εγκατασταθεί ένας πυρήνας αρχείων για το ESP32 μέσω του board manager ώστε να μπορέσουμε να χρησιμοποιήσουμε το Arduino IDE.

Για την εγκατάσταση αυτού , ανοίγουμε το Arduino IDE , πατώντας Preferences (επιλογές) ανοίγει η καρτέλα όπου στο πλαίσιο Additional Boards Manager URLs: γράφουμε την διεύθυνση :

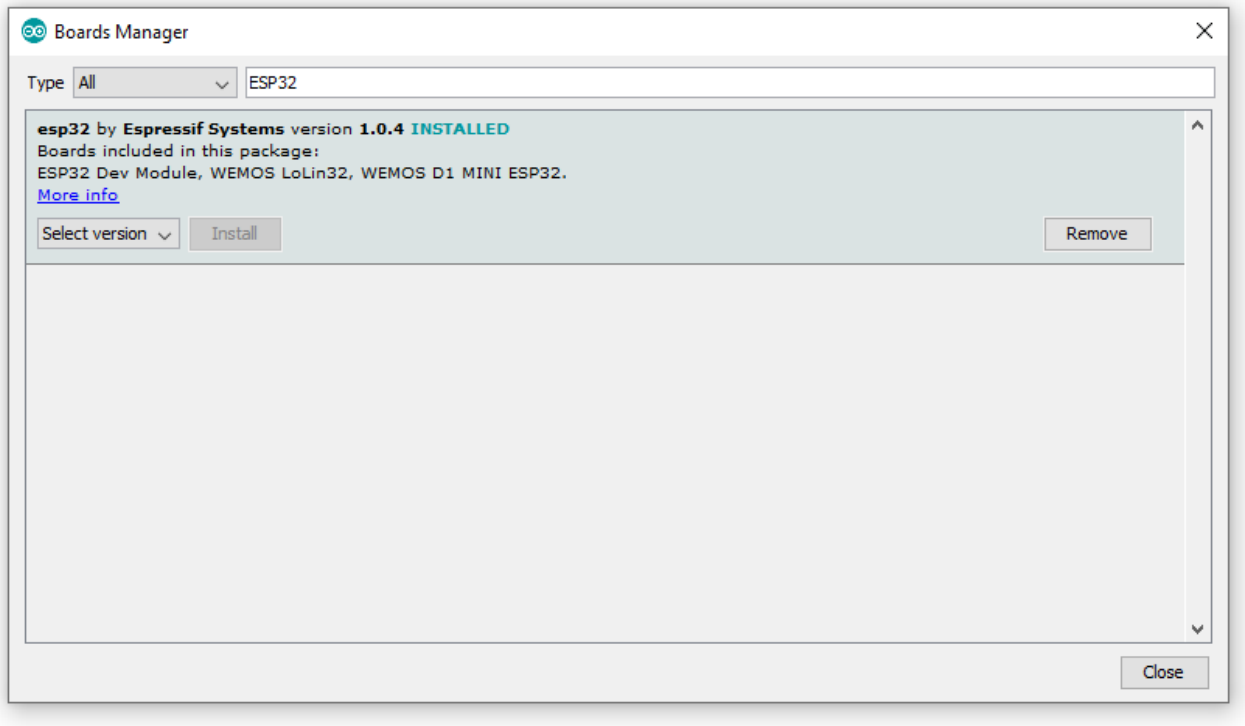


Σχήμα 2.3 Εισαγωγή URL στην καρτέλα του μενού Preferences του Arduino IDE για την προσθήκη του ESP32 core

Έπειτα ανοίγουμε τον Board Manager. Με αυτήν την σειρά Tools > Board > Boards Manager γράφουμε στην αναζήτηση ESP32 και κατεβάζουμε τον πυρήνα.

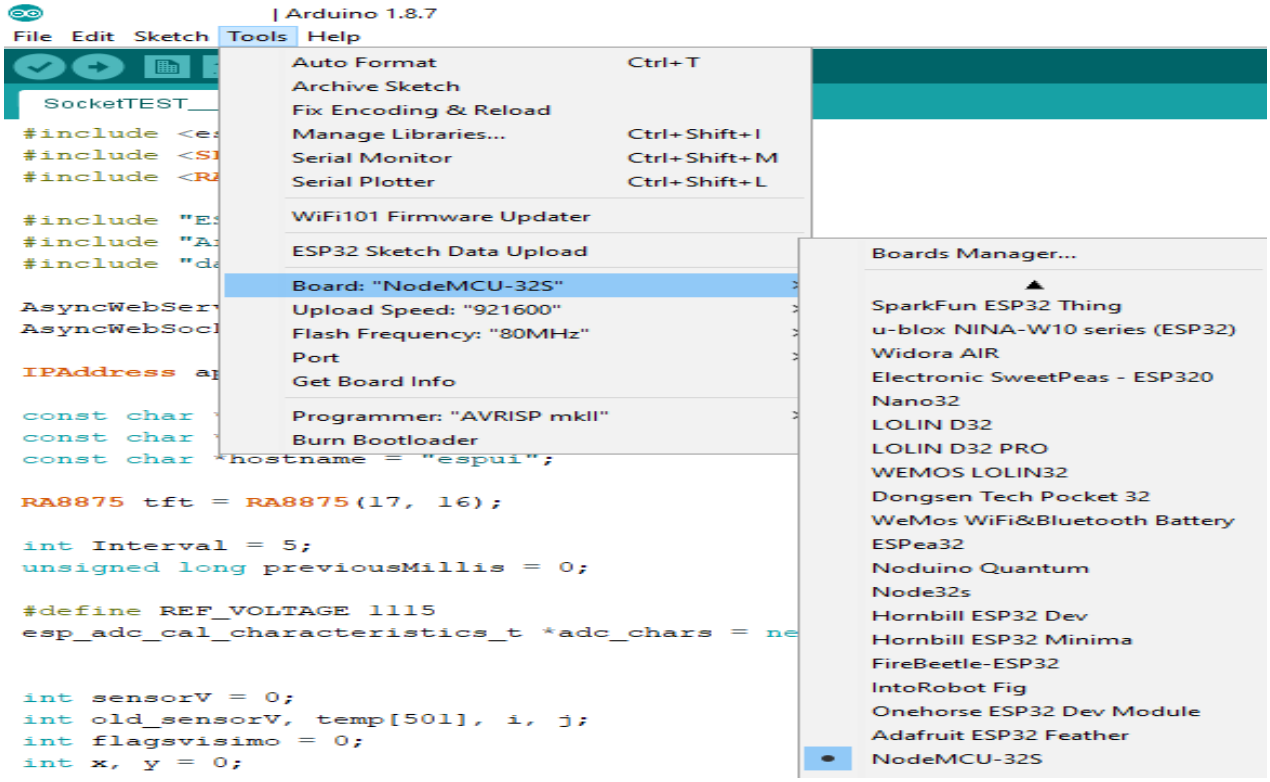


Σχήμα 2.4 Επιλογή Tools – Boards Manager στο Arduino IDE για την κατέβασμα του ESP32 core



Σχήμα 2.5 Αναζήτηση ESP32 στο Boards Manager του Arduino IDE

Μόλις κατεβεί ο πυρήνας ολοκληρώνεται και ολοκληρωθεί η εγκατάσταση επιλέγουμε ως board χρήσης το NodeMCU-32S και διαλέγοντας την κατάλληλη σειριακή θύρα Port ώστε να μπορεί να γίνει η επικοινωνία μέσω του USB.



Σχήμα 2.6 Επιλογή της πλακέτας NodeMCU-32s για προγραμματισμό μέσω του Arduino IDE

ΚΕΦΑΛΑΙΟ 3

ΑΙΣΘΗΤΗΡΙΟ ΠΙΕΣΗΣ

3.1 Εισαγωγή

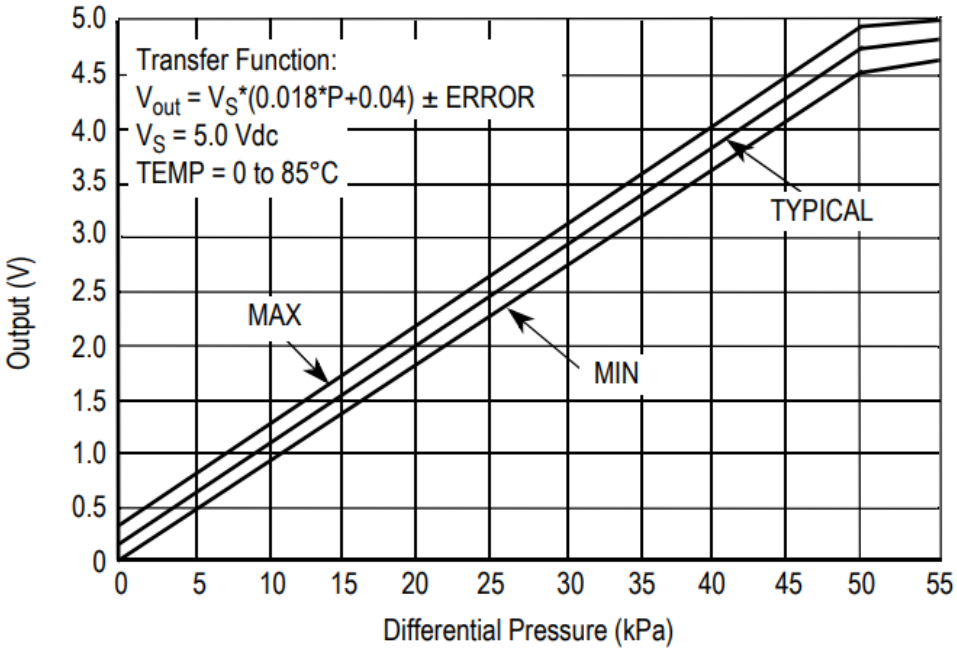
Στην κατασκευή που εξετάζεται αναφέρθηκε ότι η μέθοδος που θα χρησιμοποιηθεί για τη μέτρηση της διαφραγματικής αναπνοής είναι αυτή της μέτρησης με αισθητήρα πίεσης. Έπειτα από έρευνα στην αγορά για τέτοιους αισθητήρες, κατέληξα στην επιλογή του MPX5050GP της Freescale Semiconductor.

Αυτός ο αισθητήρας πίεσης είναι ιδανικός δεδομένου ότι έχει σχεδιασθεί ειδικά για όργανα πίεσης, μπορεί να μετράει πίεση στην περιοχή από 0 kPa έως 50 kPa που αντιστοιχεί σε 0 έως 376mmHg. Η τάση εξόδου του αισθητήρα πίεσης, δεδομένης μιας τάσης εισόδου 5V, κυμαίνεται θεωρητικά από 0 έως 4.7V, το οποίο είναι ένα ιδανικό εύρος για είσοδο σε ένα μετατροπέα αναλογικού σήματος σε ψηφιακό (ADC).

Η μετατροπή αυτή ακολουθεί τη συνάρτηση μεταφοράς του αισθητήρα, σύμφωνα με το Σχ. 3.1.

$$V_{out} = V_S\ (0.018P+0.04) \pm Error \qquad (3.1)$$

όπου ή V_{out} μπορεί να κυμανθεί σύμφωνα με τον κατασκευαστή από την ελάχιστη τάση V_{off} που η τυπική της τιμή είναι 0.2V, μέχρι τη μέγιστη τάση V_{FSO} που η τυπική της τιμή είναι 4.7V. Η ευαισθησία τυπικά είναι στα 90mV ανά kPa. Το σφάλμα *Error* του αισθητήρα κυμαίνεται από -1.25 kPa έως +1.25 kPa, το οποίο οφείλεται στις εσωτερικές ατέλειες του μετατροπέα.



Σχήμα 3.1 Σχέση τάσης προς πίεση σε περιβάλλον θερμοκρασίας από 0 ως 85 °C

Όπως συμπεραίνουμε ο μετατροπέας πίεσης είναι ήδη signal-conditioned από την εσωτερική λειτουργία ενισχυτή του συγκεκριμένου αισθητήρα (δηλαδή το αναλογικό σήμα έχει ήδη μετατραπεί σε τέτοια μορφή που είναι έτοιμο να ανταποκριθεί στις απαιτήσεις του επόμενου σταδίου επεξεργασίας) και συνεπώς μπορεί θεωρητικά να συνδεθεί απευθείας στην είσοδο ενός ADC, εφόσον περάσει πρώτα από τη διαδικασία του φιλτραρίσματος.

3.2 Φιλτράρισμα

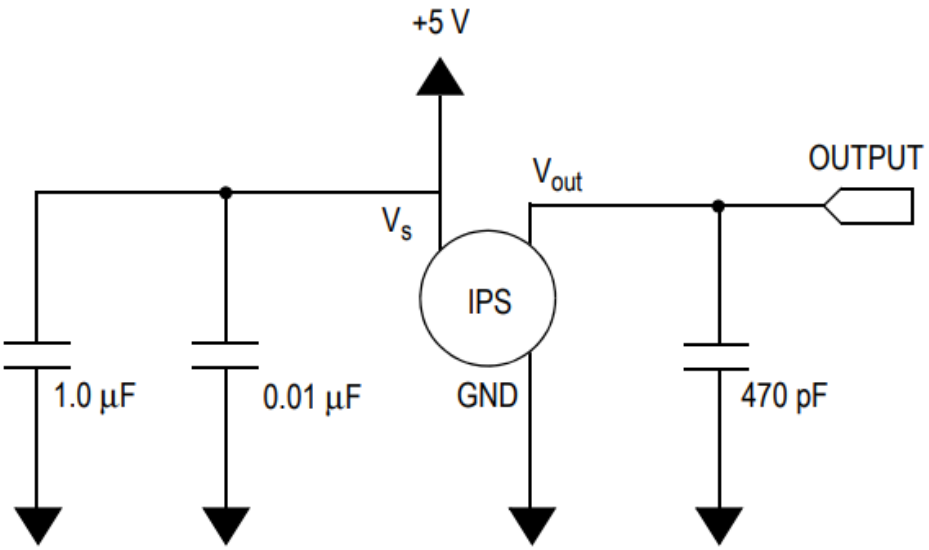
Οι μετρήσεις από τον πραγματικό κόσμο συχνά περιέχουν θόρυβο. Ο θόρυβος είναι το μέρος του σήματος που δεν είναι επιθυμητό. Ίσως προέρχεται από ηλεκτρικό θόρυβο. Οι τυχαίες παραλλαγές στη μέτρηση ενός αισθητήρα, που διακρίνονται κατά το διάβασμα ενός μικροελεγκτή με τη βοήθεια του μετατροπέα από αναλογικό σε ψηφιακό. Ο θόρυβος προκύπτει επίσης από τα εσωτερικά χαρακτηριστικά του αισθητήρα. Το φιλτράρισμα είναι μια μέθοδος για να αφαιρέσουμε μερικά από τα ανεπιθύμητα σήματα και ένα επιτευχθεί ένα πιο ομαλό αποτέλεσμα.

Σύμφωνα με το Σχ. 3.2 , ο κατασκευαστής προτείνει δύο decoupling πυκνωτές με τιμές 1.0μF και 0.01μF, ταυτόχρονα με έναν πυκνωτή τιμής 470pF για φιλτράρισμα της εξόδου του.

Στο Σχ. 3.3, φαίνεται η σχεδίαση ενός signal conditioning κυκλώματος, πέρα από αυτό που υπάρχει ήδη μέσα στον αισθητήρα. Με την βοήθεια του τελεστικού ενισχυτή TLC272 (U1A) ενισχύουμε το σήμα του αισθητήρα πίεσης (VOUT_SENSOR) με κέρδος 2, που καθορίζεται από το λόγο των αντιστάσεων R_2 και R_1.

$$A_V = \left(\frac{R_2}{R_1}\right) + 1 \tag{3.2}$$

Ενισχύοντας την εξόδου του αισθητήρα πίεσης με κέρδος 2, σημαίνει ότι δεν αξιοποιείται όλο το εύρος μέτρησής του. Αυτό εξηγείται διότι η θεωρητική έξοδος 0 έως 4.7V για 0 έως 50kPa, γίνεται με την ενίσχυση από 0 έως 9.4V, που δεν είναι εφικτό ο αισθητήρας να δώσει έξοδο πάνω από την τάση τροφοδοσίας του 5V.



Σχήμα 3.2 Προτεινόμενα στοιχεία φιλτραρίσματος του MPX5050GP από τον κατασκευαστή

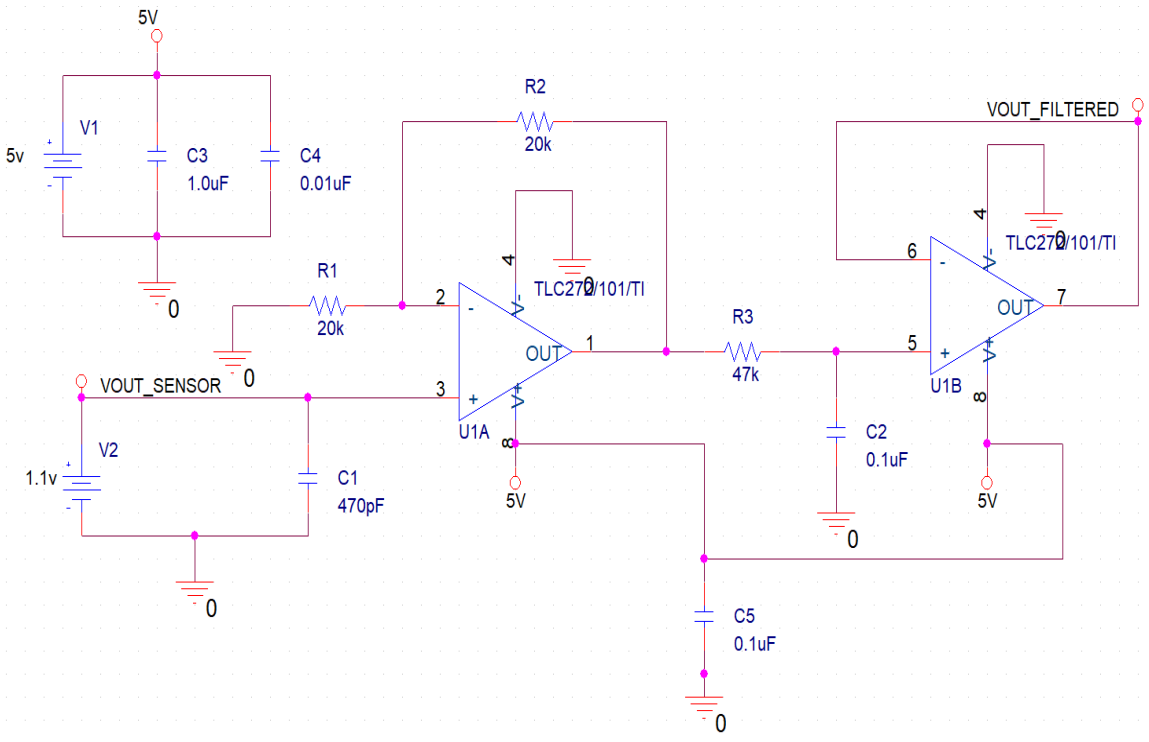
Άρα συμπεραίνεται ότι το νέο εύρος μέτρησης γίνεται από 0 έως 25 kPa, όπου αντιστοιχίζεται σε τάση από 0 έως 4.7V, το οποίο καλύπτει τις απαιτήσεις της εφαρμογής. Το κόστος του μειωμένου εύρους εξισορροπείται, εφόσον μετά την ενίσχυση η έξοδος του αισθητήρα από 90mV πηγαίνει στα 180mV ανά kPa, οπότε οι μεταβολές της πίεσης τώρα γίνονται καλύτερα αντιληπτές.

Η ενισχυμένη έξοδος του πρώτου τελεστικού U1A, περνάει από ένα χαμηλοπερατό φίλτρο 1ης τάξης, που αποτελείται από το δικτύωμα της αντίστασης R_3 και του πυκνωτή C_2. Η συχνότητα αποκοπής του φίλτρου ορίζεται από την εξίσωση

$$f_c = \frac{1}{2\pi RC} \tag{3.3}$$

όπου σύμφωνα με τις τιμές των στοιχείων προκύπτει η συχνότητα αποκοπής στα 33.87 Hz. Ο τελεστικός ενισχυτής U1B σε διάταξη follower υποδέχεται το φιλτραρισμένο σήμα στη θετική του είσοδο και το βγάζει αυτούσιο στην έξοδό του στο ποδαράκι 7. Οι πυκνωτές C_1, C_3, C_4 όπως αναφέρθηκε είναι οι προτεινόμενοι από τον κατασκευαστή. Ο πυκνωτής C_5 χρησιμοποιείται ως decoupling για τον TLC272.

Πέρα από το παθητικό φίλτρο πρώτης τάξης, έχει υλοποιηθεί ένα βαθυπερατό ψηφιακό φίλτρο Butterworth με συχνότητα αποκοπής 8 Hz. Το φίλτρο αυτό είναι ενσωματωμένο στον ESP32 σε μορφή κώδικα και κάθε αναλογική μέτρηση του αισθητηρίου πίεσης πρώτα εισάγεται σε αυτό και μετά χρησιμοποιείται στις κύριες λειτουργίες του προγράμματος.



Σχήμα 3.3 Κύκλωμα ενίσχυσης και χαμηλοπερατού φίλτρου 1ης Τάξης

3.3 Είσοδος στον ADC του ESP32

Το επόμενο βήμα για να ολοκληρωθεί η διαδικασία ανάγνωσης της μέτρησης του αισθητηρίου είναι η προσαρμογή της εξαγόμενης τάσης στην κλίμακα του μικροελεγκτή. Αυτό γίνεται επειδή ο μετατροπέας αναλογικού σε ψηφιακό του ESP32 μπορεί να λειτουργήσει από 0 έως 3.3 Volt στα μέγιστα όρια του. Για να λειτουργήσει στο εύρος αυτό θα χρειαστεί να ενεργοποιηθεί ο εσωτερικός εξασθενητής.

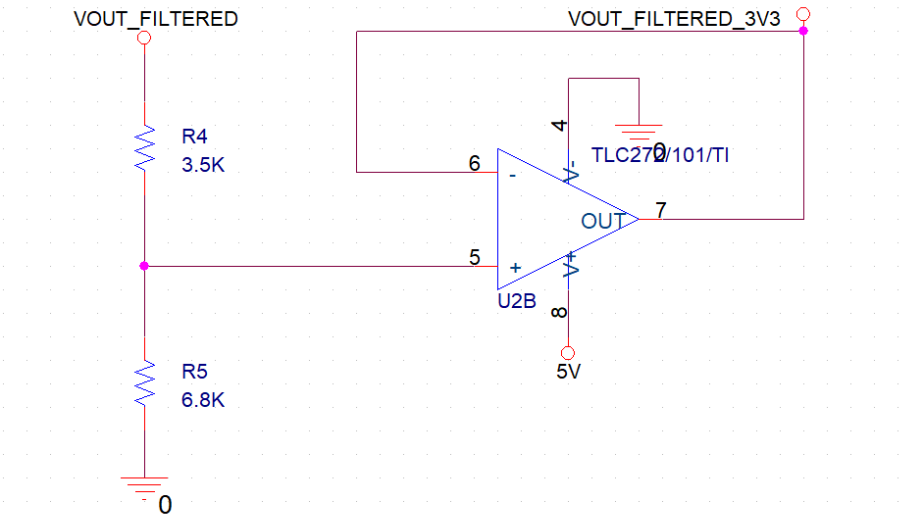
- 0dB εξασθένηση (ADC_ATTEN_DB_0), δίνει εύρος μέτρησης από 0 έως 1.1 Volt.
- 2.5dB εξασθένηση (ADC_ATTEN_DB_2_5), δίνει εύρος μέτρησης από 0 έως 1.5 Volt.
- 6dB εξασθένηση (ADC_ATTEN_DB_6), δίνει εύρος μέτρησης από 0 έως 2.2 Volt.
- 11dB εξασθένηση (ADC_ATTEN_DB_11), δίνει εύρος μέτρησης από 0 έως 3.9 Volt αλλά ταυτόχρονα περιορίζεται από την τάσης τροφοδοσίας των 3.3V.

Οπότε λόγω αυτού, η επιλογή της εξασθένησης γίνεται στα 11dB. Εφόσον έχει ρυθμιστεί ο ADC στο κατάλληλο εύρος που επιλέχτηκε, σειρά έχει να υποβιβαστεί η τάση του αισθητηρίου επίσης στο ίδιο εύρος. Αυτό επιτυγχάνεται με τη βοήθεια ενός διαιρέτη τάσης και ενός τελεστικού ενισχυτή σε συνδεσμολογία follower, όπως εικονίζεται στο Σχ. 3.4.

Ο παράγοντας υποβιβασμού που κάθε φορά πολλαπλασιάζει το σήμα ισούται με 0.66, όπως προκύπτει από το λόγο των αντιστάσεων, σύμφωνα με την Εξ. 3.4.

$$\frac{R_5}{(R_4+R_5)} \tag{3.4}$$

Η τιμή 0.66 προκύπτει από τη διαίρεση 3.3 Volt προς 5 Volt, δηλαδή, η επιθυμητή έξοδος προς την αρχική. Ο τελεστικός ενισχυτής σε συνδεσμολογία follower βοηθάει ώστε να προσφέρει χαμηλή σύνθετη αντίσταση εξόδου στο διαιρέτη τάσης. Με αυτόν τον τρόπο δεν θα φορτιστεί ο ADC του ESP32.



Σχήμα 3.4 Προσαρμογή εύρους του αισθητηρίου MPX5050GP στα 3.3Volt

Μια ακόμη σημαντική διαδικασία που χρειάζεται να γίνει, είναι αυτή της βαθμονόμησης (calibration) του ADC. Σύμφωνα με την Espressif υπάρχουν τρεις τρόποι για να επιτευχθεί αυτό.

1. Δύο σημείων (Two Point). Ο χρήστης θα πρέπει να εξάγει τις μετρήσεις του ADC στην περιοχή από 150mV έως τα 850mV και θα πρέπει να εγγραφούν στην εσωτερική του μνήμη.
2. eFuse V_{REF} (Εσωτερική τάση αναφοράς , γραμμένη στο τσιπ). Είναι η τιμή της εσωτερικής τάσης αναφοράς του μικροελεγκτή που έχει εγγραφεί κατά τη διαδικασία κατασκευής του από το εργοστάσιο.
3. Default V_{REF} (Προκαθορισμένη τάση αναφοράς). Είναι η τιμή της τάσης αναφοράς που έχει μετρηθεί από το χρήστη με τη χρήση εξωτερικής μονάδας μέτρησης αναλογικής τάσης (παλμογράφος ή πολύμετρο ακριβείας).

Η τάσης V_{REF} κυμαίνεται από 1000mV έως 1200mV και κάθε τσιπ ESP32 έχει διαφορετική τιμή. Στο συγκεκριμένο τσιπ που εξετάζεται, η τάση αυτή μετρήθηκε και ισούται με 1098mV. Για αυτόν τον λόγο δίνεται από τον κατασκευαστή μια διεπαφή προγραμματισμού εφαρμογών (Calibration API) για τη βαθμονόμηση. Το API αυτό δίνει μια συνάρτηση που παίρνει ως ορίσματα την ADC μονάδα που χρησιμοποιείς (ADC_UNIT_1), τον αριθμό εξασθένησης (11dB) , την ανάλυση (12 bit) και την αναφορά που θα επιλέξεις από τις τρεις (1098mV).

Στον Πίνακα 3.1 φαίνεται κάποια δειγματοληψία που πραγματοποιήθηκε για να φανεί η διαφορά των αποτελεσμάτων. Τα σφάλμα που προκύπτει πριν την βαθμονόμηση είναι αρκετά εμφανές. Οφείλεται στη μη γραμμικότητα του μετατροπέα και στο Offset error που είναι της τάξης των 140mV. Με τη βοήθεια της συνάρτησης βαθμονόμησης το σφάλμα ελαχιστοποιείται αισθητά και είναι της τάξης των 3mV, που είναι αρκετά ικανοποιητικό και αποδεκτό για τη συσκευή που εξετάζεται. Σε αυτό το σημείο πρέπει να αναφερθεί ότι όλες οι τιμές μέτρησης του Πίνακα 3.1 πρώτα φιλτραρίστηκαν με το βαθυπερατό ψηφιακό φίλτρο Butterworth με συχνότητα αποκοπής 8Hz, που χρησιμοποιείται και στο αισθητήριο πίεσης.

Πίνακας 3.1 Δείγματα μετρήσεων του ADC πριν και μετά την βαθμονόμηση

ΤΑΣΗ ΑΝΑΦΟΡΑΣ	ΠΡΙΝ ΤΗΝ ΒΑΘΜΟΝΟΜΗΣΗ	ΜΕΤΑ ΤΗΝ ΒΑΘΜΟΝΟΜΗΣΗ
200 mV	63mV	203 mV
400 mV	257mV	400 mV
600 mV	460 mV	598 mV
800 mV	653 mV	796 mV
1000 mV	856 mV	1001 mV
1200 mV	1057 mV	1200 mV
1400 mV	1260 mV	1403 mV
1600 mV	1457 mV	1600 mV
1800 mV	1658 mV	1802 mV
2000 mV	1855 mV	2000 mV
2200 mV	2052 mV	2195 mV
2400 mV	2256 mV	2399 mV
2600 mV	2476 mV	2595 mV
2800 mV	2741 mV	2803 mV
3000 mV	3068 mV	3010 mV

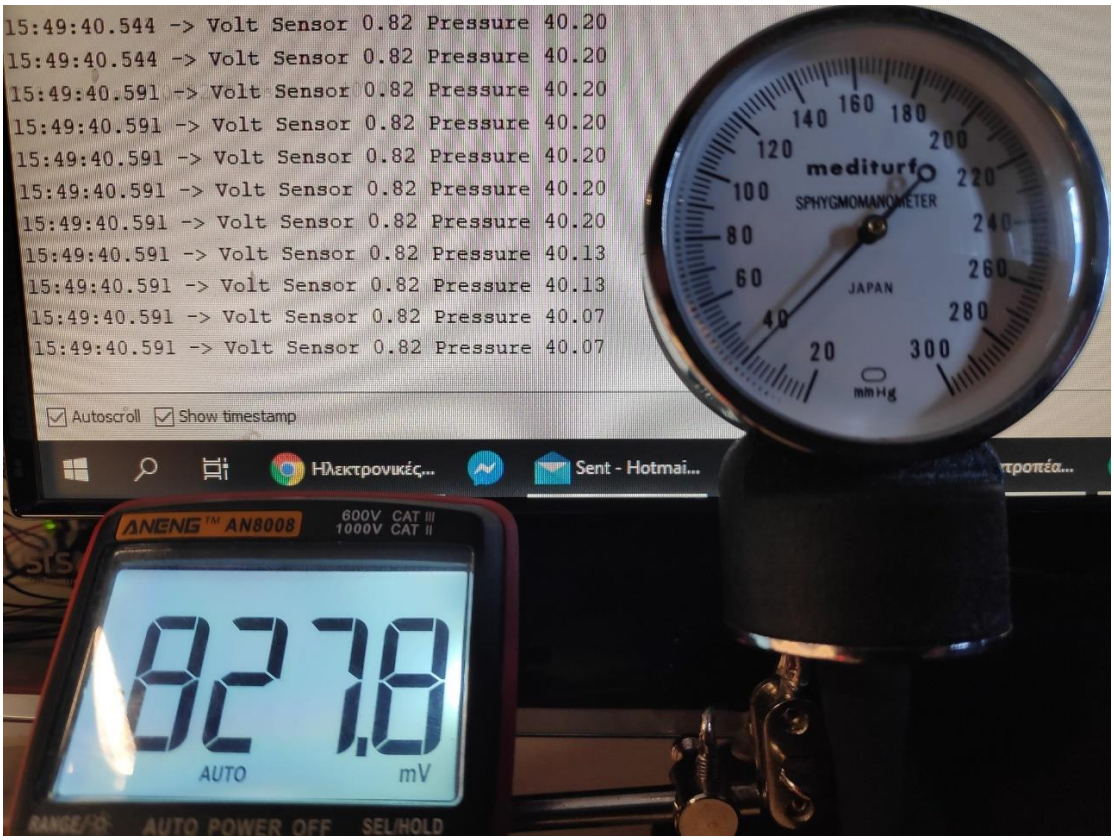
3.4 Αξιολόγηση αποτελεσμάτων

Το τελευταίο και εξίσου σημαντικό βήμα στη διαδικασία μια ακριβής μέτρησης είναι αυτή της αξιολόγησης. Για αυτόν το λόγο χρειάζεται ένα όργανο αναφοράς, στο οποίο μπορούν να βασιστούν και να συγκριθούν τα αποτελέσματα εξόδου του αισθητηρίου πίεσης.

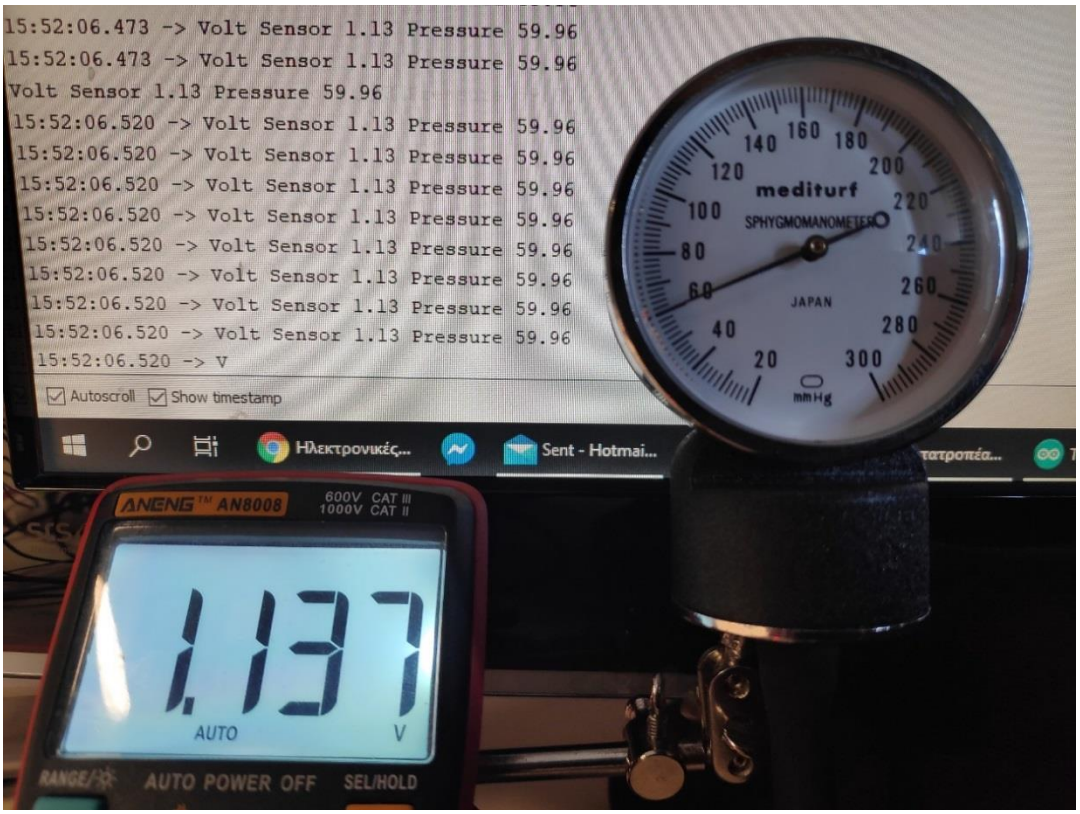
Το όργανο που χρησιμοποιήθηκε σε αυτό το πείραμα είναι το αναλογικό μανόμετρο. Το μανόμετρο είναι αριθμημένο από 0-300mmHg και περιέχει μια βελόνα/δείκτη που καθορίζει τη μετρούμενη πίεση.

Ελήφθησαν μετρήσεις με τις τιμές 40, 60 και 80 mmHg, όπως φαίνονται αντιστοίχως στα Σχ. 3.5, 3.6 και 3.7. Στην περιοχή των 40mmHg, το αισθητήριο μέτρησε από 40.07 έως 40.20 mmHg.

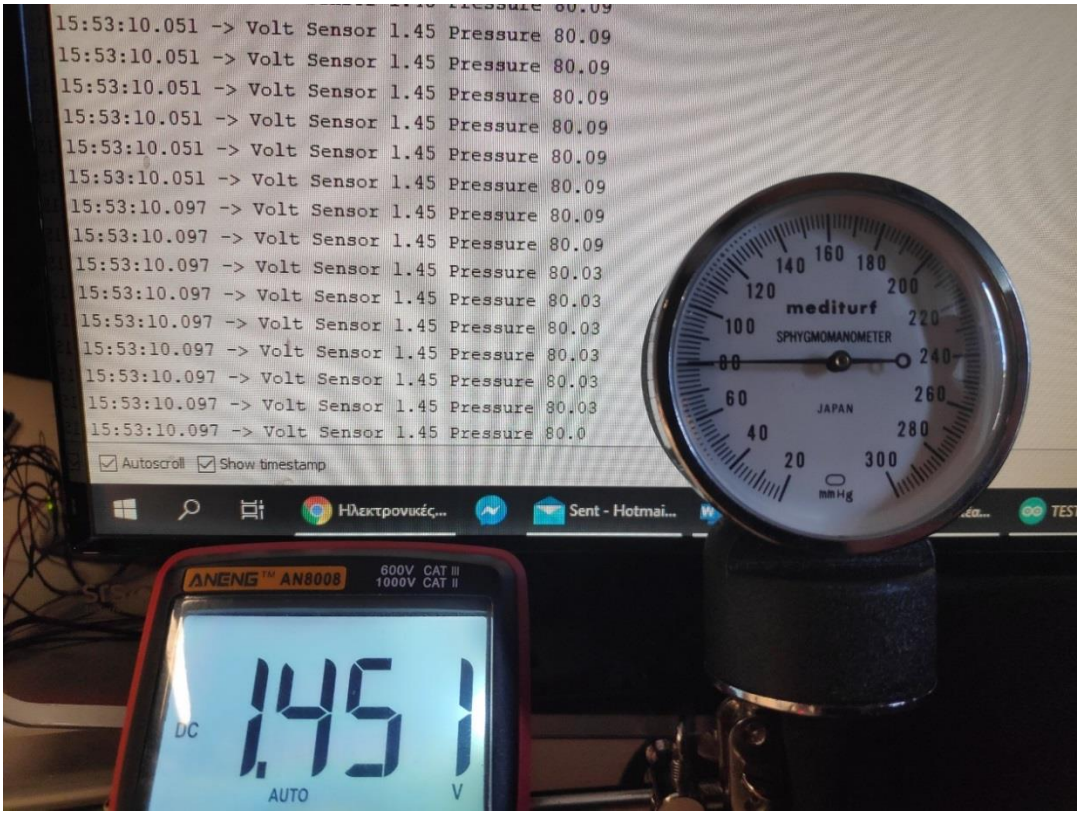
Στην περιοχή των 60 mmHg, η μέτρηση ήταν 59.96mmHg και στην περιοχή των 80 mmHg η τιμή ήταν 80.03 έως 80.09 mmHg. Τα αποτέλεσμα των μετρήσεων ότι πολύ κοντά στην τιμή αναφοράς και αυτό καθιστά όλη τη διαδικασία προσαρμογής, που εξετάζεται στο κεφάλαιο αυτό, επιτυχή.



Σχήμα 3.5 Μέτρηση τιμής μέσω του αισθητηρίου MPX505GP στην τιμή αναφοράς 40mmHg σύμφωνα με το μανόμετρο



Σχήμα 3.6 Μέτρηση τιμής μέσω του αισθητηρίου MPX505GP στην τιμή αναφοράς 60mmHg σύμφωνα με το μανόμετρο



Σχήμα 3.7 Μέτρηση τιμής μέσω του αισθητηρίου MPX505GP στην τιμή αναφοράς 80mmHg σύμφωνα με το μανόμετρο

ΚΕΦΑΛΑΙΟ 4

ΕΠΙΚΟΙΝΩΝΙΑ ΟΘΟΝΗΣ ΚΑΙ ΚΥΡΙΩΣ ΠΡΟΓΡΑΜΑ

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο εξετάζεται το πρόγραμμα που υλοποιήθηκε και τα γραφικά που επιλέχτηκαν ώστε να δώσουν στον χρήστη την οπτική ανατροφοδότηση που χρειάζεται για να πετύχει και να εκμάθει την διαφραγματική αναπνοή. Πραγματοποιήθηκε έρευνα για αγορά οθόνης ικανοποιητικού μεγέθους για αυτήν τη χρήση και τελικά επιλέχτηκε μια οθόνη 7 ιντσών TFT LCD από την κινέζικη εταιρεία BuyDisplay. Τα κριτήρια επιλογής της ήταν το κόστος, το μέγεθος ,το πρωτόκολλο επικοινωνίας καθώς και οι διαθέσιμοι οδηγοί ώστε να υλοποιηθούν οι απαιτήσεις της συσκευής. Η επικοινωνία επιτυγχάνεται μέσω SPI τεσσάρων αγωγών.

Το SPI πρωτόκολλο είναι μια σειριακή μέθοδος επικοινωνίας που ανακαλύφθηκε από την Motorola χρησιμοποιείται για κοντινές αποστάσεις και η χρήση του γίνεται κατά κόρον στα ενσωματωμένα συστήματα. Ένα μοναδικό πλεονέκτημα του SPI είναι το γεγονός ότι τα δεδομένα μπορούν να μεταφερθούν χωρίς διακοπή. Οποιοσδήποτε αριθμός bit μπορεί να σταλεί ή να ληφθεί σε συνεχή ροή. Οι συσκευές που επικοινωνούν μέσω SPI βρίσκονται σε σχέση master-slave. Ο master είναι η συσκευή ελέγχου (συνήθως ένας μικροελεγκτής), ενώ ο slave (συνήθως ένας αισθητήρας, μια οθόνη ή ένα τσιπ μνήμης) παίρνει οδηγίες από τον κύριο. Η απλούστερη διαμόρφωση του SPI το σύστημα ενός slave – master , αλλά ένας master μπορεί να ελέγξει περισσότερους από έναν slaves.

Το σήμα ρολογιού συγχρονίζει την έξοδο των bit δεδομένων από τον master με τη δειγματοληψία των bit από το slave. Ένα bit δεδομένων μεταφέρεται σε κάθε κύκλο ρολογιού, έτσι η ταχύτητα μεταφοράς δεδομένων καθορίζεται από τη συχνότητα του σήματος ρολογιού. Η επικοινωνία SPI ξεκινά πάντα από τον master αφού αυτός διαμορφώνει και παράγει το σήμα ρολογιού.

Κάθε πρωτόκολλο επικοινωνίας όπου οι συσκευές μοιράζονται ένα σήμα ρολογιού είναι γνωστό ως σύγχρονο. Το SPI είναι ένα σύγχρονο πρωτόκολλο επικοινωνίας. Υπάρχουν επίσης ασύγχρονες μέθοδοι που δεν χρησιμοποιούν σήμα ρολογιού. Για παράδειγμα, στην επικοινωνία UART, και οι δύο πλευρές έχουν ρυθμιστεί σε προεπιλεγμένο ρυθμό baud-rate (ρυθμός baud) που υπαγορεύει την ταχύτητα και το χρόνο μετάδοσης δεδομένων.

Το σήμα ρολογιού στο SPI μπορεί να τροποποιηθεί χρησιμοποιώντας τις ιδιότητες της πολικότητας του ρολογιού και της φάσης του ρολογιού. Η πολικότητα του ρολογιού μπορεί να ρυθμιστεί από τον master ώστε να επιτρέπεται η έξοδος bit και η δειγματοληψία είτε στην άνοδο είτε στην κάθοδο του κύκλου ρολογιού. Η φάση ρολογιού μπορεί να ρυθμιστεί ώστε η έξοδος και η δειγματοληψία να πραγματοποιούνται είτε στο πρώτο άκρο είτε στο δεύτερο άκρο του κύκλου ρολογιού, ανεξάρτητα από το αν ανεβαίνει ή πέφτει.

Οι τέσσερείς αγωγοί που χρησιμοποιεί αυτό το πρωτόκολλο είναι το MOSI , MISO , SCLK , CS.

- MOSI (Master Output Slave Input) : Η έξοδος του master της επικοινωνίας, με αυτό τον αγωγό μεταδίδονται τα δεδομένα από τον master στους slaves.

- MISO (Master Input Slave Output) : Η είσοδος του master της επικοινωνίας , με την χρήση αυτού του αγωγού επιτυγχάνεται η αντίθετη διαδικασία δηλαδή η μεταφορά των δεδομένων από έναν η περισσότερους σκλάβους στον master.

- Ένα σήμα SCLK ή Serial Clock (σειριακό ρολόι) εξέρχεται από την κύρια συσκευή. Το SCLK καθορίζει πότε θα διαβιβάζονται τα bits των δεδομένων.

- CS (Chip Select) : Μέσω αυτού του αγωγού προσδιορίζεται ποιος slave είναι ενεργός στον δίαυλο. Το CS επιτρέπει στο δίαυλο επικοινωνίας να χρησιμοποιεί πολλούς σκλάβους, όπως απαιτείται.

4.2 Ελεγκτής οθόνης

Ο ελεγκτής που χρησιμοποιεί η οθόνη που εξετάζεται είναι ο RA8875 κατασκευασμένος από την RAiO Technology. Πρόκειται για έναν μικροεπεξεργαστή που μετατρέπει τον κώδικα λογισμικού του πελάτη σε πληροφορίες που μπορεί να κατανοήσει η οθόνη LCD. Στη συνέχεια, η οθόνη LCD εμφανίζει γραφικά, χαρακτήρες, εικόνες και αριθμούς που μπορεί να δει ο τελικός χρήστης. Είναι ένα ισχυρό τσιπ προγράμματος οδήγησης TFT. Στο εσωτερικό του υπάρχει 768KB RAM, οπότε μπορεί να κάνει buffering στην οθόνη (και ανάλογα με το μέγεθος της οθόνης έχει επίσης δυνατότητα δύο επιπέδων γραφικών). Το τσιπ έχει μια σειρά από σχήματα επιταχυνόμενου υλικού όπως γραμμές, ορθογώνια, τρίγωνα, ελλείψεις, ενσωματωμένα και στρογγυλά ορθογώνια.

Κάποιες από τις βασικές λειτουργίες που υποστηρίζει φαίνονται παρακάτω :

- Βάθος χρώματος TFT : 256 χρώματα ή 65 χιλιάδες χρώματα
- Υποστηριζόμενο μέγεθος TFT :
    1. 800x480 Pixels ανάλυση σε λειτουργία δύο επιπέδων (2 Layers) με 256 χρώματα.
    2. 800x480 Pixels ανάλυση σε λειτουργία ενός επιπέδου (1 Layer) με 65 χιλιάδες χρώματα.
    3. 480x272 Pixels ανάλυση με δυνατότητα λειτουργείας δύο επιπέδων και ταυτόχρονη δυνατότητα βάθους χρώματος 65 χιλιάδων.
- Υποστηρίζει επικοινωνίες με την χρήση :
    1. Παράλληλο πρότυπο 8080/6800 σε μέγεθος διαύλου 8/16 bit.
    2. Σειριακά πρωτόκολλα I2C ή SPI τριών η τεσσάρων αγωγών.
- Δυνατότητα scrolling των γραφικών της οθόνης σε οριζόντια η κατακόρυφη κατεύθυνση
- Ενσωματωμένη μνήμη ROM μεγέθους 10KB για γραφικά χαρακτήρων με μέγεθος φόντου 8x16 Dots και υποστηρίζει σετ χαρακτήρων ISO/IEC 8859-1/2/3/4.
- Δυνατότητες μεγιστοποίησης φόντου χαρακτήρων X1,X2,X3,X4 φορές το κανονικό τους μέγεθος.
- Κάθετη περιστροφή χαρακτήρων
- Ενσωματωμένο ελεγκτή ανίχνευσης αφής

- Δυνατότητα Sleep Mode (Κατάσταση ύπνου) για χαμηλή κατανάλωση ενέργειας όταν η συσκευή δεν χρησιμοποιείται.
- Ενσωματωμένος ταλαντωτής κρυστάλλου με προγραμματιζόμενο PLL για παραμετροποίηση συχνότητας λειτουργίας.
- Τάση λειτουργίας από 3.0V έως 3.6V
- 2 Προγραμματιζόμενα PWM για την ρύθμιση του οπίσθιου φωτισμού της οθόνης ή και για κάποιον άλλο σκοπό.

Η οθόνη που χρησιμοποιείται σε αυτήν την εφαρμογή προγραμματίστηκε σε ανάλυση 800x480 σε λειτουργεία δύο επιπέδων και 256 συνδυασμών χρωμάτων. Έγινε χρήση των επιταχυμένων λειτουργιών σχεδίασης σχημάτων που προσφέρει ο ελεγκτής και η επικοινωνία με τον ESP32 επιτευχθεί με την χρήση του πρωτοκόλλου SPI τεσσάρων αγωγών. Από τα τέσσερα περιφερειακά SPI που διαθέτει το ESP32 , μόνο τα δύο είναι διαθέσιμα για τον χρήστη γιατί τα υπόλοιπα δύο χρησιμοποιούνται για την εσωτερική σύνδεση της μνήμης FLASH (SPI0 και SPI1). Η συγκεκριμένη επικοινωνία με τον RA8875 ελεγκτή έχει τα παρακάτω χαρακτηριστικά.

- Η επικοινωνία πραγματοποιείται στο περιφερειακό SPI3 (VSPI).
- Τα Pins του ESP32 που χρησιμοποιούνται για το SPI είναι το GPIO19 σαν MISO, το GPIO23 σαν MOSI, το GPIO18 σαν SCK και το GPIO16 σαν CS
- Το ρολόι χρονισμού παλμών λειτουργεί στην συχνότητα 8MHz (Ο RA8875 μπορεί να ανταποκριθεί και να είναι σταθερός μέχρι 12MHz , μετά από αυτήν την συχνότητα θεωρείται πάνω από τις προδιαγραφές του.
- Εκτός από τη ρύθμιση της συχνότητας ρολογιού, ο master πρέπει επίσης να διαμορφώσει την πολικότητα (CPOL) και τη φάση του ρολογιού (CPHA) σε σχέση με τον RA8875. Η έξοδος του ρολογιού σε κατάσταση αδράνειας βρίσκεται σε λογικό HIGH (CPOL=1) ενώ τα δεδομένα αλλάζουν στο κατερχόμενο επίπεδο του παλμού (failing edge) και γίνεται δειγματοληψία των δεδομένων στο ανερχόμενο επίπεδο (CPHA=1). Αυτός ο συνδυασμός πολικότητας και φάσης ρολογιού αντιστοιχίζεται σε (mode3) σύμφωνα με τον κατασκευαστή.

Ο RA8875 χρησιμοποιεί registers (καταχωρητές). Ο καταχωρητής είναι τύπος μικρής αλλά πολύ γρήγορης μνήμης που βρίσκεται μέσα στο τσιπ του ελεγκτή και του προσφέρει βελτίωση στην ταχύτητα εκτέλεσης των λειτουργειών που του ζητούνται μέσα από το SPI.

Σύμφωνα με τον Πίνακα 4.1 υπάρχουν τέσσερις διαφορετικοί κύκλοι πρόσβασης που μπορεί να καταλάβει ένας καταχωρητής και καθορίζεται η κάθε μια από πρώτο byte που θα σταλθεί. Το RS bit είναι το MSB (Most significant bit) και το αμέσως επόμενο το RW bit. Η Data Write πρόσβαση (Εγγραφή Δεδομένων) ισούται με το byte σε δεκαεξαδική μορφή 0x00. Η Data Read πρόσβαση (Διάβασμα Δεδομένων) ισούται με το byte 0x40. Η CMD Write πρόσβαση (Εγγραφή Εντολής) χρησιμοποιείται για να επιλέξεις τον καταχωρητή και ισούται με το byte 0x80. Η Status Read πρόσβαση (Διάβασμα κατάστασης) χρησιμοποιείται για να ξέρεις αν ο καταχωρητής είναι σε κατάσταση αδράνειας η σε κατάσταση busy (απασχολημένος) και ισούται με το byte 0xC0.

Πίνακας 4.1 Οι κύκλοι πρόσβασης καταχωρητή του ελεγκτή RA8875 και τα απαραίτητα bit επιλογής

RS	WR#	Access Cycle
0	0	Data Write
0	1	Data Read
1	0	CMD Write
1	1	Status Read

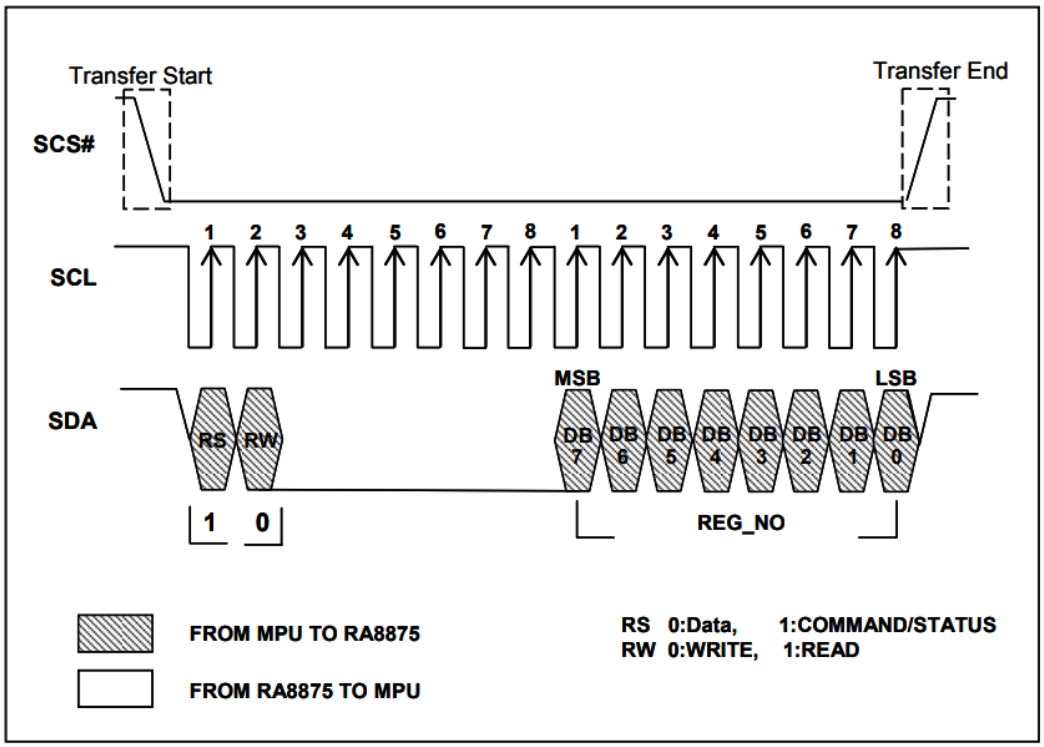
Η διαδικασία εγγραφής η ανάγνωσης καταχωρητή περιγράφεται στο datasheet του κατασκευαστή και ο master του SPI πρέπει να ακολουθήσει όλα τα χαρακτηριστικά βήματα που χρειάζεται για να επιτευχθεί αυτό τα οποία περιγράφονται στο Σχήμα 4.1 με παράδειγμα Εγγραφής Εντολής (CMD Write).

1. Βήμα 1ο : Έναρξη SPI Transcation.
2. Βήμα 2ο : Ρύθμιση του pin CS (Chip Select) σε κατάσταση λογικού μηδέν (LOW)
3. Βήμα 3ο : Αποστολή byte κύκλου πρόσβασης (στο παράδειγμα του Σχήματος 4.1 στέλνεται το 0x80 RS=1 RW=0).
4. Βήμα 4ο : Αποστολή byte διεύθυνσης καταχωρητή.
5. Βήμα 5ο : Ρύθμιση του pin CS (Chip Select) σε κατάσταση λογικού ένα (HIGH)
6. Βήμα 6ο : Λήξη SPI Transcation.

Με τα παραπάνω βήματα έχει επιλεγεί ο συγκεκριμένος καταχωρητής, το επόμενο βήμα για μια ολοκληρωμένη διαδικασία είναι είτε το διάβασμα των δεδομένων του , είτε το γράψιμο δεδομένων. Κατά παρόμοιο τρόπο με παραπάνω βήματα για την διαδικασία γράψιματος δεδομένων με διαφορά στο Βήμα 3ο το byte που αποστέλεται είναι το 0x00 και στο Βήμα 4ο γίνεται αποστολή του byte των δεδομένων.

Στην πραγματικότητα σε πολλές λειτουργίες ειδικά σε αυτές που έχουν μέγαλο όγκο εργασίας (π.χ σχεδίαση ένα μεγάλου παραλληλογράμου 300x200) χρείαζεται πέρα από την επιλογή καταχωρητή και γράψιμο δεδομένων σε αυτών και μια ακόμη διαδικασία. Είναι απαραίτητο να γίνει και ένα συνεχής έλεγχο για το αν η διαδικασία έχει ολοκληρωθεί, διότι δεν μπορείς χρησιμοποιήσεις άλλη λειτουργεία ενώ παράλληλα εκτελεί κάτι άλλο ο RA8875.

Οπότε πέρα από τα παραπάνω βήματα (CMD Write και Data Write) χρησιμοποιείται και ένα (Status Read) για να δηλώσει πότε ο RA8875 έχει τελειώσει την ολοκλήρωση του γραφίματος, στο συγκεκριμένο παράδειγμα την σχεδίαση του παραλληλογράμου 300x200 pixels.



Σχήμα 4.1 Αναλυτική απεικόνιση επικοινωνίας του κεντρικού επεξεργαστή master (ESP32) με τον ελεγκτή slave (RA8875) με σκοπό την εγγραφή εντολής σε έναν καταχωρητή

4.3 Γραφικά συσκευής

Ένα από τα πια σημαντικά σημεία μια συσκευής/ιστοσελίδας/προγράμματος είναι ο σχεδιασμός διεπαφής χρήστη (UI). Πρόκειται για τον προγραμματισμό της εμφάνισης των πραγμάτων, με σκοπό τη διευκόλυνση της χρηστικότητας και τη βελτίωση της εμπειρίας του χρήστη. Η εμφάνιση των πραγμάτων επηρεάζεται από το χρώματα που θα επιλεγούν, τις γραμματοσειρές και το μέγεθός τους, τα σχήματα τις εικόνες-εικονίδια που θα χρησιμοποιηθούν για να δώσουν μια πιο φιλική εικόνα προς τον χρήστη. Βασικό κριτήριο οπότε των σχημάτων που επιλέχτηκαν και των γραφικών που δημιουργήθηκαν σε αυτήν την συσκευή είναι καθαρή απεικόνιση των δεδομένων και η εύκολή κατανόηση τους. Παρακάτω αναλύονται τα γραφικά που επιλέχθηκαν για τις ανάγκες της μέτρησης της διαφραγματικής αναπνοής και της μέτρησης της προσπάθειας εκτέλεσής της.

Η οθόνη χρησιμοποιεί την δυνατότητα των δυο επιπέδων που προσφέρει ο RA8875. Το επίπεδο 1 όπως φαίνεται στο Σχήμα 4.2 περιέχει όλα τα γραφικά που είναι στατικά δηλαδή κατά την διάρκεια της μέτρησης δεν μεταβάλλονται. Σχηματίστηκε ο πίνακας υποδοχής των τιμών των προσπαθειών, η γραφική παράσταση πίεσης/χρόνου και το περίγραμμα του εικονιδίου της μπαταρίας. Έπειτα σχεδιάστηκαν όλα τα απαραίτητα μηνύματα στην οθόνη , όπως οι τίτλοι των δύο αξόνων (κατακόρυφος άξονας : Pressure σε mmHg και οριζόντιος άξονας : Time σε δευτερόλεπτα s). Το μήνυμα (Mode) που σηματοδοτεί την επιλογή δυσκολίας για την εκτέλεση της διαφραγματικής αναπνοής. Το μήνυμα (Current Value) που περιγράφει την τιμή της πίεσης σε πραγματικό χρόνο.

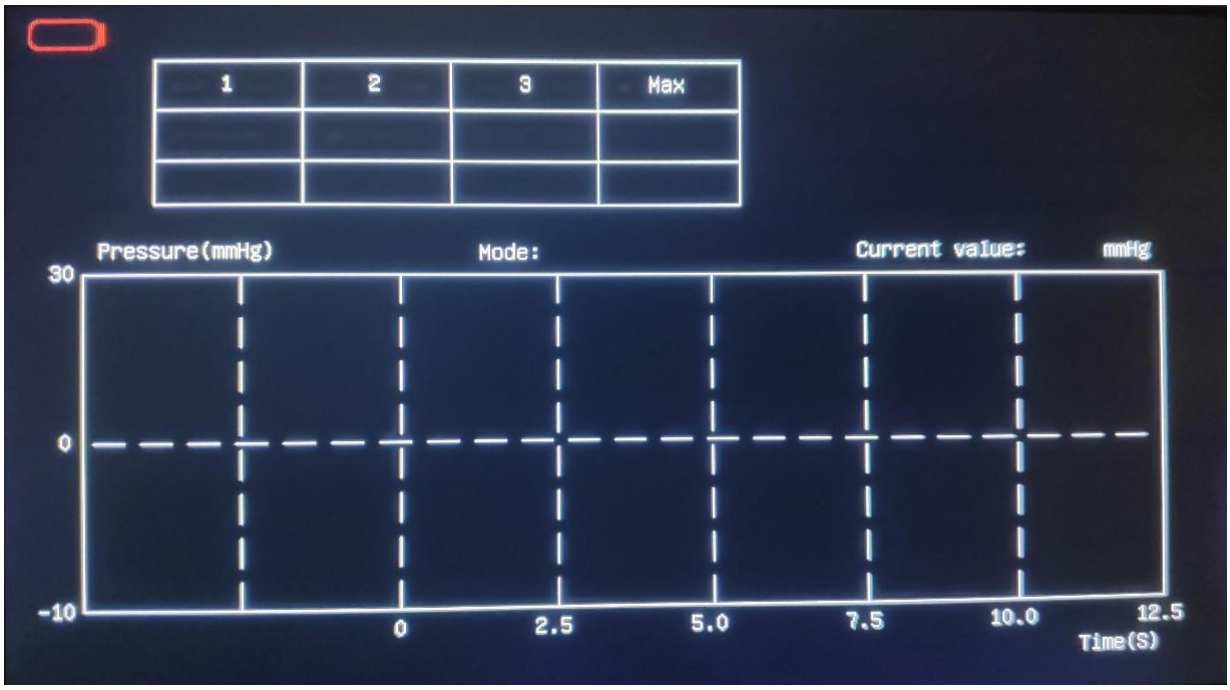
Για να σχηματιστεί μια γραμμή ή ένα παραλληλόγραμμο αρκεί να δοθούν οι συντεταγμένες στους καταχωρητές. Οι συντεταγμένες καθορίζονται από ένα σημείο αρχής και ένα σημείο τέλους . Το κάθε σημείο έχει ένα ζεύγος τιμών *X*, *Y* όπου *X* η θέση στο οριζόντιο επίπεδο και *Y* η θέση στο κάθετο

επίπεδο. Η ανάλυση της οθόνης που επιλέχτηκε είναι 800x480 που σημαίνει ότι το επίπεδο X έχει 800 διαφορετικά σημεία (pixel) ενώ το κάθετο επίπεδο 480. Η συντεταγμένη των X οπότε παίρνει τιμές από 0 έως 800, ενώ η συντεταγμένη *Y* παίρνει τιμές από 0 έως 480. Μια ολοκληρωμένη συνάρτηση οπότε χρειάζεται τέσσερις συντεταγμένες οι οποίες και οι δύο ξεπερνάνε τον αριθμό 255 που μπορεί να εκφραστεί με ένα byte , οπότε χρειάζονται δύο byte για να περιγράψουν κάθε συντεταγμένη.

- REG[91h] Draw Line/Square Horizontal Start Address Register0 (DLHSR0) : Δέχεται το πρώτο byte του αριθμού τής συντεταγμένης *X* του αρχικού σημείου της ευθείας/παραλληλογράμμου.
- REG[92h] Draw Line/Square Horizontal Start Address Register1 (DLHSR1) : Δέχεται το δεύτερο byte του αριθμού τής συντεταγμένης *X* του αρχικού σημείου της ευθείας/παραλληλογράμμου.
- REG[93h] Draw Line/Square Vertical Start Address Register0 (DLVSR0) : Δέχεται το πρώτο byte της του αριθμού της συντεταγμένης *Y* του αρχικού σημείου της ευθείας/παραλληλογράμμου.
- REG[94h] Draw Line/Square Vertical Start Address Register1 (DLVSR1) : Δέχεται το δεύτερο byte του αριθμού της συντεταγμένης *Y* του αρχικού σημείου της ευθείας/παραλληλογράμμου.
- REG[95h] Draw Line/Square Horizontal End Address Register0 (DLHER0) : Δέχεται το πρώτο byte του αριθμού τής συντεταγμένης *X* του τελικού σημείου της ευθείας/παραλληλογράμμου.
- REG[96h] Draw Line/Square Horizontal End Address Register1 (DLHER1) : Δέχεται το δεύτερο byte του αριθμού τής συντεταγμένης *X* του τελικού σημείου της ευθείας/παραλληλογράμμου.
- REG[97h] Draw Line/Square Vertical End Address Register0 (DLVER0) : Δέχεται το πρώτο byte της του αριθμού της συντεταγμένης *Y* του τελικού σημείου της ευθείας/παραλληλογράμμου.
- REG[98h] Draw Line/Square Vertical End Address Register1 (DLVER1) : Δέχεται το δεύτερο byte του αριθμού της συντεταγμένης *Y* του αρχικού σημείου της ευθείας/παραλληλογράμμου.

Με την χρήση των παραπάνω καταχωριτών REG[91-98h] στέλνονται στον RA8875 τα δεδομένα των συντεταγμένων. Εφόσον έχουν εισαχθεί όλα τα δεδομένα αρχής και τέλους του σχήματος ευθείας ή παραλληλογράμμου, το επόμενο βήμα είναι η επιλογή του μπροστινού χρώματος (Foreground color). Με την χρήση των τριών καταχωριτών για τα τρία βασικά χρώματα RGB κόκκινο, πράσινο, μπλε που μπορεί να επιτευχθούν 256 συνδυασμοί. Τα 8 bits διαιρούνται σε 3 bits που περιγράφουν το κόκκινο χρώμα , 3 bits για το πράσινο χρώμα και 2 bits για το μπλε.

- REG[65h] Foreground Color Register 2 (FGCR2) : Καταχωρητής για την επιλογή του μπλε χρώματος στον τελικό συνδυασμό. Τα bits που γίνονται set είναι τα 2 πρώτα bit[1:0].
- REG[64h] Foreground Color Register 1 (FGCR1) : Καταχωρητής για την επιλογή του πράσινου χρώματος στον τελικό συνδυασμό. Τα bits που γίνονται set είναι τα 3 πρώτα bit[2:0].
- REG[63h] Foreground Color Register 0 (FGCR0) : Καταχωρητής για την επιλογή του κόκκινου χρώματος στον τελικό συνδυασμό. Τα bits που γίνονται set είναι τα τρία πρώτα bit[2:0].



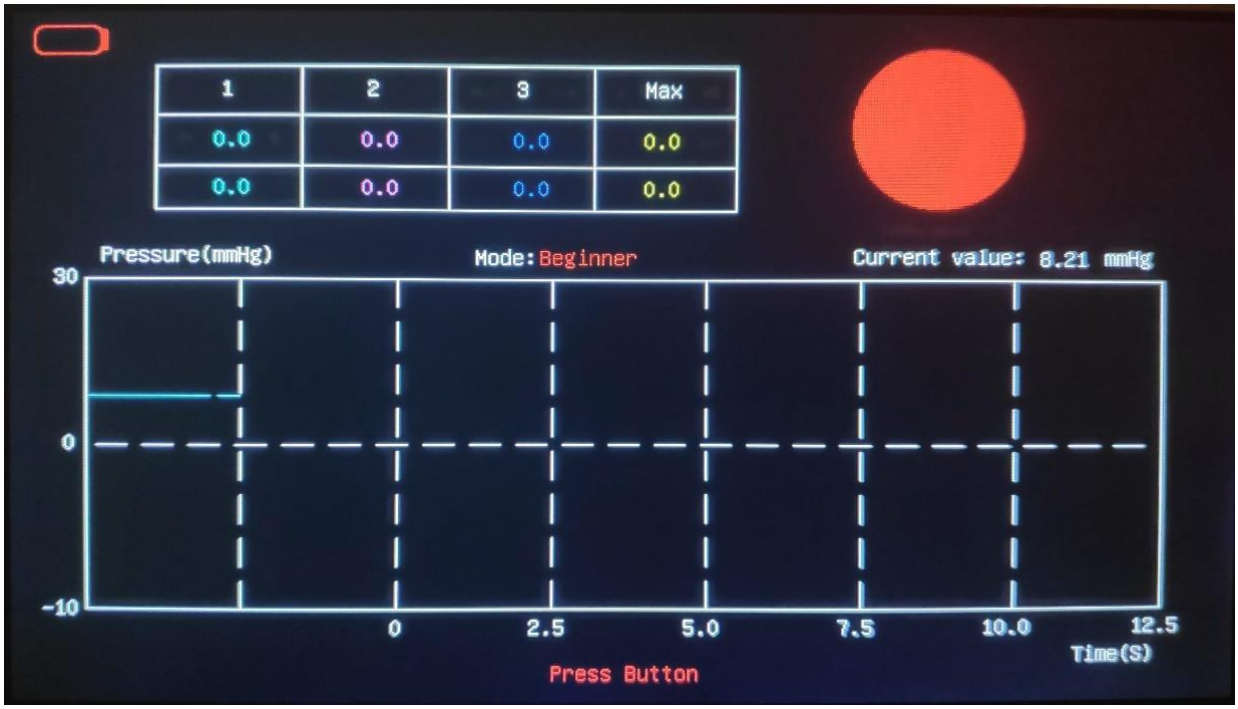
Σχήμα 4.2 Στατικά γραφικά οθόνης (επίπεδο 1) για την μέτρηση της διαφραγματικής αναπνοής σε οθόνη TFT 7 ιντσών με την χρήση του ελεγκτή οδήγησης RA8875

Το τελευταίο βήμα για την ολοκληρωμένη σχεδίαση του σχήματος ευθείας/παραλληλογράμμου είναι η επικοινωνία με τον καταχωρητή (REG[90h] Draw Line/Circle/Square Control Register (DCR)). Για την επιλογή του σχήματος γίνονται set το πρώτο bit 0 που πρέπει να έχει τιμή 0 και το bit 4 που θέτοντας ως μηδέν γίνεται η επιλογή σχεδίασης γραμμής ή θέτοντάς το 1 για σχεδίαση παραλληλογράμμου. Για την δυνατότητα γεμίσματος του σχήματος με το χρώμα που επιλέχθηκε αρκεί να γίνει set το bit 5 ως 1. Θέτοντας το bit 7 ως 1 αρχίζει η διαδικασία σχεδίασης στην οθόνη TFT. Όπως αναφέρθηκε και στο κεφάλαιο 4.2 , θα πρέπει το ESP32 να κάνει συνεχής διάβασμα (polling) τον καταχωρητή για να σηματοδοτήσει το τέλος της διαδικασίας, μόνο έτσι ο RA8875 μπορεί να εκτελέσει επόμενες λειτουργίες. Αυτή την δυνατότητα εμφάνισης του τέλους , την εμφανίζει ο συγκεκριμένος καταχωρητής (κύκλος πρόσβασης Data Read) στο bit 7. Όταν η λειτουργία είναι σε εξέλιξή αυτό το bit έχει τιμή 1 , ενώ όταν η λειτουργία ολοκληρώνεται έχει τιμή 0. Όλα τα σχήματα στο επίπεδο 1 σχηματιστήκαν με την βοήθεια των συναρτήσεων γραμμών και παραλληλογράμμου όπως αναλύθηκαν παραπάνω.

Το επίπεδο 2 όπως φαίνεται στο Σχήμα 4.3 περιέχει την συνολική εικόνα της οθόνης δηλαδή τα στατικά στοιχεία σε συνδυασμό με τα δυναμικά στοιχεία δηλαδή όλα αυτά που αλλάζουν κατά την διάρκεια της μέτρησης.

Τα κύρια δυναμικά στοιχεία της οθόνης είναι η το σήμα της πίεσης που σχηματίζεται πάνω στην γραφική παράσταση πίεσης/χρόνου σε μορφή γραμμής και η μπάλα μέτρησης που κατάσταση ισορροπίας σχηματίζεται σαν κύκλος και μετά αλλάζει σε ελλειπτική μορφή. Η έλλειψη για να σχηματιστεί χρειάζεται τέσσερις συντεταγμένες και χρώμα με τον ίδιο τρόπο που περιεγράφηκε για την ευθεία η το παραλληλόγραμμο. Οι πρώτες δύο συντεταγμένες δηλώνουν το κέντρο της έλλειψης , και οι άλλες δύο το μήκος ανοίγματος κατά *X* και κατά *Y* ως προς το σημείο αναφοράς που είναι το κέντρο. Οι καταχωρητές REG[A1-8h] χρησιμοποιούνται για την επιλογή των συντεταγμένων του σχήματος.

- REG[A5h] Draw Ellipse/Circle Square Center Horizontal Address Register0 (DEHR0) : Δέχεται το πρώτο byte του αριθμού τής συντεταγμένης *X* του κέντρου της έλλειψης.
- REG[A6h] Draw Ellipse/Circle Square Center Horizontal Address Register1 (DEHR1) : Δέχεται το δεύτερο byte του αριθμού τής συντεταγμένης *X* του κέντρου της έλλειψης.
- REG[A7h] Draw Ellipse/Circle Square Center Vertical Address Register0 (DEVR0) : Δέχεται το πρώτο byte του αριθμού τής συντεταγμένης *Y* του κέντρου της έλλειψης.
- REG[A8h] Draw Ellipse/Circle Square Center Vertical Address Register1 (DEVR1) : Δέχεται το δεύτερο byte του αριθμού της συντεταγμένης *Y* του κέντρου της έλλειψης.
- REG[A1h] Draw Ellipse/Circle Square Long axis Setting Register (ELL_A0) : Δέχεται το πρώτο byte του αριθμού που δείχνει το μήκος ανοίγματος κατά *X από το κέντρο της έλλειψης*.
- REG[A2h] Draw Ellipse/Circle Square Long axis Setting Register (ELL_A1) : : Δέχεται το δεύτερο byte του αριθμού που δείχνει το μήκος ανοίγματος κατά *X από το κέντρο της έλλειψης*.
- REG[A3h] Draw Ellipse/Circle Square Short axis Setting Register (ELL_B0) : : Δέχεται το πρώτο byte του αριθμού που δείχνει το μήκος ανοίγματος κατά *Y από το κέντρο της έλλειψης*.
- REG[A4h] Draw Ellipse/Circle Square Short axis Setting Register (ELL_B1) : Δέχεται το δεύτερο byte του αριθμού που δείχνει το μήκος ανοίγματος κατά *Y από το κέντρο της έλλειψης*.



Σχήμα 4.3 Η συνολική απεικόνιση των γραφικών της οθόνης για την μέτρηση της διαφραγματικής αναπνοής σε οθόνη TFT 7 ιντσών με την χρήση του ελεγκτή οδήγησης RA8875.

Έπειτα κατά όμοιο τρόπο γίνεται η επιλογή του μπροστινού χρώματος RGB με την χρήση των καταχωριτών REG[63-65h] όπως αναφέρθηκε και στην ευθεία/παραλληλόγραμμο. Το τελευταίο βήμα είναι το setup του καταχωρητή (REG[A0h] Draw Ellipse/Ellipse Curve/Circle Square Control Register). Τα bit 4 και 5 παίρνουν την τιμή 0 ώστε να επιλεγεί το σχήμα της έλλειψης και οχι τα άλλα δύο που προσφέρει αυτός ο καταχωρητής (κυρτή έλλειψη ή στρογγυλοποιημένο τετράγωνο). Ρυθμίζοντας το bit 6 γίνεται επιλογή αν η έλλειψη θα σχεδιαστεί με γέμισμα χρώματος στο εσωτερικός της ή όχι. Για τον σχεδιασμό της μπάλας μέτρησης χρειάζεται το γέμισμα χρώματος άρα αυτό το bit παίρνει την τιμή 1. Η διαδικασία σχεδίασης ξεκινάει θέτοντας το bit 7 ως 1. Το ESP32 πραγματοποιεί διάβασμα του καταχωρητή μέχρι το bit 7 πάρει την τιμή 0 , που σηματοδοτεί το τέλος της σχεδίαση.

Τα δυναμικά κείμενα που χρησιμοποιούνται στην συσκευή είναι η τιμή του ‘Mode:’ που στο παράδειγμα έχει τιμή ‘Beginner’, το μήνυμα ‘Press Button’, οι τιμές του πίνακα προσπαθειών που στο παράδειγμα είναι μηδενικές και η αριθμητική τιμή της πίεσης σε πραγματικό χρόνο που αναγράφεται δίπλα από το μήνυμα ‘Current value:’ που στο παράδειγμα είναι ‘8.21’. Για την δημιουργία κειμένου για την απεικόνιση μνημάτων στην οθόνη χρειάζεται κάθε φορά να επιλέγονται οι συντεταγμένες σημείου που γίνεται η εγγραφή. Οι παρακάτω καταχωρητές χρησιμοποιούνται για αυτόν τον σκοπό.

- REG[2Ah] Font Write Cursor Horizontal Position Register 0 (F_CURXL) : Δέχεται το πρώτο byte του αριθμού τής συντεταγμένης *X* του σημείου αναφοράς που τυπώνεται το κείμενο.
- REG[2Bh] Font Write Cursor Horizontal Position Register 1 (F_CURXH) : Δέχεται το δεύτερο byte του αριθμού τής συντεταγμένης *X* του σημείου αναφοράς που τυπώνεται το κείμενο.
- REG[2Ch] Font Write Cursor Vertical Position Register 0 (F_CURYL) : Δέχεται το πρώτο byte του αριθμού τής συντεταγμένης *Y* του σημείου αναφοράς που τυπώνεται το κείμενο.
- REG[2Dh] Font Write Cursor Vertical Position Register 1 (F_CURYH) : Δέχεται το δεύτερο byte του αριθμού τής συντεταγμένης *Y* του σημείου αναφοράς που τυπώνεται το κείμενο.

Έπειτα στέλνεται ως data κάθε χαρακτήρας ξεχωριστά της λέξης-μηνύματος που γίνεται η απεικόνιση στην οθόνη στον καταχωρητή (REG[02h] Memory Read/Write Command (MRWC)) ολοκληρώνοντας την διαδικασία.

4.4 Ανάλυση λειτουργίας συσκευής

Στην ενότητα 4.3 αναλύθηκε ο τρόπος προγραμματισμού της οθόνης σε επίπεδο δημιουργίας σχημάτων και μηνυμάτων μέσω των καταχωριτών που προσφέρει ο RA8875. Στο παρόν κεφάλαιο αναλύεται όλο το πρόγραμμα μέτρησης και απεικόνισης της διαφραγματικής αναπνοής. Το πρόγραμμα χωρίζεται σε δύο μέρη στην κατάσταση αναμονής και την κατάσταση μέτρησης.

Στην κατάσταση αναμονής το σήμα της πίεσης σχηματίζεται μέχρι την πρώτη κάθετη διακεκομμένη γραμμή και έπειτα ξεκινάει πάλι από την αρχή. Η μπάλα μέτρησης δεν ανταποκρίνεται σε αυτήν την κατάσταση στις μεταβολές της πίεσης και παραμένει σαν σταθερός κύκλος. Το μήνυμα στην κάτω κεντρική πλευρά της οθόνης που εμφανίζεται ‘Press Button’ είναι σταθερό και δηλώνει στον χρήστη ότι για να ξεκινήσει η διαδικασία μέτρησης θα πρέπει να πατηθεί το κουμπί έναρξης μέτρησης.

Για να ξεκινήσει η διαδικασία μέτρησης το σήμα πίεσης θα πρέπει θεωρητικά να βρίσκεται στο 0 (πίεση αναφοράς 60mmHg), ωστόσο επιτρέπεται μια διακύμανση μεταξύ ± 3mmHg που μπορεί να

οφείλεται σε μικρές κινήσεις της κοιλιάς λόγω της αναπνοής. Εφόσον το σήμα βρίσκεται στα όρια που αναφέρθηκαν και πατηθεί το κουμπί έναρξης ξεκινάει η διαδικασία. Δίνεται στον χρήστη 1,5 δευτερόλεπτο που αποτελεί το χρόνο προετοιμασίας για την εκτέλεση της διαφραγματικής αναπνοής. Σε αυτό το σημείο εμφανίζεται νέο μήνυμα 'Relax' με κίτρινο χρώμα στην οθόνη στην θέση του 'Press Button' και η μπάλα μέτρησης αλλάζει σε κίτρινο χρώμα. Αν ο χρήστης δεν είναι ήρεμος στον χρόνο προετοιμασίας που του δίνεται δηλαδή πάει να εκτελέσει την διαφραγματική αναπνοή λανθασμένα ο χρόνος αυτός αυξάνεται και παραμένει αυτή η κατάσταση μέχρι το σήμα της πίεσης να σταθεροποιηθεί. Σε αυτήν την φάση το πρόγραμμα υπολογίζει το μέσο όρο πίεσης που δέχεται η ζώνη ως αναφορά για την μέτρηση. Μόλις περάσει η κατάσταση προετοιμασίας το πρόγραμμα μπαίνει στην κατάσταση μέτρησης. Την θέση του μηνύματος 'Relax' παίρνει το μήνυμα 'Measuring' με πράσινη γραμματοσειρά για να δώσει στον χρήστη το μήνυμα να ξεκινήσει την εκτέλεση της διαφραγματικής αναπνοής. Ο μέγιστος χρόνος προσπάθειας είναι 12.5 δευτερόλεπτα και μέσα σε αυτόν πρέπει να γίνει η εκτέλεση μόνο μια φορά. Η μπάλα φουσκώματος αλλάζει στο πράσινο χρώμα από το κίτρινο και αλλάζει σχήμα αναλόγως την μεταβολή της πίεσης. Πιο συγκεκριμένα κατά την αναπνοή ο ασθενής φουσκώνει το στομάχι του και εισπνέει αέρα ανεβάζοντας την πίεση στην φουσκωτή περιοχή της ζώνης, παράλληλα η μπάλα μέτρησης με την άνοδο της πίεσης που ασκείται στο αισθητήριο διαστέλλεται παίρνοντας μια πιο ελλειπτική μορφή. Με παρόμοιο τρόπο κατά την εκπνοή ο ασθενής μαζεύει το στομάχι του και παράλληλα η μπάλα συρρικνώνεται. Για να θεωρηθεί μια προσπάθεια πετυχημένη αρκεί να εκπληρωθούν δύο συνθήκες. Η πίεση που θα ασκηθεί στην ζώνη να είναι κατά 3mmHg (threshold) μεγαλύτερη της πίεση αναφοράς που υπολογίστηκε (με αυτόν τον τρόπο γίνεται ο υπολογισμός της εισπνοής κατά την διαφραγματική αναπνοή), ταυτόχρονα η ελάχιστη πίεση μέτρησης να είναι μικρότερη από την πίεση αναφοράς (με αυτόν τον τρόπο υπολογίζεται η εκπνοή κατά την διαφραγματική αναπνοή). Αν αυτές οι δύο συνθήκες δεν εκπληρωθούν η προσπάθεια ακυρώνεται και στον πίνακα προσπαθειών στην στήλη κάτω από τον αριθμό προσπάθειας (1 2 ή 3) γράφεται ο αριθμός -1 που σημαίνει σφάλμα μέτρησης. Αν η μέτρηση εκπληρωθεί σωστά , τότε στην στήλη με τον αριθμό της προσπάθειας του πίνακα συμπληρώνεται στο πάνω κελί η μεγίστη διακύμανση της πίεσης που προκαλείται κατά την διαφραγματική αναπνοή ενώ στο κάτω η διακύμανση που προκαλείται λόγω της εισπνοής. Κατά τη διάρκεια της διαδικασίας μέτρησης αποθηκεύονται σε έναν πίνακα όλες οι τιμές πίεσης που σχηματίζουν την κυματομορφή στην οθόνη. Αν η προσπάθεια θεωρηθεί επιτυχημένη τότε γυρνάει η οθόνη στο επίπεδο 1 (επίπεδο στατικών σχημάτων) και σχηματίζεται η επιτυχημένη κυματομορφή διαφραγματικής αναπνοής με τις αποθηκευμένες τιμές του πίνακα και μόλις τελειώσει γυρνάει ξανά στο επίπεδο 2. Σε αυτό το σημείο στέλνεται ο πίνακας μέσω WiFi στον ηλεκτρονικό υπολογιστή για απεικόνιση των δεδομένων στην web εφαρμογή, όλη αυτή η διαδικασία αναλύεται λεπτομερώς σε επόμενο κεφάλαιο.

Η επανεκκίνηση της συσκευής (soft reset) μπορεί να προκληθεί με το κουμπί έναρξης προσπάθειας , αν αυτό πατηθεί παρατεταμένα για 6 δευτερόλεπτα είτε πατώντας το μετά την ολοκλήρωση των τριών επιτυχημένων προσπαθειών. Κατά τη διαδικασία του soft reset η οθόνη καθαρίζεται πλήρως από όλα τα γραφικά στα δύο επίπεδα. Ξανασχεδιάζονται όλα τα στατικά σχήματα , καθαρίζεται ο πίνακας προσπαθειών και γυρνάει ξανά στο επίπεδο 2 περιμένοντας το ξεκίνημα ενός νέου κύκλου προσπαθειών μέτρησης διαφραγματικής αναπνοής.

Βασικό σημείο που αξίζει να αναφερθεί είναι αυτό του σβησίματος ενός τμήματος της οθόνης. Για την επίτευξη αυτού του στόχου χρησιμοποιείται ο σχεδιασμός ενός παραλληλογράμμου με τον τρόπο που αναφέρθηκε παραπάνω, με διαστάσεις καθορισμένες από τον χρήστη που εξαρτώνται από

το μέγεθος της περιοχής προς σβήσιμο και με γέμισμα μαύρου χρώματος (όπως το οπίσθιο χρώμα της οθόνης). Ανάγκη σβησίματος περιορισμένου χώρου που δεν εξυπηρετεί αυτός ο τρόπος απαιτείται στην μπάλα μέτρησης κατά την σμίκρυνση της. Η λύση που δόθηκε είναι ότι όταν μικραίνει τυπώνεται ένα περίγραμμα έλλειψης (έλλειψη χωρίς γέμισμα χρώματος) με τις προηγούμενες συντεταγμένες.

Το πρόγραμμα διαθέτει δυνατότητα αλλαγής 'Mode' , που εκφράζει την κατάρτιση του χρήστη στην εκτέλεση της διαφραγματικής αναπνοής. Αν ο χρήστης είναι πρωτάρης 'Beginner' , δηλαδή είναι στις πρώτες προσπάθειες εκτέλεσης διαφραγματικής αναπνοής το πιο πιθανό είναι να μην μπορεί να χτυπήσει το ανώτατο όριο 30mmHg διακύμανση κατά την αναπνοή. Σε αντίθετη περίπτωση , υπάρχει δυνατότητα εναλλαγής 'Mode' σε κατάρτιση 'Advanced' προχωρημένου σταδίου όπου αλλάζει το ανώτατο όριο 60mmHg και κατώτατο -15mmHg από το προηγούμενο κατώτατο στα -10mmHg. Για την δυνατότητα αλλαγής 'Mode' θα πρέπει να πατηθεί το κουμπί εκκίνησης προσπάθειας παρατεταμένα για 2 δευτερόλεπτα.

ΚΕΦΑΛΑΙΟ 5

WEB ΕΦΑΡΜΟΓΗ

5.1 Εισαγωγή

Σε αυτό το κεφάλαιο αναλύεται η web εφαρμογή που υλοποιήθηκε για την συσκευή και όλες οι τεχνολογίες που χρησιμοποιήθηκαν για αυτόν τον σκοπό. Βασικός στόχος ήταν να διαμορφωθεί ένα ολοκληρωμένο σύστημα ασύρματης μεταφοράς δεδομένων και αποθήκευση τους σε έναν ηλεκτρονικό υπολογιστή. Με αυτόν τον τρόπο η συσκευή αγγίζει εμπορικές προδιαγραφές σε επίπεδο χρηστικότητας. Η ευχρηστία της εφαρμογής παίζει καθοριστικό ρόλο στην επιλογή της και στην άνεση του τελικού χρήστη, στην προκειμένη περίπτωση ενός φυσικοθεραπευτή. Η αξιοπιστία, η αποδοτικότητα και η φιλικότητα προς τον χρήστη έπαιξαν επίσης καθοριστικό ρόλο ως κριτήρια δημιουργίας της web εφαρμογής.

5.2 Διαμόρφωση επικοινωνίας

Είναι προφανής η ανάγκη ύπαρξης δικτύου στο οποίο να συνδέονται και να επικοινωνούν όλα τα στοιχεία που το απαρτίζουν μεταξύ τους. Το IP δίκτυο χρησιμοποιεί την τεχνολογία μεταγωγής πακέτων για τη μεταφορά των δεδομένων. Τα δεδομένα κόβονται σε κομμάτια που ονομάζονται πακέτα και σε κάθε πακέτο μπαίνει μια “επικεφαλίδα” με τις διευθύνσεις του υπολογιστή - αποστολέα και του υπολογιστή - παραλήπτη. Σημειώνεται ότι σε κάθε υπολογιστική μονάδα του τοπικού δικτύου αντιστοιχίζεται μία διεύθυνση που ονομάζεται διεύθυνση IP.

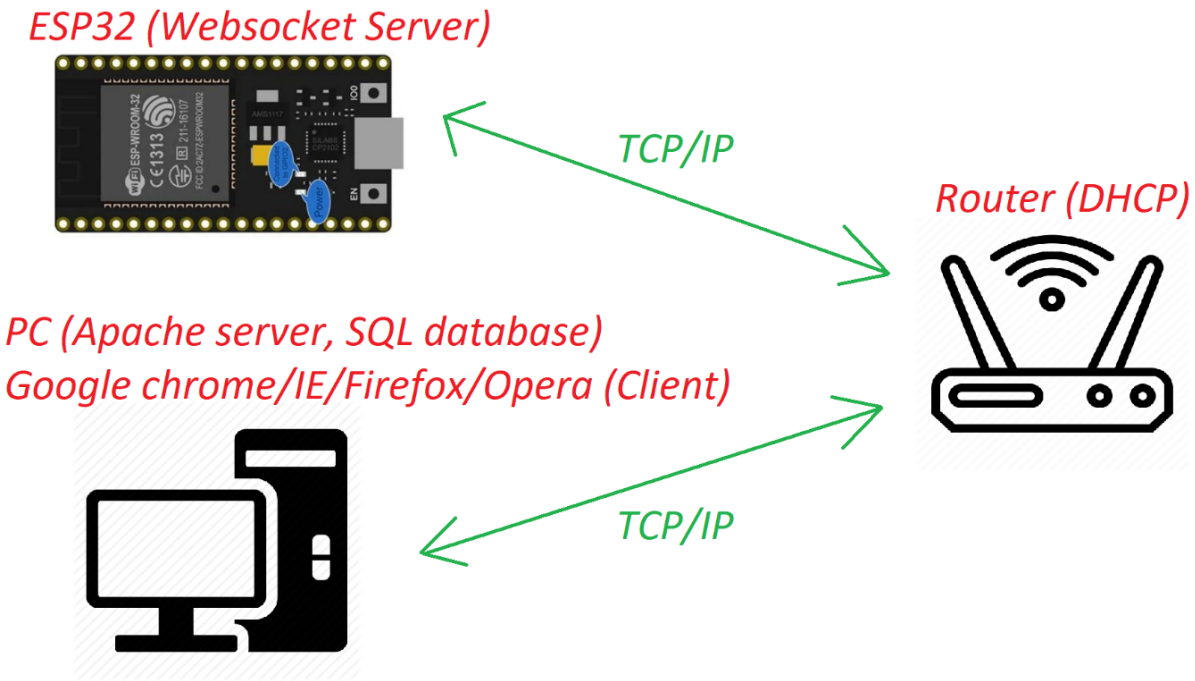
Το TCP προσφέρει ένα αξιόπιστο πρωτόκολλο πάνω από το IP. Εγγυάται ότι τα πακέτα θα παραδοθούν στον προορισμό τους, ότι θα φτάσουν με τη σειρά με την οποία στάλθηκαν και ότι τα περιεχόμενα των πακέτων θα φτάσουν αναλλοίωτα (δηλαδή όπως στάλθηκαν). Το TCP δουλεύει ως εξής : το κάθε πακέτο δεδομένων αριθμείται. Ο υπολογιστής - παραλήπτης και ο υπολογιστής - αποστολέας, αλλά όχι οι ενδιάμεσοι υπολογιστές, παρακολουθούν τους αριθμούς των πακέτων και ανταλλάσσουν μεταξύ τους πληροφορίες. Ο παραλήπτης λαμβάνει το πρώτο πακέτο, το δεύτερο, κλπ. Σε περίπτωση που παρουσιαστεί κάποιο πρόβλημα στο δίκτυο είτε χαθεί κάποιο πακέτο κατά τη διάρκεια της μετάδοσης, το ξαναζητάει και ο αποστολέας είναι υπεύθυνος για την αναμετάδοση του. Ο παραλήπτης ελέγχει επίσης αν το περιεχόμενο των πακέτων φτάνει σωστά. Η μέθοδος αυτή εξασφαλίζει αξιοπιστία και ταχύτητα διότι οι ενδιάμεσοι υπολογιστές δεν εκτελούν ελέγχους.

Τον πυρήνα του δικτύου, (και κάθε άλλου ιδιωτικού δικτύου στην περιοχή 192.168.X.X), αποτελεί ο δρομολογητής του διαδικτύου (Router). Όλες οι διατάξεις συνδέονται με τον πυρήνα του δικτύου. Η εφαρμογή DHCP που τρέχει μόνιμα στο δρομολογητή, εκχωρεί διεύθυνση IP σε κάθε υπολογιστική μονάδα που θα συνδεθεί ασύρματα ή ενσύρματα σε αυτόν.

Ο όρος DHCP (Dynamic Host Configuration Protocol) αναφέρεται σε ένα ειδικό μηχανισμό διαχείρισης TCP/IP πρωτοκόλλων. Το πρωτόκολλο είναι ουσιαστικά ένα λογισμικό που τρέχει σε έναν υπολογιστή και κανονίζει όλα τα θέματα επικοινωνίας με αυτόν τον υπολογιστή και άλλους που χρησιμοποιούν αυτό το πρωτόκολλο ως γλώσσα. Για να δουλέψει το ίδιο λογισμικό σε τόσους πολλούς υπολογιστές υπάρχει η ανάγκη να το ξεκινήσουμε σε κάθε υπολογιστή με τις αντίστοιχες

παραμέτρους για αυτόν και για τη θέση του στο δίκτυο. Η αρχικοποίηση αυτή μπορεί να γίνει κατά τη διάρκεια του φορτώματος (αν το πρωτόκολλο είναι συγχωνευμένο στο λειτουργικό σύστημα) ή με την κλήση του πρωτοκόλλου από κάποια εφαρμογή (αν το πρωτόκολλο υπάρχει στην εφαρμογή). Οι παράμετροι αυτές μπορούν να οριστούν τοπικά, για κάθε υπολογιστή ξεχωριστά. Κάτι τέτοιο όμως δημιουργεί αρκετά προβλήματα . Χρειάζεται πάρα πολύ εργασία από τον διαχειριστή του δικτύου η οποία είναι χρονοβόρα και επιρρεπής σε λάθη. Το να διατηρούνται οι παράμετροι ενημερωμένες χρειάζεται συνεχή δουλειά η οποία αυξάνεται γεωμετρικά με τις αλλαγές που συμβαίνουν στο δίκτυο, ειδικά αν υπάρχουν υπολογιστές που αλλάζουν συνεχώς θέση (π.χ. φορητοί Η/Υ) . Η αλλαγή μίας παραμέτρου κοινής για τους υπολογιστές σε ένα subnet (π.χ. τοπική διεύθυνση ενός router) απαιτεί αλλαγές σε κάθε υπολογιστή . Μερικά μηχανήματα μπορεί να λειτουργούν ως τερματικά. Κάτι τέτοιο σημαίνει ότι δεν έχουν αποθηκευτικό χώρο για να κρατήσουν τις ρυθμίσεις . Σε περιπτώσεις έλλειψης διευθύνσεων ή ενός δικτύου που αλλάζει συνέχεια είναι χάσιμο χρόνου να δίνουμε σε έναν μη σταθερό υπολογιστή μόνιμη διεύθυνση. Μία καλύτερη προσέγγιση θα ήταν να χρησιμοποιούνται ομάδες διευθύνσεων από ομάδες υπολογιστών. Η «χειροκίνητη» ρύθμιση τέτοιου είδους δεν παρέχει εύκολο τρόπο για να γίνει αυτό.

Σε ορισμένες περιπτώσεις και όταν αυτό είναι αναγκαίο μπορούν να αποδοθούν στους υπολογιστές του δικτύου διευθύνσεις IP οι οποίες δεν είναι στατικές αλλά δυναμικές. ,Αυτό σημαίνει πώς ο κάθε υπολογιστής δε θα έχει τη δική του σταθερή διεύθυνση αλλά κάθε φορά που θα συνδέεται στο δίκτυο θα ζητά να του χορηγηθεί μια ελεύθερη IP διεύθυνση από το σύνολο διευθύνσεων το οποίο είναι καθορισμένο εκ των προτέρων . Αυτή είναι και η ουσία του πρωτοκόλλου DHCP. Η εφαρμογή του πρωτοκόλλου DHCP σε ένα δίκτυο , λαμβάνει χώρα ακολουθεί την αρχιτεκτονική πελάτη διακομιστή (client server).



Σχήμα 5.1 : Απεικόνισή του ολοκληρωμένου μοντέλου επικοινωνίας σύμφωνα με την αρχιτεκτονική client-server πάνω στο πρωτόκολλο TCP/IP

Στην επιστήμη των υπολογιστών το μοντέλο αρχιτεκτονικής λογισμικού client-server (πελάτη-διακομιστή) αποτελεί μία συνήθη μέθοδο ανάπτυξης λογισμικού στην οποία ο πελάτης (ένα τμήμα λογισμικού) ζητά κάτι (π.χ. έναν πόρο, τα αποτελέσματα ενός υπολογισμού) και ένα άλλο τμήμα λογισμικού, ο διακομιστής (ή εξυπηρετητής), του το επιστρέφει. Κάθε διακομιστής μπορεί να εξυπηρετεί πολλαπλούς πελάτες.

Όπως φαίνεται και στο Σχ. 5.1, ο ESP32 σηκώνει έναν WebSocket server για τις απαιτήσεις του συστήματος ενώ ο υπολογιστής παίζει τον ρόλο ταυτόχρονος του client μέσω ενός φυλλομετρητή (browser) και του server μέσω του Apache web server. Παρακάτω αναλύονται όλες αυτές οι τεχνολογίες.

WebSocket

Το WebSocket είναι μια επέκταση ενός HTTP αιτήματος. Το Πρωτόκολλο Μεταφοράς Υπερκειμένου (HyperText Transfer Protocol, HTTP) είναι ένα πρωτόκολλο επικοινωνίας. Αποτελεί το κύριο πρωτόκολλο που χρησιμοποιείται στους φυλλομετρητές του Παγκοσμίου Ιστού για να μεταφέρει δεδομένα ανάμεσα σε έναν διακομιστή (server) και έναν πελάτη (client). Το WebSocket είναι μία μοναδική σύνδεση υποδοχής (socket connection) ή τερματική σύνδεση, που επιτρέπει ταυτόχρονη και αμφίδρομη (full-duplex, bi-directional) επικοινωνία πάνω σε μια TCP σύνδεση. Με το πρωτόκολλο WebSocket, το HTTP αίτημα, μετατρέπεται σε ένα μοναδικό αίτημα, για το άνοιγμα μια σύνδεσης υποδοχής, χρησιμοποιώντας την ίδια σύνδεση τόσο από τον πελάτη προς τον εξυπηρετητή, όσο και από τον εξυπηρετητή προς τον πελάτη. Με την τεχνολογία WebSocket, μειώνεται σημαντικά η καθυστέρηση επικοινωνίας και ανταλλαγής δεδομένων, διότι από την στιγμή που εγκατασταθεί μια σύνδεση WebSocket, ο εξυπηρετητής μπορεί και στέλνει πίσω στον πελάτη δεδομένα, όσο αυτά είναι διαθέσιμα. Σε αντίθεση με την τεχνική Polling, με το WebSocket γίνεται ένα μοναδικό αίτημα, ενώ στην συνέχεια, ο εξυπηρετητής δεν χρειάζεται να περιμένει αίτημα από τον πελάτη. Παρόμοια, ο πελάτης μπορεί να στέλνει μηνύματα στον εξυπηρετητή ανά πάσα στιγμή. Αυτό το μοντέλο αποστολής ενός μοναδικού αιτήματος μειώνει σημαντικά την καθυστέρηση, αντί της τεχνικής Polling, με την οποία στέλνεται ένα αίτημα ανά τακτά χρονικά διαστήματα, ανεξάρτητα των δεδομένων που είναι διαθέσιμα. Η TCP θύρα που χρησιμοποιείται από τα websockets είναι η 80 που είναι η καθιερωμένη θύρα του HTTP πρωτοκόλλου. Για να εγκατασταθεί μια σύνδεση πρέπει ο περιηγητής ιστού να στείλει ένα WebSocket handshake αίτημα και ο web server να απαντήσει με μία WebSocket handshake απάντηση. Παρόλο που αρχικά σχεδιάστηκε για χρήση από διακομιστές ιστού, και περιηγητές ιστού αντίστοιχα, μπορεί να χρησιμοποιηθεί σε οποιαδήποτε εφαρμογή που ακολουθεί το μοντέλο πελάτη-διακομιστή. Έτσι, προσφέρεται εύκολη, αμφίδρομη διαρκής επικοινωνία, για όσο η σύνδεση παραμένει ανοιχτή. Πολύ σημαντικό πλεονέκτημα των WebSockets είναι η ευκολία που προσφέρουν στον προγραμματιστή στην ανάπτυξη ενός προγράμματος βασισμένου σε αυτά. Πρακτικά χρειάζεται να υλοποιηθούν 4 βασικές συναρτήσεις. Σε διάφορες υλοποιήσεις υπάρχουν και πιο εξειδικευμένες συναρτήσεις αλλά οι 4 βασικές είναι οι εξής:

- onOpen(): Εκτελείται όταν το WebSocket ανοίγει.
- onClose(): Εκτελείται όταν το WebSocket κλείνει.
- onMessage(): Εκτελείται όταν το WebSocket δεχθεί μήνυμα.
- onError(): Εκτελείται όταν συμβεί κάποιο σφάλμα.

Apache Web Server

Για την υλοποίηση μιας web εφαρμογής είναι απαραίτητο να υπάρχει ένας web Server ο οποίος θα στέλνει τα αρχεία της σελίδας κατά την ζήτηση τους από τον φυλλομετρητή και θα επικοινωνεί με την βάση δεδομένων. Για την μεταφορά των αρχείων και των αιτημάτων χρησιμοποιείται το πρωτόκολλο HTTP. Το Apache HTTP Server Project είναι μια συνεργατική προσπάθεια ανάπτυξης λογισμικού που στοχεύει στη δημιουργία μιας ισχυρής, εμπορικής ποιότητικής, λειτουργικής και δωρεάν διαθέσιμου πηγαίου κώδικα υλοποίησης ενός διακομιστή HTTP (Web). Το έργο διαχειρίζεται από κοινού μια ομάδα εθελοντών που βρίσκονται σε όλο τον κόσμο, χρησιμοποιώντας το Διαδίκτυο και τον Ιστό για να επικοινωνούν, να σχεδιάζουν και να αναπτύσσουν τον διακομιστή και τα σχετικά έγγραφα. Επιπλέον, εκατοντάδες χρήστες έχουν συνεισφέρει ιδέες, κώδικα και τεκμηρίωση στο έργο.

Παρόλο που ονομάζουμε Apache έναν διακομιστή ιστού, δεν είναι ένας φυσικός διακομιστής, αλλά ένα λογισμικό που εκτελείται σε έναν διακομιστή. Η δουλειά του είναι να δημιουργήσει μια σύνδεση μεταξύ ενός διακομιστή και των προγραμμάτων περιήγησης των επισκεπτών του ιστότοπου (Firefox, Google Chrome, Safari κ.λπ.), ενώ παράλληλα παραδίδει αρχεία μεταξύ τους (δομή πελάτη-διακομιστή). Το Apache είναι ένα λογισμικό πολλαπλών πλατφορμών, επομένως λειτουργεί σε διακομιστές Unix και Windows. Όταν ένας επισκέπτης θέλει να φορτώσει μια σελίδα στον ιστότοπο, για παράδειγμα, την αρχική σελίδα ή τη σελίδα "Σχετικά με εμάς", το πρόγραμμα περιήγησής του στέλνει ένα αίτημα στον διακομιστή και ο Apache επιστρέφει μια απάντηση με όλα τα ζητούμενα αρχεία (κείμενο, εικόνες κ.λπ). Ο διακομιστής και ο πελάτης επικοινωνούν μέσω του πρωτοκόλλου HTTP και το λογισμικό Apache είναι υπεύθυνο για την ομαλή και ασφαλή επικοινωνία μεταξύ των δύο μηχανών. Ο Apache είναι εξαιρετικά προσαρμόσιμος, καθώς έχει δομή βασισμένη σε λειτουργικές μονάδες. Οι μονάδες επιτρέπουν στους διαχειριστές διακομιστών να ενεργοποιούν και να απενεργοποιούν πρόσθετες λειτουργίες. Διαθέτει λειτουργικές μονάδες για ασφάλεια, προσωρινή αποθήκευση, επανεγγραφή διευθύνσεων URL, έλεγχο ταυτότητας κωδικού πρόσβασης και πολλά άλλα.

5.3 Ανάλυση και κατασκευή της Web εφαρμογής και επικοινωνία με τον ESP32

Αρχεία web εφαρμογής (ή ιστοσελίδας)

Μια web εφαρμογή (εφαρμογή ιστού) είναι ένα λογισμικό εφαρμογών που εκτελείται σε έναν διακομιστή ιστού, σε αντίθεση με τα προγράμματα λογισμικού που βασίζονται σε υπολογιστή και αποθηκεύονται τοπικά στο Λειτουργικό Σύστημα (OS) της συσκευής. Η πρόσβαση στις εφαρμογές Ιστού γίνεται από τον χρήστη μέσω ενός προγράμματος περιήγησης ιστού με ενεργή σύνδεση σε ένα τοπικό δίκτυο (ή το Διαδίκτυο). Τα βασικά στοιχεία που απαρτίζουν μια web εφαρμογή είναι αρχεία HTML , CSS και JavaScript.

Η HTML (HyperText Markup Language) Γλώσσα Σήμανσης Υπερκειμένου είναι η κύρια γλώσσα σήμανσης για τις ιστοσελίδες, και τα στοιχεία της είναι τα βασικά δομικά στοιχεία των ιστοσελίδων. Η HTML γράφεται υπό μορφή elements (στοιχείων) HTML τα οποία αποτελούνται από ετικέτες (tags), οι οποίες περικλείονται μέσα σε σύμβολα «μεγαλύτερο από» και «μικρότερο από» (για παράδειγμα

<html>), μέσα στο περιεχόμενο της ιστοσελίδας. Οι ετικέτες HTML συνήθως λειτουργούν ανά ζεύγη (για παράδειγμα <h1> και </h1>), με την πρώτη να ονομάζεται ετικέτα έναρξης και τη δεύτερη ετικέτα λήξης (ή σε άλλες περιπτώσεις ετικέτα ανοίγματος και ετικέτα κλεισίματος αντίστοιχα). Ανάμεσα στις ετικέτες, οι σχεδιαστές ιστοσελίδων μπορούν να τοποθετήσουν κείμενο, πίνακες, εικόνες κλπ. Ο σκοπός ενός web browser είναι να διαβάζει τα έγγραφα HTML και να τα συνθέσει σε σελίδες που μπορεί κανείς να διαβάσει ή να ακούσει. Ο browser δεν εμφανίζει τις ετικέτες HTML, αλλά τις χρησιμοποιεί για να παρουσιάσει το περιεχόμενο της σελίδας. Τα στοιχεία της HTML χρησιμοποιούνται για να κτίσουν όλους του ιστότοπους. Η HTML επιτρέπει την ενσωμάτωση εικόνων και άλλων αντικειμένων μέσα στη σελίδα, και μπορεί να χρησιμοποιηθεί για να εμφανίσει διαδραστικές φόρμες. Παρέχει τις μεθόδους δημιουργίας δομημένων εγγράφων (δηλαδή εγγράφων που αποτελούνται από το περιεχόμενο που μεταφέρουν και από τον κώδικα μορφοποίησης του περιεχομένου) καθορίζοντας δομικά σημαντικά στοιχεία για το κείμενο, όπως κεφαλίδες, παραγράφους, λίστες, συνδέσμους, παραθέσεις και άλλα. Μπορούν επίσης να ενσωματώνονται σενάρια εντολών σε γλώσσες όπως η JavaScript, τα οποία επηρεάζουν τη συμπεριφορά των ιστοσελίδων HTML και από στατικές τις κάνουν διαδραστικές. Οι Web browsers μπορούν επίσης να αναφέρονται σε στυλ μορφοποίησης CSS για να ορίζουν την εμφάνιση και τη διάταξη του κειμένου και του υπόλοιπου υλικού. Ο οργανισμός W3C, ο οποίος δημιουργεί και συντηρεί τα πρότυπα για την HTML και τα CSS, ενθαρρύνει τη χρήση των CSS αντί διαφόρων στοιχείων της HTML για σκοπούς παρουσίασης του περιεχομένου

Η CSS (Cascading Style Sheets – διαδοχικά φύλλα ύφους ή επάλληλα φύλλα ύφους) είναι μια γλώσσα υπολογιστή που ανήκει στην κατηγορία των γλωσσών φύλλων ύφους που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μια γλώσσα σήμανσης. Χρησιμοποιείται δηλαδή για τον έλεγχο της εμφάνισης ενός εγγράφου που γράφτηκε σε γλώσσα HTML, δηλαδή για τον έλεγχο της εμφάνισης μιας ιστοσελίδας και γενικότερα ενός ιστοτόπου. Η CSS είναι μια γλώσσα υπολογιστή προορισμένη να αναπτύσσει στιλιστικά μια ιστοσελίδα δηλαδή να διαμορφώνει περισσότερα χαρακτηριστικά, χρώματα, στοίχιση και δίνει περισσότερες δυνατότητες σε σχέση με την html. Για μια όμορφη και καλοσχεδιασμένη ιστοσελίδα η χρήση της CSS κρίνεται ως απαραίτητη.

Η JavaScript (JS) είναι διερμηνευμένη γλώσσα προγραμματισμού για ηλεκτρονικούς υπολογιστές. Αρχικά αποτέλεσε μέρος της υλοποίησης των φυλλομετρητών Ιστού, ώστε τα σενάρια από την πλευρά του πελάτη (client-side scripts) να μπορούν να επικοινωνούν με τον χρήστη, να ανταλλάσσουν δεδομένα ασύγχρονα και να αλλάζουν δυναμικά το περιεχόμενο του εγγράφου που εμφανίζεται. Η JavaScript είναι μια γλώσσα σεναρίων που βασίζεται στα πρωτότυπα (prototype-based), είναι δυναμική, με ασθενείς τύπους και έχει συναρτήσεις ως αντικείμενα πρώτης τάξης. Η σύνταξή της είναι επηρεασμένη από τη C. Η JavaScript αντιγράφει πολλά ονόματα και συμβάσεις ονοματοδοσίας από τη Java, αλλά γενικά οι δύο αυτές γλώσσες δε σχετίζονται και έχουν πολύ διαφορετική σημασιολογία. Είναι γλώσσα βασισμένη σε διαφορετικά προγραμματιστικά παραδείγματα (multi-paradigm), υποστηρίζοντας αντικειμενοστρεφές, προστακτικό και συναρτησιακό στυλ προγραμματισμού.

Η γενικά ιδέα της web εφαρμογής είναι να υποστηρίζει την συσκευή σε θέμα λειτουργιών, δηλαδή να μπορεί να επεμβαίνει στην λειτουργία της μέτρησης και να απεικονίζει και εξίσου τις προσπάθειες της διαφραγματικής αναπνοής , βασίστηκε στο σενάριο ότι ο ασθενής θα χρησιμοποιεί την συσκευή σε κάποιο απομακρυσμένο σημείο από την θέση του φυσικοθεραπευτή , ο οποίος θα κάθεται

μπροστά από έναν υπολογιστή η λάπτοπ και θα παρακολουθεί την μέτρηση μέσω της εφαρμογής. Προσφέρεται δυνατότητα δημιουργίας προφίλ ασθενή με κάποια βασικά στοιχεία και η δυνατότητα αποθήκευσης των δεδομένων και ανάκτησης αργότερα (ιστορικό ασθενή).

Η εφαρμογή περιέχει τρεις ιστοσελίδες που η κάθε μια τους είναι φτιαγμένη για μια συγκεκριμένη εργασία. Το κοινό συστατικό που βρίσκεται σε όλες τις σελίδες είναι το navigation menu (μενού πλοήγησης) σε χρώμα μπλε navy στο αριστερό σημείο που συμπεριλαμβάνει τα ονόματα των σελίδων. Κάνοντας αριστερό κλικ στα ονόματα αυτά γίνεται αυτόματη μεταφορά στην εκάστοτε διεύθυνση. Τα στοιχεία 'History' , 'Registration' και 'Home' είναι στοιχεία HTML τύπου <a>. Η ετικέτα <a> ορίζει έναν υπερσύνδεσμο, ο οποίος χρησιμοποιείται για τη σύνδεση από τη μία σελίδα στην άλλη. Το πιο σημαντικό χαρακτηριστικό του στοιχείου <a> είναι το χαρακτηριστικό (attribute) 'href', το οποίο δείχνει τον προορισμό του συνδέσμου.

Τα χρώματα και οι γραμματοσειρές και τα μεγέθη των στοιχείων η στοίχιση και όλες οι οδηγίες που χρειάζονται περιγράφονται σε αρχεία CSS τα οποία εισάγονται στην αρχή του αρχείου της σελίδας. Τα αρχεία CSS που χρησιμοποιούνται σε σελίδες στον Διαδίκτυο συνήθως είναι μεγάλου μήκους με πολλούς κανόνες και δεν περιορίζονται σε ένα μόνο αρχείο κάτι που κάνει την ανάλυση τους και γραμμή προς γραμμή αδύνατο. Το γενικό σκεπτικό είναι ότι αυτοί οι κανόνες αποδίδονται στην μεγαλύτερη των περιπτώσεων σε γκρουπάκια που ονομάζονται κλάσεις. Οπότε για να περαστούν αυτοί οι κανόνες σε οποιοδήποτε element στην σελίδα αρκεί να συμπληρωθούν το ονόματα των κλάσεων χωρισμένα με κενό στο attribute του element 'class'. Ένα επίσης πολύ βοηθητικό framework CSS που δίνει πάρα πολλές ευκολίες διαχείρισης είναι το Bootstrap το οποίο χρησιμοποιείται και σε αυτές τις σελίδες. Με απλά λόγια, το Bootstrap είναι μια μεγάλη γκάμα εργαλείων που βασίζονται σε επαναχρησιμοποιήσιμο κώδικα, ώστε να μην χρειάζεται να ανάπτυξη των δομικών στοιχείων ενός ιστότοπου από το μηδέν.

Στο Σχ.5.2 φαίνεται η σελίδα 'Home' της web εφαρμογής που υλοποιήθηκε. Τα κύρια δομικά μέρη αυτής είναι :

- Τα πλαίσια προσπαθειών ,όπου εκεί αναγράφονται οι μετρήσεις που στέλνονται από την συσκευή στην σελίδα. Η πάνω μέτρηση με την ετικέτα 'Inhaling & Breathing' αφορά την μέτρηση της μεγίστης διακύμανση της πίεσης που προκαλείται κατά την διαφραγματική αναπνοή ενώ κάτω από την ετικέτα 'Inhaling' η τιμή της διακύμανσης της πίεσης που προκαλείται λόγω της εισπνοής. Τα χρώματα των ετικετών 'Attempt' ακολουθούν τον ίδιο χρωματισμό όπως την συσκευή.
- Την γραφική παράσταση των προσπαθειών , όπου εκεί σχηματίζονται οι κυματομορφές της πίεσης σε συνάρτηση με τον χρόνο όπως προκύπτουν κατά την μέτρηση. Συνολικά σχηματίζονται τρεις κυματομορφές με τα αντίστοιχα χρώματα όπως και τα πλαίσια προσπαθειών και της συσκευής. Για την δημιουργία της γραφικής παράσταση έγινε χρήση μιας επιπρόσθετη βιβλιοθήκης σχεδίασης βασισμένη πάνω στην Javascript (CanvasJS).
- Ο Έλεγχος της συσκευής περιλαμβάνει την δυνατότητα αλλαγής του mode όπως αναφέρθηκε στην ενότητα 4.4 που από την συσκευή που εκεί γίνεται χειροκίνητα με το πάτημα του κουμπιού έναρξης για 2 συνεχόμενα δευτερόλεπτα. Πατώντας το κουμπί 'Start' δίνεται σήμα στην συσκευή να γίνει η έναρξη της προσπάθειας. Πατώντας το κουμπί 'Reset' δίνεται σήμα στην συσκευή να κάνει ένα soft-reset. Πατώντας το κουμπί 'Save' , η σελίδα επικοινωνεί με την βάση για αποθήκευση των δύο τιμών της καλύτερης προσπάθειας.

- Η μπάρα πρόοδού ολοκληρωμένης μέτρησης γεμίζει σε πράσινο χρώμα κάθε φορά που γίνεται μια ολοκληρωμένη προσπάθεια σωστά και ανεβαίνει κάθε φορά κατά ποσοστό 33%. Μετά το πέρας των τριών επιτυχημένων προσπαθειών γεμίζει κατά ποσοστό 100% σηματοδοτώντας το τέλος των τριών ολοκληρωμένων προσπαθειών μέτρησης της διαφραγματικής αναπνοής.
- Η επιλογή χρήστη περιλαμβάνει ένα drop-down menu όπου πατώντας το εμφανίζονται όλοι η εγγεγραμμένοι χρήστες και επιλέγεται αυτός για τον οποίον γίνονται οι μετρήσεις και η αποθήκευση τους.

Παρακάτω αναλύεται ο προγραμματισμός στην Javascript που αφορά όλη την δυναμική της σελίδας , δηλαδή τις αλλαγές που γίνονται στις τιμές στο πλαίσιο προσπαθειών στις καρτέλες 'Attempt', την δημιουργία του γραφήματος, την λειτουργία των κουμπιών 'Start-Reset-Start', και την απεικόνιση της μπάρας προόδου. Κατά το άνοιγμα της σελίδας , δημιουργείται μια websocket σύνδεση (ως client) στον websocket server που έχει στήσει η συσκευή και υλοποιείται η συνάρτηση *OnMessage(data)* , που εκτελείται κάθε φορά που ο server στέλνει δεδομένα στο socket. Η συνάρτηση δέχεται ως όρισμα τα δεδομένα (data) και με μια συνθήκη (αναλόγως το μέγεθος των δεδομένων) περνάνε είτε συνάρτηση που σχεδιάζει το γράφημα είτε συνάρτηση που ενημερώνει το πλαίσιο προσπαθειών, την μπάρα προόδου και το πλαίσιο ελέγχου της συσκευής. Τα δεδομένα που έρχονται από την συσκευή (ESP32) μέσω του websocket ακολουθούν το πρότυπο JSON.

Οι κανόνες για την σύνταξή της μορφής JSON είναι οι εξής.

- Τα δεδομένα πηγαίνουν σε ζευγάρια (κλειδή-τιμή)
- Τα δεδομέναν διαχωρίζονται με κόμμα
- Τα άγκιστρα περιέχουν αντικείμενα (objects)
- Οι αγκύλες περιέχουν πίνακες

Τα πακέτα δεδομένων που έρχονται από το ESP32 έχουνε τρείς μορφές. Το ένα είναι υπεύθυνο για να μεταφέρει όλες τις τιμές της μέτρησης για να διαγράψουν την κυματομορφή και έχει την μορφή.

[{"X":"Τιμή 1", "Y":"Τιμή 1"},[{"X":"Τιμή 2", "Y":"Τιμή 2"}, …. {"X":"Τιμή N", "Y":"Τιμή N"}]

για κάθε *σημείο δηλαδή σχηματίζεται και ένα object με την* τιμή *X* που περιγράφει την πίεση, ενώ η τιμή Y περιγράφει τον χρόνο για κάθε σημείο. Το άλλο πακέτο έχει την μορφή.

[{"attempt": "Τιμή προσπάθειας", "maxPr1":"Τιμή 1", "maxPr2":"Τιμή 2" , "maxPr3":"Τιμή 3", "maxPr4":"Τιμή maxPr (best attempt)", "minPr1": "Τιμή 1", "minPr2":" Τιμή 2", "minPr3":" Τιμή 3", "minPr4":" Τιμή minPr (best attempt)"}]

και ενημερώνει την σελίδα μέσω του κλειδιού *attempt* σε ποια προσπάθεια βρίσκεται ο χρήστης (από 1 έως 3) ενώ τα *maxPr* και *minPr* περιέχουν τις τιμές των *Inhaling&Exhaling* και *Inhaling* αντίστοιχα για κάθε προσπάθεια. Ο αριθμός 4 αναφέρεται στην καλύτερη προσπάθεια. Το τελευταίο πακέτο έχει την μορφή.

[{"situation": "Τιμή κατάστασης"}] ή [{"reset": "reset"}] ή [{"mode": "Τιμή mode"}]

Όπου το κλειδί *situation* έχει την τιμή κατάστασης που μπορεί να είναι 'Waiting to start' δηλαδή η κατάσταση της μέτρησης να είναι σε αναμονή για να ξεκινήσει , 'Relax' που σημαίνει η μέτρηση έχει ξεκινήσει και βρίσκεται στην κατάσταση ηρεμίας όπως έχει αναφερθεί στο Κεφ. 4.4, 'Measuring' που σημαίνει ότι η μέτρηση βρίσκεται σε εξέλιξη. Το πακέτο *reset* στέλνεται στην σελίδα σε οποιαδήποτε περίπτωση προκληθεί soft-reset στο πρόγραμμα μέτρησης του ESP32. Το πακέτο *mode* στέλνεται σε περίπτωση που γίνει αλλαγή από 'Beginner' σε 'Advanced' ή το αντίθετο, η τιμή που στέλνεται είναι το '0' όταν η τιμή είναι 'Beginner' και '1' όταν η τιμή είναι 'Advanced'.

- **Συνάρτηση σχεδίασης γραφήματος.**

Το γράφημα σχεδιάζεται μέσα στο element με attribute 'id' με τιμή 'chartContainer' της σελίδας Κάθε φορά που στέλνονται νέα *data* από το ESP32 στην σελίδα περνάνε στο κύριο κλειδί *'data'* του γραφήματος και γίνεται επανασχεδίαση του με τα νέα δεδομένα. Αυτή η συνάρτηση καλείται από την *OnMessage(data)*, τα *data* περιέχουν το πακέτο δεδομένων με όλες τις συντεταγμένες των σημείων που διαγράφουν την κυματομορφή. Κάθε φορά δηλαδή δημιουργείται ένα object που περιγράφει την κυματομορφή με τα κλειδιά *'type', 'color' , 'markertype', 'datapoints'* . Αναλόγως την προσπάθεια στην οποία βρίσκεται αποφασίζεται και το χρώμα το οποίο περνάει στο κλειδί *'color'*, τα δεδομένα *data* περνάνε ως τιμή στο κλειδί *'datapoints'* το κλειδί *'markertype'* παίρνει πάντα την τιμή *'none'* γιατί η κυματομορφή δεν πρέπει να έχει *markers* στα σημεία που την απαρτίζουν. Το κλειδί *'type'* έχει σταθερή τιμή *'line'* που περιγράφει το σχήμα (γραμμή στην συγκεκριμένη λειτουργία) των ενώσεων μεταξύ των σημείων. Αυτό το object εισάγεται στο κλειδί *'data'* του γραφήματος.

Το γράφημα έχει και άλλα κλειδιά που περιγράφουν διάφορες ιδιότητες του άξονα *X* και του άξονα *Y*. Κατά τον Y εισάγονται τα εξής ζευγάρια: *(title: "Pressure (mmHg)", titleFontSize: 20, viewportMaximum: maximumPr, viewportMinimum: minimumPr*). Η τιμή του κλειδιού *title* περιγράφει τον τίτλο του άξονα που αναφέρεται στην πίεση, η τιμή του κλειδιού *titleFontSize* περιγράφει το μέγεθος της γραμματοσειράς του τίτλου , και οι τιμές των κλειδιών *viewportMaximum* και *viewportMinimum* αναφέρονται στην τιμή αρχής και τέλους του άξονα και αυτές οι δύο εξαρτώνται άμεσα από το *mode* της συσκευής. Παρόμοια στον X είσαγονται τα εξής ζευγάρια: *{title: "Time (S)", titleFontSize: 20, viewportMinimum: 0, viewportMaximum: 12.5}.* Συνολικά δηλαδή το γράφημα έχει τρία κύρια κλειδιά , το '*axisX*' , το '*axisY*' και το '*data*' , με τα ζευγάρια κλειδιών-τιμών όπως αναφέρθηκαν παραπάνω.

- **Συνάρτηση ενημέρωσης (πλαίσιο προσπαθειών, μπάρα προόδου, πλαίσιο ελέγχου)**

Με αυτήν την συνάρτηση μπαίνουν όλες οι τιμές που έρχονται από το ESP32 στα html elements της σελίδας. Η ταξινόμηση των δεδομένων γίνεται με μια συνθήκη που εξετάζει αν το εκάστοτε όνομα του κλειδιού περιέχεται στα πακέτα δεδομένων JSON. Η επιλογή των elements γίνεται με την βοήθεια της Javascript και πιο συγκεκριμένα με το βοηθητικό framework JQuery που μπορεί να χρησιμοποιηθεί εφόσον γίνει include μέσα στην σελίδα. Η σύνταξη γίνεται ως εξής *$('#id')* για την επιλογή ενός element μέσω του *id* του όπως εφαρμόζεται στην συγκεκριμένη σελίδα. Για παράδειγμα για να γίνει μια αλλαγή στην εμφάνιση του δηλαδή κατά CSS στο χρώμα του element η σύνταξη είναι ως εξής *$('#id').css("color", "τιμή χρώματος").*

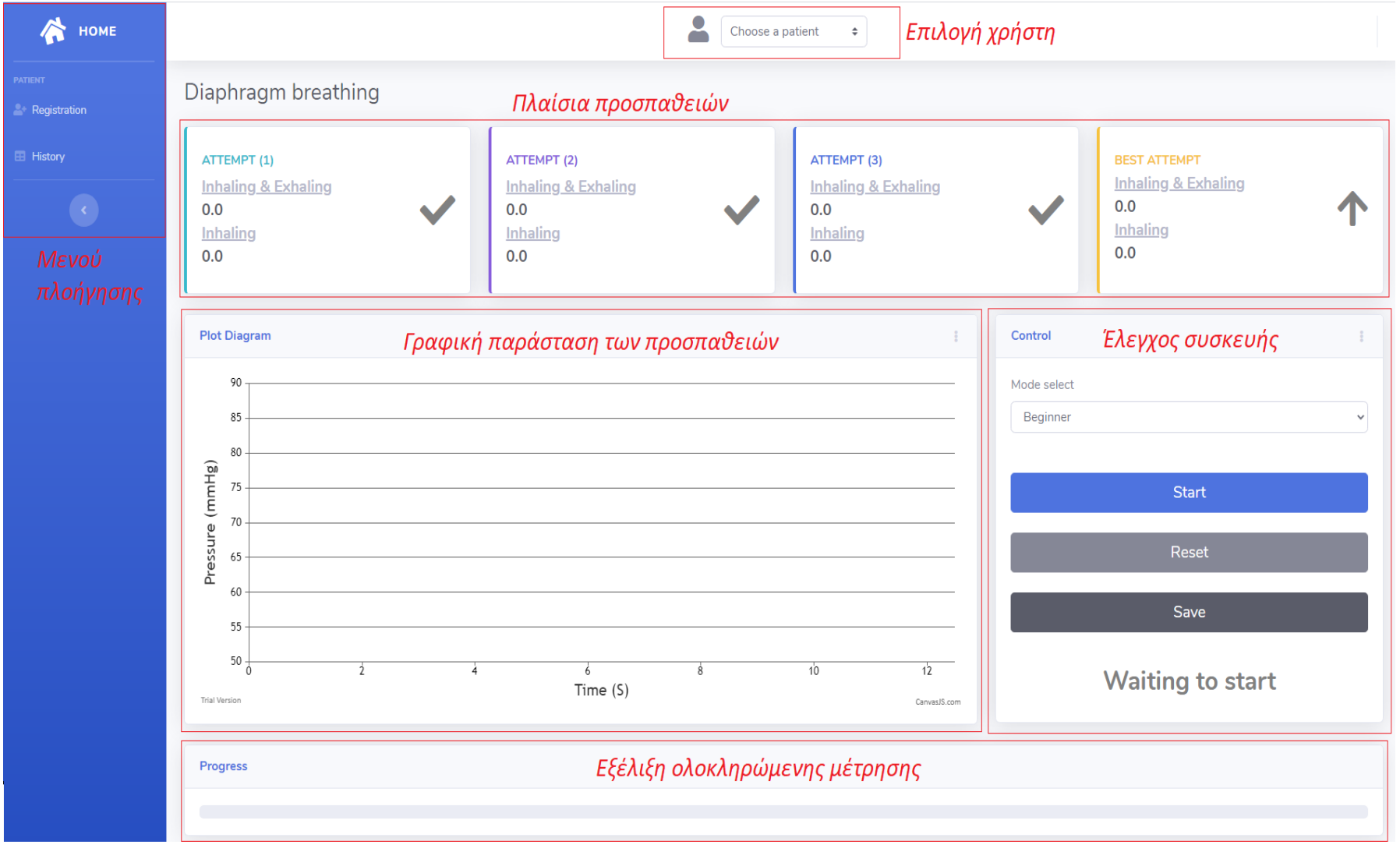
Αν βρεθεί το κλειδί *'attempt'* τότε γίνεται επιλογή του element μέσω του attribute 'id' που αποτελείται από το string (*attempt"+p[key]+"_check*) που περιέχει το check στο πλαίσιο προσπαθειών της προσπάθειας (από 1 έως 3) και αλλάζει το χρώμα του σε πράσινο *,* που το *p[key]* αναφέρεται στην τιμή του κλειδιού 'attempt', παράλληλα μέσω της εντολής *$("#attempt_bar").css("width", "ποσοστό πλάτους τις %")* το element που περιγράφει την μπάρα πρόοδού με attribute *'id = attempt_bar'* γίνεται αλλαγή στην CSS ιδιότητα του width και αναλόγως την τιμή του κλειδιού *attempt (*αν ισούται με 1 , το width γίνεται 33% , αν ισούται με 2 , το width γίνεται 66% , αν ισούται με 3 , το width γίνεται 100%).

Αν βρεθεί το κλειδί *'situation'* στο πακέτο δεδομένων JSON, τότε γίνεται επιλογή του element μέσω του attribute *'id=situation'*, το κείμενο του element παίρνει την τιμή του κλειδιού και αλλάζει το χρώμα του αναλόγως την τιμή. Αν αυτή η τιμή είναι 'Relax' τότε το χρώμα γίνεται κίτρινο , αν έχει την τιμή 'Measuring' το χρώμα γίνεται πράσινο και αν η τιμή ισούται με 'Waiting to start' το χρώμα γίνεται γκρι όπως φαίνεται στο Σχ.5.2.

Αν βρεθεί το κλειδί 'reset' τότε γίνεται διαγραφή του σχεδιαγράμματος και η επανασχεδίασή του με 'data' να ισούνται με μηδέν με την βοήθεια της συνάρτησης σχεδίασης γραφήματος. Επιπρόσθετα κατά το *reset* της σελίδας , διαγράφονται όλες οι τιμές στο πλαίσιο ελέγχου και παίρνουν την τιμή '0.0' και η μπάρα πρόοδού έχει μήκος γεμίσματος 0%.

Αν βρεθεί το κλειδί 'mode' επιλέγεται το element 'Mode select' στο πλαίσιο ελέγχου και αλλάζει η τιμή σε 'Beginner' αν η τιμή του κλειδιού 'mode' είναι μηδέν αλλιώς γίνεται 'Advanced' αν η τιμή ισούται με ένα. Αυτό επιτυγχάνεται με την χρήση της εντολής *$('[name=options]').val(p[key]),* όπου επιλέγεται το <select> HTML element που έχει το attribute 'name=options' με το p[key] να περιγράφει την τιμή του κλειδιού 'mode'. Παράλληλα με την αλλαγή του Mode αλλάζουν και τα όρια της κλίμακας της πίεσης του γραφήματος (άξονας Y). Στο Beginner mode τα όρια είναι 50-90 , ενώ στο Advanced mode τα όρια είναι 45-120.

Αν βρεθεί το κλειδί 'minPr1…4' η το κλειδί 'maxPr1…4' αναφέρεται στις τιμές του πλαισίου προσπαθειών και έχει να κάνει με τις μετρήσεις τις διαφραγματικής αναπνοής. Επιλέγεται το element με attribute 'id' ίδιο με το κλειδί (Π.χ *id='minPr*1') και με την χρήση της εντολής $("#"+key).html("Τίμη του κλειδιού") συμπληρώνονται όλες οι τιμές με ένα ψηφίο μετά την υποδιαστολή.



Σχήμα 5.2 Σελίδα home (κύρια σελίδα web εφαρμογής)

❖ **Event handlers (χειριστές συμβάντων)**

Μέχρι τώρα αναφέρθηκαν όλα οι ενέργειες που γίνονται μόλις σταλθούν δεδομένα από την συσκευή προς την σελίδα. Για να γίνει αντίθετη διαδικασία αξιοποιούνται οι event handlers που προσφέρουν την δυνατότητα αυτόματης εκτέλεσης μόλις προκληθούν. Κάποια από τα συμβάντα αυτά είναι είτε όταν πατηθεί ένα κουμπί της σελίδας , είτε αλλάξει τιμή ένα element επιλογής. Στη συγκεκριμένη σελίδα χρησιμοποιούνται τρείς ξεχωριστεί event handlers τύπου 'onclick' για τα τρία ξεχωριστά κουμπιά (*start*,*reset*,*save*) και ένας event handler τύπου 'onchange' που σηματοδοτεί την αλλαγή τιμής στο <select> element *Mode.*

Με την χρήση της εντολής *$("#resetbut").click(function() {}* υλοποιείται ο handler για το κουμπί *Reset* της σελίδας *(element <button> με attribute id='resetbut')* μέσα στην συνάρτηση *(function() {})* στέλνεται το string *'reset'* στον websocket server (ESP32) και γίνεται το reset (όπως αναφέρθηκε πιο πάνω στην περίπτωση που βρεθεί το κλειδί 'reset' στα εισερχόμενα δεδομένα).

Με την χρήση της εντολής *$("#startbut").click(function() {}* υλοποιείται ο handler για το κουμπί *Start* της σελίδας *(element <button> με attribute id='startbut')* μέσα στην συνάρτηση *(function() {})* στέλνεται το string *'start'* στον websocket server (ESP32) και αυτό δίνει στο πρόγραμμα της συσκευής την έγκριση στο να ξεκινήσει την προσπάθεια μέτρησης εφόσον πληροί όλες τις προϋποθέσεις όπως αναφέρεται στο Κεφ. 4.4.

Με την χρήση της εντολής *$("#savebut").click(function() {}* υλοποιείται ο handler για το κουμπί *Save* της σελίδας *(element button με attribute id='savetbut')* μέσα στην συνάρτηση *(function() {})* ελέγχεται αν η προσπάθειες έχουν ολοκληρωθεί (δηλαδή έχουν συμπληρωθεί και οι τρείς) και ταυτόχρονα έχει επιλεγεί κάποιος χρήστης δηλαδή στο element select με attribute '*id=user_select'* η επιλογή *option δεν είναι 'Choose a patient'*. Αν εκπληρώνονται αυτές οι δύο συνθήκες τότε οι δύο τιμές στην καρτέλα 'BEST ATTEMPT' στο πλαίσιο προσπαθειών στέλνονται στην βάση δεδομένων κάτι που αναλύεται στο παρακάτω Κεφ. 5.3

Με την χρήση της εντολής *$('#mode').on('change', function() {}* υλοποιείται ο handler για το element επιλογής 'Mode select' μέσα στην συνάρτηση *(function() {})* εξετάζεται ποιο είναι το value της επιλεγμένης τιμής option του <select>. Αν ισούται με μηδέν τότε στέλνεται το string *'mode0'* στον websocket server (ESP32) , αλλιώς στέλνεται *'mode1'.* Αναλόγως το Mode που επιλέγεται αλλάζονται τα όρια της κλίμακας του γραφήματός στην σελίδα και το ESP32.

Πίνακας 5.1 Ο MySQL πίνακας users για την αποθήκευση των στοιχείων των χρηστών

Name	Type	Length	Decimals	Not null	Virtual	Key	Comment
id	int	11	0	☑	☐	🔑 1	
name	varchar	50	0	☑	☐		
surname	varchar	50	0	☑	☐		
age	int	11	0	☑	☐		
activity	int	11	0	☑	☐		

5.4 Δημιουργία βάσης δεδομένων και επικοινωνία με τις σελίδες

Στο προηγούμενο Κεφ. 5.2 αναφέρθηκε ο τρόπος που επικοινωνεί αμφίδρομα το ESP32 με την σελίδα μέσω του πρωτοκόλλου websocket και όλα τα δεδομένα που στέλνονται από και προς την σελίδα. Σε αυτό το κεφάλαιο αναλύεται η δημιουργία της βάσης δεδομένων και πώς γίνεται η επικοινωνία για είτε για αποθήκευση δεδομένων στην βάση , είτε ανάκτηση τους. Η βάση δεδομένων που κατασκευάστηκε είναι τύπου SQL. Η SQL (Structured Query Language) είναι μια τυποποιημένη γλώσσα προγραμματισμού που χρησιμοποιείται για τη διαχείριση σχεσιακών βάσεων δεδομένων και την εκτέλεση διαφόρων λειτουργιών στα δεδομένα σε αυτές. Αρχικά δημιουργήθηκε τη δεκαετία του 1970, η SQL χρησιμοποιείται τακτικά όχι μόνο από διαχειριστές βάσεων δεδομένων, αλλά και από προγραμματιστές που γράφουν σενάρια ενοποίησης δεδομένων και αναλυτές δεδομένων που θέλουν να δημιουργήσουν και να εκτελέσουν αναλυτικά ερωτήματα. Οι χρήσεις της SQL περιλαμβάνουν την τροποποίηση δομών πινάκων βάσεων δεδομένων και ευρετηρίου. προσθήκη, ενημέρωση και διαγραφή σειρών δεδομένων · και ανάκτηση υποσυνόλων πληροφοριών από μια βάση δεδομένων για εφαρμογές επεξεργασίας συναλλαγών και αναλυτικών στοιχείων. Τα ερωτήματα και άλλες λειτουργίες SQL λαμβάνουν τη μορφή εντολών που γράφονται ως queries. Επίσης γνωστά στις βάσεις δεδομένων SQL, τα σχεσιακά συστήματα περιλαμβάνουν ένα σύνολο πινάκων που περιέχουν δεδομένα σε σειρές και στήλες. Κάθε στήλη σε έναν πίνακα αντιστοιχεί σε μια κατηγορία δεδομένων - για παράδειγμα, όνομα πελάτη ή διεύθυνση - ενώ κάθε σειρά περιέχει μια τιμή δεδομένων για τη διασταυρούμενη στήλη. Στην βάση που υλοποιήθηκε χρησιμοποιείται συγκεκριμένα ή MySQL. Επειδή είναι ανοικτού κώδικα (open source), οποιοσδήποτε μπορεί να κατεβάσει τη MySQL και να την διαμορφώσει με βάση τις ανάγκες του, σύμφωνα πάντα με την γενική άδεια χρήσης. Είναι γνωστή κυρίως για την ταχύτητα, την αξιοπιστία, και την ευελιξία που παρέχει. Η MySQL αυτή τη στιγμή μπορεί να λειτουργήσει σε περιβάλλον Linux, Unix, και Windows.

Για την συγκεκριμένη εφαρμογή χρειάστηκαν δύο πίνακες. Ένας πίνακας που θα αποθηκεύει ένα προφίλ χρήστη όπως φαίνεται στον Πίνακα 5.1 και ένας πίνακας για να αποθηκεύει τα δεδομένα μετρήσεων για την διατήρηση ιστορικού. Ο πίνακας *users* περιλαμβάνει 5 στήλες, το *id* αναφέρεται στον αριθμό εγγραφής χρήστης και είναι μοναδικός (τύπου ακέραιος μήκους 11 ψηφίων), *name* και *surname* για το όνομα και το επίθετο(τύπου χαρακτήρων μήκους 50 ψηφίων), τη στήλη *age* που αναφέρεται στην ηλικία και την στήλη *activity* που αθλητική δραστηριότητα του χρήστη παίρνει τιμές από (1 έως 4).

Πίνακας 5.2 Ο MySQL πίνακας attempts για την αποθήκευση των στοιχείων της μέτρησης

	Name	Type	Length	Decimals	Not null	Virtual	Key	Comment
▶	id	int	11	0	☑	☐		
	maxPr	decimal	10	1	☑	☐		
	minPr	decimal	10	1	☑	☐		
	date	datetime	0	0	☑	☐		

Σχήμα 5.3 Σελίδα Registration (δημιουργίας χρήστη)

Τα στοιχεία σύνδεσης για την βάση στη συγκεκριμένη εφαρμογή είναι τα παρακάτω:

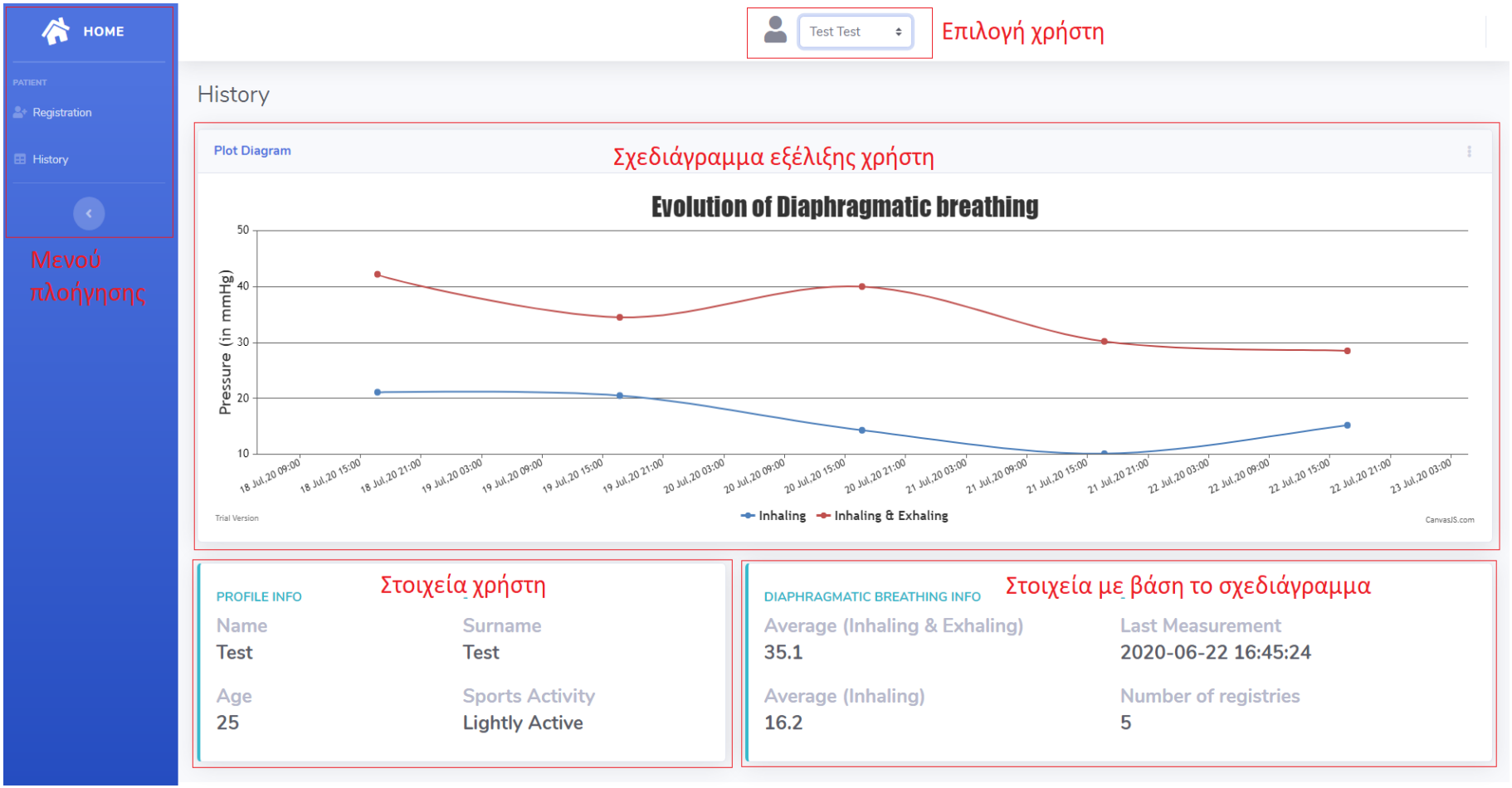
- Όνομα server -> *localhost*
- Όνομα χρήστη -> *root*
- Κωδικός -> κενό
- Όνομα βάσης δεδομένων -> *diabreathing*

Στο Σχ. 5.3 φαίνεται η σελίδα 'Registration' της web εφαρμογής που υλοποιήθηκε. Το πλαίσιο με τα στοιχεία του χρήστη αποτελείται από τέσσερα input (εισαγωγής) element πεδία , ένα πεδίο επιλογής <select> element και ένα element <button> 'Register'. Όλα αυτά περικλείονται από ένα element τύπου <form> (φόρμας). Οι φόρμες χρησιμοποιούνται για ομαδοποιήσουν τα δεδομένα που περιέχονται μέσα τους σε ζευγάρια κλειδιά-τιμές. Τα κάθε κλειδί σχηματίζεται από το attribute 'name' που έχει το element και η τιμή είναι αυτή που εισάγεται στο πεδίο (για τα input elements) ή τιμή που επιλέγεται (select element). Πατώντας το 'Register' <button> περνάνε τα ζευγάρια κλειδιά-τιμές μέσω μια συνάρτησης που εκτελεί ajax και μεταφέρει τα δεδομένα σε ένα PHP αρχείο 'register_user.php' που υλοποιεί σύνδεση και query στην βάση δεδομένων.

Η Ajax (Ασύγχρονη JavaScript και XML) είναι ένα σύνολο από τεχνικές που χρησιμοποιούν πολλές τεχνολογίες του διαδικτύου απο την πλευρά του πελάτη για να δημιουργήσουν ασύγχρονες web εφαρμογές. Με Ajax, οι web εφαρμογές μπορούν να στέλνουν και να ανακτούν δεδομένα από έναν διακομιστή(server) ασύγχρονα (τρέχοντας στο παρασκήνιο), χωρίς να παρεμβαίνουν στην εμφάνιση και τη συμπεριφορά της υπάρχουσας σελίδας. Πρακτικά τώρα αντί για XML χρησιμοποιείται η μορφή JSON για λόγους μεγαλύτερης ευκολίας. Η PHP (PHP: Hypertext Preprocessor) είναι μια γλώσσα προγραμματισμού για τη δημιουργία σελίδων web με δυναμικό περιεχόμενο και γίνεται εκμετάλλευση η ύπαρξη συναρτήσεων ώστε να γίνει η επικοινωνία με την βάση δεδομένων.

Η jQuery δίνει την χρήση της ajax μεταφοράς δεδομένων με την συνάρτηση $.ajax({ type: "POST", url: (διεύθυνση PHP αρχείου), data: (δεδομένα που αποστέλλονται), success: function(εισερχόμενα δεδομένα) {} }). Στη συγκεκριμένη κατάσταση στέλνονται τα δεδομένα της φόρμας με την HTTP μέθοδο POST στο PHP αρχείο 'register_user.php'. Σε όλες τις αποστολές των δεδομένων στα PHP αρχεία στη συγκεκριμένη web εφαρμογή χρησιμοποιείται η POST μέθοδος. Τα δεδομένα που αποστέλλονται με POST αποθηκεύονται στο σώμα αιτήσεων του αιτήματος HTTP δεν μπορούν να αρχειοθετηθούν, δεν έχουν όριο στο μέγεθος τους και δεν παραμένουν στο ιστορικό του φυλλομετρητή κάτι που καθιστά αυτήν την μέθοδο την πιο κοινή και ασφαλή ταυτόχρονα σε ένα HTTP request.

Στο αρχείο γίνεται η σύνδεση με την βάση χρησιμοποιώντας τα στοιχεία σύνδεσης και εκτελείται το query "*INSERT INTO users (name , surname , age , activity) VALUES ('$name' , '$surname' , '$age' , '$activity')*" . με τίς τιμές *$name, $surname, $age, $activity* να είναι οι τιμές των POST δεδομένων που στάλθηκαν από την φόρμα μέσω ajax της σελίδας Registration.



Σχήμα 5.4 Σελίδα History (ιστορικού χρήστη)

Στο Σχ. 5.4 φαίνεται η τελευταία σελίδα ‘History’ της web εφαρμογής που υλοποιήθηκε. Ο κύριος σκοπός αυτής της σελίδας είναι να αποτυπωθεί η εξέλιξη των μετρήσεων του χρήστη που αφορούν την διαφραγματική αναπνοή με βάση τις μετρήσεις ανά τον χρόνο.

Κάθε φορά που φορτώνει η σελίδα ‘History’ γίνεται ένα ajax request στο PHP αρχείο ‘get_users.php’. Σε αυτό το αρχείο γίνεται η σύνδεση με την βάση χρησιμοποιώντας τα στοιχεία σύνδεσης και εκτελείται το query "SELECT * FROM users" που σημαίνει ότι ζητούνται όλες οι εγγραφές δεδομένων στον πίνακα users. Έπειτα διαμορφώνεται ένας πίνακας μορφής JSON από το objects που αριθμός τους ισούται με το πλήθος *N* των γραμμένων χρηστών.

[{"id":"Id χρήστη 1", "name":"Όνομα χρήστη 1", "surname":"Επώνυμο χρήστη 1"},{"id":"Id χρήστη 2", "name":"Όνομα χρήστη 2", "surname":"Επώνυμο χρήστη 2"}, … {"id":"Id χρήστη N", "name":"Όνομα χρήστη N", "surname":"Επώνυμο χρήστη N"}]

Έπειτα ο JSON πίνακας γίνεται parsing στην σελίδα και στο element <select> επιλογής χρήστη εισάγονται σαν <option> elements με attribute ‘value’ το *Id χρήστη* και σαν εσωτερικό κείμενο το ‘Επίθετο κενό Όνομα’. Αυτή η διαδικασία γίνεται και στην σελίδα ‘Home’. Η αποθήκευση των δεδομένων των καλύτερων μετρήσεων (BEST ATTEMPT) γίνεται με ajax call στο αρχείο PHP ‘save_data’ με POST data *(minPr4*, *maxPr4*, *id*). Εκεί γίνεται η σύνδεση με την βάση χρησιμοποιώντας τα στοιχεία σύνδεσης και εκτελείται το query *"INSERT INTO attempts (minPr , maxPr , id , date) VALUES ('$minPr' , '$maxPr' , '$id' , now())"* . με τίς τιμές *$minPr', $maxPr', $ id'* να είναι οι τιμές των POST δεδομένων που στάλθηκαν από την σελίδα μέσω ajax , ενώ η τιμή *now()* είναι η τιμή της ημερομηνίας και ώρας την στιγμή της αποθήκευσης.

- **Event handlers (χειριστές συμβάντων)**

Στη συγκεκριμένη σελίδα χρησιμοποιείται ένας event handler τύπου ‘onchange’ που σηματοδοτεί την αλλαγή τιμής στο <select> element επιλογής χρήστη.Με την χρήση της εντολής *$('#user_select).on('change', function() {}* υλοποιείται ο handler για το <select> element επιλογής χρήστη μέσα στην συνάρτηση *(function() {})* εξετάζεται ποιο με τι ισούται το attribute ‘value’ της επιλεγμένης τιμής option του <select>. Αυτή η τιμή αντιπροσωπεύει το *Id χρήστη* (δηλαδή την ταυτότητα του , καθώς είναι μοναδικό). Με αυτήν την τιμή *Id ώς* POST δεδομένο γίνονται δύο Ajax το πρώτο στο αρχείο PHP ‘get_data.php’. Εκεί εκτελείται το query *"SELECT * FROM attempts WHERE id = '$id' ORDER BY date ASC"* που σημαίνει ότι ζητούνται όλες οι εγγραφές εκείνες πού έχουν το ίδιο *'$id'* με το *Id χρήστη* που επιλέχτηκε από το του πίνακα *attempts*. Τα δεδομένα έρχονται ταξινομημένα σύμφωνα με την ημερομηνία εγγραφής, η σειρά είναι από το παλαιότερο προς το νεότερο. Η διαμόρφωση του πίνακα JSON από τα objects με πλήθος *N* είναι τη μορφής.

[{"minPr":"Τιμή minPr 1", "maxPr ":" Τιμή maxPr 1", "date":"Ημερομηνία εγγραφής 1"},{"minPr":"Τιμή minPr 2", "maxPr ":" Τιμή maxPr 2", "date":"Ημερομηνία εγγραφής 2"}, … "},{"minPr":"Τιμή minPr N", "maxPr ":" Τιμή maxPr N", "date":"Ημερομηνία εγγραφής N"}]

Στο δεύτερο PHP αρχείο ‘choose_user.php’ εκτελείται το query *"SELECT * FROM users WHERE id = '$id' "* που ζητούνται όλα τα στοιχεία του χρήστη που ισούται με το *Id χρήστη* που επιλέχτηκε. Η διαμόρφωση του πίνακα JSON (περιέχει ένα object) είναι της μορφής.

[{"name":"Όνομα χρήστη", "surname":"Επώνυμο χρήστη", "age":"Ηλικία χρήστη",
"activity":"Αθλητική δραστηριότητα"}]

- **Συνάρτηση σχεδίασης γραφήματος.**

Το γράφημα σχεδιάζεται μέσα στο element με attribute 'id' με τιμή 'chartContainer' της σελίδας Κάθε φορά που γίνεται αλλαγή χρήστη νέα *data* από το την βάση δεδομένων στην σελίδα περνάνε στο κύριο κλειδί *'data'* του γραφήματος και γίνεται επανασχεδίαση του με τα νέα δεδομένα. Σε αυτό το γράφημα σχηματίζονται δύο κυματομορφές μία που αφορά την εξέλιξη της τιμής *minpr* (περιγράφει την αναπνοή) και μια που αφορά την εξέλιξη της τιμής *maxpr* (περιγράφει την αναπνοή και την εκπνοή μαζί). Κάθε φορά δηλαδή δημιουργούνται δύο object που περιγράφουν τις κυματομορφές με κοινά κλειδιά '*name', 'type', 'yValueFormatString', 'datapoints', 'showInLegend'*. Στο object της κυματομορφής που αποδίδονται τα δεδομένα της τιμής *minpr*, το κλειδί name' έχει τιμή *"Inhaling"*, τα δεδομένα *data* (*minPr , date*) περνάνε ως τιμή στο κλειδί *'datapoints'*, ενώ στο object που αποδίδονται τα δεδομένα της τιμής *maxpr*, το κλειδί name' έχει τιμή *"Inhaling & Exhaling"*, τα δεδομένα *data* (*maxPr , date*) περνάνε ως τιμή στο κλειδί *'datapoints'. Τ*ο κλειδί *'showInLegend'* παίρνει πάντα την τιμή *'true'* που εμφανίζει υπόμνημα το όνομα της κυματομορφής και το χρώμα της. Το κλειδί *'type'* έχει σταθερή τιμή *'spline'* που περιγράφει το σχήμα (καμπυλωτή γραμμή στην συγκεκριμένη λειτουργία) των ενώσεων μεταξύ των σημείων. Αυτά τα δύο object εισάγονται στο κύριο κλειδί *'data'* του γραφήματος.

Το γράφημα έχει και άλλα κύρια κλειδιά που περιγράφουν διάφορες ιδιότητες του γραφήματος. Στο κύριο κλειδί '*title'* του γραφήματος *text: "Evolution of Diaphragmatic breathing"*. Η τιμή του κλειδιού *'text'* περιγράφει τον τίτλο του γραφήματος. Στο κύριο κλειδί *'axisX'*, υπάρχει το ζευγάρι κλειδιού-τιμής *valueFormatString: "DD MMM,YY HH:mm"* που αναφέρεται στον τρόπο αναπαράστασης της ημερομηνίας στον άξονα *X* του γραφήματος, (Μέρα – Μήνας – Χρόνος, Ώρα:λεπτά). Στο κύριο κλειδί *'axisY'*, υπάρχει το ζευγάρι κλειδιού-τιμής *title: "Pressure (in mmHg)"* που αναφέρεται στον τίτλο του άξονα Y και το ζευγάρι κλειδιού-τιμής *includeZero:* false που γίνεται η επιλογή να μην συμπεριληφθεί το μηδέν σαν αρχική τιμή του άξονα. Στο κύριο κλειδί *'legend'*, υπάρχει το ζευγάρι κλειδιού-τιμής *fontSize: 20* που δείχνει το φόντο της λεζάντας. Στο τελευταίο κύριο κλειδί *'toolTip*, υπάρχει το ζευγάρι κλειδιού-τιμής *shared:true* που σημαίνει ότι οι δύο κυματομορφές μοιράζονται το ίδιο tooltip.

- **Συνάρτηση ενημέρωσης (στοιχεία χρήστη, στοιχεία με βάση το σχεδιάγραμμα)**

Με αυτήν την συνάρτηση μπαίνουν όλες οι τιμές που έρχονται από την βάση δεδομένων στα html elements της σελίδας. Κατά την αλλαγή χρήστη , διαγράφονται όλες οι τιμές στην καρτέλα των στοιχειών μέσα από το διάγραμμα και παίρνουν την τιμή '-' πριν υποδεχτούν τα νέα δεδομένα. Επιλέγονται τα element *Name, Surname, Age, Sports Activity* στα στοιχεία χρήστη και αλλάζει η τιμή του σύμφωνα με το JSON object που εισήχθη μέσω ajax στο αρχείο PHP 'choose_user.php' . Αυτό επιτυγχάνεται με την χρήση των εντολών *$("#name").html(v.name); $("#surname").html(v.surname) $("#age").html(v.age)* όπου *v* το JSON object. Για το πέρασμα της τιμής του *Sports Activity* εξετάζεται τι τιμή ακεραίου έχει *(*1 = 'Sedentary' , 2='Lightly Active', 3='Active', 4='Very active'*)* και αναλόγως χτίζεται το μήνυμα *activity_text $("#activity").html(activity_text).* Τα στοιχεία με βάση το σχεδιάγραμμα υπολογίζονται με τα δεδομένα JSON που ήρθαν από το αρχείο 'get_data.php'. Υπολογίζεται ο μέσος όρος των τιμών *minPr*, *maxPr* ένα άθροισμα εγγραφών και την τελευταία ημέρα

εγγραφής. Αυτό επιτυγχάνεται με την χρήση των εντολών *$("#ave_maxPr").html("Μέσος όρος maxPr") $("#ave_minPr").html("Μέσος όρος minPr") $("#last_date").html("Τελευταία ημέρα εγγραφής") $("#sum_attempts").html("άθροισμα εγγραφών").*

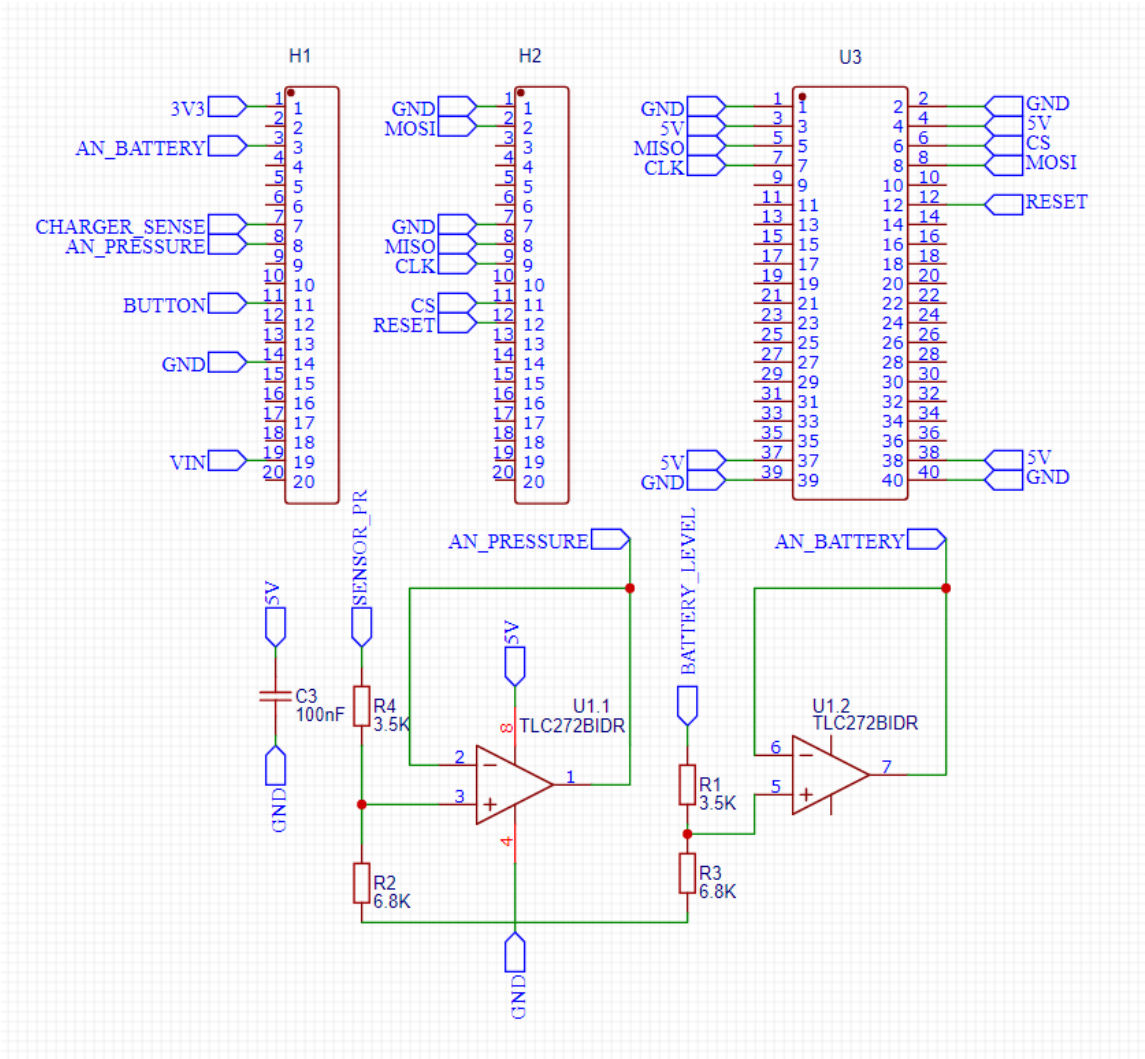
ΚΕΦΑΛΑΙΟ 6

Η ΟΛΟΚΛΗΡΩΜΕΝΗ ΣΥΣΚΕΥΗ

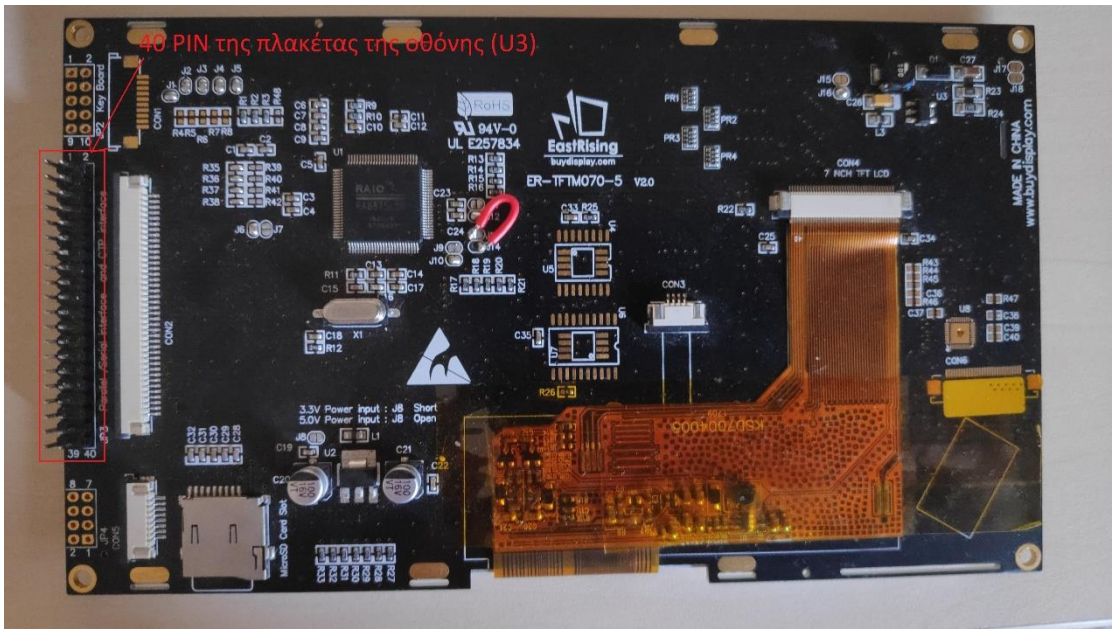
6.1 Εισαγωγή

Στο αυτό το κεφάλαιο παρουσιάζεται η συσκευή με όλα τα κομμάτια που την απαρτίζουν , και αναλύεται το καθένα ξεχωριστά με κριτήριο τον ρόλο του μέσα στο συνολικό σύστημα. Τα βασικά στοιχεία είναι η πλακέτα του αισθητηρίου πίεσης , η βασική πλακέτα του ESP32 που κατασκευάστηκε για την επικοινωνία όλων των στοιχείων, η μπαταρία του συστήματος και η πλακέτα φόρτισης-σταθεροποίησης.

6.2 Βασική στοιχεία της συσκευής



Σχήμα 6.1 Σχηματικό διάγραμμα βασικής πλακέτας του συστήματος



Σχήμα 6.2 Πλακέτα επικοινωνίας του RA8875 με την οθόνη 7 ιντσών

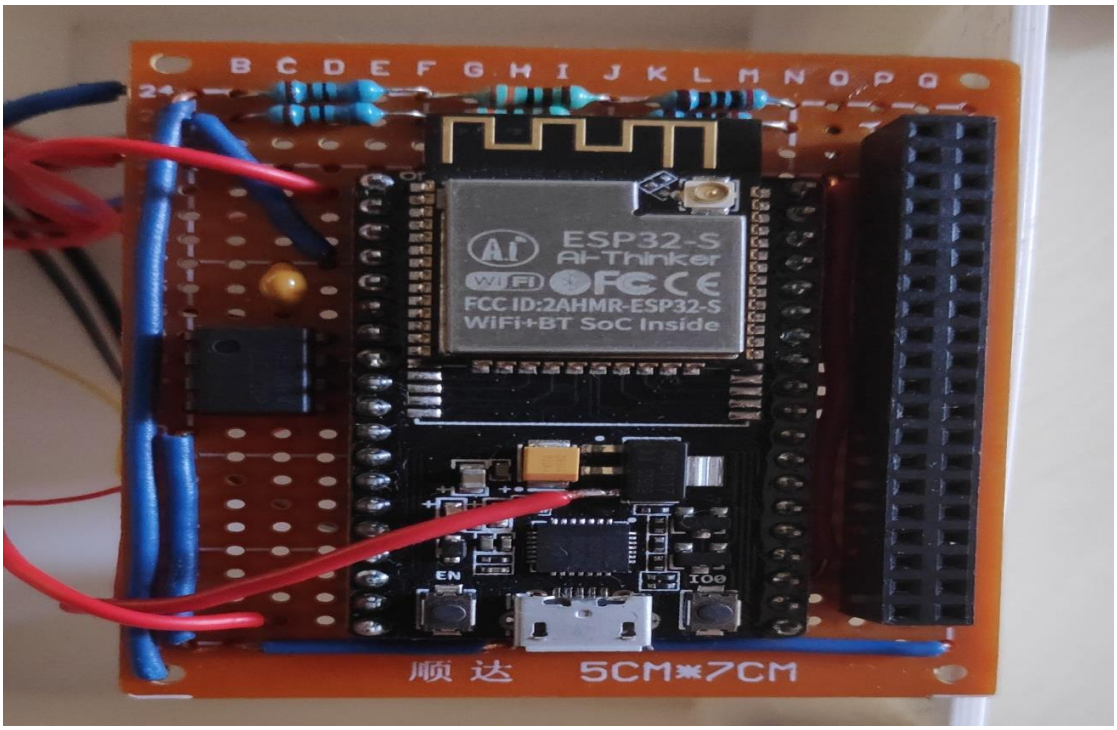
Οι σειρές των ακίδων H1, H2 αναφέρονται στα pins της αναπτυξιακής πλακέτας NodeMCU-32s όπως αναφέρθηκε στο Κεφ. 2.3. Η σειρά των 40 pins *U3* όπως φαίνεται στο Σχ. 6.2, αναφέρεται στα pins που βγάζει η υποστηρικτική πλακέτα για τον RA8875 και την οθόνη των 7 ιντσών. Η οθόνη έχει 4 pins για την τροφοδοσία των 5V και 4 pins για την σύνδεση στο GND. Ο τελεστικός ενισχυτής TLC272 χρησιμοποιείται για να ρίξει την τάση του αισθητηρίου πίεσης και της τάσης μέτρησης της μπαταρίας στα επίπεδα ανοχής του ADC (από 0 έως 3.3V) μέσω του διαιρέτη τάσης και την συνδεσμολογία follower. Το pin 'CHARGER_SENSE' συνδέεται με το GPIO32 του ESP32 και χρησιμοποιείται για να γίνει η ανίχνευση της σύνδεσης του φορτιστή. Το pin 'BUTTON' συνδέεται στο GPIO27 του ESP32 και αναφέρεται στο κουμπί έναρξης προσπάθειας του προγράμματος.

Η βασική πλακέτα του συστήματος (Σχ. 6.3) κατασκευάστηκε σε προτρυπημένη πλακέτα για λόγους κόστους και για λόγους συνεχής ανάπτυξης αυτής της συσκευής με επιπρόσθετες λειτουργίες. Τα καλώδια που χρησιμοποιήθηκαν για του αγωγούς τροφοδοσίας και γείωσης

είναι ικανοποιητικού πάχους για να ικανοποιήσουν τις ανάγκες μεταφοράς τουλάχιστον 1A, ενώ τα καλώδια μεταφοράς σήματος είναι τύπου 30AWG με διάμετρο χαλκού 0.25mm.

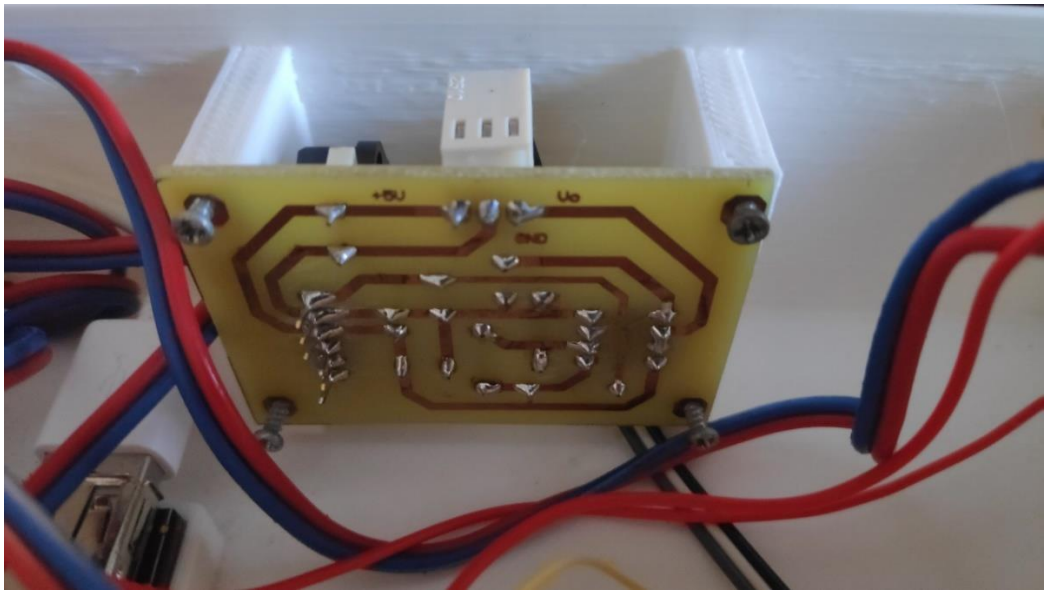
Πίνακας 6.1 Οι συνδέσεις της οθόνης με τον ESP32

ESP32 (NodeMCU-32s)	RA8875 (U3 Connector)
GPIO23 VSPI MOSI	8 *(SDI)*
GPIO19 VSPI MISO	5 *(SDO)*
GPIO18 VSPI SCK	7 (CLK Pin)
GPIO17 (CS)	6 *(CS Pin)*
GPIO16	12 *(RESET Pin)*



Σχήμα 6.3 Πλακέτα του βασικού κυκλώματος (σχηματικό διάγραμμα στο Σχ. 6.1)

Στο Σχ. 6.4 φαίνεται η πλακέτα του αισθητηρίου πίεσης , σύμφωνα με το σχηματικό που αναφέρθηκε στο Κεφ 3. στο Σχ. 3.3. Η πλακέτα δεν έγινε ενιαία με την βασική πλακέτα και συνδέεται με τρία καλώδια με έναν connector τύπου 3 pin JST. Ό λόγος που έγινε αυτό είναι διότι ο σωλήνας που συνδέει την ζώνη με το αισθητήριο πίεσης επιλέχτηκε να έχει την δυνατότητα απόσπασης, οπότε το αισθητήριο θα έπρεπε να βρίσκεται σε θέση που να έχει κοντά στα άκρα του κουτιού που περικλείει όλη την συσκευή.



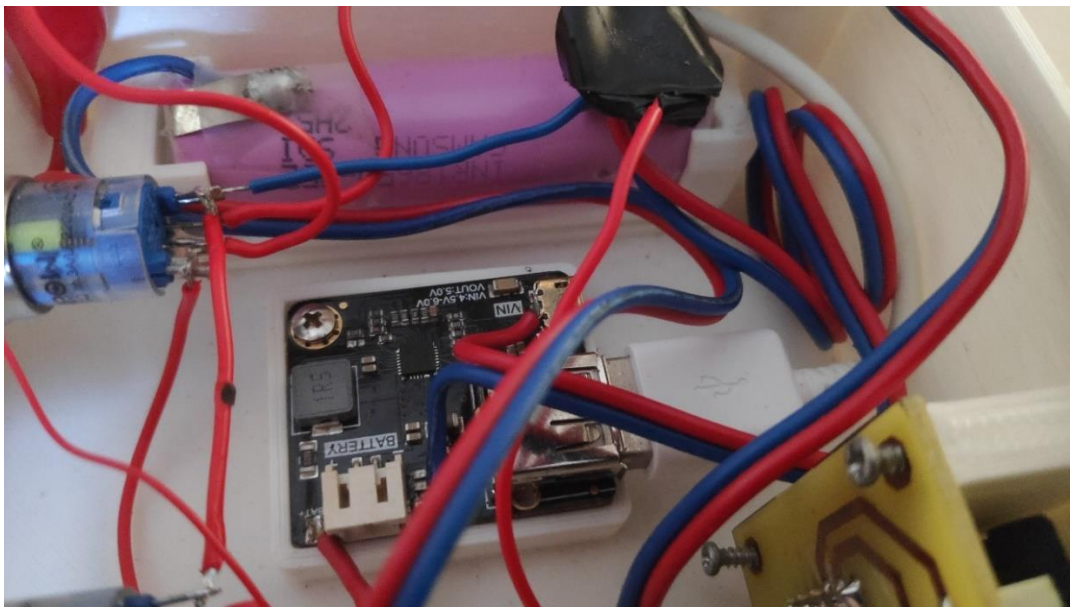
Σχήμα 6.4 Πλακέτα για το αισθητήριο πίεσης MPX5050GP (Κύκλωμα ενίσχυσης και χαμηλοπερατού φίλτρου 1ης Τάξης, σχηματικό διάγραμμα στο Σχ. 3.3)

6.3 Φορτότητα συσκευής

Η μπαταρία που χρησιμοποιείται για την συσκευή είναι τύπου Li-Ion 18650 στα 3500 mAh και το ακριβής μοντέλο της είναι το INR18650-35E με κατασκευάστρια εταιρεία την Samsung. Κάποια από τα τεχνικά χαρακτηριστικά της είναι τα εξής.

1. Μέγιστο ρεύμα φόρτισης : 2A
2. Μέγιστη τάσης εκφόρτισης : 2.65V
3. Μέγιστο ρεύμα εκφόρτισης : 8A (σε συνεχή χρήση) , 13 (στιγμιαίο ρεύμα)

Ο φορτιστής-σταθεροποιητής για την ανάγκη παροχής 5V και φόρτισης της μπαταρίας που επιλέχτηκε βασίζεται στο τσιπ MP2636 της Monolithic Power Systems. Το MP2636 έχει λειτουργία εναλλαγής ισχύος συστήματος, σχεδιασμένος για μπαταρίες Li-ion ή Li-Polymer μιας 1 cell και χρησιμοποιείται σε ένα ευρύ φάσμα φορητών εφαρμογών. Το MP2636 μπορεί να λειτουργήσει τόσο σε λειτουργία φόρτισης όσο και σε λειτουργία boost (ενίσχυσης), επιτρέποντας πλήρη διαχείριση συστήματος και διαχείριση ισχύος μπαταρίας. Όταν συνδέεται ο φορτιστής στο σύστημα , η το τσιπ λειτουργεί σε κατάσταση φόρτισης. Ανιχνεύει αυτόματα την τάση της μπαταρίας και φορτίζει την μπαταρία σε τρεις φάσεις: trickle current, constant current (φάση σταθερού ρεύματος) , constant voltage (φάση σταθερής τάσης). Άλλα χαρακτηριστικά περιλαμβάνουν τον τερματισμό φόρτισης στα 4.2V και την αυτόματη επαναφόρτιση. Αυτό το τσιπ επίσης δίνει προτεραιότητα ρεύματος στο φορτίο και μετά στην φόρτιση της μπαταρίας. Στην περίπτωση που ο φορτιστής δεν είναι συνδεδεμένος , το MP2636 μεταβαίνει σε λειτουργία boost. Το MP2636 επιτρέπει επίσης προστασία βραχυκυκλώματος στην έξοδο και αποσυνδέσει πλήρως την μπαταρία από το φορτίο σε περίπτωση ύπαρξης βραχυκυκλώματος. Η κανονική λειτουργία θα ανακάμψει μόλις αφαιρεθεί το σφάλμα βραχυκυκλώματος. Η πλακέτα MP2636 που τοποθετήθηκε στο σύστημα κατασκευάστηκε από την εταιρεία DFRobot. Η συσκευή (οθόνη και ESP32) σε λειτουργεία τραβάνε από την μπαταρία 670mA κατά μέσο όρο μαζί με τις απώλειες του Boost του MP2636. Αυτό δίνει αυτονομία στην συσκευή κοντά στις 8 – 9 συνεχόμενης λειτουργίας.



Σχήμα 6.5 Πλακέτα φόρτισης-σταθεροποίησης στα 5V και η μπαταρία Samsung INR18650

ΚΕΦΑΛΑΙΟ 7

ΣΥΜΠΕΡΑΣΜΑΤΑ

Ο σχεδιασμός, ο προγραμματισμός και η υλοποίηση της εργασίας για την κατασκευή μιας συσκευής για την μέτρηση της διαφραγματικής αναπνοής ολοκληρώθηκε επιτυχώς. Οι κύριοι στόχοι που τέθηκαν από την αρχή ήταν η εύρεση ενός μαθηματικού τρόπου για να γίνει η μέτρηση, η εύρεση κατάλληλων γραφημάτων για την απεικόνιση της μέτρησης με κύριο κριτήριο την οπτική ανατροφοδότηση, η ασύρματη επικοινωνία με έναν ηλεκτρονικό υπολογιστή για την μεταφορά των δεδομένων και την απεικόνιση τους , η αποθήκευση των δεδομένων. Χρειάστηκαν πολλά εργαλεία για να συνδράμουν στην δημιουργία αυτού του αποτελέσματος και οι προκλήσεις ήταν μεγάλες. Το κομμάτι του προγραμματισμού ήταν πολύ απαιτητικό και ειδικά για τον κώδικα που γράφτηκε στο ESP32 , ο οποίος θα έπρεπε να συνδυάσει πολλές λειτουργείες ταυτόχρονα και όλες αυτά να λειτουργήσουν άρτια μεταξύ τους. Έγινε χρήση βοηθητικών βιβλιοθηκών για αυτόν στον στόχο που στις περισσότερες χρειάστηκαν πολλές παρεμβάσεις και παραμετροποιήσεις για να υπάρχει η επιθυμητή λειτουργία του συστήματος, κάτι που απαιτούσε την εν βάθος ενασχόληση και κατανόησή τους. Το κόστος των εξαρτημάτων που χρησιμοποιήθηκαν ήταν χαμηλό για αυτά που πρόσφεραν, το πιο ακριβό συστατικό ήταν η οθόνη εφτά ιντσών κάτι που ήταν αναμενόμενο.

Στο θέμα των πλακετών της συσκευής, για λόγους επερχόμενης ανάπτυξης δεν κατασκευάστηκε μια ενιαία πλακέτα επαγγελματικού τύπου για την σύνδεση όλων των στοιχείων. Η εξέλιξη της συσκευής σε θέματα λογισμικού η υλικών θα συνεχίζεται εφόσον υπάρχει η απαραίτητη ζήτηση και η αναγνώριση της βοήθειας που προσφέρει. Πάντα υπάρχει χώρος για βελτιώσεις και οι προτάσεις τρίτων είναι επιθυμητές. Κάποιες βελτιώσεις για μελλοντική αναβάθμιση που εξετάστηκαν μετά το πέρας της ολοκλήρωσης της κατασκευής παρατίθενται παρακάτω.

- Απομακρυσμένη αποστολή δεδομένων μέσω Cloud Service, ώστε η συσκευή και ο υπολογιστής να μην βρίσκονται στο ίδιο τοπικό δίκτυο όπως συμβαίνει στον τρόπο που υλοποιήθηκε.
- Εμφάνιση περισσότερων μνημάτων στην οθόνη εφτά ιντσών κατά την διαδικασία μέτρησης, με αποτέλεσμα ο χρήστης να προσαρμοστεί ανάλογα και να γίνει η εξοικείωση γρηγορότερα.
- Δυνατότητα αλλαγής κάποιων βασικών μεταβλητών για την βοήθεια της μέτρησης στο πρόγραμμα του ESP32 μέσω της web εφαρμογής και αποθήκευση τους για μελλοντική χρήση.
- Σε περίπτωση αδράνειας , ενεργοποίηση της δυνατότητας βαθύ ύπνου στην οθόνη και το ESP32 με σκοπό την μείωση της άσκοπης κατανάλωσης ενέργειας.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Anraku M, Shargall Y, *Surgical conditions of the diaphragm: anatomy and physiology*. Thorac Surg Clin, 2009.

[2] Arráez-Aybar LA, González-Gómez CC, Torres-García AJ, Morgagni-Larrey, *Parasternal diaphragmatic hernia in the adult*. 2009.

[3] J Cardiopulm Rehabil. *Breathing retraining in chronic obstructive pulmonary disease.* Breslin EH, 1995.

[4] Cahalin LP, Braga M, Matsuo Y, Hernandez ED, *Efficacy of diaphragmatic breathing in persons with chronic obstructive pulmonary disease: a review of the literature. J Cardiopulm Rehabil*, 2002.

[5] Gosselink R, *Breathing techniques in patients with chronic obstructive pulmonary disease (COPD)*. Chron Respir Dis, 2004.

[6] Fernandes M, Cukier A, Feltrim MIZ, *Efficacy of diaphragmatic breathing in patients with chronic obstructive pulmonary disease*. Chron Respir Dis, 2011.

[7] Lewis LK, Williams MT, Olds T, *Short-term effects on outcomes related to the mechanism of intervention and physiological outcomes but insufficient evidence of clinical benefits for breathing control: a systematic review*. Aust J Physiother, 2007.

[8] Taisa Daiana da Costa, Maria de Fatima Fernandes Vara, Camila Santos Cristino, Tyene Zoraski Zanella, Guilherme Nunes Nogueira Neto and Percy Nohama, *Breathing Monitoring and Pattern Recognition with Wearable Sensors*, 2019.

[8] "Strain Gauges Chapter 9 - Electrical Instrumentation Signals,", [Online]. Available: https://www.allaboutcircuits.com/textbook/direct-current/chpt-9/strain-gauges/

[9] Seong Won Park, Partha Sarati Das, Ashok Chhetry, and Jae Yeong Park *, A Flexible Capacitive Pressure Sensor for Wearable Respiration Monitoring System* , Article in IEEE Sensors Journal, DOI: 10.1109/JSEN.2017.2749233 , 2017.

[10] Haijuan Liu, Shaopeng G2, Sujie Chen3 ,Tsugumi Kuramoto-Ahuja, Tamae Sato, Junichiro Kaneko, Ko Onoda and Hitoshi Maruyama, *Application of Using a Wearable Strain Sensor on Respiratory Evaluation in Physiotherapy-Changing with Aging in Breathing Pattern*, 2018.

[11] Espressif Systems, "*This document provides the specifications of ESP32 family of chips", ESP32 datasheet , 2020*

[12] Freescale semiconductor, "*Integrated Silicon Pressure Sensor On-Chip Signal Conditioned, Temperature Compensated and Calibrated*", MPX5050GP datasheet, 2010.

[13] Wikipedia, "*SPI protocol*" [Online]. Available: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface.

[14] RAiO Technology Inc, "*RAiO RA8875 Character/Graphic TFT LCD Controller",* RA8875 , 2012.

[15] Wikipedia, "*Internet protocol suite*" [Online]. Available: https://en.wikipedia.org/wiki/Internet_protocol_suite.

[16] Wikipedia, "*Dynamic Host Configuration Protocol*" [Online]. Available: https://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol.

[17] Wikipedia, "Websocket" [Online]. Available: https://en.wikipedia.org/wiki/WebSocket.

[18] "*What Is Apache Web Server? A Basic Look at What It Is and How It Works*" [Online]. Available: https://kinsta.com/knowledgebase/what-is-apache/ , 2019.

[19] Wikipedia, "Javascript" [Online]. Available: https://en.wikipedia.org/wiki/JavaScript.

[20] Wikipedia, "HTML" [Online]. Available: https://el.wikipedia.org/wiki/HTML.

[21] Wikipedia, "CSS" [Online]. Available: https://el.wikipedia.org/wiki/CSS.

[22] MPS, "*MP2636 3.0A Single Cell Switch Mode Battery Charger with Power Path Management (PPM) and 3.0A System Boost Current*" MP2636 , 2018.

[23] SΛMSUNG, "*for Lithium-ion rechargeable cell Model name : INR18650-35E*" , Battery INR18650 , 2015.

ΠΑΡΑΡΤΗΜΑ

Παράρτημα 1. Κώδικας του ESP32 σε C++

```
#include <esp_adc_cal.h>
#include <SPI.h>
#include <RA8875.h>

#include "ESPAsyncWebServer.h"
#include "ArduinoJson.h"

AsyncWebServer server(80);
AsyncWebSocket ws("/ws");

IPAddress apIP(192, 168, 1, 1);

const char *ssid = "***";
const char *password = "***";
const char *hostname = "***";

int websock_start=0;
int websock_relax=0;
int websock_measuring=0;

RA8875 tft = RA8875(17, 16);  //CSS PIN , RESET PIN

int Interval = 5;
unsigned long previousMillis = 0;

#define REF_VOLTAGE 1098//1115
esp_adc_cal_characteristics_t *adc_chars = new esp_adc_cal_characteristics_t;

int sensorV = 0;
int old_sensorV, temp[501], i, j;
int flagsvisimo = 0;
int x, y = 0;
int y2;
int y_1;
int start = 1;
int count = 0;
int x_mpalas, old_xmpalas, y_mpalas, old_ympalas;
float t;
unsigned int b_colour = 0xF800;
unsigned int old_b_colour;
unsigned int plot_colour = RA8875_CYAN;
int ii, jj;
float tempr, Prfm;
float b[5] = {1.8322e-4, 7.3286e-4, 10.993e-4, 7.3286e-4, 1.8322e-4};
float a[5] = {1, -3.344068, 4.238864, -2.409343, 0.517478};
float Prk[5] = {0, 0, 0, 0, 0};
```

```
    float yk[5] = {0, 0, 0, 0, 0};
    float m_med[5] = {0, 0, 0, 0, 0};
    float sort_med[5] = {0, 0, 0, 0, 0};
    float Prf;
    float Pr1;
    float Pres;

    int Option=0;

    int mult = 2;
    float unit_P = 7.5;
    int Error = 200;
    float Vs = 5;
    float Pr_Min[2] = {50,45};
    float Pr_Max[2] = {90,120};
    float shakes = 3;
    float threshold = 0.4 * unit_P;
    float kPa_min_ss = 57;
    float kPa_max_ss = 63;
    float min_max_v = 112.5;

    int N = 3;
    int C_end = 2 * 100 * mult;
    int C_relax = 1.5 * 100 * mult;
    int max_meas_time = 12.5 * 100 * mult;

    int I1;
    int I1_1 = 0;
    int attempt = 0;
    int end_time = 0;
    int relax_time = 0;
    float relaxPr = 0;
    float minPr = min_max_v;
    float maxPr = 0;
    int C_I1 = 0;
    int ind_shake = 0;
    float Tbl[3][5] = {{0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}};


    int flagin = 0;
    float Balloon;
    float Balloon1;
    int reset_flag;
    int yolo = 0;
    int error_flag = 0;
    int reset_times = 0;
    int hi;
    unsigned int colors[4] = {RA8875_CYAN, RA8875_MAGENTA, RA8875_BLUE, RA8875_CYAN};
//0x7FFFD4 πορτοκαλί
    int y11;
    int draw_plot;
```

```
float sensorV_1 = 0;
float old_sensorV_1;
int I2;
int old_I2;
int chrg = 20;
int counter1s = 0;
int counter500ms =0;
int sumbat = 0;
int countbat = 0;
int battery_state =0;
int old_battery_state =0;

int up_Pres;
int down_Pres;
int balance_Pres;

float Vout;

bool wifi_is_enabled = false;

void onWsEvent(AsyncWebSocket * server, AsyncWebSocketClient * client, AwsEventType type,
void * arg, uint8_t *data, size_t len){
  if(type == WS_EVT_CONNECT){
    Serial.printf("ws[%s][%u] connect\n", server->url(), client->id());
    //client->printf("Hello Client %u :)", client->id());
    //client->ping();
    Option=0; reset_times=0; reset();

  } else if(type == WS_EVT_DISCONNECT){
    Serial.printf("ws[%s][%u] disconnect\n", server->url(), client->id());
  } else if(type == WS_EVT_ERROR){
    Serial.printf("ws[%s][%u] error(%u): %s\n", server->url(), client->id(), *((uint16_t*)arg),
(char*)data);
  } else if(type == WS_EVT_PONG){
    Serial.printf("ws[%s][%u] pong[%u]: %s\n", server->url(), client->id(), len, (len)?(char*)data:"");
  } else if(type == WS_EVT_DATA){
    String msg = "";
    msg.reserve(len + 1);
    Serial.println ("Websocket Data");
    for (size_t i = 0; i < len; i++) {
      msg += (char)data[i];
    }

    if (msg.startsWith("reset")) {
      reset_times=0; reset();
    }
    if (msg.startsWith("start")) {
      websock_start=1;
    }
    if (msg.startsWith("mode0")) {
      Option=0;
      reset_times=0;
```

```
    reset();
   }
   if (msg.startsWith("mode1")) {
    Option=1;
    reset_times=0;
    reset();
   }

  }
}

void jsonDom(/*AsyncWebSocketClient *client */) {
 String json;

 Serial.println(ESP.getMaxAllocHeap());
 DynamicJsonDocument document(30000); //30000

 hi = 250;
 int itemx=0;
  while (temp[hi - 250] != 0) {
   JsonObject item = document.createNestedObject();
   item["x"] = itemx*0.001 * 25;

   int y2toPress = temp[hi - 250];
   int itemy;

   if ( y2toPress < 300 )
    itemy = map( y2toPress, 300 , 186 , 60 , Pr_Max[Option] );
   else
    itemy = map( y2toPress, 301 , 415 , 60 , Pr_Min[Option] );

  item["y"] = itemy; //415 - temp[hi - 250];

  temp[hi - 250] = 0; //ERASE TEMP DATA AFTER SEND TO WEB

  hi++;
  itemx++;
 }

 serializeJson(document, json);
 document.shrinkToFit();
 ws.textAll(json);
 document.clear();

}

void jsonAttempts() {

 String json;
 DynamicJsonDocument document(500);
 JsonObject item = document.createNestedObject();
 for (int i=1;i<=4;i++){
```

```
  String maxPr = "maxPr"+String(i);
  String minPr = "minPr"+String(i);
  item[maxPr]=Tbl[1][i];
  item[minPr]=Tbl[2][i];

 }
 item["attempt"]=attempt;

 serializeJson(document, json);
 document.shrinkToFit();
 ws.textAll(json);
 document.clear();

}

void jsonMessage(String cur) {

 String json;
 DynamicJsonDocument document(150);
 JsonObject item = document.createNestedObject();

 if (cur=="reset") {
  item[cur]=cur;
 }
 else{
  item["situation"]=cur;
 }

 serializeJson(document, json);
 document.shrinkToFit();
 ws.textAll(json);
 document.clear();

}

void jsonOption(int option) {

 String json;
 DynamicJsonDocument document(150);
 JsonObject item = document.createNestedObject();

 item["mode"]=option;

 serializeJson(document, json);
 document.shrinkToFit();
 ws.textAll(json);
 document.clear();

}

void setup() {
```

```
  Serial.begin(115200);

  analogReadResolution(12);
  esp_adc_cal_value_t val_type = esp_adc_cal_characterize(ADC_UNIT_1, ADC_ATTEN_DB_11,
ADC_WIDTH_BIT_12, REF_VOLTAGE, adc_chars);

  tft.begin(RA8875_800x480);

  //SPI.beginTransaction(SPISettings(8000000, MSBFIRST, SPI_MODE0));
  tft.useLayers(true);//turn on layers
  tft.writeTo(L1);
  pinakas(Pr_Max[Option],Pr_Min[Option]);
  pinakas2_kai_mpataria();
  tft.writeTo(L2);//from this point we write on layer 2
  tft.layerEffect(OR);

  old_sensorV = 0;
  old_xmpalas = 55;
  old_ympalas = 55;
  old_b_colour = 0xF800;



  pinMode(27, INPUT_PULLDOWN);
  pinMode(32, INPUT_PULLDOWN);

  old_sensorV_1 = 0;
  old_I2 = 0;

}

void loop() {

  if( millis() - previousMillis >= Interval || start==1){
      previousMillis  = millis();

   float Vs5 = 4.56;
   float Vs3 = 3.25;
   float error = 0.2;
   float Voutc;

   Prf = filtering_rawADC(33);
   Vout = analogRead_cal(Prf,33, ADC_ATTEN_DB_11) ;
   Pr1 =  (( 7.5 * (Vout - 0.04*Vs5 + error*0.018*Vs5) / (0.018*Vs5) ) / 2 ) *  (Vs5/(Vs3)); //SAME
  if (Pr1 < Pr_Min[Option])
    Pres = Pr_Min[Option];
  else if (Pr1 > Pr_Max[Option])
    Pres = Pr_Max[Option];
  else
    Pres = Pr1;
```

```
  Voutc = Pres_to_Volt ( Pr1 );
  //Serial.print("Volt Sensor "+String(Vout));
  //Serial.print(" ADC value "+String(Prf));
  //Serial.print(" Voutc "+String(Voutc));
  //Serial.println(" Pressure "+String(Pr1));

  mainMeasurement();
  if (start==1) {
   start=0;
   y2 = 250;
   y_1=y2;
   x=50;
  }
 }

 main_func();
 print_sensorV();
 plot();
 mpataria();
 ball();

 if (!wifi_is_enabled) {
  //WiFi.setTxPower(WIFI_POWER_MINUS_1dBm);
  start_wifi();
 // WiFi.setTxPower(WIFI_POWER_MINUS_1dBm);
  wifi_is_enabled = true;
 }

}

void main_func(void) {

 I1 = digitalRead(27);
 I2 = digitalRead(32);

 if (Tbl[2][attempt] == -1) {
  plot_colour = colors[attempt - 1];
 }

//tft.setCursor(500, 161);
//tft.println(Balloon,2);

 if (reset_flag == 1) {
  jsonMessage("reset");

  if (Tbl[1][1] != 0) {
   tft.fillRoundRect(130, 80, 45, 15, 2, RA8875_BLACK);
   tft.fillRoundRect(130, 113, 45, 15, 2, RA8875_BLACK);
  }
  if (Tbl[1][2] != 0) {
   tft.fillRoundRect(225, 80, 45, 15, 2, RA8875_BLACK);
   tft.fillRoundRect(225, 113, 45, 15, 2, RA8875_BLACK);
```

```
        }
        if (Tbl[1][3] != 0) {
          tft.fillRoundRect(225 + 97, 80, 45, 15, 2, RA8875_BLACK);
          tft.fillRoundRect(225 + 97, 113, 45, 15, 2, RA8875_BLACK);
        }
        if (Tbl[1][4] != 0) {
          tft.fillRoundRect(225 + 97 + 85, 80, 45, 15, 2, RA8875_BLACK);
          tft.fillRoundRect(225 + 97 + 85, 113, 45, 15, 2, RA8875_BLACK);
        }

        for (int rt = 0; rt <= 2; rt++) { //Σβήσιμο Tbl , όλα μηδενικά
          for (int ry = 0; ry <= 4; ry++) {
            Tbl[rt][ry] = 0;
          }
        }

        if (start == 0) {
          tft.writeTo(L1);
          tft.clearScreen();
          tft.fillRoundRect(50, 185, 700, 230, 4, RA8875_BLACK);
          pinakas(Pr_Max[Option],Pr_Min[Option]);
          pinakas2_kai_mpataria();
          tft.writeTo(L2);
          x = 50;
        }

        //AFTER RESET
        tft.setCursor(262+80, 161);
        tft.setTextColor(RA8875_RED,RA8875_BLACK);
        if (Option==0) tft.print("Beginner");
        else tft.print("Advanced");
       //tft.fillScreen(RA8875_BLACK);

      }

      if ( (error_flag == 1  && Tbl[2][attempt] == -1) || (end_time == C_end && (relax_time != 0 ||
(Tbl[2][4] != 0)) && maxPr != 1) || reset_flag == 1 || start == 1 ) {

        jsonMessage("Waiting to start");
        int reason;
        if (error_flag == 1  && Tbl[2][attempt] == -1) reason = 1;
        if (end_time == C_end && (relax_time != 0 || (Tbl[2][4] != 0)) && maxPr != 1) reason = 2;
        if (reset_flag == 1) reason = 3;
        if (start==1) reason = 4;


        Serial.print("I'm in here for reason number "+String(reason));

        Serial.print(" draw_plot before IF "+String(draw_plot));

        if (reset_flag != 1 && error_flag != 1 && start != 1)
          draw_plot = 1; //Draw the saved plot attempt
```

```
        Serial.println(" draw_plot after IF "+String(draw_plot));

       plot_colour = colors[attempt];

        if (relax_time != 0 && Tbl[2][attempt] != 0  && Tbl[2][attempt] != -1)
          relax_time = 0;

        if (reset_flag == 1 && Tbl[2][4] == 0) {
          reset_flag = 0; plot_colour = colors[0]; attempt = 0;
        }

        if (error_flag == 1 && Tbl[2][attempt] == -1) {
          relax_time = 0; error_flag = 0; Serial.println("I'm in error_flag "+String(error_flag));
        }

        if (attempt == 1 || attempt == 0) {
          tft.setTextColor(RA8875_CYAN, RA8875_BLACK);
          tft.setCursor(132, 80);
          tft.println(Tbl[1][1], 1);
          tft.setCursor(132, 113);
          tft.println(Tbl[2][1], 1);
        }

        if (attempt == 2 || attempt == 0) {
          tft.setTextColor(RA8875_MAGENTA, RA8875_BLACK);
          tft.setCursor(227, 80);
          tft.println(Tbl[1][2], 1);
          tft.setCursor(227, 113);
          tft.println(Tbl[2][2], 1);
        }

        if ((Tbl[2][4] != 0 && attempt == N && maxPr != 0 && maxPr != 1) || (attempt == N && maxPr
== Error) || attempt == 0) {
          tft.setTextColor(RA8875_BLUE, RA8875_BLACK);
          tft.setCursor(227 + 97, 80);
          tft.println(Tbl[1][3], 1);
          tft.setCursor(227 + 97, 113);
          tft.println(Tbl[2][3], 1);
          tft.setTextColor(RA8875_YELLOW, RA8875_BLACK);
          tft.setCursor(227 + 97 + 85, 80);
          tft.println(Tbl[1][4], 1);
          tft.setCursor(227 + 97 + 85, 113);
          tft.println(Tbl[2][4], 1);

          if (Tbl[2][4] != 0 && attempt == N && maxPr != 0) {
            maxPr = 0;
          }
        }
```

```
        if ( (attempt < N) || (attempt == N && Tbl[2][4] != 0 && maxPr != 1) || (attempt == N &&
maxPr == Error) ) {
          if (flagin == 1) {
            tft.fillEllipse(600, 80, old_xmpalas, old_ympalas , RA8875_BLACK); //Σβήστρα για την μπάλα
μετά την μέτρηση
            flagin = 0;
          }
          b_colour = RA8875_RED;
          tft.setTextColor(RA8875_RED, RA8875_BLACK);
          tft.setCursor(350, 450);
          tft.print("Press Button");
          Balloon = 0;
          if (attempt == N && maxPr != Error)
            maxPr = 1;
        }

        if (relax_time == 0 && attempt == 0) {
          tft.fillRoundRect(50, 185, 700, 230, 4, RA8875_BLACK);
        }
        tft.fillRect(50, 185, 1, 415 - 185, RA8875_BLACK);
        tft.fillRect(51, 185, 1, 415 - 185, RA8875_BLACK);
        tft.fillRect(52, 185, 1, 415 - 185, RA8875_BLACK);
        tft.fillRect(53, 185, 1, 415 - 185, RA8875_BLACK);
        tft.fillRect(54, 185, 1, 415 - 185, RA8875_BLACK);

        if (Tbl[2][attempt] == -1 && draw_plot != 1) {     //EXTRA && draw_plot != 1
          tft.fillRoundRect(250, 185, 600, 230, 4, RA8875_BLACK);
          hi = 250;
          while (temp[hi - 250] != 0) {
            temp[hi - 250] = 0;
            hi++;
          }
          x = 50;
          // relax_time=0;
        }

        tft.fillEllipse(600, 80, 55, 55, b_colour);



      }


      if (relax_time == 0) {
        if (x >= 149) {
          x = 50;
          if (flagsvisimo == 0)
            flagsvisimo = 1;
        }
      }
```

```
  else if (relax_time > 0 && relax_time <= C_relax) {
    if (relax_time == 1) jsonMessage("Relax");
   if (x == 50) {
    tft.fillRect(50, 185, 1, 415 - 185, RA8875_BLACK);
    tft.fillRect(51, 185, 1, 415 - 185, RA8875_BLACK);
    tft.fillRect(52, 185, 1, 415 - 185, RA8875_BLACK);
    tft.fillRect(53, 185, 1, 415 - 185, RA8875_BLACK);
    tft.fillRect(54, 185, 1, 415 - 185, RA8875_BLACK);
   }
   if (relax_time < 5)
    tft.fillRoundRect(350, 450, 100, 15, 2, RA8875_BLACK);
   if (relax_time < 10) {
    b_colour = RA8875_YELLOW;
    tft.setTextColor(RA8875_YELLOW, RA8875_BLACK);
    tft.setCursor(380, 450);
    tft.print("Relax");
    Balloon = 0;
   }
   if (x >= 245) x = 150;
  }
  else {
   if (relax_time < C_relax + 5) {
    if (relax_time == C_relax + 1) jsonMessage("Measuring");
    b_colour = 0x07E0;
    tft.setTextColor(RA8875_GREEN, RA8875_BLACK);
    tft.setCursor(365, 450);
    tft.print("Measuring");
    x = 250;
    tft.fillRect(50, 185, 200, 415 - 185, RA8875_BLACK);
   }
   flagin = 1;
   if  (attempt > 1) {
    //Balloon = 2 * (Pres - relaxPr / C_relax) / Tbl[1][1];
    Balloon = 2 * (Pres - relaxPr / C_relax) / 20;
    if (Balloon > 2.3 ) Balloon = 2.3; //MAX VALUE CUT
    if (maxPr == Error) Balloon = 0;
   }
   else {
    Balloon1 = (Pres - relaxPr / C_relax) / (0.5 * unit_P);  //Configured was 1.5*unit_P
    if (Balloon1 < 2 && Balloon1 > -1)
     Balloon = Balloon1;
    else if (Balloon1 > 2)
     Balloon = 2;
    else
     Balloon = -1;
   }
  }



}
```

```
void print_sensorV(void) {
  if ((sensorV != old_sensorV) || (start == 1)  ) {
    tft.setTextColor(RA8875_WHITE, RA8875_BLACK);
    tft.setCursor(699, 161);
    tft.println(Pres-60, 1);
    old_sensorV = sensorV;
  }
}



void plot(void) {


  if (x < 245) {
    if (x == 145 && relax_time == 0)
      tft.fillRect(50, 185, 5, 415 - 185, RA8875_BLACK);

    if (x == 240 && relax_time <= 400)
      tft.fillRect(150, 185, 5, 415 - 185, RA8875_BLACK);

    if (x != 50)
      tft.fillRect(x + 5, 185, 1, 415 - 185, RA8875_BLACK);

  }


   //DRAW OLD PLOT
  if (attempt > 0 && relax_time == 0 && draw_plot == 1) {
    Serial.println("Drawing old PLOT");
    tft.writeTo(L1);
    y11 = temp[0];
    hi = 250;
    while (temp[hi - 250] != 0) {
      tft.drawLine(hi, y11, hi + 1, temp[hi - 250], colors[attempt - 1]);
      y11 = temp[hi - 250];
      //temp[hi - 250] = 0;
      hi++;
    }
    tft.writeTo(L2);
    draw_plot = 0;
    x = 50;

    jsonDom();   //SEND DATA TO WEB
    jsonAttempts();
  }
```

```
 tft.drawLine(x, y_1, x + 1, y2, plot_colour);

 if (x >= 250)
  temp[x - 250] = y2;
 y_1 = y2;

}


void pinakas(int up_Pres1, int down_Pres1) {
 //Πίνακας
 tft.drawRect(50, 185, 700, 230, RA8875_WHITE);
 tft.drawRect(680, 155, 70, 30, RA8875_WHITE);

 for (i = 150; i <= 750-10; i = i + 100) {  //-10 to not DRAW the LAST LINE
  for (j = 185; j <= 415; j = j + 30) {
   tft.drawFastVLine(i, j, 20, RA8875_WHITE);
  }
 }

 for (i = 57; i <= 730; i = i + 30) {
  tft.drawFastHLine(i, 300, 20, RA8875_WHITE); //300 η γραμμή στο κέντρο
 }


 tft.setTextColor(RA8875_WHITE);
 tft.setCursor(60, 160);
 tft.print("Pressure(mmHg)");

 tft.setCursor(29, 175);
 tft.println(up_Pres1 - 60, DEC);
 tft.setCursor(29-8, 405);
 tft.println(-(60-down_Pres1), DEC);
 tft.setCursor(35, 290);
 tft.println(0, DEC);

 t = 0;
 for (i = 250; i <= 750; i = i + 100) {
  if (t<10)
  tft.setCursor(i - 15, 418);
  else
  tft.setCursor(i - 20, 418);
  if (t!=0)
  tft.println(t, 1);
  else{
  tft.setCursor(i - 4, 418);
  tft.println(t,0);
  }
  t = t + 2.5;
 }

 tft.setCursor(690, 438);
```

```
  tft.print("Time(S)");

  tft.setCursor(220+80, 161);
  tft.print("Mode:");

}




void ball (void) {

  if (start == 1) {
    x_mpalas = 55;
    y_mpalas = 55;
    tft.fillEllipse(600, 80, x_mpalas, y_mpalas, b_colour );
    //start = 0;
  }
  if (b_colour != old_b_colour || Balloon != 0) {

    x_mpalas = 55 + int(Balloon * 20); //Callibration poso tha megalonei to X , 20*φορές το Balloon
    y_mpalas = 55 + int(Balloon * 20) / 6;

    if (x_mpalas < 1)
      x_mpalas = 1;

    if (x_mpalas < old_xmpalas)
      tft.drawEllipse(600, 80, old_xmpalas, old_ympalas , RA8875_BLACK);
    // Delete extra pixels left if we have a fast change 'START'


     int fullLoops = 2;
    if ( old_xmpalas - x_mpalas > 2 ) { //SUDDEN CHANGE
         fullLoops = 15;
      }
    for (int ci=1;ci<=fullLoops;ci++){
      if (x_mpalas < old_xmpalas ) {   //if (x_mpalas < old_xmpalas + 1){
        tft.drawEllipse(600, 80, old_xmpalas + ci, old_ympalas + ci , RA8875_BLACK);
       }
    }


    // Delete extra pixels left if we have a fast change 'END'
    tft.fillEllipse(600, 80, x_mpalas, y_mpalas, b_colour);
    old_xmpalas = x_mpalas;
    old_ympalas = y_mpalas;

    old_b_colour = b_colour;
  }
}
```

```
float filtering_rawADC( int adcPin ) {

   for (ii = 4; ii >= 1; ii--) {
    m_med[ii] = m_med[ii - 1];
   }
   m_med[0] = analogRead(adcPin);
   for (ii = 0; ii <= 4; ii++) {
    sort_med[ii] = m_med[ii];
   }
   for (jj = 0; jj <= 3; jj++) {
    for (ii = jj ; ii <= 4; ii++) {
     if ( sort_med[jj] > sort_med[ii] ) {
      tempr = sort_med[jj];
      sort_med[jj] = sort_med[ii];
      sort_med[ii] = tempr;
     }
    }
   }
   Prfm = sort_med[2];
   for (ii = 4; ii >= 1; ii--) {
    Prk[ii] = Prk[ii - 1];
    yk[ii] = yk[ii - 1];
   }
   Prk[0] = Prfm;
   yk[0] = 0;
   for (ii = 0; ii <= 4; ii++) {
    yk[0] = yk[0] + b[ii] * Prk[ii] - a[ii] * yk[ii];
   }
   Prf = yk[0];

  return Prf;
}

float analogRead_cal(int analValue , uint8_t channel, adc_atten_t attenuation) {
 adc1_channel_t channelNum;
 analogSetCycles(10);
 switch (channel) {
  case (36):
   channelNum = ADC1_CHANNEL_0;
   break;

  case (39):
   channelNum = ADC1_CHANNEL_3;
   break;

  case (34):
   channelNum = ADC1_CHANNEL_6;
   break;

  case (35):
   channelNum = ADC1_CHANNEL_7;
```

```
    break;

   case (32):
    channelNum = ADC1_CHANNEL_4;
    break;

   case (33):
    channelNum = ADC1_CHANNEL_5;
    break;
  }
  adc1_config_channel_atten(channelNum, attenuation);

  if (channelNum==ADC1_CHANNEL_5) {
   return float(esp_adc_cal_raw_to_voltage(analValue, adc_chars)*0.001);
  }
  else
   return esp_adc_cal_raw_to_voltage(analValue, adc_chars);
}


void pinakas2_kai_mpataria(void) {

  tft.drawRoundRect(15, 10, 45, 20, 6, RA8875_RED); //Μπαταρία
  tft.fillRoundRect(59, 13, 4, 14, 2, RA8875_RED); //Μπαταρία

  tft.drawRect(95, 38, 375, 100, RA8875_WHITE);
  for (i = 189; i <= 470; i = i + 94) {
   tft.drawFastVLine(i, 38, 100, RA8875_WHITE);
  }
  for (float b = 72.5; b <= 138.5; b = b + 34.5) {
   tft.drawFastHLine(95, b, 375, RA8875_WHITE);
  }

  tft.setTextColor(RA8875_WHITE);
  tft.setCursor(137, 45);
  tft.println(1, DEC);
  tft.setCursor(231, 45);
  tft.println(2, DEC);
  tft.setCursor(327, 45);
  tft.println(3, DEC);
  tft.setCursor(410, 45);
  tft.print("Max");
}


void mainMeasurement (void) {

  count++;
  counter1s++;
  counter500ms++;

  if (I1 == 1) {
```

```
  C_I1 = C_I1 + 1;
}

if ( attempt == 0 && (Pres > kPa_max_ss || Pres < kPa_min_ss)) {
  reset();
}
else if ( (I1 > I1_1 && relax_time == 0) || ( websock_start==1 && relax_time==0) ) {
  if (Pres < kPa_max_ss && Pres > kPa_min_ss) {
    if (maxPr == Error) attempt = attempt - 1;
    attempt = attempt + 1; end_time = 0; relax_time = 0;
    minPr = min_max_v; maxPr = 0; relaxPr = 0; ind_shake = 0;
  }
  else if (attempt == N && Tbl[2][4] != 0)
    attempt = attempt + 1;

  if (websock_start==1) {
    websock_start=0;
  }
  if (relax_time <10)
    x = 150;

}

if (C_I1 == 6 * 100 * mult) {
  ESP.restart();//reset();  C_I1 = 0;  x = 50; //tft.fillScreen(RA8875_BLACK);
}
else if (C_I1 < 6 * 100 * mult && I1 == 0)
  C_I1 = 0;

if (C_I1 == 1 * 200 * mult) {
 Option++;
 if (Option>1) Option=0;
 jsonOption(Option);
 reset();
}

if (attempt >= 1 && attempt <= N /*&& Pres > Pr_Min[Option] */) {
  reset_times = 0;

  if (end_time < C_end) {
    if (ind_shake == 0) relax_time = relax_time + 1;
    else {
      relax_time = 1;
      minPr = min_max_v;
      maxPr = 0;
      relaxPr = 0;
      ind_shake = 0;
    }

    if (Pres < minPr) minPr = Pres;
    if (Pres > maxPr) maxPr = Pres;
```

```
          if (relax_time <= C_relax) {
            relaxPr = relaxPr + Pres;
            if ( abs(Pres - relaxPr / relax_time) > shakes ) ind_shake = 1;
          }

          else if ( abs(Pres - relaxPr / C_relax) < threshold && (relaxPr / C_relax - minPr) > 1 * threshold)
{
            end_time = end_time + 1;
          }

          else if (relax_time > (max_meas_time + C_relax)) {
            end_time = C_end;
            maxPr = Error; minPr = Error + 1;
            error_flag = 1;
          }

          else
            end_time = 0;

        }
        else {

          if (relax_time != 0) {
            Tbl[1][attempt] = maxPr - minPr;
            if (maxPr == Error)
              Tbl[2][attempt] = -1;
            else {
              Serial.println ("maxPr = "+String(maxPr));
              Serial.println ("relaxPr = "+String(relaxPr));
              Serial.println ("C_relax = "+String(C_relax));
              Tbl[2][attempt] = maxPr - relaxPr / C_relax ;
              Serial.println ("minPr = "+String(minPr));
              Serial.println ("maxPr - minPr = "+String(Tbl[1][attempt]));
              Serial.println ("maxPr - relaxPr / C_relax = "+String(Tbl[2][attempt]));
            }
          }

        }

        if (attempt == N && relax_time == 0 && maxPr < Error) {
          if (Tbl[2][attempt + 1] == 0) {
            float T11 = Tbl[1][1];  float T21 = Tbl[2][1];
            for (int ind = 2; ind <= N; ind++) {
              if ( T11 < Tbl[1][ind] ) {
                T11 = Tbl[1][ind];
                T21 = Tbl[2][ind];
              }
            }
            Tbl[1][attempt + 1] = T11; Tbl[2][attempt + 1] = T21;
          }
        }
      }
```

```
  else
    reset();

  I1_1 = I1;

if (count == 5) {

    sensorV = Prf;
    up_Pres = Pres_to_Volt(Pr_Max[Option]);
    down_Pres = Pres_to_Volt(Pr_Min[Option]);
    balance_Pres = Pres_to_Volt(60);
    if ( sensorV >= balance_Pres )
    y2 = map( sensorV, balance_Pres-1 , up_Pres , 300 , 186 );
    else
    y2 = map( sensorV, balance_Pres+1 , down_Pres , 301, 415);
    if (y2 < 186) y2 = 186;  if (y2 > 415)  y2 = 415;

    count = 0;
    x++;

  }

  if (x > 749) {
    x = 250;
    flagsvisimo = 1;
  }


}


void reset(void) {

  if (reset_times == 0) {
    reset_flag = 1;
    end_time = 0; relax_time = 0; relaxPr = 0; ind_shake = 0;
    minPr = min_max_v; maxPr = 0;
    reset_times = 1;
  }

}

float Pres_to_Volt(float param) {

  float Vs5 = 4.56;
  float Vs3 = 3.25;
  float error = 0.2;
  float Voutc =  ( 0.018*Vs5*Vs3*2*param + 7.5*Vs5*(0.04*Vs5 - error*0.018*Vs5) ) / (Vs5*7.5);

  if (param==120.00) Voutc = 2322;
  if (param==90.00)  Voutc = 1749;
```

```
  if (param==60.00)  Voutc = 1176;
  if (param==50.00)  Voutc = 985;
  if (param==45.00)  Voutc = 889;

  return Voutc;

}

void mpataria(void) {

  int volt_step=180;

  if (counter1s == 200) {
   sensorV_1 = analogRead_cal(analogRead(36),36, ADC_ATTEN_DB_11) ;
   counter1s=0;
  }

  if ( battery_state != old_battery_state ) {
   tft.fillRect(20, 13, 40, 14, RA8875_BLACK);
   old_battery_state=battery_state;
  }

  if (I2 == LOW) {

    if (I2 != old_I2 ) {
      tft.fillRect(20, 13, 40, 14, RA8875_BLACK);
    }

     if ( sensorV_1 != old_sensorV_1 || (I2 != old_I2 ) ) {

       if (sensorV_1 <= 1980) {
         tft.fillRect(20, 13, 40, 14, RA8875_BLACK);
         battery_state=0;
       }

       if (sensorV_1 > 1980              && sensorV_1 <= (1980 + volt_step )) {
         tft.fillRect(20, 13, 3, 14, RA8875_RED);
         battery_state=1;
       }

       if (sensorV_1 > (1980 + volt_step ) && sensorV_1 <= (1980 + volt_step*2)) {
         tft.fillRect(20, 13, 3, 14, RA8875_GREEN);
         tft.fillRect(30, 13, 3, 14, RA8875_GREEN);
         battery_state=2;
       }

       if (sensorV_1 > (1980 + volt_step*2) && sensorV_1 <= (1980 + volt_step*3)) {
         tft.fillRect(20, 13, 3, 14, RA8875_GREEN);
         tft.fillRect(30, 13, 3, 14, RA8875_GREEN);
         tft.fillRect(40, 13, 3, 14, RA8875_GREEN);
         battery_state=3;
       }
```

```
      if (sensorV_1 > (1980 + volt_step*3) && sensorV_1 <= (1980 + volt_step*4)) {
        tft.fillRect(20, 13, 3, 14, RA8875_GREEN);
        tft.fillRect(30, 13, 3, 14, RA8875_GREEN);
        tft.fillRect(40, 13, 3, 14, RA8875_GREEN);
        tft.fillRect(50, 13, 3, 14, RA8875_GREEN);
        battery_state=4;
       }

     old_sensorV = sensorV_1;
     }
   old_I2 = I2;

  }

  if (I2 == HIGH) {
   if (I2 != old_I2 ) {
    tft.fillRect(20, 13, 40, 14, RA8875_BLACK);
    chrg=20;
   }
    if (counter500ms >= 100) {
      if (chrg >= 60) {
        tft.fillRect(20, 13, 40, 14, RA8875_BLACK);
        chrg = 20;
      }
      tft.fillRect(chrg, 13, 3, 14, RA8875_GREEN);
      chrg = chrg + 10;
      counter500ms=0;
    }
    old_I2 = I2;
  }

}

void start_wifi(void) {

  WiFi.begin(ssid, password);
  Serial.print("\n\nTry to connect to existing network");

    uint8_t timeout = 10;

    // Wait for connection, 5s timeout
    do {
      delay(500);
      Serial.print(".");
      timeout--;
    } while (timeout && WiFi.status() != WL_CONNECTED);

  if (WiFi.status() != WL_CONNECTED) {

  tft.setTextColor(RA8875_WHITE);
  tft.setCursor(94, 13);
```

```
 tft.print("WiFi Connection : Error");

 }

 else {

 Serial.println("\n\nWiFi parameters:");
 Serial.print("Mode: ");
 Serial.println("Station");
 Serial.print("IP address: ");
 Serial.println(WiFi.localIP());

 tft.setTextColor(RA8875_WHITE);
 tft.setCursor(95, 13);
 tft.print("WiFi: Connected!");
 tft.setCursor(265, 13);
 tft.print("IP address: ");
 tft.print(WiFi.localIP());

 ws.onEvent(onWsEvent);
 server.addHandler(&ws);

 server.onNotFound(
  [](AsyncWebServerRequest *request) {
  request->send(404);
 });

 server.begin();

 }

}
```

Παράρτημα 2. Κώδικας Javascript για την σελίδα Home

```
<script >
var chart;
var attempt=0;
var maximumPr=90;
var minimumPr=50;

function start() {

 if (
  window.location.port != "" ||
  window.location.port != 80 ||
  window.location.port != 443
 ) {
  websock = new WebSocket(
   "ws://" + "192.168.1.9" + ":" + window.location.port + "/ws"
  );
```

```
 } else {
  websock = new WebSocket("ws://" + window.location.hostname + "/ws");
 }
 websock.onopen = function(evt) {
  console.log("websock open");

 };
 websock.onclose = function(evt) {
  console.log("websock close");
 };
 websock.onerror = function(evt) {
  console.log(evt);
 };

 websock.onmessage = function(evt) {

  console.log(evt);

  var data = JSON.parse(evt.data);

     console.log(data);

     if (  data.length >= 10 ) {

      callbackFunc(data);

     }

     else {

       attemptFunc(data);

     }

 };

}

function create_chart()  {

 chart = new CanvasJS.Chart("chartContainer",
  {

  axisY:
  {
   title: "Pressure (mmHg)",
   titleFontSize: 20,
   stripLines: [{
    value: 60,  //Better than viewportMinimum
    color: "transparent",
    label: "Dummy Long Label",
    labelMaxWidth: 30,
```

```
          labelPlacement: "outside",
          labelBackgroundColor: "transparent",
          labelFontColor: "transparent"
         }],
         viewportMaximum: maximumPr,
         viewportMinimum: minimumPr
        },

        axisX:
        {
         title: "Time (S)",
         titleFontSize: 20,
            viewportMinimum: 0,
            viewportMaximum: 12.5
        },
        data: [
         {
           type: "spline",
                    color: "green",
                    markerType: "none",
           dataPoints: [
           { x: 10, y: 0 },
           ]
         }
            ]
        });

        chart.render();

    }


    create_chart();
    chart_reset ();

function callbackFunc (data) {

      var data_color;

     switch(chart.options.data.length) {
      case 1:
        data_color = "#00FFFF";
        break;
      case 2:
        data_color = "#9400D3";
        break;
      case 3:
        data_color = "#00008B";
        break;
      default:
        data_color = "black";
    }
```

```
      var newSeries = {
      type: "line",
      color: data_color,
    markerType: "none",
      dataPoints: data
    };
      chart.options.data.push(newSeries);
      chart.render();
}



 function chart_reset () {

 chart.destroy();
 create_chart();


 for (i=1;i<=4;i++)  {
  attempt=0;
  $("#maxPr"+i).html("0.0");
  $("#minPr"+i).html("0.0");

  $("#attempt"+i+"_check").css("color","grey");  //green #3ce23c


  $("#attempt_bar").css("width", "0");
  $("#situation").html("Waiting to start");


 }


 }


 $("#resetbut").click(function() {
    websock.send("reset");
    chart_reset ();
 });


 $("#startbut").click(function() {
     websock.send("start");
 });


 $('#user_select').on('change', function() {

     if ( $( "#user_select option:selected" ).value != "0") {
       $("#user_select option[value='0']").remove();
   }
```

```
});

$("#savebut").click(function() {

  if ( $( "#user_select option:selected" ).attr('id') == "noselect") {
                alert ( "Please first choose a user" );
                $('#user_select').addClass("highlght").focus();
    }
    else {

      if (attempt==3) {

                save_data ( $( "#user_select option:selected" ).attr('value') ,
$("#minPr4").html() , $("#maxPr4").html() );

            }

       else {
                alert ( "Please complete all 3 attempts \n"+"Your have completed
"+attempt+" attempts");
            }


     //websock.send("saved");
    }

});

$('#mode').on('change', function() {

     if (this.value=="0") {
            maximumPr=90;
            minimumPr=50;
            websock.send("mode0");
     }


   if (this.value=="1") {
            maximumPr=120;
            minimumPr=45;
            websock.send("mode1");
     }

    chart_reset ();


});
```

```
start();

function attemptFunc(data) {

 var p = data[0];

 for (var key in p) {
  if (p.hasOwnProperty(key)) {
            if (key =="attempt") {  //attemptNumber
                    attempt=p[key];
                $("#attempt"+p[key]+"_check").css("color","#3ce23c");  //green #3ce23c

                switch(p[key]) {
                        case 1:
                                $("#attempt_bar").css("width", "33%");
                        break;
                        case 2:
                                $("#attempt_bar").css("width", "66%");
                        break;
                        case 3:
                                $("#attempt_bar").css("width", "100%");
                                $("#attempt4_check").css("color","#3ce23c");
                        break;
                        default:
                                $("#attempt_bar").css("width", "0");
       }

            }
    else if ( key.includes("situation") ) {
               $("#"+key).html(p[key]);
               if (p[key]=="Measuring")       $("#"+key).css("color","#3ce23c");
                    if (p[key]=="Relax")          $("#"+key).css("color","#e4cb19");
                    if (p[key]=="Waiting to start")  $("#"+key).css("color","grey");
            }
    else if ( key.includes("reset") ) {
              chart_reset ();
            }
    else if ( key.includes("mode") ) {

               $('[name=options]').val( p[key] );//To select Blue
                    if (p[key]==0) {
                            maximumPr=90;
                            minimumPr=50;
                    }
                    if (p[key]==1) {
                            maximumPr=120;
                            minimumPr=45;
                    }
                    chart_reset ();
```

```
                }
        else {
                var numb = p[key];
                        numb = numb.toFixed(1);
                        //document.getElementById(key).innerHTML = numb;
                        $("#"+key).html(numb);
                }

  }
 }


 }


function fetch () {

                $.ajax({
        url: 'get_users.php',
        dataType:'json',
                        async: false,
        success:function(data) {

                        console.log (data);

                $(data).each( ( i , v ) => {

           $('#user_select')
                                .append($("<option></option>")
                                .attr("value", v.id)
                                .text( v.surname+" "+v.name ));

                         });
        }
      });
}


      function save_data ( id, minPr , maxPr )
   {


      $.ajax({
         url: "save_data.php",
         method:"POST",
         data:{id:id, minPr:minPr, maxPr:maxPr},

         success:function(data){
                console.log(data);
                        $('#result').html("<div class='alert alert-success'>"+data+"</div>");

                        setTimeout(function () { $('#result').html('') } , 5000) ;
```

```
            }
        });

    }
</script>
```

Παράρτημα 3. Κώδικας Javascript για την σελίδα History

```
<script>
var chart;

var tbl_diaphragmatic = [];

function create_chart()  {

chart = new CanvasJS.Chart("chartContainer", {
        animationEnabled: true,
        title:{
                text: "Evolution of Diaphragmatic breathing"
        },
        axisX: {
                valueFormatString: "DD MMM,YY HH:mm",
        },
        axisY: {
                title: "Pressure (in mmHg)",
                includeZero: false,
        },
        legend:{
                fontSize: 20
        },
        toolTip:{
                shared: true
        },
        data: []
});
chart.render();

}

create_chart();



function toggleDataSeries(e){
        if (typeof(e.dataSeries.visible) === "undefined" || e.dataSeries.visible) {
                e.dataSeries.visible = false;
        }
        else{
                e.dataSeries.visible = true;
        }
```

```
            chart.render();
}


function fetch_users () {

                    $.ajax({
        url: 'get_users.php',
        dataType:'json',
                            async: false,
        success:function(data) {

                            console.log (data);

          $(data).each( ( i , v ) => {

          $('#user_select')
                                    .append($("<option></option>")
                                    .attr("value", v.id)
                                    .text( v.surname+" "+v.name ));

                             });


        }
      });
}

async function fetch_data ( id ) {

                    await $.ajax({
        url: 'get_data.php',
                            method:"POST",
        data:{ id:id },
        dataType:"json",
        success:function(data) {

                            console.log (data);
                            callback_chart ( data );
        }
      });

                    await $.ajax({
        url: 'choose_user.php',
                            method:"POST",
        data:{ id:id },
        dataType:"json",
        success:function(data) {

                            callback_info ( data );
```

```
        }
    });

}

function callback_chart ( data ) {

        var datapoints_minPr = [];
        var datapoints_maxPr = [];


        chart.destroy();
        create_chart();

  var sum_minPr=0;
        var sum_maxPr=0;
        var count = data.length;

          $(data).each( ( i , v ) => {

                var formattedDate = new Date(v.date);
                var d = formattedDate.getDate();
           var m =  formattedDate.getMonth();
    m += 1;  // JavaScript months are 0-11
    var y = formattedDate.getFullYear();
                var h = formattedDate.getHours();
                var n = formattedDate.getMinutes();

                datapoints_minPr.push( {x:new Date(y,m,d,h,n) ,y:parseFloat(v.minPr)} );
                datapoints_maxPr.push( {x:new Date(y,m,d,h,n) ,y:parseFloat(v.maxPr)} );

                sum_minPr = sum_minPr + parseFloat(v.minPr);
                sum_maxPr = sum_maxPr + parseFloat(v.maxPr);

           if (i==count-1) {
                        tbl_diaphragmatic[0] = sum_maxPr/data.length;
                        tbl_diaphragmatic[2] = v.date;
                        tbl_diaphragmatic[1] = sum_minPr/data.length;
                        tbl_diaphragmatic[3] = data.length;
                }

          });

        var minPr_data =  {
                name: "Inhaling",
                type: "spline",
                yValueFormatString: "#0.## mmHg",
                showInLegend: true,
                dataPoints: datapoints_minPr
        }
  chart.options.data.push(minPr_data);
        var maxPr_data =  {
```

```
                name: "Inhaling & Exhaling",
                type: "spline",
      yValueFormatString: "#0.## mmHg",
                showInLegend: true,
                dataPoints: datapoints_maxPr
            }
            chart.options.data.push(maxPr_data);

    chart.render();

}

function callback_info ( data ) {

   $(data).each( ( i , v ) => {

      var activity_text;

            $("#name").html(v.name);
            $("#surname").html(v.surname);
            $("#age").html(v.age);

            switch(parseInt(v.activity)) {
                case 1:
                activity_text = "Sedentary";
                break;
                case 2:
                activity_text = "Lightly Active";
                break;
                case 3:
                activity_text = "Active";
                break;
       case 4:
                activity_text = "Very Active";
            }
            $("#activity").html(activity_text);

            });

    if (tbl_diaphragmatic[0]==0) {
                $("#ave_maxPr").html("-");
                $("#ave_minPr").html("-");
                $("#last_date").html("-");
                $("#sum_attempts").html("-");
            }
            else {
            $("#ave_maxPr").html(tbl_diaphragmatic[0].toFixed(1));
            $("#ave_minPr").html(tbl_diaphragmatic[1].toFixed(1));
            $("#last_date").html(tbl_diaphragmatic[2]);
            $("#sum_attempts").html(tbl_diaphragmatic[3]);
            }
            for (i=0;i<=3;i++){
```

```
            tbl_diaphragmatic[i]=0;
           }

}

  $('#user_select').on('change', function() {

           if ( $( "#user_select option:selected" ).value != "0") {
             $("#user_select option[value='0']").remove();
    }

     fetch_data ( this.value );


  });
</script>
```

Παράρτημα 4. Κώδικας Javascript για την σελίδα Registration

```
<script>
$("#sumbitForm").submit(function(e) {

   e.preventDefault(); // avoid to execute the actual submit of the form.

   var form = $(this);
   var url = form.attr('action');

   $.ajax({
        type: "POST",
        url: url,
        data: form.serialize(), // serializes the form's elements.
        success: function(data)
        {
                              $("#alertsuccess").show(); $("#alertsuccess").html(data);
                              setTimeout(function () { $("#alertsuccess").hide(); } , 5000) ;

        }
       });

});

function sumbitFunc(){

           $("#sumbitForm").submit();

}
</script>
```

Παράρτημα 5. Κώδικας PHP για το αρχείο *choose_user.php*

```
<?php
 $connect = mysqli_connect("localhost", "root", "", "diabreathing");

 mysqli_set_charset($connect, "utf8");

 $id = $_POST['id'];

 $sql = "SELECT * FROM users WHERE id = '$id' ";
 $result = mysqli_query($connect, $sql);

 $arr = array();

 $rows = mysqli_num_rows($result);

 if($rows > 0)
 {
     while($row = mysqli_fetch_array($result))
     {
      $myObj = array( "name"=>$row["name"] , "surname"=>$row["surname"] , "age"=>$row["age"] ,
"activity"=>$row["activity"] );
       array_push ( $arr , $myObj );
             }
 }
 echo json_encode($arr);
 mysqli_close($connect);
?>
```

Παράρτημα 6. Κώδικας PHP για το αρχείο *get_data.php*

```
<?php
 $connect = mysqli_connect("localhost", "root", "", "diabreathing");

 mysqli_set_charset($connect, "utf8");

 $id = $_POST['id'];

 $sql = "SELECT * FROM attempts WHERE id = '$id' ORDER BY date ASC";
 $result = mysqli_query($connect, $sql);

 $arr = array();

 $rows = mysqli_num_rows($result);

 if($rows > 0)
 {
     while($row = mysqli_fetch_array($result))
     {
      $myObj = array("minPr"=>$row["minPr"], "maxPr"=>$row["maxPr"] , "date"=>$row["date"]);
```

```
            array_push ( $arr , $myObj );

            }
}
 echo json_encode($arr);
 mysqli_close($connect);
?>
```

Παράρτημα 7. Κώδικας PHP για το αρχείο *get_users.php*

```
<?php
 $connect = mysqli_connect("localhost", "root", "", "diabreathing");

 mysqli_set_charset($connect, "utf8");

 $sql = "SELECT * FROM users";
 $result = mysqli_query($connect, $sql);

 $arr = array();

 $rows = mysqli_num_rows($result);

 if($rows > 0)
 {

     while($row = mysqli_fetch_array($result))
     {
      $myObj = array("id"=>$row["id"], "name"=>$row["name"] , "surname"=>$row["surname"]);

              array_push ( $arr , $myObj );

             }
}
 echo json_encode($arr);
 mysqli_close($connect);
?>
```

Παράρτημα 8. Κώδικας PHP για το αρχείο *register_user.php*

```
<?php

$servername = "localhost";
$username = "root";
$password = "";
$dbname = "diabreathing";

$conn = mysqli_connect($servername, $username, $password, $dbname);

if (mysqli_connect_errno())
  {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
```

```
 }

mysqli_set_charset($conn, "utf8");

$name = $_POST['name'];
$surname = $_POST['surname'];
$age = $_POST['age'];
$activity = $_POST['activity'];

$sql = "INSERT INTO users ( name , surname , age , activity ) VALUES ( '$name' , '$surname' , '$age' ,
'$activity' )";

if ($name != "" && $surname !="" && $age!="") {

        if ($conn->query($sql) === TRUE) {
                echo "New patient profile created successfully";
        } else {
                echo "Error: " . $sql . "<br>" . $conn->error;
        }
}
else {
        echo "Please fill out all required fields!";
}
mysqli_close($conn);
?>
```

Παράρτημα 9. Κώδικας PHP για το αρχείο *save_data.php*

```
<?php

$servername = "localhost";
$username = "root";
$password = "";
$dbname = "diabreathing";

$conn = mysqli_connect($servername, $username, $password, $dbname);

if (mysqli_connect_errno())
 {
 echo "Failed to connect to MySQL: " . mysqli_connect_error();
 }

mysqli_set_charset($conn, "utf8");

$minPr = $_POST['minPr'];
$maxPr = $_POST['maxPr'];
$id = $_POST['id'];

$sql = "INSERT INTO attempts ( minPr , maxPr , id  , date ) VALUES ( '$minPr' , '$maxPr' , '$id' , now()
)";
```

```
        if ($conn->query($sql) === TRUE) {
                echo "New patient data updated successfully";
        } else {
                echo "Error: " . $sql . "<br>" . $conn->error;
        }
 mysqli_close($conn);
?>
```