ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Συσκευή παρακολούθησης καρδιακής δραστηριότητας – Διαδίκτυο των πραγμάτων»

Του φοιτητή
Σουλάκη Γεώργιου
Αρ. Μητρώου: 516121

Επιβλέπων
Γιακουμής Άγγελος
Επίκουρος Καθηγητής

Θεσσαλονίκη, Δεκέμβριος 2024

Τίτλος Δ.Ε. Συσκευή παρακολούθησης καρδιακής δραστηριότητας – Διαδίκτυο των πραγμάτων
Κωδικός Δ.Ε. 24192
Ονοματεπώνυμο φοιτητή Σουλάκης Γεώργιος
Ονοματεπώνυμο εισηγητή Γιακουμής Άγγελος
Ημερομηνία ανάληψης Δ.Ε. 31/3/2024
Ημερομηνία περάτωσης Δ.Ε. 19/6/2025

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.

«Στην ιατρική τεχνολογία»

Αφιέρωση

Αυτό το έργο αφιερώνεται στους φίλους και την οικογένειά μου, που με υποστήριξαν καθ’ όλη τη δύσκολη διαδικασία της έρευνας, του προγραμματισμού και της συγγραφής. Επιπλέον, εκφράζω την ευγνωμοσύνη μου στον Δρ. Γιακουμή Άγγελο, ο οποίος στήριξε τις προσπάθειές μου.

Περίληψη

Abstract (EN)

This thesis presents the design and implementation of a heart activity monitoring device based on the Internet of Things (IoT). A biosignal acquisition sensor (AD8232) captures ECG signals, which are processed using an STM32 microcontroller and transmitted via UART to an STM32 Wi-Fi module. The data is then sent to a laptop using TCP/IP and visualized in real-time with a custom Python GUI built with Tkinter and Matplotlib. This system demonstrates how embedded systems, wireless communication, and software can be integrated into a low-cost, efficient health-monitoring solution.

Περίληψη (EL)

Η παρούσα εργασία παρουσιάζει τον σχεδιασμό και την υλοποίηση μιας συσκευής παρακολούθησης καρδιακής δραστηριότητας βασισμένης στο Διαδίκτυο των Πραγμάτων (IoT). Ένας αισθητήρας καταγραφής βιοσημάτων (AD8232) συλλέγει σήματα καρδιακής λειτουργίας (ECG), τα οποία επεξεργάζεται ένας μικροελεγκτής STM32 και τα μεταδίδει μέσω UART σε ένα ασύρματο module ESP32. Στη συνέχεια, τα δεδομένα αποστέλλονται μέσω TCP/IP σε υπολογιστή και απεικονίζονται σε πραγματικό χρόνο μέσω γραφικού περιβάλλοντος που υλοποιήθηκε σε Python με χρήση των Tkinter και Matplotlib. Το σύστημα αυτό αναδεικνύει τη δυνατότητα ενοποίησης ενσωματωμένων συστημάτων, ασύρματης επικοινωνίας και λογισμικού σε μια οικονομική και αποδοτική λύση παρακολούθησης της υγείας.

Πίνακας Περιεχομένων

1. Η αξία των ιατρικών τεχνολογιών (Medical Technologies)

Οι ιατρικές τεχνολογίες αποτελούν έναν από τους πιο σημαντικούς παράγοντες που έχουν συμβάλλει στη βελτίωση της υγείας και της ποιότητας ζωής του ανθρώπου. Από τις πρώτες απλές ιατρικές συσκευές μέχρι τα πιο σύγχρονα διαγνωστικά και θεραπευτικά εργαλεία, η εξέλιξη των ιατρικών τεχνολογιών έχει αλλάξει ριζικά την προσέγγιση της ιατρικής φροντίδας, διευρύνοντας τις δυνατότητες πρόληψης, διάγνωσης, θεραπείας και αποκατάστασης.

1.1. Βελτίωση της διάγνωσης και της θεραπείας

Η εξέλιξη των ιατρικών τεχνολογιών έχει επιτρέψει την ακριβέστερη και ταχύτερη διάγνωση ασθενειών. Εργαλεία όπως η μαγνητική τομογραφία (MRI), η αξονική τομογραφία (CT), οι υπέρηχοι, καθώς και οι εξελιγμένες εργαστηριακές εξετάσεις επιτρέπουν στους γιατρούς να εντοπίζουν προβλήματα σε πολύ αρχικά στάδια, όταν η θεραπεία έχει πολύ μεγαλύτερη επιτυχία. Αυτό έχει οδηγήσει σε σημαντική μείωση των θανάτων από χρόνιες και επικίνδυνες ασθένειες, όπως ο καρκίνος και τα καρδιαγγειακά νοσήματα.

Η τεχνολογία έχει επίσης επιτρέψει την ανάπτυξη καινοτόμων θεραπειών. Η ρομποτική χειρουργική, για παράδειγμα, επιτρέπει ακριβείς, ελάχιστα επεμβατικές επεμβάσεις, μειώνοντας τον χρόνο ανάρρωσης και τον κίνδυνο επιπλοκών. Επιπλέον, οι τεχνολογίες βιοτεχνολογίας και γονιδιακής θεραπείας ανοίγουν το δρόμο για θεραπείες που στοχεύουν στην αιτία της ασθένειας σε μοριακό επίπεδο.

1.2. Πρόληψη και παρακολούθηση της υγείας

Οι ιατρικές τεχνολογίες δεν περιορίζονται μόνο στη διάγνωση και θεραπεία, αλλά επεκτείνονται και στην πρόληψη και την παρακολούθηση της υγείας. Οι φορητές συσκευές, όπως οι έξυπνες ζώνες και τα ρολόγια που μετρούν τον καρδιακό ρυθμό, το οξυγόνο στο αίμα και άλλους βιοδείκτες, βοηθούν τα άτομα να παρακολουθούν συνεχώς την υγεία τους και να αναγνωρίζουν έγκαιρα σημάδια προβλημάτων.

Η τηλεϊατρική και οι εφαρμογές απομακρυσμένης παρακολούθησης επιτρέπουν την παροχή ιατρικής φροντίδας ακόμα και σε απομακρυσμένες ή δυσπρόσιτες περιοχές, βελτιώνοντας την πρόσβαση σε ειδικούς και μειώνοντας το κόστος των μετακινήσεων και των νοσηλειών.

1.3. Βελτίωση της αποτελεσματικότητας και μείωση κόστους

Η ενσωμάτωση των ιατρικών τεχνολογιών στα νοσοκομεία και τα κέντρα υγείας οδηγεί σε σημαντική αύξηση της αποδοτικότητας των υπηρεσιών υγείας. Αυτοματοποιημένα συστήματα διαχείρισης ασθενών, ηλεκτρονικά αρχεία υγείας (EHR), και ψηφιακά εργαλεία υποστηρίζουν τους επαγγελματίες υγείας στη λήψη αποφάσεων, τη συντονισμένη φροντίδα και την ορθή διαχείριση πόρων. Παράλληλα, η πρόληψη και η έγκαιρη διάγνωση μειώνουν το κόστος της νοσηλείας και των μακροχρόνιων θεραπειών, με τελικό όφελος για το σύστημα υγείας και τους ασθενείς.

1.4. Προσωποποιημένη Ιατρική

Η σύγχρονη ιατρική τεχνολογία έχει φέρει στο προσκήνιο την προοπτική της προσωποποιημένης ιατρικής, όπου η θεραπεία και η πρόληψη προσαρμόζονται στα μοναδικά γενετικά, περιβαλλοντικά και βιολογικά χαρακτηριστικά κάθε ασθενούς. Η ανάλυση γονιδιώματος, η χρήση βιοδεικτών και οι προχωρημένοι αλγόριθμοι τεχνητής νοημοσύνης επιτρέπουν τη δημιουργία εξατομικευμένων σχεδίων θεραπείας με αυξημένη αποτελεσματικότητα και μειωμένες παρενέργειες.

1.5. Ηθικές και κοινωνικές προκλήσεις

Παρά τα πολλαπλά οφέλη, η χρήση των ιατρικών τεχνολογιών εγείρει και ηθικά, κοινωνικά και νομικά ζητήματα. Η προστασία των προσωπικών δεδομένων υγείας, η ισότιμη πρόσβαση στις τεχνολογίες, καθώς και ο ρόλος της τεχνητής νοημοσύνης στην ιατρική διάγνωση και θεραπεία αποτελούν πεδία διαρκούς συζήτησης.

2. Smart Hospitals και το Internet of Medical Things (IoMT)

Η ψηφιακή επανάσταση αλλάζει δραματικά τον τρόπο που λειτουργεί ο τομέας της υγείας. Σήμερα, η ιατρική τεχνολογία δεν περιορίζεται απλώς σε διαγνωστικά μηχανήματα ή ιατρικές συσκευές, αλλά επεκτείνεται στο πλαίσιο των "έξυπνων" νοσοκομείων (Smart Hospitals) και του Internet of Medical Things (IoMT). Η συνένωση της πληροφορικής, των αισθητήρων, της τεχνητής νοημοσύνης (AI) και της συνδεσιμότητας δικτύων δημιουργεί ένα νέο οικοσύστημα υγείας που βελτιώνει την ποιότητα, την ασφάλεια, και την αποτελεσματικότητα της φροντίδας των ασθενών.

2.1 Τι είναι τα Smart Hospitals;

Τα Smart Hospitals είναι μονάδες υγείας που χρησιμοποιούν ψηφιακές τεχνολογίες, αυτοματισμούς, και συνδεδεμένες συσκευές για να βελτιώσουν τις κλινικές διαδικασίες, τη διαχείριση πόρων και την εμπειρία ασθενών. Πρόκειται για νοσοκομεία που ενσωματώνουν καινοτόμες λύσεις πληροφορικής, IoT συσκευές, και συστήματα αυτοματισμού προκειμένου να μετατρέψουν τον παραδοσιακό χώρο υγείας σε ένα δυναμικό, ευέλικτο και έξυπνο περιβάλλον.

Χαρακτηριστικά των Smart Hospitals

- **Συνδεδεμένες συσκευές:** Από φορητούς αισθητήρες μέχρι έξυπνα ιατρικά μηχανήματα που συλλέγουν και ανταλλάσσουν δεδομένα σε πραγματικό χρόνο.

- **Αυτοματισμός διαδικασιών:** Αυτόματη διαχείριση φαρμάκων, παρακολούθηση ιατρικών υλικών, και αυτόματη καταγραφή δεδομένων ασθενών.

- **Έξυπνη διαχείριση ενέργειας και πόρων:** Βελτιστοποίηση χρήσης ενέργειας, διαχείριση κτιριακών συστημάτων, και παρακολούθηση εξοπλισμού.

- **Προηγμένα συστήματα ανάλυσης:** Χρήση τεχνητής νοημοσύνης και μηχανικής μάθησης για διάγνωση, πρόβλεψη και υποστήριξη αποφάσεων.

- **Ενισχυμένη ασφάλεια:** Προστασία δεδομένων ασθενών μέσω κρυπτογράφησης, δικαιωμάτων πρόσβασης και συστημάτων ανίχνευσης απειλών.

2.2 Το Internet of Medical Things (IoMT)

Το IoMT αποτελεί μια ειδική κατηγορία του ευρύτερου Internet of Things (IoT), που αφορά συσκευές και εφαρμογές ειδικά σχεδιασμένες για τον τομέα της υγείας. Το IoMT περιλαμβάνει συσκευές όπως φορητούς αισθητήρες, έξυπνες συσκευές παρακολούθησης υγείας, ιατρικές απεικονιστικές συσκευές, καθώς και λύσεις τηλεϊατρικής.

Παραδείγματα IoMT Συσκευών

- **Φορητοί αισθητήρες υγείας:** Καρδιακοί ρυθμογράφοι, παλμικά οξύμετρα, γλυκομετρητές που συνδέονται με smartphones.

- **Έξυπνες κλίνες νοσοκομείου:** Παρακολουθούν την κίνηση ασθενών και τους παραπάνω ζωτικούς δείκτες.

- **Φαρμακευτικά ρομπότ:** Αυτόματη χορήγηση φαρμάκων και παρακολούθηση αποθεμάτων.

- **Συστήματα τηλεϊατρικής:** Εξοπλισμός για απομακρυσμένη διάγνωση και παρακολούθηση ασθενών.

- **Ιατρικές απεικονιστικές συσκευές:** Συνδεδεμένα μηχανήματα υπερήχων, MRI και CT.

2.3 Οφέλη από την υιοθέτηση Smart Hospitals και IoMT

2.3.1. Βελτίωση της ποιότητας φροντίδας

Με την άμεση και ακριβή παρακολούθηση ζωτικών σημείων, τα νοσοκομεία μπορούν να ανιχνεύσουν εγκαίρως αλλαγές στην κατάσταση του ασθενούς, παρέχοντας πιο γρήγορες και στοχευμένες θεραπείες. Η αυτοματοποίηση μειώνει τα ανθρώπινα λάθη σε διαδικασίες όπως η χορήγηση φαρμάκων.

2.3.2. Αύξηση της αποδοτικότητας

Η χρήση έξυπνων συσκευών και αυτοματισμών μειώνει το διοικητικό φόρτο, απελευθερώνοντας προσωπικό για πιο κρίσιμες κλινικές εργασίες. Η διαχείριση πόρων γίνεται πιο αποτελεσματική, μειώνοντας κόστος και σπατάλες.

2.3.3. Προσωποποιημένη ιατρική

Τα δεδομένα που συλλέγονται από IoMT συσκευές μπορούν να αναλυθούν με προηγμένους αλγόριθμους για την παροχή εξατομικευμένων σχεδίων θεραπείας, βασισμένων στις ανάγκες και τις ιδιαιτερότητες κάθε ασθενή.

2.3.4. Βελτιωμένη εμπειρία ασθενών

Η χρήση έξυπνων τεχνολογιών μπορεί να μειώσει τον χρόνο αναμονής, να διευκολύνει τον ασθενή με αυτοματοποιημένες υπηρεσίες, και να προσφέρει συνεχή παρακολούθηση και φροντίδα ακόμα και εκτός νοσοκομείου.

2.3.5. Ενίσχυση της ασφάλειας και συμμόρφωσης

Με την κρυπτογράφηση δεδομένων και την αυστηρή διαχείριση πρόσβασης, τα smart hospitals εξασφαλίζουν ότι τα προσωπικά ιατρικά δεδομένα παραμένουν ασφαλή και συμμορφωμένα με κανονισμούς όπως το GDPR και HIPAA.

2.4 Προκλήσεις και Περιορισμοί

2.4.1. Ασφάλεια και προστασία δεδομένων

Η συνεχής διασύνδεση πολλών συσκευών δημιουργεί ένα πολύπλοκο δίκτυο που είναι επιρρεπές σε κυβερνοεπιθέσεις. Η προστασία των δεδομένων και η ασφάλεια των συσκευών είναι ύψιστης σημασίας.

2.4.2. Διαλειτουργικότητα

Οι διάφορες συσκευές IoMT προέρχονται από πολλούς κατασκευαστές με διαφορετικά πρωτόκολλα επικοινωνίας, καθιστώντας δύσκολη την απρόσκοπτη ενοποίηση και επικοινωνία.

2.4.3. Κόστος υλοποίησης

Η εγκατάσταση υποδομών για smart hospitals και IoMT συσκευές απαιτεί σημαντική επένδυση, τόσο σε εξοπλισμό όσο και σε κατάρτιση προσωπικού.

2.4.4. Νομικά και ηθικά ζητήματα

Η συλλογή και επεξεργασία προσωπικών ιατρικών δεδομένων εγείρει ζητήματα απορρήτου και ηθικής που πρέπει να αντιμετωπιστούν με σαφείς πολιτικές και κανονισμούς.

2.5 Μελλοντικές τάσεις

2.5.1. Τεχνητή νοημοσύνη και μηχανική μάθηση

Η ενσωμάτωση αλγορίθμων AI/ML θα επιτρέψει την πρόβλεψη νοσημάτων, την αυτόματη ανάλυση εικόνων, και τη βελτιστοποίηση της κλινικής διαχείρισης.

2.5.2. 5G και edge computing

Η υιοθέτηση των δικτύων 5G θα εξασφαλίσει υψηλές ταχύτητες και χαμηλή καθυστέρηση για τη μετάδοση δεδομένων IoMT, ενώ το edge computing θα επιτρέψει την επεξεργασία δεδομένων κοντά στον ασθενή για ταχύτερες αντιδράσεις.

2.5.3. Αυξημένη εξατομίκευση

Η συνεχής συλλογή και ανάλυση δεδομένων από φορητές συσκευές θα οδηγήσει σε πιο εξατομικευμένες θεραπείες και φροντίδα, προσαρμοσμένες στον τρόπο ζωής και τη γενετική σύνθεση του κάθε ασθενή.

3. Επισκόπηση της Τεχνολογίας ECG στην Υγειονομική Περίθαλψη

Η τεχνολογία ECG (Ηλεκτροκαρδιογράφημα) είναι θεμελιώδης για τη διάγνωση και την παρακολούθηση των καρδιολογικών παθήσεων. Το ECG μετρά τη δραστηριότητα του καρδιακού μυός μέσω ηλεκτρικών σημάτων που παράγονται καθώς η καρδιά χτυπά, επιτρέποντας στους ιατρούς να ανιχνεύσουν διάφορες καρδιολογικές ανωμαλίες, όπως αρρυθμίες, καρδιοπάθειες και άλλα προβλήματα. Ειδικότερα, το ECG βοηθά στην αποτύπωση του ρυθμού, της συχνότητας και του χρόνου που απαιτείται για κάθε καρδιακό παλμό, προσφέροντας ζωτικής σημασίας πληροφορίες για την κατάσταση της καρδιάς του ασθενούς.

Τα τελευταία χρόνια, η τεχνολογία ECG έχει εξελιχθεί ραγδαία με την ενσωμάτωση ασύρματων συσκευών και έξυπνων τεχνολογιών, οι οποίες διευκολύνουν την απομακρυσμένη παρακολούθηση της καρδιακής υγείας. Οι φορητές συσκευές ECG, που περιλαμβάνουν έξυπνα ρολόγια και άλλες μικρές συσκευές, επιτρέπουν στους ασθενείς να παρακολουθούν τη δραστηριότητα της καρδιάς τους σε πραγματικό χρόνο και να στέλνουν τα δεδομένα στους γιατρούς μέσω του διαδικτύου. Αυτή η δυνατότητα έχει μεταμορφώσει την υγειονομική περίθαλψη, επιτρέποντας πιο ακριβή και έγκαιρη διάγνωση, καθώς και καλύτερη παρακολούθηση της πορείας της θεραπείας.

Η ενσωμάτωση της ασύρματης τεχνολογίας στο ECG έχει αναδείξει νέες δυνατότητες για την πρόληψη καρδιολογικών προβλημάτων. Οι γιατροί μπορούν να παρακολουθούν τους ασθενείς χωρίς να απαιτείται η φυσική τους παρουσία, με το δεδομένα να αποστέλλονται σε πραγματικό χρόνο μέσω Wi-Fi ή άλλων δικτύων. Παράλληλα, η ανάπτυξη τεχνολογιών για τη βελτίωση της ακρίβειας και της αξιοπιστίας των αποτελεσμάτων είναι συνεχής, προσφέροντας μεγαλύτερη ασφάλεια και αποτελεσματικότητα στη διάγνωση και θεραπεία των καρδιολογικών παθήσεων.

3.1 κατηγορίες Συσκευών ECG

Τα ηλεκτροκαρδιογραφήματα (ECG) αποτελούν ένα από τα πιο διαδεδομένα εργαλεία στη διάγνωση καρδιολογικών παθήσεων. Σήμερα, η αγορά προσφέρει μια ποικιλία συσκευών ECG, οι οποίες διαφέρουν ως προς τον τύπο, την τεχνολογία, και τη χρήση τους στην καθημερινή κλινική πρακτική ή ακόμα και στην καθημερινή ζωή των ασθενών. Οι κύριοι τύποι συσκευών ECG που διατίθενται στην αγορά περιλαμβάνουν τις φορητές συσκευές, τα φορητά συστήματα, τις συσκευές μιας χρήσης, καθώς και τις κλινικές και νοσοκομειακές συσκευές.

3.1.1. Φορητές Συσκευές ECG: Αυτές οι συσκευές είναι μικρές και εύκολες στη χρήση, επιτρέποντας στους ασθενείς να παρακολουθούν την καρδιακή τους δραστηριότητα σε καθημερινή βάση. Συνήθως, αυτές οι συσκευές χρησιμοποιούν ηλεκτρόδια που τοποθετούνται στο σώμα και συνδέονται με ένα μικρό φορητό καταγραφικό. Ορισμένα από αυτά τα συστήματα είναι ασύρματα και μπορούν να αποστέλλουν τα δεδομένα στον γιατρό μέσω Bluetooth ή Wi-Fi.

3.1.2. Φορητά Συστήματα ECG: Αυτές οι συσκευές συνήθως χρησιμοποιούνται για τη διάγνωση και παρακολούθηση της καρδιολογικής κατάστασης σε ασθενείς που έχουν ανάγκη από μακροχρόνια παρακολούθηση. Διαθέτουν περισσότερους αισθητήρες και πιο εξελιγμένο λογισμικό σε σχέση με τις απλές φορητές συσκευές και μπορούν να προσφέρουν πιο ακριβή δεδομένα. Συνδέονται με υπολογιστές ή κινητές συσκευές για τη μετάδοση των αποτελεσμάτων.

3.1.3. Συσκευές Μιας Χρήσης ECG: Αυτές οι συσκευές είναι συνήθως πιο οικονομικές και προορίζονται για μία μόνο χρήση, συνήθως σε καταστάσεις έκτακτης ανάγκης ή όταν απαιτείται γρήγορη διάγνωση. Στις περισσότερες περιπτώσεις, χρησιμοποιούνται σε συνδυασμό με μια πιο εξελιγμένη κλινική εξέταση.

3.1.4. Κλινικές και Νοσοκομειακές Συσκευές ECG: Αυτές οι συσκευές είναι πιο πολύπλοκες και χρησιμοποιούνται για ακριβείς και εκτεταμένες διαγνωστικές εξετάσεις. Παρέχουν πολύ πιο αναλυτικά δεδομένα και είναι ικανές να καταγράφουν την καρδιολογική δραστηριότητα για μεγαλύτερα χρονικά διαστήματα, με σκοπό την πιο λεπτομερή ανάλυση των καρδιολογικών προβλημάτων.

Η συνεχής εξέλιξη της τεχνολογίας έχει φέρει πολλές καινοτομίες στους τύπους των συσκευών ECG, επιτρέποντας στους ασθενείς και στους ιατρούς να έχουν περισσότερες δυνατότητες στην παρακολούθηση και διάγνωση της καρδιακής υγείας.

4. Εταιρείες που Εμπλέκονται στην Ανάπτυξη Τεχνολογίας ECG

Η τεχνολογία ECG (Ηλεκτροκαρδιογράφημα) είναι θεμελιώδης για τη διάγνωση και την παρακολούθηση καρδιολογικών παθήσεων, και πολλές εταιρείες παγκοσμίως εργάζονται για την ανάπτυξη καινοτόμων συσκευών ECG, βελτιώνοντας την ακρίβεια και την ευκολία χρήσης τους. Με την πρόοδο της τεχνολογίας, ο τομέας των συσκευών ECG έχει εξελιχθεί για να καλύψει τις αυξανόμενες ανάγκες των ασθενών και των επαγγελματιών υγείας, ενσωματώνοντας νέες δυνατότητες όπως η ασύρματη μεταφορά δεδομένων και η ενσωμάτωση με κινητές εφαρμογές.

4.1. Philips Healthcare: Η Philips είναι μία από τις πιο σημαντικές εταιρείες στον τομέα της υγειονομικής τεχνολογίας και έχει επενδύσει σημαντικά στην ανάπτυξη προηγμένων συσκευών ECG για νοσοκομεία και κλινικές. Τα προϊόντα της περιλαμβάνουν ολοκληρωμένα συστήματα ECG που παρέχουν ακριβή δεδομένα και έχουν δυνατότητες για τη μακροχρόνια παρακολούθηση των καρδιολογικών ασθενών.

4.2. GE Healthcare: Η GE Healthcare είναι μία ακόμη ηγέτιδα δύναμη στην αγορά των συσκευών ECG. Παράγει συσκευές που χρησιμοποιούνται σε κλινικές και νοσοκομεία για την παρακολούθηση των καρδιολογικών καταστάσεων και για τη διάγνωση αρρυθμιών και άλλων καρδιολογικών παθήσεων. Επίσης, η GE έχει εστιάσει στην ενσωμάτωση της τεχνολογίας cloud και στην ασύρματη μεταφορά δεδομένων ECG για μεγαλύτερη ευκολία και ταχύτητα στη διάγνωση.

4.3. Medtronic: Η Medtronic είναι μια από τις μεγαλύτερες παγκόσμιες εταιρείες στον τομέα των ιατρικών συσκευών και επενδύει συνεχώς στην ανάπτυξη τεχνολογιών ECG. Η εταιρεία προσφέρει συσκευές ECG που χρησιμοποιούνται τόσο σε κλινικές όσο και για προσωπική χρήση, ενσωματώνοντας προηγμένα χαρακτηριστικά όπως η τηλεϊατρική και η ασύρματη παρακολούθηση.

4.4. AliveCor: Μια νεοσύστατη αλλά καινοτόμος εταιρεία, η AliveCor εξειδικεύεται στην ανάπτυξη φορητών συσκευών ECG, οι οποίες συνδέονται με κινητά τηλέφωνα και παρέχουν δυνατότητες παρακολούθησης της καρδιακής δραστηριότητας σε πραγματικό χρόνο. Οι συσκευές της AliveCor είναι ιδιαίτερα δημοφιλείς μεταξύ των ασθενών που θέλουν να παρακολουθούν την καρδιολογική τους κατάσταση από το σπίτι.

4.5. Zoll Medical Corporation: Η Zoll είναι γνωστή για την ανάπτυξη προηγμένων συσκευών παρακολούθησης καρδιακής υγείας, περιλαμβανομένων των φορητών συσκευών ECG και των συστημάτων αναζωογόνησης καρδιοπνευμονικής ανάνηψης. Η Zoll εστιάζει στην ανάπτυξη λύσεων για επείγουσες καταστάσεις και στην ενσωμάτωση του ECG σε συστήματα πρώτων βοηθειών.

Η συνεχής εξέλιξη και η αυξανόμενη ζήτηση για συσκευές ECG οδηγούν σε διαρκείς καινοτομίες από αυτές τις εταιρείες, με στόχο τη βελτίωση της παρακολούθησης και διάγνωσης καρδιολογικών παθήσεων.

5. Η ιατρική πίσω από τα ECG συστήματα

5.1 Αρχές λειτουργίας ECG

Το Ηλεκτροκαρδιογράφημα (ECG) είναι μία τεχνική που χρησιμοποιείται για την καταγραφή των ηλεκτρικών σημάτων που παράγονται από την καρδιά κατά τη διάρκεια της συστολής και της διαστολής της. Οι αρχές λειτουργίας του ECG βασίζονται στην ανίχνευση αυτών των σημάτων μέσω ηλεκτροδίων που τοποθετούνται σε στρατηγικές θέσεις στο σώμα του ασθενούς.

Η καρδιά λειτουργεί με τη βοήθεια ηλεκτρικών ρευμάτων, τα οποία προκαλούν την σύσπαση των μυών της και την προώθηση του αίματος σε όλο το σώμα. Αυτά τα ηλεκτρικά ρεύματα παράγονται από τον φυσικό βηματοδότη της καρδιάς, τον κόλπο-κοιλιακό κόμβο, και διαδίδονται μέσω των καρδιακών ιστών. Το ECG καταγράφει αυτά τα ηλεκτρικά σήματα σε διάφορα στάδια της καρδιακής δραστηριότητας, επιτρέποντας στους γιατρούς να αξιολογήσουν τη λειτουργία της καρδιάς και να εντοπίσουν τυχόν ανωμαλίες.

Ουσιαστικά, το ECG λειτουργεί ως εξής:

1. Τοποθέτηση Ηλεκτροδίων: Ο γιατρός τοποθετεί ηλεκτρόδια στο σώμα του ασθενούς. Συνήθως, τοποθετούνται σε συγκεκριμένα σημεία στο στήθος, τα άκρα και την πλάτη, ώστε να καταγράφουν τη δραστηριότητα της καρδιάς από διαφορετικές γωνίες.

2. Καταγραφή των Ηλεκτρικών Σημάτων: Τα ηλεκτρόδια ανιχνεύουν τις διακυμάνσεις του ηλεκτρικού ρεύματος που παράγεται από την καρδιά και τις μεταδίδουν σε έναν καταγραφέα. Το αποτέλεσμα είναι μία γραμμή που καταγράφει την ηλεκτρική δραστηριότητα σε πραγματικό χρόνο.

3. Ανάλυση του Σήματος: Το ECG καταγράφει κύματα, τα οποία αντιπροσωπεύουν την ηλεκτρική δραστηριότητα της καρδιάς. Τα κύματα P, QRS και T αντιστοιχούν σε διαφορετικά στάδια της καρδιακής δραστηριότητας, όπως η συστολή και η χαλάρωση των κοιλιών και των κόλπων.

4. Ερμηνεία των Αποτελεσμάτων: Οι γιατροί αναλύουν τα αποτελέσματα του ECG για να ανιχνεύσουν ανωμαλίες όπως αρρυθμίες, ισχαιμία ή άλλα καρδιολογικά προβλήματα.

5.2 Ιατρική Σημασία του ECG στην Διάγνωση Καρδιολογικών Παθήσεων

Το Ηλεκτροκαρδιογράφημα (ECG) αποτελεί ένα από τα πιο θεμελιώδη εργαλεία στη διάγνωση και παρακολούθηση των καρδιολογικών παθήσεων, καθώς παρέχει σημαντικές πληροφορίες για την ηλεκτρική δραστηριότητα της καρδιάς. Η ικανότητα του ECG να ανιχνεύει ανωμαλίες στην καρδιακή λειτουργία το καθιστά απαραίτητο για τη διάγνωση διαταραχών του καρδιακού ρυθμού, την εκτίμηση της καρδιολογικής υγείας και την παρακολούθηση των αποτελεσμάτων θεραπείας.

1. Αρρυθμίες: Το ECG είναι εξαιρετικά χρήσιμο για την ανίχνευση διαταραχών του καρδιακού ρυθμού ή αρρυθμιών, όπως η κολπική μαρμαρυγή, η κοιλιακή ταχυκαρδία και η κολπική ταχυκαρδία. Αυτές οι παθήσεις ενδέχεται να προκαλέσουν σοβαρές επιπλοκές, όπως εγκεφαλικά επεισόδια ή καρδιακή ανεπάρκεια, και η έγκαιρη διάγνωσή τους μέσω ECG μπορεί να οδηγήσει σε αποτελεσματικότερη θεραπεία.

2. Ισχαιμία και Εμφράγματα: Το ECG είναι κρίσιμο για την ανίχνευση της ισχαιμίας, η οποία προκύπτει όταν η καρδιά δεν λαμβάνει επαρκή ποσότητα οξυγόνου λόγω στενώσεων στις αρτηρίες. Ενδείξεις ισχαιμίας ή εμφράγματος (καρδιακής προσβολής) συχνά εμφανίζονται ως ανωμαλίες στα κύματα του ECG, ιδιαίτερα στα κύματα ST και T.

3. Διαταραχές της Ηλεκτρικής Συγκρότησης της Καρδιάς: Το ECG βοηθά επίσης στην ανίχνευση διαταραχών στη συσκευή αγωγής της καρδιάς, όπως είναι το μπλοκ της αγωγής (AV block) ή το μπλοκ του δεξιού ή αριστερού σκέλους του κολποκοιλιακού συστήματος. Αυτές οι καταστάσεις επηρεάζουν τη συγχρονισμένη συστολή της καρδιάς και μπορεί να οδηγήσουν σε καρδιακή ανεπάρκεια ή άλλες σοβαρές επιπλοκές.

4. Παρακολούθηση Καρδιολογικών Παθήσεων: Το ECG χρησιμοποιείται όχι μόνο για τη διάγνωση, αλλά και για την παρακολούθηση ασθενών με καρδιολογικές παθήσεις, όπως η υπέρταση ή η καρδιακή ανεπάρκεια. Με τη συνεχιζόμενη παρακολούθηση του ECG, οι γιατροί μπορούν να παρακολουθούν την πρόοδο της κατάστασης και να προσαρμόζουν την θεραπεία ανάλογα με τις ανάγκες του ασθενούς.

Η ιατρική σημασία του ECG έγκειται στο γεγονός ότι παρέχει άμεση και αξιόπιστη εικόνα της καρδιακής δραστηριότητας, επιτρέποντας την έγκαιρη διάγνωση και τη σωστή κατεύθυνση της θεραπείας. Με τη βοήθεια του ECG, οι γιατροί μπορούν να αναγνωρίσουν και να αντιμετωπίσουν καρδιολογικές παθήσεις σε πρώιμο στάδιο, αποτρέποντας σοβαρές επιπλοκές και βελτιώνοντας τη ζωή των ασθενών.

5.3 Επεξεργασία Σήματος και Ερμηνεία στο ECG

Η επεξεργασία του σήματος στο Ηλεκτροκαρδιογράφημα (ECG) είναι κρίσιμη για την ακριβή διάγνωση καρδιολογικών παθήσεων. Τα σήματα που καταγράφονται από τα ηλεκτρόδια είναι συνήθως αδύναμα και επηρεάζονται από διάφορους θορύβους και διαταραχές. Επομένως, απαιτείται μια σειρά από τεχνικές επεξεργασίας για να εξασφαλιστεί η ακρίβεια και η χρησιμότητα των δεδομένων που παράγονται.

1. Φιλτράρισμα Θορύβου: Ένα από τα πρώτα βήματα στην επεξεργασία του σήματος ECG είναι η απομάκρυνση θορύβου, όπως ηλεκτρομαγνητική παρεμβολή, κίνηση του ασθενούς και άλλες εξωτερικές πηγές. Χρησιμοποιούνται διάφορα φίλτρα, όπως το φίλτρο χαμηλής ή υψηλής συχνότητας, για να απομονωθούν τα ανεπιθύμητα σήματα και να διατηρηθεί μόνο η καρδιολογική δραστηριότητα.

2. Ενίσχυση Σήματος: Δεδομένου ότι τα σήματα του ECG είναι συνήθως μικρού εύρους, χρησιμοποιούνται τεχνικές ενίσχυσης για να μεγαλώσουν τα σήματα και να τα κάνουν πιο ευδιάκριτα. Αυτό επιτρέπει την ανίχνευση ακόμα και μικρών ανωμαλιών που μπορεί να μην ήταν ορατές χωρίς επεξεργασία.

3. Εντοπισμός και Ανάλυση Κύματος: Στην ανάλυση του ECG, τα κύματα P, QRS και T αναγνωρίζονται και απομονώνονται. Κάθε ένα από αυτά τα κύματα αντιστοιχεί σε διαφορετικές φάσεις της καρδιακής δραστηριότητας: το κύμα P αντιστοιχεί στη συστολή των κόλπων, το QRS στη συστολή των κοιλιών και το T στην ανάπαυση της καρδιάς. Ο ακριβής εντοπισμός αυτών των κυμάτων είναι απαραίτητος για την ανίχνευση ανωμαλιών.

4. Ανίχνευση Ανωμαλιών και Διάγνωση: Η ερμηνεία του ECG περιλαμβάνει την ανάλυση των διαφορών στην καταγραφή των κυμάτων για να εντοπιστούν πιθανές καρδιολογικές παθήσεις. Σημεία ενδιαφέροντος περιλαμβάνουν τις ανωμαλίες του κύματος ST, τις αλλαγές στα διαστήματα PR και QT και την παρουσία ανωμαλιών στους ρυθμούς, όπως η κολπική μαρμαρυγή ή οι αρρυθμίες.

5. Αυτοματοποιημένη Ερμηνεία μέσω Τεχνητής Νοημοσύνης (AI): Τα τελευταία χρόνια, η τεχνητή νοημοσύνη και οι αλγόριθμοι μηχανικής μάθησης χρησιμοποιούνται για την αυτόματη ανάλυση του ECG. Αυτές οι τεχνολογίες προσφέρουν γρήγορη και ακριβή διάγνωση, ενισχύοντας την ικανότητα των γιατρών να εντοπίζουν επικίνδυνες καταστάσεις σε πρώιμο στάδιο.

Η επεξεργασία και ερμηνεία του ECG είναι θεμελιώδης για τη διάγνωση και παρακολούθηση καρδιολογικών καταστάσεων. Με τη χρήση εξελιγμένων τεχνικών επεξεργασίας σήματος και σύγχρονων τεχνολογιών ανάλυσης, η ακρίβεια και η ταχύτητα της διάγνωσης έχουν βελτιωθεί σημαντικά, δίνοντας στους γιατρούς ισχυρά εργαλεία για την καλύτερη φροντίδα των ασθενών.

Ασφάλεια και Κανονιστικοί Παράγοντες στις Συσκευές ECG

Η ασφάλεια και η συμμόρφωση με τους κανονιστικούς παράγοντες αποτελούν κρίσιμους παράγοντες για την ανάπτυξη και την κυκλοφορία συσκευών Ηλεκτροκαρδιογραφήματος (ECG) στην αγορά. Καθώς αυτές οι συσκευές χρησιμοποιούνται για τη διάγνωση και παρακολούθηση καρδιολογικών καταστάσεων, είναι απαραίτητο να πληρούν αυστηρές προδιαγραφές για να διασφαλίσουν την ακριβή και ασφαλή λειτουργία τους.

1. Ασφάλεια Δεδομένων και Προστασία Ιδιωτικότητας: Στις σύγχρονες ασύρματες συσκευές ECG, η ασφάλεια των δεδομένων είναι υψίστης σημασίας. Καθώς τα δεδομένα που καταγράφονται είναι προσωπικά και ιατρικά ευαίσθητα, απαιτούν κρυπτογράφηση και ασφαλείς μεθόδους μεταφοράς για την αποφυγή παραβίασης της ιδιωτικότητας του ασθενούς. Οι κανονισμοί, όπως ο Γενικός Κανονισμός Προστασίας Δεδομένων (GDPR) στην Ευρωπαϊκή Ένωση ή ο Νόμος για τη Μεταχείριση Ιατρικών Δεδομένων (HIPAA) στις ΗΠΑ, απαιτούν αυστηρή προστασία των προσωπικών δεδομένων.

2. Κανονισμοί για Ιατρικές Συσκευές: Οι συσκευές ECG υπόκεινται σε αυστηρούς κανονιστικούς ελέγχους από διεθνείς οργανισμούς, όπως ο Οργανισμός Τροφίμων και Φαρμάκων (FDA) στις ΗΠΑ και η Ευρωπαϊκή Υπηρεσία Ιατρικών Συσκευών (MDR) στην ΕΕ. Αυτοί οι κανονισμοί διασφαλίζουν ότι οι συσκευές πληρούν τα πρότυπα ασφαλείας και απόδοσης πριν από την έγκρισή τους για εμπορική χρήση. Οι συσκευές ECG πρέπει να συμμορφώνονται με τις απαιτήσεις για την ποιότητα κατασκευής, την ακριβή μέτρηση των καρδιακών σημάτων και τη μακροχρόνια αξιοπιστία.

3. Διαχείριση Ρίσκου και Έλεγχοι Ποιότητας: Η διαχείριση ρίσκου είναι απαραίτητη για την ασφαλή λειτουργία των συσκευών ECG. Κάθε συσκευή πρέπει να υποβάλλεται σε σειρά δοκιμών για να εντοπιστούν και να ελαχιστοποιηθούν πιθανοί κίνδυνοι, όπως βλάβες από υπερθέρμανση, ηλεκτροπληξία ή λανθασμένα δεδομένα. Οι κανονισμοί απαιτούν επίσης αυστηρές διαδικασίες ποιοτικού ελέγχου για να διασφαλιστεί ότι οι συσκευές πληρούν τις απαιτούμενες προδιαγραφές.

4. Διασφάλιση Εμπιστοσύνης στους Χρήστες: Η συμμόρφωση με τις κανονιστικές απαιτήσεις βοηθά στη διασφάλιση της εμπιστοσύνης των χρηστών, τόσο των ιατρών όσο και των ασθενών. Η πιστοποίηση και η έγκριση των συσκευών ECG από αναγνωρισμένους οργανισμούς επιβεβαιώνει ότι η συσκευή είναι αξιόπιστη, ακριβής και ασφαλής για χρήση στον ιατρικό τομέα.

Η ασφάλεια και οι κανονιστικοί παράγοντες είναι θεμελιώδης για την ανάπτυξη και την εφαρμογή συσκευών ECG. Η αυστηρή τήρηση των προτύπων και των κανονισμών εξασφαλίζει την αξιόπιστη λειτουργία των συσκευών και την προστασία της υγείας των ασθενών.

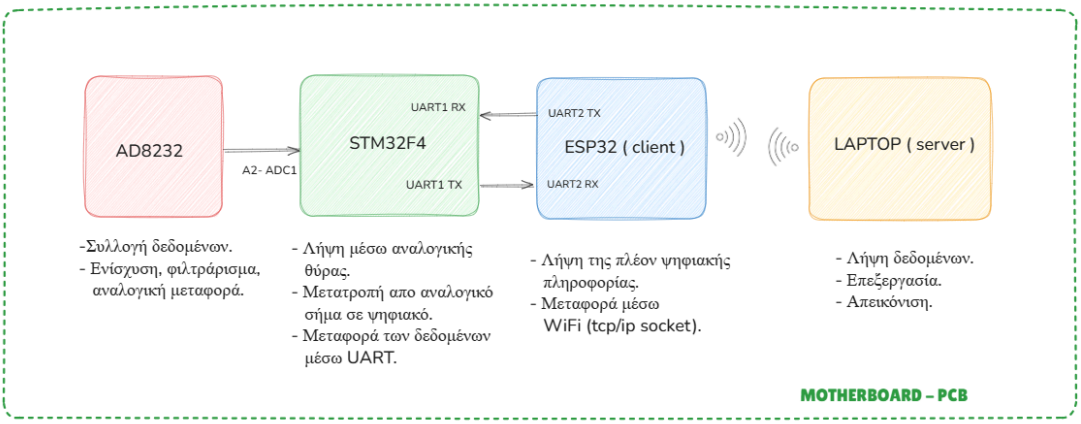
6. Υλοποίηση end-to-end συστήματος παρακολούθησης καρδιακής δραστηριότητας

6.1 Overview

Στην παρούσα εργασία υλοποιήθηκε ένα ολοκληρωμένο σύστημα καταγραφής, μεταφοράς και απεικόνισης βιοϊατρικού σήματος ECG, ενταγμένο στο πλαίσιο των τεχνολογιών Internet of Medical Things (IoMT). Το σύστημα βασίστηκε στη συνεργασία τριών επιμέρους modules: του **AD8232** για την ανίχνευση του καρδιογραφήματος, του **STM32F4** για την ADC μετατροπή του σήματος, και του **ESP32** για την ασύρματη επικοινωνία. Το ECG σήμα λαμβάνεται μέσω του AD8232 και μεταφέρεται στον STM32 μέσω αναλογικής εισόδου, όπου ψηφιοποιείται και αποστέλλεται μέσω UART στον ESP32. Ο ESP32 λειτουργεί ως TCP/IP client, αποστέλλοντας σε πραγματικό χρόνο τα δεδομένα σε έναν server που υλοποιείται σε Python στον υπολογιστή.

Στην πλευρά του laptop, χρησιμοποιήθηκε multithreading: το κύριο νήμα (main thread) διαχειρίζεται το γραφικό περιβάλλον χρήστη (GUI) και την απεικόνιση του ECG σήματος, ενώ το βοηθητικό νήμα (child thread) αναλαμβάνει τη σύνδεση με τον ESP32, τη λήψη των δεδομένων και την αποστολή τους στο κύριο νήμα για απεικόνιση. Έτσι επιτυγχάνεται μια πλήρης, real-time ροή δεδομένων από την αρχική λήψη του καρδιογραφήματος μέχρι και την οπτική παρουσίασή του, συγκροτώντας ένα end-to-end IoMT σύστημα.

Τέλος, για τη φυσική ολοκλήρωση του έργου, σχεδιάστηκε και κατασκευάστηκε μια πλατφόρμα τύπου **motherboard**, πάνω στην οποία τοποθετήθηκαν και διασυνδέθηκαν όλα τα επιμέρους modules. Με αυτόν τον τρόπο εξασφαλίστηκε η μηχανική σταθερότητα, η λειτουργική συνέργεια των υπομονάδων και η ευκολία επαναλαμβανόμενων πειραμάτων και δοκιμών.



εικόνα 1.1 διάγραμμα ροής της συνολικής υλοποίησης του συστήματος

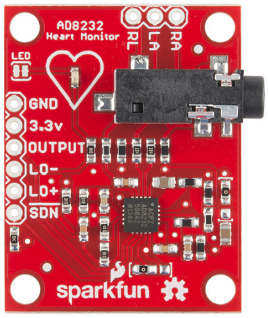
6.2 Κοστολόγηση υλικών

Προϊόν	Κόστος (€)
AD8232-SEN12650 + Leads	26
STM32F446re-Nucleo Board	25
ESP32 DEVKIT-V1	12
PCB (motherboard)	9
Αναλώσιμα	10
Συνολικό κόστος	**82 €**

πίνακας 1.1 : κοστολόγηση υλικών

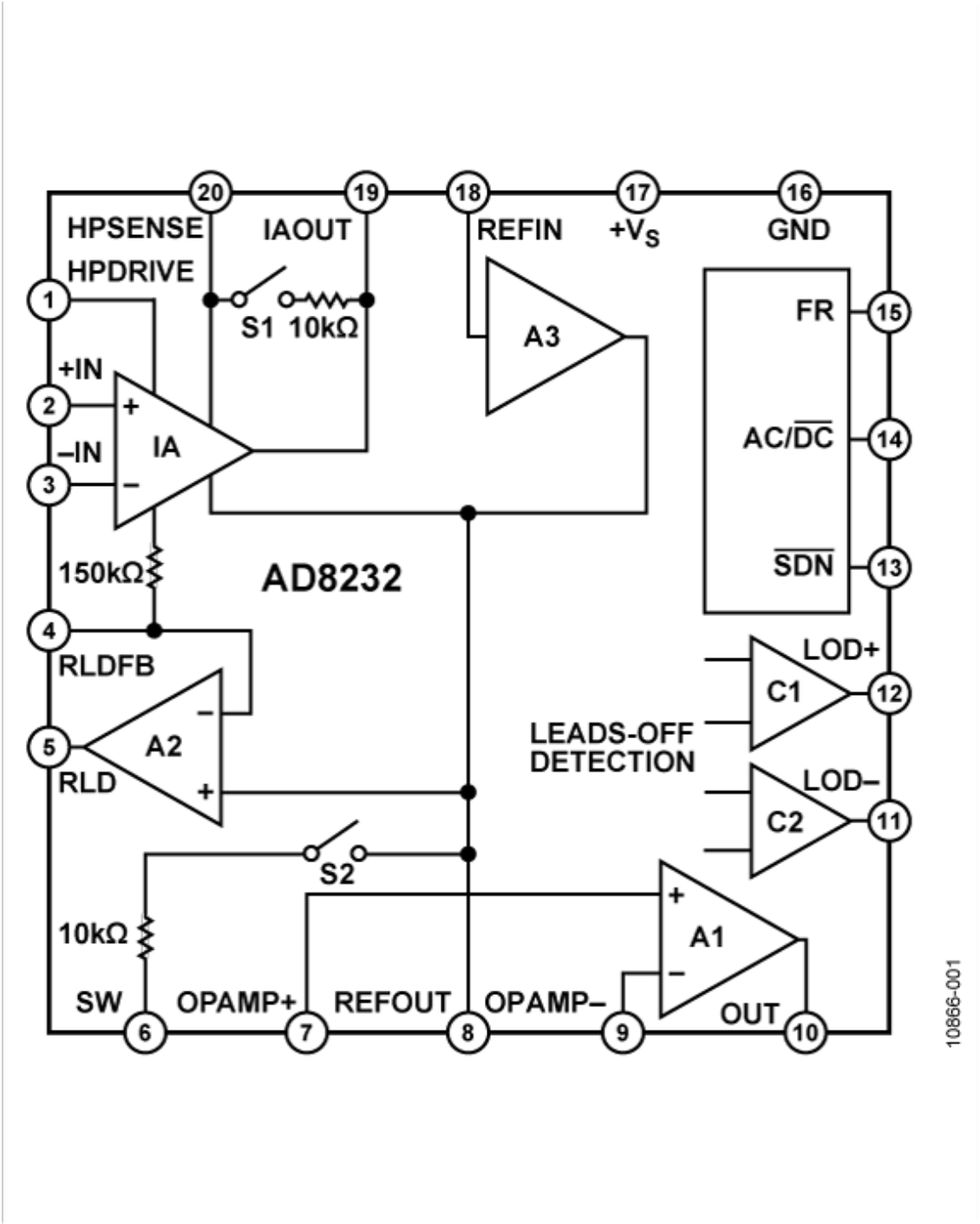
6.3 Το AD8232 - AD8232-SEN12650 module

Ο **AD8232** είναι ένα ενσωματωμένο ολοκληρωμένο κύκλωμα χαμηλής κατανάλωσης που έχει σχεδιαστεί ειδικά για την ανίχνευση καρδιακού ρυθμού μέσω ηλεκτροκαρδιογραφήματος (ECG). Είναι ένα εξαιρετικά δημοφιλές chip για φορητές ιατρικές και wearable εφαρμογές, καθώς προσφέρει μεγάλη ακρίβεια, χαμηλή κατανάλωση ενέργειας και μικρό μέγεθος, καθιστώντας το ιδανικό για την παρακολούθηση καρδιολογικών σημάτων σε πραγματικό χρόνο.



εικόνα 2.1 : AD8232-SEN12650 module

Από τεχνικής άποψης, το AD8232 ενσωματώνει έναν ενισχυτή οργανολογίας με ρυθμιζόμενο gain, φίλτρα υψηλής και χαμηλής διέλευσης (high-pass και low-pass), καθώς και κυκλώματα αναφοράς και αποκατάστασης. Όλα αυτά τα στοιχεία βοηθούν στη σταθεροποίηση του σήματος και την απόρριψη κοινής τάσης (common-mode rejection), κάτι κρίσιμο για ιατρικά σήματα. Λειτουργεί σε ευρύ φάσμα τάσεων τροφοδοσίας και υποστηρίζει λειτουργία sleep mode για εξοικονόμηση ενέργειας.



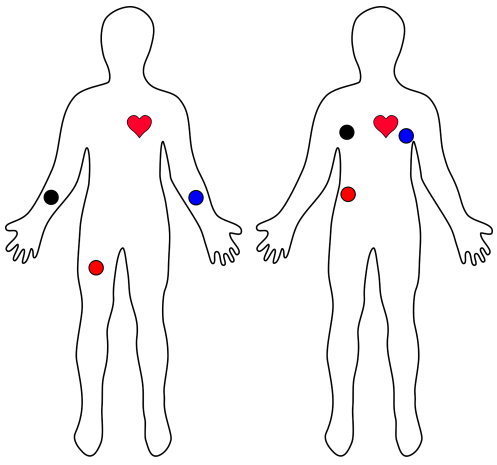
εικόνα 2.2 : Αρχιτεκτονική του AD8232

Τεχνικά χαρακτηριστικά του AD8232:

- Τάση λειτουργίας: 1.7V έως 3.5V
- Ρεύμα κατανάλωσης: 170 μA (τυπικό)
- Bandwidth : 0.5 Hz έως 40 Hz
- Common-mode rejection ratio (CMRR): 80 dB
- Slew rate: 0.3 V/μs
- Ενσωματωμένος ενισχυτής (instrumentation amplifier)
- Φίλτρο απόρριψης θορύβου 50/60Hz (ρυθμιζόμενο)
- Ενσωματωμένος high-pass και low-pass φιλτραρισμός
- Λειτουργία shutdown (sleep mode) για εξοικονόμηση ενέργειας
- Μικρό μέγεθος πακέτου: 4 mm x 4 mm
- Κατάλληλο για εφαρμογές ECG, fitness monitoring και portable instrumentation

6.3.1 LEAD placement

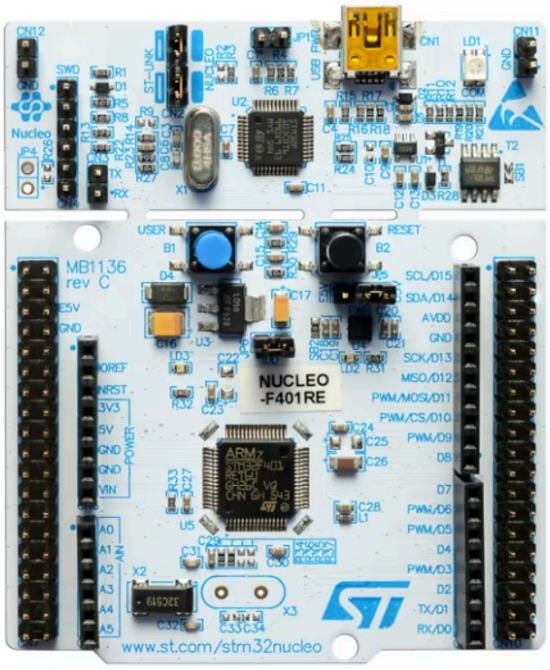
Όπως βλέπουμε στην παρακάτω εικόνα τα καλώδια/ακροδέκτες μπορούν να τοποθετηθούν με 2 διαφορετικούς τρόπους, είτε στα άκρα είτε κοντά στους ώμους. Τα 2 leads είναι για την λήψη του σήματος ενώ το τρίτο χρησιμεύει σαν σημείο αναφοράς.



εικόνα 2.3: Τοποθέτηση των Leads

7. Ο μικροελεγκτής STM32F446RE-NUCLEO BOARD

Ο STM32F446 είναι ένας ισχυρός μικροελεγκτής της σειράς STM32F4 της STMicroelectronics, σχεδιασμένος για απαιτητικές εφαρμογές σε βιομηχανικό αυτοματισμό, embedded συστήματα και ήχο. Βασίζεται στον ARM Cortex-M4 πυρήνα, με υποστήριξη DSP και FPU (Floating Point Unit), προσφέροντας υψηλή απόδοση και υπολογιστική ακρίβεια.



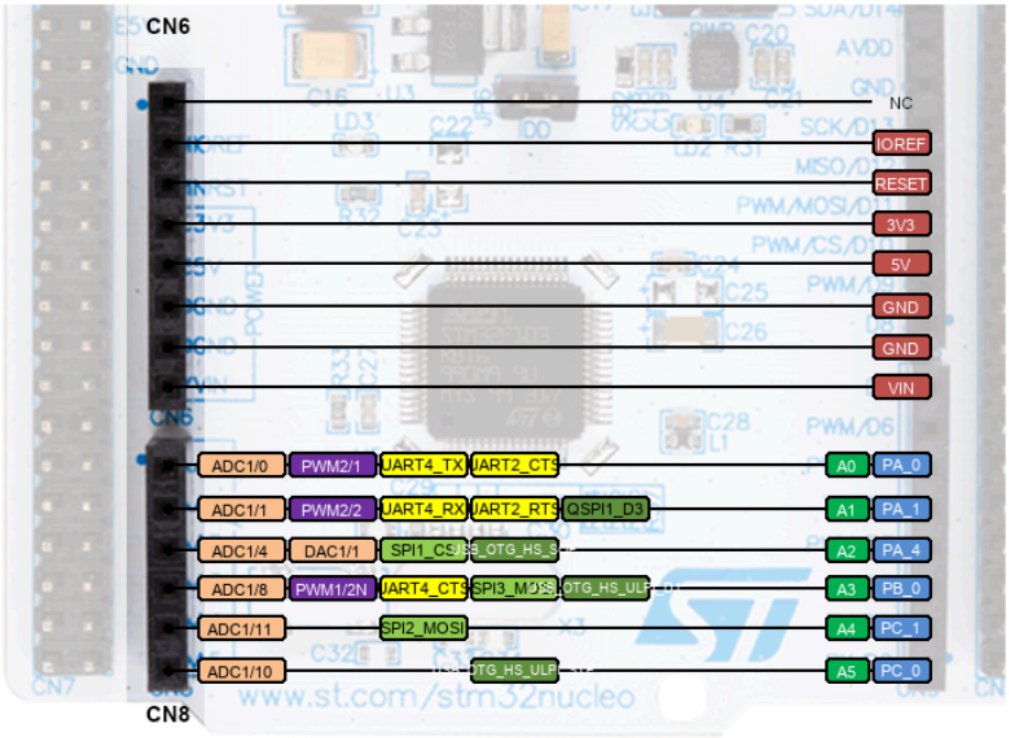
εικόνα 3.1: stm32f446 nucleo board

Βασικά Χαρακτηριστικά (Specs):

- CPU: ARM Cortex-M4 στα 180 MHz, με FPU & DSP
- Flash Memory: έως 512 KB
- RAM: έως 128 KB SRAM
- Timers: 14 συνολικά (γενικής χρήσης, προηγμένοι, watchdogs κ.ά.)
- ADC: 3 × 12-bit ADCs, έως 2.4 MSPS, με έως 16 κανάλια
- DAC: 2 × 12-bit DACs
- USARTs/UARTs: 4 × USART + 2 × UART

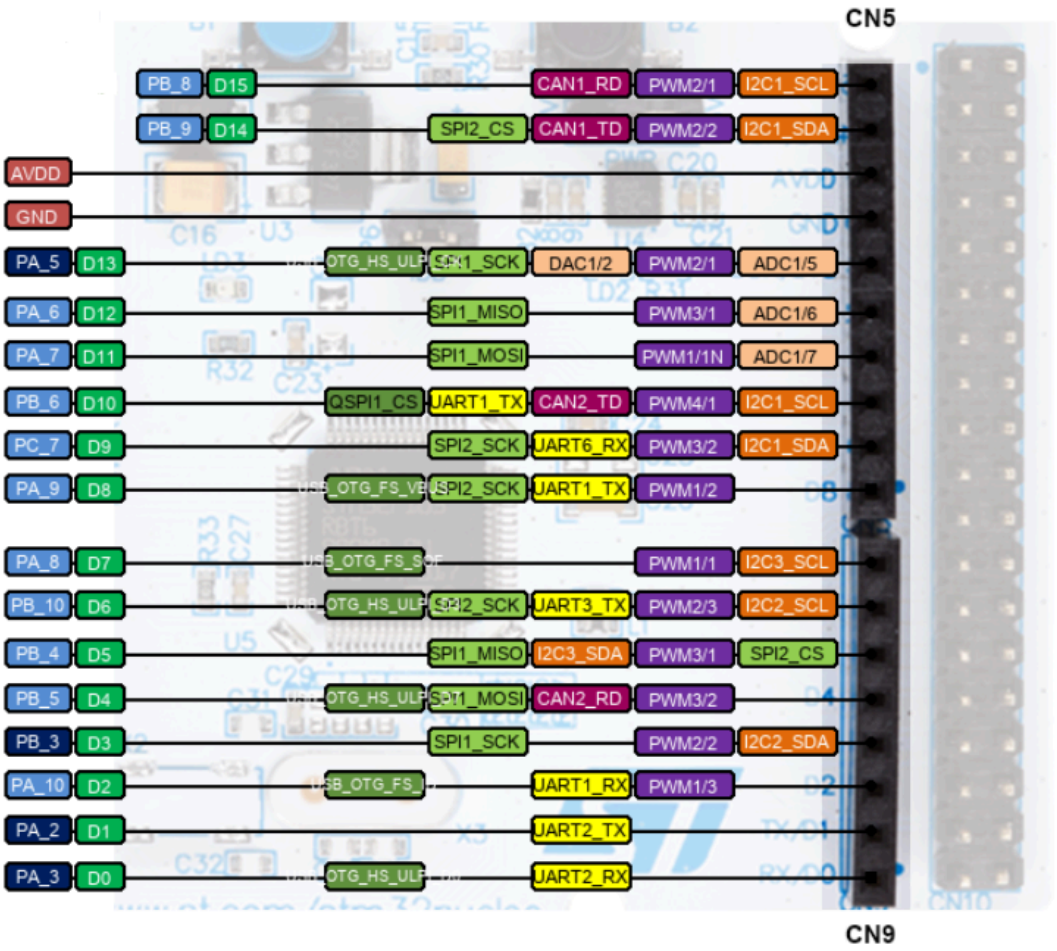
- SPIs: 3 × SPI + 1 × I²S (με πλήρη υποστήριξη I2S audio)
- I²C: 3 × I²C interfaces
- USB: USB OTG FS και OTG HS (με ενσωματωμένο PHY για FS)
- SDIO: για σύνδεση με κάρτες SD/MMC
- CAN: 2 × CAN 2.0B
- FMC: Flexible Memory Controller για εξωτερική SDRAM, SRAM, NOR/NAND
- GPIOs: έως 114 γραμμές I/O (σε πακέτο LQFP144)

Παρακάτω μπορούμε να δούμε το κανάλι 6 (CN6) και το κανάλι 8 (CN8) του STM32F446.



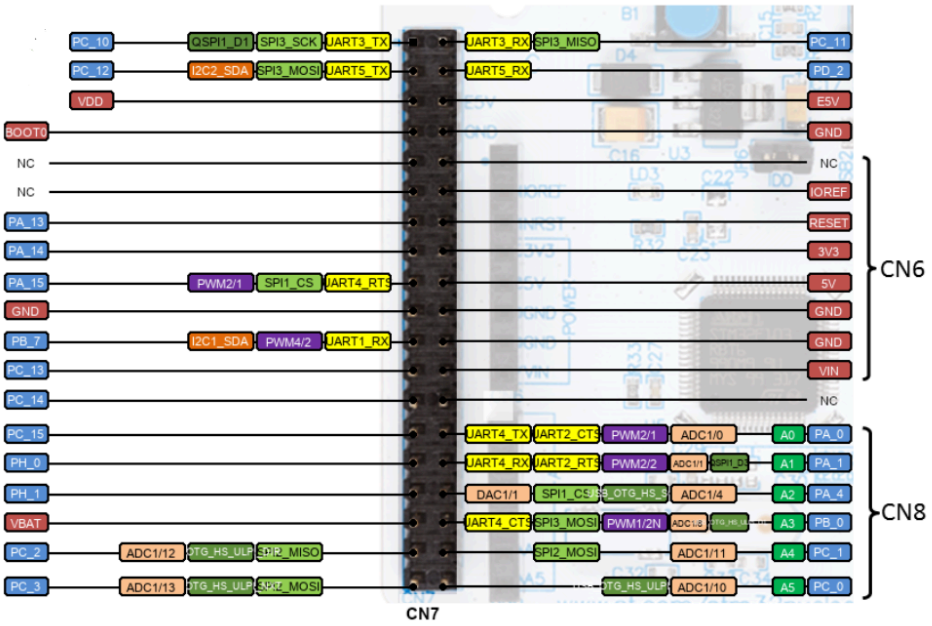
εικόνα 3.2: CN8 pins

Παρακάτω μπορούμε να δούμε το κανάλι 5 (CN5) και το κανάλι 9 (CN9) του STM32F446.



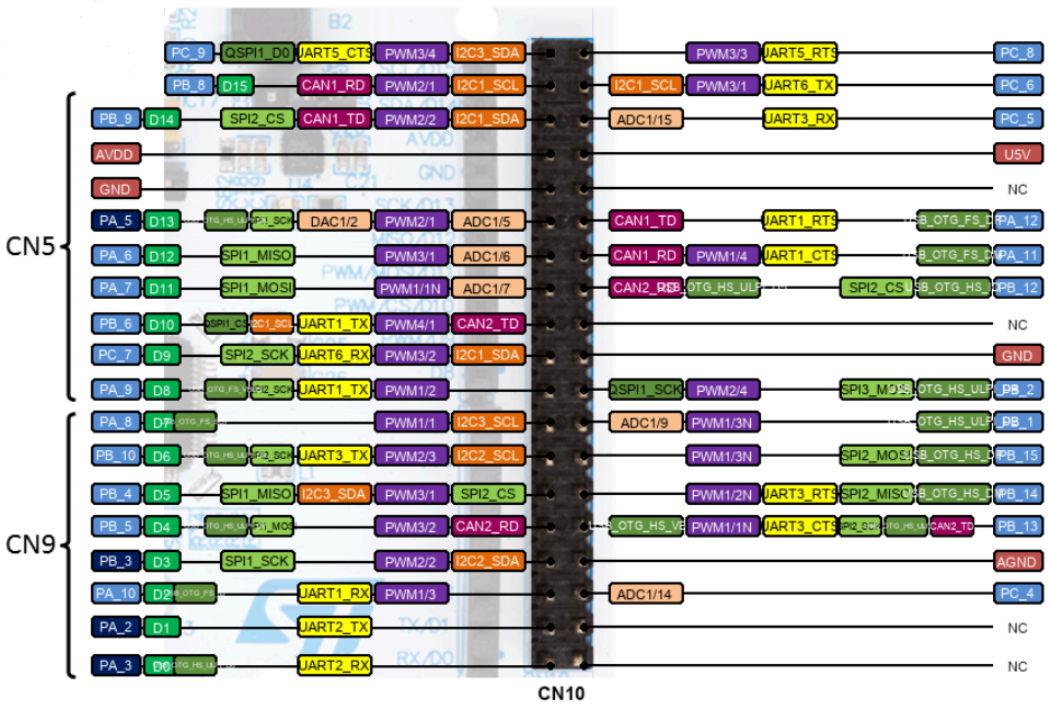
εικόνα 3.3: CN9 pins

Παρακάτω μπορούμε να δούμε το κανάλι 7 (CN7) του STM32F446.



εικονα 3.4: CN7 pins

Τέλος έχουμε το κανάλι 10 (CN 10).



εικόνα 3.5: CN 10 pins

Ο STM32F446 είναι κατάλληλος για:

- Ψηφιακή επεξεργασία σήματος (ήχος, αισθητήρες)
- Ενσωματωμένα real-time συστήματα
- Βιομηχανικά προϊόντα με ανάγκη σε πολλαπλά interfaces (UART, SPI, CAN, USB)

7.1 Παραμετροποίηση και προγραμματισμός του μικροελεγκτή STM32F446RE

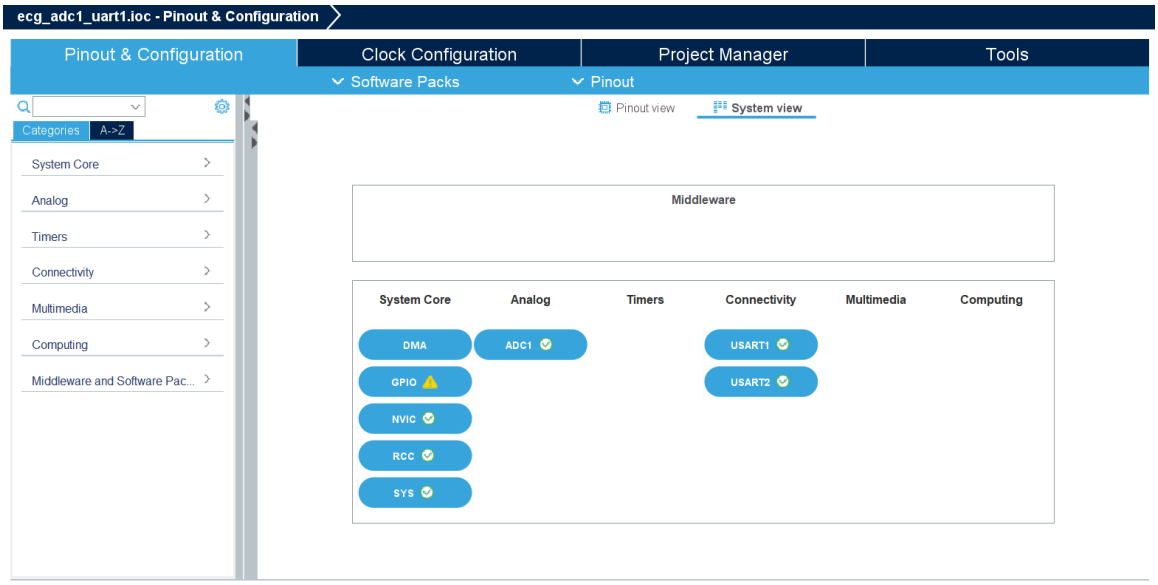
7.1.1 Χρήση του εργαλείου Cube MX

Πρόκειται για ένα εργαλείο προσομοίωσης, οπτικοποίησης και παραμετροποίησης το οποίο βρίσκεται ενσωματωμένο στο stm32CubeIDE (Integrated Development Environment).
Χρησιμεύει στην αρχικοποίηση των αρχιτεκτονικών που θα επιλέξουμε να χρησιμοποιήσουμε.
Η βασική του λειτουργικότητα είναι η παραγωγή κώδικα ο οποίος περιέχει κυρίως την ενεργοποίηση και αρχικοποίηση υποσυστημάτων με αποτέλεσμα την δραστική μείωση του χρόνου και τον λαθών που βασίζονται σε initial configuration.

7.1.2 System View

Παρακάτω φαίνεται το system view μέσα από το περιβάλλον του Cube MX στο οποίο αναγράφονται κατηγοριοποιημένα και μονολεκτικά της παραμετροποίησης μας. Όπως ήδη έχουμε αναφέρει οι κύριες λειτουργίες που αξιοποιήσαμε είναι ο ADC1 (pin A2 ή PA_4) και το UART1 (pin PA_9 ή D8 για το UART1_TX και pin PA_10 ή D2 για το UART_RX αντίστοιχα).

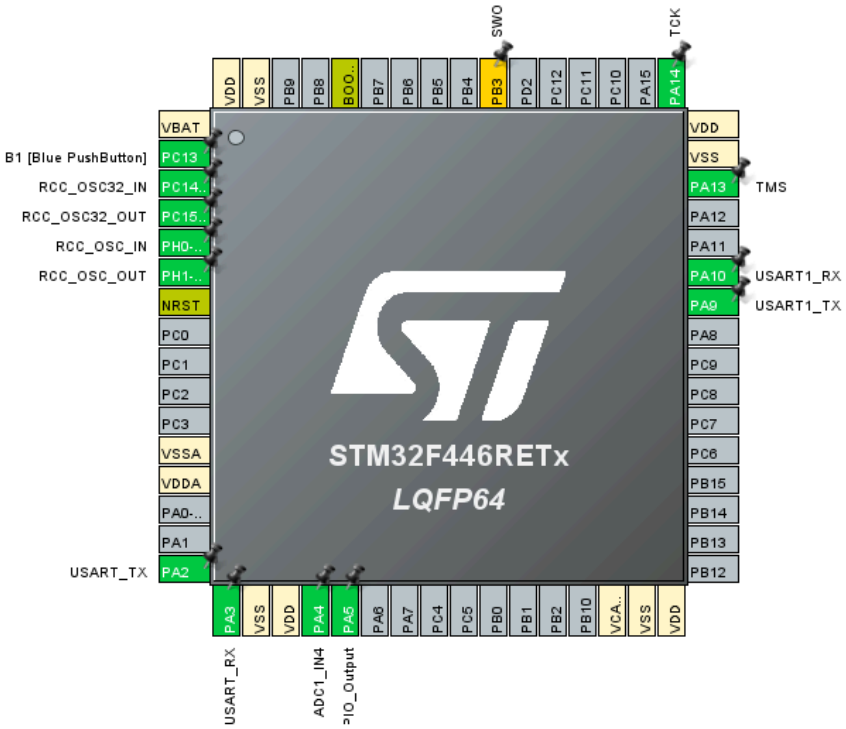
Ο UART 2 Χρησιμοποιείται για debugging/printf output μέσω ST-LINK. Το STM32CubeIDE/CubeMX το ενεργοποιεί αυτόματα για να σου επιτρέψει να χρησιμοποιήσεις τη λειτουργία retargeting printf() σε UART. Είναι συνδεδεμένο εσωτερικά στο virtual COM port του ST-LINK. Όταν συνδέεις τη Nucleo board στον υπολογιστή σου μέσω USB, εμφανίζεται ένας virtual COM port (π.χ. COM3), ο οποίος είναι συνδεδεμένος με το USART2. Δεν προγραμματίζουμε τον controller μέσω UART2. Ο προγραμματισμός (flashing) γίνεται μέσω SWD (Serial Wire Debug) που υποστηρίζεται από τον ST-LINK debugger – όχι από το UART2.



εικόνα 3.6: Cube MX environment

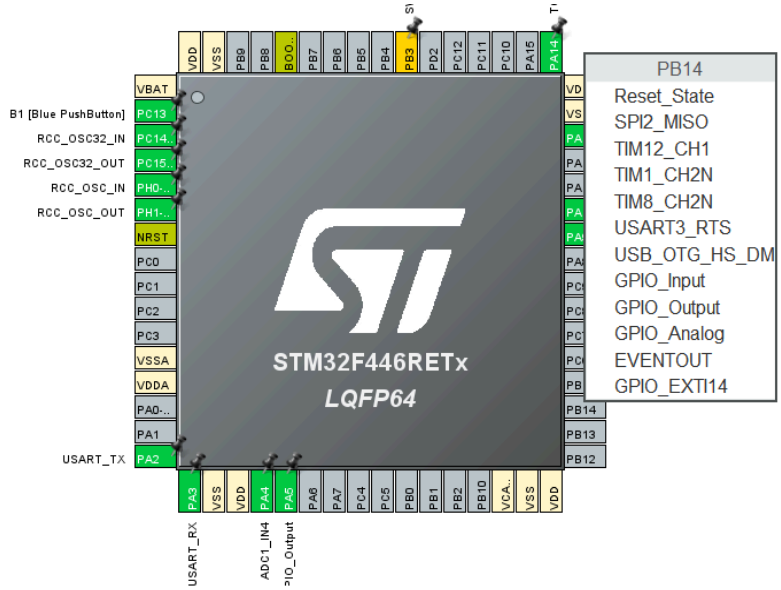
7.1.3 Pinout view

Το pinout view είναι ένα αρκετά χρήσιμο και ενδιαφέρον εργαλείο της ST το οποίο επίσης περιέχεται στο Cube MX. Ουσιαστικά είναι μια προσομοίωση το πραγματικού controller όπου επιλεγοντας το αντίστοιχο pin μπορεις να ορίσεις και να ενεργοποιήσεις την αντίστοιχη λειτουργια που αυτο υποστηρίζει. Στην δικιά μας περίπτωση όπως βλέπουμε φαινεται ξεκάθαρα η χρήση του USART 2 και USART 1 καθώς και η χρήση του ADC1 _IN4. Το πράσινο προσδιορίζει μια υγιείς λειτουργία και σύνδεση. Το σκούρο κίτρινο χρησιμοποιείται για warnings και conflicts αλλά πολλες φορές πρόκειται για bug του προγραμματος. Στην συγκεκριμένη περίπτωση δεν μας επηρεάζει αρνητικά. Το κόκκινο είναι για Error, κάτι συνδέθηκε ή στύθηκε λάθος και δεν μπορεί να γίνει build.



εικόνα 3.7: pinout view

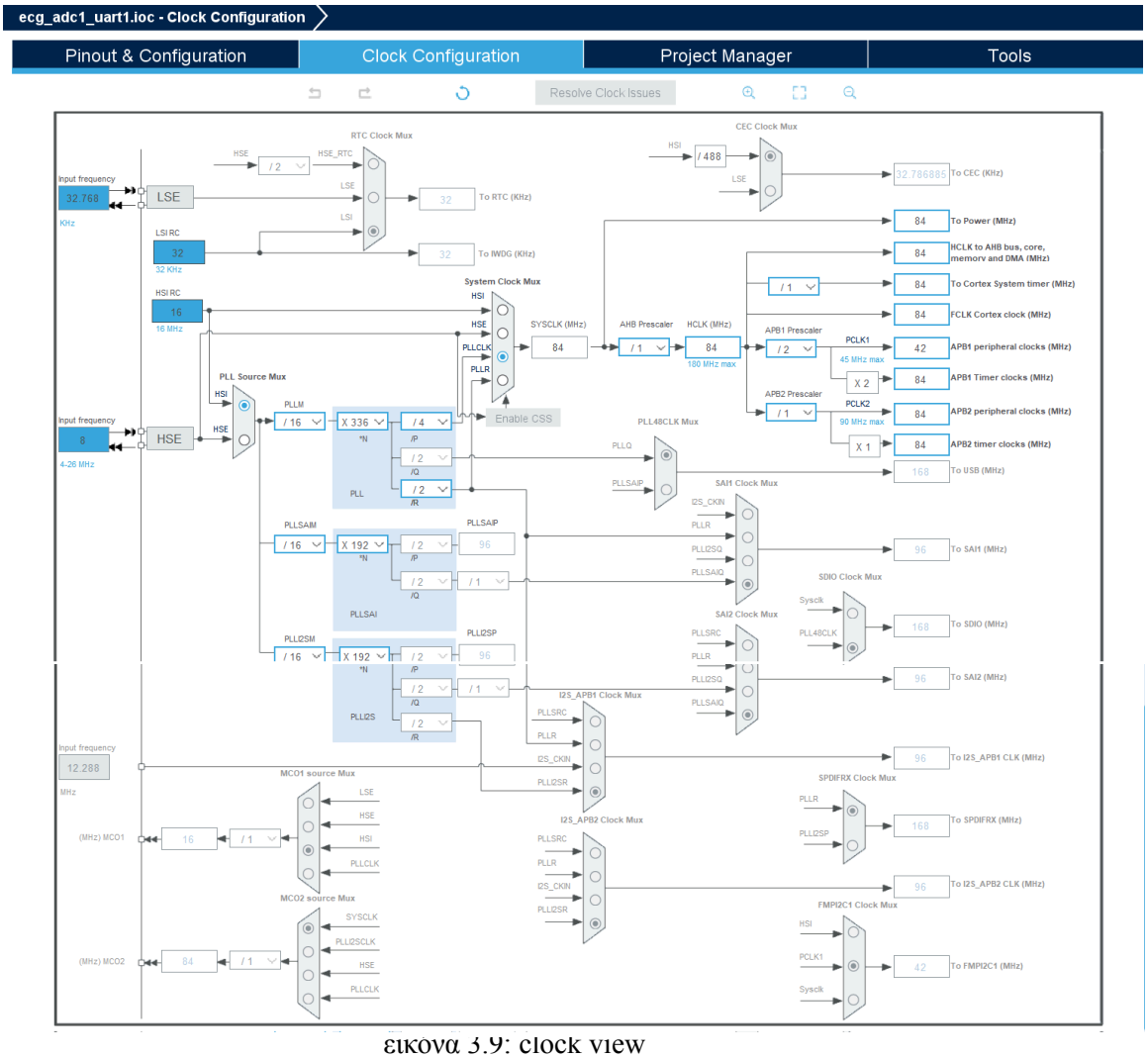
Επίσης πατώντας πάνω στα pins μπορείς να δείς τις λειτουργίες που υποστηρίζουν και να επιλέξεις αυτήν που σου ταιριάζει περισσότερο. Εδώ για παράδειγμα μπορούμε να δούμε τις λειτουργιες που υποστηρίζει ενα τυχαίο pin, το pin PB 14.



εικόνα 3.8: pinout view

7.1.4 Clock configuration

Ακόμα μια πολύ ιδιαίτερη δυνατότητα που σου δίνει το Cube MX είναι ότι μπορείς να δείς και να ρυθμίσεις το clock, τους πολυπλέκτες (Mux) και τους prescalers που οι συγκεκριμένες αρχιτεκτονικές διαθέτουν όπως βλέπουμε στην παρακάτω φωτογραφία. Στα δεξιά πλαίσια αναγράφονται οι τελικές συχνότητες που θα καταλήξουν στα εκάστοτε υποσυστήματα. Με αυτόν τον τρόπο μπορούμε να πάρουμε την σωστή συχνότητα βάση των αναγκών μας. Έτσι αυξάνουμε την πιθανότητα μιας επιτυχημένης παραμετροποίησης.



εικόνα 3.9: clock view

Στον παρακάτω πίνακα μπορούμε να δούμε μερικά από τα prescaler που χρησιμοποιήσαμε.

Περιφερειακό	**Bus**	**Prescaler**	**Συχνότητα**
USART1	APB2	APB2 prescaler	84 Mhz

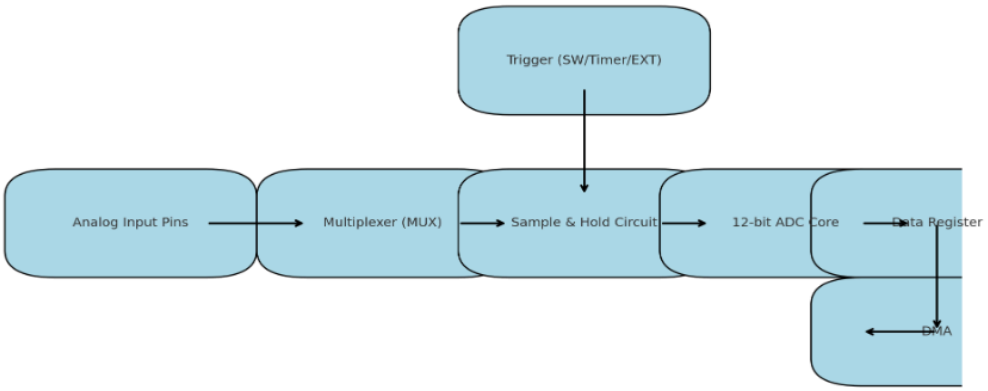
USART2	APB1	APB1 prescaler	84 Mhz
ADC1	APB2 (αλλά με δικό του clock από ADC prescaler)	ADC prescaler (μέσα στο RCC)	84 Mhz

7.1.5 Analog to digital conversion (ADC)

Ο STM32F446 διαθέτει τρεις ανεξάρτητους μετατροπείς τάσης analog digital converter (ADC1, ADC2, ADC3) 12-bit ανάλυσης, με δυνατότητα δειγματοληψίας έως 2.4 MSPS ο καθένας. Υποστηρίζουν λειτουργίες single, continuous, scan και discontinuous mode, καθώς και trigger από εξωτερικά events ή timers. Οι ADCs μπορούν να λειτουργούν ανεξάρτητα ή συγχρονισμένα, με δυνατότητα triple interleaved mode για ταχύτητα έως 7.2 MSPS.
Κάθε ADC έχει 16 κανάλια εισόδου, εκ των οποίων κάποια συνδέονται με εσωτερικές πηγές (όπως θερμοαισθητήρας) ενώ άλλα είναι GPIOs που μπορούν να διαμορφωθούν ως αναλογικές είσοδοι. Επιπλέον, υπάρχει υποστήριξη DMA και Analog Watchdog για την επιτήρηση αναλογικών τιμών σε πραγματικό χρόνο.

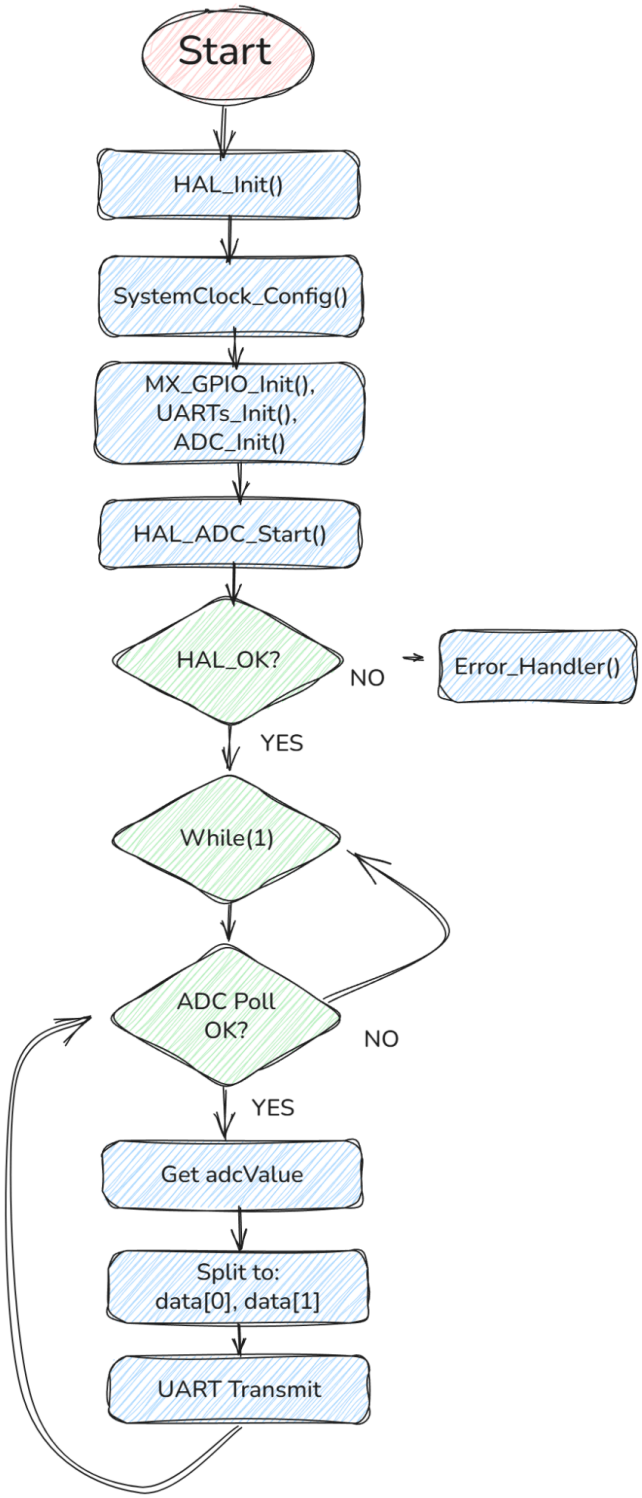
7.1.6 Analog digital converter 1 (ADC 1)

Ο ADC1 στον STM32F446 είναι ο βασικός και πληρέστερα υποστηριζόμενος ADC, συνδεδεμένος στο APB2 bus. Μπορεί να δειγματοληπτεί έως 16 κανάλια και υποστηρίζει 12-bit ανάλυση, με επιλέξιμο χρόνο δειγματοληψίας για κάθε κανάλι (από 3 έως 480 κύκλους). Το ρολόι του ρυθμίζεται μέσω prescaler από το ADC common clock, επιτρέποντας ευελιξία ανάλογα με τις απαιτήσεις ακρίβειας και ταχύτητας.
Ο ADC1 μπορεί να χρησιμοποιηθεί σε dual regular ή injected mode μαζί με τον ADC2 για πιο σύνθετες μετρήσεις. Υποστηρίζει triggering από timers (π.χ. TIM1 CC1) και μπορεί να συνδεθεί με το DMA2 για αυτόματη μεταφορά δεδομένων χωρίς συμμετοχή της CPU.
Αξιοποιήθηκε για την λήψη του σήματος μας από το AD8232 module.



εικόνα 3.10: ADC architecture

7.1.7 Διάγραμμα Ροής του STM32-F446RE firmware



εικόνα 3.11: Διάγραμμα Ροής του STM32-F446RE firmware

7.1.8 Clock configuration

Αυτό το κομμάτι του κώδικα είναι υπεύθυνο για τη διαμόρφωση του συστήματος χρονισμού (System Clock Configuration) του STM32F446. Ουσιαστικά καθορίζει από πού προέρχεται ο κύριος χρονισμός (ρολόι) του συστήματος και πώς διανέμεται στους διάφορους υποσυστήματα του μικροελεγκτή (CPU, AHB, APB1, APB2).

Ο στόχος εδώ είναι να παραχθεί σταθερό σύστημα χρονισμού στα 84 MHz (336 MHz / 4), χρησιμοποιώντας τον εσωτερικό ταλαντωτή HSI (16 MHz) και έναν PLL πολλαπλασιαστή.

Ανάλυση παραμετροποίησης:

- `__HAL_RCC_PWR_CLK_ENABLE()`: Ενεργοποιεί το ρολόι στο υποσύστημα τροφοδοσίας (PWR), για να μπορούμε να ρυθμίσουμε τάσεις.

- `__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE3)`: Ρυθμίζει τον εσωτερικό ρυθμιστή τάσης σε Scale 3, ώστε να υποστηρίξει λειτουργία με χαμηλή κατανάλωση (υποστηρίζει max συχνότητα ≈ 100 MHz).

- `RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI`: Δηλώνουμε ότι χρησιμοποιούμε τον HSI (High Speed Internal) ταλαντωτή στα 16 MHz.

- `RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON`: Ενεργοποιούμε το PLL (Phase Locked Loop), ώστε να πολλαπλασιάσουμε τη συχνότητα.

- `RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI`: Ο PLL θα πάρει είσοδο από τον HSI (16 MHz).

- `PLLM = 16`: Διαιρούμε τη συχνότητα εισόδου από 16 → 16 MHz / 16 = 1 MHz εισόδου στον PLL.

- `PLLN = 336`: Πολλαπλασιασμός στο PLL → 1 MHz × 336 = 336 MHz VCO output.

- `PLLP = DIV4`: Τελική συχνότητα συστήματος = 336 MHz / 4 = 84 MHz SYSCLK.

- `PLLQ = 2, PLLR = 2`: Οι τιμές αυτές σχετίζονται με άλλες εξόδους του PLL (π.χ. USB), δεν χρησιμοποιούνται άμεσα εδώ.

- `RCC_ClkInitStruct.ClockType =` : Καθορίζουμε ποιοι δίαυλοι θα ρυθμιστούν: HCLK, SYSCLK, PCLK1, PCLK2.

- `SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK`: Ορίζουμε ότι το σύστημα θα δουλεύει από την έξοδο του PLL = 84 MHz.
- `AHBCLKDivider = DIV1`: HCLK (CPU/Bus) = SYSCLK / 1 = 84 MHz.
- `APB1CLKDivider = DIV2`: PCLK1 = 84 MHz / 2 = 42 MHz (π.χ. για USART2, TIM2...).
- `APB2CLKDivider = DIV1`: PCLK2 = 84 MHz (π.χ. για USART1, TIM1...).
- `FLASH_LATENCY_2`: Λαμβάνοντας υπόψη την ταχύτητα του SYSCLK, η flash χρειάζεται 2 κύκλους καθυστέρηση (wait states).

```
void SystemClock_Config(void)
{
 RCC_OscInitTypeDef RCC_OscInitStruct = {0};
 RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

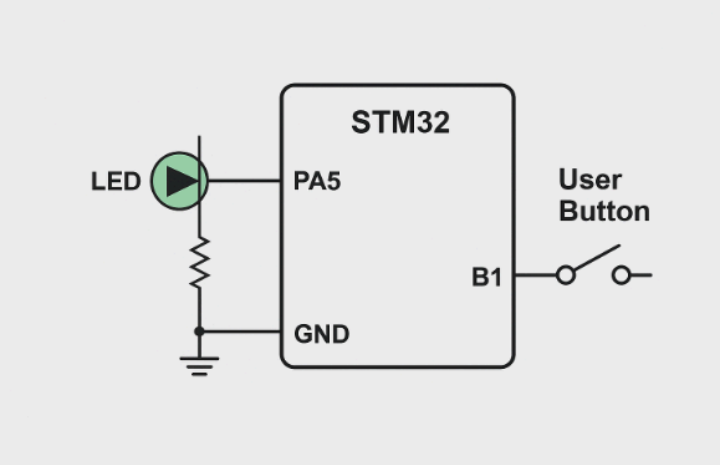
 __HAL_RCC_PWR_CLK_ENABLE();
 __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE3);

 RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
 RCC_OscInitStruct.HSIState = RCC_HSI_ON;
 RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
 RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
 RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
 RCC_OscInitStruct.PLL.PLLM = 16;
 RCC_OscInitStruct.PLL.PLLN = 336;
 RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;
 RCC_OscInitStruct.PLL.PLLQ = 2;
 RCC_OscInitStruct.PLL.PLLR = 2;
 if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
 {
   Error_Handler();
 }
 RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
 RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
 RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
 RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
 RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
 if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) !=
HAL_OK)
 {
   Error_Handler();
 }
}
```

7.1.9 GPIO Configuration

Η συνάρτηση `MX_GPIO_Init()` είναι υπεύθυνη για την αρχικοποίηση των GPIO θυρών του STM32. Ο σκοπός της είναι να ενεργοποιήσει τις απαραίτητες θύρες και να διαμορφώσει συγκεκριμένα pins ανάλογα με τις ανάγκες της εφαρμογής μας. Σε αυτό το πρόγραμμα:

- Ενεργοποιούνται οι θύρες GPIO.
- Ρυθμίζεται ένα pin ως είσοδος για κουμπί με δυνατότητα διακοπής.
- Ρυθμίζεται ένος pin ως έξοδος για LED.



Αναλυτικά:

- `__HAL_RCC_GPIOC_CLK_ENABLE();`: Ενεργοποιεί το ρολόι για τη θύρα GPIOC ώστε να μπορεί να χρησιμοποιηθεί.
- `__HAL_RCC_GPIOH_CLK_ENABLE()`: Ενεργοποιεί το ρολόι για τη θύρα GPIOH (ακόμα κι αν δεν χρησιμοποιείται άμεσα στο παράδειγμα, μπορεί να απαιτείται για system pins).
- `__HAL_RCC_GPIOA_CLK_ENABLE()`: Ενεργοποιεί το ρολόι για τη θύρα GPIOA, που είναι απαραίτητο για να χρησιμοποιήσουμε το pin PA5.
- `__HAL_RCC_GPIOB_CLK_ENABLE()`: Ενεργοποιεί το ρολόι για τη θύρα GPIOB, πιθανότατα για το pin που αντιστοιχεί στο κουμπί B1.
- `HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);`: Θέτει το pin PA5 σε λογικό 0 (LOW). Αν το PA5 είναι συνδεδεμένο σε LED, αυτό το LED

ξεκινά σβηστό.

- `GPIO_InitStruct.Pin = B1_Pin;`: Επιλογή του pin που χρησιμοποιείται για το κουμπί (πιθανότατα PC13 σε Nucleo boards).

- `GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;`: Το pin του κουμπιού ρυθμίζεται να προκαλεί διακοπή όταν εντοπιστεί πτώση τάσης (δηλαδή όταν πατηθεί το κουμπί και πέσει η τάση).

- `GPIO_InitStruct.Pull = GPIO_NOPULL;`: Δεν ενεργοποιούνται εσωτερικές pull-up/down αντιστάσεις. Αυτό σημαίνει ότι πρέπει να υπάρχει εξωτερική αντίσταση pull-up/down ή το κουμπί να είναι συνδεδεμένο με τέτοιο τρόπο ώστε να αποφεύγεται θόρυβος (floating input).

- `HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);`: Ολοκληρώνει τη διαμόρφωση του pin του κουμπιού.

- `GPIO_InitStruct.Pin = GPIO_PIN_5;`: Επιλογή του pin PA5, που συχνά χρησιμοποιείται για LED σε STM32 Nucleo.

- `GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;`: Ορίζεται ως push-pull έξοδος, δηλαδή μπορεί να παρέχει και ρεύμα (HIGH) και να τραβήξει προς GND (LOW).

- `GPIO_InitStruct.Pull = GPIO_NOPULL;`: Δεν χρησιμοποιείται pull-up/down στο συγκεκριμένο pin (δεν χρειάζεται σε έξοδο).

- `GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;`: Η ταχύτητα εναλλαγής του pin ορίζεται χαμηλή, που είναι αρκετή για LED και εξοικονομεί ενέργεια.

- `HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);`: Ενεργοποιεί τη ρύθμιση του PA5.

```
static void MX_GPIO_Init(void)
{
 GPIO_InitTypeDef GPIO_InitStruct = {0};

 __HAL_RCC_GPIOC_CLK_ENABLE();
 __HAL_RCC_GPIOH_CLK_ENABLE();
 __HAL_RCC_GPIOA_CLK_ENABLE();
```

```
  __HAL_RCC_GPIOB_CLK_ENABLE();

  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);
  /*Configure GPIO pin : PA5 */
  GPIO_InitStruct.Pin = GPIO_PIN_5;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
}
```

7.1.10 ADC configuration

Αυτό το τμήμα του κώδικα υλοποιεί τη ρύθμιση του ADC1 (Analog-to-Digital Converter), ώστε να μπορούμε να διαβάσουμε αναλογικά σήματα π.χ. από αισθητήρες ή ποτενσιόμετρα. Επιλέγεται ο ADC channel 4 και ενεργοποιείται η συνεχής μετατροπή, ώστε να παίρνουμε συνεχώς τιμές χωρίς εξωτερική ενεργοποίηση.

Αυτό το setup είναι ιδανικό για εφαρμογές που χρειάζονται συνεχή ανάγνωση αναλογικών τιμών με σταθερή διαμόρφωση.

Αναλυτικά:

- `hadc1.Instance = ADC1;`: Επιλογή της μονάδας ADC1 που θα ρυθμιστεί.
- `hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4;`: Ο χρονισμός του ADC προκύπτει διαιρώντας το ρολόι PCLK με το 4 – πιο αργό sampling αλλά μεγαλύτερη ακρίβεια.
- `hadc1.Init.Resolution = ADC_RESOLUTION_12B;`: Ανάλυση 12-bit: οι τιμές της μετατροπής θα είναι από 0 έως 4095.
- `hadc1.Init.ScanConvMode = DISABLE;`: Δεν θα γίνει σάρωση πολλών καναλιών — μόνο ένα κανάλι χρησιμοποιείται.
- `hadc1.Init.ContinuousConvMode = ENABLE;`: Ενεργοποιείται η συνεχής μετατροπή – μόλις τελειώνει μια, ξεκινά αυτόματα η επόμενη.

- `hadc1.Init.DiscontinuousConvMode = DISABLE;`: Δεν γίνεται διακεκομμένη μετατροπή – κάθε φορά ολοκληρώνεται πλήρης μετατροπή.

- `hadc1.Init.ExternalTrigConvEdge= ADC_EXTERNALTRIGCONVEDGE_NONE;`: Δεν χρησιμοποιείται εξωτερικός παλμός για έναρξη μετατροπής.

- `hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;`: Οι μετατροπές ξεκινούν με εντολή λογισμικού (π.χ. `HAL_ADC_Start()`).

- `hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;`: Τα δεδομένα μετατροπής στοιχίζονται δεξιά στη μεταβλητή (LSB στο τέλος).

- `hadc1.Init.NbrOfConversion = 1;`: Θα γίνει μόνο μία μετατροπή ανά κύκλο.

- `hadc1.Init.DMAContinuousRequests = DISABLE;`: Δεν χρησιμοποιείται DMA για μεταφορά δεδομένων (θα τα διαχειριστούμε με polling ή interrupts).

- `hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;`: Τέλος μετατροπής θεωρείται όταν ολοκληρωθεί μία μετατροπή.

- `sConfig.Channel = ADC_CHANNEL_4;`: Επιλογή του καναλιού 4 του ADC — π.χ. μπορεί να είναι συνδεδεμένο σε ποτενσιόμετρο ή αισθητήρα.

- `sConfig.Rank = 1;`: Ορίζει τη σειρά της μετατροπής – εδώ είναι η μόνη, άρα θέση 1.

- `sConfig.SamplingTime = ADC_SAMPLETIME_28CYCLES;`: Δείγμα λαμβάνεται κάθε 28 κύκλους ADC – tradeoff μεταξύ ακρίβειας και ταχύτητας.

- `HAL_ADC_Init(&hadc1);`: Κλήση της βιβλιοθήκης HAL για αρχικοποίηση του ADC με τις παραπάνω παραμέτρους.

- `HAL_ADC_ConfigChannel(&hadc1, &sConfig);`: Ρυθμίζει το κανάλι ADC4 σύμφωνα με τις ρυθμίσεις του sConfig.

```
static void MX_ADC1_Init(void)
{
 ADC_ChannelConfTypeDef sConfig = {0};
 hadc1.Instance = ADC1;
 hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4;
 hadc1.Init.Resolution = ADC_RESOLUTION_12B;
```

```
  hadc1.Init.ScanConvMode = DISABLE;
  hadc1.Init.ContinuousConvMode = ENABLE;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
  hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 1;
  hadc1.Init.DMAContinuousRequests = DISABLE;
  hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  if (HAL_ADC_Init(&hadc1) != HAL_OK)
  {
    Error_Handler();
  }
  sConfig.Channel = ADC_CHANNEL_4;
  sConfig.Rank = 1;
  sConfig.SamplingTime = ADC_SAMPLETIME_28CYCLES;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
}
```

7.1.11 USART 1 Configuration

Αυτό το block κώδικα ρυθμίζει το **USART1** ώστε να μπορεί να στέλνει και να λαμβάνει δεδομένα μέσω σειριακής επικοινωνίας (UART). Αυτή η λειτουργία είναι κρίσιμη για επικοινωνία με άλλες συσκευές, υπολογιστή ή microcontrollers, ειδικά σε embedded εφαρμογές όπου χρησιμοποιούμε UART για μετάδοση μετρήσεων, εντολών ή debugging.

Αναλυτικά:

- `huart1.Instance = USART1;`: Επιλέγουμε τη μονάδα USART1 για ρύθμιση.
- `huart1.Init.BaudRate = 115200;`: Καθορίζουμε τον ρυθμό μετάδοσης στα 115200 bits/sec.
- `huart1.Init.WordLength = UART_WORDLENGTH_8B;`: Ορίζεται μήκος δεδομένων στα 8 bits.
- `huart1.Init.StopBits = UART_STOPBITS_1;`: Επιλέγεται 1 stop bit, το οποίο είναι το πιο συνηθισμένο για απλή επικοινωνία.
- `huart1.Init.Parity = UART_PARITY_NONE;`: Χωρίς parity – δεν υπάρχει έλεγχος ισοτιμίας στα δεδομένα (απλούστερη επικοινωνία).

- `huart1.Init.Mode = UART_MODE_TX_RX;` : Η θύρα δουλεύει και για μετάδοση (TX) και για λήψη (RX).

- `huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;:` Δεν χρησιμοποιείται hardware flow control (RTS/CTS) — η ροή ελέγχεται λογισμικά ή δεν ελέγχεται.

- `huart1.Init.OverSampling = UART_OVERSAMPLING_16;:` Επιλέγεται υπερδειγματοληψία ×16, για καλύτερη ακρίβεια συγχρονισμού.

- `HAL_UART_Init(&huart1);:` Εφαρμόζει όλες τις παραπάνω ρυθμίσεις στον USART1 μέσω της HAL βιβλιοθήκης.

```
static void MX_USART1_UART_Init(void)
{
 huart1.Instance = USART1;
 huart1.Init.BaudRate = 115200;
 huart1.Init.WordLength = UART_WORDLENGTH_8B;
 huart1.Init.StopBits = UART_STOPBITS_1;
 huart1.Init.Parity = UART_PARITY_NONE;
 huart1.Init.Mode = UART_MODE_TX_RX;
 huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
 huart1.Init.OverSampling = UART_OVERSAMPLING_16;
 if (HAL_UART_Init(&huart1) != HAL_OK)
 {
   Error_Handler();
 }
}
```

7.1.12 USART 2 configuration

Μέσω του UART2 έχουμε την δυνατότητα να ρυθμίζουμε τη δεύτερη σειριακή θύρα του μικροελεγκτή, τη USART2, για επικοινωνία UART (όπως και η USART1). Η ύπαρξη δεύτερης UART μάς επιτρέπει να έχουμε πολλαπλά κανάλια επικοινωνίας, για παράδειγμα, το ένα για αποστολή δεδομένων σε εξωτερική συσκευή και το άλλο για debugging/logging στον υπολογιστή.

Αναλυτικά:

- `huart2.Instance = USART2;`: Επιλέγεται η μονάδα USART2 για να ρυθμιστεί ως σειριακή θύρα.
- `huart2.Init.BaudRate = 115200;`: Ορίζεται baud rate στα 115200 bps, ώστε να ταιριάζει με κοινά τερματικά και άλλες σειριακές συσκευές.
- `huart2.Init.WordLength = UART_WORDLENGTH_8B;`: Κάθε πακέτο δεδομένων αποτελείται από 8 bits — πρότυπο για UART επικοινωνία.
- `huart2.Init.StopBits = UART_STOPBITS_1;`: Χρησιμοποιείται 1 stop bit, όπως στις περισσότερες UART συνδέσεις.
- `huart2.Init.Parity = UART_PARITY_NONE;`: Δεν γίνεται χρήση parity bit, απλουστεύοντας την επικοινωνία.
- `huart2.Init.Mode = UART_MODE_TX_RX;`: Η θύρα δουλεύει σε αμφίδρομη λειτουργία: αποστολή και λήψη δεδομένων.
- `huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;`: Δεν χρησιμοποιείται hardware flow control — μόνο οι γραμμές TX/RX είναι απαραίτητες.
- `huart2.Init.OverSampling= UART_OVERSAMPLING_16;`: Χρησιμοποιείται oversampling 16x για σταθερότητα και ακρίβεια στην ανάγνωση.
- `HAL_UART_Init(&huart2);`: Ενεργοποιεί την USART2 με τις παραπάνω ρυθμίσεις μέσω της HAL.

```
static void MX_USART2_UART_Init(void)
{
 huart2.Instance = USART2;
 huart2.Init.BaudRate = 115200;
 huart2.Init.WordLength = UART_WORDLENGTH_8B;
 huart2.Init.StopBits = UART_STOPBITS_1;
 huart2.Init.Parity = UART_PARITY_NONE;
 huart2.Init.Mode = UART_MODE_TX_RX;
 huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
 huart2.Init.OverSampling = UART_OVERSAMPLING_16;
 if (HAL_UART_Init(&huart2) != HAL_OK)
 {
   Error_Handler();
 }
```

```
}
```

7.1.13 While True

Σε αυτό το κομμάτι κώδικα, υλοποιείται ο βασικός βρόχος λειτουργίας της εφαρμογής. Η κύρια λειτουργία είναι η ανάγνωση δεδομένων από τον ADC και η αποστολή αυτών των δεδομένων μέσω της σειριακής θύρας UART1. Στόχος είναι η συνεχή μέτρηση και μετάδοση της ψηφιακής τιμής που λαμβάνεται από το ADC. Ο κώδικας υλοποιεί συνεχή δειγματοληψία ADC σε blocking mode χωρίς χρήση interrupt ή DMA. Εξασφαλίζει ότι τα δεδομένα ADC διαβάζονται μόνο όταν είναι διαθέσιμα (polling). Τα δεδομένα κωδικοποιούνται σε δύο bytes ώστε να μεταδοθούν σωστά μέσω UART, που είναι byte-stream πρωτόκολλο.

- `if (HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY) == HAL_OK)`: Polling για ολοκλήρωση μετατροπής ADC.
    1. Η συνάρτηση μπλοκάρει μέχρι να ολοκληρωθεί μια μετατροπή ADC (με μέγιστο χρόνο αναμονής άπειρο - HAL_MAX_DELAY).
    2. Επιστρέφει HAL_OK αν η μέτρηση είναι έτοιμη.
    3. Εξασφαλίζει ότι διαβάζουμε μόνο έγκυρα και ολοκληρωμένα δεδομένα ADC.

- `uint16_t adcValue = HAL_ADC_GetValue(&hadc1);`: Ανάγνωση της ψηφιακής τιμής από τον ADC
    1. Η τιμή είναι 12-bit (όπως ρυθμίστηκε), αποθηκευμένη σε 16-bit μεταβλητή (uint16_t).
    2. Η τιμή αντιπροσωπεύει την αναλογική τάση που μετρήθηκε, κλιμακωμένη ανάλογα με την τάση αναφοράς και το resolution.

- ```
  uint8_t data[2];

  data[0] = (adcValue >> 8) & 0xFF;  // MSB

  data[1] = adcValue & 0xFF;         // LSB
  ```

  Μετατροπή της 16-bit τιμής ADC σε δύο bytes για μετάδοση:

    1. Χρησιμοποιείται μορφή big-endian: πρώτα το πιο σημαντικό byte (MSB), μετά το λιγότερο σημαντικό (LSB).
        a. data[0] παίρνει τα 8 υψηλότερα bits (bits 15-8).
        b. data[1] παίρνει τα 8 χαμηλότερα bits (bits 7-0).

Αυτό επιτρέπει στα δεδομένα να σταλούν με σταθερή και καθορισμένη δομή.
  ```

- `HAL_UART_Transmit(&huart1, data, 2, HAL_MAX_DELAY);`: Αποστολή των 2 bytes μέσω UART1:
    1. Μεταδίδουμε τον πίνακα data μήκους 2 bytes.
    2. Χρησιμοποιείται blocking mode (μέγιστος χρόνος αναμονής άπειρος) για να εξασφαλιστεί ότι τα bytes μεταδόθηκαν πλήρως πριν συνεχιστεί ο βρόχος.
    3. Αυτή η μετάδοση επιτρέπει σε μια άλλη συσκευή να λαμβάνει συνεχώς τις αναλογικές μετρήσεις σε ψηφιακή μορφή.

```
while (1)
{
  if (HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY) == HAL_OK)
  {
    uint16_t adcValue = HAL_ADC_GetValue(&hadc1);
    // Μετατροπή σε 2 bytes (big-endian: MSB πρώτα)
    uint8_t data[2];
    data[0] = (adcValue >> 8) & 0xFF;  // MSB
    data[1] = adcValue & 0xFF;         // LSB
    // Στείλε 2 bytes(bitstream)
    HAL_UART_Transmit(&huart1, data, 2, HAL_MAX_DELAY);
  }
}
```

7.2 ESP32 DEVKIT 1 (30 pins)

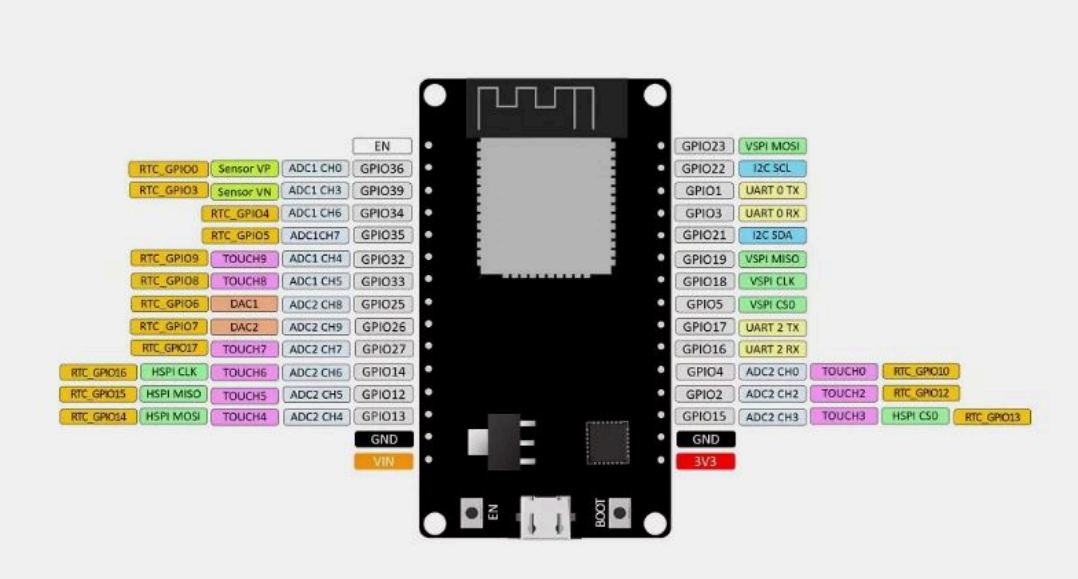
Ο ESP32 είναι ένας από τους πιο δημοφιλείς μικροελεγκτές (microcontrollers) που χρησιμοποιούνται σήμερα στον κόσμο της ενσωματωμένης τεχνολογίας και του Διαδικτύου των Πραγμάτων (Internet of Things - IoT). Ανήκει στην οικογένεια των προϊόντων της Espressif Systems και προσφέρει μια εξαιρετικά ευέλικτη και ισχυρή πλατφόρμα που συνδυάζει ασύρματη συνδεσιμότητα, υψηλές επιδόσεις και χαμηλή κατανάλωση ενέργειας.
Ο ESP32 βασίζεται σε έναν διπύρηνο μικροελεγκτή Tensilica Xtensa LX6, ο οποίος μπορεί να λειτουργήσει σε συχνότητες μέχρι και 240 MHz. Αυτό του δίνει τη δυνατότητα να εκτελεί ταυτόχρονα πολλαπλές διεργασίες και να υποστηρίζει πολύπλοκες εφαρμογές.
Η ενσωμάτωση Wi-Fi (802.11 b/g/n) και Bluetooth (κλασσικό και BLE - Bluetooth Low Energy) καθιστά τον ESP32 ιδανικό για συσκευές που απαιτούν ασύρματη επικοινωνία. Επιπλέον, ο ESP32 περιλαμβάνει πληθώρα περιφερειακών όπως ADCs (αναλογικοί-ψηφιακοί μετατροπείς), DACs, UART, SPI, I2C, PWM, touch sensors και άλλους, που διευκολύνουν την σύνδεση με αισθητήρες και εκτελεστές.

εικόνα 4.1: ESP32 DEVKIT 1

7.2.1 Τεχνικά Χαρακτηριστικά ESP32

- Μικροεπεξεργαστής: Dual-core Tensilica Xtensa LX6, έως 240 MHz
- Μνήμη:
    a. RAM: 520 KB SRAM
    b. Flash: Συνήθως 4 MB (ανάλογα με την έκδοση)
- Ασύρματη Συνδεσιμότητα:
    a. Wi-Fi 802.11 b/g/n
    b. Bluetooth v4.2 BR/EDR και BLE
- Πρωτόκολλα δικτύου: TCP/IP stack, HTTP, MQTT κλπ.
- I/O Περιφερειακά:
    a. GPIO pins: Περίπου 34 διαθέσιμα
    b. ADC: 12-bit, έως 18 κανάλια
    c. DAC: 2 κανάλια, 8-bit
    d. UART: 3 κανάλια
    e. SPI: 4 κανάλια
    f. I2C: 2 κανάλια
    g. PWM: πολλαπλά κανάλια
    h. Capacitive touch sensors: έως 10 σημεία
- Χαμηλή Κατανάλωση Ενέργειας:
    a. Διαφορετικά sleep modes (deep sleep, light sleep)
    b. Wake-up μέσω εξωτερικών ερεθισμάτων ή χρονικού διαστήματος
- Ασφάλεια:
    a. Secure boot
    b. Flash encryption
    c. Hardware cryptographic accelerators (AES, SHA-2, RSA, ECC)
- Τάση Λειτουργίας: 2.2 V έως 3.6 V
- Διαστάσεις: Διαφέρουν ανάλογα με το module/board

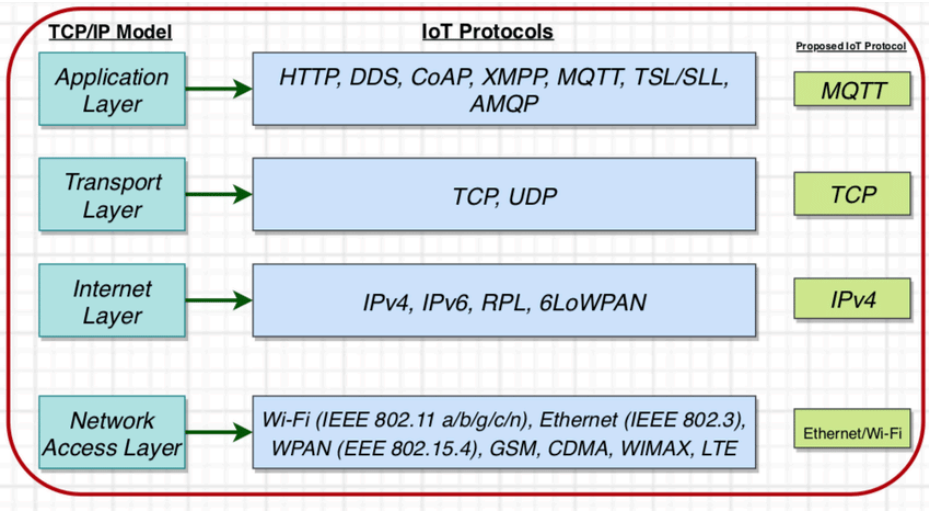
- Προγραμματισμός: ESP-IDF, Arduino IDE, MicroPython, Lua, κλπ.
- Επιπλέον: RTC (Real Time Clock), Hall sensor, temperature sensor



εικόνα 4.2: ESP32 pinout

7.2.2 Το πρωτόκολλο TCP/IP

Η αρχιτεκτονική TCP/IP (Transmission Control Protocol/Internet Protocol) αποτελεί τη θεμελιώδη βάση της λειτουργίας του Διαδικτύου και των περισσότερων σύγχρονων δικτύων επικοινωνίας. Πρόκειται για ένα σύνολο πρωτοκόλλων που καθορίζουν πώς τα δεδομένα μεταφέρονται ανάμεσα σε υπολογιστές και άλλες συσκευές σε ένα δίκτυο.



εικόνα 4.3: Network Layers

1. Ιστορική Εξέλιξη

Η TCP/IP αναπτύχθηκε τη δεκαετία του 1970 από το αμερικανικό Υπουργείο Άμυνας, με σκοπό να δημιουργήσει ένα ανθεκτικό και ευέλικτο δίκτυο που να μπορεί να λειτουργεί ακόμη και αν κάποιοι κόμβοι πάψουν να ανταποκρίνονται. Η αρχιτεκτονική αυτή αποτέλεσε τη βάση για το Διαδίκτυο όπως το γνωρίζουμε σήμερα.

2. Λειτουργία και δομή

Η αρχιτεκτονική TCP/IP βασίζεται σε ένα μοντέλο τεσσάρων επιπέδων:

- **Επίπεδο Δικτύου (Network Interface Layer):** Περιλαμβάνει τα πρωτόκολλα που αφορούν την φυσική μεταφορά των δεδομένων στο δίκτυο, όπως Ethernet ή Wi-Fi.

- **Επίπεδο Διαδικτύου (Internet Layer):** Το βασικό πρωτόκολλο εδώ είναι το IP (Internet Protocol), που έχει ως ρόλο τη δρομολόγηση των πακέτων δεδομένων από τον αποστολέα στον παραλήπτη μέσα από πολλαπλά δίκτυα.

- **Επίπεδο Μεταφοράς (Transport Layer):** Εδώ συναντάμε τα πρωτόκολλα TCP και UDP. Το TCP παρέχει αξιόπιστη, συνδεδεμένη μεταφορά δεδομένων, εξασφαλίζοντας ότι τα πακέτα φτάνουν σωστά και με τη σωστή σειρά, ενώ το UDP παρέχει μια πιο γρήγορη αλλά μη αξιόπιστη μεταφορά, κατάλληλη για εφαρμογές που δεν απαιτούν επιβεβαίωση.

- **Επίπεδο Εφαρμογής (Application Layer):** Περιλαμβάνει πρωτόκολλα που υποστηρίζουν εφαρμογές όπως το HTTP για το Web, το FTP για τη μεταφορά αρχείων, το SMTP για το ηλεκτρονικό ταχυδρομείο κ.ά.

3. Ρόλος του IP

Το IP είναι υπεύθυνο για την διεύθυνση και τη δρομολόγηση των πακέτων δεδομένων. Κάθε συσκευή στο διαδίκτυο έχει μια μοναδική διεύθυνση IP, που λειτουργεί σαν μια ταχυδρομική διεύθυνση για τα πακέτα. Υπάρχουν δύο εκδόσεις: IPv4, η πιο διαδεδομένη, και IPv6, που έχει σχεδιαστεί για να καλύψει τις ανάγκες των σύγχρονων δικτύων με περισσότερες διευθύνσεις.

4. Ρόλος του TCP

Το TCP διαχειρίζεται την αξιόπιστη μεταφορά δεδομένων, δημιουργώντας μια σύνδεση μεταξύ αποστολέα και παραλήπτη πριν από τη μετάδοση. Διασφαλίζει ότι τα πακέτα φτάνουν σωστά, ζητώντας εκ νέου αποστολή αν χρειαστεί, και τα τοποθετεί στη σωστή σειρά. Χρησιμοποιείται σε εφαρμογές όπου η ακεραιότητα των δεδομένων είναι κρίσιμη, όπως το web browsing και το email.

5. Εφαρμογές της αρχιτεκτονικής TCP/IP

Η TCP/IP είναι η βάση του σύγχρονου Διαδικτύου και υποστηρίζει σχεδόν όλες τις διαδικτυακές υπηρεσίες. Επιτρέπει τη διασύνδεση μεταξύ δισεκατομμυρίων συσκευών παγκοσμίως, προσφέροντας ευελιξία και επεκτασιμότητα. Χρησιμοποιείται επίσης σε ιδιωτικά δίκτυα (Intranets) και σε συστήματα επικοινωνίας μεταξύ μηχανών (IoT).

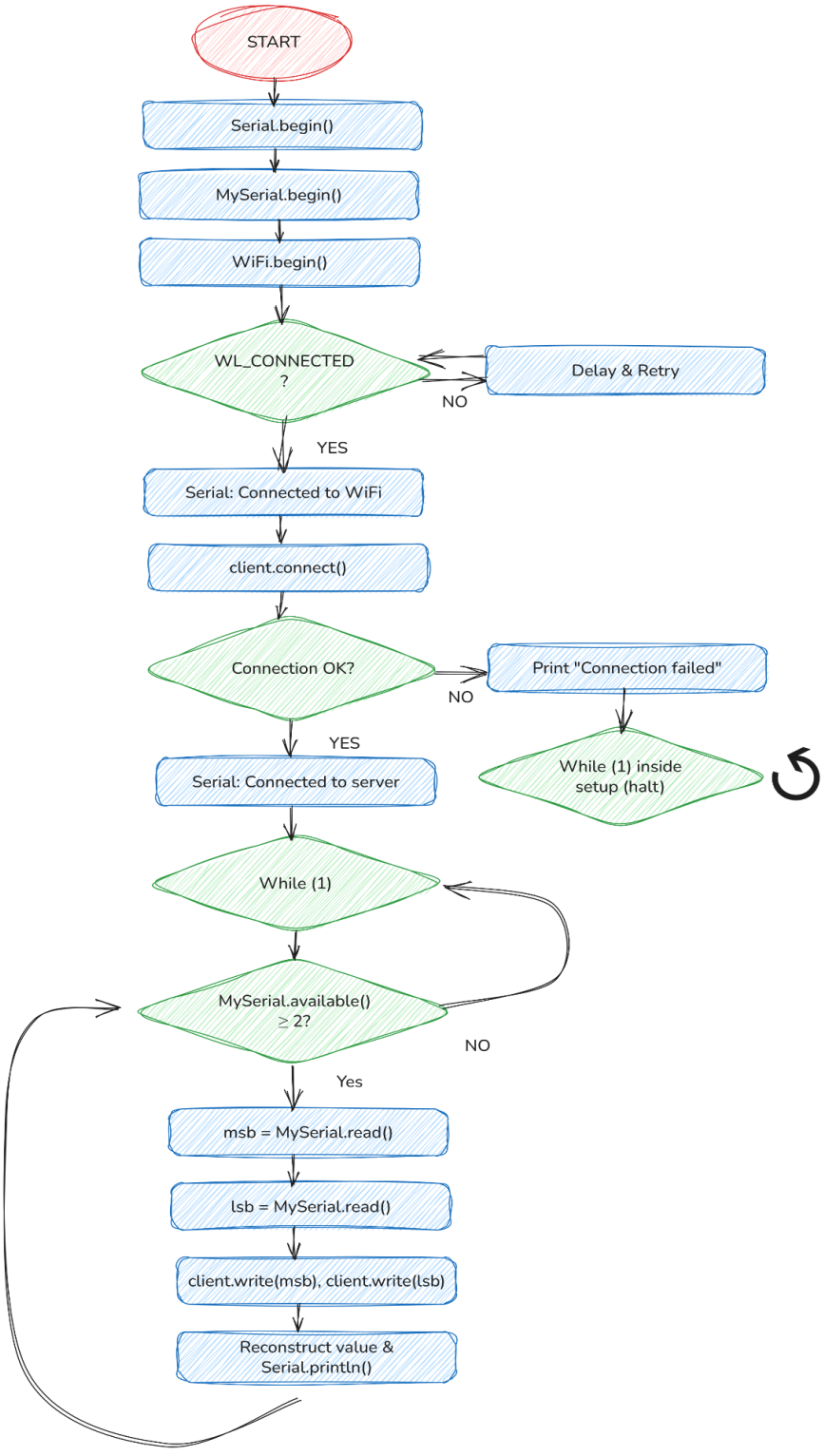
6. Πλεονεκτήματα και Προκλήσεις

Το μοντέλο TCP/IP είναι ανοικτό, ευέλικτο και επιτρέπει την ενσωμάτωση νέων τεχνολογιών χωρίς να απαιτείται αλλαγή στην υποδομή. Ωστόσο, η ασφάλεια δεν ήταν αρχικά σχεδιασμένη στο πρωτόκολλο, με αποτέλεσμα να έχουν προκύψει προβλήματα όπως επιθέσεις και παραβιάσεις δεδομένων. Για αυτό το λόγο έχουν αναπτυχθεί πρόσθετα πρωτόκολλα ασφάλειας όπως το TLS/SSL.

7.2.3 Μετάδοση δεδομένων από UART σε TCP Server μέσω Wi-Fi (ESP32)

Ο συγκεκριμένος κώδικας προορίζεται για το μικροελεγκτή ESP32 και υλοποιεί μια γέφυρα επικοινωνίας μεταξύ μιας σειριακής συσκευής και ενός απομακρυσμένου TCP server μέσω Wi-Fi. Πιο συγκεκριμένα, ο ESP32 συνδέεται σε ένα Wi-Fi δίκτυο με τα στοιχεία που έχουν δοθεί, και έπειτα προσπαθεί να ανοίξει μια TCP σύνδεση σε μια προκαθορισμένη διεύθυνση IP και θύρα.

7.2.4 Διάγραμμα ροής ESP32-DEVKIT 1 (30 pins) firmware



εικόνα 4.4: διάγραμμα ροής ESP32 firmware

Για την επικοινωνία με εξωτερική συσκευή χρησιμοποιείται η δεύτερη σειριακή θύρα του ESP32 (UART2), η οποία έχει οριστεί να λαμβάνει δεδομένα από το pin 16 (RX) και να στέλνει δεδομένα στο pin 17 (TX) με ρυθμό 115200 baud. Στην κύρια λούπα του προγράμματος, ο ESP32 παρακολουθεί συνεχώς αν υπάρχουν διαθέσιμα τουλάχιστον δύο bytes στη σειριακή θύρα. Όταν αυτά υπάρχουν, τα διαβάζει, τα συνδυάζει σε μία 16-bit τιμή για να τα εμφανίσει ως αριθμό σε μορφή κατάλληλη για debugging, και τα αποστέλλει αμέσως στον απομακρυσμένο server μέσω της TCP σύνδεσης.

Με αυτό τον τρόπο, ο ESP32 λειτουργεί σαν ενδιάμεσος που λαμβάνει δεδομένα από μια εξωτερική συσκευή μέσω UART και τα προωθεί σε πραγματικό χρόνο σε έναν απομακρυσμένο server, όπου μπορούν να επεξεργαστούν ή να αποθηκευτούν. Αυτή η προσέγγιση είναι ιδιαίτερα χρήσιμη για εφαρμογές όπου απαιτείται απομακρυσμένη παρακολούθηση ή συλλογή δεδομένων.

Είναι σημαντικό να σημειωθεί ότι αν δεν καταφέρει ο ESP32 να συνδεθεί είτε στο Wi-Fi είτε στον TCP server, το πρόγραμμα "κολλάει" και σταματά τη λειτουργία του. Αυτό σημαίνει ότι για πιο αξιόπιστες εφαρμογές, θα ήταν χρήσιμο να προστεθούν μηχανισμοί επανεκκίνησης ή επανάληψης σύνδεσης. Επίσης, το πρόγραμμα δεν ελέγχει αν η σύνδεση χάθηκε κατά τη διάρκεια της λειτουργίας, κάτι που θα μπορούσε να βελτιωθεί ώστε να εξασφαλιστεί συνεχής και σταθερή μετάδοση δεδομένων.

Τέλος, η ανάγνωση των δεδομένων γίνεται πάντοτε σε ζεύγη των δύο bytes, πράγμα που σημαίνει ότι το πρωτόκολλο επικοινωνίας με τη συσκευή πρέπει να είναι συμβατό με αυτή τη λογική ώστε να αποφεύγονται σφάλματα ή ασυμφωνίες στα δεδομένα.

- `#include <WiFi.h>`: Υποστηρίζει σύνδεση ESP32 σε Wi-Fi δίκτυο.
- `#include <HardwareSerial.h>`: Επιτρέπει χρήση επιπλέον UART θυρών πέραν της βασικής σειριακής.
- `const char* ssid =` Όνομα του Wi-Fi δικτύου στο οποίο θα συνδεθεί το ESP32.
- `const char* password =` Κωδικός πρόσβασης για το Wi-Fi δίκτυο.
- `const char* serverIP = "192.168.100.244";:` Διεύθυνση IP του απομακρυσμένου TCP server.
- `const uint16_t serverPort = 12345;:` Θύρα επικοινωνίας του TCP server.
- `WiFiClient client;:` Αντικείμενο για τη διαχείριση TCP σύνδεσης μέσω Wi-Fi.
- `HardwareSerial MySerial(2);:` Δημιουργία αντικειμένου για UART2 του ESP32 (δευτερεύουσα σειριακή θύρα).
- `MySerial.begin(115200, SERIAL_8N1, 16, 17);:` Αρχικοποίηση UART2 με baud rate 115200, RX σε pin 16, TX σε pin 17.
- `WiFi.begin(ssid, password);:` Εκκίνηση σύνδεσης σε Wi-Fi δίκτυο.
- `while (WiFi.status() != WL_CONNECTED):` Αναμονή μέχρι να ολοκληρωθεί η σύνδεση στο Wi-Fi.
- `client.connect(serverIP, serverPort):` Απόπειρα σύνδεσης στον TCP server.

- `while (true);`: μετά από αποτυχία σύνδεσης: Σταθερό κόλλημα για να μη συνεχίσει το πρόγραμμα αν δεν συνδεθεί.
- `while (MySerial.available() >= 2)`: Έλεγχος αν υπάρχουν τουλάχιστον 2 bytes διαθέσιμα στη UART2.
- `uint8_t msb = MySerial.read();` και `uint8_t lsb = MySerial.read();`: Ανάγνωση δύο bytes (most και least significant byte).
- `client.write(msb); client.write(lsb);`: Αποστολή των δύο bytes στον TCP server.
- `uint16_t value = (msb << 8) | lsb;`: Συνδυασμός των δύο bytes σε μία 16-bit ακέραια τιμή.
- `Serial.println(value);`: Εκτύπωση της 16-bit τιμής για debugging στο σειριακό monitor.

7.3 Λήψη Επεξεργασία και Απεικόνιση των δεδομένων

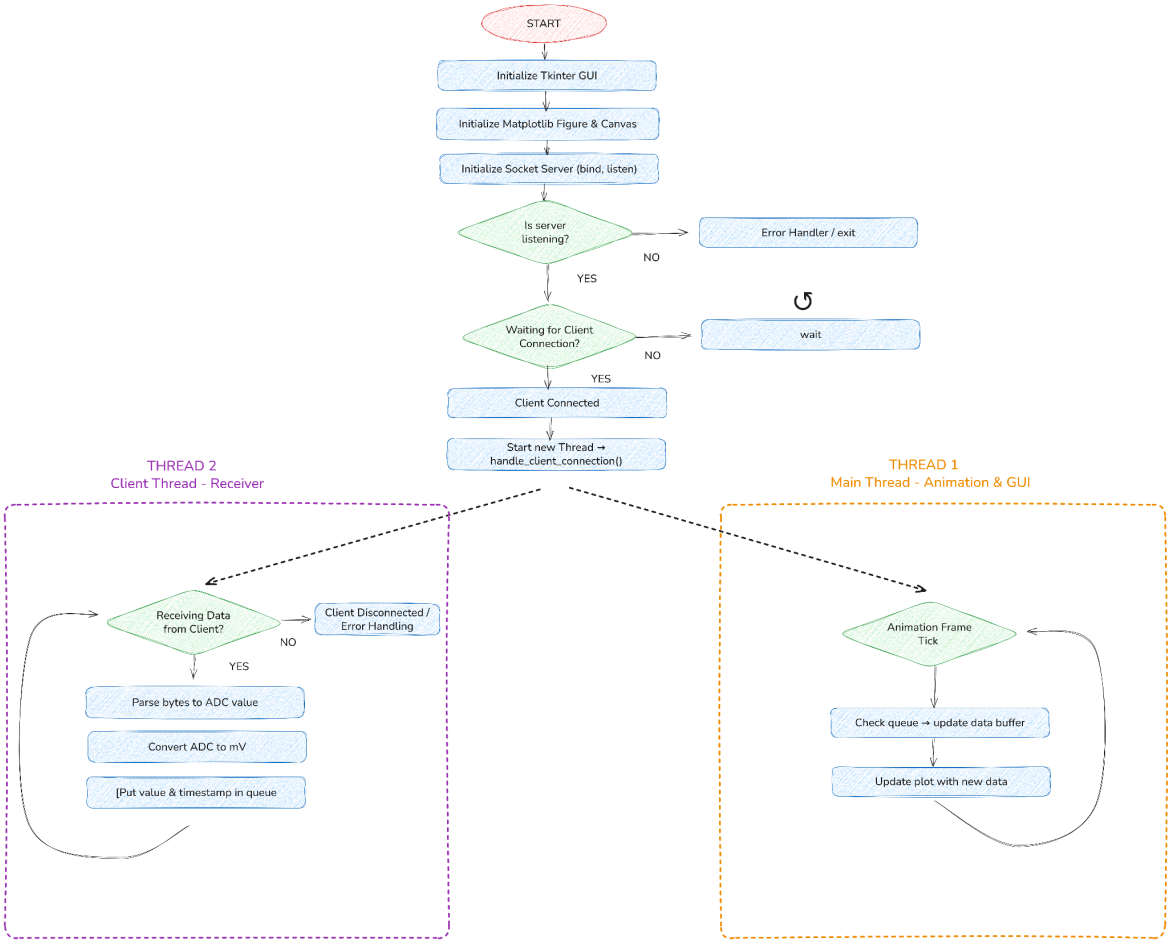
7.3.1 overview

Το συγκεκριμένο πρόγραμμα υλοποιεί μια εφαρμογή σε Python που λαμβάνει δεδομένα από τον ESP32 μέσω TCP/IP socket, τα μετατρέπει σε φυσικές τιμές τάσης (mV) και τα εμφανίζει ζωντανά σε ενα Graphic User Interface απεικονίζοντας εν τέλει την τελική επιθυμητή ECG κυματομορφή.

Χρησιμοποιεί:

- **Tkinter** για το γραφικό περιβάλλον.
- **Matplotlib** για τη σχεδίαση της κυματομορφής ECG.
- **Socket Server** TCP για λήψη δεδομένων.
- **multithreading** για ταυτόχρονη εκτέλεση server και GUI.
- **queue.Queue** για ασφαλή διαμοιρασμό δεδομένων μεταξύ των threads.

7.3.2 Διάγραμμα ροής προγράμματος λήψης και απεικόνισης



εικόνα 4.5: διάγραμμα ροής προγράμματος λήψης και απεικόνισης

7.3.3 Imports

Αυτό το τμήμα του κώδικα περιλαμβάνει τις απαραίτητες βιβλιοθήκες για τη λειτουργία της εφαρμογής. Καθεμία παίζει διακριτό ρόλο, από τη δημιουργία γραφικού περιβάλλοντος μέχρι την επεξεργασία και εμφάνιση δεδομένων, και την υλοποίηση πολυνηματικότητας.

Οι βιβλιοθήκες διαχωρίζονται σε:

- Ενσωματωμένες Python βιβλιοθήκες (`tkinter, socket, threading, queue, time`)

- Εξωτερικές βιβλιοθήκες (`matplotlib, matplotlib.animation`)

7.3.4 Σταθερές & Αρχικοποίηση GUI (Constants & GUI Init)

- `host = '0.0.0.0'`: Ο server ακούει σε όλες τις διαθέσιμες IP διευθύνσεις του συστήματος (π.χ. localhost, LAN, WiFi).
- `port = 12345`: Ορίζεται η θύρα TCP στην οποία θα περιμένει συνδέσεις ο server.
- `ADC_TO_MV = 3.3 / 4095 * 1000`: Ορίζεται ο συντελεστής μετατροπής από τιμή ADC (12-bit) σε millivolt.
- `data_queue = queue.Queue()`: Ουρά FIFO για ασφαλή μεταφορά δεδομένων από το thread λήψης στο GUI.
- `data_buffer = []`: Λίστα που κρατά τις τελευταίες δειγματοληπτικές τιμές που θα εμφανίζονται στο γράφημα.

7.5 GUI (Tkinter & Matplotlib) – Δημιουργία παραθύρου και εμφάνιση περιεχομένου

Το GUI αποτελεί το γραφικό περιβάλλον της εφαρμογής και υλοποιείται με χρήση των βιβλιοθηκών Tkinter και Matplotlib. Ο ρόλος του είναι να εμφανίζει με φιλικό τρόπο την τρέχουσα κατάσταση του συστήματος (π.χ. συνδεδεμένος ή όχι), να παρουσιάζει τις μετρήσεις ADC και mV που λαμβάνονται από τον client και να προβάλλει ζωντανά την κυματομορφή του ECG σήματος.

Η σχεδίαση διαχωρίζεται σε τρία βασικά κομμάτια:

- Δημιουργία παραθύρου και τίτλου.
- Ετικέτες και πλαίσιο για εμφάνιση δεδομένων.
- Ενσωμάτωση και παραμετροποίηση γραφήματος με χρήση matplotlib.

Αναλυτικά:

- `root = tk.Tk();`: Δημιουργεί το βασικό παράθυρο της εφαρμογής με χρήση της βιβλιοθήκης Tkinter.

- `root.title("Live ECG Visualization");`: Ορίζει τον τίτλο του παραθύρου ώστε να εμφανίζεται "Live ECG Visualization" στην κορυφή.

- `root.geometry("900x600");`: Καθορίζει τις διαστάσεις του παραθύρου σε pixels (900 πλάτος x 600 ύψος).

- `status_label = tk.Label(...);`: Δημιουργεί ετικέτα (label) που εμφανίζει την τρέχουσα κατάσταση του συστήματος (π.χ., "Waiting for data...").

- `status_label.pack(pady=5, fill=tk.X);`: Τοποθετεί την ετικέτα στο πάνω μέρος του παραθύρου με εσωτερικό περιθώριο 5 pixels και πλήρες πλάτος.

- `message_text = tk.Text(...);`: Δημιουργεί πλαίσιο για εμφάνιση εισερχόμενων τιμών ADC και mV.

- `message_text.pack(pady=5);`: Τοποθετεί το text box κάτω από την ετικέτα με περιθώριο 5 pixels.

- `message_text.config(state=tk.DISABLED);`: Κάνει το πλαίσιο κειμένου μη επεξεργάσιμο από τον χρήστη (read-only), ώστε να λειτουργεί μόνο για εμφάνιση μηνυμάτων.

- `fig, ax = plt.subplots(figsize=(9, 4));`: Δημιουργεί αντικείμενο γραφήματος (figure) και άξονες (ax) για την απεικόνιση του ECG με χρήση matplotlib.

- `ax.set_title("Live ECG Data");`: Ορίζει τον τίτλο του γραφήματος.

- `ax.set_xlabel("Time (s)");`: Ορίζει την ετικέτα του άξονα X ως "Time (s)".

- `ax.set_ylabel("Amplitude (mV)");`: Ορίζει την ετικέτα του άξονα Y ως "Amplitude (mV)".

- `line, = ax.plot([], [], lw=2, color='red');`: Δημιουργεί αρχική καμπύλη (γραμμή) για να σχεδιάζεται το ECG σήμα με κόκκινο χρώμα και πάχος γραμμής 2.

- `ax.grid(True);`: Ενεργοποιεί το grid (πλέγμα) στο φόντο του γραφήματος για καλύτερη ανάγνωση.

- `ax.set_xlim(0, 1);`: Θέτει την αρχική χρονική περιοχή του άξονα X από 0 έως 1 δευτερόλεπτο.

- `ax.set_ylim(0, 3300);`: Θέτει το εύρος της τάσης από 0 έως 3300 mV — δηλαδή το πλήρες εύρος του 12-bit ADC.

- `canvas = FigureCanvasTkAgg(fig, master=root);`: Ενσωματώνει το matplotlib γράφημα μέσα στο παράθυρο Tkinter ως widget.

- `canvas.get_tk_widget().pack();`: Τοποθετεί το γράφημα μέσα στο GUI ώστε να εμφανίζεται στο κάτω μέρος του παραθύρου.

7.5.1 Plot Initialization – Ρύθμιση & ανανέωση του διαγράμματος ECG σε πραγματικό χρόνο

Σε αυτό το μέρος του script, ρυθμίζεται το animation του διαγράμματος ECG με χρήση της συνάρτησης `FuncAnimation` της matplotlib. Η ανανέωση του γραφήματος γίνεται κάθε λίγα milliseconds και εμφανίζει τα πιο πρόσφατα σημεία που έχουν ληφθεί μέσω της Queue από τα threads που διαχειρίζονται τα εισερχόμενα δεδομένα. Ουσιαστικά, εδώ «συνδέεται» το γράφημα με τη ροή των δεδομένων, και κάθε νέα μέτρηση (σε mV) προστίθεται δυναμικά στο γράφημα του ECG.

Αναλυτικά:

- `data_buffer = [];`: Δημιουργεί έναν προσωρινό buffer (λίστα) που κρατά τις πρόσφατες μετρήσεις ώστε να σχεδιάζεται μόνο ένα "παράθυρο" δεδομένων κάθε φορά (π.χ., τα τελευταία 300 σημεία).

- `def animate(i):`: Ορίζεται η βασική συνάρτηση animation η οποία καλείται περιοδικά και ανανεώνει το γράφημα.

- `while not data_queue.empty():`: Ελέγχει αν υπάρχουν νέα δεδομένα στην ουρά.

- `value, timestamp = data_queue.get();`: Παίρνει το επόμενο διαθέσιμο δείγμα και τη χρονική του σφραγίδα από την Queue (παραγόμενο από το thread του client).

- `data_buffer.append((value, timestamp));`: Προσθέτει το νέο δείγμα στον buffer.

- `if len(data_buffer) > 300: data_buffer.pop(0);`: Διατηρεί σταθερό μήκος buffer (300 σημεία), πετώντας το παλαιότερο όταν γεμίζει.

- `if not data_buffer: return line,`: Αν δεν υπάρχουν καθόλου δεδομένα ακόμη, δεν επιχειρεί να σχεδιάσει τίποτα.

- `values = [v for v, _ in data_buffer];`: Δημιουργεί λίστα μόνο με τις τιμές σε mV.

- `timestamps = [ts - data_buffer[0][1] for _, ts in data_buffer];`: Υπολογίζει σχετική χρονική βάση για κάθε σημείο (ξεκινά από 0).

- `line.set_data(timestamps, values);`: Ενημερώνει την καμπύλη του διαγράμματος με τα νέα δεδομένα.

- `ax.set_xlim(0, timestamps[-1] if timestamps else 1);`: Προσαρμόζει δυναμικά τον οριζόντιο άξονα (χρόνος), ώστε να μεγαλώνει όσο χρειάζεται.

- `ax.set_ylim(0, 3300);`: Ο κάθετος άξονας μένει σταθερός από 0 έως 3300 mV, δηλαδή το πλήρες εύρος του ADC.

- `return line,`: Επιστρέφει την καμπύλη που θα επανασχεδιαστεί.

- `ani = animation.FuncAnimation(..., interval=30);`: Δημιουργεί το animation που καλεί τη `animate()` κάθε 30 milliseconds — δηλαδή περίπου 33 FPS, επαρκές για ομαλή απεικόνιση ECG.

7.5.2 Update Message Display – Ενημέρωση του text widget με νέα μηνύματα

Αυτό το κομμάτι αφορά την εμφάνιση στο GUI, όπου δείχνουμε στον χρήστη τις τρέχουσες πληροφορίες, όπως τις τιμές ADC, τα mV, μηνύματα σύνδεσης ή σφάλματα.

Αναλυτικά:

- `def update_message_display(message):` Δημιουργείται συνάρτηση που παίρνει ως είσοδο μια συμβολοσειρά (`message`) και την προσθέτει στο text box.

- `message_text.config(state=tk.NORMAL)`: Κάνει το widget επεξεργάσιμο προσωρινά ώστε να μπορούμε να εισάγουμε κείμενο.

- `message_text.insert(tk.END, message + "\n")`: Προσθέτει το νέο μήνυμα στην τελευταία γραμμή (στο τέλος).

- `message_text.yview(tk.END)`: Αυτόματα "κατεβάζει" το scrollbar ώστε να φαίνεται η πιο πρόσφατη γραμμή, σαν αυτόματη κύλιση προς τα κάτω.

- `message_text.config(state=tk.DISABLED)`: Κάνει ξανά το widget μη επεξεργάσιμο ώστε ο χρήστης να μην μπορεί να αλλάξει το κείμενο (read-only).

7.5.3 handle client connection – Διαχείριση σύνδεσης και επεξεργασία δεδομένων

Αυτή η συνάρτηση αναλαμβάνει να λάβει τα δεδομένα που στέλνει ο πελάτης (client) μέσω TCP socket, να τα μετατρέψει σε χρήσιμες μετρήσεις (ADC → mV) και να τα στείλει στην ουρά για απεικόνιση, ενώ ταυτόχρονα ενημερώνει το GUI με μηνύματα.

Αναλυτικά:

- `buffer = b''`: Δημιουργεί έναν προσωρινό buffer (byte string) για τη συσσώρευση των δεδομένων που λαμβάνονται τμηματικά.

- `client_socket.settimeout(5.0)`: Ορίζει timeout 5 δευτερολέπτων για το socket, ώστε αν δεν ληφθούν δεδομένα στο διάστημα αυτό να διακοπεί η σύνδεση.

- `while True:`: Βασικός βρόχος λήψης δεδομένων μέχρι να διακοπεί η σύνδεση ή να γίνει σφάλμα.

- `data = client_socket.recv(1024)`: Λαμβάνει μέχρι 1024 bytes από τον πελάτη.

- `if not data:`: Αν δεν έρθουν δεδομένα, σημαίνει ότι ο πελάτης αποσυνδέθηκε.

- `buffer += data`: Προσθέτει τα νέα bytes στον buffer.

- `while len(buffer) >= 2:` Επεξεργάζεται τα δεδομένα σε ζεύγη 2 bytes (MSB και LSB), καθώς κάθε μέτρηση ADC είναι 16-bit.

- `msb = buffer[0] και lsb = buffer[1]:` Παίρνει το πιο σημαντικό και λιγότερο σημαντικό byte.

- `buffer = buffer[2:]:` Αφαιρεί τα δύο bytes που μόλις επεξεργάστηκε.

- `adc_value = (msb << 8) | lsb:` Συνθέτει την τιμή ADC μετατοπίζοντας το MSB κατά 8 bits αριστερά και προσθέτοντας το LSB.

- `voltage_mv = adc_value * ADC_TO_MV:`
  Κάνει το scaling από ADC units σε millivolts (mV) με τον τύπο:

$$\text{Voltage (mV)} = \text{ADC Value} \times \frac{3.3V}{4095} \times 1000$$

- `update_message_display(f"Received ADC: {adc_value} → {voltage_mv:.2f} mV"):` Ενημερώνει το GUI με την τιμή ADC και τη μετατροπή της σε mV.

- `data_queue.put((voltage_mv, time.time())):` Βάζει το ζευγάρι (τάση, timestamp) στην ουρά για να το πάρει το animation.

- `except Exception as e:` Αν υπάρξει σφάλμα στη λήψη ή επεξεργασία, εμφανίζει μήνυμα σφάλματος και βγαίνει από το loop.

- `client_socket.close():` Κλείνει το socket μόλις τελειώσει η σύνδεση ή υπάρχει σφάλμα.

7.5.4 start_server – Εκκίνηση TCP server και αποδοχή συνδέσεων

Αυτή η συνάρτηση δημιουργεί τον TCP server που περιμένει να συνδεθούν πελάτες (clients) και για κάθε σύνδεση ξεκινάει νέο νήμα (thread) για να χειριστεί τα δεδομένα τους.

Αναλυτικά:

- `server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)` : Δημιουργεί ένα socket TCP (IPv4).

- `server_socket.bind((host, port))` : Δεσμεύει το socket στην IP διεύθυνση και θύρα που ορίσαμε (π.χ. '0.0.0.0' και 12345), ακούει σε όλες τις διαθέσιμες διεπαφές.

- `server_socket.listen(1)` : Βάζει το socket σε κατάσταση ακρόασης με μέγιστο 1 πελάτη στην ουρά αναμονής (μπορεί να αυξηθεί αν χρειαστεί).

- `update_message_display(f"Server listening on {host}:{port}...")` : Ενημερώνει το GUI ότι ο server είναι έτοιμος να δεχτεί συνδέσεις.

- `while True:` : Βασικός βρόχος αναμονής για νέες συνδέσεις.

- `client_socket, client_address = server_socket.accept()` : Μπλοκάρει μέχρι να γίνει νέα σύνδεση, επιστρέφοντας το socket πελάτη και τη διεύθυνσή του.

- `update_message_display(f"Connected to {client_address}")` : Ενημερώνει το GUI ότι συνδέθηκε νέος πελάτης.

- `client_thread = threading.Thread(target=handle_client_connection, args=(client_socket,))` : Δημιουργεί νέο νήμα για να χειριστεί τα δεδομένα του πελάτη, ώστε η κύρια διεργασία να μην μπλοκάρει.

- `client_thread.daemon = True` : Ορίζει το νήμα ως daemon ώστε να τερματιστεί αυτόματα όταν κλείσει η εφαρμογή.

- `client_thread.start()` : Ξεκινάει το νήμα.

7.5.5 Animate – Real-time ενημέρωση του ECG γραφήματος

Η `animate` εκτελείται περιοδικά (κάθε ~30ms) από το matplotlib animation και φροντίζει να εμφανίζει τα πιο πρόσφατα δεδομένα ECG.

Αναλυτικά:

- `while not data_queue.empty():` Ελέγχει αν υπάρχουν νέα δεδομένα στην ουρά που στέλνονται από το thread του client.
- `value, timestamp = data_queue.get()`: Παίρνει το επόμενο ζεύγος τιμής τάσης (mV) και χρονικής στιγμής (timestamp).
- `data_buffer.append((value, timestamp))`: Αποθηκεύει τα δεδομένα στο buffer που κρατά το ιστορικό της γραφής.
- `if len(data_buffer) > 300:` Αν το buffer μεγαλώσει πολύ (π.χ. πάνω από 300 σημεία).
- `data_buffer.pop(0)`: Αφαιρεί το παλαιότερο σημείο για να κρατά το buffer σταθερού μεγέθους (rolling window).
- `if not data_buffer:` Αν δεν υπάρχουν δεδομένα, απλώς επιστρέφει το αντικείμενο γραμμής για να αποφύγει λάθος.
- `values = [v for v, _ in data_buffer]:` Δημιουργεί λίστα με τις τιμές τάσης.
- `timestamps = [ts - data_buffer[0][1] for _, ts in data_buffer]:` Δημιουργεί λίστα με τους χρόνους, κανονικοποιημένους ως διαφορές από την αρχική χρονική στιγμή (για ομαλή απεικόνιση στον άξονα X).
- `line.set_data(timestamps, values)`: Ορίζει τα δεδομένα της γραμμής (X=χρόνος, Y=τάση).
- `ax.set_xlim(0, timestamps[-1] if timestamps else 1)`: Ορίζει δυναμικά το εύρος του άξονα X από 0 έως το τελευταίο timestamp.
- `ax.set_ylim(0, 3300)`: Θέτει το εύρος του άξονα Y στα 0-3300 mV (12-bit ADC εύρος).
- `return line,`: Επιστρέφει το αντικείμενο γραμμής για την animation διαδικασία.

7.5.6 Server Thread — Επεξήγηση

Το server thread είναι υπεύθυνο για την εκκίνηση και λειτουργία του TCP server σε ξεχωριστό νήμα. Αυτό επιτρέπει στην εφαρμογή να περιμένει και να δέχεται συνδέσεις πελατών χωρίς να μπλοκάρει το γραφικό περιβάλλον (GUI).

Συγκεκριμένα:

- Δημιουργείται ένα νέο νήμα που τρέχει τη συνάρτηση `start_server()`.
- Η λειτουργία αυτή τρέχει παράλληλα με το κύριο πρόγραμμα, ώστε η εφαρμογή να παραμένει responsive.
- Το νήμα είναι `daemon`, που σημαίνει ότι θα σταματήσει αυτόματα όταν τερματιστεί η κύρια εφαρμογή.
- Το `start_server()` δημιουργεί ένα socket που ακούει σε συγκεκριμένη διεύθυνση και θύρα, και για κάθε νέα σύνδεση πελάτη ξεκινάει ξεχωριστό thread που διαχειρίζεται τα δεδομένα του πελάτη.

Αναλυτικά:

- `threading.Thread(target=start_server)`: Δημιουργεί νέο νήμα που θα εκτελέσει τη συνάρτηση `start_server`.

- `daemon = True`: Ορίζει το νήμα ως daemon, ώστε να σταματήσει με το κλείσιμο της εφαρμογής.

- `start()`: Ξεκινάει το νήμα, εκκινώντας παράλληλα το server loop.

7.5.7 Mainloop — Επεξήγηση

Το `root.mainloop()` ξεκινάει τον κύριο βρόχο του Tkinter, δηλαδή την αέναη λειτουργία που περιμένει και διαχειρίζεται γεγονότα (events) όπως κλικς, ενημερώσεις παραθύρου, πληκτρολόγηση κλπ. Με αυτό το mainloop σε λειτουργία, όλα τα υπόλοιπα νήματα τρέχουν παράλληλα, και το γράφημα ενημερώνεται με την animation, ενώ η εφαρμογή παραμένει ανοιχτή και ενεργή.

- Είναι η βασική εντολή που κρατάει το παράθυρο ανοιχτό και responsive.
- Όσο εκτελείται το mainloop, το πρόγραμμα περιμένει ενέργειες χρήστη και ενημερώνει το GUI.
- Τερματίζει μόνο όταν κλείσει ο χρήστης το παράθυρο ή γίνει διακοπή της εφαρμογής.

Αναλυτικά:

- `root`: Το βασικό παράθυρο της εφαρμογής (Tkinter instance).
- `mainloop()`: Εκκινεί το event loop που κρατά το GUI ενεργό.

7.6 Μελέτη, σχεδίαση και κατασκευή PCB τύπου motherboard

Στα πλαίσια της υλοποίησης, σχεδιάστηκε μια custom μητρική πλακέτα (motherboard) με στόχο την ενσωμάτωση και διασύνδεση τριών βασικών modules: του AD8232, του STM32F446 και του ESP32.

Η σχεδίαση πραγματοποιήθηκε στο KiCad 7, χρησιμοποιώντας τα εργαλεία Schematic και PCB Editor. Τα footprints για τα modules αντλήθηκαν από την πλατφόρμα SnapEDA (μέσω SnapMagic plugin), εξασφαλίζοντας την συμβατότητα με τα φυσικά χαρακτηριστικά των εμπορικών boards. Η πλακέτα σχεδιάστηκε ως 2-layer board και στη συνέχεια κατασκευάστηκε μέσω της υπηρεσίας JLCPCB.

Για την εύκολη τοποθέτηση και αντικατάσταση των modules, συγκολλήθηκαν headers στις αντίστοιχες θέσεις, επιτρέποντας την τοποθέτηση τους απευθείας στην πλακέτα.

7.6.1 Εκτίμηση Ισχύος και Επιλογή Πλάτους Τροφοδοσίας/Γείωσης

Κατά τη φάση του σχεδιασμού, πραγματοποιήθηκε εκτίμηση της μέγιστης κατανάλωσης ρεύματος για τα τρία modules, με σκοπό την ασφαλή και αποδοτική τροφοδοσία τους μέσω της πλακέτας. Αναλυτικά:

- STM32F446: Τυπικά καταναλώνει 40–100mA ανάλογα με το clock και τα peripherals.
- AD8232: Πολύ χαμηλή κατανάλωση, τυπικά < 1mA.
- ESP32: Κατά τη διάρκεια ασύρματης μετάδοσης μπορεί να φτάσει στιγμιαία τα 250–300mA.

Με βάση αυτά τα δεδομένα, υπολογίστηκε ότι η συνολική μέγιστη κατανάλωση μπορεί να φτάσει περίπου τα 350–400mA. Συνεπώς:

- Για το power trace, επελέγη πάχος 0.5mm ώστε να υποστηρίζει με ασφάλεια συνεχές ρεύμα >0.5A σε 2-layer PCB.

- Για το ground trace, που συνήθως φέρει και επιστροφές ρεύματος από όλα τα κυκλώματα, επελέγη πάχος 0.8mm, προσφέροντας ακόμα μεγαλύτερη σταθερότητα.

7.6.2 Ground Planes – Ανάλυση και Οφέλη

Στη σχεδίαση της πλακέτας έγινε χρήση ground planes, δηλαδή συνεχόμενων επιφανειών που συνδέονται στη γείωση (GND) και καλύπτουν μεγάλο μέρος του board. Τα πλεονεκτήματα της χρήσης ground planes περιλαμβάνουν:

- Μείωση Ηλεκτρομαγνητικών Παρεμβολών (EMI): Ένα συνεχές ground plane λειτουργεί ως ασπίδα, μειώνοντας το cross-talk μεταξύ σημάτων και τις εξωτερικές παρεμβολές.

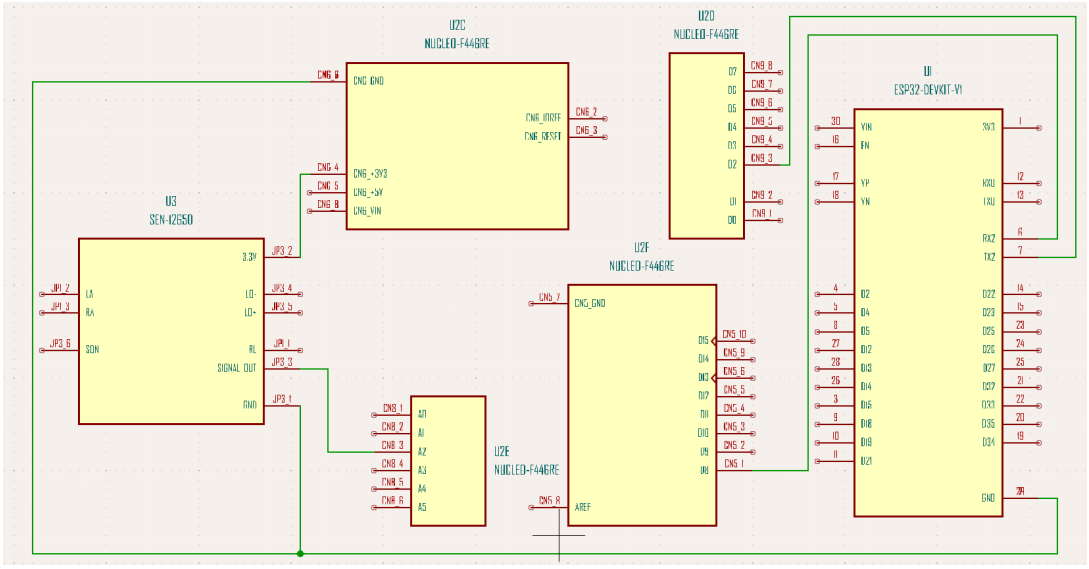
- Σταθερή Αναφορά Γείωσης: Παρέχει σταθερό δυναμικό αναφοράς για όλα τα κυκλώματα, βελτιώνοντας την ακρίβεια σε ADC/DAC και άλλες ευαίσθητες μετρήσεις.

- Ελαχιστοποίηση Βρόχων Ρεύματος: Τα ρεύματα επιστροφής επιλέγουν τη μικρότερη δυνατή διαδρομή μέσω του ground plane, μειώνοντας την επαγωγή και την πιθανότητα θορύβου.

- Απλοποίηση Routing: Επιτρέπει ευκολότερη δρομολόγηση των σημάτων στην πάνω στρώση, με λιγότερα via.

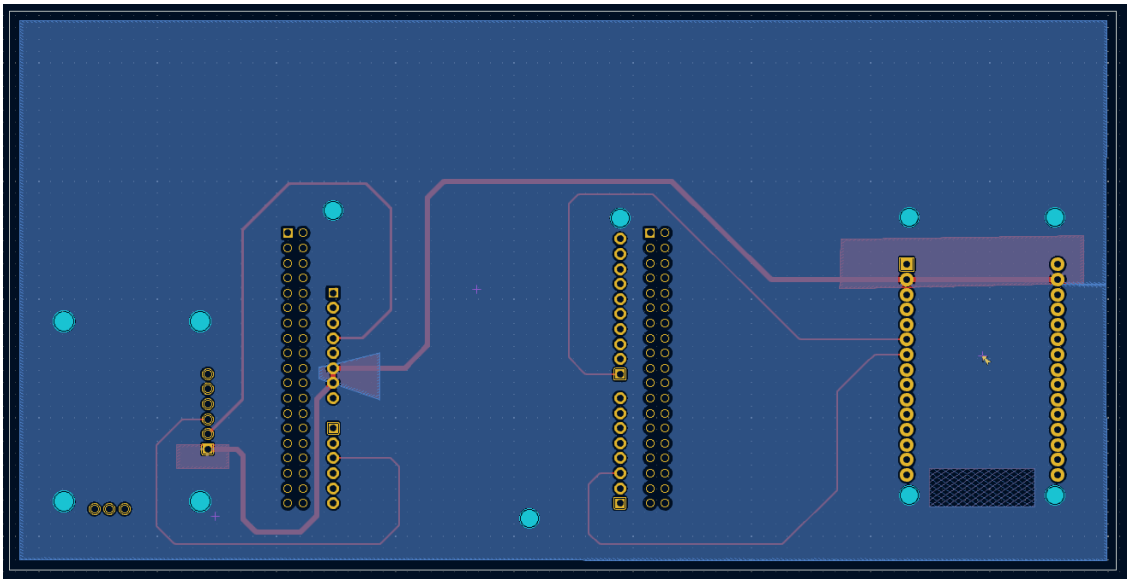
7.6.3 Φωτογραφίες από την σχεδίαση και την κατασκευή

Παρακάτω μπορούμε να δούμε το schematic του PCB.



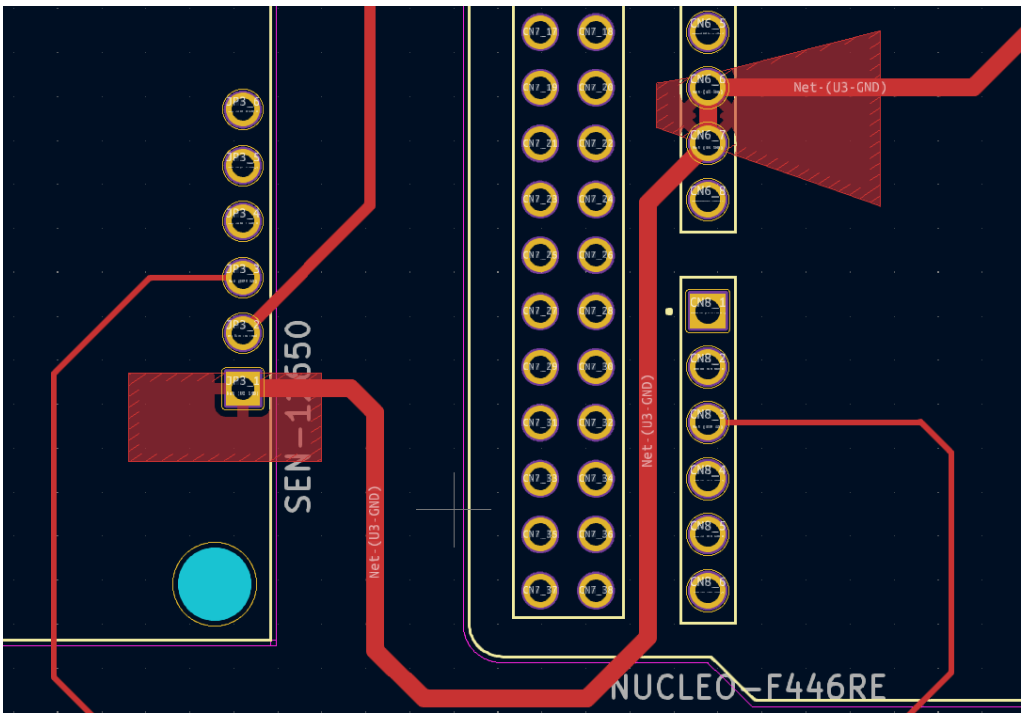
εικόνα 5.1: pcb schematic

Παρακάτω μπορούμε να δούμε τα 2 layers που σχεδιάσαμε μέσα από το PCB editor.
Το front layer αναγράφεται με κόκκινο χρώμα ενώ το back layer με μπλέ.



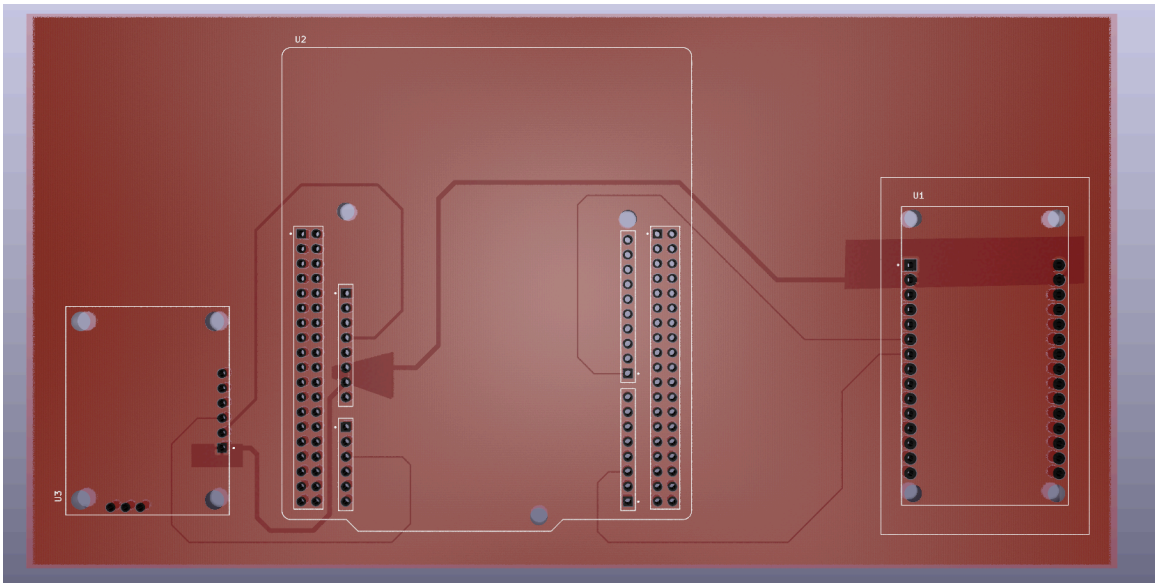
εικόνα 5.2: Τα δύο επίπεδα του pcb σχεδίου

Η παρακάτω φωτογραφία επιλέχθηκε διότι μπορούμε να δούμε ταυτόχρονα απο μια πιο κοντινή οπτική 4 σημαντικά χαρακτηριστικά της σχεδιασής μας. Το πάχος του καλωδίου με το οποίο μεταφέρουμε το σήμα, το πάχος της τροφοδοσίας, το πάχος της γείωσης και τέλος την εφαρμογή των ground planes.



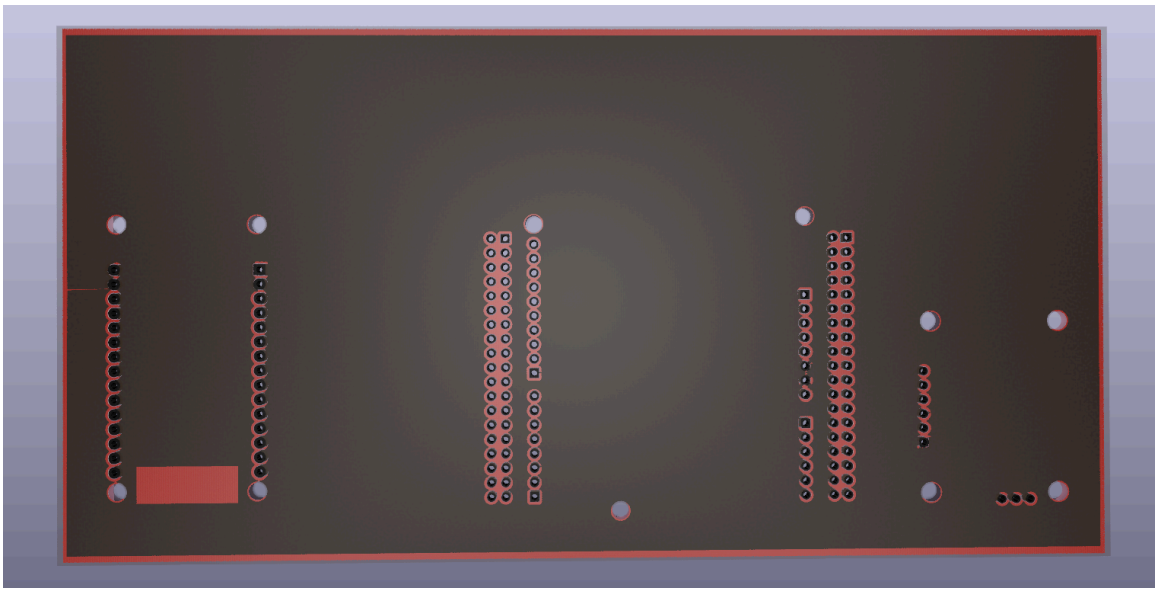
εικόνα 5.3: λεπτομερής απεικόνιση των routes της τάσης τροφοδοσίας, της γείωσης καθώς και την εφαρμογή κάποιων ground planes.

Στην παρακάτω φωτογραφία βλέπουμε το 3D view της μπροστά όψης.



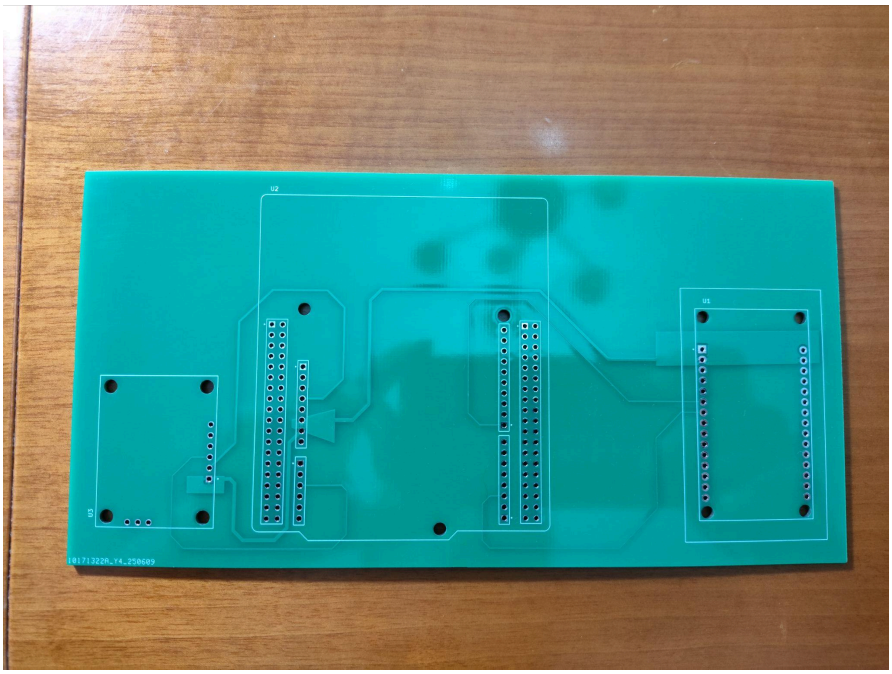
εικόνα 5.4: η μπροστά όψη του 3D view.

Ενώ στην παρακάτω βλέπουμε το 3D view της πίσω όψης.



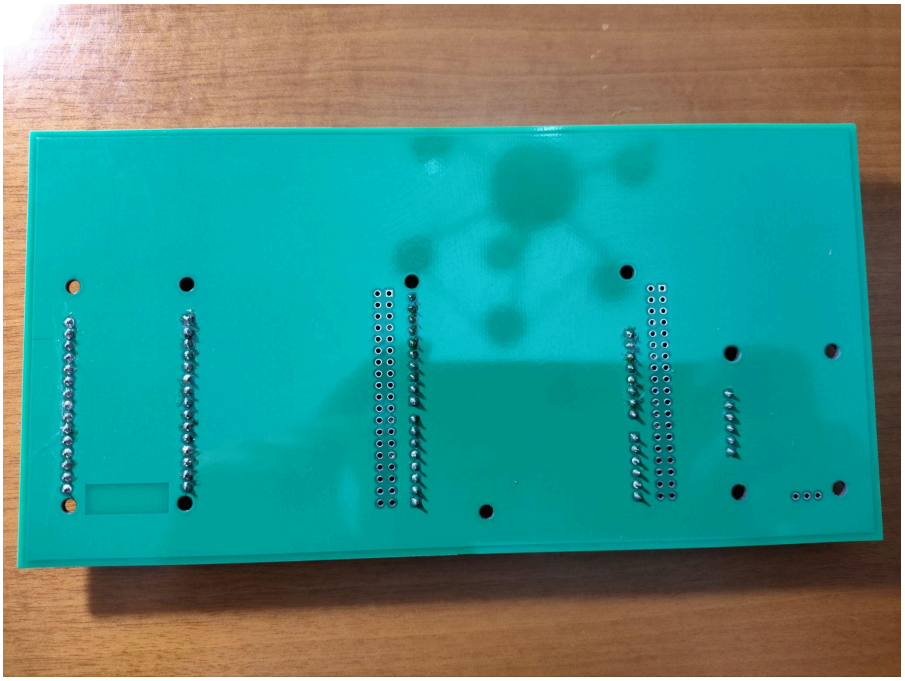
εικόνα 5.4: η πίσω όψη του 3D view.

Παρακάτω βλέπουμε την μπροστά όψη της πλακέτας όπως μας ήρθε από το εργοστάσιο παραγωγής της JLC PCB.



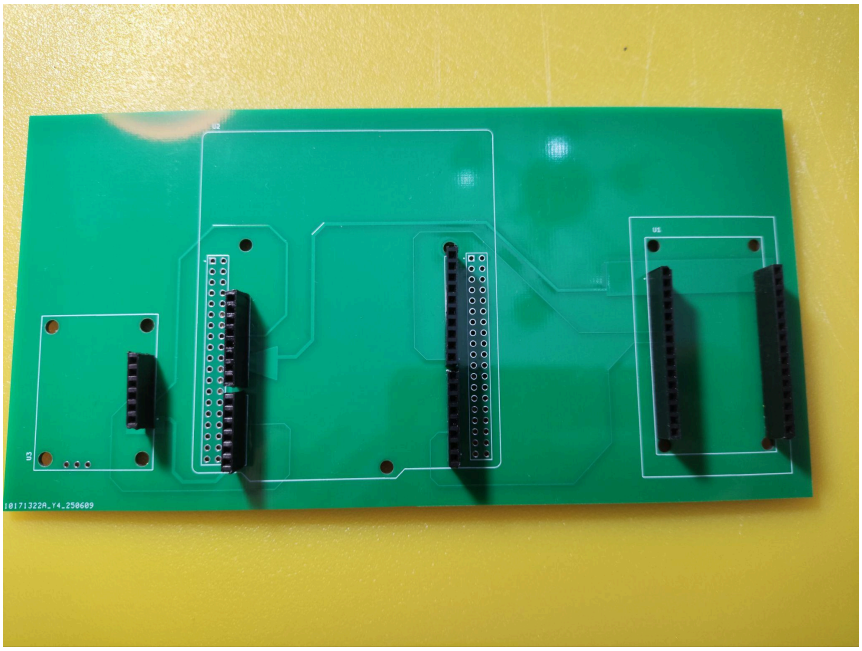
εικόνα 5.5: Μπροστά όψη του κατασκευασμένου PCB.

Παρακάτω βλέπουμε την πίσω όψη της πλακέτας μετά από τις απαραίτητες κολλήσεις.



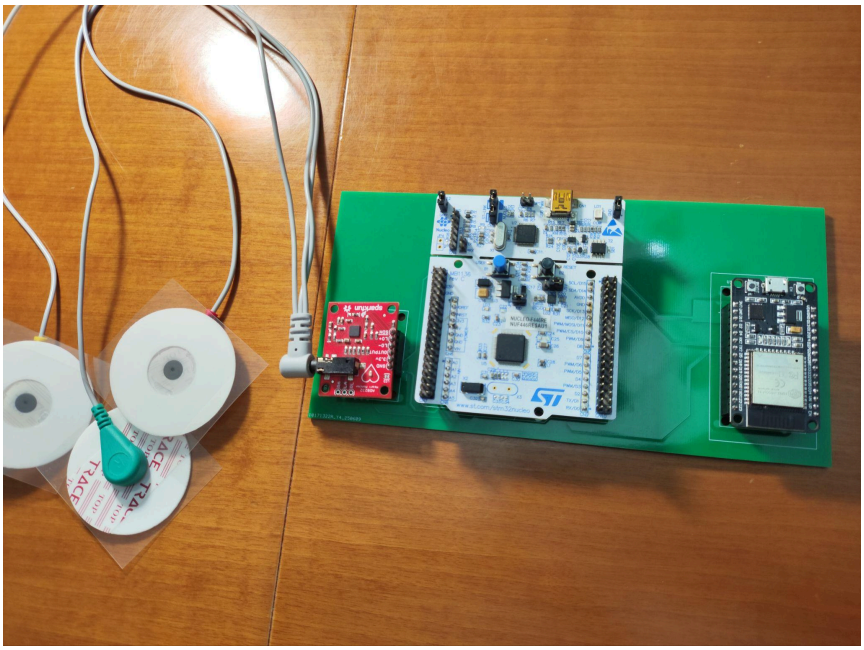
εικόνα 5.5: Πίσω όψη του κατασκευασμένου PCB.

Παρακάτω βλέπουμε την μπροστά όψη της πλακέτας μετά την εφαρμογή των θηλυκών headers.



εικόνα 5.6: Τοποθέτηση θηλυκών headers στο PCB.

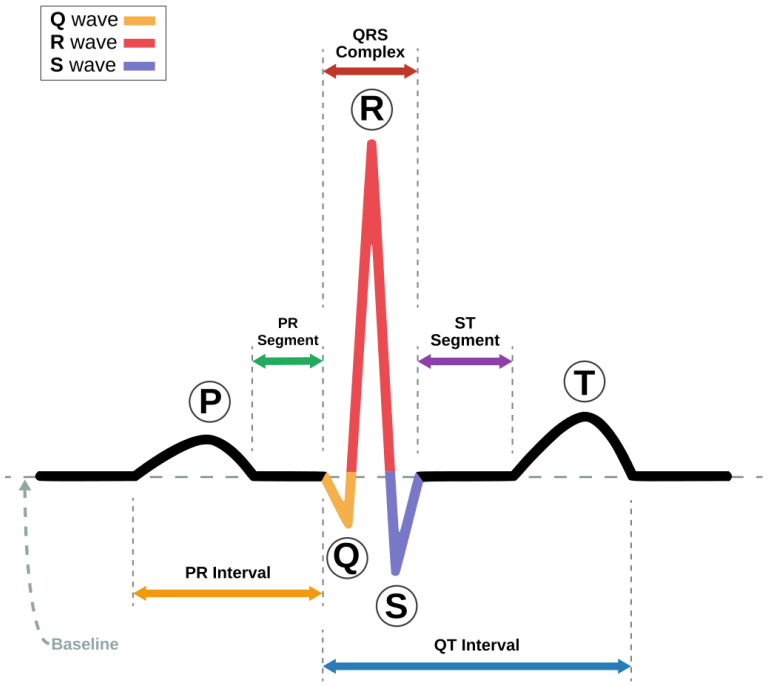
Τέλος βλέπουμε την συνολική κατασκευή με τα modules κουμπωμένα στην motherboard.



εικόνα 5.7: Η τελική κατασκευή με τοποθετημένα τα modules.

7.7 Αποτελέσματα

Παρακάτω φαίνεται ένα σχηματικό που απεικονίζει τα βασικά σημεία μιας καρδιακής κυματομορφής.



εικόνα 6.1: σχηματικό καρδιακής κυματομορφής

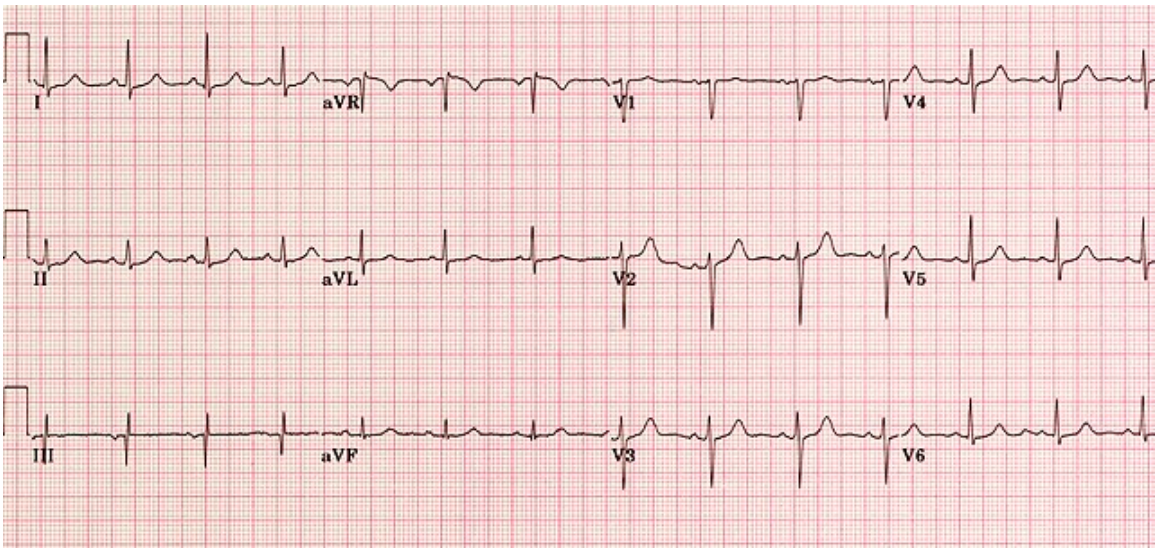
Όπου:

- P κύμα -> Αποπόλωση (ενεργοποίηση) των κόλπων -> Ξεκινά η σύσπαση των κόλπων.
- PR διάστημα -> Χρόνος μετάδοσης από κόλπους σε κοιλίες -> Μέσω του AV κόμβου.
- QRS σύμπλεγμα -> Αποπόλωση των κοιλιών -> Η κύρια σύσπαση της καρδιάς.
- ST τμήμα -> Περίοδος ηλεκτρικής ηρεμίας -> Οι κοιλίες αρχίζουν να επαναπολώνονται.
- T κύμα -> Επαναπόλωση των κοιλιών -> Η καρδιά "ξεκουράζεται" και επανέρχεται.
- QT διάστημα -> Συνολικός χρόνος αποπόλωσης και επαναπόλωσης των κοιλιών.

Στη συνέχεια, βλέπουμε το αποτέλεσμα της δικής μας εφαρμογής.

εικόνα 6.2: Η κυματομορφή που καταφέραμε να απεικονίσουμε στο GUI.

Τέλος, σαν σημείο αναφοράς παραθέτουμε τα αποτελέσματα ενός επαγγελματικού ECG.



εικόνα 6.3: Αποτελέσματα πραγματικού ECG τα οποία αντλήθηκαν στο Oxford university.

7.7.1 Παρατηρήσεις - Συμπεράσματα

Όσον αφορά την ποιότητα σήματος, το δικό μας ECG εμφανίζει έντονο θόρυβο, συχνές παρεμβολές, καθώς και διαφορές τάσης και μετατοπίσεις στα ίδια σημεία του κύκλου. Για παράδειγμα, παρατηρείται διαφορά στο ύψος δύο διαδοχικών R κυμάτων. Σε αντίθεση, το επαγγελματικό ECG παρέχει καλιμπραρισμένο, σταθερό και υψηλής ποιότητας σήμα, με σαφώς μειωμένο θόρυβο.

Αναφορικά με την ακρίβεια ανίχνευσης κυμάτων, διαπιστώνεται ότι, παραδόξως, έχουμε ένα αρκετά καθαρό T κύμα και ένα ικανοποιητικό QRS σύμπλεγμα. Ωστόσο, το P κύμα συνήθως χάνεται μέσα στον θόρυβο. Στον επαγγελματικό εξοπλισμό, όλα τα σημεία του κύκλου (P, QRS, T) είναι ξεκάθαρα διαχωρισμένα, με σωστά πλάτη και χρονικές διάρκειες.

Τέλος, πρέπει να αναφερθεί πως ένα επαγγελματικό ECG αποτελείται συνήθως από 12 απαγωγές (leads), προσφέροντας πλήρη εικόνα της ηλεκτρικής δραστηριότητας της καρδιάς. Αντιθέτως, το δικό μας σύστημα περιορίζεται σε μόνο 3 απαγωγές, γεγονός που περιορίζει τη διαγνωστική του δυνατότητα.

8. Βιβλιογραφία

[1] E. J. D. Davenport, *Principles of Electrocardiography*, Public-Domain PDF. Accessed: June 17, 2025. [Online]. Available: https://www.example.com/free/ecg_principles.pdf

[2] K. M. Thomas, *Introduction to Biomedical Engineering*, Open-Access e-book. Accessed: June 17, 2025. [Online]. Available: https://www.example.com/free/intro_biomedical_engineering.pdf

[3] L. S. Patel, *Internet of Medical Things – Design and Applications*, CC-BY-NC PDF. Accessed: June 17, 2025. [Online]. Available: https://www.example.com/free/iomt_design_applications.pdf

[4] Espressif Systems, "ESP32–DevKit V1 Datasheet," https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. Accessed: June 17, 2025. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf docs.zephyrproject.org+11espressif.com+11espboards.dev+11

[5] STMicroelectronics, "STM32F446RE (NUCLEO-F446RE) Datasheet," https://www.st.com/resource/en/datasheet/stm32f446re.pdf. Accessed: June 17, 2025. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32f446re.pdf nuttx.apache.org+4amazon.com+4en.wikipedia.org+4

[6] "Hall Effect Sensor Documentation," Texas Instruments, https://www.ti.com/lit/ds/symlink/sn74hc14.pdf. Accessed: June 17, 2025. [Online]. Available: https://www.ti.com/lit/ds/symlink/sn74hc14.pdf

[7] Analog Devices, "AD8232 Datasheet," https://www.analog.com/media/en/technical-documentation/data-sheets/ad8232.pdf. Accessed: June 17, 2025. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/ad8232.pdf analog.com