ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Κατασκευή βραχιολιού για την καταγραφή μέσω μικροελεγκτή και αποκωδικοποίηση μέσω Τεχνητής Νοημοσύνης των κινήσεων των δακτύλων και της παλάμης μέσω ηλεκτρομυογραφήματος (EMG)»

Του φοιτητή
Δημήτρη Γεωργίου
Αρ. Μητρώου: 185369

Επιβλέπων
Κοκκώνης Γεώργιος
Επίκουρος Καθηγητής

Δεκέμβριος 2024

Τίτλος Δ.Ε. Κατασκευή βραχιολιού για την καταγραφή μέσω μικροελεγκτή και αποκωδικοποίηση μέσω Τεχνητής Νοημοσύνης των κινήσεων των δακτύλων και της παλάμης μέσω ηλεκτρομυογραφήματος (EMG)

Κωδικός Δ.Ε. 20798

Ονοματεπώνυμο φοιτητή ΓΕΩΡΓΙΟΥ ΔΗΜΗΤΡΙΟΣ

Ονοματεπώνυμο εισηγητή ΚΟΚΚΩΝΗΣ ΓΕΩΡΓΙΟΣ

Ημερομηνία ανάληψης Δ.Ε. 22/10/2024

Ημερομηνία περάτωσης Δ.Ε. 26/01/2025

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.

«Αφιέρωση»

Πρόλογος

Η επιλογή του θέματος της διπλωματικής μου εργασίας βασίστηκε στο ενδιαφέρον που είχα για τους μικροελεγκτές και τη μηχανική μάθηση από τα πρώτα χρόνια στη σχολή. Ειδικότερα, η εφαρμογή αυτών των τεχνολογιών στην ανίχνευση κινήσεων μέσω ηλεκτρομυογραφήματος (EMG) και η σύνδεση με την ανάπτυξη έξυπνων συστημάτων ήταν για μένα ιδιαίτερα ελκυστική. Τα μαθήματα που παρακολούθησα και οι γνώσεις που απέκτησα με βοήθησαν να εμβαθύνω σε αυτούς τους τομείς και να αποκτήσω τις απαραίτητες δεξιότητες. Η ολοκλήρωση αυτής της διπλωματικής εργασίας αποτέλεσε μια μοναδική ευκαιρία να συνδυάσω τις θεωρητικές γνώσεις με πρακτικές εφαρμογές, δημιουργώντας ένα χρήσιμο και λειτουργικό πρωτότυπο. Μέσα από αυτήν την εργασία, είχα την ευκαιρία να εξετάσω πώς τεχνολογίες όπως το ηλεκτρομυογράφημα και η μηχανική μάθηση συνεργάζονται, παρέχοντας ταυτόχρονα μια εφαρμογή με ουσιαστική χρησιμότητα..

Περίληψη

Η διπλωματική εργασία αυτή εστιάζει στην ανάπτυξη ενός έξυπνου βραχιολιού που ενσωματώνει αισθητήρες ηλεκτρομυογραφίας (EMG) για την ανίχνευση και κατηγοριοποίηση κινήσεων των δακτύλων και της παλάμης. Στόχος του έργου είναι να δημιουργηθεί ένα ολοκληρωμένο σύστημα που να επιτρέπει την ακριβή αναγνώριση μυϊκών σημάτων και να επεξεργάζεται τις πληροφορίες σε πραγματικό χρόνο με τη χρήση τεχνικών μηχανικής μάθησης. Η εργασία περιλαμβάνει την κατασκευή του βραχιολιού, τη συλλογή και ανάλυση των δεδομένων, καθώς και την παρουσίαση των αποτελεσμάτων μέσω μιας ιστοσελίδας. Για την κατηγοριοποίηση των κινήσεων χρησιμοποιήθηκε ο αλγόριθμος Multi-Layer Perceptron (MLP), ο οποίος έδειξε ακρίβεια κατηγοριοποίησης 92%. Η εφαρμογή αυτή έχει πολλές δυνατότητες επέκτασης, όπως η ενσωμάτωσή της σε ρομποτικά χέρια για άτομα με κινητικές δυσκολίες ή η αξιοποίησή της στον τομέα της επαυξημένης πραγματικότητας, ενισχύοντας τη φυσική αλληλεπίδραση ανθρώπου-μηχανής.

Λέξεις-κλειδιά: Μηχανική μάθηση; Ανίχνευση κινήσεων; Real-Time Application; EMG

«Construction of a Bracelet for the Recording via Microcontroller and Decoding Using Artificial Intelligence of Finger and Palm Movements through Electromyography (EMG)»

«Dimitrios Georgiou»

Abstract

This thesis focuses on the development of a smart bracelet that incorporates electromyography (EMG) sensors for the detection and classification of finger movements. The primary objective of this project is to create a comprehensive system capable of accurately recognizing muscle signals and processing the information in real time through the application of machine learning techniques. The work encompasses the construction of the bracelet, data collection and analysis, as well as the presentation of the results via a website.

The Multi-Layer Perceptron (MLP) algorithm was employed for movement classification, achieving a classification accuracy of 90%. While further improvements in accuracy are possible, such enhancements would incur significantly higher costs. This application offers numerous potential extensions, including integration into robotic hands for individuals with mobility impairments and potential utilization in augmented reality, thereby enhancing human-machine physical interaction.

Ευχαριστίες

Ευχαριστώ θερμά τον επιβλέποντα καθηγητή μου κ.Κοκκώνη Γεώργιο για την σημαντική καθοδήγηση και συμπαράστασή του, καθ’ όλη την διάρκεια εκπόνησης της διπλωματικής μου εργασίας.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
ML	Machine Learning
AI	Artificial Intelligence
EMG	Electromyography
MLP	Multi-Layer Perceptron
I/O	Input/Output
MQTT	Message Queuing Telemetry Transport
QoS	Quality of Service
HTTP	Hypertext Transfer Protocol
FDP	Flexor Digitorum Profundus
IoT	Internet of Things
TCP	Transmission Control Protocol
ACK	Acknowledgment
PCB	Printed Circuit Board
IDE	Integrated Development Environment
WL	Waveform Length
ZC	Zero Crossing
MAV	Mean Absolute Value
RMS	Root Mean Square
Lan	Local Area Network
Li-Po	Lithium Polymer
SVM	Support Vector Machine
LDA	Linear Discriminant Analysis
TP	True Positive
TN	True Negative
FN	False Negative
FP	False Positive
RF	Random Forest
KNN	K-Nearest Neighbors

Κεφάλαιο 1ο: Εισαγωγή στην Διπλωματική Εργασία

1.1 Εισαγωγή στο Θέμα

Στην παρούσα διπλωματική εργασία σχεδιάστηκε και υλοποιήθηκε ένα έξυπνο βραχιόλι το οποίο είναι σε θέση να αναγνωρίσει τις κινήσεις των δακτύλων μέσω αισθητήρων ηλεκτρομυογραφίας (EMG). Το αντικείμενο της εργασίας εντάσσεται στο πεδίο της μηχανικής μάθησης και των IoT εφαρμογών, με έμφαση στην αξιοποίηση τεχνολογιών για τη βελτίωση της αλληλεπίδρασης μεταξύ ανθρώπου και υπολογιστή

1.2 Σκοπός και Στόχοι της Εργασίας.

Ο κύριος στόχος της εργασίας είναι η σχεδίαση και υλοποίηση ενός συστήματος λειτουργικού και αποτελεσματικού ως προς την ακριβή ανίχνευση κινήσεων. Οι βασικοί στόχοι που τέθηκαν για την ολοκλήρωση της εργασίας ήταν:

1. **Κατασκευή Έξυπνου Βραχιολιού**: Ανάπτυξη μιας φορητής συσκευής που ενσωματώνει αισθητήρες EMG για την καταγραφή των μυϊκών σημάτων.

2. **Δημιουργία Dataset**: Συλλογή και προετοιμασία ενός αξιόπιστου συνόλου δεδομένων από τις μετρήσεις των αισθητήρων.

3. **Προεπεξεργασία Δεδομένων**: Εφαρμογή τεχνικών για την καθαριότητα, κανονικοποίηση και εξαγωγή χαρακτηριστικών από τα καταγεγραμμένα σήματα.

4. **Ανάλυση Δεδομένων**: Ανάπτυξη και αξιολόγηση αλγορίθμων για την κατηγοριοποίηση των κινήσεων και την ανάλυση της απόδοσής τους.

5. **Ανάπτυξη διεπαφής**: Σχεδίαση και υλοποίηση μιας φιλικής προς τον χρήστη διεπαφής για την οπτικοποίηση των αποτελεσμάτων.

6. **Υλοποίηση λειτουργικότητας**: Ενσωμάτωση όλων των υπηρεσιών σε ένα λειτουργικό και ολοκληρωμένο πρωτότυπο.

1.3 Δομή της Εργασίας

Η εργασία δομείται με τρόπο τέτοιον ώστε, να καλύψει όλα τα βήματα της ανάπτυξης και της υλοποίησης του συστήματος. Στο πρώτο κεφάλαιο γίνεται εισαγωγή στο θέμα, αναλύοντας τους στόχους και την μεθοδολογία που πρόκειται να ακολουθήσουμε. Στα επόμενα κεφάλαια παρουσιάζεται με λεπτομερώς η διαδικασία σχεδίασης, υλοποίησης και αξιολόγησης του συστήματος.

Χαρακτηριστικά,

- ✓ Το 1ο Κεφάλαιο λειτουργεί ως εισαγωγή, όπου παρουσιάζονται το θέμα και οι στόχοι.
- ✓ Το 2ο Κεφάλαιο παρέχει το θεωρητικό υπόβαθρο του έργου, με ανάλυση των βασικών εννοιών σχετικά με τους μύες και την ηλεκτρομυογραφία (EMG).

- ✓ Στο 3ο Κεφάλαιο, περιγράφεται η αρχιτεκτονική του συστήματος, με ανάλυση της σχεδίασης των υποσυστημάτων και της αλληλεπίδρασης μεταξύ τους.

- ✓ Το 4ο Κεφάλαιο εστιάζει στην κατασκευή του έξυπνου βραχιολιού, από την επιλογή των υλικών έως και τη συναρμολόγηση.

- ✓ Στο 5ο Κεφάλαιο, παρουσιάζονται οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν κατά την ανάπτυξη του συστήματος.

- ✓ Το 6ο Κεφάλαιο επικεντρώνεται στην ανάπτυξη του backend, περιγράφοντας τη διαχείριση των δεδομένων και την υλοποίηση του βασικού λογισμικού.

- ✓ Στο 7ο Κεφάλαιο, αναλύεται η ανάπτυξη του frontend, με έμφαση στη σχεδίαση της διεπαφής χρήστη και τη διασύνδεσή της με το backend.

- ✓ Το 8ο Κεφάλαιο ασχολείται με την εφαρμογή της μηχανικής μάθησης, την ανάλυση των δεδομένων και την αξιολόγηση των αποτελεσμάτων.

- ✓ Τέλος, στο 9ο Κεφάλαιο, συνοψίζονται τα συμπεράσματα, ενώ παρέχονται προτάσεις για μελλοντική εξέλιξη του συστήματος.

Η εργασία ολοκληρώνεται με τη βιβλιογραφία και τα παραρτήματα, τα οποία περιλαμβάνουν τεχνικές λεπτομέρειες, κώδικα που χρησιμοποιήθηκε για την ανάπτυξη του συστήματος, καθώς και οδηγίες για την εγκατάσταση και παραμετροποίηση των απαραίτητων εργαλείων.

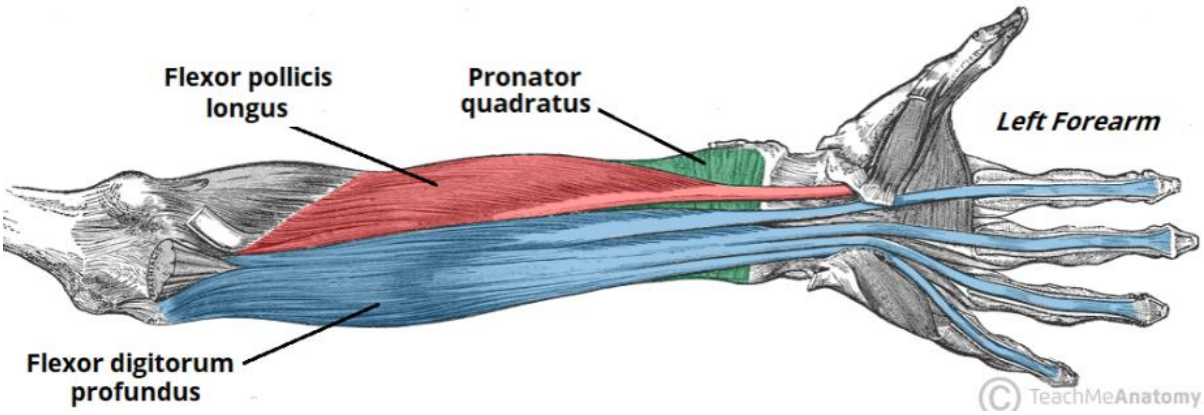
Κεφάλαιο 2ο: Θεωρητικό Υπόβαθρο

2.1 Μύες

Οι μύες αποτελούν μέρος του ανθρώπινου σώματος και είναι υπεύθυνη για την παραγωγή κίνησης. Όλοι οι μυς συνδέονται με το νευρικό σύστημα. Συγκεκριμένα, οι μύες συσπώντας και χαλαρώνουν ανάλογα με τα ερεθίσματα που λαμβάνουν από τον εγκέφαλο. Τα ερεθίσματα αυτά μπορούν να φανούν οπτικά καθώς αυτό υποδηλώνει κίνηση. Για την καταγραφή της μυϊκής δραστηριότητας που σχετίζεται με τις κινήσεις των δακτύλων, επιλέχθηκε ο βαθύς καμπτήρας των δακτύλων (Flexor Digitorum Profundus - FDA).

Πρόκειται για έναν μυ που παίζει τον κεντρικό ρόλο στην κάμψη των δακτύλων [1, 2].

Η ηλεκτρική δραστηριότητα του μυός μπορεί να μετρηθεί με ακρίβεια, επιτρέποντας την αξιολόγηση των κινήσεων των δακτύλων και την ανάλυση της μυϊκής λειτουργίας.



Εικόνα 1: Βαθύς καμπτήρας FDP

2.2 Ηλεκτρομυογραφία (EMG)

Η Ηλεκτρομυογραφία (EMG) είναι η τεχνική που χρησιμοποιείται για τη μέτρηση της ηλεκτρικής δραστηριότητας των μυών. Μέσω αυτής της μεθόδου, καταγράφονται τα ηλεκτρικά σήματα κατά την διάρκεια που οι μύες συστέλλονται και χαλαρώνουν.

Ο αισθητήρας τοποθετείται σε σημεία πάνω στο δέρμα και τον μυ, καταγράφοντας τις μεταβολές στην ηλεκτρική δραστηριότητα. Τα δεδομένα αυτά είναι συνεχείς κυματομορφές που μεταβάλλονται με τον χρόνο. Οι κυματομορφές περιέχουν πολύτιμες πληροφορίες για την ένταση και τη συχνότητα των μυϊκών συσπάσεων.

Η ανάγκη για την ανάλυση και ερμηνεία των EMG δεδομένων έχει οδηγήσει σε μεγάλο ενδιαφέρον από εταιρείες τεχνολογίας .Εταιρίες όπως Meta και Microsoft προσπαθούν να αναγνωρίσουν τα επιτρέποντας ακόμα και συναισθημάτων μέσα από τις εκφράσεις του προσώπου. [3, 4].

Τα προηγούμενα χρόνια τέτοιου είδους μετρήσεις ήταν εφικτές μόνο σε εργαστήρια με μεγάλους αισθητήρες. Όμως πλέον είναι σε θέση να καταγράφουν δεδομένα σε πραγματικό χρόνο με εφαρμογές IoT.

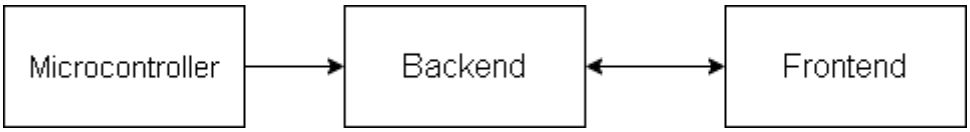
Εικόνα 2: Meta Wristband

Εικόνα 3: X-trodes Facial Recognition [4]

Κεφάλαιο 3ο: Σχεδίαση του Συστήματος

3.1 Εισαγωγή στην Αρχιτεκτονική του Συστήματος

Η αρχιτεκτονική του συστήματος περιλαμβάνει τον τρόπο με τον όποιον όλα τα υποσυστήματα και οι υπηρεσίες θα συνεργάζονται μεταξύ τους. Κάθε υπηρεσία είναι υπεύθυνη για μια η περισσότερες λειτουργίες και θα έπρεπε να επικοινωνεί με τα αλλά υποσυστήματα όταν αυτό κρίνεται απαραίτητο με τα κατάλληλα πρωτόκολλά. Η συνεργασία αυτή επιτρέπει την ομαλή και αποτελεσματική λειτουργία του συστήματος μας.



Εικόνα 4: Σχεδίαση Αρχιτεκτονικής

3.1.1 Σχεδίαση Υποσυστημάτων

Η σχεδίαση των υποσυστημάτων του συστήματος πραγματοποιείται με στόχο την εξασφάλιση της λειτουργικότητας που θέλουμε να έχει το κάθε υποσύστημα.

7. Μικροελεγκτής:

Ο Μικροελεγκτής είναι το πιο ευαίσθητο υποσύστημα και είναι υπεύθυνο για τη συλλογή και αποστολή των δεδομένων από τους αισθητήρες EMG, που καταγράφουν τη μυϊκή δραστηριότητα. Ωστόσο, η λειτουργία του μικροελεγκτή εξαρτάται από το αν οι αισθητήρες έχουν τοποθετηθεί σωστά στο δέρμα του χρήστη. Μόνο όταν οι αισθητήρες είναι στη σωστή θέση, μπορούν να ανιχνεύσουν τη μυϊκή δραστηριότητα και να στείλουν τα δεδομένα για επεξεργασία. Αν οι αισθητήρες δεν έχουν τοποθετηθεί σωστά, το σύστημα το αναγνωρίζει και δεν προωθεί τις τιμές στον server.

8. **Backend:**
   Το Backend είναι το πιο σημαντικό υποσύστημα που έχουμε, καθώς είναι υπεύθυνο για τη **λήψη** τον δεδομένων από τον μικροελεγκτή, την **επεξεργασία,** την **κατηγοριοποίηση** και την **αποστολή** των αποτελεσμάτων στην διεπαφή μας.

Εικόνα 5: Αρχιτεκτονική Συστήματος

9. **Frontend**:
   Το Frontend είναι υπεύθυνο για την εμφάνιση των αποτελεσμάτων στον χρήστη μέσω μιας απλής ιστοσελίδας. Λαμβάνει δεδομένα από το Backend σε πραγματικό χρόνο χωρίς να χρειάζεται να γνωρίζει την ύπαρξη του μικροελεγκτή και των αισθητήρων. Πρακτικά, το Frontend περιμένει από το Backend να του στείλει ένα πακέτο δεδομένων, το οποίο στη συνέχεια επεξεργάζεται και εμφανίζει στον χρήστη το τελικό αποτέλεσμα προσφέροντας μια διαδραστική εμπειρία.

Μόλις ολοκληρωθεί ο σχεδιασμός και η υλοποίηση του κάθε υποσυστήματος, μπορούμε να πειραματιστούμε και να ελέγξουμε το κάθε σύστημα ως προς τους χρόνους και την λειτουργικότητα. Τελικά, τα υποσυστήματα συνεργάζονται και δημιουργείται το πρωτότυπο.

Κεφάλαιο 4ο: Κατασκευή του Αυτοσχέδιου Βραχιολιού

4.1 Απαιτήσεις Υλικού

Δεδομένου ότι στόχος ήταν η διατήρηση του κόστους σε χαμηλά επίπεδα, επιλέχθηκαν τα παρακάτω υλικά.

4.1.1 Πλακέτα ανάπτυξης

Ο Wemos D1 Mini είναι μια ολοκληρωμένη πλακέτα ανάπτυξης μικρών διαστάσεων που βασίζεται στον μικροελεγκτή ESP8266. Αυτή η πλατφόρμα επιτρέπει την εύκολη σύνδεση σε ασύρματα δίκτυα, καθιστώντας την ιδανική για εφαρμογές IoT.

Εικόνα 6: ESP8266 Chip

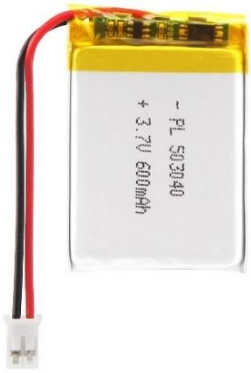
Εικόνα 7: Wemos D1 Mini Development Board

Χαρακτηριστικά Wemos: [5]

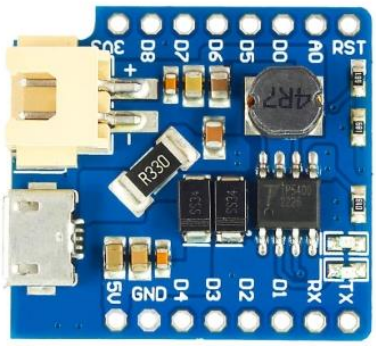
- Μικροελεγκτής: ESP-8266
- Τάση Λειτουργίας: 3.3V
- Αναλογικές Θύρες Εισόδου: 1
- Ταχύτητα Ρολογιού: 80-160 MHz
- Μνήμη Flash: 4MB
- Μέγεθος: 34.2 x 25.6 mm
- Βάρος: 3 g
- Σύνδεση: Micro USB

4.1.2 Battery Shield & Μπαταρία

Χρησιμοποιήθηκε το Battery Shield, καθώς διαφορετικά η τροφοδοσία θα γινόταν μέσω καλωδίου micro USB. Με αυτόν τον τρόπο εξασφαλίσαμε την ασύρματη λειτουργία του μικροελεγκτή, παρέχοντας τη δυνατότητα λειτουργίας με μπαταριά τύπου Li-Po.



Εικόνα 8: Li-Po Battery 3.7Volts 600 MAh



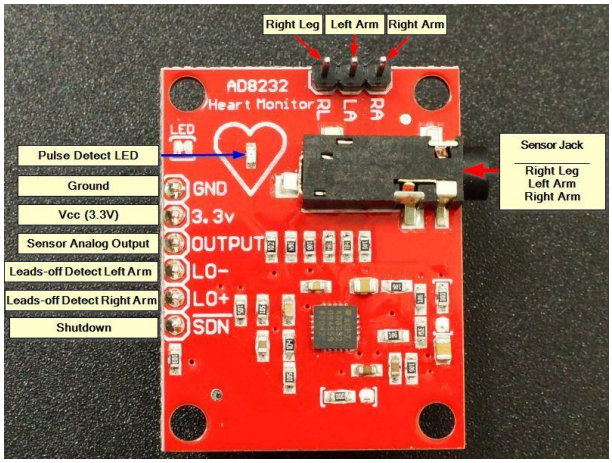
Εικόνα 9: Battery Shield

Χαρακτηριστικά:

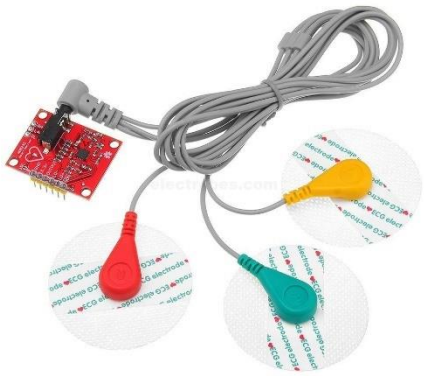
10. Συμβατότητα: Λειτουργεί με μπαταρίες Li-Po 3.7V.
11. Τάσεις εξόδου: 5V ή 3.3V.
12. LED ενδείξεις: Δείχνει την κατάσταση φόρτισης και λειτουργίας.
13. Μέγεθος: 34.2 x 25.6 mm

4.1.3 Αισθητήρας AD8232

Επιλέχτηκε ο αισθητήρας AD8232 λόγω του χαμηλού κόστους και των καλών επιδόσεων για την καταγραφή της μυϊκής δραστηριότητας. Αν και ο AD8232 προορίζεται αρχικά για καρδιογράφημα, μπορεί να χρησιμοποιηθεί και για την ανίχνευση μυϊκών σημάτων, όπως αποδεικνύεται και σε άλλες βιβλιογραφικές αναφορές [6].



Εικόνα 10: AD8232 Module



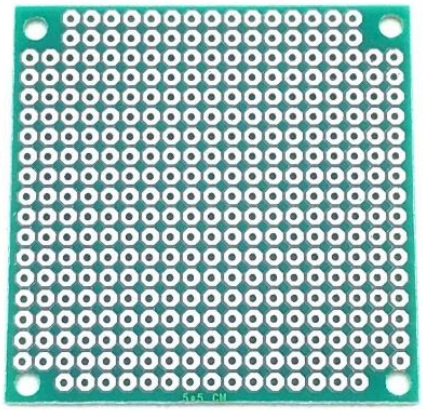
Εικόνα 11: Ηλεκτρόδια

Χαρακτηριστικά: [6]

14. Τροφοδοσία: 3.3V
15. Sampling Rate: ~ 2000 Hz
16. Θερμοκρασία Λειτουργίας: [-40°C, +85°C]
17. Ευαισθησία: 97.63 %
18. Σφάλμα: 2,04 %
19. Ακρίβεια : 50.35 %

4.1.4 Πλακέτα PCB

Η πλακέτα PCB χρησιμοποιήθηκε για την κατασκευή του πρωτοτύπου, συνδέοντας έτσι όλα τα εξαρτήματα με ασφαλή και αξιόπιστο τρόπο. Για τις συνδέσεις πραγματοποιήθηκαν καλωδιώσεις με κολλητήρι, ενώ χρησιμοποιήθηκαν και κάποια καλώδια σύνδεσης.



Εικόνα 12: PCB Board

Εικόνα 13: Jumper Wires

4.2 Πίνακας Κόστους Υλικών

Ακολουθεί ο πίνακας με το κόστος των υλικών που χρησιμοποιήθηκαν.

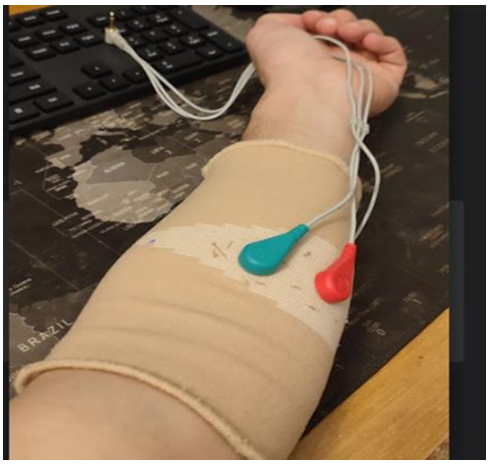
Πίνακας 1: Κόστος υλικών Οκτώβριος 2024

Υλικά	Κόστος κατά προσέγγιση
Μικροελεγκτής Wemos D1 Mini	12 €
AD8232 Module	12 €
Επίδεσμος	10 €

Battery Shield	6 €
Ηλεκτρόδια	5€
Μπαταρία LiPo 3.7 V 600mAh	6 €
Καλώδια Σύνδεσης	3 €
Πλακέτα PCB	1 €

4.3 Συναρμολόγηση

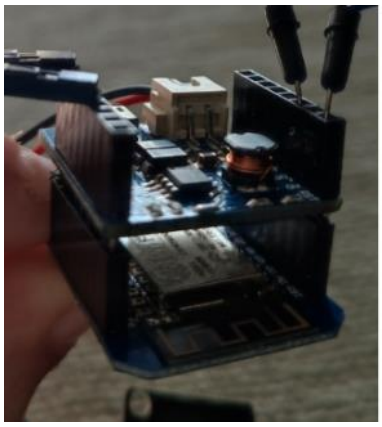
Αρχικά, εξετάστηκε η λύση του 3D Printing, καθώς προσέφερε μεγάλη ευελιξία στον σχεδιασμό. Ωστόσο, η πολυπλοκότητα του το κατέστησε ακατάλληλο. Έτσι μόλις παρέλαβα τα υλικά ξεκίνησα να τροποποιώ τον επίδεσμο με τρόπο τέτοιον ώστε να αγκαλιάζει τον μυ που ήθελα να στοχεύσω. Ξεκίνησα κάνοντας σχισμές επάνω στον επίδεσμο και περνώντας τα ηλεκτρόδια ανάποδα. Αυτή η προσέγγιση ήταν ιδανική, καθώς ο επίδεσμος ήταν αρκετά μεγάλος και κάλυπτε όλη την επιφάνεια του μυ, δίνοντάς μου τη δυνατότητα να τοποθετήσω τα ηλεκτρόδια σε οποιαδήποτε θέση επιθυμούσα. Επίσης ήταν κατάλληλος για χρήση από ανθρώπους με διαφορετική σωματοδομή. Στην συνέχεια έραψα τα ηλεκτρόδια κάνοντας τα πιο σταθερά.



Εικόνα 14: Αυτοσχέδιος επίδεσμος μπροστινή όψη

Εικόνα 15: Αυτοσχέδιος επίδεσμος πίσω όψη

Στη διαδικασία συναρμολόγησης, ο μικροελεγκτής Wemos D1 Mini και το Battery Shield είναι σχεδιασμένα με τρόπο τέτοιο να προσαρμόζεται απευθείας το ένα πάνω στο άλλο χωρίς πρόσθετα καλώδια. Όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 16: Wemos & Battery Shield

Στη συνέχεια, έπρεπε να επιλέξουμε ποιες υποδοχές θα χρησιμοποιήσουμε έτσι ώστε να μπορέσουμε να προγραμματίσουμε τον μικροελεγκτή στην συνέχεια. Ο παρακάτω πίνακας δείχνει την αντιστοίχιση των επαφών του αισθητήρα με τα αντίστοιχα πινάκια του μικροελεγκτή.

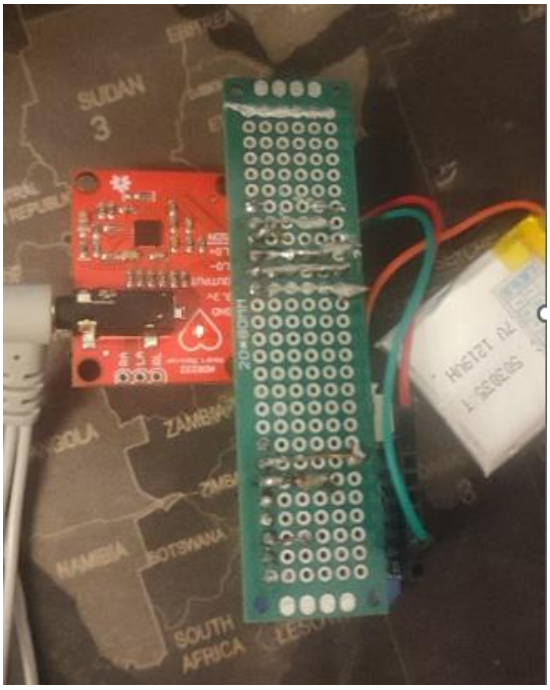
Πίνακας 2: Αντιστοίχιση επαφών AD8232 με Wemos.

AD8232	Wemos D1 Mini
L0+	D0
L0-	D1
OUTPUT	A0
GND	GND
3.3V	3.3V

Αφού έχουμε τελειώσει με τις συνδέσεις και έχοντας αποφασίσει την αντιστοίχιση, μπορούμε τώρα να αρχίσουμε να προγραμματίζουμε και να πειραματιστούμε με τον μικροελεγκτή.

΄Στις παρακάτω εικόνες αποτυπώνεται ο μικροελεγκτής μαζί με τις απαραίτητες συνδέσεις στα πλαίσια του πρωτοτύπου.

Εικόνα 17: Μικροελεγκτής(Πίσω όψη)



Εικόνα 18: Μικροελεγκτής (Μπροστινή Όψη)

Κεφάλαιο 5ο: Τεχνολογίες Ανοιχτού Λογισμικού

5.1 Εισαγωγή

5.1.1 Τι είναι το ανοιχτό λογισμικό;

Το ανοιχτό λογισμικό αποτελεί μια κατηγορία λογισμικού της οποίας ο κώδικας είναι διαθέσιμος στο κοινό, επιτρέποντας την ελεύθερη πρόσβαση, τροποποίηση και διανομή του.

Η φιλοσοφία του ανοιχτού λογισμικού βασίζεται στην ιδέα ότι η γνώση και η τεχνολογία πρέπει να είναι διαθέσιμες σε όλους, ώστε να μπορούν να προσαρμόζονται και να εξελίσσονται με τη συνεργασία της κοινότητας. Αυτό το μοντέλο ανάπτυξης προωθεί τη διαφάνεια, την ευελιξία και την καινοτομία, καθώς δίνει τη δυνατότητα στους χρήστες να συμμετέχουν ενεργά στη βελτίωση και τη διαμόρφωση του λογισμικού. [7]

Η χρήση του ανοιχτού λογισμικού παρέχει σημαντικά πλεονεκτήματα τόσο στους μεμονωμένους χρήστες όσο και στους οργανισμούς. Ένα από τα κύρια πλεονεκτήματά του είναι το χαμηλό κόστος, καθώς συχνά διατίθεται δωρεάν. Ένα άλλο βασικό χαρακτηριστικό είναι η ευελιξία που προσφέρει. Οι χρήστες μπορούν να τροποποιούν το λογισμικό στις με βάση τις δικές τους ανάγκες.

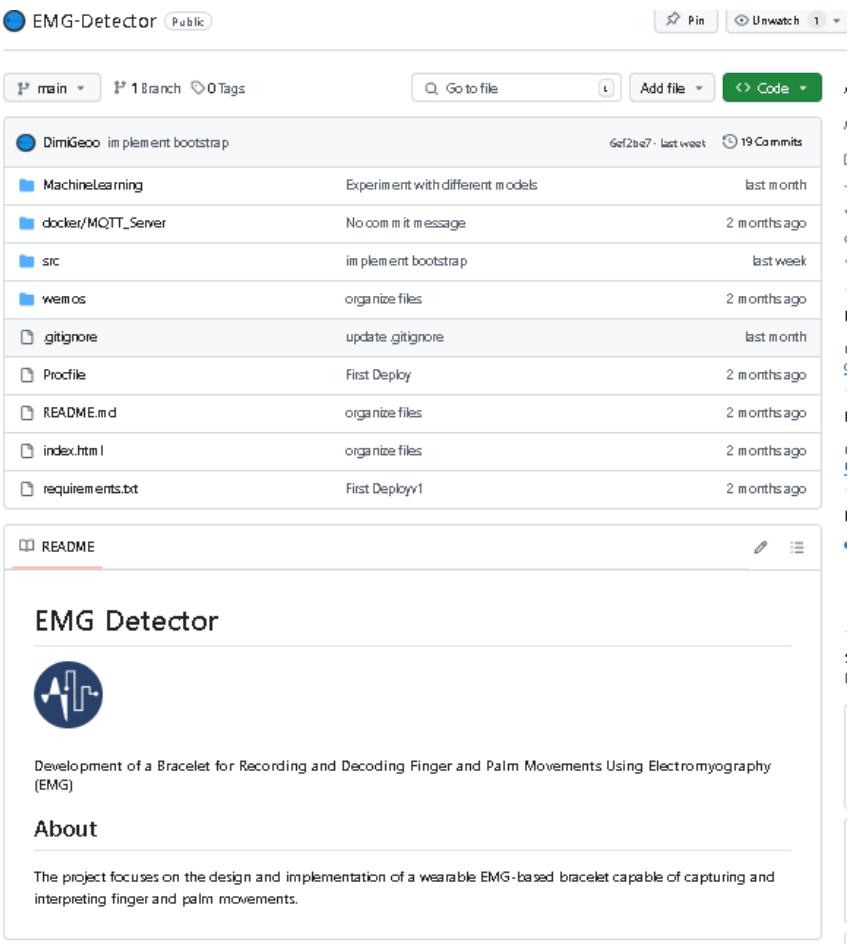
5.2 Τεχνολογίες που Χρησιμοποιήθηκαν

Σε αυτό το κεφάλαιο αναφέρονται επιγραμματικά οι τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση τις ΔΕ. Κάθε μία από αυτές συνέβαλε στην ανάπτυξη και στην αποτελεσματική λειτουργία του έξυπνου βραχιολιού.

5.2.1 Git

Το Git είναι ένα Version Control System που επιτρέπει την εύκολη διαχείριση του κώδικα, την παρακολούθηση αλλαγών σε βάθος χρόνου. Είναι ιδιαίτερα χρήσιμο όταν ένα project υλοποιείται από μεγάλες ομάδες. Μπορούμε να δημιουργούμε αντίγραφα ασφαλείας κάθε φόρα που αποθηκεύουμε την πρόοδο μας. Μπορούμε να εργαζόμαστε σε διαφορετικούς κλάδους (branches) και μόλις είναι έτοιμος ο κώδικας που υλοποιούμε να το εντάξουμε στον βασικό μας κλάδο (main), έπειτα από τον κατάλληλο έλεγχο. Μας επιτρέπει την ένωση (merge) διαφορετικών κλάδων, διευκολύνοντας την ενσωμάτωση των αλλαγών σε κάποιο άλλο σημείο του κώδικα. [8]

Στα πλαίσια της διπλωματικής μου εργασίας, αποφάσισα να χρησιμοποιήσω το GitHub για την παρακολούθηση των αλλαγών στον κώδικα και την καταγραφή της προόδου της εργασίας.



Εικόνα 19: Στιγμιότυπο από το αποθετήριο μου στο GitHub

5.2.2 Docker

Το Docker είναι μια τεχνολογία ανοιχτού κώδικα που χρησιμοποιείται για τη δημιουργία, ανάπτυξη και διαχείριση εφαρμογών μέσα σε απομονωμένα περιβάλλοντα εκτέλεσης (containers). Τα containers είναι ελαφριές και αυτόνομες μονάδες λογισμικού που περιέχουν όλα τα απαραίτητα αρχεία και ρυθμίσεις για την ομαλή εκτέλεση της εφαρμογής. [9]

Ασφάλεια & Απομόνωση: Κάθε container είναι απομονωμένο, αποτρέποντας συγκρούσεις εξαρτήσεων και αυξάνοντας την ασφάλεια.

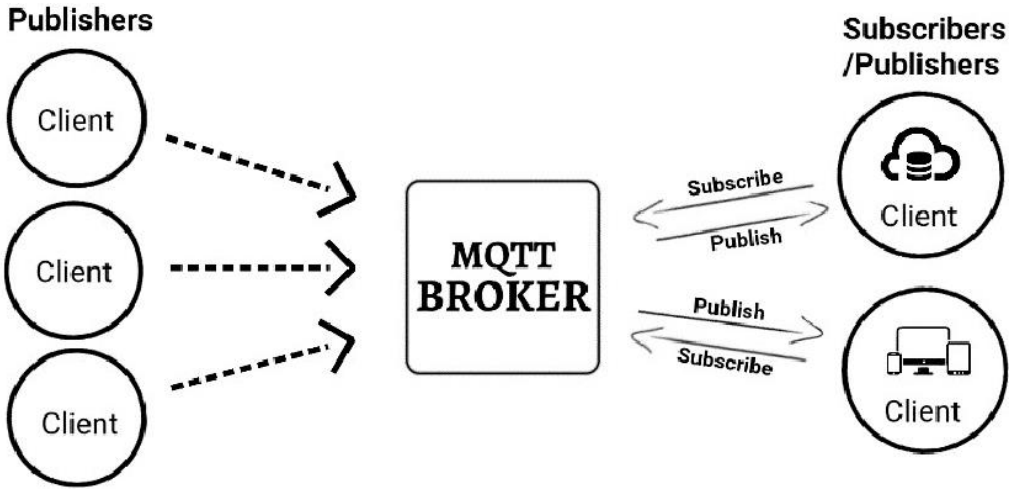
Περιβάλλον Εφαρμογής: Το περιβάλλον της εφαρμογής καθορίζεται μέσα στα αρχεία Dockerfile ή docker-compose.yml. Μέσα σε αυτά τα αρχεία δηλώνουμε όλες οι βιβλιοθήκες που χρησιμοποιούνται μαζί με τον αριθμό έκδοσης που θέλουμε να χρησιμοποιήσουμε στο project . Επίσης καθορίζονται τα δικαιώματα, μεταβλητές περιβάλλοντος και τυχόν συσχετίσεις με άλλα containers.

Απόδοση. Τα containers είναι ελαφριά και αποδοτικά, καθώς περιλαμβάνουν μόνο ό,τι χρειάζεται για την εκτέλεση της εφαρμογής. Μπορούν να εκκινούν, να σταματούν και να διαγράφονται εύκολα, ενώ ενεργοποιούνται μόνο όταν χρειάζεται, εξοικονομώντας πόρους σε σχέση με τα τοπικά περιβάλλοντα. Μπορούν να εκκινήσουν, να σταματήσουν και να διαγραφούν εύκολα με μία εντολή.

docker start | stop | rm container-name

5.2.3 MQTT Protocol

Το MQTT είναι ένα πρωτόκολλο επικοινωνίας που σχεδιάστηκε για περιπτώσεις όπου η αποστολή και λήψη δεδομένων πρέπει να γίνεται γρήγορα και με χαμηλή κατανάλωση. Η ελαφριά και αποδοτική αρχιτεκτονική του το καθιστά ιδανική επιλογή για εφαρμογές IoT. Το MQTT χρησιμοποιεί το πρότυπο publish/subscribe. [10]



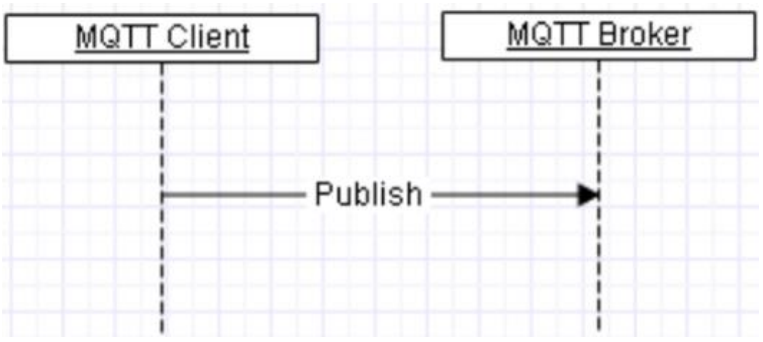
Εικόνα 20: Αρχιτεκτονική MQTT

Στο πρότυπο αυτό, οι χρήστες μπορούν να λειτουργούν ως αποστολέας (publishers), παραλήπτης (subscribers), ή και τα δύο. Ο ρόλος του διαμεσολαβητής (broker/server) είναι να λαμβάνει τα μηνύματα και να τα προωθεί στους παραλήπτες.

Ο αποστολέας δημιουργεί το μηνύματα, και το στέλνει στον broker. Ο παραλήπτης προκυμμένου να λάβει το μήνυμα θα πρέπει να έχει ήδη εκδήλωση το ενδιαφέρον του σε θέματα (topics) που τον ενδιαφέρουν. Κάθε μήνυμα συνοδεύεται από τις κεφαλίδες που απαιτεί το TCP πρωτόκολλο καθώς και κάποιες επιπλέον όπως θέμα (Topic) και την αξιοπιστία με την όποια θα παραδοθεί το μήνυμα(QoS).

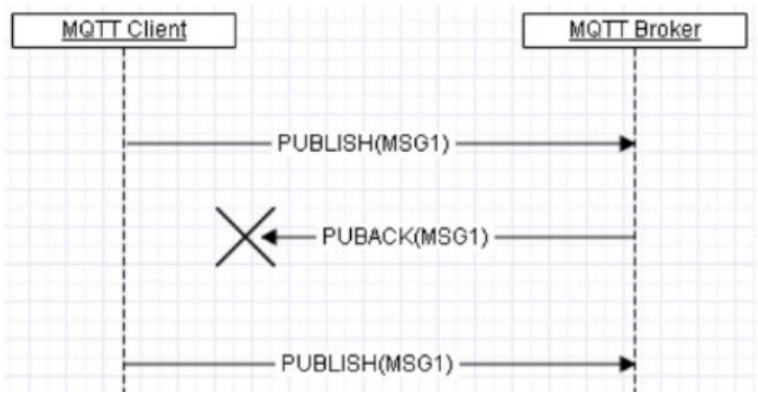
Η κεφαλίδα QoS μπορεί να πάρει 3 τιμές.

20. **QoS 0**: Το πακέτο αποστέλλεται χωρίς επιβεβαίωση και δεν ξαναστέλνεται αν χαθεί. Δεν υπάρχει εγγύηση για την παράδοση του μηνύματος.



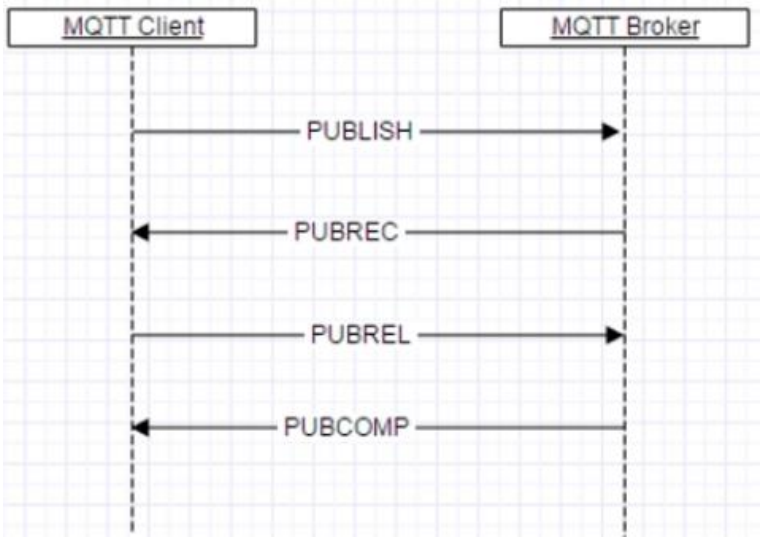
Εικόνα 21: MQTT QoS 0

21. **QoS 1**: Το πακέτο παραδίδεται τουλάχιστον μία φορά και ο αποστολέας περιμένει επιβεβαίωση . Η παράδοση του μηνύματος είναι εγγυημένη, αλλά μπορεί να παραληφθεί περισσότερες από μία φορές.



Εικόνα 22: MQTT QoS 1

22. **QoS 2**: Το πακέτο παραδίδεται ακριβώς μία φορά με την υψηλότερη αξιοπιστία, αλλά με μεγαλύτερο κόστος σε πόρους και καθυστέρηση.



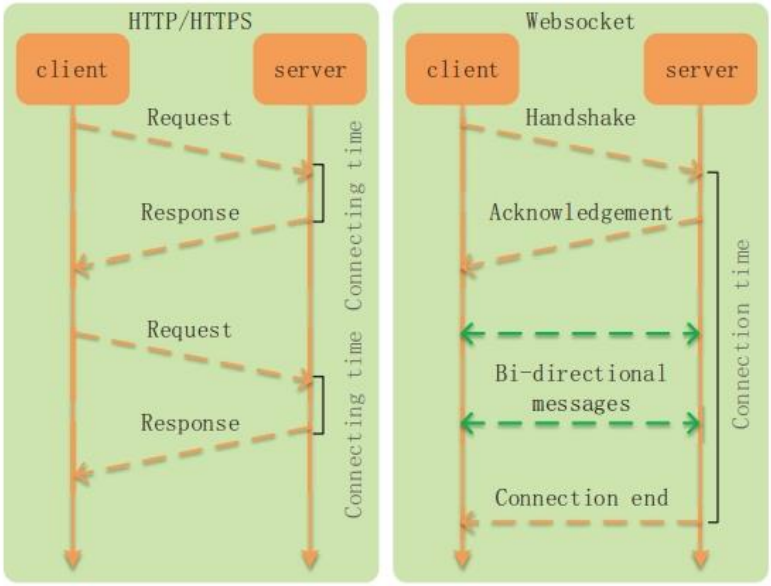
Εικόνα 23: MQTT QoS 2

5.2.4 WebSocket

Το WebSocket είναι ένα πρωτόκολλο επικοινωνίας που στηρίζεται στο TCP και HTTP. Ωστόσο, σε αντίθεση με το HTTP, η σύνδεση WebSocket επιτρέπει την αμφίδρομη επικοινωνία μεταξύ client και server χωρίς περιορισμούς. Η μόνη προϋπόθεση είναι να δημιουργηθεί μία αρχική σύνδεση, η οποία παραμένει ανοιχτή κατά την διάρκεια της επικοινωνίας. Η επικοινωνία με το WebSocket μειώνει τον χρόνο που απαιτείται για τη δημιουργία νέων συνδέσεων, εξοικονομεί πόρους, επιτρέποντας την άμεση και συνεχή ανταλλαγή δεδομένων. [11]

Τα βήματα για την επικοινωνία μέσω WebSockets είναι τα εξής:

1. **Αρχικοποίηση**: Ο πελάτης τη διαδικασία δημιουργώντας μια σύνδεση.
2. **Server**: Ο διακομιστής απαντά στο αίτημα και ολοκληρώνει τη σύνδεση.
3. **Επικοινωνία**: Μπορούν να στέλνονται δεδομένα και από τις δυο μεριές
4. **Τερματισμός Σύνδεσης**: Η σύνδεση τερματίζεται.



Εικόνα 24: Σύγκριση HTTP με WebSocket [13]

5.2.5 HTTP

Το HTTP είναι ένα πρωτόκολλο που χρησιμοποιείται για τη μεταφορά δεδομένων στο διαδίκτυο. Επιτρέπει στους περιηγητές ιστού να επικοινωνούν με διακομιστές και λειτουργεί με το μοντέλο Request–Response. [12].

5.2.6 Serial Protocol

Η σειριακή επικοινωνία είναι μια διαδικασία μετάδοσης δεδομένων, όπου τα bits μεταφέρονται διαδοχικά, το ένα μετά το άλλο, μέσω ενός μόνο καναλιού. [13]

5.2.7 Χρήση Python και FastAPI

Με τη χρήση της γλώσσας προγραμματισμού Python και του framework FastAPI, το οποίο χρησιμοποιήθηκε για την ανάπτυξη της λειτουργικότητας τόσο του backend όσο και του frontend, πραγματοποιήθηκαν δοκιμές και πειράματα σε διάφορα σημεία της εργασίας, με τον κώδικα να είναι διαθέσιμος στο GitHub. [14, 15, 16]

5.2.8 Βιβλιοθήκες Python

Στην εργασία χρησιμοποιούνται αρκετές βιβλιοθήκες της Python, οι οποίες διευκολύνουν τη διαδικασία ανάπτυξης και υλοποίησης της εφαρμογής.

Κάθε βιβλιοθήκη έχει το δικό της ρόλο και χρησιμοποιείται για να καλύψει συγκεκριμένες ανάγκες τις εργασίας.

1. **Scikit-learn (sklearn)**: Βιβλιοθήκη για εκπαίδευση μοντέλων μηχανικής μάθησης και ανάλυση δεδομένων.
2. **FastAPI**: Γρήγορη και εύκολη βιβλιοθήκη για την ανάπτυξη web APIs. Επιτρέπει την άμεση ενσωμάτωσή της με άλλες υπηρεσίες.
3. **Paho-MQTT**: Επιτρέπει την επικοινωνία με το πρωτόκολλο MQTT.
4. **Serial**: Επιτρέπει τη σειριακή επικοινωνία.
5. **Joblib**: Χρησιμοποιείται για την αποθήκευση και ανάκτηση των αλγορίθμων μηχανικής μάθησης.
6. **os**: Χρησιμοποιήθηκε για αποθήκευση αρχείων και ανάγνωση των μεταβλητων περιβάλλοντος.
7. **Numpy**: Παρέχει εργαλεία για αριθμητικούς υπολογισμούς και διαχείριση πινάκων.
8. **Pandas**: Εργαλεία για επεξεργασία και ανάλυση δεδομένων σε μορφή πίνακα.
9. **Matplotlib**: Χρησιμοποιείται για τη δημιουργία γραφημάτων.

Οι παραπάνω βιβλιοθήκες χρησιμοποιήθηκαν στα πλαίσια αυτής της εργασίας για την υλοποίηση και την επέκταση των λειτουργιών της εφαρμογής.

5.3 Εργαλεία Ανάπτυξης Κώδικα:

5.3.1 Visual Studio Code

Το Visual Studio Code είναι ένας εργαλείο επεξεργασίας κώδικα που υποστηρίζει πολλές γλώσσες προγραμματισμού, Συγκεκριμένα χρησιμοποιείται για την ανάπτυξη του backend του frontend του αλλά και διαφόρων δοκιμών. [16]

5.3.2 Arduino IDE

Το Arduino IDE χρησιμοποιήθηκε για την ανάπτυξη και τον προγραμματισμό του μικροελεγκτή. Η μεταγλώττιση και η φόρτωση του κώδικα πραγματοποιούνται εύκολα μέσω του περιβάλλοντος ανάπτυξης. [17]

Για να προγραμματίσουμε τις πλακέτες ESP8266 απαιτείται η εγκατάσταση της βιβλιοθήκης **ESP8266 package**, η οποία δεν περιλαμβάνεται από προεπιλογή.

Για να προσθέσουμε αυτή τη βιβλιοθήκη μέσω του **Boards Manager**, θα πρέπει να ακολουθούμε τα παρακάτω βήματα:

1. Ανοίγουμε το **Arduino IDE**.
2. Επιλέγουμε το μενού **File > Preferences**.
3. Στο πεδίο **Additional Boards Manager URLs**, προσθέτουμε την εξής διεύθυνση: https://arduino.esp8266.com/stable/package_esp8266com_index.json
4. Κλείνουμε το παράθυρο και πηγαίνουμε στο μενού **Tools > Board > Boards Manager**.
5. Αναζητούμε το **esp8266** και, όταν το βρούμε, επιλέγουμε την εγκατάσταση.

Με αυτόν τον τρόπο μπορούμε να προσθέσουμε και άλλες βιβλιοθήκες στο Arduino IDE επεκτείνοντας έτσι τις δυνατότητες του.

5.3.3 Εγκατάσταση Απαραίτητων Εργαλείων

Σε αυτή την ενότητα περιγράφεται η διαδικασία εγκατάστασης των απαραίτητων εργαλείων ανάπτυξης για το έργο, όπως Visual Studio Code, Docker, Python και Git, χρησιμοποιώντας τον διαχειριστή πακέτων Chocolatey. Με τη χρήση μιας μόνο εντολής, μπορούμε να εγκαταστήσουμε όλα τα απαραίτητα εργαλεία ταυτόχρονα, εξοικονομώντας χρόνο και απλοποιώντας τη διαδικασία εγκατάστασης. Η μοναδική προϋπόθεση είναι να έχουμε ήδη εγκαταστήσει το **Chocolatey** στον υπολογιστή μας.

Η εντολή που χρησιμοποιούμε για την εγκατάσταση των εργαλείων είναι η εξής:

```
choco install arduino vscode docker-desktop python git -y
```

Τώρα έχοντας εγκαταστήσει την γλώσσα προγραμματισμού python μπορούμε να εγκαταστήσουμε και τις απαραίτητες βιβλιοθήκες. Οι όποιές βρίσκονται στο αρχείο requirements.txt

- git clone https://github.com/DimiGeoo/EMG-Detector
- pip -r install EMG-Detector/requirements.txt

Με αυτά τα βήματα, έχουμε εγκαταστήσει όλα τα απαραίτητα εργαλεία και βιβλιοθήκες, και μπορούμε να προχωρήσουμε στην ανάπτυξη του έργου με το πλήρες περιβάλλον ανάπτυξης.

Κεφάλαιο 6ο: Ανάπτυξη Backend

6.1 Εισαγωγή

Η ανάπτυξη του backend αποτελεί το πιο κρίσιμο κομμάτι του συστήματος μας, καθώς γίνεται η κατηγοριοποίηση. Το μοντέλο μηχανικής μάθησης που θα χρησιμοποιήσουμε σε αυτό το κεφάλαιο είναι ενδεικτικό. Περισσότερες πληροφορίες για το τελικό μοντέλο θα παρουσιαστούν στο 9ο κεφάλαιο. Αρχικά οι server μας θα πρέπει να υλοποιηθούν με τρόπο τέτοιον ώστε να μας δίνεται εύκολα η δυνατότητα να κάνουμε αλλαγές στο μέλλον. Η ανάπτυξη του backend βασίστηκε εξολοκλήρου στη σχεδίαση που έγινε στο 3ο κεφάλαιο.

6.2 Επιλογή πρωτόκολλων επικοινωνίας

6.2.1 Επικοινωνία Microcontroller → Backend

Η πρώτη επιλογή για την επικοινωνία μεταξύ του μικροελεγκτή και του backend ήταν το πρωτόκολλο HTTP. Ωστόσο, οι καθυστερήσεις που παρουσιάζει καθιστά το πρωτόκολλο ακατάλληλο για εφαρμογές πραγματικού χρόνου. Επομένως, η επιλογή του MQTT φαινόταν να είναι μονόδρομος, καθώς προσφέρει χαμηλή καθυστέρηση και είναι ιδανικό για εφαρμογές πραγματικού χρόνου.

Αρχικά χρησιμοποίησα τον διακομιστή HiveMQ έναν διαμεσολαβητή (broker) που βασίζεται στο πρωτόκολλο MQTT. Δεδομένου ότι υπήρχαν κάποιοι περιορισμοί ως προς τις δωρεάν υπηρεσίες που θα μπορούσα να χρησιμοποιήσω και τον χρόνο που ήθελε ένα πακέτο να φτάσει στους διακομιστές τις πλατφόρμας. Οι περιορισμοί αυτοί δεν μου επέτρεπαν να συνεχίσω με αυτήν την επιλογή.

Εξέτασα την χρήση του πρωτοκόλλου Serial, το οποίο θα μπορούσε να προσφέρει τις καλύτερες δυνατές επιδόσεις. Ωστόσο, η ανάγκη για ασύρματη επικοινωνίας δεν μου επέτρεψε να συνεχίσω.

Το αποτέλεσμα ήταν η εγκατάσταση του MQTT Server, μειώνοντας την απόσταση μεταφοράς των δεδομένων μέσα στο τοπικό δίκτυο LAN. Αυτό μου επέτρεψε να έχω πλήρη έλεγχο του Backend.

6.2.2 Επικοινωνία Backend → Frontend

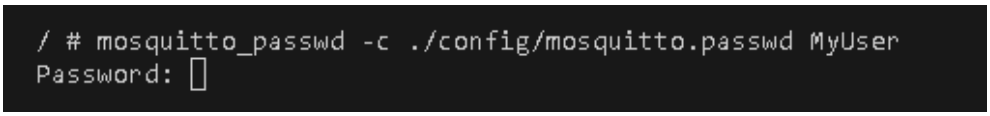
Γνωρίζοντας ότι το frontend θα έχει την μορφή μιας ιστοσελίδας θα έπρεπε να βρω έναν τρόπο δυναμικής ενημερώσεις, χωρίς την ανάγκη επαναφόρτωσης τις ιστοσελίδας. Εξετάστηκαν οι επιλογές AJAX και WebSocket, με το WebSocket να ξεχωρίζει λόγω της απόδοσης.

6.3 Υπηρεσίες του Backend

Αναλύεται η λειτουργία των υπηρεσιών του backend που αναπτύχθηκαν για την υποστήριξη της επικοινωνίας και της κατηγοριοποίησης.

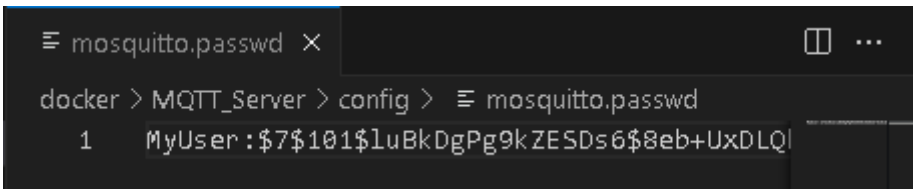
6.3.1 MQTT Server

Η εγκατάσταση του eclipse mosquitto έγινε με την βοήθεια του Docker κάποιες βασικές παραμετροποιήσεις που θα έπρεπε να κάνω ήταν να αποτρέψω την ανώνυμη πρόσβαση. Δημιούργησα έναν προεγγεγραμμένο χρήστη με τη χρήση του εργαλείου mosquitto_passwd, και αποθήκευσα τα δεδομένα

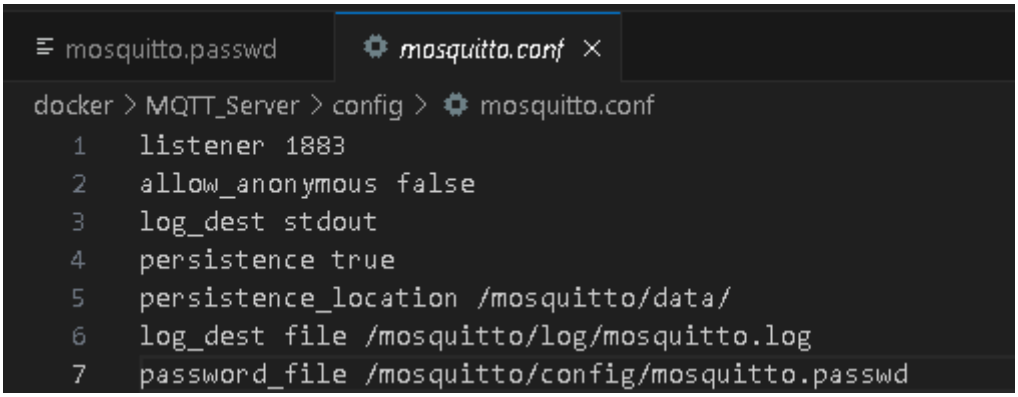


Εικόνα 25: Δημιουργία Χρήστη

Μόλις ολοκληρώσουμε την δημιουργία του χρήστη θα δημιουργηθεί ένα αρχείο που περιέχει ευαίσθητα δεδομένα όπως username,κωδικός πρόσβασης



Εικόνα 26: Κρυπτογραφημένο αρχείο passwd



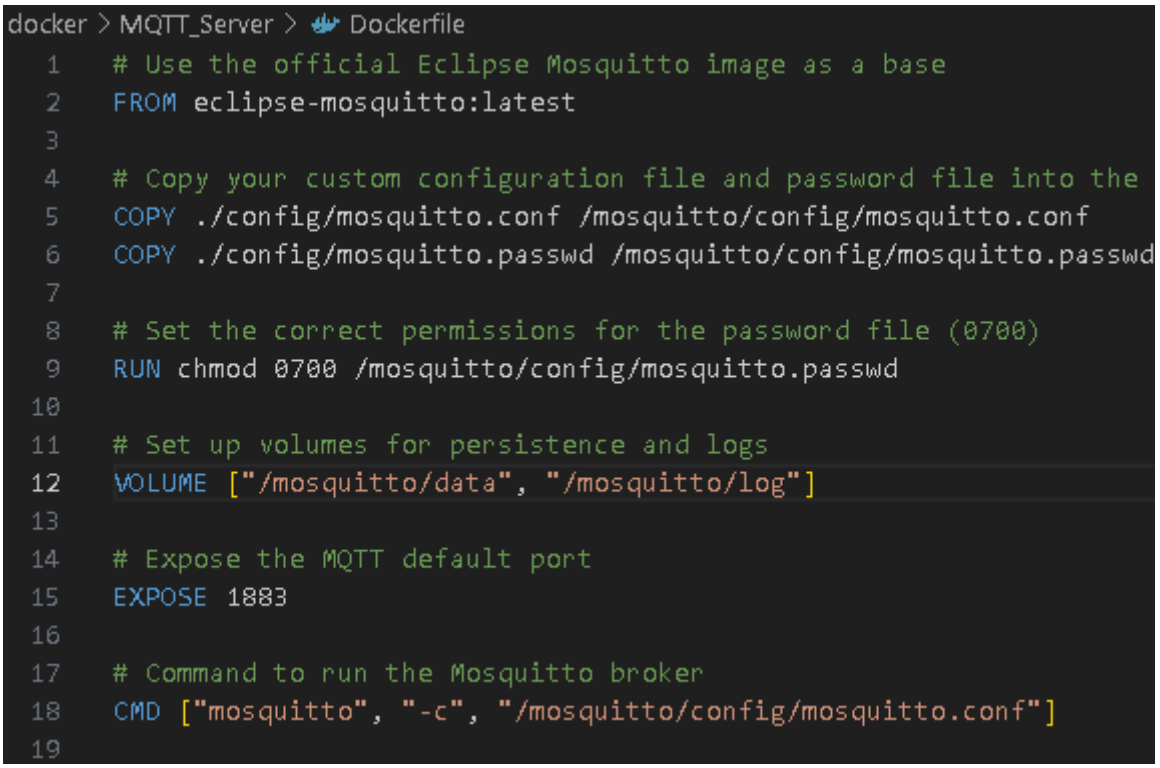
Εικόνα 27: Configuration File

Το αρχείο ρυθμίσεων **mosquitto.conf** μας δίνει την δυνατότητα να παραμετροποιήσουμε τον Server.

Στο αρχείο ρυθμίζουμε:

- Ορίζουμε το Port
- Αποτρέπουμε την ανώνυμη σύνδεση
- Ρυθμίσεις για Logfile καθώς και αποθήκευση των δεδομένων.

Σε αυτό το σημείο θα δημιουργήσουμε το Dockerfile. Σε αυτό το αρχείο θα πρέπει να περιλαμβάνονται όλες οι απαραίτητες ρυθμίσεις, βιβλιοθήκες και εξαρτήματα που απαιτούνται για την κατασκευή του Docker image.



Εικόνα 28: Dockerfile

Αφού ολοκληρώσουμε τις ρυθμίσεις, το επόμενο βήμα είναι να δημιουργήσουμε το **Docker image** και να εκκινήσουμε το **container**. Αυτό πραγματοποιείται με την εξής εντολή:

1. docker build -t emg_mqtt .

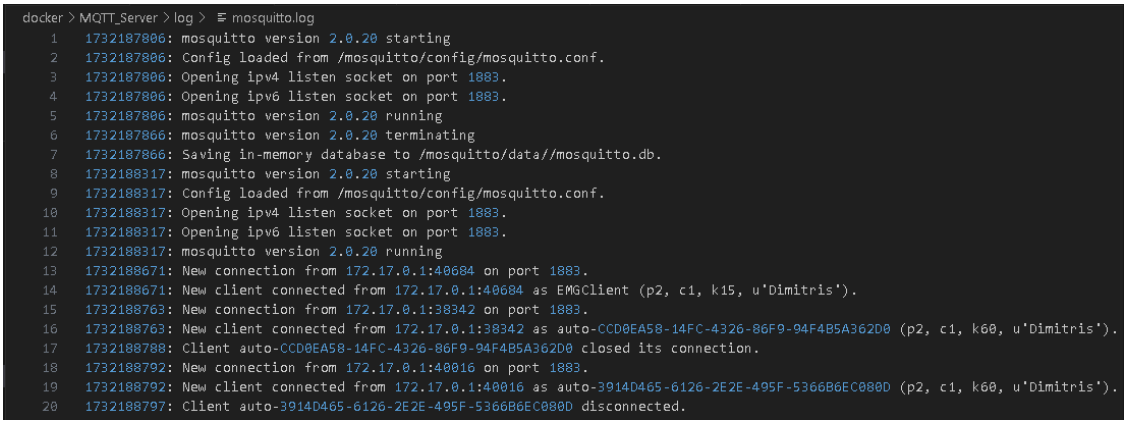
2. docker run -d --name emg_mqtt_container `
   -p 1883:1883 `
   v ${PWD}\data:/mosquitto/data `
   -v ${PWD}\log:/mosquitto/log `

Σε αυτό το στάδιο, καθορίζουμε τις διαδρομές τόσο εντός του container όσο και στον υπολογιστή μας για την αποθήκευση του **logfile** και του αρχείου **db** που περιέχει το ιστορικό των μηνυμάτων.

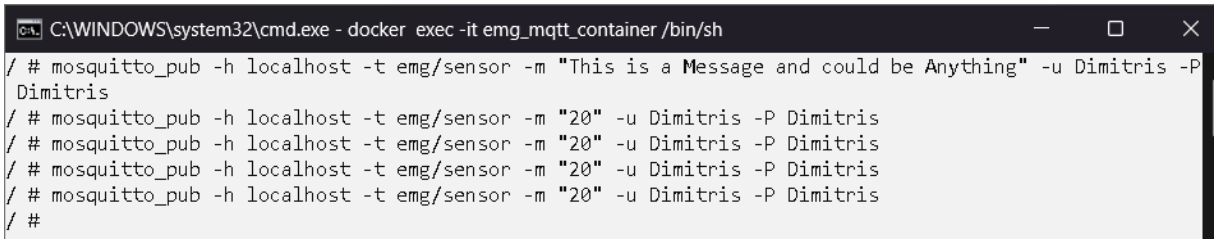
Από αυτό το σημείο και μετά, η διαχείριση του container γίνεται εύκολη και άμεση

- Εκκίνηση του container: docker start emg_mqtt_container.
- Διακοπή του container: docker stop emg_mqtt_container.

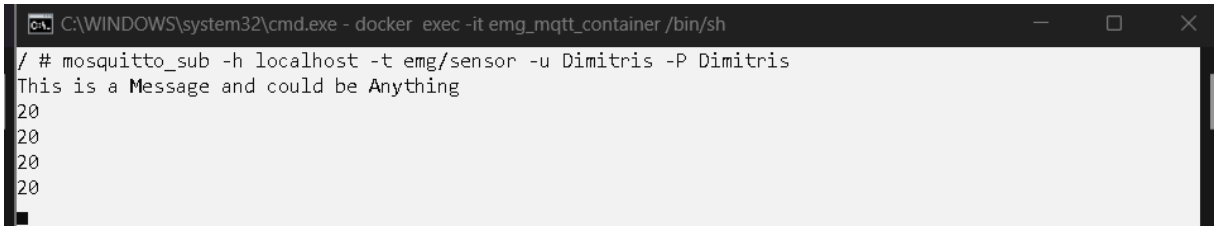
Στις παρακάτω εικόνες παρουσιάζεται το αρχείο καταγραφής (log file) του Server καθώς και παραδείγματα επικοινωνίας μέσω την γραμμή εντολών:



Εικόνα 29: Αρχείο καταγραφής του MQTT Server



Εικόνα 30: Παράδειγμα MQTT Publish



Εικόνα 31: Παράδειγμα MQTT Subscribe

6.3.2 Web Server & Κατηγοριοποίηση

Ο Web Server έχει σχεδιαστεί για να εξασφαλίζει την αποτελεσματική επικοινωνία μεταξύ του frontend και του backend, ενώ ταυτόχρονα υποστηρίζει την κατηγοριοποίηση των δεδομένων EMG.

Βασικές Λειτουργίες

- **Λήψη δεδομένων**
- **Επεξεργασία Δεδομένων**.
- **Κατηγοριοποίηση των δεδομένων**.
- **Υλοποίηση WebSocket**.

Η συγκεκριμένη υλοποίηση έχει γίνει με την βοήθεια του FastAPI

Για την διαδικασία της κατηγοριοποίησης θα αναφερθούμε αναλυτικότερα στο 9ο κεφάλαιο. Σε αυτό το κεφάλαιο θα ασχοληθούμε με την υλοποίηση. Έχουμε αποθηκευμένο το μοντέλο μας τοπικά και θέλουμε να το φορτώσουμε στον Server για να μπορεί να κατηγοριοποιήσει τα δεδομένα.

```python
# Load the model
def load_model(model_path: str, scaler_path: str) -> None:
    global model, scaler
    try:
        # Load the model
        model = joblib.load(model_path)
        print("Model loaded successfully.")

        # Load the scaler
        scaler = joblib.load(scaler_path)
        print("Scaler loaded successfully.")

    except Exception as e:
        print(f"Error loading model or scaler: {str(e)}")
```

Εικόνα 32: Φόρτωση Μοντέλου

Σημαντικό είναι να χρησιμοποιήσουμε τις ίδιες παραμέτρους και τα ίδια χαρακτηριστικά που εφαρμόστηκαν κατά την διαδικασία τις εκπαίδευσής. Σε κάθε σε κάθε άλλη περίπτωση θα λαμβάνουμε είτε κάποιο σφάλμα είτε λάθος κατηγοριοποίηση.

Αυτές οι παράμετροι είναι:

1. Ο ρυθμός δειγματοληψίας (Sampling Rate)
2. Το παράθυρο δεδομένων (Window Size)
3. Χαρακτηριστικά του dataset

Για τον υπολογισμό του πραγματικού ρυθμού αποστολής δεδομένων (Sampling Rate) δημιουργήθηκε ένα script που αποθηκεύτε τα μηνύματα για ην περίοδο ενός λεπτού και στην συνέχεια υπολογιζόταν ο ρυθμός σε ms.

```
# Print the number of messages received
print(f"\nMessage Received: {get_message_count()}")
print(f"\nDuration: {DURATION}")
print(f"Average : {(DURATION/get_message_count())*1000}ms\n")
```

Εικόνα 33: Ρυθμός δειγματοληψίας (Sampling Rate)

Δεδομένου ότι χρειαζόμαστε μια εφαρμογή που θα δουλεύει με δεδομένα πραγματικού χρόνου υλοποιούμε προγραμματιστικά με τρόπο τέτοιο ώστε, αν δεν λάβουμε κανένα πακέτο σε διάστημα ενός δευτερόλεπτου θεωρούμε ότι δεν υπάρχει σύνδεση. Για να συμβεί αυτό σημαίνει ότι ίσως έχουν αποσυνδεθεί τα ηλεκτρόδια ή ότι διακόπηκε η σύνδεση. Όμως στο σημείο μας ενδιαφέρει μόνο ότι δεν έχουμε επαρκές δεδομένα για την σωστή κατηγοριοποίηση.

Τα δεδομένα αυτά τα αποθηκεύουμε στην μνήμη με την μορφή buffer μόλις το buffer γίνει ίσο με Window Size σημαίνει ότι είμαστε έτοιμοι να κάνουμε την κατηγοριοποίηση. Για να γίνει αυτό θα πρέπει πρώτα να εξάγουμε τα χαρακτηρίστηκα που απαιτούνται και να ρωτήσουμε το μοντέλο ποια είναι η εκτίμηση του, για τα συγκεκριμένο σύνολο δεδομένων.

```
def classify_data(buffer) -> Any:
    features = extract_features(buffer)
    prediction = model.predict([features][0])
    return prediction
```

Εικόνα 34: Συνάρτηση κατηγοριοποίησης

Τα αποτελέσματα θα αποθηκευτούν στην μεταβλητή latest_classification καις την συνέχεια θα τροποποιούν έτσι ώστε να ταιριάζουν στην δομή που θέλουμε να έχει το WebSocket.

```python
# WebSocket endpoint
@app.websocket("/ws")
async def websocket_endpoint(websocket: WebSocket) -> None:
    await websocket.accept()
    try:
        while True:
            # Check if new data has been received
            if (mqtt_data_received and time.time() - last_message_time <= MESSAGE_TIMEOUT):
                status = {"connected": True, "classification": latest_classification}
            else:
                status = {
                    "connected": False,
                    "classification": {"name": " ", "image": " "},
                }
            await websocket.send_json(status)
            await asyncio.sleep(0.25)  # Small delay
    except Exception as e:
        print(f"WebSocket error: {e}")
    finally:
        await websocket.close()
```

Εικόνα 35: Υλοποίηση WebSocket

Μετά την αρχική αποδοχή της σύνδεσης, ο Server παρακολουθεί αν υπάρχουν νέα δεδομένα και στέλνει περιοδικά το μήνυμα με την μορφή που έχουμε επιλέξει στο 3ο κεφάλαιο. Η διαδικασία είναι σχεδιασμένη για να διασφαλίζει την συνεχή επικοινωνία, ενώ σε περιπτώσεις σφάλματος ο τερματισμός τις σύνδεσης γίνεται με ασφάλεια.

Τέλος για την δημιουργία του backend αρκεί η εκτέλεση της εντολής:

```
uvicorn src.main:app --host 0.0.0.0 --port 8000
```

Η εντολή αυτή εκκινεί τον server υλοποιώντας τις υπηρεσίες που έχουν αναφερθεί σε αυτό το κεφάλαιο, επιτρέποντας την πρόσβαση από οποιαδήποτε συσκευή του τοπικού μας δικτύου. Στην παρακάτω εικόνα ακολουθεί ένα παράδειγμα εκτέλεσης της εντολής.

```
PS D:\GitHub\git\Personal\EMG-Detector> uvicorn src.main:app --host 0.0.0.0 --port 8000
Model loaded successfully.
Scaler loaded successfully.
INFO:     Started server process [17060]
INFO:     Waiting for application startup.
MQTT Connected
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     127.0.0.1:32555 - "GET / HTTP/1.1" 200 OK
INFO:     ('127.0.0.1', 32559) - "WebSocket /ws" [accepted]
INFO:     connection open
```

Εικόνα 36: Παράδειγμα εκτέλεσης Webserver

Ο πλήρης κώδικας του Backend περιλαμβάνεται στο Παράρτημα Β. Ενώ χρήσιμες οδηγίες υπάρχουν και στο GitHub.

6.4 Επεξήγηση του κώδικα

Η εφαρμογή υλοποιείται με την χρήση FastAPI, MQTT και WebSocket για τη λήψη δεδομένων, την επεξεργασία και την παρουσίαση αποτελεσμάτων σε πραγματικό χρόνο.

Φόρτωση Μοντέλου: Στην αρχή της εφαρμογής, το εκπαιδευμένο μοντέλο μηχανικής μάθησης και ο scaler φορτώνονται από τα αρχεία τους μέσω της βιβλιοθήκης joblib. Αυτό εξασφαλίζει ότι το μοντέλο είναι έτοιμο για να κατηγοριοποιεί τα δεδομένα που λαμβάνονται μέσω του MQTT.

```python
# Load the model
def load_model(model_path: str, scaler_path: str) -> None:
    global model, scaler
    try:
        # Load the model
        model = joblib.load(model_path)
        print("Model loaded successfully.")
        # Load the scaler
        scaler = joblib.load(scaler_path)
        print("Scaler loaded successfully.")
    except Exception as e:
        print(f"Error loading model or scaler: {str(e)}")
```

Εικόνα 37: Φόρτωση μοντέλου

Λήψη μηνυμάτων MQTT: Όταν τα δεδομένα αποστέλλονται μέσω MQTT, η συνάρτηση on_message τα αποθηκεύει σε μια μεταβλητή. Όταν το μέγεθος τις μεταβλητής γίνει ίσο με το μέγεθος του window size περνάνε από τη διαδικασία εξαγωγής χαρακτηριστικών και κατηγοριοποιούνται μέσω του μοντέλου μηχανικής μάθησης. Το αποτέλεσμα αποθηκεύεται στην μορφή που έχουμε ήδη σχεδιάσει και είναι έτοιμο να σταλεί με το πρωτόκολλο WebSocket.

```python
def on_message(client, userdata, message) -> None:
    global latest_classification, mqtt_data_received, last_message_time, buffer
    last_message_time = time.time()
    mqtt_data_received = True
    # Decode the incoming message and convert it to an integer
    data = int(message.payload.decode("utf-8"))

    buffer.append(data)

    if len(buffer) > buffer_size:
        # Perform classification when buffer is full
        prediction = classify_data(buffer)
        latest_classification = map_class_to_info(prediction)
        # Clear the buffer after classification
        buffer.clear()
        print(f"Updated classification: {latest_classification}")
```

Εικόνα 38 Λήψη μηνυμάτων MQTT

Ο κώδικας του WebSocket ανοίγει μια σύνδεση μεταξύ του server και του πελάτη, επιτρέποντας την αμφίδρομη επικοινωνία σε πραγματικό χρόνο. Όταν ο πελάτης συνδέεται, ο server στέλνει συνεχώς τα αποτελέσματα της κατηγοριοποίησης.

```python
# WebSocket endpoint
@app.websocket("/ws")
async def websocket_endpoint(websocket: WebSocket) -> None:
    await websocket.accept()···
            if (
                mqtt_data_received
                and time.time() - last_message_time <= MESSAGE_TIMEOUT
            ):
                status = {"connected": True, "classification": latest_classification}
            else:
                status = {
                    "connected": False,
                    "classification": {"name": " ", "image": " "},
                }
            await websocket.send_json(status)
            await asyncio.sleep(0.25)  # Small delay
    except Exception as e:
        print(f"WebSocket error: {e}")
    finally:
        await websocket.close()
```

Εικόνα 39 Υλοποίηση WebSocket

Webserver: Ο Web Server στον κώδικα χειρίζεται τα αιτήματα. Όταν ο χρήστης επισκέπτεται τη σελίδα του server, ο server στέλνει την HTML σελίδα που βρίσκεται σε προκαθορισμένο αρχείο και κάνει την ιστοσελίδα πρόσβαση σε όλους τους χρήστες του τοπικού δικτύου.

```python
@app.get("/")
async def get() -> HTMLResponse:
    return HTMLResponse(content=read_html_content(html_file_path))
```

Εικόνα 40 Αίτημα Request (/)

Κεφάλαιο 7ο: Ανάπτυξη Κώδικα Μικροελεγκτή

Σε αυτό το κεφάλαιο παρουσιάζεται η διαδικασία προγραμματισμού του μικροελεγκτή. Ο προγραμματισμός ακολουθεί τα παρακάτω βήματα, σύνδεση του μικροελεγκτή στο δίκτυο Wi-Fi, την επικοινωνία με τον MQTT Server και την αποστολή των δεδομένων από τον αισθητήρα στον Server με τη σωστή μορφή. Ο κώδικας μεταγλωττίστηκε και φορτώθηκε στον μικροελεγκτή μέσω του περιβάλλοντος Arduino IDE

A. Αρχικοποίηση συστήματος: Ο μικροελεγκτής αρχικοποιείται, με τις απαραίτητες μεταβλητές Ενώ οι συνδέσεις γίνονται με τον τρόπο που σχεδιαστήκαν αρχικά στο 3ο κεφάλαιο.

```
// Wi-Fi credentials
const char* ssid = "D*******";
const char* password = "D*******";

// MQTT Broker details
const char* mqttServer = "192.168.1.5";  // IP Address of the MQTT Broker
const int mqttPort = 1883;
const char* mqttUser = "D*******";
const char* mqttPassword = "D*******";

const int detect_plus = D0;
const int detect_minus = D1;

// Topic
const char* mqttTopic = "emg/sensor";

// EMG Sensor Pin
const int emgPin = A0;
```

Εικόνα 41:Αρχικοποίηση Μεταβλητών

B. Σύνδεση στο δίκτυο Wi-Fi: Ο μικροελεγκτής συνδέεται στο τοπικό δίκτυο Wi-Fi με τις μεταβλητές (ssid και password). Η διαδικασία σύνδεσης περιλαμβάνει κάποια βήματα επαλήθευσης, έτσι ώστε να διασφαλιστεί η επιτυχής πρόσβαση στο δίκτυο, και να εμφανιστεί το ανάλογο μήνυμα.

```
// Connect to Wi-Fi
WiFi.begin(ssid, password);
Serial.print("Connecting to Wi-Fi");
while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.print(".");
}
Serial.println("\nConnected to Wi-Fi");
```

Εικόνα 42: Σύνδεση στο Wi-Fi

C. Σύνδεση με τον MQTT Broker: Μόλις συνδεθούμε στο internet μπορούμε να συνδεθούμε στον MQTT broker που θα πρέπει να βρίσκεται στο ίδιο τοπικό δίκτυο. Σε κάθε περίπτωση επιτυχίας ή αποτυχίας εμφανίζεται κάποιο μήνυμα στην κονσόλα.

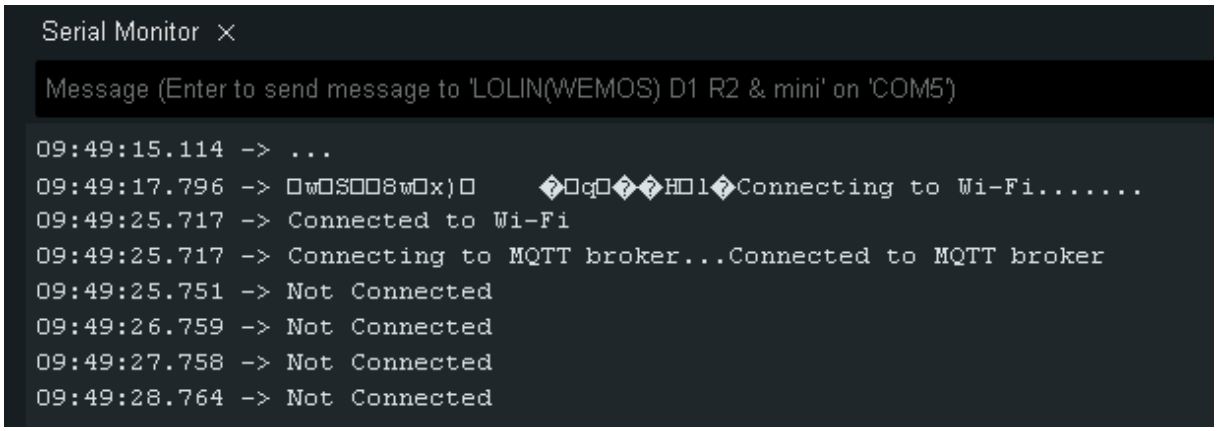
```
  // Connect to MQTT Broker
  client.setServer(mqttServer, mqttPort);
  while (!client.connected()) {
    Serial.print("Connecting to MQTT broker...");
    if (client.connect("EMGClient", mqttUser, mqttPassword)) {
      Serial.println("Connected to MQTT broker");
    } else {
      Serial.print("Failed, rc=");
      Serial.print(client.state());

      delay(2000);
    }
  }
}
```

Εικόνα 43: Σύνδεση στον MQTT Server

D. Αποστολή δεδομένων: Τελευταίο βήμα είναι να δημοσιεύσουμε (publish) τις τιμές που προέρχονται από το pin A0 που είναι η έξοδος του αισθητήρα. στον broker μέσω του MQTT πρωτοκόλλου. Η αποστολή πραγματοποιείται σε προκαθορισμένο topic emg/sensor. Επιπλέον, γίνεται έλεγχος της σύνδεσης του αισθητήρα, και σε περίπτωση που δεν έχει συνδεθεί σωστά εμφανίζεται το μήνυμα "Not Connected".

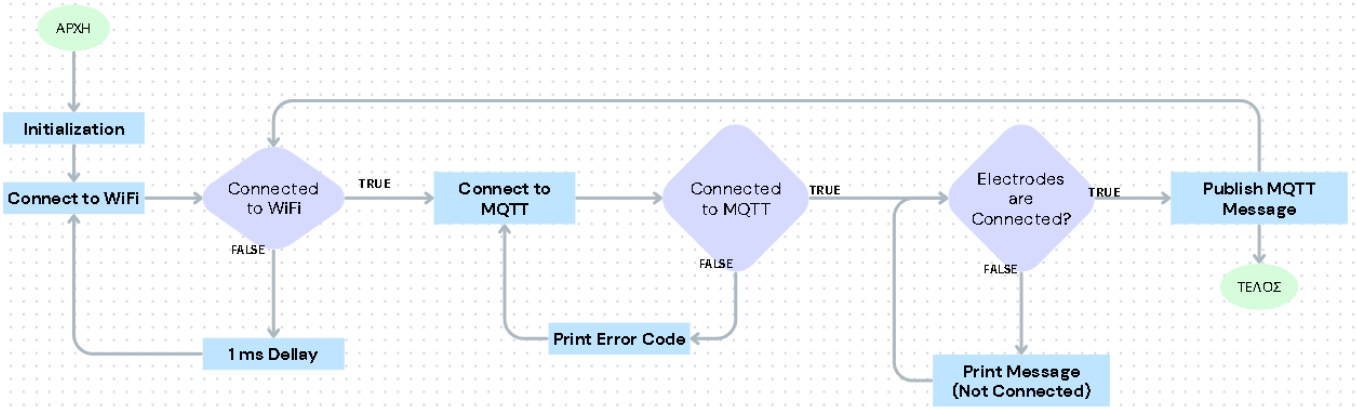
```
void loop() {
  // Ensure MQTT connection
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  if((digitalRead(detect_plus) == 1)||(digitalRead(detect_minus) == 1)){
    Serial.println("Not Connected");
    delay(1000);
  }
  else{
  // Send the value of Sensor to MQTT Broker
    client.publish(mqttTopic,  String(analogRead(emgPin)).c_str());
  }
}
```

Εικόνα 44: Κώδικας Μικροελεγκτή



Εικόνα 45 Serial Monitor

Σε περίπτωση αποσύνδεσης είτε από το Wi-Fi είτε από τον MQTT broker, η συσκευή προσπαθεί να επανασυνδεθεί αυτόματα, διασφαλίζοντας τη συνεχή λειτουργία του συστήματος. Ο κώδικας του μικροελεγκτή βρίσκεται εδώ, αναπτύχθηκε στο περιβάλλον Arduino IDE και η γλώσσα προγραμματισμού που χρησιμοποίησα είναι η C++.



Εικόνα 46 Διάγραμμά Ροής Μικροελεγκτή

Ενώ οι βιβλιοθήκες που χρησιμοποιήθηκαν ήταν :

ESP8266WiFi: Χρησιμοποιείται για να συνδέσει τον μικροελεγκτή στο δίκτυο Wi-Fi, επιτρέποντας την επικοινωνία μέσω του διαδικτύου.

PubSubClient: Επιτρέπει την επικοινωνία μέσω του πρωτοκόλλου MQTT, διευκολύνοντας την αποστολή και λήψη μηνυμάτων σε έναν MQTT broker.

Κεφάλαιο 8ο: Ανάπτυξη Fronted

Στο παρόν κεφάλαιο, εξετάζεται η ανάπτυξη του frontend της εφαρμογής, το οποίο υλοποιείται μέσω του FastAPI framework και παρέχει τη δυνατότητα παρακολούθησης των αποτελεσμάτων ταξινόμησης σε πραγματικό χρόνο. Χρησιμοποιώντας τεχνολογίες όπως WebSocket, HTML, CSS και JavaScript. Η επικοινωνία με το backend πραγματοποιείται μέσω του πρωτοκόλλου WebSocket, το οποίο υποστηρίζει αμφίδρομη επικοινωνία, επιτρέποντας στον Server να στέλνει δεδομένα στην ιστοσελίδα χωρίς την ανάγκη επαναφόρτισης της σελίδας.

Το WebSocket έχει ήδη υλοποιηθεί στο backend (6ο κεφάλαιο) και είναι διαθέσιμο στη διεύθυνση:

ws://localhost:8000/ws.

Τα δεδομένα που στέλνονται από το backend στο frontend έχουν την παρακάτω δομή:

```
{
  "connected":" true | false",
  "classification": {
    "name": "Gesture Name",
    "image": "http://example.com/icons.png"
  }
}
```

Εικόνα 47: Δομή του μηνύματος (WebSocket)

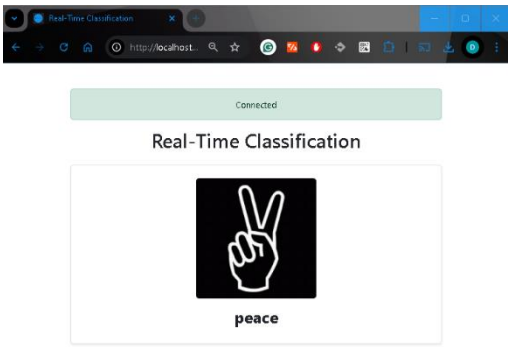
Όλη η λειτουργικότητα επιτυγχάνεται με την χρήση JavaScript. Πιο συγκεκριμένα οι συναρτήσεις updateStatus και updateUI είναι υπεύθυνες για την λειτουργικότητα.

```
<script>
    //   Websocket functionality is comming from backend
    const socket = new WebSocket("ws://localhost:8000/ws");
    let previousConnectedState = false;

    socket.onmessage = (event) => {
        const data = JSON.parse(event.data);
        updateStatus(data.connected);
        if (data.connected) {
            updateUI(data.classification);
        }
    };

    socket.onopen = () => {updateStatus(true);};
    socket.onclose = () => {updateStatus(false);};
    socket.onerror = (error) => {console.error("WebSocket error:", error);};

    function updateStatus(connected) {…

    function updateUI(classData) {…

</script>
```

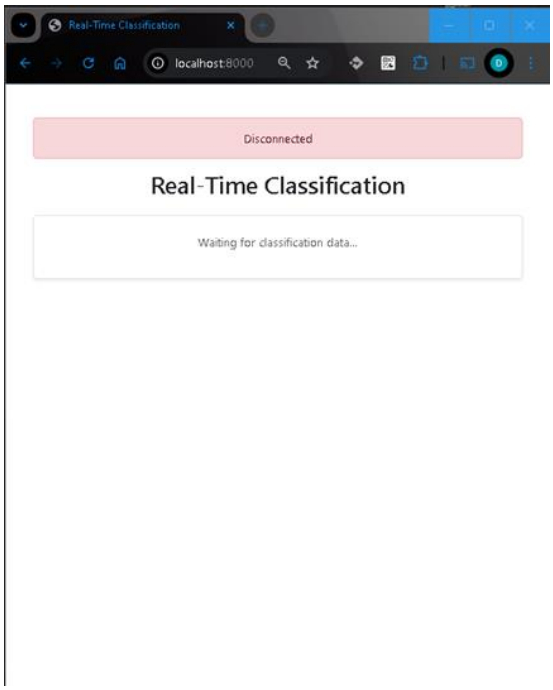
Εικόνα 48: Κώδικας JavaScript

- updateStatus: Ελέγχει αν η κατάσταση σύνδεσης είναι ενεργή ή όχι , Και ανανεώνει δυναμικά την διεπαφή. Σε περίπτωση που δεν έχουμε επικοινωνία κρύβεται το πεδίο των αποτελεσμάτων.
- updateUI: Η συνάρτηση αυτή ενημερώνει την ιστοσελίδα και εμφανίζει τα αποτελέσματα ταξινόμησης. Ελέγχει αν τα δεδομένα ταξινόμησης είναι έγκυρα διασφαλίζει ότι ο χρήστης βλέπει πάντα τα πιο πρόσφατα αποτελέσματα ταξινόμησης σε πραγματικό χρόνο.

Εικόνα 49: Κίνηση Peace



Εικόνα 50: Ιστοσελίδα όταν υπάρχουν δεδομένα.



Εικόνα 51: Ιστοσελίδα όταν δεν υπάρχουν δεδομένα

Με αυτόν τον τρόπο, παρουσιάζονται τα αποτελέσματα.

Η συγκεκριμένη διεπαφή δημιουργήθηκε με μοναδικό στόχο την οπτικοποίηση των αποτελεσμάτων, βελτιώνοντας σημαντικά την εμπειρία του χρήστη σε σχέση με την αρχική υλοποίηση μέσω γραμμής εντολών.

Κεφάλαιο 9ο: Μηχανική Μάθηση και Αξιολόγηση

9.1 Εισαγωγή στη Μηχανική Μάθηση

Στην ενότητα αυτή, αναλύονται οι αλγόριθμοι μηχανικής μάθησης που χρησιμοποιήσαμε καθώς και όλα τα βήματα που έγιναν για την κατηγοριοποίηση των δεδομένων (EMG).Από τις κινήσεις που επιλέχτηκαν, μέχρι και το τελικό βήμα που είναι η εκπαίδευση/αξιολόγηση του μοντέλου.

Συμπεριλαμβάνοντας την προεπεξεργασία των δεδομένων, την εξαγωγή χαρακτηριστικών και την τελική εκπαίδευση του μοντέλου μας. Τέλος αξιολογούνται οι διάφοροι αλγόριθμοι κατηγοριοποίησης και με σκοπό να καταλήξουμε στον πιο αποδοτικό.

9.2 Αλγόριθμοι Μηχανικής Μάθησης

9.3 Επιλογή Κινήσεων

Αρχικά, στόχος ήταν να αναγνωρίσουμε όλες τις πιθανές κινήσεις του χεριού και των δακτύλων. Ωστόσο, λόγω των περιορισμών που παρουσίαζαν οι αισθητήρες, δεν ήταν εφικτό να καταγράψουμε ή να αναγνωρίσουμε όλες τις κινήσεις με ακρίβεια τουλάχιστον με τον συγκεκριμένο αισθητήρα. Τελικά οι κινήσεις που επιλέχθηκαν είναι αυτές που μπορούν να καταγραφούν με μεγαλύτερη ακρίβεια από το μοντέλο.

1. Χαλάρωση
2. Γροθιά
3. Peace
4. Κάμψη μικρού δακτύλου
5. Κάμψη μεσαίου δακτύλου
6. Κάμψη παράμεσου δακτύλου

Εικόνα 52: Κάμψη μικρού δακτύλου

Εικόνα 53: Κάμψη μεσαίου δακτύλου

Εικόνα 54: Κάμψη μικρού δακτύλου

Εικόνα 55: Χαλάρωση

Εικόνα 56: Peace

9.4 Κατασκευή Dataset

Η δημιουργία του dataset αποτελεί μια απαιτητική διαδικασία και πραγματοποιήθηκε με τη βοήθεια ενός python script που αναπτύχθηκε ειδικά για τον σκοπό αυτό. Συγκεκριμένα, το script αναλάμβανε την παρακολούθηση των δεδομένων που αποστέλλονταν από τον μικροελεγκτή και την καταγραφή τους σε αρχείο CSV, με βάση την αντίστοιχη κλάση (label) και τον χρόνο καταγραφής που είχαμε ορίσει. Συνολικά, καταγράφηκαν 600 κινήσεις από κάθε κλάση.

Το script αρχικά συνδεόταν με τον MQTT broker για να λαμβάνει τα δεδομένα και στο κώδικα υλοποιείται η διαδικασία του labeling και την αποθήκευσή.

```
try:
    while True:
        label = input(
            f"Enter the label for the next {MINITES} minites (or type 'exit' to stop): "
        ).strip()
        if label.lower() == "exit":
            break
        collect_and_save_label(label)
except KeyboardInterrupt:
    print("Exiting...")
finally:
    client.loop_stop()
    client.disconnect()
```

Εικόνα 57: Κατασκευή Dataset (Κώδικας)

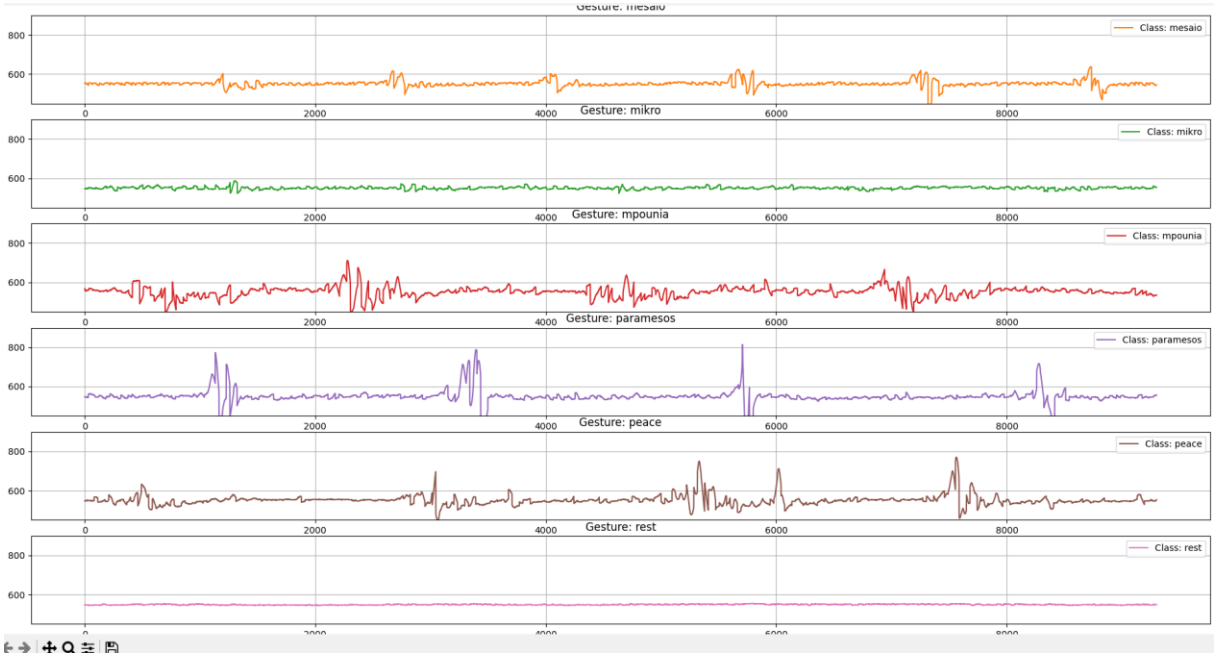
Αρχικά πληκτρολογούμε την κλάση που θέλουμε και για τα επόμενα 10 λεπτά το σύστημα μας κάνει καταγραφή ενώ πραγματοποιούμε την αντίστοιχη κίνηση με περίοδο ενός δευτερολέπτου.

Μόλις ολοκληρωθούν τα 10 λεπτά αποθηκεύονται τα αρχεία μας και μας εμφανίζεται μια προτροπή για να προχωρήσουμε με την επόμενη κλάση.

```
def collect_and_save_label(label) -> None:
    global data_buffer
    labeled_data = []
    print(f"Collecting data for label '{label}' for{MINITES} minites.")

    start_time = time.time()
    while time.time() - start_time < 60 * MINITES:
        with lock:
            labeled_data.extend(data_buffer)
            data_buffer.clear()
        time.sleep(0.5)  # Check the buffer every 500ms

    # Save labeled data to CSV
    with open(CSV_FILE, mode="a", newline="") as file:
        writer = csv.writer(file)
        writer.writerows([[label, value] for value in labeled_data])
        print(f"Saved {len(labeled_data)} data points for label '{label}' to CSV.")
```

Εικόνα 58: Κατασκευή Dataset (Κώδικας)

Στην παρακάτω εικόνα φαίνονται ενδεικτικά το ένα μικρό μέρος του συνόλου δεδομένων που καταγράψαμε.



Εικόνα 59: Ενδεικτικές τιμές του Dataset

9.5 Προεπεξεργασία Δεδομένων

Σε αυτή την ενότητα, θα αναλυθούν τα βήματα που ακολουθήθηκαν για την επεξεργασία των δεδομένων, έτσι ώστε να είναι έτοιμα για να εκπαιδευτούν από τον κατηγοριοποιητη.

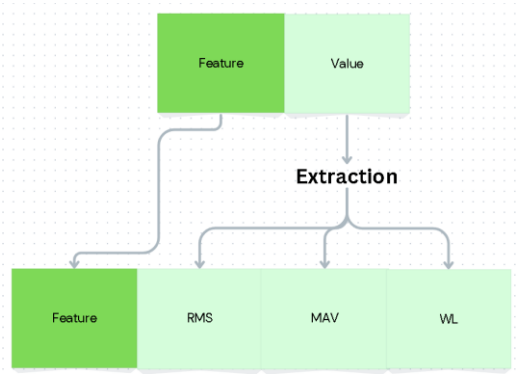
Ιδιαίτερη έμφαση δίνεται στη χρήση του WindowSize το οποίο χρησιμοποιείται για την κατανομή των δεδομένων σε μικρότερα υποσύνολα, διευκολύνοντας την ανάλυση και την εξαγωγή επιπλέον χρήσιμων χαρακτηριστικών που θα μας βοηθήσουν στην κατηγοριοποίηση. Επιπλέον χρησιμοποιείται ένας κανονικοποιητής για να φέρουμε τα χαρακτηριστικά σε συγκεκριμένη μορφή

9.5.1 Window Size

Το παράθυρο (window size) λειτουργεί ως μορφή αποθήκευσης δεδομένων. Καθώς μια τιμή από μόνη τις δεν περιέχει κάποια πληροφορία καθώς στέλνονται εκατοντάδες ανά δευτερόλεπτο. Κάθε παράθυρο αντιπροσωπεύει μια χρονική διάρκεια της κάθε περιόδου του σήματος. Δεδομένου ότι κάθε σήμα έχει περίοδο περίπου 1,3 δευτερόλεπτο και ο ρυθμός είναι περίπου 1860Hz υπολογίζεται ότι το παράθυρο θα πρέπει να έχει περίπου 2500 τιμές.

9.5.2 Εξαγωγή χαρακτηριστικών

Η εξαγωγή χαρακτηριστικών μας βοηθάει να εξάγουμε περισσότερα χαρακτηριστικά, καθώς βοηθά στη μετατροπή των ακατέργαστων μετρήσεων του σήματος σε χρήσιμες παραμέτρους. Στα πλαίσια τις ΔΕ υπάρχουν πολλές βιβλιογραφικές αναφορές ότι χαρακτηριστικά όπως μέση τετραγωνική ρίζα, η μέση απόλυτη τιμή και το μήκος κύματος είναι χρήσιμα χαρακτηριστικά σε εφαρμογές με δεδομένα EMG. [18, 19]



Εικόνα 60: Εξαγωγή χαρακτηριστικών

Η διαδικασία εξαγωγής χαρακτηριστικών είναι σημαντική, καθώς μας επιτρέπει να φιλτράρουμε τον θόρυβο που μπορεί να επηρεάζει τα δεδομένα, απομονώνοντας τα χαρακτηριστικά που είναι πραγματικά χρήσιμα. Τελικά τα χαρακτηριστικά καταλήγουν να έχουν 4 διαστάσεις, διευκολύνοντας την ανάλυση και την κατηγοριοποίηση των μοντέλων.

Μέση Τετραγωνική Ρίζα (RMS):

Η τιμή τις μέσης τετραγωνικής ρίζας δείχνει πόσο έντονο είναι το σήμα στο παράθυρο, μετρώντας την ενέργεια του σήματος. Αν έχει υψηλή τιμή, σημαίνει ότι η μυϊκή δραστηριότητα είναι έντονη.

rms = np.sqrt(np.mean(np.square(window)))

Μέση Απόλυτη Τιμή (MAV):

Η **μέση απόλυτη τιμή** υπολογίζει την μέση τιμή του σήματος χωρίς να παίρνει υπόψη αν είναι θετική ή αρνητική. Μας λέει πόσο έντονο είναι το σήμα ανεξαρτήτως κατεύθυνσης.

mav = np.mean(np.abs(window))

Μήκος Κύματος (WL):

Το μήκος κύματος μας δείχνει πόσο ασταθές είναι το σήμα, υπολογίζοντας πόσο αλλάζουν οι τιμές του σήματος μεταξύ τους. Το υψηλό μήκος κύματος σημαίνει υψηλή πολυπλοκότητα

wl = np.sum(np.abs(np.diff(window)))

Στην παρακάτω εικόνα απεικονίζεται η συνάρτηση που είναι υπεύθυνη για την εξαγωγή των χαρακτηριστικών.

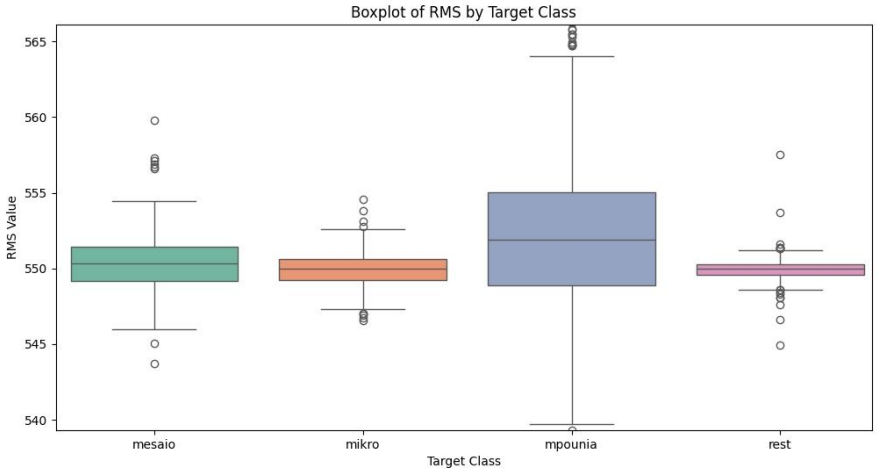
```python
# Feature extraction
def process_window(window) -> list:
    rms = np.sqrt(np.mean(np.square(window)))
    mav = np.mean(np.abs(window))
    wl = np.sum(np.abs(np.diff(window)))

    return [rms, mav, wl]
```

Εικόνα 61: Εξαγωγή χαρακτηριστικών

9.5.3 Οπτικοποίηση του Dataset

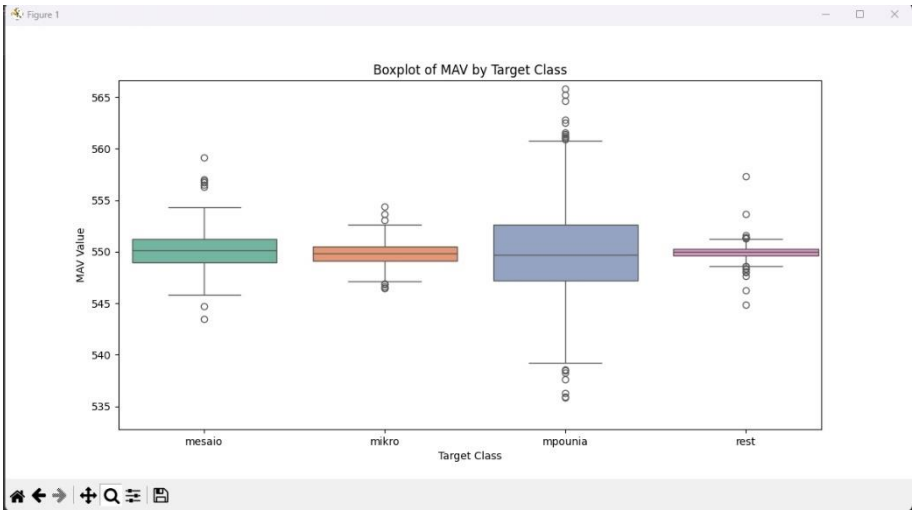
Η οπτικοποίηση των δεδομένων αποτελεί ένα σημαντικό βήμα στην προεπεξεργασία και ανάλυση των χαρακτηριστικών, παρέχοντας πολύτιμες πληροφορίες για την κατανομή, τη μέση τιμή και τη διασπορά των χαρακτηριστικών (RMS, MAV, και WL) για κάθε κλάση.

Μέσω των γραφημάτων, αποκτούμε μια σαφή εικόνα της συμπεριφοράς των δεδομένων, η οποία είναι απαραίτητη για την εξαγωγή χρήσιμων συμπερασμάτων.



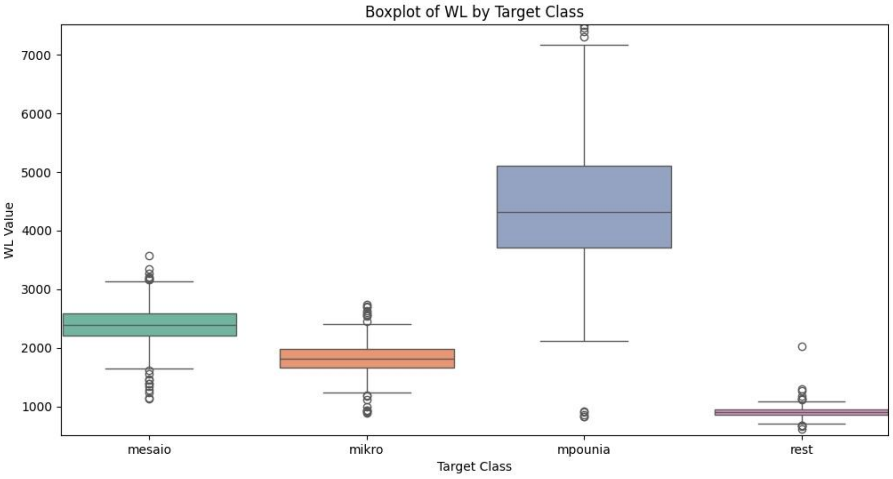
Εικόνα 62: Οπτικοποίηση Dataset (RMS)

Η χρήση αυτών των γραφημάτων καθιστά εύκολο τον εντοπισμό των ακραίων τιμών, ώστε να αποφασίσουμε εάν απαιτείται καθαρισμός των δεδομένων. Επιπλέον, μας επιτρέπει να αξιολογήσουμε ποια χαρακτηριστικά προσφέρουν τη μεγαλύτερη πληροφορία για τον διαχωρισμό των κλάσεων.



Εικόνα 63: Οπτικοποίηση Dataset (MAV)

Σε σύνολα δεδομένων με πολλά χαρακτηριστικά, είναι πολύ σημαντικό να δώσουμε προτεραιότητα σε αυτά που περιέχουν τη περισσότερη πληροφορία, για να βελτιστοποιούμε τη διαδικασία εκπαίδευσης του μοντέλου και να μειώνουμε την πολυπλοκότητα.



Εικόνα 64: Οπτικοποίηση Dataset (WL)

Καθώς τα δεδομένα μας συλλέχθηκαν με τον ίδιο τρόπο που θα χρησιμοποιηθούν για την κατηγοριοποίηση, δεν απαιτείται περαιτέρω καθαρισμός. Αυτό διασφαλίζει ότι τα χαρακτηριστικά που έχουν εξαχθεί ανταποκρίνονται με ακρίβεια στις πραγματικές συνθήκες και τις παραμέτρους που θα ληφθούν υπόψη κατά τη διάρκεια της κατηγοριοποίησης σε πραγματικό χρόνο.

9.5.4 Κανονικοποίηση

Ο κανονικοποιητής χρησιμοποιείται για να μετατρέψει τις τιμές των χαρακτηριστικών σε μια κοινή κλίμακα, έτσι ώστε οι τιμές να είναι εύκολα συγκρίσιμες. Επίσης, η κανονικοποίηση είναι ιδιαίτερα σημαντική όταν τα χαρακτηριστικά των δεδομένων έχουν διαφορετικό εύρος.

Στην περίπτωσή μας, το εύρος των τιμών ήταν ήδη προκαθορισμένο στο διάστημα 0-1024, λόγω της αναλογικής εξόδου του αισθητήρα. Εξετάσαμε δύο βασικές επιλογές για την κανονικοποίηση:

- StandardScaler: Κεντράρει τα δεδομένα γύρω από το μηδέν και τα κλιμακώνει βάσει της τυπικής τους απόκλισης.
- MinMaxScaler: Μετατρέπει τα χαρακτηριστικά σε ένα προκαθορισμένο εύρος τιμών, συνήθως στο διάστημα [-1, 1].

Μετά από αξιολόγηση των δύο μεθόδων κανονικοποίησης, καταλήξαμε στη χρήση του **StandardScaler** για την κανονικοποίηση των δεδομένων. Αν και οι επιδόσεις των δύο μεθόδων ήταν παρόμοιες,

9.6 Κατηγοριοποίηση και Μοντέλα Μηχανικής Μάθησης

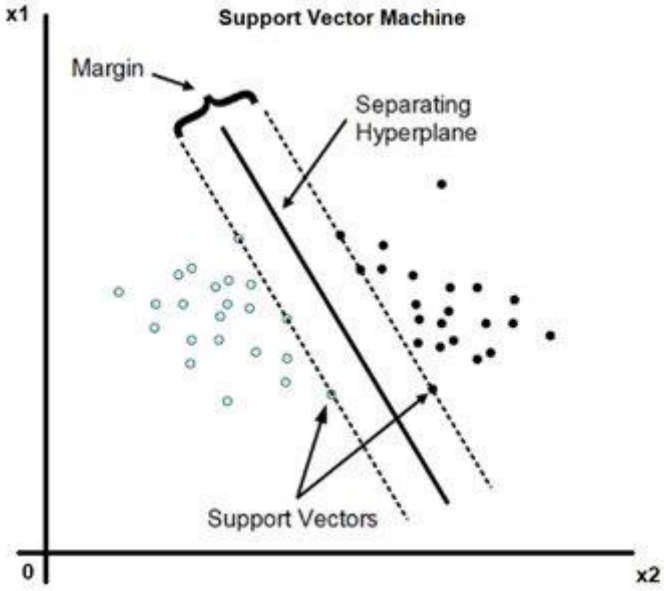
Καθώς στο συγκεκριμένο πρόβλημα η διάκριση των κατηγοριών δεν ήταν πάντοτε γραμμική, καθίσταται δύσκολη η πρόβλεψη της καταλληλότερης μεθόδου εκ των προτέρων. Για το λόγο αυτό, δοκιμάστηκαν διάφοροι σύγχρονοι αλγόριθμοι κατηγοριοποίησης γραμμική, μη-γραμμικοί αλλά και μοντέλα νευρωνικών δικτύων και .

9.6.1 Γραμμικοί Αλγόριθμοι:

Τα γραμμικά προβλήματα βασίζονται στην υπόθεση ότι υπάρχει μια γραμμική σχέση μεταξύ των εισόδων και των εξόδων.

SVM:

Ο αλγόριθμος Support Vector Machine είναι αλγόριθμος επιβλεπόμενης μηχανικής μάθησης που χρησιμοποιείται τόσο για προβλήματα ταξινόμησης όσο και παλινδρόμησης. Ο βασικός του στόχος είναι να βρει την ευθεία ή το υπερεπίπεδο που να διαχωρίζει τις κατηγορίες με τον πιο αποτελεσματικό τρόπο. [22]



Εικόνα 65: SVM [22]

Τα δεδομένα που βρίσκονται πιο κοντά στη βέλτιστη διαχωριστική γραμμή ονομάζονται διανύσματα υποστήριξης και είναι καθοριστικής σημασίας, καθώς η θέση τους επηρεάζει τον τρόπο με τον όποιο θα δημιουργηθεί η διαχωριστική γραμμή. Περιθώριο (margin) είναι η απόσταση μεταξύ τις διαχωριστικής ευθείας και των διανυσμάτων υποστήριξης.

Η διαδικασία τις εκπαίδευσης περιλαμβάνει:

1. **Αναγνώριση Διανυσμάτων Υποστήριξης**: Τα σημεία που βρίσκονται κοντά στα διανύσματα υποστήριξης είναι τα πιο σημαντικά. Τα υπόλοιπα σημεία δεν επηρεάζουν καθόλου.
2. **Μέγιστο Περιθώριο**: Στη συνέχεια, ο αλγόριθμος προσπαθεί να μεγιστοποιήσει το περιθώριο δηλαδή την απόσταση μεταξύ τις διαχωριστικής γραμμής και των διανυσμάτων υποστήριξης.

3. **Βελτιστοποίηση Σφαλμάτων**: Ο αλγόριθμος ελέγχει την μεταβλητή χαλαρότητας , η οποία καθορίζει κατά πόσο επιτρέπονται τα σφάλματα.

Ο SVM είναι κατάλληλος τόσο για γραμμικά όσο και για μη γραμμικά διαχωρίσιμα δεδομένα. Στα μη γραμμικά διαχωρίσιμα προβλήματα χρησιμοποιούμε μια τεχνική που μετασχηματίζουμε τα δεδομένα με την βοήθεια των συναρτήσεων kernel , σε περισσότερες διαστάσεις όπου εκεί το πρόβλημα επιλύεται γραμμικά.

LDA (Linear Discriminant Analysis):

Ο αλγόριθμος LDA βασίζεται στον κανόνα ταξινόμησης του Bayes, έναν κανόνα που βασίζετε σε πιθανότητες. Χρησιμοποιείται για την μείωση διαστάσεων και την ταξινόμηση δεδομένων. Είναι πολύ αποτελεσματικός όταν τα δεδομένα έχουν πολλές διαστάσεις και ακολουθούν την κανονική κατανομή.

Η εκπαίδευση ξεκινάει με τον υπολογισμό της μέσης τιμής της κάθε κατηγορίας. Στην συνέχεια υπολογίζεται ο κοινός πίνακας διακύμανσης. Στον όποιο περιγράφεται πώς τα δεδομένα διασκορπίζονται γύρω από τις μέσες τιμές και μας βοηθάει τον αλγόριθμο να εξετάσει τις συσχετίσεις. Ο στόχος είναι να μεγιστοποιήσει την απόσταση μεταξύ των κατηγοριών και ταυτόχρονα να ελαχιστοποιήσει τη διασπορά μέσα σε κάθε κατηγορία Εντοπίζονται γραμμές ή επίπεδα στο χώρο που διαχωρίζουν καλύτερα τις κατηγορίες.

9.6.2 Μη Γραμμικοί Αλγόριθμοι:

Οι μη γραμμικοί αλγόριθμοι χρησιμοποιούνται όταν η σχέση μεταξύ των χαρακτηριστικών και των εξόδων δεν μπορεί να αναπαρασταθεί με μια γραμμική εξίσωση. Αυτοί οι αλγόριθμοι είναι ικανοί να χειριστούν πιο περίπλοκες σχέσεις μεταξύ των δεδομένων και συνήθως απαιτούν περισσότερους υπολογιστικούς πόρους.

Χρησιμοποιούν τεχνικές όπως νευρωνικά δίκτυα, δέντρα απόφασης και συναρτήσεις kernel. Αν και έχουν μεγάλο υπολογιστικό κόστος σε πολλές περιπτώσεις είναι η καλύτερη επιλογή.

KNN (K-Nearest Neighbors):

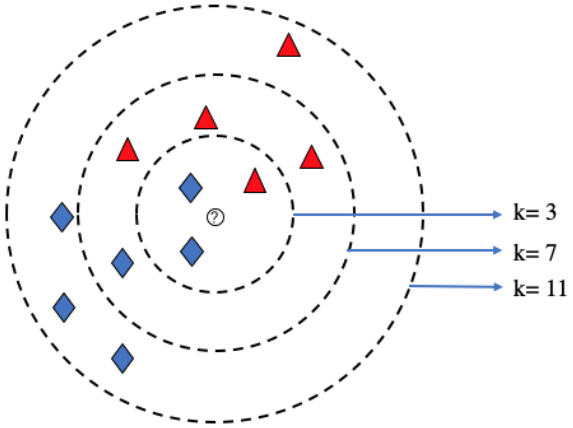
Ο αλγόριθμος k-Nearest Neighbors (k-NN) είναι ένας αλγόριθμος μη επιβλεπόμενης μάθησης που χρησιμοποιείται κυρίως για ταξινόμηση. Η βασική ιδέα πίσω από τον k-NN είναι ότι, για κάθε νέο δείγμα, η κατηγορία του καθορίζεται από την πλειοψηφία των "k" πιο κοντινών γειτόνων του στο σύνολο των δεδομένων. Ο υπολογισμός τις απόστασης γίνετέ με την βοήθεια του παρακάτω τύπου τις ευκλείδεια απόστασης:

Ευκλείδεια απόσταση:

$$d(p_1, p_2) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

1. **Καθορισμός της απόστασης**: Για να βρει τους κοντινότερους γείτονες, ο αλγόριθμος υπολογίζει την απόσταση μεταξύ του νέου δείγματος και των υπολοίπων δεδομένων. Συνήθως χρησιμοποιείτε η ευκλείδεια ή η απόσταση

2. **Επιλογή του k**: Η τιμή του "k" καθορίζει πόσους γείτονες θα λάβουμε υπόψη. Αν, για παράδειγμα, το k είναι 3, το νέο δείγμα ταξινομείται ανάλογα με την πλειοψηφία των τριών πιο κοντινών σημείων.



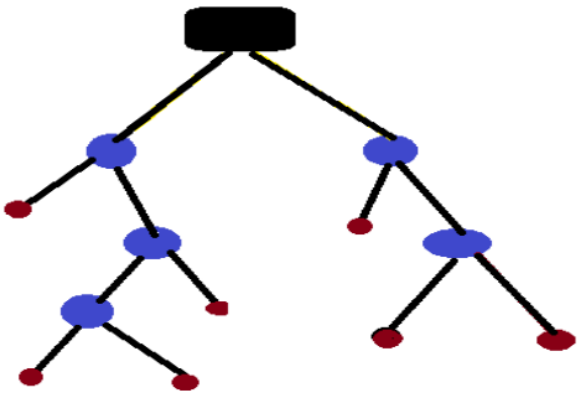
Εικόνα 66: KNN [25]

Ο αλγόριθμος k-NN είναι απλός στην υλοποίηση, δεν απαιτεί εκπαίδευση και είναι ευαίσθητος στην επιλογή των παραμέτρων.

Ωστόσο, μπορεί να γίνει υπολογιστικά ακριβός για μεγάλα σύνολα δεδομένων, καθώς απαιτεί να υπολογίζει τις αποστάσεις για κάθε νέο δείγμα σε όλο το σύνολο των δεδομένων.

9.6.3 Δέντρα Απόφασης:

Τα δέντρα απόφασης λειτουργούν ως εργαλεία που μας καθοδηγούν στην διαδικασία λήψης αποφάσεων. Ξεκινάμε από την κορυφή του δέντρου και, με βάση τα χαρακτηριστικά των δεδομένων, ακολουθούμε τα κλαδιά μέχρι να καταλήξουμε στην τελική απόφαση ή κατηγορία.



Εικόνα 67: Δομή των Δέντρων απόφασης [21]

Random Forest:

Ο αλγόριθμος Random Forest βασίζεται στη δημιουργία και τον συνδυασμό πολλαπλών δέντρων απόφαση. Κάθε δέντρο εξετάζει ένα τυχαίο υποσύνολο και από αυτό επιλέγεται τυχαία ένα υποσύνολο χαρακτηριστικών. Με αυτήν την προσέγγιση κάποια χαρακτηριστικά ενδέχεται να χρησιμοποιηθούν περισσότερες από μία φορές, ενώ άλλα ενδέχεται να μην χρησιμοποιηθούν καθόλου. Η τυχαιότητα στην επιλογή δεδομένων και χαρακτηριστικών μειώνει την εξάρτηση από μεμονωμένα λάθη ή θορύβους στα δεδομένα και το καθιστά ισχυρό στην επίλυση προβλημάτων.

Στη διαδικασία εκπαίδευσης, επιλέγεται το χαρακτηριστικό που προσφέρει την περισσότερη πληροφορία, χρησιμοποιώντας κριτήρια όπως η εντροπία.

Παράμετροι του αλγορίθμου:

- **Αριθμός Δέντρων:** Καθορίζει τον αριθμό των δέντρων που θα δημιουργηθούν.
- **Μέγιστο Βάθος:** Καθορίζει το μέγιστο βάθος που μπορεί να έχει κάθε δέντρο.
- **Ελάχιστος Αριθμός Δειγμάτων:** Καθορίζει τον ελάχιστο αριθμό δειγμάτων που απαιτούνται για να πραγματοποιηθεί διαχωρισμός σε έναν κόμβο.
- **Τυχαία Χαρακτηριστικά:** Ορίζει τον μέγιστο αριθμό των χαρακτηριστικών που θα εξετάζονται τυχαία σε κάθε κόμβο.

Ο αλγόριθμος συνδυάζει τις προβλέψεις όλων των δέντρων, με κάθε δέντρο να παρέχει την κατηγορία που θεωρεί ως πιο πιθανή. Η τελική απόφαση λαμβάνεται μέσω της διαδικασίας της πλειοψηφικής ψήφου.

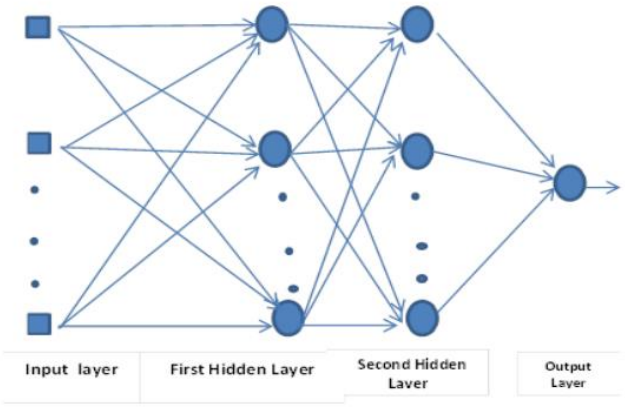
9.6.4 Νευρωνικά Δίκτυα:

Τα νευρωνικά δίκτυα προσπαθούν να προσημειώσουν την λειτουργία του ανθρωπίνου εγκεφάλου. Κάθε νευρωνικό δίκτυο αποτελείται από:

1. Είσοδοι: Είναι τα χαρακτηριστικά (features).

2. Κρυφά Επίπεδα: Τα κρυφά επίπεδα επεξεργάζονται τα δεδομένα με τη βοήθεια συναρτήσεων ενεργοποίησης. Κάθε νευρώνας είναι συνδεδεμένος (με βάρη) με νευρώνες από το επόμενο επιπέδου ή με την έξοδο. Τα βάρη αυτά καθορίζουν τη σημασία κάθε εισόδου για την έξοδο του νευρώνα.

3. Έξοδοι: Είναι το επιθυμητό αποτέλεσμα που προβλέπει το δίκτυο, όπως η κλάση στην οποία ανήκει η είσοδος.



Εικόνα 68: Αρχιτεκτονική ενός Νευρωνικού Δικτύου [21]

Multi-Layer Perceptron (MLP)

Ο αλγόριθμος MLP είναι ένας από τους πιο δημοφιλείς αλγόριθμους νευρωνικών δικτύων. Οι παράμετροι του, παίζουν κρίσιμο ρόλο στην απόδοση. Καθώς επηρεάζουν την ικανότητα του να μαθαίνει και να προσαρμόζεται σε διαφορετικά δεδομένα και προβλήματα.

Παράμετροι του αλγορίθμου:

- **Βήμα εκπαίδευσης:** Το **βήμα εκπαίδευσης** καθορίζει πόσο θα αλλάξουν τα βάρη του μοντέλου σε κάθε εποχή.

- **Μέγιστος Αριθμός Εποχών:** Ο αριθμός αυτός καθορίζει τον μέγιστο αριθμό των εποχών που θα εκπαιδευτεί το μοντέλο μας.. Αν το μοντέλο δεν βελτιώνεται με το πέρασμα των εποχών θα σταματήσει νωρίτερα.

- **Συνάρτηση ενεργοποίησης:** Είναι υπεύθυνη για την εισαγωγή μη γραμμικότητας στο μοντέλο, επιτρέποντας στο δίκτυο να μάθει περίπλοκες σχέσεις στα δεδομένα.

Τελικά, ο αλγόριθμος MLP είναι, ικανός να λύσει περίπλοκες σχέσεις στα δεδομένα. Ωστόσο, λόγω της φύσης της μεθόδου backpropagation, υπάρχει η πιθανότητα το μοντέλο να καταλήξει σε τοπικό ελάχιστο, κάτι που μπορεί να περιορίσει την απόδοσή του.

9.7 Σύγκριση Αλγορίθμων και Αξιολόγηση Αποτελεσμάτων

Η αξιολόγηση των αλγορίθμων γίνεται συνήθως με τη χρήση διάφορων μετρικών.Οι πιο κοινές μετρικές αξιολόγησης είναι οι Accuracy, Precision , Recall και F1-Score. Αυτές οι μετρικές υπολογίζονται χρησιμοποιώντας τα τέσσερα βασικά αποτελέσματα της ταξινόμησης:

- **True Positives** TP: Οι σωστές προβλέψεις του αλγορίθμου για τις θετικές περιπτώσεις.

- **True Negatives** TN: Οι σωστές προβλέψεις για τις αρνητικές περιπτώσεις.

- **False Positives** FP: Οι περιπτώσεις όπου το μοντέλο προβλέπει λανθασμένα ως θετικές, ενώ είναι αρνητικές.

- **False Negatives** FN: Οι περιπτώσεις όπου το μοντέλο προβλέπει λανθασμένα ως αρνητικές, ενώ είναι θετικές

Αυτά τα αποτελέσματα χρησιμοποιούνται για τον υπολογισμό των μετρικών αξιολόγησης όπως φαίνεται παρακάτω.

1. **Accuracy**: Η ακρίβεια υπολογίζεται ως το ποσοστό των σωστών προβλέψεων σε σχέση με το σύνολο των παρατηρήσεων. Είναι χρήσιμη όταν οι κατηγορίες είναι ισόρροπες.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

2. **Precision** : Είναι το ποσοστό των σωστών προβλέψεων πάνω από το σύνολο των προβλέψεων του αλγορίθμου.

$$Precision = \frac{TP}{TP+FN}$$

3. **Recall**: Μετρά πόσο καλά το μοντέλο εντοπίζει τις θετικές περιπτώσεις, δηλαδή το ποσοστό των σωστών θετικών προβλέψεων από το σύνολο των πραγματικά θετικών παραδειγμάτων.

$$Recall = \frac{TP}{TP+FN}$$

4. **F1-Score**: Είναι ο μέσος όρος της ακρίβειας και τις ευαισθησίας.

$$F1Score = 2 * \frac{Precision * Recall}{Presision + Recall}$$

Αυτές οι μετρικές βοηθούν στην πιο ολοκληρωμένη εκτίμηση της απόδοσης του αλγορίθμου, καθώς η αξιολόγηση μπορεί να διαφέρει ανάλογα με το τι είναι πιο σημαντικό για το συγκεκριμένο πρόβλημα.

Στα παρακάτω παραδείγματά υπάρχουν οι κλάσεις (Mikro,Mesaio,Mpounia,Rest) και παρακάτω παρατίθεται ο συγκεντρωτικός πίνακας με τις επιδόσεις των μοντέλων. Στον παρακάτω πίνακα εμφανίζονται οι επιδώσεις.

Οι παρακάνω τιμές υπολογίζονται με την βοήθεια των συναρτήσεων που έχουν ήδη υλοποιηθεί στα πλαίσια των μετρικών τις βιβλιοθήκης scikit-learn : accuracy_score, recall_score, precision_score, f1_score.

```
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred, average="macro")
precision = precision_score(y_test, y_pred, average="macro")
f1 = f1_score(y_test, y_pred, average="macro")
```

Εικόνα 69: Μετρικές Αξιολόγησης

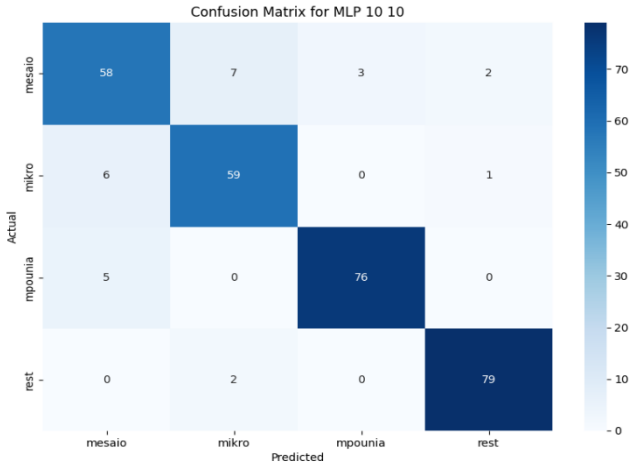
Πίνακας 3 Συγκεντρωτικός πίνακας με τις επιδώσεις τον μοντέλων

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
MLP	0.91	0.91	0.91	0.91
SVM (rbf)	0.90	0.90	0.90	0.90
Random Forest	0.90	0.90	0.90	0.90
K-NN	0.90	0.89	0.89	0.89
SVM (Linear)	0.90	0.90	0.90	0.90
LDA	0.90	0.90	0.90	0.90

Από τις επιδόσεις των δοκιμαστικών μοντέλων φαίνεται ότι η ακρίβεια κυμαίνεται σε υψηλά ποσοστά, γεγονός που δείχνει ότι τα μοντέλα ανταποκρίνονται καλά στις απαιτήσεις του συγκεκριμένου προβλήματος. Σε περίπτωση που συμπεριλαμβάναμε κλάσεις όπως κάμψη παράμεσου, και το σύμβολο peace τα ποσοστά ακρίβειάς θα δεν θα ξεπερνούσαν το 65%.

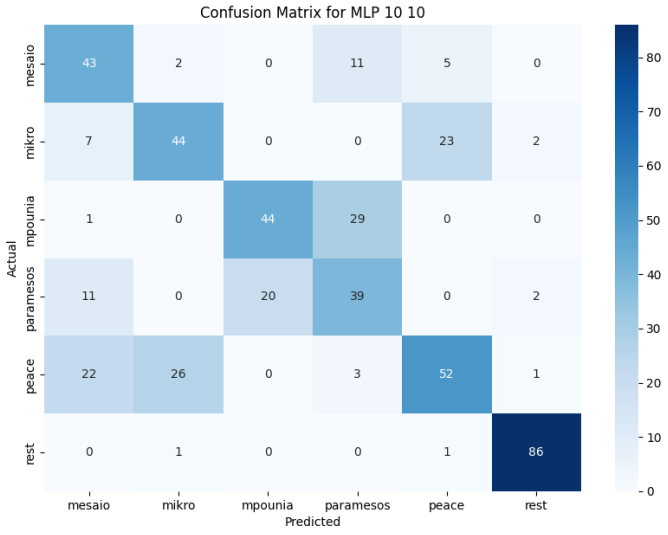
Πίνακας Σύγχυσης

Ο Πίνακας Σύγχυσης είναι ένα ακόμα εργαλείο αξιολόγησης που χρησιμοποιείται για να δείξει οπτικά τις επιδόσεις του αλγορίθμου. Αναπαριστά τα αποτελέσματα του μοντέλου συγκρίνοντας τις πραγματικές ετικέτες των δεδομένων με τις προβλέψεις του αλγορίθμου. Μέσω αυτού του πίνακα, είναι σαφές πόσα παραδείγματα ταξινομήθηκαν σωστά και πόσα λανθασμένα.



Εικόνα 70: Πίνακας Σύγχυσης (Βασικές κινήσεις)

Η κύρια δυσκολία που εντοπίζεται είναι στην αναγνώριση της κάμψης του μεσαίο δάχτυλου , καθώς σε ορισμένες περιπτώσεις το μοντέλο μπερδεύει αυτή την κατηγορία με την μπουνιά ή με την κάμψη του μικρού δακτύλου. Παρ' όλα αυτά, αυτή δεν επηρεάζει σημαντικά τα αποτελέσματα. Η συγκεκριμένη υλοποίηση που είναι και η βέλτιστή αποτελείται από τον αλγόριθμο MLP με 2 κρυφά στρώματα από 10 νέρωνες σε κάθε στρώμα.



Εικόνα 71:Πίνακας Σύγχυσης(Συνδυαστικές κίνησης)

Στην περίπτωση που θα θέλαμε να συμπεριλάβουμε στις κλάσεις μας την κίνηση peace και την κάμψη του παράμεσου. Οι αλγόριθμοι όπως ενδεικτικά φαίνεται παρακάτω δεν είναι σε θέση να αναγνωρίσουν με επιτυχία την κλάση. Τα αποτελέσματα ακρίβειας που πήραμε σε αυτές τις περίπτωση περιπτώσεις κυμαίνονται από 55 έως 65 %.

Κεφάλαιο 10ο: Συμπεράσματα και Προτάσεις Βελτίωσης

10.1 Συμπεράσματα

Η παρούσα εργασία επικεντρώνεται στη δημιουργία ενός καινοτόμου συστήματος που δέχεται σήματα ηλεκτρομυογραφίας (ΕΜΓ) για καταγραφή και αποκωδικοποίηση κινήσεων των δακτύλων και της παλάμης.

Η αρχιτεκτονική του συστήματος βασίζεται σε ένα αυτοσχέδιο βραχιόλι με ενσωματωμένους αισθητήρες EMG που καταγράφουν τα σήματα από τις κινήσεις. Ο μικροελεγκτής λαμβάνει αυτά τα σήματα και τα στέλνει στον server, όπου αναλύονται και κατηγοριοποιούνται. Τα αποτελέσματα παρουσιάζονται σε πραγματικό χρόνο μέσω μιας διεπαφής, επιτρέποντας στον χρήστη να βλέπει άμεσα τις αναγνωρισμένες κινήσεις.

Το πρωτότυπο βραχιόλι διαθέτει αυτοσχέδια κατασκευή για την τοποθέτηση των αισθητήρων στην κατάλληλη θέση πάνω στο χέρι, εξασφαλίζοντας τη σωστή καταγραφή των μυϊκών σημάτων. Στην συνέχεια οι αισθητήρες λαμβάνουν το σήμα από των μυ , τα οποία αντιπροσωπεύουν τη δραστηριότητα των μυών κατά τη διάρκεια συγκεκριμένων κινήσεων. Τα σήματα χρησιμοποιούνται για την εκπαίδευση του μοντέλου, καθώς η περίοδος των σημάτων ήταν 1.5 δευτερόλεπτο καθοριστικέ το παράθυρο συλλογής δεδομένων με βάση αυτήν την περίοδο. Στην συνέχεια χρησιμοποιήσαμε χαρακτηριστικά όπως το Root Mean Square (RMS), το Mean Absolute Value (MAV) και το Waveform Length (WL) τα όποια αποδεικνύεται είναι ιδιαίτερα χρήσιμα για τέτοιο είδους προβλήματα.

Στο στάδιο της επεξεργασίας, τα δεδομένα στέλνονται από τους αισθητήρες μεταφέρονται μέσω του μικροελεγκτή στον server χρησιμοποιώντας το πρωτόκολλο MQTT. Εκεί εφαρμόζονται αλγόριθμοι μηχανικής μάθησης, εκπαιδευμένοι να αναγνωρίζουν συγκεκριμένες κινήσεις από τα σήματα. Κατά τη διάρκεια της ανάπτυξης, δοκιμάστηκαν διάφοροι αλγόριθμοι, όπως SVM, Random Forest, MLP και KNN για την επιλογή του καταλληλότερου αλγορίθμου. Στο τέλος τις κατηγοριοποίησης τα δεδομένα στέλνονταν στην διεπαφή μέσω του πρωτοκόλλου WebSocket επιτρέποντας την άμεση οπτικοποίηση των αποτελεσμάτων σε πραγματικό χρόνο.

Η διαδικασία αξιολόγησης έγινε με την βοήθεια μετρικών όπως Accuracy, Precision, Recall και F1-Score. Επιλέχθηκαν κινήσεις που είναι αναγνωρίσιμες και πρακτικές, έτσι ώστε το σύστημα να επιτυγχάνει υψηλή ακρίβεια. Επίσης προσπαθήσαμε να αναλύσουμε συνδυαστικές κινήσεις, αλλά τα σήματα που λαμβάναμε ήταν σε μεγάλο βαθμό ίδια και ήταν δύσκολο να κατηγοριοποιηθούν σωστά.

Ο τελικός αλγόριθμος που επιλέχθηκε ήταν ο MLP, ο οποίος πέτυχε ακρίβεια 90% στην αναγνώριση των βασικών κινήσεων.

10.2 Προτάσεις Βελτίωσης

Κάποιες προτάσεις βελτίωσης θα ήταν:

1. Το βραχιόλι να είχε μικρότερο μέγεθος να ήταν εργονομικό και βολικό για να μπορεί να χρησιμοποιηθεί ακόμα και στην καθημερινότητα.

2. Επίσης, η αναγνώριση περισσότερων και πιο περίπλοκων κινήσεων θα ενίσχυε την αποτελεσματικότητα του συστήματος.

3. Το backend θα μπορούσε να υλοποιηθεί εξολοκλήρου στον μικροελεγκτή ελαχιστοποιώντας την απόσταση και τον χρόνο μεταφοράς των δεδομένων.

4. Η διεπαφή θα μπορούσε να γίνει σε ένα περιβάλλοντος επαυξημένης πραγματικότητας προσφέροντας μια πιο διαδραστική εμπειρία χρήσης.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Ν. Καρανδρέας, Εισαγωγή στην Ηλεκτρομυογραφία και Ηλεκτρονευρογραφία, Παρισιάνου Α.Ε, 2016.

[2] Α. Τσιόκανος, Ανατομία, ΤΕΦΑΑ Πανεπιστήμιο Θεσσαλίας.

[3] Reuters, "Facebook Buying CTRL-labs, a Start-Up Looking to Guide Computers With Brain Signals," *The New York Times,* 2019.

[4] "x-Trodes," [Online]. Available: https://xtrodes.com/knowledge-base/.

[5] "Wemos D1," [Online]. Available: https://www.wemos.cc/en/latest/d1/d1_mini.html.

[6] J. C. D. C. Leila G. Ablao, Machine Learning Sleep Phase Monitoring using, ICSE, 2021.

[7] "AD8232 Manual," [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/ad8232.pdf.

[8] A. Fuggetta, Open source software—an evaluation, Politecnico di Milano.

[9] "Git," [Online]. Available: https://git-scm.com/.

[10] Docker. [Online]. Available: https://www.docker.com/.

[11] H. M. H. G. A. Marwa O Al Enany, "A Comparative analysis of MQTT and IoT application protocols," IEEE, 2021.

[12] W. H. H. Z. Shiqi Guan, Real-Time Data Transmission Method Based on Websocket Protocol for Networked Control System Laboratory, IEEE, 2019.

[13] A. Diaconu, The WebSocket, 2022.

[14] F. H. C. K. J. D. O. F. Donelson Smith, What TCP/IP protocol headers can tell us about the web, SIGMETRICS, 2001.

[15] P. D. Dawoud Shenouda Dawoud, Serial Communication Protocols and Standards, River, 2020.

[16] Python. [Online]. Available: https://www.python.org/.

[17] M. E. McDonald, Fastapi: Modern Python Web Development, O'Reilly Media, 2020.

[18] Scikit-Learn. [Online]. Available: https://scikit-learn.org/stable/.

[19] "Visual Studio," [Online]. Available: https://code.visualstudio.com/.

[20] "Arduino," [Online]. Available: https://www.arduino.cc/.

[21] O. T. Mohamed Garouche, Development of a Low-Cost Portable EMG, MDPI, 2023.

[22] A. A. K. O. F. Salman Mohd Khan, Pattern recognition of EMG signals, IOP Publishing, 2021.

[23] S. G. A. D. Aleena Swetapadma, A Study on Support Vector Machine based Linear, ICISS, 2019.

[24] W. Zhang, A Distributed Storage and Computation k-Nearest Neighbor Algorithm, IEEE, 2020.

[25] R. K. N. A. I. M. Jehad Ali, Random Forests and Decision Trees, IJCS, 2012.

[26] M. R. M. H. S. S. S. A. Faisal Shehzad, A Scalable System-on-Chip Acceleration for Deep Neural Networks, IEEE, 2021.

[27] E. Parliament, "What is artificial intelligence and how is it used?," 2020.

[28] P. N. Stuart Russell, Artificial Intelligence: A Modern Approach (4th Edition), 2020.

[29] C. V.-V. C. S.-A. Deyby Huamanchahua and D. S.-Q. Hector Valcarcel-Castillo, Electro-Stimulator for Limbs Strength Enhancement: An Innovative and Technological Review, IEE, 2023.

[30] "Matplotlib," [Online]. Available: https://matplotlib.org/.