



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ

ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

**ΜΕΛΕΤΗ, ΣΧΕΔΙΑΣΗ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΕΝΟΣ
ΣΥΣΤΗΜΑΤΟΣ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ
ΑΣΘΕΝΩΝ ΜΕ ΧΡΗΣΗ ΥΛΙΚΟΥ ΓΙΑ ΤΗΝ
ΚΑΤΑΓΡΑΦΗ ΖΩΤΙΚΩΝ ΛΕΙΤΟΥΡΓΙΩΝ (ECG,
ΟΧΙΜΕΤΕΡ) ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΗΣ ΟΡΑΣΗΣ
ΓΙΑ ΕΦΑΡΜΟΓΕΣ ΣΕ ΕΞΥΠΝΕΣ ΠΟΛΕΙΣ
(SMART CITY)**

ΑΥΛΩΝΙΤΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ-ΟΔΥΣΣΕΑΣ

ΕΠΙΒΛΕΠΩΝ: ΚΙΤΣΟΣ ΠΑΡΑΣΚΕΥΑΣ, ΚΑΘΗΓΗΤΗΣ

ΠΑΤΡΑ 2025

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή Πάτρα,
Ημερομηνία

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Παρασκευάς Κίτσος
2. Νικόλας Πετρέλης
3. Σωτήρης Χριστοδούλου

Υπεύθυνη Δήλωση Φοιτητή

Βεβαιώνω ότι είμαι συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αντές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τη συγκεκριμένη εργασία.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Πελοποννήσου δεν υποδηλώνει απαραίτητας και αποδοχή των απόψεων του συγγραφέα εκ μέρους του Τμήματος.

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του φοιτητή που την εκπόνησε. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης ο συγγραφέας/δημιουργός εκχωρεί στο Πανεπιστήμιο Πελοποννήσου, μη αποκλειστική άδεια χρήσης του δικαιώματος αναπαραγωγής, προσαρμογής, δημόσιου δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσής τους διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος και για όλο το χρόνο διάρκειας των δικαιωμάτων πνευματικής ιδιοκτησίας. Η ανοικτή πρόσβαση στο πλήρες κείμενο για μελέτη και ανάγνωση δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, αποθήκευση, πώληση, εμπορική χρήση, μετάδοση, διανομή, έκδοση, εκτέλεση, «μεταφόρτωση» (downloading), «ανάρτηση» (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού. Ο συγγραφέας/δημιουργός διατηρεί το σύνολο των ηθικών και περιουσιακών του δικαιωμάτων.

Περίληψη

Η παρούσα διπλωματική εργασία αφορά τη μελέτη, σχεδίαση και υλοποίηση ενός συστήματος παρακολούθησης ασθενών, το οποίο χρησιμοποιεί υλικό για την καταγραφή ζωτικών λειτουργιών (ECG, οξύμετρο) και υπολογιστική όραση, με εφαρμογές σε έξυπνες πόλεις. Στόχος του συστήματος είναι η απομακρυσμένη παρακολούθηση ηλικιωμένων και ατόμων με χρόνιες παθήσεις, που διαμένουν, σε αστικές, μη-αστηκές ή σε απομακρυσμένες περιοχές, όπου δεν έχουν εύκολη πρόσβαση σε υγειονομικές υπηρεσίες.

Το σύστημα αποτελείται ξεχωριστά από έναν αισθητήρα που μετρά τους καρδιακούς παλμούς και τα επίπεδα οξυγόνου στο αίμα, έναν αισθητήρα καρδιογραφήματος, καθώς και μια κάμερα που αναγνωρίζει τη στάση του σώματος (όρθιος, καθιστός, ξαπλωμένος). Τα δεδομένα των αισθητήρων μέτρησης καρδιακών παλμών, μέτρησης οξυγόνου και καρδιογραφήματος μεταδίδονται μέσω διαδικτύου σε εφαρμογή κινητού τηλεφώνου, επιτρέποντας την απομακρυσμένη παρακολούθηση των ασθενών. Για την υλοποίηση χρησιμοποιούνται οι πλακέτες Arduino Nano 33 BLE Sense και ESP-8266 Wemos D1 Mini, οι οποίες προσφέρουν δυνατότητες ενσωμάτωσης τεχνητής νοημοσύνης και ασύρματης επικοινωνίας.

Η επικοινωνία των αισθητήρων με την εφαρμογή βασίζεται στο πρωτόκολλο MQTT, το οποίο επιτρέπει την ασφαλή και αποδοτική μετάδοση δεδομένων. Επιπλέον, χρησιμοποιούνται τεχνικές μηχανικής μάθησης και νευρωνικών δικτύων για την ανάλυση των δεδομένων της κάμερας, επιτρέποντας την ανίχνευση της στάσης του σώματος.

Το σύστημα συμβάλλει στην πρόληψη σοβαρών καταστάσεων, βελτιώνοντας την ποιότητα ζωής των ασθενών και διευκολύνοντας την απομακρυσμένη ιατρική παρακολούθηση. Τα αποτελέσματα της μελέτης δείχνουν ότι η προσέγγιση αυτή

μπορεί να αποτελέσει σημαντικό εργαλείο για την υποστήριξη του συστήματος υγείας σε έξυπνες πόλεις, μειώνοντας τον χρόνο απόκρισης σε επείγοντα περιστατικά και ενισχύοντας την αυτονομία των ασθενών.

Abstract

This thesis focuses on the study, design, and implementation of a patient monitoring system that utilizes hardware for recording vital signs (ECG, oximeter) and computer vision, with applications in smart cities. The system aims to enable remote monitoring of elderly individuals and patients with chronic illnesses who live in urban, as well in rural and remote areas and lack easy access to healthcare services.

The system consists of separate sensors for measuring heart rate, blood oxygen levels, and electrocardiograms, as well as a camera that detects body posture (standing, sitting, lying down). The data from the heart rate, oxygen level, and ECG sensors are transmitted via the internet to a mobile application, allowing remote monitoring of patients. The implementation is based on the Arduino Nano 33 BLE Sense and ESP-8266 Wemos D1 Mini boards, which provide capabilities for artificial intelligence integration and wireless communication.

Sensor communication with the application is based on the MQTT protocol, ensuring secure and efficient data transmission. Additionally, machine learning techniques and neural networks are employed for analyzing camera data, enabling accurate body posture detection.

The system contributes to the prevention of critical conditions, improving patients' quality of life and facilitating remote medical monitoring. The findings of this study indicate that this approach can serve as a valuable tool in supporting healthcare systems within smart cities, reducing response times in emergencies and enhancing patient autonomy.

Περιεχόμενα

Περίληψη

Abstract

ΕΙΣΑΓΩΓΗ.....	10
1 ΠΕΡΙΓΡΑΦΗ ARDUINO.....	11
1.1 ΣΚΟΠΟΣ ΔΗΜΙΟΥΡΓΙΑΣ ΤΟΥ ARDUINO	11
1.2 ΔΥΝΑΤΟΤΗΤΕΣ ARDUINO	12
1.3 ΠΛΕΟΝΕΚΤΗΜΑΤΑ ARDUINO	12
1.4 ΕΚΔΟΧΕΣ ARDUINO	13
1.5 ΠΑΡΑΔΕΙΓΜΑΤΑ ΧΡΗΣΗΣ ARDUINO ΣΕ SMARTCITY.....	12
1.6 ΕΠΙΛΟΓΗ ARDUINO NANO 33 BLE SENSE	14
ΘΕΩΡΙΑ	15
2 ARDUINO NANO 33 BLE SENSE.....	15
2.1 ΔΥΝΑΤΟΤΗΤΕΣ ARDUINO NANO 33 BLE SENSE	15
2.1 ΜΝΗΜΗ	16
2.2 ΕΠΕΞΕΡΓΑΣΤΗΣ	17
2.3 INERTIAL MEASUREMENT UNIT (IMU)	17
2.4 ΤΡΟΦΟΔΟΣΙΑ	17
2.5 IMU ΑΝΙΧΝΕΥΣΗ ΚΙΝΗΣΗΣ	18
2.6 ΜΙΚΡΟΦΩΝΟ	18
2.7 ΑΙΣΘΗΤΗΡΑΣ ΜΙΚΡΗΣ ΑΠΟΣΤΑΣΗΣ ΚΑΙ ΑΝΙΧΝΕΥΣΗ ΧΕΙΡΟΝΟΜΙΑΣ	18
2.8 ΑΙΣΘΗΤΗΡΑΣ ΘΕΡΜΟΚΡΑΣΙΑΣ ΚΑΙ ΥΓΡΑΣΙΑΣ	18
2.9 ΒΑΡΟΜΕΤΡΙΚΟΣ ΑΙΣΘΗΤΗΡΑΣ ΠΙΕΣΗΣ	18
2.10 ΚΡΥΠΤΟΓΡΑΦΙΚΟ ΤΣΙΠ	19
3 ESP-8266 WEMOS D1 MINI	19
3.1 ΠΕΡΙΓΡΑΦΗ ΠΛΑΚΕΤΑΣ ESP-8266	19
3.2 INTERRUPT	20
3.3 PWM (PULSE WIDTH MODULATION)	20
3.4 I2C	21
3.5 SPI (SERIAL PERIPHERAL INTERFACE)	21
3.6 1-WIRE	21

3.7 UART (UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER) ΑΚΙΔΕΣ	21
3.8 ΑΝΑΛΟΓΙΚΗ ΑΚΙΔΑ	21
3.9 MNHMH ESP-8266	22
4 MAX 30102	22
4.1 ΠΕΡΙΓΡΑΦΗ ΑΙΣΘΗΤΗΡΑ MAX30102	22
4.2 FIFO – FIRST IN FIRST OUT	23
4.3 ΔΙΑΚΟΠΕΣ (INTERRUPTS)	24
4.3.1 FIFO ALMOST FULL FLAG	24
4.3.2 NEW FIFO DATA READY	24
4.3.3 AMBIENT LIGHT CANCELLATION OVERFLOW	24
4.3.4 POWER READY FLAG	24
4.3.5 INTERNAL TEMPERATURE READY FLAG	25
4.4 ΨΗΦΙΑΚΕΣ ΘΥΡΕΣ ΑΙΣΘΗΤΗΡΑ	25
5 ΛΕΙΤΟΥΡΓΙΑ ΑΙΣΘΗΤΗΡΩΝ ΜΕΤΡΗΣΗΣ ΚΑΡΔΙΑΚΩΝ ΠΑΛΜΩΝ, ΟΞΥΓΟΝΟΥ ΚΑΙ ΚΑΡΔΙΟΓΡΑΦΗΜΑ	26
5.1 ΔΙΑΦΟΡΑ ΚΑΡΔΙΑΚΟΥ ΠΑΛΜΟΥ ΚΑΙ ΣΦΥΓΜΟΥ	26
5.2 ΦΩΤΟΠΛΗΘΥΣΜΟΓΡΑΦΙΑ ΚΑΙ ΚΑΡΔΙΑΚΟΣ ΠΑΛΜΟΣ	26
5.3 ΑΝΙΧΝΕΥΣΗ ΚΑΙ ΜΕΤΡΗΣΗ ΤΟΥ ΚΟΡΕΣΜΟΥ ΟΞΥΓΟΝΟΥ (SpO ₂) ΣΤΟ ΑΙΜΑ	28
5.4 ΗΛΕΚΤΡΟΚΑΡΔΙΟΓΡΑΦΙΑ ΚΑΙ ΚΑΡΔΙΑΚΗ ΛΕΙΤΟΥΡΓΙΑ	29
6 AD8232 HEART MONITOR	32
6.1 ΠΕΡΙΓΡΑΦΗ ΑΙΣΘΗΤΗΡΑ AD8232	32
6.2 ΗΛΕΚΤΡΟΔΙΑ, ΗΛΕΚΤΡΟΛΥΤΕΣ ΚΑΙ ΦΙΛΤΡΑΡΙΣΜΑ ΣΗΜΑΤΩΝ	33
6.3 ΕΝΙΣΧΥΤΗΣ ΔΕΞΙΟΥ ΠΟΔΙΟΥ	34
6.4 ΣΥΧΝΟΤΗΤΑ ΔΙΑΣΤΑΥΡΩΣΗΣ ΚΑΙ ΣΤΑΘΕΡΟΤΗΤΑ ΚΥΚΛΩΜΑΤΟΣ	34
6.5 ΛΕΙΤΟΥΡΓΙΑ ΓΡΗΓΟΡΗΣ ΑΠΟΚΑΤΑΣΤΑΣΗΣ	35
6.6 ΛΕΙΤΟΥΡΓΙΑ ΚΑΙ ΑΝΙΧΝΕΥΣΗ ΑΠΟΣΥΝΔΕΣΗΣ ΗΛΕΚΤΡΟΔΙΩΝ ΣΤΟ AD8232	36
6.7 ΕΝΙΣΧΥΤΗΣ ΟΡΓΑΝΩΝ	37
6.8 ΈΞΟΔΟΣ ΣΗΜΑΤΟΣ ΣΤΟ AD8232	37
6.9 REFERENCE BUFFER	38

6.10 ΑΝΤΙΣΤΑΣΕΙΣ PULL-UP KAI PULL-DOWN	38
7 MESSAGE QUEUING TELEMETRY TRANSPORT (MQTT)	
ΠΡΩΤΟΚΟΛΛΟ	40
7.1 ΠΕΡΙΓΡΑΦΗ ΠΡΩΤΟΚΟΛΛΟΥ MQTT	40
7.2 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΟΥ MQTT	40
7.3 ΘΕΜΑΤΑ (TOPICS)	42
7.4 ΔΙΑΤΗΡΟΥΜΕΝΟ ΜΗΝΥΜΑ (RETAINED MESSAGE)	43
7.5 ΤΕΣΣΕΡΙΣ ΒΑΣΙΚΕΣ ΕΝΕΡΓΕΙΕΣ ΤΟΥ ΠΡΩΤΟΚΟΛΛΟΥ MQTT ...	43
7.6 ΑΣΦΑΛΕΙΑ ΠΡΩΤΟΚΟΛΛΟΥ MQTT	44
8 ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ	45
8.1 ΙΣΤΟΡΙΑ ΥΠΟΛΟΓΙΣΤΙΚΗΣ ΌΡΑΣΗΣ	46
8.1.1 ΣΗΜΑΣΙΑ ΤΗΣ ΥΠΟΛΟΓΙΣΤΙΚΗΣ ΟΡΑΣΗΣ	47
8.1.2 ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ ΚΑΙ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ ΣΤΗΝ ΥΠΟΛΟΓΙΣΤΙΚΗ ΟΡΑΣΗ	48
8.2 ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ	49
8.2.1 ΜΕΘΟΔΟΙ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ	50
8.2.2 ΑΛΓΟΡΙΘΜΟΙ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ	52
8.3 ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ	53
8.3.1 ΤΥΠΟΙ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ	55
8.4 ΒΑΘΙΑ ΜΑΘΗΣΗ	56
8.4.1 ΔΙΑΦΟΡΕΣ ΜΗΧΑΝΙΚΗΣ ΜΕ ΒΑΘΙΑΣ ΜΑΘΗΣΗΣ	57
8.4.2 ΒΑΘΙΑ ΜΑΘΗΣΗ ΚΑΙ ΠΡΟΗΓΜΕΝΑ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ	58
8.4.3 ΑΠΑΙΤΗΣΕΙΣ ΚΑΙ ΕΦΑΡΜΟΓΕΣ ΒΑΘΙΑΣ ΜΑΘΗΣΗΣ	59
ΠΕΙΡΑΜΑΤΙΚΟ ΜΕΡΟΣ	60
9 ΣΚΟΠΟΣ ΠΕΙΡΑΜΑΤΙΚΟΥ ΜΕΡΟΥΣ	60
9.1 ΥΛΟΠΟΙΗΣΗ ΚΥΚΛΩΜΑΤΟΣ ΜΕΤΡΗΣΗΣ ΚΑΡΔΙΑΚΩΝ ΠΑΛΜΩΝ ΚΑΙ ΟΞΥΓΟΝΟΥ	60
9.2 ΥΛΟΠΟΙΗΣΗ ΚΥΚΛΩΜΑΤΟΣ ΑΠΕΙΚΟΝΙΣΗΣ ΚΑΡΔΙΟΓΡΑΦΗΜΑΤΟΣ	62
9.3 ΑΠΟΤΕΛΕΣΜΑΤΑ ΚΥΚΛΩΜΑΤΩΝ	63
9.3.1 ΑΠΟΤΕΛΕΣΜΑΤΑ ΕΦΑΡΜΟΓΗΣ ΚΑΡΔΙΑΚΩΝ ΠΑΛΜΩΝ	64
9.3.2 ΑΠΟΤΕΛΕΣΜΑΤΑ ΤΗΣ ΜΕΤΡΗΣΗΣ ΤΟΥ ΟΞΥΓΟΝΟΥ	67

9.3.3 ΑΠΟΤΕΛΕΣΜΑΤΑ ΑΠΕΙΚΟΝΙΣΗΣ	
ΗΛΕΚΤΡΟΚΑΡΔΙΟΓΡΑΦΗΜΑΤΟΣ	69
9.3.4 ΑΠΟΤΕΛΕΣΜΑΤΑ ΤΟΥ ΜΟΝΤΕΛΟΥ ΑΝΑΓΝΩΡΙΣΗΣ ΣΤΑΣΗΣ ΑΝΘΡΩΠΟΥ	72
9.4 ΜΟΝΤΕΛΟ ΑΝΑΓΝΩΡΙΣΗΣ ΣΤΑΣΗΣ ΚΑΙ ΜΕΤΡΗΣΗ ΠΑΛΜΩΝ	73
10 ΣΥΜΠΕΡΑΣΜΑΤΑ	78
ΠΑΡΑΡΤΗΜΑ Α: ΟΔΗΓΙΕΣ ΧΡΗΣΗΣ	79
A1 ΔΙΑΔΙΚΑΣΙΑ ΕΓΚΑΤΑΣΤΑΣΗΣ ΒΙΒΛΙΟΘΗΚΩΝ	79
A2 ΟΔΗΓΟΙ ΓΙΑ ΠΛΑΚΕΤΕΣ	80
A3 ΔΙΑΔΙΚΑΣΙΑ ΕΚΤΕΛΕΣΗΣ	82
A4 ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΕΦΑΡΜΟΓΗΣ IOTMQTTPANEL	84
A5 ΡΥΘΜΙΣΗ ΕΦΑΡΜΟΓΗΣ ΓΙΑ ΤΗΝ ΑΠΕΙΚΟΝΙΣΗ ΤΟΥ ΗΛΕΚΤΡΟΚΑΡΔΙΟΓΡΑΦΗΜΑΤΟΣ	87
A6 ΜΟΝΤΕΛΟ ΥΠΟΛΟΓΙΣΤΙΚΗΣ ΌΡΑΣΗΣ	88
ΠΑΡΑΡΤΗΜΑ Β: ΚΩΔΙΚΑΣ	98
B1 ΚΩΔΙΚΑΣ ARDUINO ΚΥΚΛΩΜΑΤΟΣ ΜΕΤΡΗΣΗΣ ΚΑΡΔΙΑΚΩΝ ΠΑΛΜΩΝ	98
B1.1 ΚΩΔΙΚΑΣ ESP-8266 ΚΥΚΛΩΜΑΤΟΣ ΜΕΤΡΗΣΗΣ ΠΑΛΜΩΝ..	102
B2 ΚΩΔΙΚΑΣ ARDUINO ΜΕΤΡΗΣΗ ΠΟΣΟΣΤΟΥ ΟΞΥΓΟΝΟΥ ΣΤΟ AIMΑ	106
B2.1 ΚΩΔΙΚΑΣ ESP-8266 ΚΥΚΛΩΜΑΤΟΣ ΜΕΤΡΗΣΗΣ ΟΞΥΓΟΝΟΥ	111
B3 ΚΩΔΙΚΑΣ ARDUINO ΑΠΕΙΚΟΝΙΣΗΣ ΗΛΕΚΤΡΟΚΑΡΔΙΟΓΡΑΦΗΜΑΤΟΣ	115
B3.1 ΚΩΔΙΚΑΣ ΓΙΑ ΤΟ ESP8266 ΕΦΑΡΜΟΓΗΣ ΗΛΕΚΤΡΟΚΑΡΔΙΟΓΡΑΦΗΜΑΤΟΣ.....	116
B4 ΚΩΔΙΚΑΣ ARDUINO ΜΟΝΤΕΛΟΥ ΑΝΑΓΝΩΡΙΣΗΣ ΣΤΑΣΗΣ	120
B5 ΚΩΔΙΚΑΣ ARDUINO ΜΟΝΤΕΛΟΥ ΑΝΑΓΝΩΡΙΣΗΣ ΚΑΙ ΜΕΤΡΗΣΗΣ ΠΑΛΜΩΝ	153
ΒΙΒΛΙΟΓΡΑΦΙΑ	98

Εισαγωγή

Στην καθημερινότητα των ηλικιωμένων ανθρώπων ή ασθενών ή ανθρώπων με μακροχρόνιες παθήσεις, ή άτομα που είναι απομακρυσμένα από τα κέντα υγείας, νοσοκομεία ή δεν έχουν άμεση πρόσβαση σε αυτά, όπως σε ένα νησί, είναι συχνό φαινόμενο να μην παρακολουθούνται ιατρικά ανά τακτά χρονικά διαστήματα και το ιστορικό τους να είναι ελλιπές. Στις έξυπνες πόλεις, η υποστήριξη των ηλικιωμένων και των ατόμων με χρόνιες παθήσεις αποτελεί βασική προτεραιότητα, αξιοποιώντας τεχνολογίες όπως η τηλεϊατρική, οι φορητές συσκευές και η απομακρυσμένη παρακολούθηση υγείας. Στην Ελλάδα, ένα χαρακτηριστικό παράδειγμα είναι το e-Trikala[1], το οποίο εφαρμόζει λύσεις όπως τηλεϊατρικές υπηρεσίες και απομακρυσμένη παρακολούθηση ασθενών, την πρόσβασή τους στις υπηρεσίες υγείας βελτιώνοντας την ποιότητα ζωής των πολιτών, γενικότερα.

Η λύση που παρατίθεται είναι η υλοποίηση ενός συστήματος το οποίο παρακολουθεί τον ασθενή με μια κάμερα για την εύρεση στάσης σώματος, δηλαδή αν στέκεται όρθιος, αν κάθεται ή αν βρίσκεται ξαπλωμένος, σε συνδυασμό με έναν αισθητήρα καρδιακών παλμών και να εμφανίζει ένα μήνυμα αποτελέσματος των αισθητήρων στον χρήστη. Επίσης, υλοποιήθηκαν εφαρμογές για τη μέτρηση του κορεσμένου οξυγόνου του αίματος, τη μέτρηση καρδιακών παλμών και την παρακολούθηση καρδιογραφήματος σε ζωντανή μετάδοση δεδομένων μέσω του διαδικτύου απεικονίζοντάς τα σε μια εφαρμογή στο κινητό για να μπορεί ο αναγνώστης να τα διαβάσει απομακρυσμένα. Αφού είναι απαραίτητη η ζωντανή μετάδωση δεδομένων, για την αναγνώριση και παρακολούθηση σοβαρών καταστάσεων. Οι εφαρμογές αυτές έχουν ως αποτέλεσμα την πρόληψη επειγουσών καταστάσεων, πρώιμη ανίχνευση προβλημάτων όπως καρδιολογικά, αναπνευστικά και παρακολούθηση ασθενειών σε κρίσιμες καταστάσεις.

1 Περιγραφή Arduino

Το Arduino[2] είναι μια ανοιχτού κώδικα πλατφόρμα βασισμένη στην εύκολη χρήση του υλικού και λογισμικού. Οι πλακέτες Arduino μπορούν να διαβάσουν τις εισόδους όπως το φως από έναν αισθητήρα ή το δάχτυλο σε ένα κουμπί και να τις μετατρέψουν σε εξόδους οπλίζοντας έναν ρελέ, ανάβοντας ένα LED, ενεργοποιώντας έναν κινητήρα. Επίσης, ο χρήστης έχει την δυνατότητα να γράψει μια σειρά εντολών και να τις στείλει στον μικροελεγκτή της πλακέτας, για να υλοποιήσει μια εφαρμογή. Για τον προγραμματισμό του Arduino χρησιμοποιείτε η γλώσσα Arduino που είναι βασισμένη στο Wiring [3] και το λογισμικό Arduino (IDE) που βασίζεται στο Processing [4].

Με την πάροδο των χρόνων, το Arduino έχει κινήσει την περιέργεια σε φοιτητές, χομπίστες, προγραμματιστές, επαγγελματίες και καλλιτέχνες. Οι συνεισφορές τους έχουν προσθέσει ένα τεράστιο όγκο προσβάσιμης γνώσης που μπορεί να βοηθήσει τους αρχάριους και τους πιο προχωρημένους είτε να εξελίξουν μια εφαρμογή είτε να αποκτήσουν γνώση πάνω στο αντικείμενο υλοποίησης.

1.1 Σκοπός δημιουργίας του Arduino

Το Arduino δημιουργήθηκε στο ίδρυμα σχεδιασμού της Ιβρέα (Ivrea) στη Ιταλία, ως ένα εύκολο εργαλείο για την δημιουργία γρήγορων πρωτότυπων και στόχευτες τους μαθητές που δεν είχαν γνώση στην ηλεκτρονική ή στον προγραμματισμό. Μόλις διαδόθηκε σε μια ευρύτερη κοινότητα τότε άρχισε να προσαρμόζεται στις νέες ανάγκες και προκλήσεις. Διαφοροποιώντας τις δυνατότητές της από απλές πλακέτες των 8-bit, σε προϊόντα για το διαδίκτυο των πραγμάτων (IoT) και εφαρμογές, 3D εκτυπωτές και ενσωματωμένα συστήματα.

1.2 Δυνατότητες Arduino

Η φιλική προς τον χρήστη και η απλότητα του έχει οδηγήσει στη χρήση του σε αμέτρητα έργα και εφαρμογές. Το λογισμικό Arduino έχει σχεδιαστεί για να είναι προσβάσιμο για αρχάριους, ενώ προσφέρει επίσης αρκετή ευελιξία για πιο έμπειρους χρήστες. Είναι συμβατό με λειτουργικά συστήματα Mac, Windows και Linux. Οι εκπαιδευτικοί και οι μαθητές το χρησιμοποιούν για να δημιουργήσουν οικονομικά επιστημονικά όργανα, να επιδείξουν αρχές της χημείας και της φυσικής ή να ξεκινήσουν τα ταξίδια τους στον προγραμματισμό και τη ρομποτική. Αρχιτέκτονες και σχεδιαστές κατασκευάζουν διαδραστικά πρωτότυπα, ενώ μουσικοί και καλλιτέχνες το χρησιμοποιούν για εγκαταστάσεις και για να εξερευνήσουν καινοτόμα μουσικά όργανα. Οι κατασκευαστές, φυσικά, το αξιοποιούν για να κατασκευάσουν πολλά από τα έργα που παρουσιάζονται σε εκδηλώσεις όπως το Maker Faire. Το Arduino χρησιμεύει ως βασικό εργαλείο για την απόκτηση νέας γνώσης. Άτομα όλων των ηλικιών, ακόμη και παιδιά μπορούν να αρχίσουν να πειραματίζονται απλά ακολουθώντας τις οδηγίες βήμα προς βήμα ενός κιτ ή αλληλοεπιδρώντας με άλλα μέλη της κοινότητας του Arduino στο διαδίκτυο.

1.3 Πλεονεκτήματα Arduino

Υπάρχουν πολλοί ακόμη μικροελεγκτές και πλατφόρμες μικροελεγκτών διαθέσιμες για το Physical Computing, όπως οι Parallax Basic Stamp, Netmedia's BX-24, Phidgets και άλλες με παρόμοια χαρακτηριστικά. Όλες αυτές οι λύσεις διευκολύνουν τις πολύπλοκες πτυχές του προγραμματισμού μικροελεγκτών, προσφέροντας ένα φιλικό και εύχρηστο περιβάλλον. Το Arduino επίσης απλοποιεί την εργασία με μικροελεγκτές, παρέχοντας όμως επιπλέον πλεονεκτήματα για εκπαιδευτικούς, μαθητές και ερασιτέχνες, σε σύγκριση με άλλες πλατφόρμες:

- **Οικονομικό:** Οι πλακέτες Arduino είναι οικονομικότερες σε σχέση με άλλες πλατφόρμες μικροελεγκτών. Η φθηνότερη έκδοση του Arduino μπορεί να συναρμολογηθεί με το χέρι ακόμη και οι προσυναρμολογημένες πλακέτες κοστίζουν λιγότερο από πενήντα ευρώ.
- **Πολλαπλές πλατφόρμες:** Το λογισμικό Arduino (IDE) λειτουργεί σε όλες τις πλατφόρμες Windows, Mac και Linux.
- **Λογισμικό ανοιχτού κώδικα και επεκτάσιμο:** Το λογισμικό του Arduino δημοσιεύεται ως ένα εργαλείο ανοιχτού κώδικα και είναι διαθέσιμο σε έμπειρους και επαγγελματίες προγραμματιστές. Η γλώσσα μπορεί να επεκταθεί με βιβλιοθήκες C++ ή με AVR-C, στη βάση της οποίας είναι χτισμένο το Arduino.
- **Υλικό ανοιχτού κώδικα και επεκτάσιμο:** Τα σχέδια των πλακετών του Arduino είναι δημόσια με άδεια Creative Commons με σκοπό την επέκτασή τους από άτομα που έχουν εμπειρία στη σχεδίαση κυκλωμάτων. Τους δίνεται η δυνατότητα να σχεδιάσουν την δική τους έκδοση του εξαρτήματος [2].

1.4 Εκδοχές Arduino

Υπάρχουν πολλές εκδοχές του Arduino και κάθε μία καλύπτει κάποιες συγκεκριμένες ανάγκες. Κάποιες κατηγορίες Arduino είναι:

- Arduino Uno
- Arduino Nano
- Arduino Micro
- Arduino Mega
- Arduino MKR Shield
- Arduino Shield

Η κάθε κατηγορία έχει υποκατηγορίες με Arduino για να υλοποιούν έναν συγκεκριμένο σκοπό. Για παράδειγμα, το Arduino Uno R3 υλοποιεί απλά συστήματα, ενώ το Arduino Uno R4 Wi-Fi έχει και τη δυνατότητα την σύνδεση στο διαδίκτυο, για την επέκταση των απλών εφαρμογών στο διαδίκτυο. Η κατηγορία των Arduino Nano, σε σχέση με Arduino Uno, παρόλο που είναι πιο μικρά σε μέγεθος εστιάζει στην υπολογιστική ισχύ και σε εφαρμογές του διαδικτύου πραγμάτων (IoT). Ανάλογα με τις ανάγκες που θέλει η εφαρμογή υλοποίησης, επιλέγεται το κατάλληλο Arduino. Επιπλέον, διατίθενται πλακέτα Arduino, που ονομάζονται κιτ, και τα οποία περιέχουν διάφορα εξαρτήματα όπως τρανζίστορ, αντιστάσεις, LED, καλώδια και αισθητήρες.

1.5 Παραδείγματα χρήσης Arduino σε Smart City

Τα τελευταία χρόνια, η χρήση πλατφορμών όπως το Arduino έχει αναδειχθεί ως βασικός πυλώνας σε έργα έξυπνων πόλεων (Smart Cities). Ενδικτικά, εφαρμογές Arduino όπως:

- Η παρακολούθιση ποιότητας αέρα, όπου εξετάζεται η ποιότητα του αέρα, ανιχνεύται η ηγρασία και η θερμοκρασία [6].
- Η έξηπνη διαχίριση φωτισμού δρόμων στην οποία η φοτηνότητα ανιχνεύται και ανάλογα ανίγουν και κλίνουν τα φώτα στους δρόμους [7].
- Συστημα προειδοποίησης πλημμύρας, όπου με έναν αισθητηρα ανιχνεύεται η στάθμη του νερού και στέλνει στο τηλέφονο ένα μήνυμα ειδοποίησης [8].

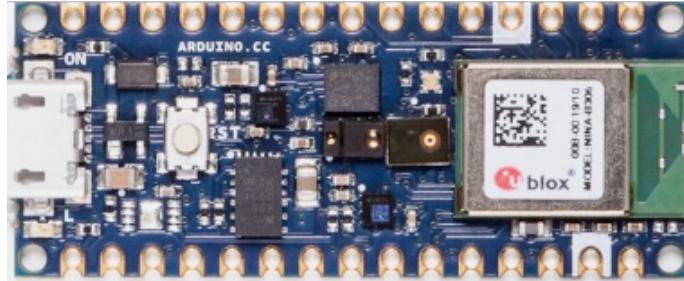
1.6 Επιλογή Arduino nano 33 Ble Sense

Το πειραματικό μέρος υλοποιήθηκε με την πλακέτα Arduino Nano 33 Ble Sense. Επιλέχθηκε το συγκεκριμένο Arduino διότι είναι ένα από τα λίγα που μπορεί να υποστηρίξει μηχανική μάθηση και υπολογιστική όραση. Ακόμα, υπάρχει διευκόλυνση στην πρόσβαση και χρήση της κάμερας, αφού είναι συμβατό με μια πλακέτα (Arduino Tiny Machine Learning Shield) που

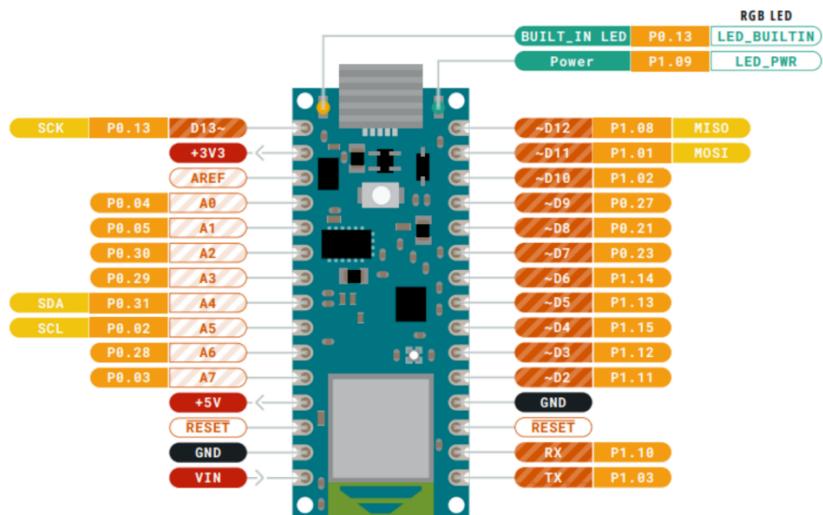
συνδέει την κάμερα με το Arduino Nano 33 Ble Sense και διαθέτει επιπλέον θύρες SCL, SDA, VCC, GND, A6, A7, D11 και D12 για την προσθήκη άλλης πλακέτας ή αισθητήρα. Επίσης, όπως εξηγείται και παρακάτω, υπάρχουν αρκετοί αισθητήρες που μπορούν να συνδυαστούν για την εξέλιξη του μοντέλου.

ΘΕΩΡΙΑ

2 Arduino Nano 33 BLE Sense



Εικόνα 2.1: Arduino Nano 33 BLE sense [9]



Σχήμα 2.1: Arduino Nano 33 BLE sense pin out [10].

2.1 Δυνατότητες Arduino Nano 33 Ble Sense

Για το πείραμα χρησιμοποιήθηκε το Arduino Nano 33 BLE Sense το οποίο έχει πολλούς διάφορους αισθητήρες περιβάλλοντος και

κυρίως επιλέχθηκε διότι υπάρχει δυνατότητα να τρέξει προγράμματα τεχνητής νοημοσύνης (TinyML, TensorFlow Lite) σε συνδυασμό με κάμερα OV7675. Επίσης υποστηρίζει Bluetooth 5 Low Energy 2,4 GHz με εσωτερική κεραία που μπορεί να χρησιμοποιηθεί για την μεταφορά δεδομένων μεταξύ συσκευών χρησιμοποιώντας τη βιβλιοθήκη ArduinoBLE. Διαθέτει έναν nRF52840 μικροελεγκτή και τρέχει σε επεξεργαστή Arm Cortex-M4F [11].

Ακόμα προσφέρει δεκατέσσερις ψηφιακές θύρες για είσοδο και έξοδο. Εκ των οποίων οι δύο είναι για σειριακή επικοινωνία δηλαδή υπάρχει μια ακίδα RX για να δέχεται και μια ακίδα TX για να στέλνει τα δεδομένα. Αυτές υποστηρίζονται από το πρωτόκολλο USART. Επίσης, υποστηρίζει πρωτόκολλο PWM στις ακίδες D3, D5, D6, D9, D10 και πρωτόκολλο SPI MOSI, SPI MISO στις D11 και D12 αντίστοιχα. Το πρωτόκολλο SPI MOSI (Master Out - Slave In) επιτρέπει στον (κυρίαρχο) κόμβος να στέλνει δεδομένα και λαμβάνει από όλους τους κόμβους (σκλάβοι). Ομοίως, το πρωτόκολλο SPI MISO (Master In - Slave Out) επιτρέπει σε έναν (σκλάβο) κόμβο να στέλνει δεδομένα στον (κυρίαρχο) κόμβο [12].

Παρέχει εννέα αναλογικές ακίδες εκ των οποίων μία θύρα υποστηρίζει SDA (Serial Data) και μία για SCL (Serial Clock) με το πρωτόκολλο I2C. Επιπρόσθετα, διαθέτει μια ακίδα AERF (Analog Reference) που μπορεί να χρησιμοποιηθεί για την παροχή εξωτερικής τάσης αναφοράς για την αναλογική σε ψηφιακή μετατροπή των εισόδων στους αναλογικούς ακροδέκτες. Δηλαδή, η τάση αναφοράς καθορίζει την τιμή για την κορυφή του εύρος εισόδου με αποτέλεσμα κάθε διακριτό βήμα στη μετατρεπόμενη έξοδο [13].

2.1 Μνήμη

Η μνήμη που έχει είναι μη πτητική (non-volatile Ram), χωρητικότητας 1 MB. Στη μη πτητική μνήμη υπάρχει δυνατότητα

τα δεδομένα να αποθηκεύονται μόνιμα και να παραμένουν εκεί ακόμα και αν δεν υπάρχει τροφοδοσία ρεύματος [14]. Επιπλών, έχει 256 KB μνήμη που χρησιμοποιείται από τον μικροελεγκτή για την προσωρινή αποθήκευση των δεδομένων κατά την εκτέλεση όπως είναι οι μεταβλητές [9].

2.2 Επεξεργαστής

Ο κύριος επεξεργαστής που διαθέτει είναι ο Arm Cortex-M4F και έχει συχνότητα λειτουργίας τα 64 MHz . Οι περισσότερες ακίδες είναι συνδεδεμένες με εξωτερικές κεφαλές, ωστόσο κάποιες άλλες διατίθενται για εσωτερικές επικοινωνίες μαζί με ασύρματα εξαρτήματα και την ενσωματωμένη I2C επικοινωνία περιφερειακών όπως IMU. Επιπλέων, σε αντίθεση με τις άλλες πλακέτες Arduino Nano οι ακίδες A5 και A4 έχουν μια εσωτερική αντίσταση (pull-up) και από προεπιλογή χρησιμοποιούν το I2C κανάλι. Με αποτέλεσμα, σύμφωνα με τον κατασκευαστή, η χρήση τους ως αναλογικού σήματος εισόδου να μην συνιστάται [9].

2.3 Inertial Measurement Unit (IMU)

Το Arduino Nano 33 Ble Sense περιέχει έναν ενσωματωμένο αισθητήρα IMU 9 αξόνων που μπορεί να χρησιμοποιηθεί για τον προσανατολισμό της πλακέτας (εξετάζοντας το διάνυσμα της επιτάχυνσης της βαρύτητας ή τη χρήση της 3D πυξίδας) ή τη μέτρηση κραδασμών, της επιτάχυνσης, των δονήσεων και την ταχύτητα περιστροφής [9].

2.4 Τροφοδοσία

Η πλακέτα μπορεί να τροφοδοτηθεί με USB ή με κάποια εξωτερική πηγή ρεύματος απευθείας στις κατάλληλες ακίδες. Το Arduino Nano 33 Ble Sense υποστηρίζει 3.3V για εισόδους/εξόδους (I/Os) και δεν είναι συμβατό με 5V. Επομένως πρέπει να υπάρχει προσοχή στα σήματα να μην συνδεθούν απευθείας στην πλακέτα, γιατί θα προκληθεί ζημιά. Επίσης σε αντίθεση με τις άλλες πλακέτες που υποστηρίζουν 5V, στη

συγκεκριμένη πλακέτα η ακίδα 5V που λειτουργεί σαν είσοδος, δεν διαθέτει τάση και συνδέεται με την τροφοδοσία USB [9].

2.5 IMU Ανίχνευση κίνησης

Διαθέτει αισθητήρα αδράνειας LSM9DS1 ο οποίος έχει ιδιαίτερα χαρακτηριστικά όπως 3D επιταχυνσιόμετρο, γυροσκόπιο και μαγνητόμετρο. Με τα παραπάνω χαρακτηριστικά επιτρέπεται η εύρεση προσανατολισμού, κίνησης ή δόνησης [15].

2.6 Μικρόφωνο

Υπάρχει μικρόφωνο MP34DT05 που λαμβάνει σε όλες τις κατευθύνσεις, για συλλογή και ανάλυση ήχου σε ζωντανή μετάδοση για τη δημιουργία φωνητικής διεπαφής, με την κλήση της βιβλιοθήκης PDM [16].

2.7 Αισθητήρας μικρής απόστασης και ανίχνευση χειρονομίας

Ο αισθητήρας APDS9960 μπορεί να χρησιμοποιηθεί για να εκτυπώσει μια απλή οδηγία μιας χειρονομίας και να ελέγξει ανάλογα με το RGB LED της πλακέτας. Επιπλέον, μπορεί να προγραμματιστεί το Arduino έτσι ώστε το ενσωματωμένο LED να αλλάζει χρώματα στο RGB LED σύμφωνα με την κατεύθυνση της χειρονομίας. Ο κώδικας μπορεί να διαβάζει απλές κινήσεις χεριών όπως πάνω-κάτω-δεξιά-αριστερά [17].

2.8 Αισθητήρας θερμοκρασίας και υγρασίας

Με τη χρήση της βιβλιοθήκης HTS221 αποτυπώνεται η θερμοκρασία και η υγρασία σε πραγματικό χρόνο του δωματίου. Ο αισθητήρας θερμοκρασίας μπορεί να κυμανθεί από -40 έως 120 βαθμούς Κελσίου, αντίστοιχα για την υγρασία από 0 έως 100% [18].

2.9 Βαρομετρικός αισθητήρας πίεσης

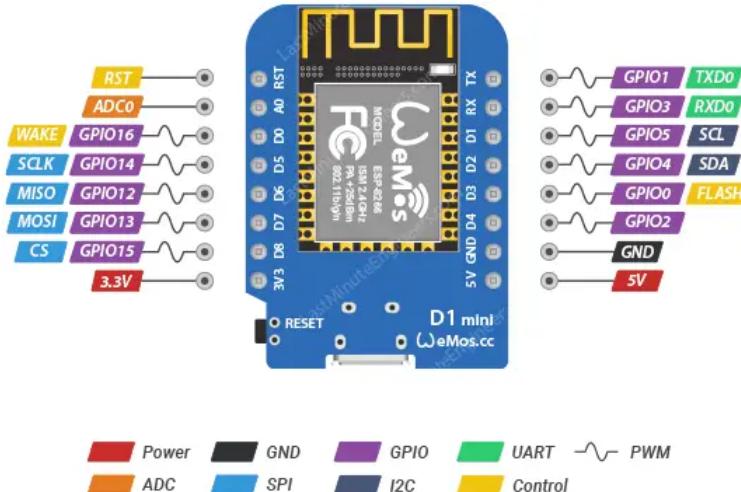
Με τον αισθητήρα LP22HB ανιχνεύεται κατά προσέγγιση το υψόμετρο από το επίπεδο της θάλασσας με τη χρήση της

ατμοσφαιρικής πίεσης. Επιπλέον μπορεί να προβλέψει τον καιρό σε βραχυπρόθεσμο χρόνο [19].

2.10 Κρυπτογραφικό τσιπ

Με το κρυπτογραφικό τσιπ παρέχεται ασφαλής τρόπο αποθήκευσης όπως πιστοποιητικά και διευκολύνει την εκτέλεση ασφαλών πρωτοκόλλων, χωρίς να αποκαλύπτει τα μυστικά σε μη κρυπτογραφημένη μορφή [9].

3 ESP-8266 Wemos D1 mini



Σχήμα 3.1: Esp-8266 Wemos D1 mini [20]

3.1 Περιγραφή πλακέτας Esp-8266

Το Esp-8266 είναι μια πλακέτα η οποία μπορεί να προγραμματιστεί εύκολα και έχει την ικανότητα να συνδεθεί με το διαδίκτυο. Επίσης χρησιμοποιείται για εφαρμογές στο διαδίκτυο των πραγμάτων. Επιπλέον διαθέτει μια κεραία Wi-Fi στο τσιπ ESP8266EX [21] και υποστηρίζει το ασύρματο πρωτόκολλο επικοινωνίας 802.11b/g/n επιτρέποντας σύνδεση με ασύρματο router μόλις ενεργοποιηθεί. Μπορεί να τροφοδοτηθεί χρησιμοποιώντας τη θύρα USB είτε μέσω της ακίδας 3.3V ή 5V

από κάποια εξωτερική πηγή ρεύματος. Τα δεδομένα φορτώνονται μέσω της θύρας USB και είναι συμβατό με το Arduino IDE πρόγραμμα. Παρέχει 11 ψηφιακές ακίδες εισόδου/έξόδου (εκτός από τη D0) υποστηρίζουν πρωτόκολλα interrupt, PWM, I2C, 1Wire και είναι συμβατό με SPI [22].

3.2 Interrupt

Όλες οι ακίδες GPIO (εκτός από την GPIO16) μπορούν να ρυθμιστούν ώστε να ενεργοποιούν μια διακοπή σε περίπτωση αύξησης, μείωσης ή αλλαγής της κατάστασης. Αυτή η δυνατότητα είναι καθοριστική για εργασίες που εξαρτώνται από γεγονότα, όπως η αντίδραση σε πάτημα κουμπιού ή η ανίχνευση σήματος από αισθητήρα [20].

3.3 PWM (Pulse Width Modulation)

Σχεδόν όλες οι ψηφιακές ακίδες εκτός της ακίδας D0 υποστηρίζονται από PWM διαμόρφωση. Η διαμόρφωση αυτή χρησιμεύει στη διαχείριση των στροφών ενός κινητήρα, στη ρύθμιση φωτεινότητας ενός LED και στην μεταφορά δεδομένων σε συστήματα με αισθητήρες ή ελεγκτές [20]. Η διαμόρφωση πλάτους παλμού (Pulse Width Modulation - PWM) είναι μια τεχνική για την επίτευξη αναλογικών αποτελεσμάτων μέσω ψηφιακών σημάτων. Λειτουργεί δημιουργώντας ένα τετραγωνικό κύμα, δηλαδή ένα σήμα που εναλλάσσεται μεταξύ των καταστάσεων ενεργοποίησης (on) και απενεργοποίησης (off). Αυτό το μοτίβο ενεργοποίησης-απενεργοποίησης μπορεί να προσομοιώσει τάσεις μεταξύ της μέγιστης τάσης λειτουργίας της πλακέτας (π.χ., 5V σε μια πλακέτα UNO, 3.3V σε μια πλακέτα Nano) και του 0V, ρυθμίζοντας την αναλογία του χρόνου που το σήμα παραμένει ενεργό σε σχέση με τον χρόνο που παραμένει ανενεργό. Η διάρκεια της κατάστασης ενεργοποίησης ονομάζεται πλάτος παλμού. Για να παραχθούν διαφορετικές αναλογικές τιμές, τροποποιείται (ή διαμορφώνεται) το πλάτος αυτού του παλμού. Όταν αυτός ο κύκλος

ενεργοποίησης-απενεργοποίησης επαναλαμβάνεται γρήγορα, για παράδειγμα σε ένα LED, δημιουργεί την εντύπωση μιας σταθερής τάσης μεταξύ 0 και Vcc (5V ή 3.3V) [23].

3.4 I2C

Το I2C είναι ένα πρωτόκολλο επικοινωνίας που εφαρμόζεται κυρίως μεταξύ συσκευών με μικρές αποστάσεις και χαμηλές ταχύτητες. Το πρωτόκολλο I2C επιτρέπει την κατευθυνόμενη επικοινωνία με αποθήκευση. Αυτό επιτυγχάνεται με δύο ακροδέκτες, το SCL (Serial Clock) που συγχρονίζει τη μεταφορά δεδομένων μεταξύ των δύο τσιπς και το SDA (Serial Data) που μεταφέρει δεδομένα. Τα πλεονεκτήματα αυτού του πρωτοκόλλου είναι η μειωμένη κατανάλωση ρεύματος και ο μικρότερος αριθμός γραμμών επικοινωνίας [24].

3.5 SPI (Serial Peripheral Interface)

Το SPI (Serial Peripheral Interface) είναι ένα πρωτόκολλο που επιτρέπει τη σειριακή, σύγχρονη αμφίδρομη επικοινωνία. Χρησιμοποιείται για την επικοινωνία μεταξύ περιφερειακών συσκευών και μικροελεγκτών για τη μεταφορά των δεδομένων με υψηλές ταχύτητες [25].

3.6 1-Wire

Το 1-Wire είναι ένα πρωτόκολλο που χρησιμοποιεί μόνο ένα καλώδιο μεταξύ μικροελεγκτή και περιφερειακής συσκευής [26].

3.7 UART (Universal Asynchronous Receiver/Transmitter) ακίδες

Διαθέτη μια ακίδα δέκτη και μια ακίδα πομπού για να επιτυγχάνεται η αποστολή ή η λήψη δεδομένων μέσω σειριακής επικοινωνίας από και προς έναν αισθητήρα ή υπολογιστή.

3.8 Αναλογική ακίδα

Ακόμα παρέχει μια αναλογική είσοδο A0 που μπορεί να μετρήσει τιμές από 0 έως 3.3V. Η A0 είναι συνδεδεμένη με έναν ενσωματωμένο μετατροπέα αναλογικού σε ψηφιακό (SAR ADC)

με ανάλυση 10-bit, επιτρέποντας την αναγνώριση 1024 διαφορετικών επιπέδων τάσης. Δηλαδή, μετατρέπει τις τάσεις εισόδου από 0 έως 3.3V (την τάση λειτουργίας του) σε ακέραιες τιμές από 0 έως 1024. Αυτό παρέχει ανάλυση περίπου 0.0032V (3.2mV) ανά μονάδα, υπολογισμένο ως 3.3V δια 1024 [20].

3.9 Μνήμη ESP-8266

Το ESP-8266 προσφέρει 4MB μη πτητική μνήμη [27], (βλέπε μνήμη Arduino Nano 33 Ble Sence).

4 Max 30102

4.1 Περιγραφή αισθητήρα Max30102

Ο αισθητήρα Max30102 χρησιμοποιείται για την μέτρηση καρδιακών παλμών και του οξυγόνου στο αίμα. Βασίζεται στη I2C ψηφιακή διεπαφή επικοινωνίας με τον μικροελεγκτή. Συνδυάζει δύο LED, ένα φωτοανιχνευτή, βελτιστοποιημένη οπτική και επεξεργασία αναλογικού σήματος χαμηλού θορύβου για την ανίχνευση σημάτων κορεσμού περιφερειακού τριχοειδούς οξυγόνου (SpO2) και καρδιακού ρυθμού (HR).



Σχήμα 4.1: Max30102

Στα δεξιά του αισθητήρα Max30100 υπάρχουν δύο LED ένα κόκκινο (RED) και μια δίοδο εκπομπής υπέρυθρο φωτός (IR). Στο αριστερό μέρος υπάρχει ο φωτοανιχνευτής.

Ο αισθητήρας μπορεί να τροφοδοτηθεί με 5V, 3.3V ή 1.8V (είσοδο/έξοδο). Αυτό συμβαίνει διότι το IC απαιτεί τροφοδοσία 1.8V και 3.3V τροφοδοσία για τα δύο LED (RED & IR LED). Το χαρακτηριστικό που τον κάνει ιδιαίτερο είναι η χαμηλή κατανάλωση ενέργειας. Καταναλώνει 600μΑ κατά την διάρκεια μιας μέτρησης και υπάρχει δυνατότητα να τεθεί σε κατάσταση αναμονής που η κατανάλωση του είναι 0.7μΑ. Εξαιτίας αυτής της χαμηλής κατανάλωσης μπορεί να χρησιμοποιηθεί φορητές συσκευές, όπως έξυπνα ρολόγια[28].

Το Max30102 διαθέτει στο τσιπ και αισθητήρα θερμοκρασίας ο οποίος μετρά θερμοκρασίες από -40°C έως +85°C με ακρίβεια ±1°C.

Το Max30102 μπορεί να ρυθμιστεί πλήρως μέσω λογισμικού καταχωρητών και τα ψηφιακά δεδομένα εξόδου του μπορούν να αποθηκευτούν σε μια προσωρινή μνήμη FIFO (First In, First Out) 32 θέσεων. Η προσωρινή μνήμη (buffer) FIFO επιτρέπει στο Max30102 να είναι συνδεδεμένο με έναν μικροελεγκτή ή επεξεργαστή σε κοινό δίαυλο, όπου τα δεδομένα δεν διαβάζονται συνεχώς από τους καταχωρητές του Max30102 [29]. Με αποτέλεσμα την εξοικονόμηση ενέργειας.

4.2 FIFO – First In First Out

Το FIFO είναι μια ουρά ένα είδος τύπου προσωρινής μνήμης (buffer). Το FIFO σημαίνει “First In, First Out”, δηλαδή ο πρώτος που θα εισαχθεί στην ουρά, θα εξέλθει και πρώτος. Η ουρά είναι μια γραμμική δομή δεδομένων στην οποία τα δεδομένα της αποθηκεύονται διαδοχικά, το ένα μετά το άλλο. Με αποτέλεσμα, η ουρά να είναι μια μονοδιάστατη δομή δεδομένων όσον αφορά την αποθήκευσή της στην κύρια μνήμη. Στην ουρά υπάρχει

δυνατότητα εισαγωγής και διαγραφής, ξεχωριστά στην αρχή και στο τέλος της λίστας [30].

4.3 Διακοπές (Interrupts)

Το Max30102 υποστηρίζει διακόπτες (Interrupts). Οι διακόπτες επιτρέπουν στον μικροελεκτή να εκτελεί δύο ή περισσότερες διεργασίες ανάλογα με την προτεραιότητα της κάθε μίας. Η διακοπή είναι ένα σήμα που εκπέμπεται από το λογισμικό ή το υλικό όταν μια διεργασία ή συμβάν χρειάζεται άμεση εκτέλεση [31]. Με αποτέλεσμα, ο μικροελεγκτής να ειδοποιείται για την διακοπή της εκτελούμενης διεργασίας και την εξυπηρέτηση της νέας διεργασίας που πρέπει να εκτελεστεί. Μετά από την εκτέλεση της νέας διεργασίας, ο μικροελεκτής επιστρέφει στην προηγούμενη διεργασία. Το Max30102, συγκεκριμένα παρέχει πέντε διαφορετικές διακοπές, που περιγράφονται παρακάτω.

4.3.1 FIFO Almost full flag

Στη μέτρηση κορεσμού του οξυγόνου (SpO₂) και καρδιακών παλμών (HR) η διακοπή ενεργοποιείται όταν ο δείκτης εγγραφής της ουράς FIFO έχει κάποιο συγκεκριμένο αριθμό ελεύθερων θέσεων [29].

4.3.2 New FIFO Data Ready

Στη μέτρηση SpO₂ και HR η διακοπή ενεργοποιείται όταν υπάρχει ένα νέο δείγμα δεδομένων στην ουρά FIFO.

4.3.3 Ambient Light Cancellation Overflow

Στην περίπτωση αυτή η διακοπή ενεργοποιείται όταν η λειτουργία διακοπής περιβάλλοντος φωτισμού της φωτοδιόδου από SpO₂/HR έχει φτάσει στο μέγιστο όριο της. Επίσης, το περιβάλλοντα φως επηρεάζει την έξοδο Analog to Digital Converter (ADC) .

4.3.4 Power Ready Flag

Κατά την ενεργοποίηση ή μετά από μια πτώση τάσης όταν η πηγή τάσης V_{DD} μεταβαίνει από κάτω από την ασφαλή χαμηλή τάση

σε επίπεδο πάνω από την ασφαλή χαμηλή τάση, η διακοπή ενεργοποιείται στέλνοντας ένα σήμα ότι η μονάδα έχει ενεργοποιηθεί και είναι έτοιμη να συλλέξει δεδομένα.

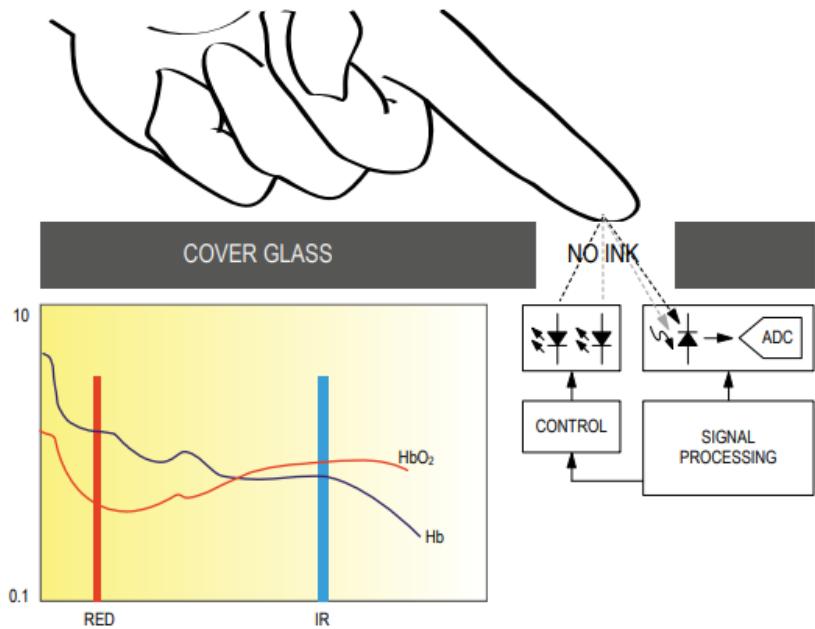
4.3.5 Internal Temperature Ready Flag

Όταν ολοκληρωθεί μια εσωτερική μετατροπή θερμοκρασίας του ολοκληρωμένου συστήματος (die), η διακοπή ενεργοποιείται για να μπορεί ο επεξεργαστής να διαβάσει τους καταχωρητές δεδομένων θερμοκρασίας [29].

4.4 Ψηφιακές θύρες αισθητήρα

Παρέχει δύο ψηφιακές θύρες SCL (serial clock), SDA (serial data) για τον συγχρονισμό του ρολογιού και την αμφίδρομη μεταφορά των δεδομένων μέσω του πρωτοκόλλου επικοινωνίας I2C. Μια θύρα γείωσης, μια θύρα INT (Active-Low Interrupt) χρησιμοποιεί ανοιχτή αποστράγγιση (Open-Drain) διάταξη που σημαίνει ότι όταν η διακοπή ενεργοποιείται το σήμα εξόδου γίνεται λογικό 0 ή χαμηλή τάση. Το Open-Drain είναι ένας τύπος εξόδου και για να λειτουργήσει απαιτείται μια εξωτερική πηγή τάσης και αντίσταση pull-up, ώστε η έξοδος όταν η διακοπή δεν είναι ενεργοποιημένη να γίνεται λογικό 1 ή τάση τροφοδοσίας VDD. Επιπλέον έχει μια θύρα RD (RED LED) για την ρύθμιση του κόκκινου φωτός και μια IRD (Infrared RED) για την ρύθμιση του υπέρυθρου φωτός [30,31]. Στο πειραματικό μέρος δεν χρησιμοποιούνται οι θύρες RD και IRD για τις αντίστοιχες ρυθμίσεις, αφού το πρωτόκολλο I2C (SCL,SDA) δίνει τις απαραίτητες ρυθμίσεις για την υλοποίηση του.

5 Λειτουργία αισθητήρων μέτρησης καρδιακών παλμών, οξυγόνου και καρδιογράφημα



Σχήμα 5.1: Διάγραμμα λειτουργίας αισθητήρα μέτρησης κορεσμού οξυγόνου και καρδιακών παλμών [32].

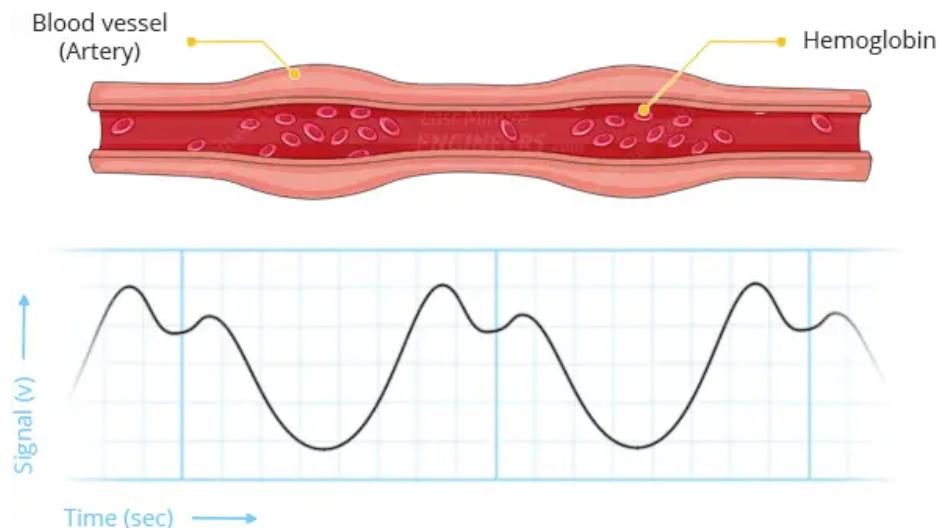
5.1 Διαφορά καρδιακού παλμού και σφυγμού

Ο καρδιακός παλμός (Heart Rate) είναι πόσες φορές η καρδιά συστέλλεται (χτυπά) ανά λεπτό, ενώ ο σφυγμός είναι πόσες φορές οι αρτηρίες διαστέλλονται ανά λεπτό εξαιτίας της συστολής της καρδιάς. Η διαφορά τους είναι στην πραγματικότητα πολύ μικρή, σε σπάνιες περιπτώσεις όπως οι καρδιακές αρρυθμίες μπορεί να παρατηρηθεί απόκλιση. Υπάρχουν δύο τρόποι ανίχνευσης καρδιακού παλμού και σφυγμού: η ηλεκτροκαρδιογραφία (ECG) και η φωτοπληθυσμογραφία (PPG) [33].

5.2 Φωτοπληθυσμογραφία και Καρδιακός Παλμός

Η φωτοπληθυσμογραφία (PPG) είναι μια διαδικασία κατά την οποία ανιχνεύεται ο κορεσμός του οξυγόνου και οι καρδιακοί παλμοί. Συγκεκριμένα χρησιμοποιείται ένα υπέρυθρο φως για

την ανίχνευση οξυγονωμένης αιμοσφαιρίνης (HbO_2) στο αίμα. Κατά την καρδιακή συστολή, η συγκέντρωση της αιμοσφαιρίνης σε μια περιοχή της αρτηρίας αυξάνεται και κατά συνέπεια η οξυγονωμένη αιμοσφαιρίνη είναι περισσότερη μετά από κάθε συστολή της καρδιάς. Η αιμοσφαιρίνη είναι υπεύθυνη για την μεταφορά του οξυγόνου που περιέχετε στο αίμα, από τους πνεύμονες στο υπόλοιπο σώμα. Όσο περισσότερη είναι η οξυγονωμένη αιμοσφαιρίνη στο αίμα τόσο περισσότερο απορροφάτε από το υπέρυθρο φως, άρα η φωτοδίοδος ανιχνεύει λιγότερο φως. Η επαναλαμβανόμενη αυτή διαδικασία προσδιορίζει τον καρδιακό παλμό (HR) [28, 33, 34, 35, 36].

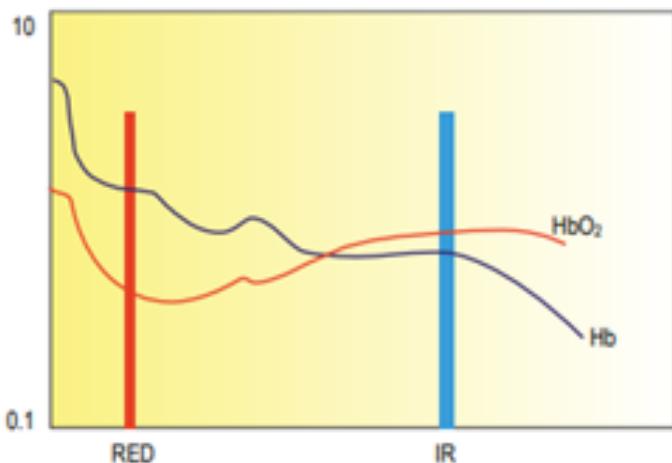


Σχήμα 5.2: Απεικόνιση αρτηρίας και καρδιακού παλμού [28].

Ο καρδιακός παλμός μπορεί να χρησιμοποιηθεί για τη διάγνωση καρδιακών ανωμαλιών όπως ταχυκαρδία ή βραδυκαρδία. Οι καρδιακοί παλμοί που κυμαίνονται σε φυσιολογικά πλαίσια για έναν ενήλικα είναι μεταξύ 60 έως 100 παλμούς το λεπτό. Κάτω από 60 παλμούς υπάρχει βραδυκαρδία και πάνω από 100 παλμούς υπάρχει ταχυκαρδία [33,37].

5.3 Ανίχνευση και Μέτρηση του Κορεσμού Οξυγόνου (SpO_2) στο Αίμα

Η ανίχνευση του κορεσμού οξυγόνου (SpO_2) στο αίμα πραγματοποιείται μέσω αλγορίθμων. Οι αλγόριθμοι χρησιμοποιούν τη διαφορά απορρόφησης μεταξύ του κόκκινου στα 660nm και υπέρυθρου φωτός στα 940nm για τον υπολογισμό της αναλογίας οξυγονωμένης και αποοξυγονωμένης αιμοσφαιρίνης. Ο κορεσμός του οξυγόνου σε φυσιολογικά πλαίσια για κάθε άνθρωπο κυμαίνεται από 94% έως 100%, η πιο συνηθισμένη μέτρηση είναι 97%. Η μέτρηση του κορεσμού του οξυγόνου στο αίμα βοηθά την αποφύγει της υποξαιμία. Δηλαδή η υποξαιμία είναι όταν το ποσοστό οξυγόνου είναι χαμηλότερο από τα φυσιολογικά πλαίσια. Η υποξαιμία μπορεί να προκαλέσει συμπτώματα όπως πονοκέφαλο, δύσπνοια, ταχυκαρδία, σύγχυση και μπλέ χρώμα στο δέρμα, στα νύχια και στα χείλια. Όπως απεικονίζεται και στο παρακάτω διάγραμμα, παρατηρείτε ότι η αποοξυγονωμένη αιμοσφαιρίνη (Hb) απορροφά περισσότερο κόκκινο φως, ενώ η οξυγονωμένη αιμοσφαιρίνη απορροφά περισσότερο υπέρυθρο φως [28,38,39].

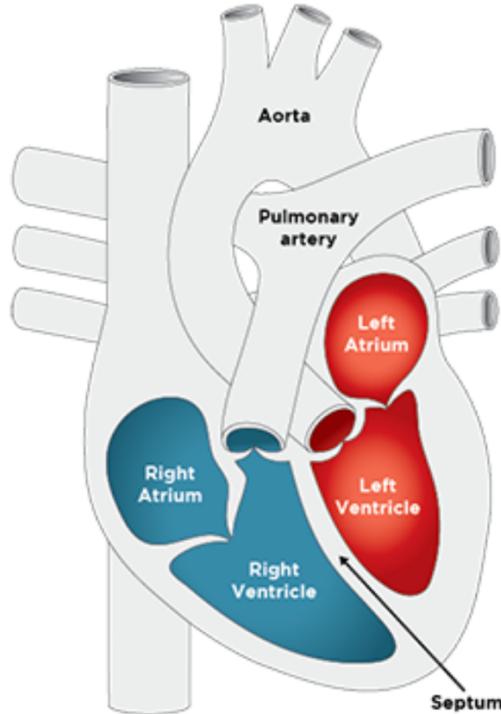


Σχήμα 5.3: Διάγραμμα απορρόφησης υπέρυθρου και κόκκινου φως [32].

Η φωτοπληθυσμογραφία χρησιμοποιείται σε φορητές συσκευές όπως έξυπνά ρολόγια για καθημερινή παρακολούθηση της υγείας.

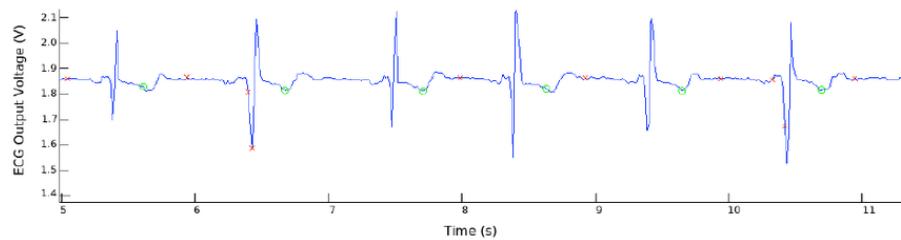
5.4 Ηλεκτροκαρδιογραφία και Καρδιακή Λειτουργία

Η ηλεκτροκαρδιογραφία (ECG) είναι μια απλή και γρήγορη διαδικασία. Ο σκοπός αυτής της διαδικασίας είναι να καταγράψει τις συστολές και την επαναφορά των μυών της καρδιάς. Αναλυτικότερα, καρδιά είναι ένα όργανο που αντλεί αίμα σε όλο το σώμα. Διαθέτει δύο κόλπους, δύο κοιλίες και τέσσερις βαλβίδες. Οι βαλβίδες εξασφαλίζουν τη σωστή κατεύθυνση του αίματος μεταξύ των θαλάμων (κόλπων και κοιλιών) της καρδιάς. Επίσης δυο φλέβες (άνω και κάτω κοίλη) μεταφέρουν αίμα με χαμηλή περιεκτικότητα σε οξυγόνο στον δεξιό κόλπο. Η άνω κοίλη μεταφέρει αίμα από το άνω μέρος του σώματος, ενώ η κάτω κοίλη φλέβα φέρνει αίμα από το κάτω μέρος του σώματος. Έπειτα το αίμα μεταφέρεται στην δεξιά κοιλία. Η δεξιά κοιλία αντλεί αίμα με χαμηλή περιεκτικότητα σε οξυγόνο στους πνεύμονες μέσω της πνευμονικής αρτηρίας. Στη συνέχεια, οι πνεύμονες οξυγονώνουν το αίμα και οι πνευμονικές φλέβες μεταφέρουν το αίμα στον αριστερό κόλπο. Αφού το αίμα έχει μεταφερθεί από τον αριστερό κόλπο στην αριστερή κοιλία, αντλείται αίμα με μεγάλη περιεκτικότητα σε οξυγόνο σε όλο το υπόλοιπο σώμα [40-43].



Σχήμα 5.4: Απεικόνιση καρδιάς [40].

Με το ηλεκτροκαρδιογράφημα παρακολουθείται η λειτουργία των κινήσεων της καρδιάς, δηλαδή των βαλβίδων, κόλπων και κοιλιών [40-43]. Το παρακάτω γράφημα περιγράφει έξη κύκλους της καρδιάς με τρία ηλεκτρόδια.

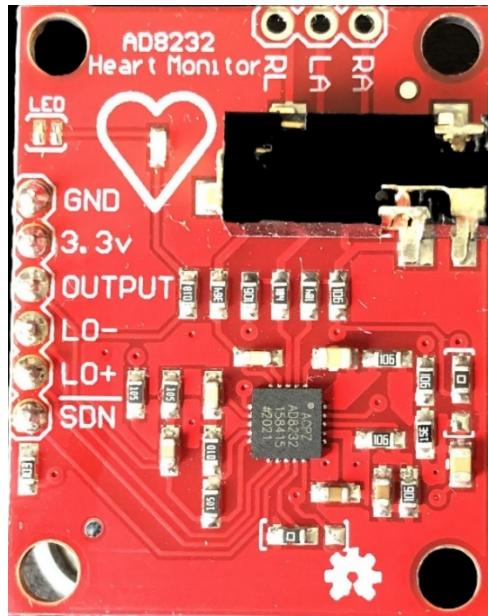


Σχήμα 5.5: Καρδιογράφημα με τρείς επαγωγούς [46].

Η ηλεκτροκαρδιογραφία βοηθά στην αναγνώριση καρδιακών ανωμαλιών, όπως αρρυθμίες στους κόλπους και κοιλίες, μειωμένη ροή αίματος στη στεφανιαία αρτηρία και

ηλεκτρολυτικές διαταραχές [42]. Για το ηλεκτροκαρδιογράφημα χρειάζονται ηλεκτρόδια που προσκολλούνται σε ορισμένα σημεία του σώματος όπως τα πόδια, χέρια και το στήθος. Τα ηλεκτρόδια δε διοχετεύουν ρεύμα στο σώμα, αλλά ανιχνεύουν τα ηλεκτρικά κύματα που δημιουργεί η καρδιά μέσω της διαδικασίας εκπόλωσης. Στη διαδικασία αυτή η κυτταρική μεμβράνη στις νευρικές ίνες διαπερνάται από τα ιόντα του νατρίου (NA^+), επιτρέποντας τη διάχυση θετικών ιόντων του νατρίου (NA^+) στο εσωτερικό ύξονα. Η κανονική πολωμένη κατάσταση είναι στα -70mV εξουδετερώνοντας από την εισροή των θετικών φορτισμένων ιόντων νατρίου, με το δυναμικό να αυξάνεται γρήγορα προς τη θετική κατεύθυνση. Δηλαδή στην εκπόλωση υπάρχει διαφορά ηλεκτρικού δυναμικού, τα καρδιακά κύτταρα μεταβαίνουν από μια κατάσταση ηρεμίας σε μια ενεργή κατάσταση. Το ηλεκτροκαρδιογράφημα ανιχνεύει P κύματα που αντιπροσωπεύουν την εκπόλωση των κόλπων, το σύμπλεγμα QRS που αντιπροσωπεύει την εκπόλωση των κοιλιών και τα κύματα T που αντιπροσωπεύουν την επαναπόλωση των κοιλιών. Η επαναπόλωση είναι αντίστροφη διαδικασία της εκπόλωσης, δηλαδή υπάρχει αλλαγή κατάστασης από θετικό σε πιο αρνητικό δυναμικό της μεμβράνης του κυττάρου της καρδιάς [42,45,46,47,48]. Επιπλέον, η ηλεκτροκαρδιογραφία είναι μια χρήσιμη διαγνωνιστική μέθοδος της καρδιάς, αλλά για την καλύτερη απεικόνιση της καρδιάς συνιστάται ο υπέρηχος.

6 AD8232 Heart Monitor



Εικόνα 6.1: Αισθητήρας AD8232.

6.1 Περιγραφή αισθητήρα AD8232

Το AD8232 είναι ένα ενσωματωμένο μπλοκ ρύθμισης σήματος για εφαρμογές μέτρησης ηλεκτροκαρδιογραφήματος (ECG) και άλλων βιοδυναμικών σημάτων. Έχει σχεδιαστεί για να εξάγει, να ενισχύει και να φίλτραρε μικρά βιοδυναμικά σήματα με συνθήκες παρουσίας θορύβου, όπως δημιουργούνται κατά τη διάρκεια της κίνησης του σώματος ή την απομακρυσμένη τοποθέτηση των ηλεκτροδίων από το σημείο μέτρησης. Αυτός ο σχεδιασμός επιτρέπει τη χαμηλή κατανάλωση ρεύματος του μετατροπέα αναλογικού σε ψηφιακού σήματος (ADC) ή σε έναν ενσωματωμένο μικροελεκτή να λαμβάνει τα σήματα εξόδου με ευκολία [49]. Τα βιοδυναμικά σήματα είναι ηλεκτρικά σήματα που παράγονται από την καρδία ή από άλλα μέρει του σώματος [50]. Επιπλέον, το AD8232 μπορεί να υλοποιήσει ένα διπολικό υψηπερατό φίλτρο (two – pole high-pass) για την εξάλειψη των κινήσεων που μπορούν να προκληθούν από την κίνηση του

ανθρώπου και το δυναμικό μισού κυττάρου των ηλεκτροδίων μέχρι και ± 300 mV. Το φίλτρο αυτό συσχετίζεται με την αρχιτεκτονική του ενισχυτή μέτρησης επιτρέποντας υψηλό κέρδος (gain) και υψηλερατό φιλτράρισμα σε ένα μόνο στάδιο, εξοικονομώντας έτσι χώρο και κόστος. Για την προσαρμογή των τριών ηλεκτροδίων στην πλακέτα AD8232 χρησιμοποιείται μια θύρα jack, αλλά έχει και συμπληρωματικά την επιλογή τριών ακίδων RA (δεξιό χέρι), LA (αριστερό χέρι), RL (δεξιό πόδι) [49].

6.2 Ηλεκτρόδια, ηλεκτρολύτες και φιλτράρισμα σημάτων

Στην επιφάνεια του ηλεκτροδίου προς τον ηλεκτρολύτη, υπάρχει μια σταθερή ροή ηλεκτρονίων μέσα και έξω από το ηλεκτρόδιο. Όταν το ρεύμα ρέει προς την μια κατεύθυνση, είναι ίσο και ακυρώνει το ρεύμα που ρέει προς την κατεύθυνση αυτή. Αυτό είναι γνωστό ως ισορροπία. Το δυναμικό σε αυτή την ισορροπία μπορεί να είναι επίσης γνωστό ως δυναμικό ισορροπίας, αντιστρέψιμο ή μισού κυττάρου (Half-cell). Αυτό το δυναμικό εξαρτάται από τη σύνθεση και τη ιοντική συγκέντρωση του ηλεκτρολύτη και από τον τρόπο ολληλεπίδρασης με το μέταλλο του ηλεκτροδίου [51]. Ο ηλεκτρολύτης τοποθετείται επάνω στο ηλεκτρόδιο για καλύτερη αγωγιμότητα.

Στο υψηλερατό φίλτρο (High Pass) η ζώνη απαγόρευσης συχνοτήτων είναι από $\omega = 0$ Hz έως $\omega = \omega_c$, ενώ η ζώνη διέλευσης συχνοτήτων είναι από $\omega = \omega_c$ έως άπειρο. Το ω_c ονομάζεται συχνότητα αποκοπής [52]. Το διπολικό φίλτρο προσφέρει μεγαλύτερη ακρίβεια στο φιλτράρισμα. Ακόμα ένας μη δεσμευμένος ενισχυτής δίνει την δυνατότητα στο AD8232 να δημιουργήσει ένα φίλτρο χαμηλής διέλευσης τριών πόλων για ακόμα μεγαλύτερη μείωση του θορύβου. Επιπλέον, ο χρήστης έχει την δυνατότητα να επιλέξει τη συχνότητα αποκοπής σε όλα τα φίλτρα για να μπορεί να υλοποιεί διάφορες εφαρμογές [49].

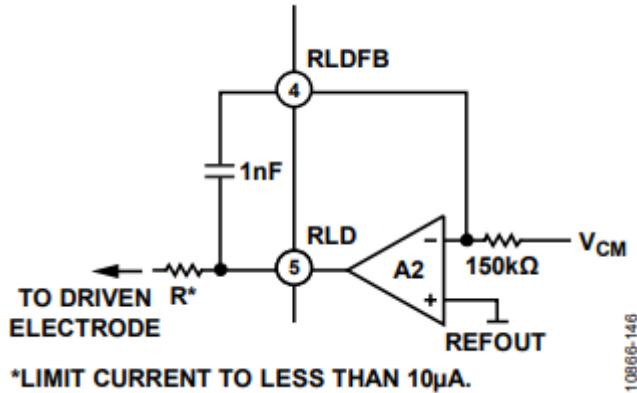
6.3 Ενισχυτής δεξιού ποδιού

Το AD8232 παρέχει έναν ενισχυτή οδήγησης δεξιού ποδιού (RLD), ο οποίος αντιστρέφει το σήμα κοινής λειτουργίας (common-mode) που είναι παρόν στις εισόδους του ενισχυτή οργάνων. Όταν από το δεξιό πόδι οδηγείται το ρεύμα που εγχέεται στο σώμα του αντικειμένου, αντισταθμίζονται οι μεταβολές του δυναμικού κοινής λειτουργίας (common-mode), έτσι βελτιώνεται η απόρριψη του σήματος της κοινής λειτουργίας (common-mode) του συστήματος [49]. Το σήμα κοινής λειτουργίας είναι η ίδια συνιστώσα της τάσης που υπάρχει και στους δύο ακροδέκτες εισόδου [50]. Ειδικότερα, το σήμα κοινής λειτουργίας που υπάρχει στις εισόδους του ενισχυτή οργάνων προέρχεται από τον ενισχυτή διαγωγιμότητας, GM1. Έπειτα συνδέεται με την αντίστροφη είσοδο του A2 μέσω μιας αντίστασης $150\text{ k}\Omega$. Ένας ολοκλήρωσής μπορεί να δημιουργηθεί συνδέοντας έναν πυκνωτή μεταξύ του RLD FB και RLD ακροδεκτών. Οπού RLD FB είναι ο ακροδέκτης ανάδρασης του κυκλώματος οδήγησης του δεξιού ποδιού (RLD). Μπορεί να υλοποιηθεί με έναν πυκνωτή 1 nF , που η συχνότητα διασταύρωσης είναι περίπου 1 kHz , όπου σε αυτήν την συχνότητα ο ενισχυτής έχει αντιστρεπτική μονάδα κέρδους (gain -1 db).

6.4 Συχνότητα διασταύρωσης και σταθερότητα κυκλώματος

Η συχνότητα διασταύρωσης είναι ένα κρίσιμο σημείο για την σταθερότητα και την απόδοση του κυκλώματος. Αυτή η διαμόρφωση έχει ως αποτέλεσμα περίπου 26dB κέρδος βρόχου που διατίθεται σε εύρος συχνοτήτων από 50Hz έως 60 Hz για απόρριψη γραμμής κοινής λειτουργίας. Μεγαλύτερες τιμές πυκνωτών μειώνουν την συχνότητα διασταύρωσης, με αποτέλεσμα μείωση του διαθέσιμου κέρδους που διατίθεται με την απόρριψη και κατά συνέπεια αυξάνεται ο θόρυβος.

Μικρότερες τιμές πυκνωτών μεταβάλλουν την συχνότητα διασταύρωσης σε υψηλότερες συχνότητες, επιτρέποντας να αυξηθεί το κέρδος. Επίσης, όταν η τιμή του κέρδους είναι αρκετά μεγάλη τότε το σύστημα μπορεί να γίνει ασταθές και μπορεί να φτάσει σε κορεσμό η έξοδος του ενισχυτή δεξιού ποδιού. Επιπροσθέτως, όταν χρησιμοποιείται ο ενισχυτής (RLD) για την οδήγηση του ηλεκτροδίου, πρέπει να υπάρχει μια αντίσταση σε σειρά με την έξοδο ώστε να περιορίζει το ρεύμα κάτω από 10 μ A, ακόμα και σε συνθήκες βλάβης [49].



Σχήμα 6.1: Μια συνηθισμένη διαμόρφωση κυκλώματος δεξιού ποδιού [49].

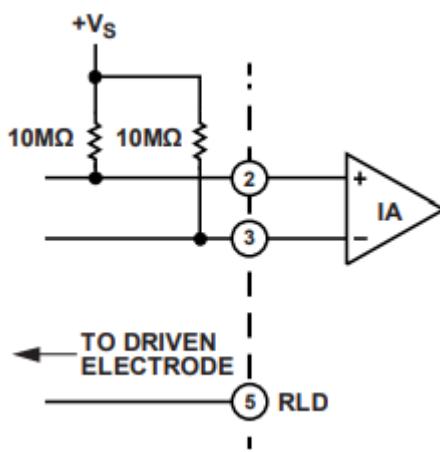
6.5 Λειτουργία γρήγορης αποκατάστασης

Εξαιτίας της χαμηλής συχνότητας αποκοπής που χρησιμοποιείται στα υψηπερατά φίλτρα σε εφαρμογές ηλεκτροκαρδιογραφήματος, τα σήματα μπορεί να χρειαστούν αρκετά δευτερόλεπτα για να σταθεροποιηθούν. Αυτός ο χρόνος αποκατάστασης μπορεί να προκαλέσει καθυστέρηση στο σήμα. Το AD8232 περιλαμβάνει μια λειτουργία γρήγορης αποκατάστασης που μειώνει την διάρκεια των μεγάλων χρόνων αποκατάστασης σε υψηπερατά φίλτρα (high-pass). Μετά από μια απότομη αλλαγή που προκαλεί τον κορεσμό του ενισχυτή όπως η αποσύνδεση των ηλεκτροδίων, το AD8232 προσαρμόζει αυτόματα την συχνότητα αποκοπής του

φίλτρου σε μια μεγαλύτερη τιμή. Αυτή η δυνατότητα επιτρέπει στο AD8232 να επαναφέρει τη λειτουργία του και να πραγματοποιεί έγκυρες μετρήσεις λίγο μετά τη σύνδεση των ηλεκτροδίων στο αντικείμενο [49].

6.6 Λειτουργία και ανίχνευση αποσύνδεσης ηλεκτροδίων στο AD8232

Το κύκλωμα του AD8232 όπως φαίνεται και στην εικόνα 6.1 τροφοδοτείται με μέγιστη τάση 3.6V, το εύρος θερμοκρασίας λειτουργίας κυμαίνεται από -40°C έως $+85^{\circ}\text{C}$. Υπάρχει δυνατότητα λειτουργιών για ανίχνευση αποσύνδεσης ηλεκτροδίων AC (Analog Current) και DC (Digital Current), οι οποίες είναι βελτιστοποιημένες για διαμορφώσεις με δύο ή τρία ηλεκτρόδια. Στην περίπτωση διαμόρφωσης τριών ηλεκτροδίων χρησιμοποιείται μόνο η DC ανίχνευσης αποσύνδεσης. Ειδικότερα, λειτουργεί ανιχνεύοντας την τάση οποιασδήποτε εισόδου του ανιχνευτή οργάνων που βρίσκεται εντός 0.5 V από την θετική γραμμή τροφοδοσίας. Σε αυτήν την περίπτωση, κάθε είσοδος πρέπει να έχει μια pull-up αντίσταση συνδεδεμένη με την θετική τροφοδοσία. Κατά την κανονική λειτουργία, το δυναμικό του αντικειμένου πρέπει να είναι στο εύρος της κοινής λειτουργίας του ενισχυτή οργάνων. Αυτό είναι εφικτό μόνο αν το τρίτο ηλεκτρόδιο είναι συνδεδεμένο στην έξοδο του ενισχυτή οδήγησης δεξιού ποδιού [49].



Σχήμα 6.2: Κύκλωμα διαμόρφωσης για DC ανίχνευσης αποσύνδεσης [49]

Εξαιτίας της DC διαμόρφωσης ανίχνευσης αποσύνδεσης ο AD8232 ελέγχει τις εισόδους ξεχωριστά. Με αποτέλεσμα να ανιχνεύει το ηλεκτρόδιο που είναι αποσυνδεδεμένο, ενεργοποιώντας την αντίστοιχη έξοδο LOD- ή LOD+ στη θετική τροφοδοσία. Για να χρησιμοποιηθεί η λειτουργία αυτή πρέπει η ακίδα AC/DC να είναι συνδεδεμένη στην γείωση. Στην περίπτωση της διαμόρφωσης ανίχνευσης αποσύνδεσης AC χρησιμοποιούνται μόνο δύο ηλεκτρόδια [49]. Το LOD (Leads off Detection) σημαίνει ανιχνευτής αποσύνδεσης ηλεκτροδίων.

6.7 Ενισχυτής οργάνων

Ο ενισχυτής οργάνων (instrumentation amplifier) είναι ένα σύμπλεγμα ενισχυτών, φίλτρων, αντιστάσεων και πυκνωτών. Συγκεκριμένα αποτελείται από δύο μεταγωγείς ενίσχυσης (transconductance amplifier), έναν ενισχυτή αποκοπής συνεχούς ρεύματος DC (HPA), έναν ολοκληρωτή διαμόρφωσης και έναν ενισχυτή λειτουργίας (operational amplifier) [49]. Ο ενισχυτής οργάνων έχει την ικανότητα να ενισχύει τα μικρά σήματα με ακρίβεια, υπό την παρουσία θορύβου.

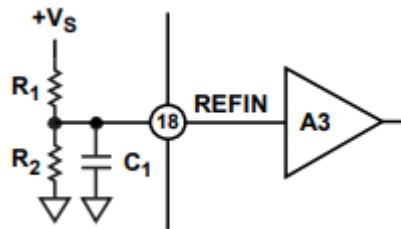
6.8 Έξοδος σήματος στο AD8232

Η ακίδα OUTPUT στην εικόνα 6.1 ή αλλιώς έξοδος λειτουργικού ενισχυτή, αναφέρεται στην έξοδο του πλήρως επεξεργασμένου σήματος καρδιακού ρυθμού. Η έξοδος αυτή μπορεί να συνδεθεί στην είσοδο ενός ADC (αναλογικού σε ψηφιακό μετατροπέα). Ακόμα, η ακίδα ανάστροφος SDN (Shutdown Control Input) με την ενεργοποίησή της το κύκλωμα μπαίνει σε λειτουργία χαμηλότερης κατανάλωσης. Ειδικότερα, με την οδήγηση του SDN στη χαμηλή στάθμη το AD8232 μπαίνει σε κατάσταση απενεργοποίησης και καταναλώνει μικρότερο ρεύμα από 200 nA, προσφέροντας σημαντική εξοικονόμηση ενέργειας. Κατά την

διάρκεια την κατάστασης απενεργοποίησης, το AD8232 δεν μπορεί να διατηρήσει την τάση REFOUT, αλλά δεν καταναλώνει την τάση REFIN. Διατηρώντας έτσι αυτήν την πρόσθετη γραμμή αγωγιμότητας από την τροφοδοσία στη γείωση [49].

6.9 Reference Buffer

Το AD8232 λειτουργεί με μόνο μία τροφοδοσία, για να απλοποιηθεί ο σχεδιασμός της μονής αυτής τροφοδοσίας το AD8232 περιλαμβάνει έναν ενισχυτή αναφοράς (reference buffer) για την δημιουργία μιας εικονικής γείωσης μεταξύ της τάσης τροφοδοσίας και της γείωσης. Τα σήματα που βρίσκονται στην έξοδο από τον ενισχυτή οργάνων αναφέρονται γύρο από αυτήν την τάση. Για παράδειγμα, αν υπάρχει μηδενική διαφορική τάση εισόδου τότε η τάση στην έξοδο από τον ενισχυτή οργάνων θα είναι η τάση αναφοράς [49].

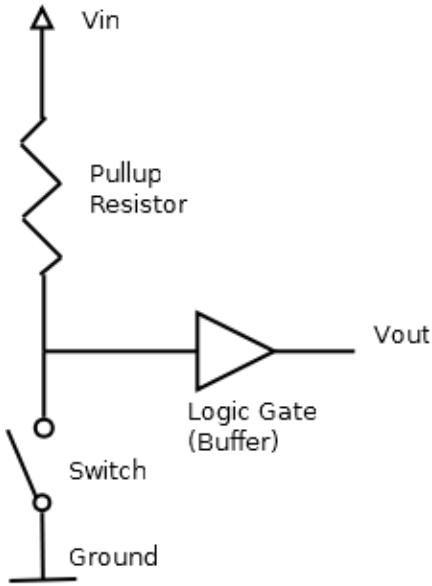


Σχήμα 6.3: Reference Buffer [49].

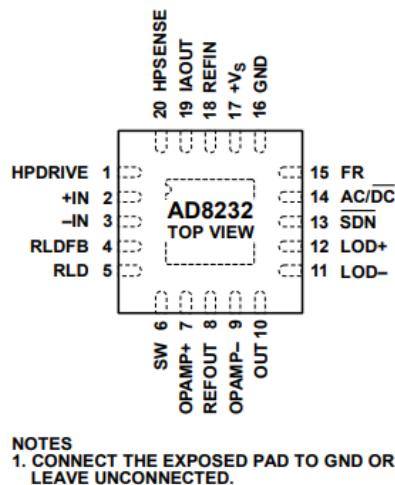
6.10 Αντιστάσεις Pull-up και Pull-down

Η Pull-up ή Pull-down αντίσταση χρησιμοποιείται συνήθως για να διατηρήσει μια κατάσταση ενός σήματος. Επίσης, χρησιμοποιούνται σε συνδυασμό με κάποιο διακόπτη ή τρανζίστορ, τα οποία διακόπτουν τη σύνδεση των επόμενων εξαρτημάτων με τη γείωση ή με την πηγή V_{CC} . Χωρίς αυτές τις αντιστάσεις θα υπήρχε απευθείας σύνδεση μεταξύ πηγής V_{CC} και γείωσης, με αποτέλεσμα να υπάρξει βραχυκύκλωμα. Όταν ο διακόπτης είναι ανοιχτός χωρίς να υπάρχει μια αντίσταση Pull-

με τότε το κύκλωμα δεν έχει κάποια προσδιορισμένη τάση, που δεν είναι επιθυμητό. Με την χρήση της Pull-up αντίστασης διασφαλίζεται μια καθορισμένη τάση όταν ο διακόπτης είναι ανοικτός. Όταν ο διακόπτης είναι κλειστός τότε η έξοδος είναι συνδεδεμένη με τη γείωση [53].



Σχήμα 6.4: Κύκλωμα απεικόνισης Pull-up αντίστασης [53].



Σχήμα 6.5: Απεικόνιση τσιπ AD8232 [49].

Pin No.	Mnemonic	Description
1	HPDRIVE	High-Pass Driver Output. Connect HPDRIVE to the capacitor in the first high-pass filter. The AD8232 drives this pin to keep HPSENSE at the same level as the reference voltage.
2	+IN	Instrumentation Amplifier Positive Input. +IN is typically connected to the left arm (LA) electrode.
3	-IN	Instrumentation Amplifier Negative Input. -IN is typically connected to the right arm (RA) electrode.
4	RLDFB	Right Leg Drive Feedback Input. RLDFB is the feedback terminal for the right leg drive circuit.
5	RLD	Right Leg Drive Output. Connect the driven electrode (typically, right leg) to the RLD pin.
6	SW	Fast Restore Switch Terminal. Connect this terminal to the output of the second high-pass filter.
7	OPAMP+	Operational Amplifier Noninverting Input.
8	REFOUT	Reference Buffer Output. The instrumentation amplifier output is referenced to this potential. Use REFOUT as a virtual ground for any point in the circuit that needs a signal reference.
9	OPAMP-	Operational Amplifier Inverting Input.
10	OUT	Operational Amplifier Output. The fully conditioned heart rate signal is present at this output. OUT can be connected to the input of an ADC.
11	LOD-	Leads Off Comparator Output. In dc leads off detection mode, LOD- is high when the electrode to -IN is disconnected, and it is low when connected. In ac leads off detection mode, LOD- is always low.
12	LOD+	Leads Off Comparator Output. In dc leads off detection mode, LOD+ is high when the +IN electrode is disconnected, and it is low when connected. In ac leads off detection mode, LOD+ is high when either the -IN or +IN electrode is disconnected, and it is low when both electrodes are connected.
13	SDN	Shutdown Control Input. Drive SDN low to enter the low power shutdown mode.
14	AC/DC	Leads Off Mode Control Input. Drive the AC/DC pin low for dc leads off mode. Drive the AC/DC pin high for ac leads off mode.
15	FR	Fast Restore Control Input. Drive FR high to enable fast recovery mode; otherwise, drive it low.
16	GND	Power Supply Ground.
17	+Vs	Power Supply Terminal.
18	REFIN	Reference Buffer Input. Use REFIN, a high impedance input terminal, to set the level of the reference buffer.
19	IAOUT	Instrumentation Amplifier Output Terminal.
20	HPSENSE	High-Pass Sense Input for Instrumentation Amplifier. Connect HPSENSE to the junction of R and C that sets the corner frequency of the dc blocking circuit.
	EP	Exposed Pad. Connect the exposed pad to GND or leave it unconnected.

Πίνακας 6.1: Επεξήγηση των ακίδων του τσιπ AD8232 [49].

7 Message Queuing Telemetry Transport (MQTT) πρωτόκολλο

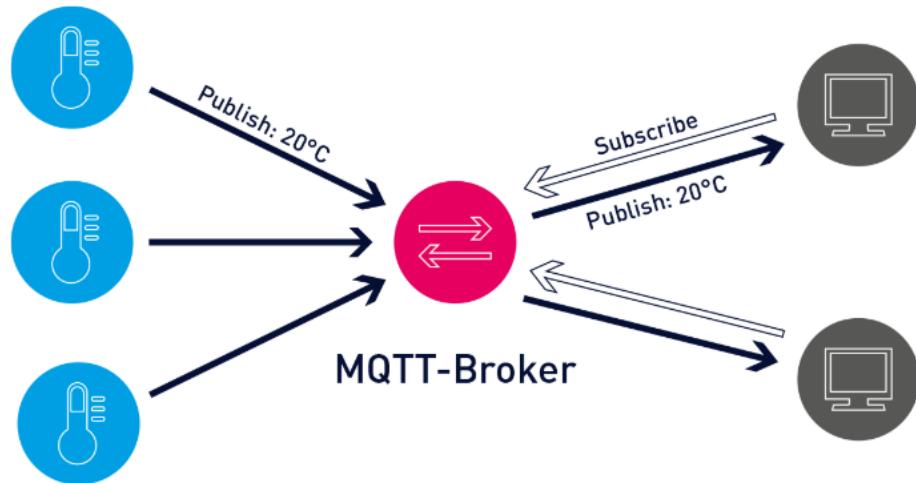
7.1 Περιγραφή πρωτοκόλλου MQTT

Το MQTT είναι ένα ελαφρύ πρωτόκολλο που λειτουργεί με βάση το μοντέλο δημοσίευσης και εγγραφής. Έχει φτιαχτεί ειδικά για συσκευές με περιορισμένες δυνατότητες και για δίκτυα με χαμηλή ταχύτητα και υψηλή καθυστέρηση. Χρησιμοποιείται συχνά σε εφαρμογές Internet of Things (IoT), διευκολύνοντας την αποδοτική επικοινωνία ανάμεσα σε αισθητήρες, ενεργοποιητές και άλλες συσκευές [54].

7.2 Αρχιτεκτονική του MQTT

Το MQTT λειτουργεί πάνω από το TCP/IP χρησιμοποιώντας την τοπολογία PUSH/SUBSCRIBE. Στην αρχιτεκτονική του MQTT υπάρχουν δύο τύποι συστήματος: πελάτες (clients) και διαμεσολαβητές (brokers). Ο διαμεσολαβητής είναι ο διακομιστής (server) με τον οποίο επικοινωνούν οι πελάτες. Ο διαμεσολαβητής λαμβάνει μηνύματα από τους πελάτες και

στέλνει μηνύματα σε άλλους πελάτες. Οι πελάτες δεν επικοινωνούν απευθείας μεταξύ τους, αλλά μέσω ενός διαμεσολαβητή. Κάθε πελάτης μπορεί να είναι είτε εκδότης, είτε συνδρομητής, είτε και τα δύο. Το MQTT είναι ένα πρωτόκολλο που βασίζεται σε γεγονότα (event-driven protocol). Δηλαδή δεν υπάρχει περιοδική ή συνεχής μετάδοση δεδομένων, κάτι που κρατά την επικοινωνία στο ελάχιστο. Ο πελάτης στέλνει μόνο όταν υπάρχει πληροφορία και ο διαμεσολαβητής διαδίδει την πληροφορία στους συνδρομητές όταν φτάσουν τα νέα δεδομένα [55].



Σχήμα 7.1: MQTT- broker [55]

Ακόμα το TCP/IP ή Transmission Control Protocol/Internet Protocol είναι ένα πρωτόκολλο επικοινωνίας που επιτρέπει σε προγράμματα και υπολογιστικές συσκευές να ανταλλάσσουν μηνύματα μέσω του δικτύου. Έχει σχεδιαστεί για να στέλνει πακέτα μέσω του διαδικτύου και να διασφαλίζει την επιτυχή παράδοση των δεδομένων και μνημάτων μέσω δικτύου [56].

Ένας άλλος τρόπος με τον οποίο το MQTT ελαχιστοποιεί τις μεταδόσεις του είναι μέσω του αυστηρού καθορισμένου μικρού μεγέθους των μηνυμάτων. Κάθε μήνυμα έχει μια

προκαθορισμένη κεφαλίδα των 2 bytes. Μπορεί να χρησιμοποιηθεί μια προαιρετική επικεφαλίδα, αλλά θα αυξηθεί το μέγεθος. Το μέγεθος του μηνύματος είναι περιορισμένο έως 256 MB. Υπάρχουν τρία επίπεδα ποιότητας εξυπηρέτησης (Quality of Service, QoS) για την ελαχιστοποίηση της μετάδοσης δεδομένων ή την μεγιστοποίηση της αξιοπιστίας.

- QoS 0 – Το οποίο προσφέρει την μικρότερη μετάδοση δεδομένων. Σε αυτό το επίπεδο κάθε μήνυμα παραδίδεται στον συνδρομητή μία φορά χωρίς να υπάρχει επιβεβαίωση. Επειδή σε αυτό το επίπεδο υποθέτει ότι η παραδώσει έχει γίνει, τα μηνύματα δεν αποθηκεύονται για παράδοση στον παραλήπτη που είναι αποσυνδεδεμένος και θα συνδεθεί αργότερα.
- QoS 1 – Ο διαμεσολαβητής προσπαθεί να παραδώσει το μήνυμα και έπειτα περιμένει επιβεβαίωση από τον συνδρομητή. Αν δεν λάβει επιβεβαίωση μέσα σε κάποιο συγκεκριμένο χρονικό πλαίσιο τότε το στέλνει ξανά. Με αυτήν τη μέθοδο ο συνδρομητής μπορεί να λάβει το μήνυμα παραπάνω από μία φορά αν ο διαμεσολαβητής δεν λάβει έγκαιρα την επιβεβαίωση του συνδρομητή.
- QoS 2 – Ο πελάτης και ο διαμεσολαβητής χρησιμοποιούν μια διαδικασία επιβεβαίωσης τεσσάρων βημάτων (handshake) για να εξασφαλισθεί ότι το μήνυμα έχει παραδοθεί και έχει σταλθεί μία φορά.

7.3 Θέματα (Topics)

Τα μηνύματα στο MQTT δημοσιεύονται ως θέματα (Topics). Τα θέματα είναι δομές σε μια ιεραρχική μορφή χρησιμοποιώντας τον χαρακτήρα κάθετος (/) ως διαχωριστικό. Αυτή η δομή παρομοιάζει τη δομή ενός δέντρου σε ένα σύστημα αρχείων υπολογιστή. Για παράδειγμα, μια δομή sensor/heartRate επιτρέπει σε έναν συνδρομητή να καθορίσει ότι πρέπει να λαμβάνει δεδομένα μόνο από πελάτες που δημοσιεύουν για το θέμα heartRate. Επίσης, αν ένας διαμεσολαβητής λάβει δεδομένα

που δημοσιεύονται σε ένα θέμα που δεν υπάρχει, τότε ο MQTT το δημιουργεί. Δηλαδή δεν απαιτείται να γίνει μια ειδική ενέργεια για τη δημιουργία του θέματος, αρκεί ο πρώτος πελάτης να στείλει δεδομένα σε ένα νέο θέμα. Το MQTT το αναγνωρίζει και το προσθέτει στον διαμεσολαβητή και στη συνέχεια άλλοι πελάτες μπορούν να εγγραφούν σε αυτό το θέμα.

7.4 Διατηρούμενο μήνυμα (retained message)

Για να διατηρηθεί το μικρό αποτύπωμα, τα μηνύματα που λαμβάνονται δεν αποθηκεύονται στον διαμεσολαβητή εκτός αν δηλωθούν με τη σημαία διατηρουμένου. Αυτό αποκαλείται διατηρούμενο μήνυμα. Χρήστες που επιθυμούν τα μηνύματα που λαμβάνουν να αποθηκεύονται πρέπει να χρησιμοποιήσουν άλλο τρόπο εκτός του πρωτόκολλου MQTT. Ακόμα για να εξασφαλιστεί ότι ο νέος συνδρομητής λαμβάνει τα μηνύματα από το θέμα, οι διαμεσολαβητές μπορούν να κρατήσουν το τελευταίο μήνυμα που στάλθηκε από το κάθε θέμα. Έτσι ο κάθε καινούργιος πελάτης που θα εγγραφεί στο θέμα ή όταν κάποιος πελάτης επανασυνδεθεί, θα λαμβάνει το διατηρούμενο μήνυμα. Διασφαλίζοντας έτσι οι συνδρομητές έχουν την τελευταία πληροφορία.

7.5 Τέσσερις βασικές ενέργειες του πρωτοκόλλου MQTT

Υπάρχουν τέσσερις βασικές ενέργειες στην επικοινωνία του πρωτοκόλλου MQTT.

- Δημοσίευση (Publish) – Στέλνει ένα πακέτο δεδομένων περιέχοντας το μήνυμα που πρέπει να αποσταλεί από έναν πελάτη στον διαμεσολαβητή. Στη συνέχεια, ο διαμεσολαβητής μπορεί να τα δημοσιεύει σε κάποιον άλλον πελάτη. Τα δεδομένα είναι συγκεκριμένα για κάθε εφαρμογή, δηλαδή μπορούν να είναι για παράδειγμα μια τιμή από έναν αισθητήρα ή μια ένδειξη ενεργοποίησης/απενεργοποίησης.
- Εγγραφή (Subscribe) – Ένας πελάτης γίνεται συνδρομητής σε ένα θέμα, δηλαδή μπορεί να λάβει δεδομένα από τον

διαμεσολαβητή. Τα θέματα μπορούν να εγγραφούν συγκεκριμένα ή μέσω κάποιων χαρακτήρων (wildcards) που επιτρέπουν την εγγραφή σε ολόκληρο τον κλάδο ή μέρος του. Για να εγγραφεί ο πελάτης στέλνει ένα πακέτο SUBSCRIBE και λαμβάνει ένα πακέτο SUBACK ως επιβεβαίωση. Αν υπάρχει κάποιο διατηρούμενο μήνυμα για το θέμα τότε το λαμβάνει.

- Ping – Ένας πελάτης μπορεί να στείλει ένα σήμα (ping) στον διαμεσολαβητή. Τα σήματα (pings) χρησιμοποιούνται για να εξασφαλίσουν αν η σύνδεση είναι ακόμα ενεργή και ότι η συνεδρία TCP δεν έχει κλείσει από κάποια άλλη συσκευή δικτύου όπως ένας δρομολογητής (Router).
- Αποσύνδεση (Disconnect) – Ο συνδρομητής ή ο εκδότης μπορούν να θέλουν να αποσυνδεθούν τότε στέλνουν ένα μήνυμα Disconnect στον διαμεσολαβητή. Έτσι, με αυτό το μήνυμα ο διαμεσολαβητής ενημερώνεται και δεν χρειάζεται να στέλνει δεδομένα στον συνδρομητή ή να λαμβάνει δεδομένα από έναν εκδότη. Αυτός ο τρόπος τερματισμού επιτρέπει στον πελάτη να ξανασυνδεθεί χρησιμοποιώντας την ίδια ταυτότητα με προηγουμένως.

7.6 Ασφάλεια πρωτοκόλλου MQTT

Το πρωτόκολλο MQTT δεν διαθέτει κάποια αποτελεσματική ασφάλεια, διότι έχει σχεδιαστεί να επιτύχει τη μικρότερη και πιο αποδοτική μετάδοση δεδομένων μέσω δαπανηρών και αναξιόπιστων γραμμών επικοινωνίας. Ωστόσο, υπάρχουν οι παρακάτω επιλογές.

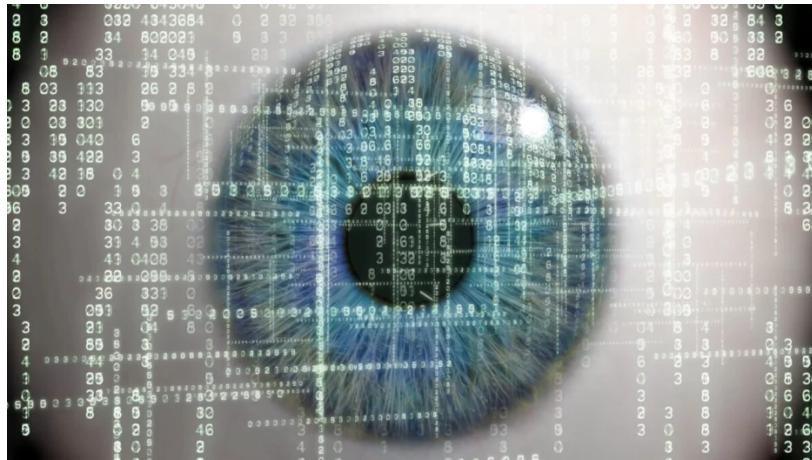
- Ασφάλεια δικτύου – Αν το ίδιο το δίκτυο είναι ασφαλές τότε δεν χρειάζεται να υπάρχει ασφάλεια του πρωτοκόλλου MQTT. Αυτή η ασφάλεια μπορεί να επιτευχθεί, για παράδειγμα, με ένα Virtual Private Network (VPN) που εξασφαλίζει την κρυπτογράφηση όλων των δεδομένων.

- Όνομα και κωδικός Χρήστη – Το MQTT επιτρέπει τη χρήση ονόματος και κωδικού χρήστη πρόσβασης για τους πελάτες για την δημιουργία μιας σύνδεσης με τον διαμεσολαβητή. Ωστόσο αυτά μεταδίδονται σε καθαρό κείμενο και όχι κρυπτογραφημένα. Με αποτέλεσμα η προστασία να γίνεται άχρηστη και ο τρόπος αυτός να λειτουργεί ως αποφυγή για ακούσιες συνδέσεις.
- SSL/TLS – Αφού το MQTT λειτουργεί πάνω από το TCP/IP, μπορεί να υπάρχει ασφάλεια των μεταδόσεων μεταξύ πελατών και διαμεσολαβητή μέσω του SSL/TLS. Αυτό όμως προσθέτει σημαντική επιβάρυνση στην επικοινωνία. Επίσης το SSL/TLS ή Secure Sockets Layer/Transport Layer Security είναι ένα πρωτόκολλο κρυπτογράφησης [55].

8 Τεχνητή Νοημοσύνη

Η Τεχνητή Νοημοσύνη (Artificial Intelligence) αναφέρεται στην ικανότητα των υπολογιστικών συστημάτων να εκτελούν εργασίες που παραδοσιακά απαιτούν ανθρώπινη νοημοσύνη, όπως η μάθηση, η κατανόηση, η επίλυση προβλημάτων, η λήψη αποφάσεων, η δημιουργικότητα και η αυτονομία. Είναι ένας τομέας της επιστήμης των υπολογιστών που αναπτύσσει και μελετά μεθόδους και λογισμικό που επιτρέπουν στις μηχανές να αντιλαμβάνονται το περιβάλλον τους, να μαθαίνουν από νέες πληροφορίες και να λαμβάνουν αποφάσεις που μεγιστοποιούν την επιτυχία των στόχων τους [57, 58].

8.1 Ιστορία Υπολογιστικής Όρασης



Εικόνα 8.1: Υπολογιστική όραση [59].

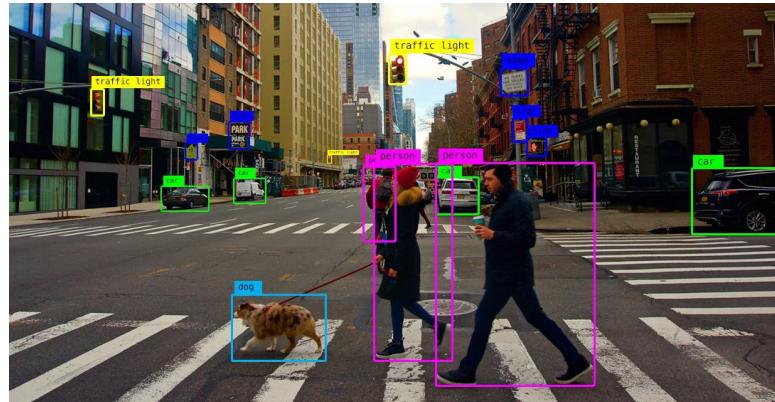
Η υπολογιστική όραση αποτελεί έναν από τους σημαντικότερους κλάδους της τεχνητής νοημοσύνης, με στόχο να επιτρέψει στους υπολογιστές να «βλέπουν» και να κατανοούν οπτικά δεδομένα, όπως κάνει ο ανθρώπινος εγκέφαλος. Η ανάπτυξή της ξεκίνησε τη δεκαετία του 1960, όταν ερευνητές επιδίωξαν να προσομοιώσουν το ανθρώπινο οπτικό σύστημα, ελπίζοντας ότι αυτό θα οδηγήσει σε πιο έξυπνες μηχανές. Η πρώτη προσπάθεια έγινε το 1966 με ένα φοιτητικό έργο, όπου ένας υπολογιστής με κάμερα θα περιέγραφε ό,τι έβλεπε.

Τη δεκαετία του 1970, οι επιστήμονες άρχισαν να αναπτύσσουν θεμελιώδεις αλγορίθμους, όπως η ανίχνευση ακμών, η επισήμανση γραμμών και η αναπαράσταση αντικειμένων μέσω γεωμετρικών σχημάτων. Στη δεκαετία του 1980, η έρευνα έγινε πιο μαθηματικά αυστηρή, εισάγοντας μεθόδους όπως η θεωρία της κλίμακας-χώρου και τα Μαρκοβιανά τυχαία πεδία (Markovian random fields). Μέχρι τη δεκαετία του 1990, βελτιώθηκαν οι τεχνικές ανακατασκευής 3D εικόνων, η βαθμονόμηση καμερών και η αναγνώριση προσώπων, όπως η τεχνική Eigenface[60].

Στις αρχές του 21ου αιώνα, η υπολογιστική όραση γνώρισε ραγδαία ανάπτυξη, κυρίως χάρη στη μηχανική μάθηση και τη βαθιά μάθηση (Deep learning). Το 2012, το νευρωνικό δίκτυο AlexNet[61] σημείωσε εντυπωσιακή επιτυχία στην αναγνώριση εικόνων, μειώνοντας δραστικά το ποσοστό σφαλμάτων. Έκτοτε, η τεχνολογία των συνελικτικών νευρωνικών δικτύων (CNNs) έχει κυριαρχήσει στην αναγνώριση και κατανόηση εικόνων.

Σήμερα, η υπολογιστική όραση χρησιμοποιείται σε ένα ευρύ φάσμα εφαρμογών, όπως η αναγνώριση προσώπων, η ιατρική διάγνωση, η αυτόνομη οδήγηση και η επεξεργασία εικόνων. Η πρόοδος της βαθιάς μάθησης συνεχίζει να βελτιώνει την ακρίβεια και την ικανότητα των συστημάτων να ερμηνεύουν οπτικά δεδομένα, φέρνοντας την τεχνητή νοημοσύνη πιο κοντά στην ανθρώπινη όραση [62, 63].

8.1.1 Σημασία της υπολογιστικής όρασης



Εικόνα 8.2 Παράδειγμα υπολογιστικής όρασης [64].

Η υπολογιστική όραση ή αλλιώς Computer Vision είναι ένα πεδίο της τεχνητής νοημοσύνης που χρησιμοποιεί την μηχανική μάθηση και τα νευρωνικά δίκτυα. Ωστε να μάθει ένας υπολογιστής ή ένα σύστημα να αντλεί πληροφορίες από ψηφιακά δεδομένα όπως φωτογραφίες και βίντεο και να κάνουν συστάσεις ή να αναλαμβάνουν ενέργειες όταν αντιλαμβάνονται κάποιο

ελάττωμα ή πρόβλημα. Με την υπολογιστική όραση επιτρέπει στον υπολογιστή να βλέπει, παρατηρεί και να καταλαβαίνει, παράλληλα η τεχνητή νοημοσύνη δίνει την δυνατότητα στον υπολογιστή να σκέφτεται. Η υπολογιστική όραση λειτουργεί παρόμοια με την ανθρώπινη όραση, αλλά μειονεκτεί σε σύγκριση με αυτήν, καθώς οι άνθρωποι έχουν το πλεονέκτημα της εμπειρίας και της εκπαίδευσης από τη γέννησή τους, στην εκτίμηση μιας απόστασής, ανίχνευσης μια κίνησης και στην διόρθωση ατελειών [63].

8.1.2 Μηχανική μάθηση και νευρωνικά δίκτυα στην υπολογιστική όραση

Η υπολογιστική όραση απαιτεί μεγάλες ποσότητες δεδομένων, ώστε να πραγματοποιούνται συνεχείς αναλύσεις για την εντόπιση των μοτίβων και αναγνώριση των εικόνων. Για παράδειγμα, για την εκπαίδευση ενός υπολογιστή να αναγνωρίζει έναν πυροσβεστήρα, πρέπει να διατίθεται ένα μεγάλο αριθμό εικόνων από πυροσβεστήρες και σχετικών αντικειμένων, ώστε να μάθει να διακρίνει και ιδιαίτερα όταν υπάρχει κάποιο ελάττωμα. Δύο βασικές τεχνολογίες καθιστούν αυτό εφικτό:

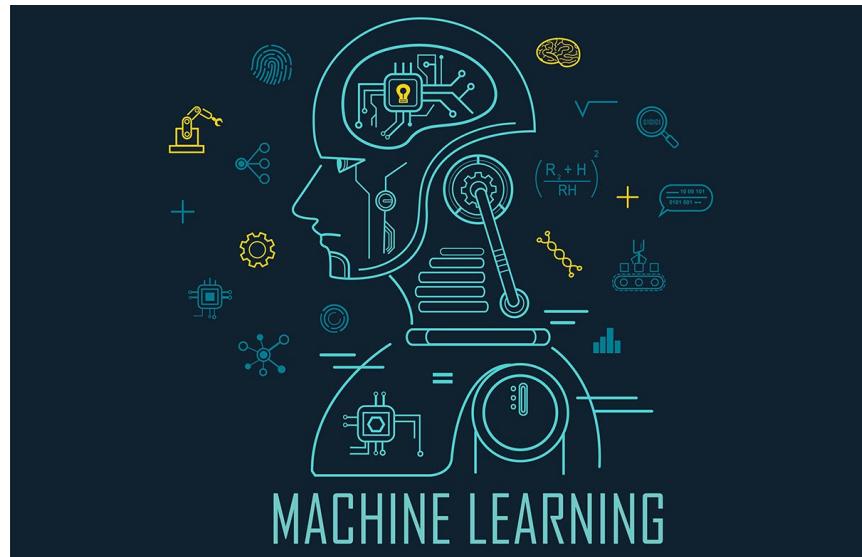
- Η βαθιά μάθηση (Deep learning), ένας τύπος μηχανικής μάθησης,
- Τα συνελικτικά νευρωνικά δίκτυα (CNNs - Convolutional Neural Networks).

Η μηχανική μάθηση χρησιμοποιεί αλγορίθμικά μοντέλα που επιτρέπουν στον υπολογιστή να κατανοεί οπτικά δεδομένα αυτόνομα. Εάν τροφοδοτηθούν αρκετά δεδομένα μέσου του μοντέλου τότε ο υπολογιστής θα ανατρέξει στα δεδομένα και θα μάθει να διαχωρίζει τις εικόνες. Σε αντίθεση, τα συνελικτικά νευρωνικά δίκτυα διευκολύνουν την διαδικασία, αναλύοντας εικόνες σε επίπεδο pixel και αποδίδοντάς τους ετικέτες ή χαρακτηρισμούς. Στη συνέχεια, εφαρμόζουν συνελίξεις (convolutions), μια μαθηματική πράξη που συνδυάζει

συναρτήσεις για να εξάγει νέες πληροφορίες. Το νευρωνικό δίκτυο εκτελεί πολλές επαναλήψεις, βελτιώνοντας σταδιακά την ακρίβεια των προβλέψεών του, μέχρι να μπορεί να αναγνωρίζει εικόνες με τρόπο παρόμοιο με την ανθρώπινη όραση.

Οπως ένας άνθρωπος αρχικά διακρίνει έντονες γραμμές και βασικά σχήματα προτού διαμορφώσει μια πλήρη εικόνα, έτσι και ένα CNN ξεκινά με απλές δομές και εμπλουτίζει τις πληροφορίες του μέσω διαδοχικών επαναλήψεων. Ενώ τα CNNs εξειδικεύονται στην ανάλυση μεμονωμένων εικόνων, τα αναδρομικά νευρωνικά δίκτυα (RNNs - Recurrent Neural Networks) χρησιμοποιούνται για επεξεργασία βίντεο, βιοηθώντας τους υπολογιστές να κατανοήσουν τη σχέση μεταξύ διαδοχικών καρέ [63].

8.2 Μηχανική Μάθηση



Εικόνα 8.3: Μηχανική Μάθηση [65].

Η μηχανική μάθηση είναι ένας κλάδος της τεχνητής νοημοσύνης που επικεντρώνεται στην ανάπτυξη συστημάτων ικανών να μαθαίνουν και να βελτιώνονται αυτόματα μέσω της εμπειρίας. Δίνει τη δυνατότητα σε υπολογιστές και μηχανές να μιμούνται

τον ανθρώπινο τρόπο μάθησης, να εκτελούν εργασίες αυτόνομα και να αυξάνουν την ακρίβεια και την απόδοσή τους καθώς εκτίθενται σε περισσότερα δεδομένα. Το Πανεπιστήμιο της Καλιφόρνια στο Μπέρικλεϊ αναλύει το σύστημα μάθησης ενός αλγορίθμου μηχανικής μάθησης σε τρία βασικά κύρια μέρη:

- **Διαδικασία Λήψης Απόφασης:** Οι αλγόριθμοι μηχανικής μάθησης έχουν ως στόχο να πραγματοποιούν προβλέψεις ή ταξινομήσεις. Με βάση δεδομένα εισόδου τα οποία μπορεί να είναι επισημασμένα ή μη ο αλγόριθμος ανιχνεύει πρότυπα και δημιουργεί μια εκτίμηση με βάση αυτά.
- **Συνάρτηση Σφάλματος:** Η συνάρτηση αυτή αξιολογεί την ακρίβεια των προβλέψεων του μοντέλου. Όταν υπάρχουν διαθέσιμα επισημασμένα δεδομένα, συγκρίνει τα αποτελέσματα του μοντέλου με τις πραγματικές τιμές, προκειμένου να εκτιμήσει την απόδοσή του.
- **Διαδικασία Βελτιστοποίησης του Μοντέλου:** Για να βελτιώσει την ακρίβειά του, το μοντέλο προσαρμόζει τις εσωτερικές του παραμέτρους (όπως τα βάρη), μειώνοντας τη διαφορά μεταξύ των προβλέψεων και των πραγματικών τιμών. Αυτή η διαδικασία είναι επαναληπτική και συνεχίζεται έως ότου το μοντέλο φτάσει σε ένα αποδεκτό επίπεδο ακρίβειας [66].

8.2.1 Μέθοδοι μηχανικής μάθησης

Υπάρχουν διάφοροι μέθοδοι μηχανικής μάθησης που καλύπτουν διαφορετικές ανάγκες. Παρακάτω αναλύονται η κάθε μία μέθοδος ξεχωριστά.

- **Εποπτευόμενη μάθηση** (supervised learning), χαρακτηρίζεται από τη χρήση επισημασμένων συνόλων δεδομένων για την εκπαίδευση αλγορίθμων, ώστε να μπορούν να ταξινομούν πληροφορίες ή να προβλέπουν αποτελέσματα με ακρίβεια. Κατά τη διαδικασία

εκπαίδευσης, το μοντέλο λαμβάνει δεδομένα εισόδου και προσαρμόζει σταδιακά τα βάρη του, έως ότου επιτευχθεί η σωστή προσαρμογή. Αυτή η διαδικασία περιλαμβάνει τη διασταυρωμένη επικύρωση (cross-validation) για να διασφαλιστεί ότι το μοντέλο δεν υπερπροσαρμόζεται (overfitting) ή υποπροσαρμόζεται (underfitting). Επιπλέον η επιβλεπόμενη μάθηση χρησιμοποιείται για την αντιμετώπιση διαφόρων πραγματικών προκλήσεων σε μεγάλη κλίμακα, όπως η αυτόματη ανίχνευση και διαχωρισμός ανεπιθύμητων μηνυμάτων (spam) από τα εισερχόμενα email. Κάποιες από τις πιο διαδεδομένες τεχνικές της επιβλεπόμενης μάθησης περιλαμβάνουν τα νευρωνικά δίκτυα, τον αλγόριθμο Naïve Bayes, τη γραμμική και λογιστική παλινδρόμηση, τα τυχαία δάση (random forest) και τις υποστηρικτικές διανυσματικές μηχανές (Support Vector Machine - SVM).

- **Η μη επιβλεπόμενη μάθηση** (unsupervised learning), χρησιμοποιεί αλγορίθμους για την ανάλυση και ομαδοποίηση μη επισημασμένων συνόλων δεδομένων σε συστάδες (clusters). Αυτοί οι αλγόριθμοι εντοπίζουν κρυφά μοτίβα και σχέσεις στα δεδομένα χωρίς την ανάγκη ανθρώπινης παρέμβασης. Η ικανότητα της μη επιβλεπόμενης μάθησης να αναγνωρίζει ομοιότητες και διαφορές την καθιστά ιδιαίτερα χρήσιμη σε εφαρμογές όπως η διερευνητική ανάλυση δεδομένων, η τμηματοποίηση πελατών, οι στρατηγικές προώθησης προϊόντων (cross-selling), καθώς και η αναγνώριση εικόνων και προτύπων. Επιπλέον, χρησιμοποιείται συχνά για τη μείωση της διαστατικότητας (dimensionality reduction), δηλαδή τη μείωση του αριθμού των χαρακτηριστικών ενός μοντέλου. Δύο ευρέως χρησιμοποιούμενες τεχνικές για αυτόν τον σκοπό είναι η Ανάλυση Κύριων Συνιστώσων (Principal Component Analysis, PCA) και η Μοναδιαία Αποσύνθεση Τιμών

(Singular Value Decomposition, SVD). Άλλοι αλγόριθμοι που χρησιμοποιούνται στη μη επιβλεπόμενη μάθηση περιλαμβάνουν τα νευρωνικά δίκτυα, τη συσταδοποίηση (clustering k-means) και τις πιθανοτικές μεθόδους συσταδοποίησης.

- Η **ημι-επιβλεπόμενη** (semi-supervised) μάθηση προσφέρει μια ενδιάμεση λύση μεταξύ επιβλεπόμενης και μη επιβλεπόμενης μάθησης. Κατά την εκπαίδευση, χρησιμοποιεί ένα μικρό σύνολο επισημασμένων δεδομένων για να καθοδηγήσει τη διαδικασία ταξινόμησης και εξαγωγής χαρακτηριστικών από ένα πολύ μεγαλύτερο, μη επισημασμένο σύνολο δεδομένων. Αυτή η προσέγγιση αντιμετωπίζει το πρόβλημα της έλλειψης επαρκών επισημασμένων δεδομένων για τους αλγορίθμους επιβλεπόμενης μάθησης. Επιπλέον, είναι ιδιαίτερα χρήσιμη όταν η επισήμανση μεγάλου όγκου δεδομένων είναι πολύ δαπανηρή ή χρονοβόρα.
- Η **ενισχυτική μάθηση** (reinforcement learning) είναι μια προσέγγιση στη μηχανική μάθηση που μοιάζει με την επιβλεπόμενη μάθηση, αλλά δεν βασίζεται σε προκαθορισμένα δεδομένα εκπαίδευσης. Αντίθετα, το μοντέλο μαθαίνει δυναμικά μέσω δοκιμής και σφάλματος. Ενισχύοντας μια σειρά επιτυχημένων αποτελεσμάτων, βελτιώνει σταδιακά τη στρατηγική του ώστε να προσδιορίσει την ιδανική σύσταση ή πολιτική για ένα συγκεκριμένο πρόβλημα [66].

8.2.2 Αλγόριθμοί μηχανικής μάθησης

Για την υλοποίηση του μοντέλου της μηχανικής μάθησης χρησιμοποιούνται κάποιοι αλγόριθμοί.

- Νευρωνικά δίκτυα (neural networks)
- Γραμμική οπισθοδόμηση (linear regression), που χρησιμοποιείται για την πρόβλεψη αριθμητικών τιμών.

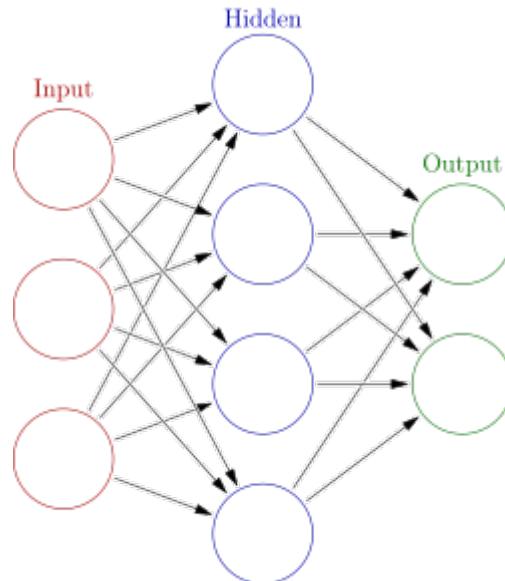
Βασίζεται στην μη γραμμική σχέση μεταξύ διαφορετικών τιμών.

- Λογιστική οπισθοδρόμηση (logistic regression), κατατάσσεται στην επιβλεπόμενη μάθηση και χρησιμοποιείται για την πρόβλεψη κατηγορικών αποτελεσμάτων, όπως δυαδικές απαντήσεις (Ναι/Οχι).
- Συσταδοποίηση (clustering) όπως αναφέρθηκε και παραπάνω ανήκει στη μη επιβλεπόμενή μάθηση. Ο αλγόριθμος αυτός εντοπίζει μοτίβα στα δεδομένα ώστε να γίνεται ομαδοποίηση. Χρησιμοποιείται στην αναγνώριση διαφορών μεταξύ στοιχείων.
- Δέντρα απόφασης (decision trees) μπορούν να χρησιμοποιηθούν για την παλινδρόμηση προβλέποντας αριθμητικές τιμές αλλά και στην ταξινόμηση κατηγοριοποιώντας δεδομένα. Λειτουργούν μέσω μιας ακολουθίας διακλαδούμενων αποφάσεων, οι οποίες μπορούν να απεικονιστούν με ένα δενδροειδές διάγραμμα. Το πλεονέκτημα τους είναι η διαφάνεια, καθώς τα καθιστά εύκολα στην επαλήθευση και στον έλεγχο.
- Τυχαίο δάσος (random forest), αυτός ο αλγόριθμος προβλέπει μια τιμή ή κατηγορία συνδυάζοντας τα αποτελέσματα από πολλαπλά δέντρα αποφάσεων. Η προσέγγιση βελτιώνει την ακρίβεια και μειώνει τον κίνδυνο υπερπροσαρμογής σε σύγκριση με τη χρήση ενός μόνο δέντρου αποφάσεων [66].

8.3 Νευρωνικά Δίκτυα

Ένα νευρωνικό δίκτυο αποτελεί ένα μοντέλο μηχανικής μάθησης που επεξεργάζεται δεδομένα με τρόπο αντίστοιχο με τη λειτουργία του ανθρώπινου εγκεφάλου. Μιμείται τη συνεργασία των βιολογικών νευρώνων για την αναγνώριση προτύπων, την εκτίμηση πιθανοτήτων και τη λήψη αποφάσεων [67].

Ένα νευρωνικό δίκτυο αποτελείται από διαδοχικά επίπεδα κόμβων ή τεχνητών νευρώνων: ένα επίπεδο εισόδου, ένα ή περισσότερα κρυφά επίπεδα και ένα επίπεδο εξόδου. Κάθε κόμβος συνδέεται με άλλους κόμβους και διαθέτει το δικό του βάρος και κατώφλι ενεργοποίησης. Όταν η έξοδος ενός κόμβου υπερβαίνει το καθορισμένο κατώφλι, ο κόμβος ενεργοποιείται και μεταβιβάζει δεδομένα στο επόμενο επίπεδο. Αν δεν ξεπεράσει το κατώφλι, δεν μεταδίδονται δεδομένα στο επόμενο στάδιο.



Σχήμα 8.2: Νευρωνικό δίκτυο [68].

Τα νευρωνικά δίκτυα χρησιμοποιούν δεδομένα εκπαίδευσης για να μάθουν και να αυξήσουν την ακρίβειά τους με την πάροδο του χρόνου. Όταν ρυθμιστούν σωστά, γίνονται ισχυρά εργαλεία στην επιστήμη των υπολογιστών και την τεχνητή νοημοσύνη, επιτρέποντας την ταχεία ταξινόμηση και ομαδοποίηση δεδομένων. Χάρη σε αυτά, εργασίες όπως η αναγνώριση ομιλίας ή εικόνων μπορούν να εκτελεστούν σε λίγα λεπτά, αντί για ώρες, αν γίνονταν χειροκίνητα από ανθρώπους. Ένα από τα πιο

διάσημα παραδείγματα εφαρμογής νευρωνικών δικτύων είναι ο αλγόριθμος αναζήτησης της Google.

Τα νευρωνικά δίκτυα αναφέρονται επίσης ως τεχνητά νευρωνικά δίκτυα (ANNs - Artificial Neural Networks) ή προσομοιωμένα νευρωνικά δίκτυα (SNNs - Simulated Neural Networks). Ανήκουν στη μηχανική μάθηση και αποτελούν τη βάση των σύγχρονων μοντέλων βαθιάς μάθησης [67].

8.3.1 Τύποι νευρωνικών δικτύων

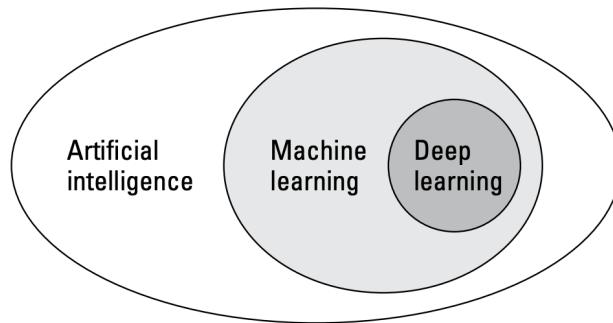
Τα νευρωνικά δίκτυα διακρίνονται σε διάφορους τύπους, καθένας από τους οποίους χρησιμοποιείται για συγκεκριμένες εφαρμογές. Αν και η παρακάτω λίστα δεν είναι πλήρης, περιλαμβάνει ορισμένους από τους πιο κοινούς τύπους νευρωνικών δικτύων και τις βασικές τους χρήσεις.

- Τα νευρωνικά δίκτυα προώθησης (Feedforward Neural Networks), γνωστά και ως νευρωνικά δίκτυα πολυεπίπεδης αντίληψης (MLPs - Multi-Layer Perceptrons), αποτελούνται από ένα επίπεδο εισόδου, ένα ή περισσότερα κρυφά επίπεδα και ένα επίπεδο εξόδου. Αν και συχνά αναφέρονται ως MLPs, στην πραγματικότητα αποτελούνται από σιγμοειδείς νευρώνες αντί για αντιληπτές (perceptron), καθώς τα περισσότερα προβλήματα του πραγματικού κόσμου είναι μη γραμμικά. Αυτά τα μοντέλα εκπαιδεύονται επεξεργαζόμενα δεδομένα και αποτελούν τη βάση για εφαρμογές όπως η υπολογιστική όραση, η επεξεργασία φυσικής γλώσσας και άλλα συστήματα που βασίζονται σε νευρωνικά δίκτυα.
- Τα συνελικτικά νευρωνικά δίκτυα (CNNs - Convolutional Neural Networks) μοιάζουν με τα νευρωνικά δίκτυα προώθησης, αλλά χρησιμοποιούνται κυρίως για αναγνώριση εικόνων, αναγνώριση μοτίβων και υπολογιστική όραση. Αυτά τα δίκτυα αξιοποιούν αρχές

της γραμμικής άλγεβρας, ειδικότερα τον πολλαπλασιασμό πινάκων, για την ανίχνευση μοτίβων σε εικόνες.

- Τα αναδρομικά νευρωνικά δίκτυα (RNNs - Recurrent Neural Networks) ξεχωρίζουν λόγω των βρόχων ανάδρασης που περιέχουν. Χρησιμοποιούνται κυρίως για την ανάλυση χρονοσειρών και την πρόβλεψη μελλοντικών τάσεων, όπως οι διακυμάνσεις της χρηματιστηριακής αγοράς ή οι προβλέψεις πωλήσεων [67].

8.4 Βαθιά μάθηση



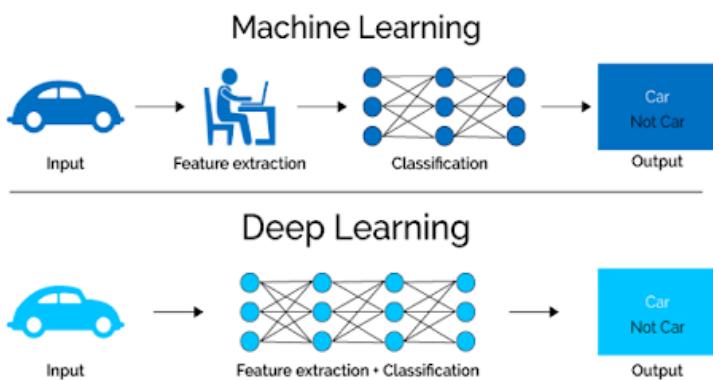
Σχήμα 8.3: Βαθιά μάθηση [69].

Η βαθιά μάθηση (Deep learning) είναι ένας εξειδικευμένος κλάδος της μηχανικής μάθησης που βασίζεται σε πολυεπίπεδα νευρωνικά δίκτυα, γνωστά ως βαθιά νευρωνικά δίκτυα (Deep neural networks), για να αναπαράγει τις περίπλοκες διαδικασίες λήψης αποφάσεων του ανθρώπινου εγκεφάλου. Σήμερα, η βαθιά μάθηση αποτελεί τη βάση για πολλές εφαρμογές τεχνητής νοημοσύνης (AI – Artificial intelligence) που συναντάμε καθημερινά.

8.4.1 Διαφορές μηχανικής με βαθιάς μάθησης

Η βασική διαφορά μεταξύ βαθιάς μάθησης και παραδοσιακής μηχανικής μάθησης έγκειται στην πολυπλοκότητα της αρχιτεκτονικής των νευρωνικών δικτύων τους. Τα συμβατικά, απλούστερα μοντέλα μηχανικής μάθησης χρησιμοποιούν απλά νευρωνικά δίκτυα με μία ή δύο υπολογιστικές στρώσεις. Αντίθετα, τα μοντέλα βαθιάς μάθησης διαθέτουν τρεις ή περισσότερες στρώσεις και συνήθως εκατοντάδες ή και χιλιάδες για να βελτιστοποιούν τη διαδικασία εκμάθησης και να αυξάνουν την απόδοσή τους.

Σε αντίθεση με τα εποπτευόμενα μοντέλα μάθησης, τα οποία απαιτούν δομημένα και με ετικέτες δεδομένα εισόδου για ακριβείς προβλέψεις, τα μοντέλα βαθιάς μάθησης μπορούν να λειτουργούν μέσω μη εποπτευόμενης μάθησης. Αυτό τους επιτρέπει να αναγνωρίζουν μοτίβα, να εξάγουν βασικά χαρακτηριστικά και να εντοπίζουν σχέσεις μέσα από ακατέργαστα, μη δομημένα δεδομένα. Επιπλέον, τα μοντέλα βαθιάς μάθησης έχουν την ικανότητα να βελτιώνουν σταδιακά τις προβλέψεις τους, αυξάνοντας την ακρίβειά τους με την πάροδο του χρόνου.



Σχήμα 8.4: Διαφορά μηχανικής με βαθιάς μάθησης [70].

8.4.2 Βαθιά μάθηση και προηγμένα νευρωνικά δίκτυα

Η βαθιά μάθηση χρησιμοποιεί νευρωνικά δίκτυα και πιο συγκεκριμένα βαθιά νευρωνικά δίκτυα για την εκπαίδευση ενός μοντέλου. Αν τα νευρωνικά δίκτυα έχουν πολλά επίπεδα τότε ονομάζονται βαθιά νευρωνικά δίκτυα, όπου κάθε επίπεδο βασίζεται στο προηγούμενο για να βελτιώσει την ακρίβεια της πρόβλεψης ή της ταξινόμησης. Αυτή η διαδοχική επεξεργασία μέσα στο δίκτυο ονομάζεται προώθηση προς τα εμπρός (forward propagation). Τα επίπεδα εισόδου και εξόδου, γνωστά ως ορατά επίπεδα (visible layers), έχουν διαφορετικούς ρόλους, το επίπεδο εισόδου λαμβάνει τα δεδομένα για επεξεργασία, ενώ το επίπεδο εξόδου παράγει την τελική πρόβλεψη ή ταξινόμηση [71].

Μια άλλη διαδικασία, είναι η οπισθοδιάδοση (backpropagation), χρησιμοποιεί αλγορίθμους όπως η κατάβαση βαθμίδας (gradient descent) για να υπολογίσει τα σφάλματα στις προβλέψεις και να προσαρμόσει τα βάρη και τις προκαταλήψεις, κινούμενη προς τα πίσω στα επίπεδα του δικτύου. Ο συνδυασμός της προώθησης προς τα εμπρός και της οπισθοδιάδοσης επιτρέπει στα νευρωνικά δίκτυα να βελτιώνουν τις προβλέψεις τους και να μειώνουν τα σφάλματα με την πάροδο του χρόνου. Επιπλέον, υπάρχουν και άλλα μοντέλα που χρησιμοποιούνται στην βαθιά μάθηση, κάποια από αυτά είναι:

- CNNs.
- RNNs.
- Αυτοκωδικοποιητές παραλλαγών (VAEs - variational autoencoders). Οι αυτόματοι κωδικοποιητές λειτουργούν κωδικοποιώντας δεδομένα χωρίς ετικέτα σε μια συμπιεσμένη αναπαράσταση και στη συνέχεια αποκωδικοποιώντας τα δεδομένα πίσω στην αρχική τους μορφή. Η λειτουργεία αυτή χρησιμοποιήθηκε για την ανακατασκευή κατεστραμμένων ή θολών εικόνων. Ακόμα, οι αυτοκωδικοποιητές παραλλαγών πρόσθεσαν

επιπλέων την παραγωγή παραλλαγών στα αρχικά δεδομένα.

- Γενετικά αντιπαραθετικά δίκτυα (GANs - Generative adversarial networks) είναι νευρωνικά δίκτυα που χρησιμοποιείται για την δημιουργία ψεύτικων αλλά ρεαλιστικών δεδομένων.
- Μοντέλο διάχυσης είναι γενετικά μοντέλα που εκπαιδεύονται χρησιμοποιώντας τη διαδικασία διάχυσης προς τα εμπρός και προς τα πίσω, μέσω μιας προοδευτικής προσθήκης και αφαίρεσης θορύβου.
- Τα μοντέλα Transformers είναι προηγμένα νευρωνικά δίκτυα που επεξεργάζονται δεδομένα παράλληλα και έχουν προάγει τη Φυσική Επεξεργασία Γλώσσας (NLP - Natural Language Processing) και άλλες εφαρμογές. Χρησιμοποιούν μια αρχιτεκτονική κωδικοποιητή-αποκωδικοποιητή όπου:
 - ο κωδικοποιητής (encoder) μετατρέπει το ακατέργαστο κείμενο σε embeddings, μια συμπυκνωμένη αναπαράσταση της σημασίας του κειμένου.
 - Ο αποκωδικοποιητής (decoder) λαμβάνει αυτά τα embeddings και προβλέπει διαδοχικά την επόμενη λέξη ή φράση, δημιουργώντας κατανοητό κείμενο [71].

8.4.3 Απαιτήσεις και εφαρμογές βαθιάς μάθησης

Η εκπαίδευση μοντέλων βαθιάς μάθησης απαιτεί τεράστια υπολογιστική ισχύ. Οι μονάδες επεξεργασίας γραφικών (GPUs) είναι ιδιαίτερα κατάλληλες για αυτές τις εργασίες, καθώς μπορούν να εκτελούν πολύπλοκους υπολογισμούς σε πολλαπλούς πυρήνες και διαθέτουν μεγάλη μνήμη. Επιπλέον, η κατανεμημένη υπολογιστική ισχύς στο cloud μπορεί να υποστηρίξει περαιτέρω αυτές τις διεργασίες. Δεδομένου ότι η εκπαίδευση βαθιών αλγορίθμων απαιτεί υψηλή υπολογιστική

απόδοση, η διαχείριση πολλαπλών GPUs τοπικά μπορεί να επιβαρύνει σημαντικά τους εσωτερικούς πόρους και να είναι ιδιαίτερα δαπανηρή για κλιμάκωση. Όσον αφορά το λογισμικό, οι περισσότερες εφαρμογές βαθιάς μάθησης αναπτύσσονται χρησιμοποιώντας JAX, PyTorch ή TensorFlow.

Η βαθιά μάθηση διαδραματίζει κεντρικό ρόλο στην επιστήμη δεδομένων, προωθώντας διάφορες εφαρμογές και υπηρεσίες που διευκολύνουν την αυτοματοποίηση. Επιτρέπει στις τεχνολογίες τεχνητής νοημοσύνης να εκτελούν αναλυτικές και φυσικές εργασίες με ελάχιστη ανθρώπινη παρέμβαση. Πολλές καινοτομίες που χρησιμοποιούμε καθημερινά όπως οι ψηφιακοί βοηθοί, τα συστήματα ανίχνευσης απάτης και τα αυτόνομα οχήματα βασίζονται στη βαθιά μάθηση [71].

ΠΕΙΡΑΜΑΤΙΚΟ ΜΕΡΟΣ

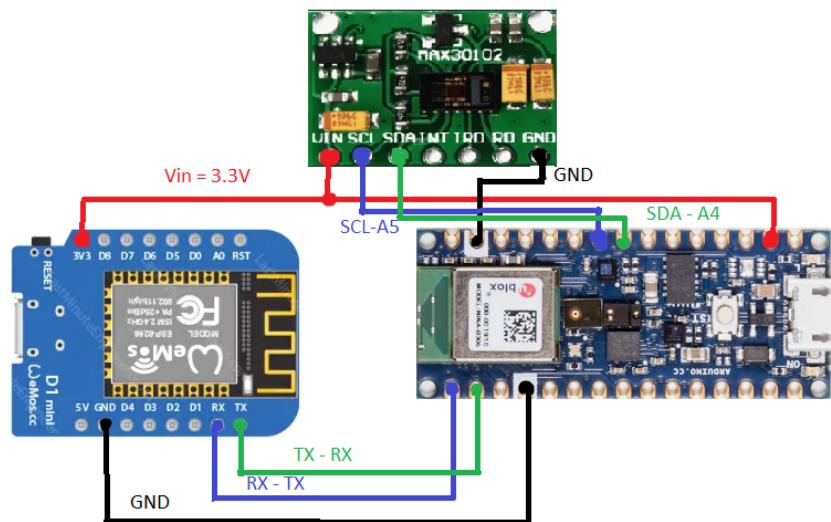
9 Σκοπός Πειραματικού μέρους

Ο σκοπός του πειραματικού μέρους είναι η υλοποίηση ενός συστήματος για απομακρυσμένη μέτρηση ενός ασθενούς με αισθητήρες που καταγράφουν την ζωτική λειτουργία. Ειδικότερα καταγράφουν το ποσοστό οξυγόνου στο αίμα, τους παλμούς της καρδίας και γίνετε απεικόνιση του ηλεκτροκαρδιογραφήματος της καρδίας. Οι μετρήσεις αυτές έγιναν ξεχωριστά. Ακόμα, αναπτύσσεται ένα μοντέλο τεχνητής νοημοσύνης AI (artificial intelligent) με την χρήση κάμερας για την υπολογιστική όραση και με συνδυασμό του αισθητήρα μέτρησης παλμών.

9.1 Υλοποίηση κυκλώματος μέτρησης καρδιακών παλμών και οξυγόνου

Για την μέτρηση παλμών χρησιμοποιήθηκε το Arduino nano 33 Ble Sence ως ένας μικροϋπολογιστής για την φόρτωση και εκτέλεση του κώδικα και την υποστηρίξη του Max30102 και το

ESP-8266 Wemos D1 mini. Το Max30102 είναι ο αισθητήρας που αναγνωρίζει τους καρδιακούς παλμούς και στέλνει σειριακά τα δεδομένα στο Arduino nano 33 Ble μέσω καλωδίων σύνδεσης (jumper wires). Συγκεκριμένα η σύνδεση γίνεται μεταξύ των ακίδων του MAX30102 και το Arduino nano 33 Ble Sense ως εξής Vin – 3.3V, SCL – A5 (SCL), SDA – A4 (SDA) και GND - GND. Αντίστοιχα το ESP-8266 συνδέεται σειριακά με Arduino 33 Ble μέσω των ακίδων TX - RX και RX – TX. Οι ακίδες πρέπει να είναι με αυτήν την σύνδεση διότι το Arduino πρέπει να στέλνει δεδομένα και το ESP να λαμβάνει δεδομένα και αντίστροφα. Η τροφοδοσία του ESP-8266 μπορεί να γίνει μέσω της θύρας τύπου C, όπου θα γίνει και η φόρτωση του κώδικα ή από το Arduino nano 33 μέσω των ακίδων 3.3V και GRN. Η τροφοδοσία και η φόρτωση του κώδικα στο Arduino θα γίνει με την θύρα micro που διαθέτη, αλλιώς θα μπορούσε να χρησιμοποιηθεί μια πηγή 5V συνεχούς ρεύματος (DC) στις ακίδες Vin για τον θετικό πόλο και GRN για το αρνητικό πόλο. Επιπλέον, η ίδια συνδεσμολογία χρησιμοποιήθηκε και για την μέτρηση του οξυγόνου.



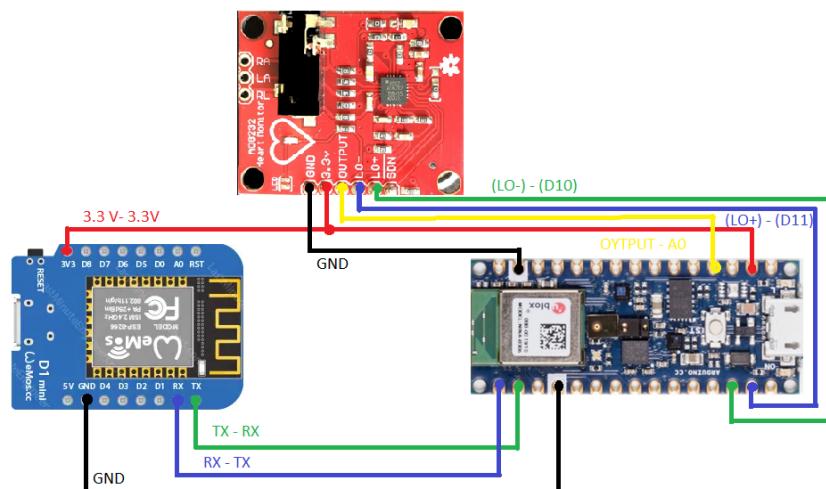
Σχήμα 9.1: Κύκλωμα μέτρησης καρδιακών παλμών και οξυγόνου.

Η εφαρμογή που χρησιμοποιήθηκε για την δημιουργία, επεξεργασία και φόρτωση του κώδικα στις πλακέτες (Arduino και ESP) είναι το λειτουργικό περιβάλλον του Arduino, το Arduino IDE. Ο κώδικας είναι σε γλώσσα C++.

Το λογισμικό του Arduino μπορεί να βρεθεί στην ιστοσελίδα του Arduino [49]. Επιλέγοντας το κατάλληλο λογισμικό (Windows, macOS, Linux) για εγκατάσταση.

9.2 Υλοποίηση κυκλώματος απεικόνισης καρδιογραφήματος

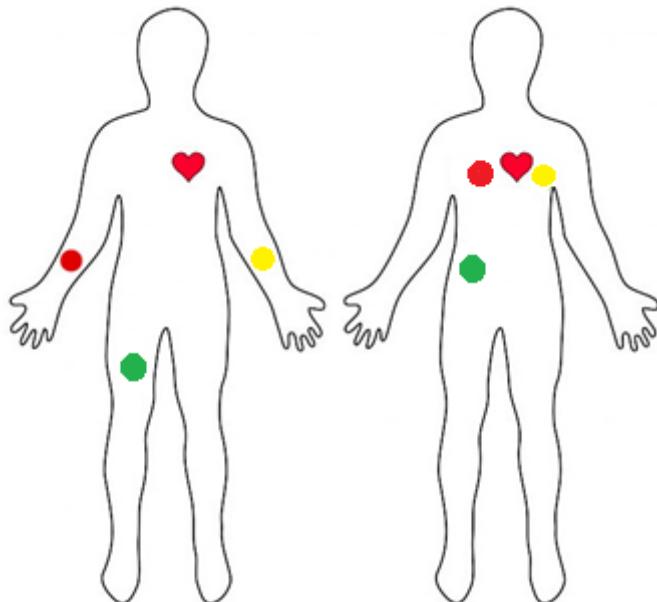
Για την απεικόνιση του καρδιογραφήματος χρησιμοποιήθηκε ένα Arduino Nano 33 Ble Sense ως ένας μικροελεκτής, ένα ESP-8266 για την σύνδεση στο δίκτυο και ένας αισθητήρας AD8232 που αντλεί δεδομένα από την καρδιά. Η συνδεσμολογία διακρίνεται στο σχήμα 1.55. Ειδικότερα, η ακίδα OUTPUT του AD8232 συνδέεται με την αναλογική ακίδα A0 του Arduino. Οι ακίδες Lo- και Lo+ με τις ψηφιακές ακίδες D10 και D11. Η τροφοδοσία γίνεται με 3.3V. Όσο αφορά για το ESP η συνδεσμολογία παραμένει η ίδια.



Σχήμα 9.2: Κύκλωμα απεικόνισης καρδιογραφήματος.

Στο παρακάτω σχήμα διακρίνονται τα σημεία που μπορούν να τοποθετηθούν τα τρία ηλεκτρόδια (Αριστερά, Πάνω δεξιά και

κάτω δεξιά) από τον αισθητήρα AD8232. Πάνω στα ηλεκτρόδια τοποθετούνται αυτοκόλλητα (Pads) μίας χρήσης και περιέχουν υγρό διάλυμα ηλεκτρολυτών, ώστε να γίνεται καλύτερη επαφή στο δέρμα μειώνοντας τον θόρυβο.

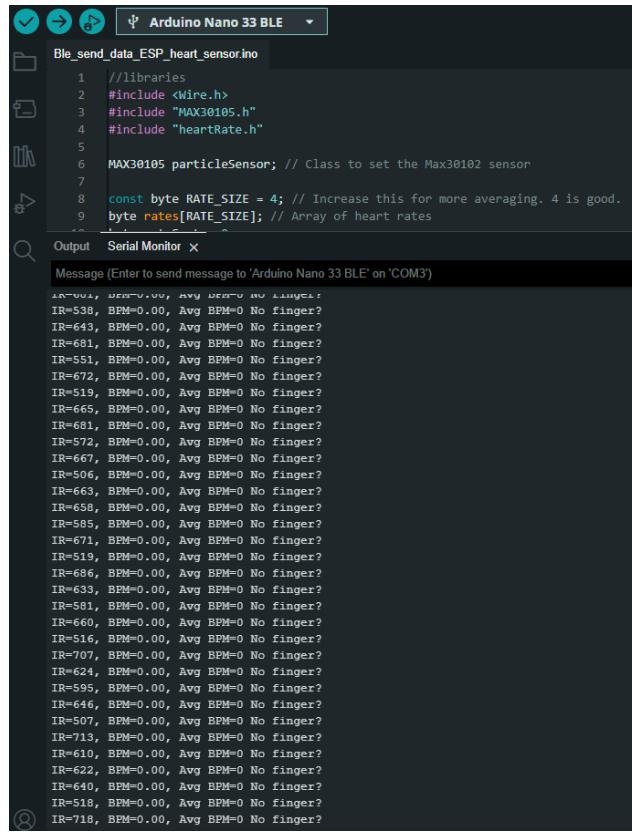


Εικόνα 9.1: Ενδεικτικά μέρη τοποθέτησης ηλεκτροδίων στο σώμα.

9.3 Αποτελέσματα κυκλωμάτων

Τα δεδομένα μπορούν να εμφανιστούν στο serial Monitor αν είναι συνδεδεμένο το Arduino στον υπολογιστή, αλλά και μέσω της εφαρμογής IoT MQTT Panel. Με αυτήν την εφαρμογή υλοποιείται η εξ αποστάσεως τηλεμετρία, αρκεί να υπάρχει σύνδεση στο διαδίκτυο και από τους δύο χρήστες.

9.3.1 Αποτελέσματα εφαρμογής καρδιακών παλμών



The screenshot shows the Arduino IDE interface with the file 'Ble_send_data_ESP_heart_sensor.ino' open. The code includes libraries for WIRE, MAX30105, and heartRate. It defines a class 'MAX30105 particleSensor' and initializes an array 'rates' of size RATE_SIZE (4). The Serial Monitor window displays a continuous stream of data from the sensor, showing values for IR and BPM for each finger (0 to 7) and an average value.

```
//libraries
#include <Wire.h>
#include "MAX30105.h"
#include "heartRate.h"

MAX30105 particleSensor; // Class to set the Max30102 sensor

const byte RATE_SIZE = 4; // Increase this for more averaging. 4 is good.
byte rates[RATE_SIZE]; // Array of heart rates

Message (Enter to send message to 'Arduino Nano 33 BLE' on 'COM3')
IR=001, BPM=0.00, Avg BPM=0 No finger?
IR=538, BPM=0.00, Avg BPM=0 No finger?
IR=643, BPM=0.00, Avg BPM=0 No finger?
IR=681, BPM=0.00, Avg BPM=0 No finger?
IR=551, BPM=0.00, Avg BPM=0 No finger?
IR=672, BPM=0.00, Avg BPM=0 No finger?
IR=519, BPM=0.00, Avg BPM=0 No finger?
IR=665, BPM=0.00, Avg BPM=0 No finger?
IR=601, BPM=0.00, Avg BPM=0 No finger?
IR=572, BPM=0.00, Avg BPM=0 No finger?
IR=667, BPM=0.00, Avg BPM=0 No finger?
IR=506, BPM=0.00, Avg BPM=0 No finger?
IR=663, BPM=0.00, Avg BPM=0 No finger?
IR=658, BPM=0.00, Avg BPM=0 No finger?
IR=585, BPM=0.00, Avg BPM=0 No finger?
IR=671, BPM=0.00, Avg BPM=0 No finger?
IR=519, BPM=0.00, Avg BPM=0 No finger?
IR=686, BPM=0.00, Avg BPM=0 No finger?
IR=633, BPM=0.00, Avg BPM=0 No finger?
IR=581, BPM=0.00, Avg BPM=0 No finger?
IR=660, BPM=0.00, Avg BPM=0 No finger?
IR=516, BPM=0.00, Avg BPM=0 No finger?
IR=707, BPM=0.00, Avg BPM=0 No finger?
IR=624, BPM=0.00, Avg BPM=0 No finger?
IR=595, BPM=0.00, Avg BPM=0 No finger?
IR=646, BPM=0.00, Avg BPM=0 No finger?
IR=507, BPM=0.00, Avg BPM=0 No finger?
IR=713, BPM=0.00, Avg BPM=0 No finger?
IR=610, BPM=0.00, Avg BPM=0 No finger?
IR=622, BPM=0.00, Avg BPM=0 No finger?
IR=640, BPM=0.00, Avg BPM=0 No finger?
IR=518, BPM=0.00, Avg BPM=0 No finger?
IR=718, BPM=0.00, Avg BPM=0 No finger?
```

Εικόνα 9.2: Αποτελέσματα χωρίς να ανιχνεύει ο αισθητήρας καρδιακών παλμών.

The screenshot shows the Arduino IDE interface with the title bar "Arduino Nano 33 BLE". A file named "Ble_send_data_ESP_heart_sensorino" is open. In the bottom right corner of the code editor, there is a small circular icon with a question mark inside.

The Serial Monitor window is active, displaying a continuous stream of data. The data consists of two columns: "IR" and "BPM". The "IR" column lists values from 130011 to 130734. The "BPM" column lists values mostly at 66, with some variations like 69.28. The output ends with the message "Sending data: 66".

```
// Libraries
#include <Wire.h>
#include "MAX30105.h"
#include "heartRate.h"

MAX30105 particleSensor; // Class to set the Max30102 sensor
const byte RATE_SIZE = 4; // Increase this for more averaging. 4 is good
byte rates[RATE_SIZE]; // Array of heart rates

Message (Enter to send message to 'Arduino Nano 33 BLE' on 'COM3')
IR=130011, BPM 66.00, Avg BPM 66
IR=130864, BPM=69.28, Avg BPM=66
IR=130836, BPM=69.28, Avg BPM=66
IR=130764, BPM=69.28, Avg BPM=66
IR=130695, BPM=69.28, Avg BPM=66
IR=130533, BPM=69.28, Avg BPM=66
IR=130522, BPM=69.28, Avg BPM=66
IR=130458, BPM=69.28, Avg BPM=66
IR=130469, BPM=69.28, Avg BPM=66
IR=130483, BPM=69.28, Avg BPM=66
IR=130474, BPM=69.28, Avg BPM=66
IR=130563, BPM=69.28, Avg BPM=66
IR=130533, BPM=69.28, Avg BPM=66
IR=130552, BPM=69.28, Avg BPM=66
IR=130560, BPM=69.28, Avg BPM=66
IR=130523, BPM=69.28, Avg BPM=66
IR=130610, BPM=69.28, Avg BPM=66
IR=130570, BPM=69.28, Avg BPM=66
IR=130594, BPM=69.28, Avg BPM=66
IR=130607, BPM=69.28, Avg BPM=66
IR=130582, BPM=69.28, Avg BPM=66
IR=130642, BPM=69.28, Avg BPM=66
IR=130596, BPM=69.28, Avg BPM=66
IR=130628, BPM=69.28, Avg BPM=66
IR=130606, BPM=69.28, Avg BPM=66
IR=130625, BPM=69.28, Avg BPM=66
IR=130692, BPM=69.28, Avg BPM=66
IR=130656, BPM=69.28, Avg BPM=66
IR=130696, BPM=69.28, Avg BPM=66
Sending data: 66
IR=130665, BPM=69.28, Avg BPM=66
IR=130696, BPM=69.28, Avg BPM=66
IR=130734, BPM=69.28, Avg BPM=66
```

Εικόνα 9.3: Αποτελέσματα με δάκτυλο πάνω στον αισθητήρα.

Παρατηρείται ότι όταν δεν έχει τοποθετηθεί το δάκτυλο πάνω στον αισθητήρα το υπέρυθρο φως είναι μικρό, ενώ γίνεται μεγάλο όταν τοποθετηθεί το δάκτυλο στον αισθητήρα. Αυτό συμβαίνει διότι το υπέρυθρο φως «ανακιλάται» από το δάκτυλο. Τα δεδομένα στέλνονται μέσω της σειριακής επικοινωνίας στο Esp κάθε 8 δευτερόλεπτα.

The screenshot shows the Arduino IDE interface. At the top, there are three circular icons: a green checkmark, a blue arrow, and a red arrow. To their right is a dropdown menu with the text "LOLIN(WEMOS) D1 R2...". Below this is a code editor window titled "ESP_send_Data_MQTT.ino" containing the following code:

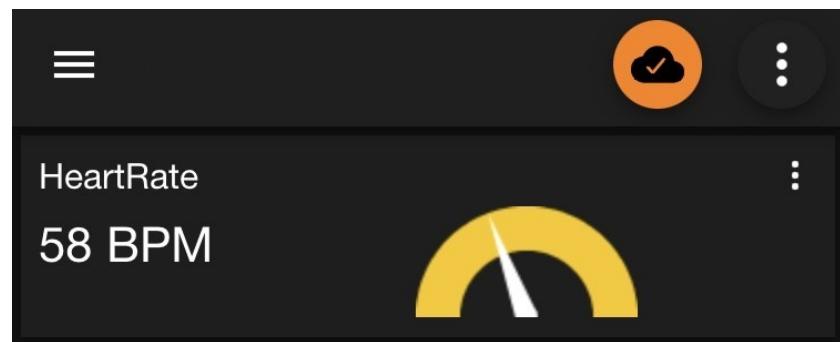
```
1 // libraries
2 #include <ESP8266WiFi.h>
3 #include <PubSubClient.h>
```

Below the code editor are two tabs: "Output" and "Serial Monitor". The "Output" tab is selected and displays the following text from the serial monitor:

```
.....
WiFi connected
IP address:
172.20.10.4
Attempting MQTT connection...connected
Received data: 58
Data sent to MQTT broker
Attempting MQTT connection...connected
```

Εικόνα 9.4: Αποτέλεσμα του Esp.

Στο εικόνα 9.4 παρατηρούμε ότι το Esp έχει συνδεθεί στο δίκτυο, στο MQTT διαμεσολαβητή και έχει λάβει τα δεδομένα από το Arduino. Έπειτα το Esp στέλνει τα δεδομένα στον MQTT διαμεσολαβητή (στην συγκεκριμένη περίτωση είναι το mosquito.org), όπου τα δεδομένα θα ληφθούν από την εφαρμογή IoTMQTTPanel που λειτουργεί ως πελάτης.

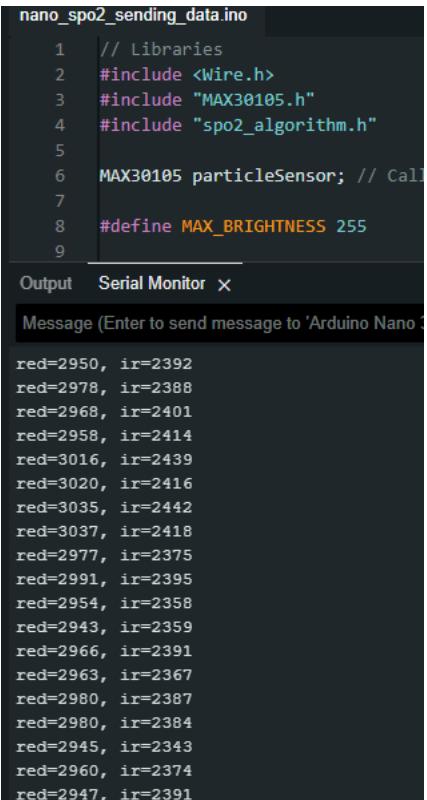


Εικόνα 9.5: Απεικόνιση δεδομένων μέσω τη εφαρμογής IoTMQTTPanel.

Το σήμα του σύννεφου πάνω στην εικόνα 9.5 δεξιά, δηλώνει ότι υπάρχει σύνδεση με τον MQTT διαμεσολαβητή.

9.3.2 Αποτελέσματα της μέτρησης του οξυγόνου

Κατά την εκκίνηση του κώδικα οι πρώτες 100 μετρήσεις είναι δείγματα που αποθηκεύονται στην προσωρινή μνήμη (buffer). Έτσι προσδιορίζεται το εύρος των σήματος.



The screenshot shows the Arduino IDE interface with the code editor open and the Serial Monitor tab selected. The code editor contains the following C++ code:

```
nano_spo2_sending_data.ino
1 // Libraries
2 #include <Wire.h>
3 #include "MAX30105.h"
4 #include "spo2_algorithm.h"
5
6 MAX30105 particleSensor; // Call
7
8 #define MAX_BRIGHTNESS 255
9
```

The Serial Monitor window displays a series of sensor readings:

```
red=2950, ir=2392
red=2978, ir=2388
red=2968, ir=2401
red=2958, ir=2414
red=3016, ir=2439
red=3020, ir=2416
red=3035, ir=2442
red=3037, ir=2418
red=2977, ir=2375
red=2991, ir=2395
red=2954, ir=2358
red=2943, ir=2359
red=2966, ir=2391
red=2963, ir=2367
red=2980, ir=2387
red=2980, ir=2384
red=2945, ir=2343
red=2960, ir=2374
red=2947, ir=2391
```

Εικόνα 9.6: Απεικόνιση αποτελεσμάτων εφαρμογής εύρεσης οξυγόνου.

The screenshot shows the Arduino IDE interface. The code editor window contains the sketch `nano_spo2_sending_data.ino` with the following content:

```
1 // Libraries
2 #include <Wire.h>
3 #include "MAX30105.h"
4 #include "spo2_algorithm.h"
5
6 MAX30105 particleSensor; // Call class
7
8 #define MAX_BRIGHTNESS 255
```

The serial monitor window is titled "Serial Monitor" and displays the following output:

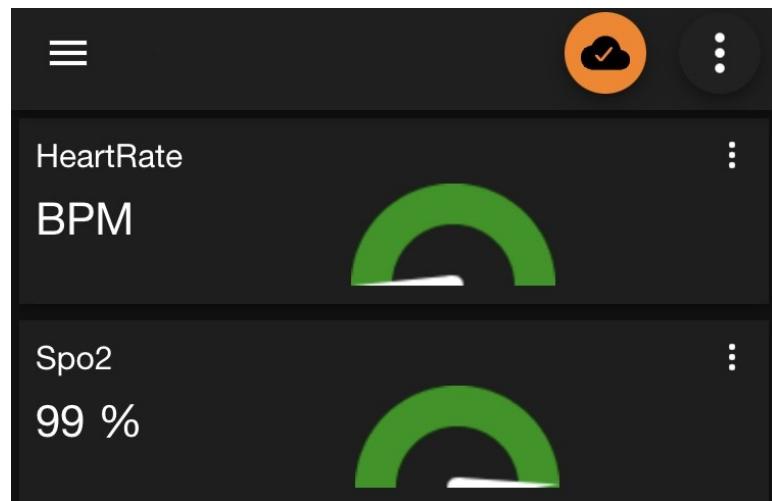
```
Message (Enter to send message to 'Arduino Nano 33 BLE' on 'COM3')

red=217456, ir=260887, SPO2=100, SPO2Valid=1
red=217059, ir=259312, SPO2=100, SPO2Valid=1
red=216883, ir=259261, SPO2=100, SPO2Valid=1
red=216965, ir=259467, SPO2=100, SPO2Valid=1
red=216948, ir=259418, SPO2=100, SPO2Valid=1
red=216995, ir=259793, SPO2=100, SPO2Valid=1
red=217098, ir=260058, SPO2=100, SPO2Valid=1
red=217201, ir=260383, SPO2=100, SPO2Valid=1
Sending data: 100
red=217258, ir=260663, SPO2=100, SPO2Valid=1
```

Εικόνα 9.7: Απεικόνιση δεδομένων οξυγόνου.

Παρατηρείται ότι το κόκκινο και το υπέρυθρο φως έχει μεγάλη τιμή. Το οξυγόνο είναι στο μέγιστο όριο 100% και η μεταβλητή Spo2Valid υποδηλώνει αν έχει τοποθετηθεί το δάκτυλο πάνω στον αισθητήρα. Επίσης στέλνονται τα δεδομένα στο ESP.

Η λειτουργία του ESP είναι η ίδια με την προηγουμένη εφαρμογή μέτρησης καρδιακών παλμών. Δηλαδή συνδέεται με το δίκτυο και με τον MQTT διαμεσολαβητή, λαμβάνει από το Arduino και στέλνει τα δεδομένα στον MQTT διαμεσολαβητή. Από το Serial Monitor επιβεβαιώνουμε ότι η παραπάνω διαδικασία λειτουργεί σωστά.



Εικόνα 9.7: Απεικόνιση δεδομένων οξυγόνου μέσω της εφαρμογής IoTMQTTPanel.

9.3.3 Αποτελέσματα απεικόνισης ηλεκτροκαρδιογραφήματος

Τα αποτελέσματα από το Serial Plotter του προγράμματος Arduino IDE απεικονίζονται στις παρακάτω εικόνες.



Εικόνα 9.8: Απεικόνιση ηλεκτροκαρδιογραφήματος μέσω Serial Plotter.

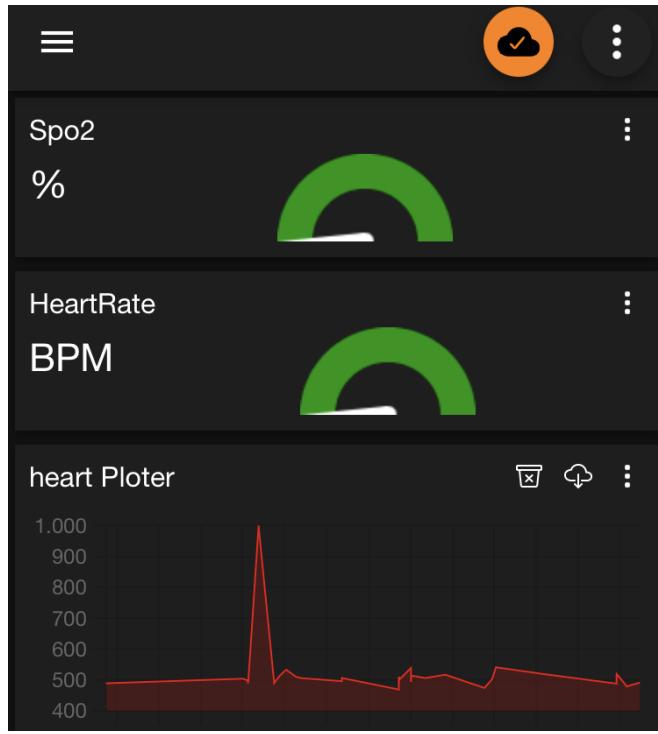
Ο άξονας X έχει μονάδα μέτρησης τον χρόνο, ενώ ο άξονας Y έχει μονάδα μέτρησης την τάση. Διακρίνεται ότι υπάρχει ίχνος θορύβου στο σήμα. Αυτό μπορεί να υπάρχει για διάφορους λόγους όπως:

- Η καλωδίωση, να μην υπάρχει καλή επαφή στην συνδεσμολογία μεταξύ αισθητήρα AD8232 και Arduino.
- Εξωτερικές παρεμβολές, από άλλες συσκευές που βρίσκονται κοντά.
- Παρεμβολές από την σειριακή θύρα σύνδεσης με τον υπολογιστή.

```
1  Data sent to MQTT broker
2  Attempting MQTT connection...connected
3  Received data: 514
4  Data sent to MQTT broker
5  Received data: 501
6  Data sent to MQTT broker
7  Received data: 515
8  Data sent to MQTT broker
9  Received data: 490
10 Data sent to MQTT broker
11 Attempting MQTT connection...connected
12 Attempting MQTT connection...connected
13 Received data: 462
14 Data sent to MQTT broker
15 Attempting MQTT connection...connected
16 Attempting MQTT connection...connected
17 Received data: 491
18 Data sent to MQTT broker
19 Attempting MQTT connection...connected
20 Attempting MQTT connection...connected
21 Received data: 495
22 Data sent to MQTT broker
23 Attempting MQTT connection...connected
24 Attempting MQTT connection...connected
25 Received data: 518
26 Failed to send data to MQTT broker
27 Attempting MQTT connection...connected
28 Attempting MQTT connection...connected
29 Received data: 504
30 Data sent to MQTT broker
31 Attempting MQTT connection...connected
32 Attempting MQTT connection...connected
33 Received data: 501
34 Data sent to MQTT broker
35 Attempting MQTT connection...connected
36 Attempting MQTT connection...connected
37 Received data: 484
38 Data sent to MQTT broker
39 Attempting MQTT connection...connected
40 Received data: 508
```

Εικόνα 9.9: Δεδομένα Serial Monitor ESP.

Παρατηρείται ότι τα δεδομένα λαμβάνονται από το Arduino, στέλνονται στο MQTT διαμεσολαβητή και αποσυνδέεται αρκετές φορές από τον MQTT διαμεσολαβητή. Αυτό μπορεί να συμβαίνει λόγω εξοικονόμησης πόρων και κατανάλωσης ή κακής σύνδεσης με το δίκτυο.



Εικόνα 9.10: Ηλεκτροκαρδιογράφημα μέσω της εφαρμογής IoTMQTTPanel.

Παρατηρείται ότι υπάρχουν πολλές απώλειες στα δεδομένα από το ESP στην εφαρμογή IoTMQTTPanel. Αυτό μπορεί να συμβένει λόγο κακής σύνδεσης με το διαδίκτυο ή λόγο των αποσυνδέσων από του σέρβερ.

9.3.4 Αποτελέσματα του μοντέλου αναγνώρισης στάσης ανθρώπου

Το μοντέλο αναγνώρισης στάσης (καθιστός, όρθιος, ξαπλωμένος) ανθρώπου δημιουργήθηκε με τη χρήση της πλατφόρμας Edge Impulse, όπου εκπαιδεύτηκε από την αρχή με δείγματα δεδομένων που συλλέχθηκαν μέσω κάμερας. Για την αναγνώριση χρησιμοποιήθηκε το FOMO (Faster Objects, More Objects), ένα ελαφρύ μοντέλο ανίχνευσης αντικειμένων βασισμένο σε νευρωνικά δίκτυα, το οποίο είναι βελτιστοποιημένο για συσκευές χαμηλής υπολογιστικής ισχύος. Περισσότερες λεπτομέρειες για τη διαδικασία εκπαίδευσης και τα δεδομένα αναλύονται στο Παράρτημα A.

```

Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 3 ms., Classification: 208 ms., Anomaly: 0 ms.):
Object detection bounding boxes:

Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 3 ms., Classification: 209 ms., Anomaly: 0 ms.):
Object detection bounding boxes:
    standing (0.503906) [ x: 16, y: 24, width: 8, height: 8 ]

Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 3 ms., Classification: 209 ms., Anomaly: 0 ms.):
Object detection bounding boxes:
    standing (0.941406) [ x: 16, y: 24, width: 8, height: 8 ]

Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 3 ms., Classification: 209 ms., Anomaly: 0 ms.):
Object detection bounding boxes:
    standing (0.957031) [ x: 16, y: 24, width: 8, height: 8 ]

```

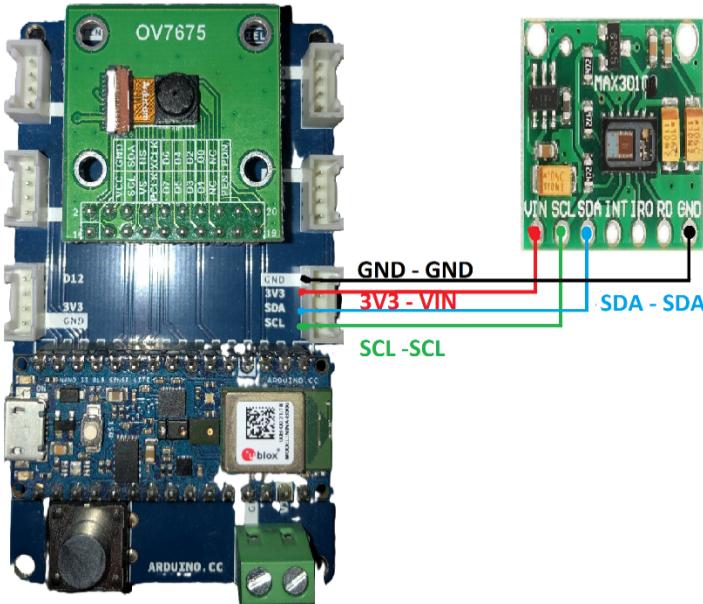
Εικόνα 9.11: Αποτέλεσμα μοντέλου αναγνώρισης στάσης.

Διακρίνεται ότι στην αρχή υπάρχει μικρό ποσοστό αναγνώρισης είναι 50.3%, ενώ όσο περνάει ο χρόνος και λαμβάνονται περισσότερες φωτογραφίες το ποσοστό αυξάνεται. Αυτό μπορεί να μειωθεί με ένα καλύτερο υπολογιστικά υλισμικό ή με την υλοποίηση ενός βελτιωμένου μοντέλου ανίχνευσης με μεγαλύτερη ακρίβεια. Επίσης παρατηρήθηκε ότι υπάρχει καθυστέρηση στη κάμερα σε σχέση με την πραγματικότητα. Αυτό μπορεί να συμβαίνει λόγο κακής σύνδεσης με το διαδίκτυο ή λόγο μικρής υπολογιστικής ισχύς.

9.4 Μοντέλο αναγνώρισης στάσης και μέτρηση παλμών

Πραγματοποιείται το παραπάνω μοντέλο δηλαδή αναγνωρίζει αν ένας άνθρωπος στέκεται όρθιος, κάθεται ή είναι ξαπλωμένος, με συνδυασμό του αισθητήρα καρδιακών παλμών MAX30102 και εμφανίζει ένα μήνυμα αποτελέσματος στο περιβάλλον της εφαρμογής IDE. Έγινε επεξεργασία και προσθήκη κώδικα στο μοντέλο που φτιάχτηκε παραπάνω, ώστε να μπορεί να ικανοποιεί τις ανάγκες.

Επιπλέον, έγινε προσπάθεια για σύνδεση του ESP με το Arduino, ώστε να μπορέσει να συνδεθεί στο δίκτυο, αλλά αυτό δεν ήταν εφικτό, διότι η θύρα TX του Arduino είναι μη διαθέσιμη λόγω της κάμερας. Καθώς η κάμερα χρησιμοποιεί τη θύρα TX για να μεταδώσει δεδομένα και, όταν ενεργοποιείται από το ESP η θύρα RX για να λάβει δεδομένα από το Arduino, εμφανίζονται σύμβολα και περίεργα γράμματα. Ακόμα έγινε προσπάθεια για την ενεργοποίηση του BlueTooth αλλά λόγο μη επαρκούς μνήμης RAM δεν είναι εφικτό, εξαιτίας του AI μοντέλου που τρέχει παράλληλα.



Σχήμα 9.3: Σχεδιάγραμμα κυκλώματος Arduino-Κάμερας-Max30102.

Η διαδικασία εκτέλεσης του κώδικα είναι η ίδια δηλαδή, ανοίγουμε το αρχείο, επιλέγουμε την θύρα που βρίσκεται το Arduino και πατάμε Upload. Για να μπορούμε να δούμε τα δεδομένα πρέπει να ανοίξουμε το παράθυρο του Serial Monitor. Πατάμε Menu -> Tools -> Serial Monitor. Ακόμα μπορεί να χρειαστεί να πατήσουμε δύο φορές το κουμπί Reset του Arduino

για να μπει σε Boot Mode, πριν πατήσουμε το Upload. Σε αυτήν την περίπτωση το Arduino θα βρίσκεται σε άλλη θύρα, άρα ρυθμίζουμε ξανά την θύρα εισόδου του Arduino. Πάλι από το Menu -> Tools -> Port και επιλέγουμε την θύρα.



Εικόνα 9.12: Εικόνα πρόβλεψης μοντέλου.

```

IR=943, BPM=96.15, Avg BPM=77
IR=945, BPM=96.15, Avg BPM=77
IR=957, BPM=96.15, Avg BPM=77
IR=951, BPM=96.15, Avg BPM=77
IR=948, BPM=96.15, Avg BPM=77
IR=947, BPM=96.15, Avg BPM=77
IR=949, BPM=96.15, Avg BPM=77
IR=947, BPM=96.15, Avg BPM=77
IR=950, BPM=96.15, Avg BPM=77
IR=952, BPM=96.15, Avg BPM=77
IR=948, BPM=96.15, Avg BPM=77
IR=946, BPM=96.15, Avg BPM=77
IR=953, BPM=96.15, Avg BPM=77
IR=953, BPM=96.15, Avg BPM=77
IR=941, BPM=96.15, Avg BPM=77
IR=940, BPM=96.15, Avg BPM=77
IR=966, BPM=96.15, Avg BPM=77
IR=947, BPM=96.15, Avg BPM=77
IR=950, BPM=96.15, Avg BPM=77
IR=941, BPM=96.15, Avg BPM=77
IR=952, BPM=96.15, Avg BPM=77
IR=937, BPM=96.15, Avg BPM=77
IR=949, BPM=96.15, Avg BPM=77
IR=960, BPM=96.15, Avg BPM=77
IR=942, BPM=96.15, Avg BPM=77
IR=949, BPM=96.15, Avg BPM=77
IR=945, BPM=96.15, Avg BPM=77

Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 3 ms., Classification: 232 ms., Anomaly: 0 ms.):
Object detection bounding boxes:
    sitting (0.878906) [ x: 24, y: 24, width: 8, height: 8 ]
Position: sitting, Heart Rate = 77

IR=950, BPM=96.15, Avg BPM=77
IR=940, BPM=96.15, Avg BPM=77
IR=954, BPM=96.15, Avg BPM=77
IR=945, BPM=96.15, Avg BPM=77
IR=949, BPM=96.15, Avg BPM=77
IR=956, BPM=96.15, Avg BPM=77
IR=948, BPM=96.15, Avg BPM=77
IR=946, BPM=96.15, Avg BPM=77
IR=942, BPM=96.15, Avg BPM=77
IR=954, BPM=96.15, Avg BPM=77
IR=950, BPM=96.15, Avg BPM=77
IR=953, BPM=96.15, Avg BPM=77
IR=952, BPM=96.15, Avg BPM=77
IR=955, BPM=96.15, Avg BPM=77
IR=950, BPM=96.15, Avg BPM=77
IR=939, BPM=96.15, Avg BPM=77
IR=948, BPM=96.15, Avg BPM=77
IR=953, BPM=96.15, Avg BPM=77
IR=946, BPM=96.15, Avg BPM=77
IR=937, BPM=96.15, Avg BPM=77
IR=944, RPM=96.15, Avg RPM=77
IR=955, BPM=96.15, Avg BPM=77
IR=938, BPM=96.15, Avg BPM=77
IR=951, BPM=96.15, Avg BPM=77
IR=953, BPM=96.15, Avg BPM=77

Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 3 ms., Classification: 232 ms., Anomaly: 0 ms.):
Object detection bounding boxes:
    sitting (0.816406) [ x: 24, y: 24, width: 8, height: 8 ]
Position: sitting, Heart Rate = 77

```

Εικόνα 9.13: Αποτελέσματα του μοντέλου.

Το μοντέλο έχει προγραμματιστεί να εκτελεί πεντακόσιες φορές την συνάρτηση για την μέτρηση του καρδιακού παλμού, να τραβά μια φωτογραφία η οποία αναγνωρίζει εάν κάθεσαι, βρίσκεσαι όρθιος ή ξαπλωμένος και να εμφανίζει ένα μήνυμα, για το πόσους παλμούς διαβάζει, το αποτέλεσμα της αναγνώρισής και αν οι παλμοί βρίσκονται σε κρίσιμες περιοχές όπως πάνω από εκατό ή κάτω από σαράντα παλμούς, εμφανίζει ένα μήνυμα προσοχής. Στο σχήμα 1.101 παρατηρείται ότι το μοντέλο

αναγνωρίζει ότι κάθεται και η διαφορά που έχουν οι δύο φωτογραφίες είναι το ποσοστό ακρίβειας δηλαδή στην πρώτη έχει αναγνωρίσει ότι κάθεται με ποσοστό 87,8% και στην δεύτερη με 81,6% ποσοστό επιτυχίας.

```
IR=2435, BPM=3.19, Avg BPM=0
IR=2453, BPM=3.19, Avg BPM=0
IR=2416, BPM=3.19, Avg BPM=0
IR=2471, BPM=3.19, Avg BPM=0
IR=2446, BPM=3.19, Avg BPM=0
IR=2411, BPM=3.19, Avg BPM=0
IR=2450, BPM=3.19, Avg BPM=0
IR=2498, BPM=3.19, Avg BPM=0
IR=2469, BPM=3.19, Avg BPM=0
IR=2430, BPM=3.19, Avg BPM=0
IR=2423, BPM=3.19, Avg BPM=0
IR=2497, BPM=3.19, Avg BPM=0
IR=2394, BPM=3.19, Avg BPM=0
IR=2448, BPM=3.19, Avg BPM=0
IR=2468, BPM=3.19, Avg BPM=0
IR=2489, BPM=3.19, Avg BPM=0
IR=2445, BPM=3.19, Avg BPM=0
IR=2455, BPM=3.19, Avg BPM=0
IR=2421, BPM=3.19, Avg BPM=0
IR=2475, BPM=3.19, Avg BPM=0
IR=2427, BPM=3.19, Avg BPM=0
IR=2469, BPM=3.19, Avg BPM=0
IR=2466, BPM=3.19, Avg BPM=0
IR=2433, BPM=3.19, Avg BPM=0
IR=2457, BPM=3.19, Avg BPM=0

Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 3 ms., Classification: 233 ms., Anomaly: 0 ms.):
Object detection bounding boxes:
    lying Down (0.656250) [ x: 24, y: 24, width: 8, height: 8 ]
Attention!! Position: lying Down, Low Heart Rate = 0
```

Εικόνα 9.14: Αποτέλεσμα μοντέλου αναγνώρισης

Στο παραπάνω σχήμα εμφανίζεται το αποτέλεσμα του μοντέλου αναγνώρισης αλλά και το μήνυμα προσοχής. Επίσης υπάρχει μια απόκλιση στην μέτρηση των παλμών $BPM = 3,19$ αλλά δεν επηρεάζει το τελικό αποτέλεσμα καθώς χρησιμοποιείτε ο μέσος όρος των παλμών. Ο μέσος όρος προκίπτει από τις τελευτές μετρήσεις των παλμών της καρδίας, ανάλογα αν η καρδιά χτυπά γρύγορα ή αργά.

10 Συμπεράσματα

Στην παρούσα διπλωματική εργασία αναπτύχθηκε ένα ολοκληρωμένο σύστημα παρακολούθησης ασθενών, το οποίο συνδυάζει αισθητήρες καταγραφής ζωτικών λειτουργιών και υπολογιστική όραση. Το σύστημα αποδείχθηκε λειτουργικό και αποδοτικό, προσφέροντας τη δυνατότητα απομακρυσμένης παρακολούθησης μέσω σύγχρονων τεχνολογιών επικοινωνίας και μηχανικής μάθησης.

Τα αποτελέσματα της μελέτης έδειξαν ότι το προτεινόμενο σύστημα μπορεί να ενισχύσει την έγκαιρη ανίχνευση κρίσιμων καταστάσεων, συμβάλλοντας στην πρόληψη σοβαρών επιπλοκών σε ασθενείς που χρήζουν συνεχούς ιατρικής παρακολούθησης. Η ακρίβεια των αισθητήρων και των αλγορίθμων αναγνώρισης στάσης του σώματος κρίθηκε ικανοποιητική, αν και παρατηρήθηκαν ορισμένες προκλήσεις, όπως η ανάγκη για βελτίωση της απόκρισης του συστήματος σε πραγματικές συνθήκες.

Μία από τις σημαντικές προκλήσεις που προέκυψαν ήταν η αξιοπιστία των αισθητήρων και η ανάγκη βαθμονόμησή τους για ακριβέστερες μετρήσεις. Επιπλέον, η υπολογιστική όραση παρουσίασε περιορισμούς σε περιβάλλοντα με μεταβαλλόμενο φωτισμό, γεγονός που επηρεάζει την αναγνώριση της στάσης του σώματος. Παρόλα αυτά, οι πρώτες δοκιμές έδειξαν θετικά αποτελέσματα, αποδεικνύοντας τη δυνατότητα ενσωμάτωσης του συστήματος σε πραγματικές εφαρμογές.

Για μελλοντική επέκταση της έρευνας, προτείνεται η ενσωμάτωση επιπλέον βιομετρικών αισθητήρων και η δοκιμή του συστήματος σε μεγαλύτερο δείγμα χρηστών. Επιπλέον, η ανάπτυξη ενός βελτιωμένου γραφικού περιβάλλοντος χρήστη θα μπορούσε να καταστήσει την πλατφόρμα πιο προσβάσιμη σε ασθενείς και επαγγελματίες υγείας.

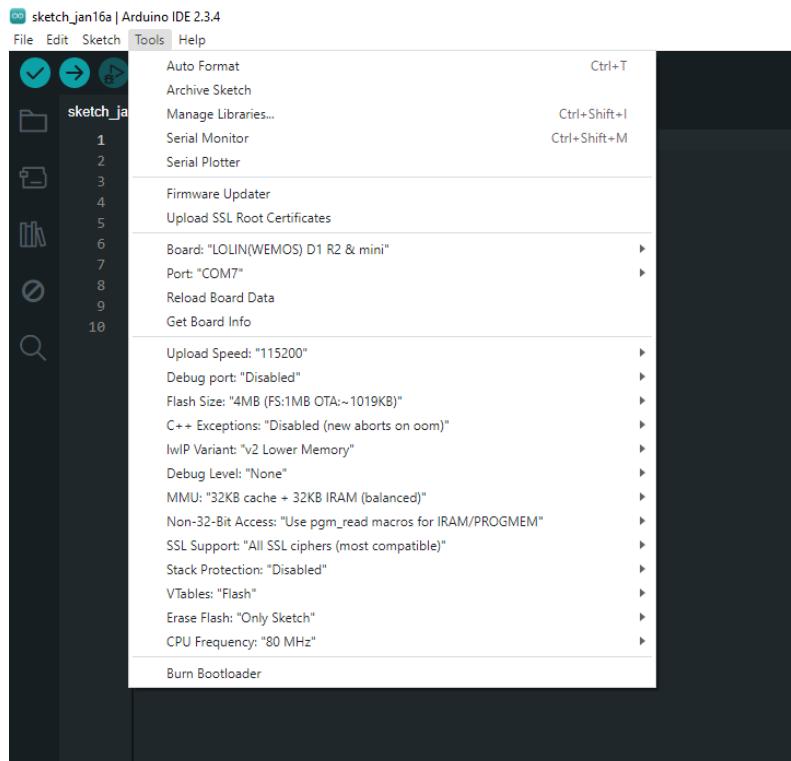
Συνολικά, η εργασία αυτή αποτελεί ένα σημαντικό βήμα προς τη βελτίωση των απομακρυσμένων ιατρικών εφαρμογών, αποδεικνύοντας ότι η τεχνολογία μπορεί να συμβάλει στην αναβάθμιση της υγειονομικής περίθαλψης και στη βελτίωση της ποιότητας ζωής των ασθενών.

Παράρτημα A: Οδηγίες χρήσης

A1 Διαδικασία εγκατάστασης Βιβλιοθηκών

Διαδικασία κατεβάσματος των βιβλιοθηκών έγινε ως εξής:

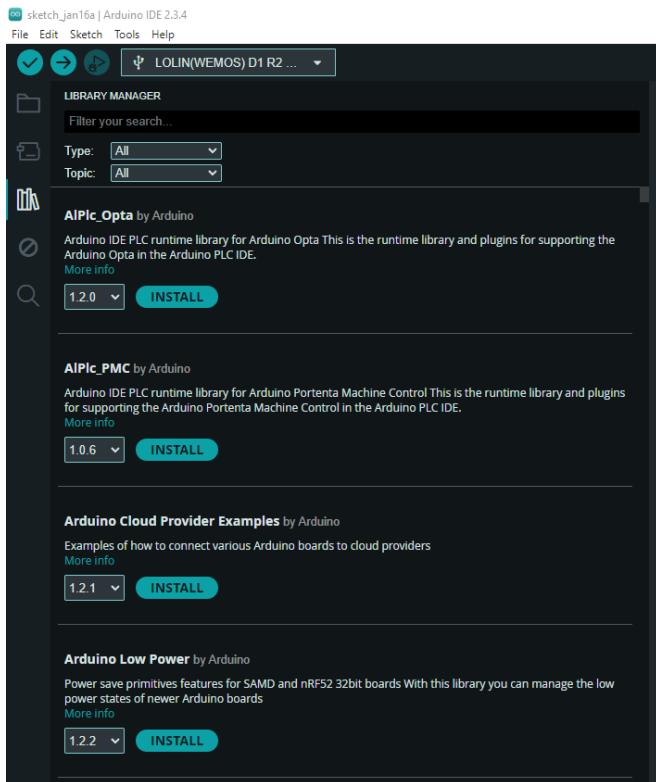
- Ανοιγμα του προγράμματος Arduino IDE.
- Από το μενού Tools γίνετε η επιλογή Manage Libraries.



Εικόνα A.1: Μενού Tools.

Από το νέο παράθυρο Library Manager, γίνετε αναζήτηση της βιβλιοθήκης.

Επιλέγοντας την έκδοση της βιβλιοθήκης και πατώντας το «Install» αρχίζει να κατεβαίνει.



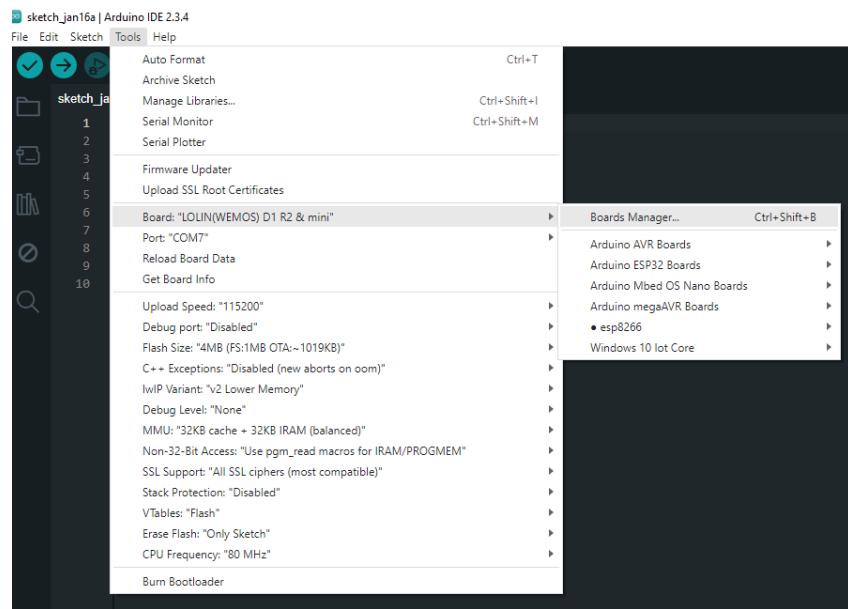
Εικόνα A.2: Library Manager.

Αν κάποια βιβλιοθήκη δεν υπάρχει μέσω του προγράμματος τότε υπάρχει δυνατότητα για κατέβασμα των βιβλιοθηκών από το διαδίκτυο.

A2 Οδηγοί για Πλακέτες

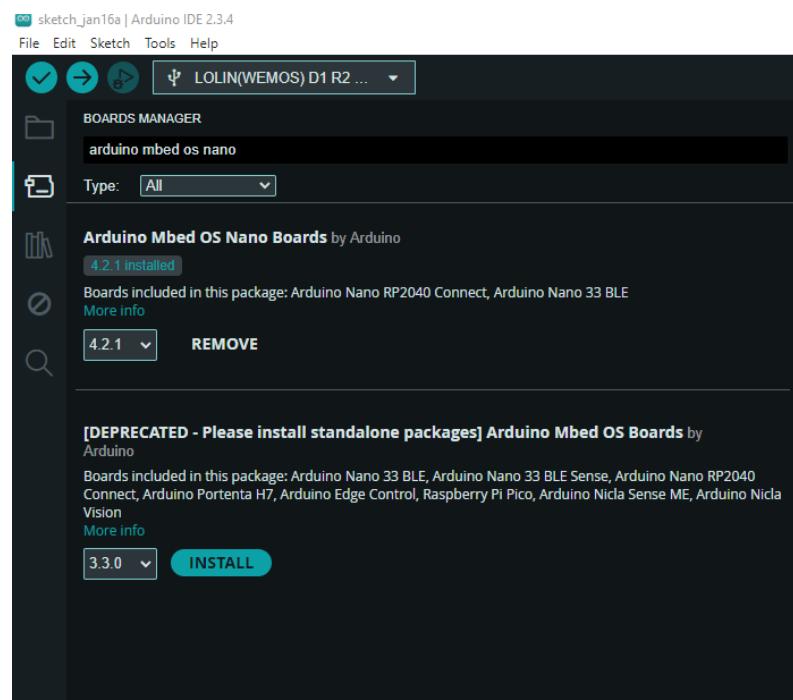
Η κάθε πλακέτα χρειάζεται κάποιο πρόγραμμα οδηγό (drivers) για μπορεί να αναγνωριστεί από τον υπολογιστή. Ακολουθούν τα βήματα για το κατέβασμα των drives.

Αφού είμαστε στο περιβάλλον του Arduino IDE, επιλέγουμε από το μενού Tools το Board και έπειτα το Board Manager.



Εικόνα A.3: Tools και Boards Μενού.

Ανοίγει το παράθυρο και αναζητούμε το Arduino Mbed OS Nano Boards. Στη συνέχει πατάμε (install) και περιμένουμε να εγκατασταθεί. Αυτό αφορά την πλακέτα Arduino nano 33 Ble Sence.



Εικόνα A.3: Παράθυρο Boards Manager.

Για την πλακέτα ESP-8266 ακολουθούνται τα παρακάτω βήματα.

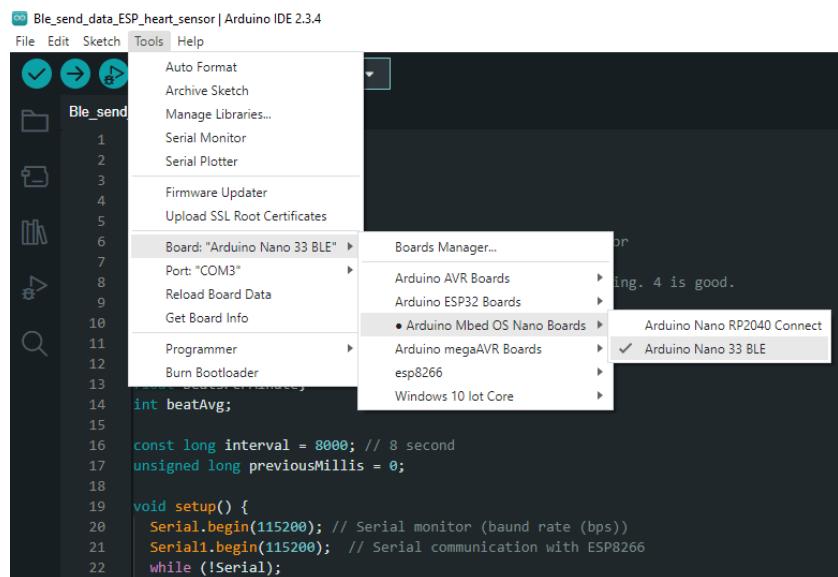
- Αντιγραφή του συνδέσμου:
- https://arduino.esp8266.com/stable/package_esp8266core_index.json
- Έπειτα στην εφαρμογή Arduino, ανοίγουμε από το μενού File -> Preferences.
- Επικολλούμε τον σύνδεσμο στο <Additional boards manager URLs>
- Πατάμε OK

Το πρόγραμμα θα ξεκινήσει να κατεβάζει το πακέτο esp8266.

A3 Διαδικασία εκτέλεσης

Η εκτέλεση του προγράμματος για το Arduino είναι:

- Ανοιγμα του αρχείου από την θέση που το έχουμε αποθήκευση.
- Επιλέγουμε την πλακέτα από το μενού Tools -> Board -> Arduino Mbed OS Nano Boards -> Arduino Nano 33 Ble.

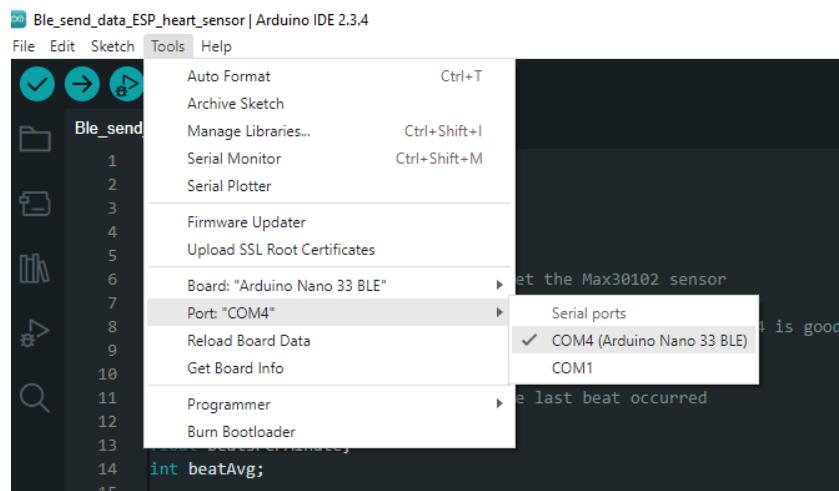


Εικόνα A.4: Επιλογή πλακέτας.

Ακολουθεί η επιλογή της θύρας που είναι συνδεδεμένο το Arduino στον υπολογιστή μέσω καλωδίου.

- Επιλέγουμε μενού Tools -> Port -> Com 4 (Arduino Nano 33 Ble) .

Στην συγκεκριμένη περίπτωση είναι στην θύρα 4, αλλά μπορεί να βρίσκεται και σε άλλη θύρα ανάλογα με την διαθεσιμότητα των θυρών.

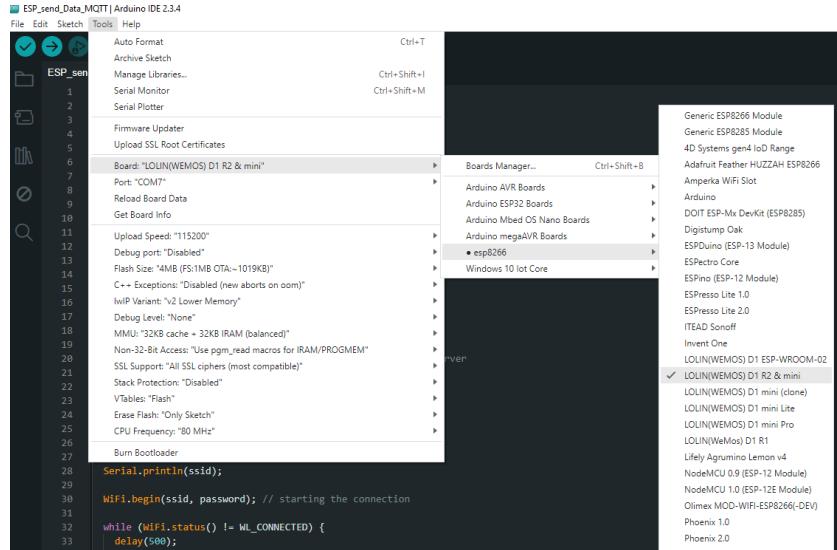


Εικόνα A.5: Επιλογή θύρας.

Στην συνέχεια πατάμε Verify. Με αυτήν την λειτουργία το λογισμικό Arduino IDE αναγνωρίζει αν υπάρχει κάποιο λάθος στην ορθότητα του κώδικα. Τέλος κάνουμε κλικ το κουμπί upload για να φορτώσουμε τον κώδικα στο Arduino.

Για το Eps8266 η διαδικασία είναι η εξής:

- Ανοίγουμε το αρχείο, επιλέγουμε από μενού
- Tools -> Boards -> esp8266 -> LOLIN(WEMOS) D1 R2 & mini



Εικόνα A.6: Επιλογή πλακέτας Esp8266.

Για την επιλογή της θύρας αντίστοιχα βήματα είναι με το Arduino.

Στο Esp8266 πρέπει να γίνουν οι παρακάτω απαραίτητες ρύθμισης για την σωστή λειτουργία.

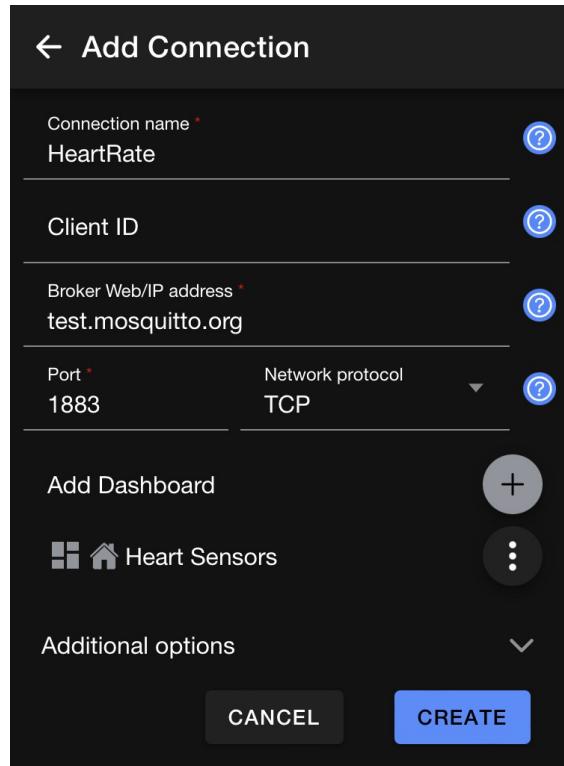
- Μενού Tools -> Upload speed -> 115200
- Μενού Tools -> Flash Size -> 4MB (FS:1MB OTA ~1019KB)
- Επειτα πατάμε Verify και Upload για να ελεγχθεί και να φορτωθεί ο κώδικας στο Esp8266.

A4 Προγραμματισμός εφαρμογής IoTMQTTPanel

Για την απεικόνιση των δεδομένων χρειάζεται μια εφαρμογή και συγκεκριμένα χρησιμοποιήθηκε η IoTMQTTPanel.

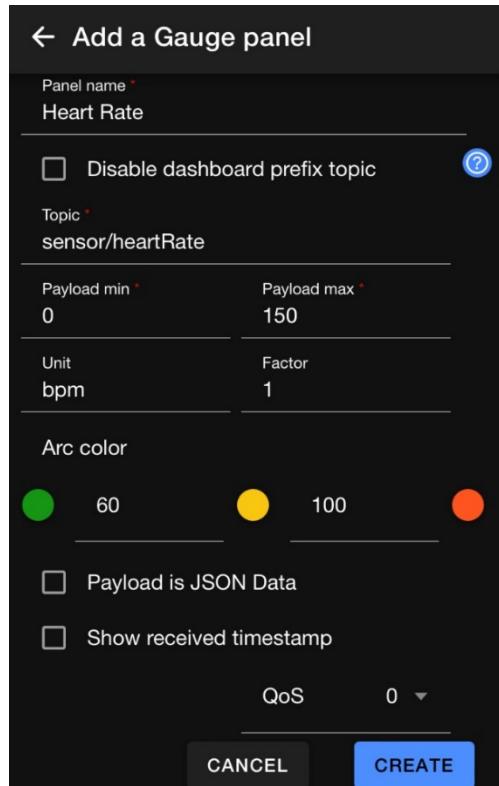
Εγκαθιστούμε την εφαρμογή μέσω του App Store, αυτή η εφαρμογή απευθύνεται για λογισμικό iOS. Δημιουργούμε ένα μια σύνδεση πατώντας Add Connection, συμπληρώνουμε όνομα της σύνδεσης π.χ. (HeartRate). Στο πεδίο Broker Web/IP address, βάζουμε είτε το test.mosquitto.org είτε το IP 142.93.191.12, το οποίο μπορεί να βρεθεί είτε κάνοντας ping στο test.mosquitto.org είτε από την ιστοσελίδα mosquito [73]. To Port είναι

προκαθορισμένο ως 1883. Στη συνέχεια, επιλέγουμε Add Dashboard, δίνουμε ένα όνομα και πατάμε Create.



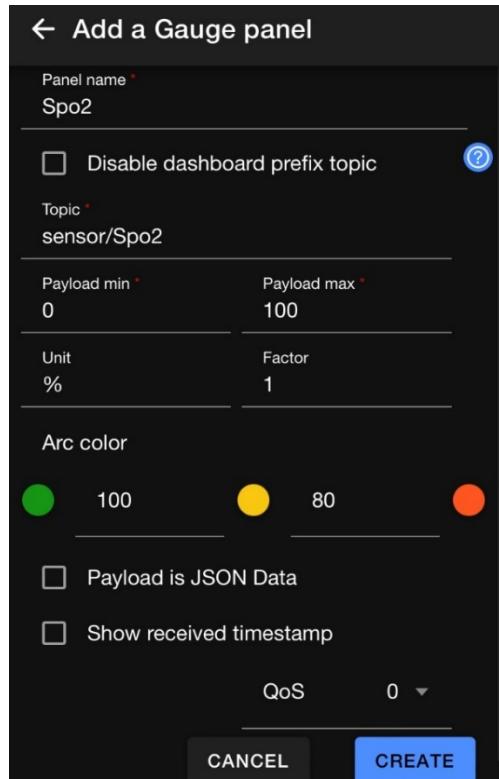
Εικόνα A.7: Δημιουργία σύνδεσης στο IoTMQTTPanel.

Αφού έχει δημιουργηθεί η σύνδεση, πατάμε το κουμπί ADD PANEL, διαλέγουμε πως θέλουμε να απεικονίζονται τα δεδομένα και επιλέγουμε Gauge. Για την απεικόνιση των καρδιακών παλμών οι ρυθμίσεις του Gauge είναι η εξής, ονομάζουμε το πάνελ, στο Topic συμπληρώνουμε όπως το έχουμε ονομάσει στον κώδικα sensor/heartRate. Στην μονάδα μέτρησης Unit θέλουμε τους κτύπους της καρδιάς ανα λεπτό δηλαδή bpm. Το εύρος τιμών είναι από 0 έως 150 παλμούς, στο χρώμα βάζουμε 60 στο πράσινο και 100 στο κόκκινο και πατάμε δημιουργία.



Εικόνα A.8: Δημιουργία Gauge πάνελ για καρδιακού παλμούς.

Ο προγραμματισμός του Gauge για την μέτρηση του ποσοστού οξυγόνου στο αίμα είναι παρόμοιος. Στην εικόνα A.9 φαίνονται οι παράμετροι για την σωστοί λειτουργία.



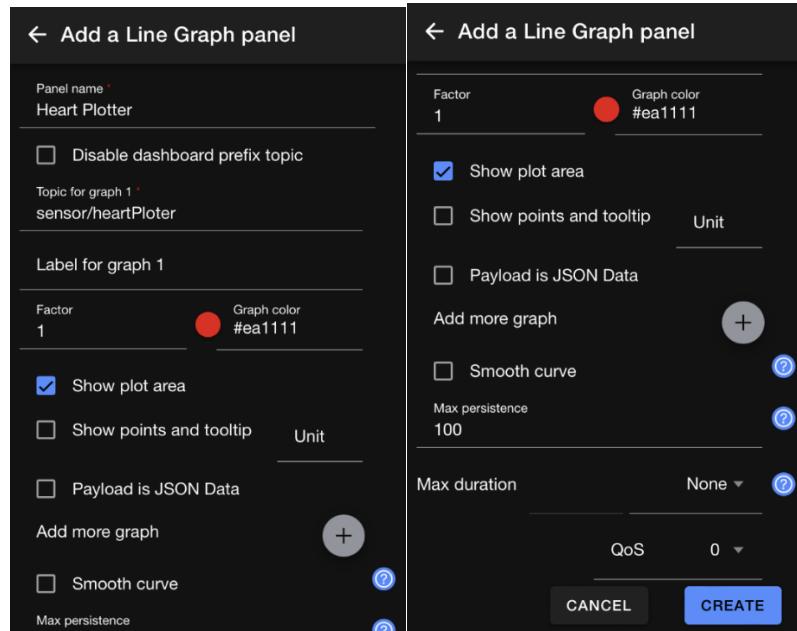
Εικόνα A.9: Ρυθμίσεις Gauge πάνελ για μέτρηση οξυγόνου.

A5 Ρύθμιση εφαρμογής για την απεικόνιση του ηλεκτροκαρδιογραφήματος.

Για την απεικόνιση του ηλεκτροκαρδιογραφήματος στην εφαρμογή ακολουθήθηκαν τα παρακάτω βήματα.

- Δημιουργία γραφήματος (Add Graph)
- Όνομα - > Heart Plotter
- Topic - > sensor/heartPloter
- Max persistence - > 100

To Persistence αναφέρεται σε έναν αριθμό που μπορούν να αποθηκευτούν τα δεδομένα όταν η εφαρμογή εκτελείτε στο παρασκήνιο. Στην συγκεκριμένη περίπτωση η διαδικασία γίνετε σε ζωντανή μετάδοση και δεν επηρεάζει το γράφημα.

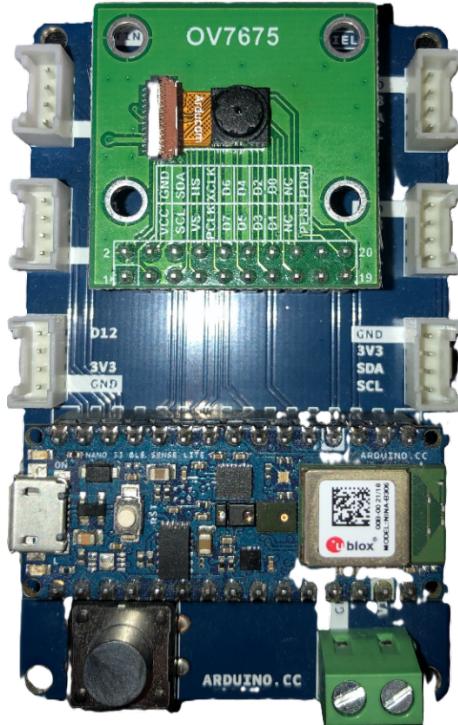


Εικόνα A.10: Ρυθμίσεις γραφήματος.

Η ρύθμιση QoS και στις τρείς περίπτωσης είναι 0 διότι χρησιμοποιείτε ζωντανή μεταδόσω των δεδομένων οπότε το πρωτόκολλο QoS 0 επιτρέπει μεγαλύτερες ταχύτητες μετάδοσης, αλλά υπάρχει απώλεια δεδομένων. Επίσης μπορούν να χρησιμοποιηθούν και τα υπόλοιπα πρωτόκολλα QoS (1,2).

A6 Μοντέλο Υπολογιστικής Όρασης

Η δημιουργία του μοντέλου αναγνώρισης πραγματοποιήθηκε με την εφαρμογή Edge Impulse η οποία είναι ανοιχτή για το κοινό μέχρι κάποιους υπολογιστικούς πόρους. Όπως φαίνεται και παρακάτω, για να μπορεί το Arduino να καταγράφει και να αναγνωρίζει τις φωτογραφίες έχει προστεθεί μια κάμερα OV7675.



Εικόνα Α.11: Κυκλώματος αναγνώρισης φωτογραφιών.

Διαδικασία δημιουργίας μοντέλου:

- Δημιουργία λογαριασμού Edge Impulse
- Δημιουργία ενός Πρότζεκτ

Αρχικά ρυθμίζουμε τα χαρακτηριστικά της συσκευής δηλαδή το Arduino nano 33 ble sense.

Κάνουμε κλικ δίπλα από το εικονίδιο του λογαριασμού «Target» Εμφανίζει ένα παράθυρο και συμπληρώνουμε τα χαρακτηριστικά.

- Target device - > Cortex-M4F 80MHz
- Processor family - > Cortex-M
- Clock rate - > 80
- Ram - > 156
- Rom - > 1
- Latency - > 100
- Πατάμε Save

Configure your target device and application budget

Target device
Define your target device requirements to inform model optimizations and performance calculations. No device yet? Use the default settings which you can change at any time.

Target device	Cortex-M4F 80MHz
Processor family	Cortex-M
Clock rate ⓘ	80 Mhz Max
Custom device name (optional) ⓘ	

Application budget
Specify the available RAM and ROM for the model's operation, along with the maximum allowed latency for your specific application. Not sure yet? Start with the defaults and modify them later on.

RAM	156 KB Max
ROM	1 MB Max
Latency ⓘ	100 ms Max

Εικόνα A.12: Ρυθμίσεις Arduino Nano 33 Ble Sence για την εφαρμογή Edge Impulse.

Για να μπορούμε να συνδέσουμε το Arduino Nano 33 Ble Sence με την εφαρμογή πρέπει να γίνουν κάποια βήματα:

- Devices -> Connect a new device -> Connect your device or development board, κάνουμε κλικ στην αναζήτηση και βρίσκουμε το Arduino, έπειτα πατάμε View installation docs. Εκεί μας δίνει αναλυτικά τις οδηγίες για την σύνδεση του Arduino στην εφαρμογή Edge impulse.
- Εγκατάσταση Python 3.
- Εγκατάσταση Node.js v20 ή μεγαλύτερη έκδοση.
- Αν δεν υπάρχει η εντολή (npm) την εγκαθιστούμε μέσω της γραμμής εντολών.
- Εκτέλεση στην γραμμή εντολών «npm install -g edge-impulse-cli»
- Εγκαθιστούμε από την ιστοσελίδα [74] το cli, ειδικότερα το αρχείο Windows exe 64 bit.

- Κάνουμε αντιγραφή το αρχείο Arduino-cli που εγκαταστήσαμε
- Εγκαθιστούμε το υλικολογισμικό από την ιστοσελίδα με τις οδηγίες [75], κάνοντας κλικ στο «Download the latest Edge Impulse firmware».
- Κάνουμε επικόλληση στον φάκελο Arduino-nano-33-ble-sense
- Συνδέουμε το Arduino στον υπολογιστή και πατάμε δύο φορές το κουμπί Reset για να μπει σε Boot mode.
- Ανοίγουμε το flash_windows από τον φάκελο Arduino-nano-33-ble-sense.
- Αφού φορτωθεί στο Arduino τότε ανοίγουμε την γραμμή εντολών και πληκτρολογούμε «edge-impulse-daemon».
- Θα μας βάλει να συμπληρώσουμε το όνομα χρήστη και κωδικό από τον λογαριασμό μας στο Edge impulse, επιλέγουμε την θύρα που έχουμε σύνδεση το Arduino και το ονοματίζουμε.

Παρατηρούμε από το μενού αριστερά στο Edge impulse Device ότι παρουσιάζετε το Arduino.

The screenshot shows the 'Your devices' section of the Edge Impulse web interface. At the top, there's a header with 'Your devices' and a 'Connect a new device' button. Below the header, a message states: 'These are devices that are connected to the Edge Impulse remote management API, or have posted data to the Ingestion SDK.' A table lists the connected device:

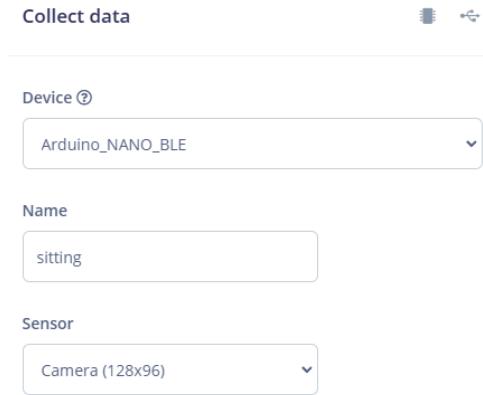
DEVICE	ID	TYPE
Arduino_NANO_BLE Connected to data acquisition (Built-in microphone, Inertial (Accelerometer, Gyroscope, Magnetometer), GPS, WiFi, BLE)	A0:63:12:AD:04:1B	ARDUINO_NANO33BLE

Εικόνα A.13: Συνδεδεμένη συσκευή στο Edge impulse

Με αυτήν την σύνδεση μπορούμε να λαμβάνουμε δεδομένα χρησιμοποιώντας το Arduino. Στη συνέχεια για την υλοποίηση του μοντέλου ακολουθήθηκε:

- Μενού -> Data acquisition

- Ακολουθούμε τις παρακάτω ρυθμίσεις για να τραβήξουμε φωτογραφίες από το Arduino.



Εικόνα A.14: Ρυθμίσεις συσκευής για συλλογή δεδομένων

- Το όνομα αλλάζει σε σχέση με την φωτογραφία που θέλουμε να τραβήξουμε, δηλαδή αν είναι ξαπλωμένος στην φωτογραφία τότε γράφουμε lying Down.
- Πατάμε το Start sampling για να αποθηκευτεί η φωτογραφία στην βάση δεδομένων.

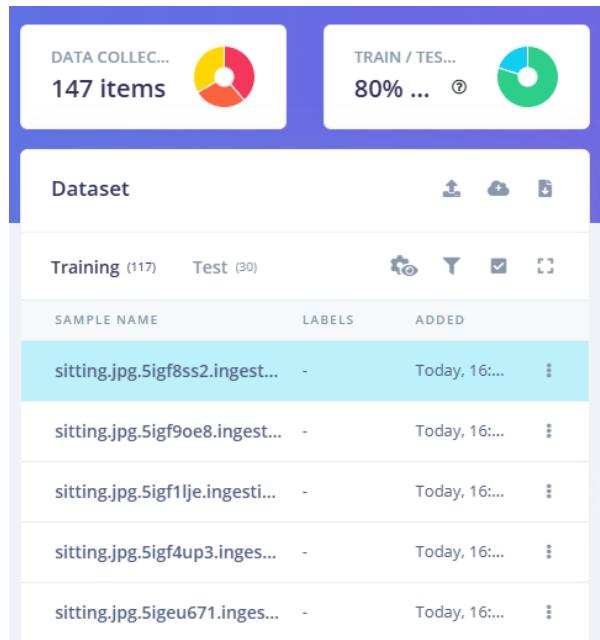
Αυτός είναι ο ένας τρόπος που χρησιμοποιήθηκε για την συλλογή δεδομένων. Ο δεύτερος τρόπος είναι:

- Τραβήχτηκαν φωτογραφίες μέσου του κινητού και το Edge impulse έχει την επιλογή να ανεβάσουμε τις φωτογραφίες στην βάση.
- Πατώντας το εικονίδιο (βελάκι πάνω) που βρίσκετε δίπλα από το Dataset



Εικόνα A.15: Ανέβασμα δεδομένων.

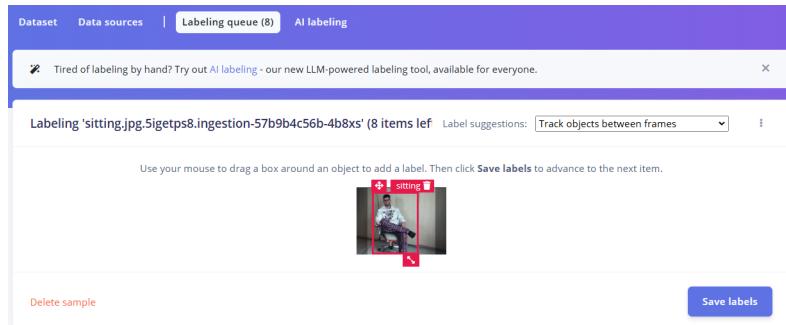
- Επιλέγουμε τις φωτογραφίες είτε μεμονωμένα είτε επιλέγοντας έναν ολόκληρο φάκελο.
- Χρησιμοποιήθηκε η αυτόματη διαχώριση των δεδομένων σε δεδομένα εκπαίδευσης και δοκιμής.
- Πατάμε Upload data.



Εικόνα A.16: Απεικόνιση δεδομένων στην βάση.

Πρέπει στα δεδομένα αυτά να δοθούν ετικέτες και να χαράξουμε τα όρια του αντικειμένου.

- Πατάμε Labeling queue.
- Εδώ εμφανίζονται όλες οι φωτογραφίες που δεν έχουν ετικέτα
- Χαράζουμε τα όρια του αντικειμένου και δίνουμε όνομα ετικέτας, όπως φαίνεται και παρακάτω.

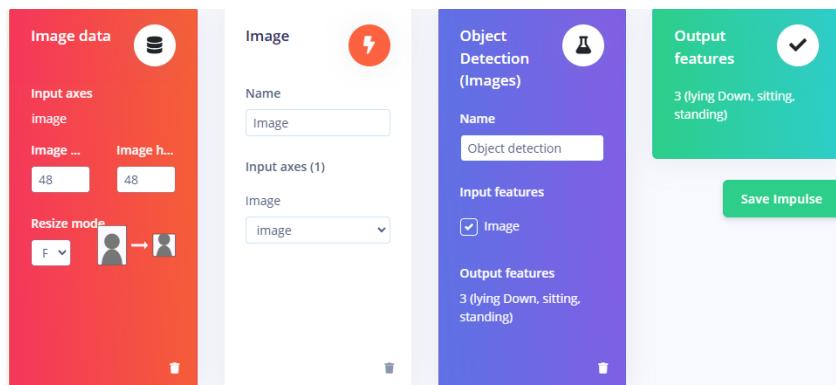


Εικόνα Α.17: Χάραξη και ονοματολογία ετικέτας σε φωτογραφίες.

Επαναλαμβάνουμε την διαδικασία για όλες τις φωτογραφίες. Συνολικά χρησιμοποιήθηκαν 270 φωτογραφίες. Το 80% αυτών χρησιμοποιήθηκε για την εκπαίδευση του μοντέλου, ενώ το υπόλοιπο 20% για τον έλεγχο (validation/testing) της απόδοσής του.

- Από το μενού αριστερά επιλέγουμε Impulse design - > Create impulse.

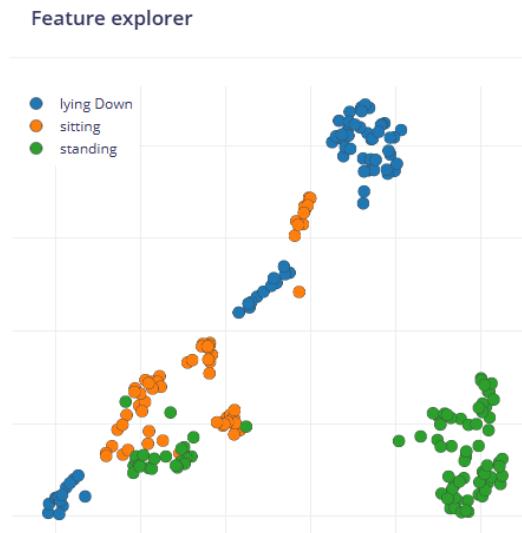
Οι ρυθμίσεις φαίνονται στο παρακάτω σχήμα.



Εικόνα Α.18: Ρυθμίσεις δημιουργίας impulse.

Μπορούμε να επιλέξουμε διάσταση εικόνας 96 X 96, αλλά χρειάζεται περισσότερους υπολογιστικούς πόρους. Έπειτα πατάμε Save και επιλέγουμε Image από το μενού αριστερά. Σε αυτό το στάδιο δημιουργούνται κάποια χαρακτηριστικά που

βοηθούν στην αναγνώριση. Από τις παραμέτρους διαλέγουμε Color depth - > Grayscale, καθώς καταναλώνει λιγότερους πόρους από το RGB και πατάμε Save parameters. Θα μας προωθήσει στην δημιουργία χαρακτηριστικών και κάνουμε κλικ στο Generate features.



Εικόνα A.19: Γράφημα χαρακτηριστικών.

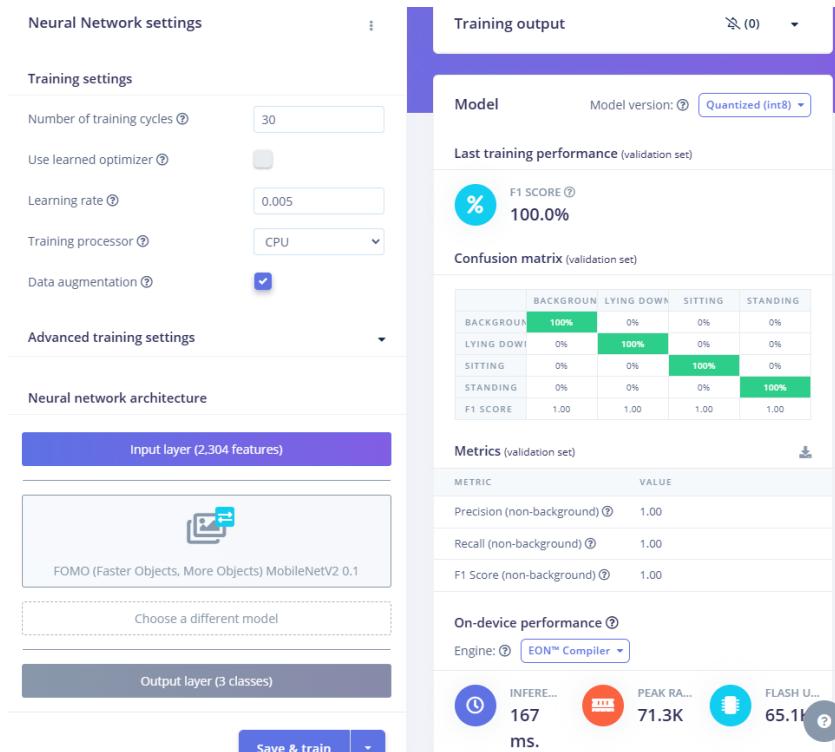
Παρατηρούμε ότι δεν είναι τόσο διακριτές οι κλάσεις (lying Down, sitting, standing). Για να ήταν τέλειο το γράφημα θα έπρεπε οι τρείς κλάσεις να ήταν η μία απέναντι από την άλλη ξεχωριστά και όχι κάποια δεδομένα να ταυτίζονται και να εφάπτονται.

Επίσης πατάμε Object detection και κάνουμε τις απαραίτητες ρυθμίσεις.

- Number of training cycles - > 30, αφορά τον αριθμό ενός πλήρους κύκλου εκπαίδευσης.
- Learning rate - > 0.005, αφορά την ταχύτητα με την οποία μαθαίνει το νευρωνικό δίκτυο. Αν το δίκτυο προσαρμόζεται γρήγορα, τότε πρέπει να μειωθεί ο ρυθμός μάθησης.

- Choose a different model -> Fomo (Faster Objects, More Objects) MobileNetV2 0.1.
- Πατάμε Save & train.

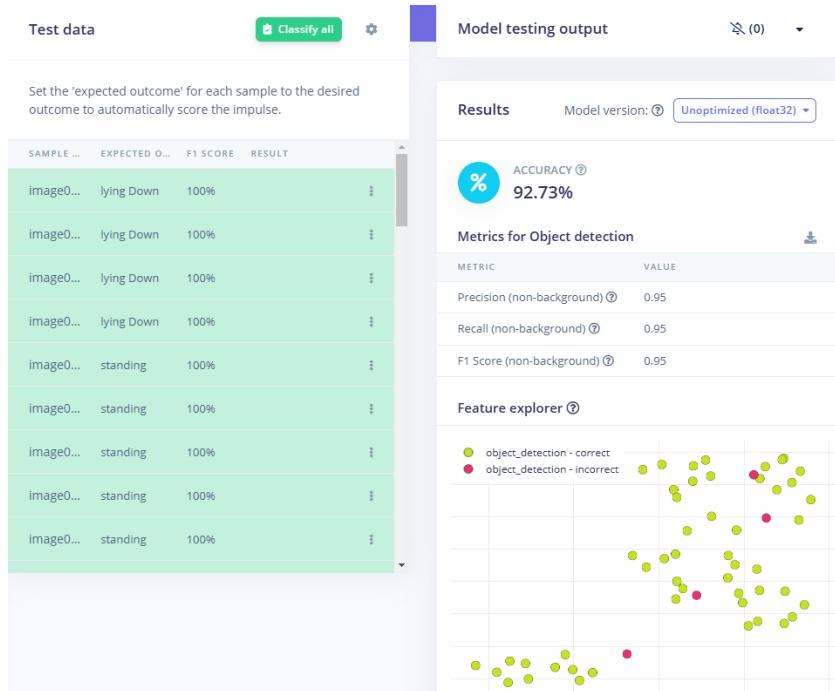
Το μοντέλο FOMO (Faster Objects, More Objects) είναι ένα ελαφρύ και αποδοτικό μοντέλο ανίχνευσης αντικειμένων, σχεδιασμένο ειδικά για συσκευές με περιορισμένους πόρους, όπως μικροελεγκτές και edge συστήματα.



Εικόνα A.20: Ρυθμίσεις και αποτελέσματα Object detection.

Φαίνεται ότι υπάρχει 100% F1 Score που αντιπροσωπεύει έναν συνδυασμό ακρίβειας και ανάκλησης.

Αφού φτιάχτηκε το μοντέλο, θα το δοκιμάσουμε με τα δεδομένα που έχουμε στην βάση και τα έχουμε κατατάσσει στα δεδομένα δοκιμής. Κάνοντας κλικ Model testing -> Classify all.



Εικόνα Α.21: Δοκιμή μοντέλου.

Διακρίνετε ότι τέσσερα από τα πενήντα πέντε δεδομένα αναγνώρισε λανθασμένα. Το ποσοστό ακρίβειας του μοντέλου είναι 92,73 %.

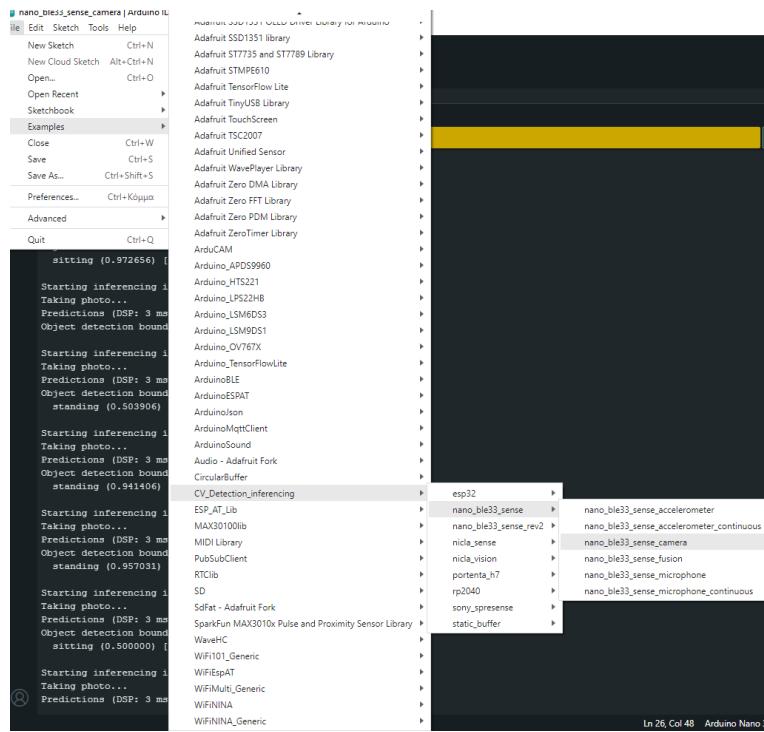
Για να κατεβάσουμε κώδικα του Arduino:

- Deployment -> Αναζητούμε Arduino library -> Πατάμε Build

Θα μας κατεβάσει ένα αρχείο σε zip.

Για να μπορούμε να το τρέξουμε τον κώδικα:

- Ανοίγουμε την εφαρμογή Arduino Ide
- Επιλέγουμε από το μενού Sketch -> Include Library -> Add ZIP Library
- Διαλέγουμε το αρχείο Zip που κατεβάσαμε.
- File -> Examples -> Όπως το έχουμε ονομάσει π.χ. CV_Detection -> nano_ble_sense -> nano_ble_sense_camera.



Εικόνα A.22: Φορτώσει μοντέλου αναγνωρίσεις στην εφαρμογή

Arduino Ide.

Παράρτημα Β: Κώδικας

B1 Κώδικας Arduino κυκλώματος μέτρησης καρδιακών παλμών

Ο κώδικας για την ορθή λειτουργία του Arduino nano 33 Ble Sence αποδίδεται παρακάτω με τα απαραίτητα σχόλια στα Αγγλικά για την κατανόηση του. Κάποιο κομμάτι του κώδικα είναι αυτούσιο από το παράδειγμα 5 της βιβλιοθήκης «SparkFun Max3010x Pulse and Sensor library» [76].

```
//libraries
```

```
#include <Wire.h>
```

```
#include "MAX30105.h"
```

```
#include "heartRate.h"
```

```

MAX30105 particleSensor; // Class to set the Max30102 sensor

const byte RATE_SIZE = 4; // Increase this for more averaging.
4 is good.

byte rates[RATE_SIZE]; // Array of heart rates

byte rateSpot = 0;

long lastBeat = 0; // Time at which the last beat occurred

float beatsPerMinute;

int beatAvg;

const long interval = 8000; // 8 second

unsigned long previousMillis = 0;

void setup() {

    Serial.begin(115200); // Serial monitor (baund rate (bps))

    Serial1.begin(115200); // Serial communication with ESP8266

    while (!Serial);

    Serial.println("Initializing...");

    // Initialize sensor

    if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) {

        Serial.println("MAX30105 was not found. Please check
wiring/power.");

```

```

        while (1);

    }

    Serial.println("Place your index finger on the sensor with steady
pressure.");

particleSensor.setup();

particleSensor.setPulseAmplitudeRed(0x0A); // Turn Red LED
to low to indicate sensor is running

particleSensor.setPulseAmplitudeGreen(0); // Turn off Green
LED

}

void loop() {

    long irValue = particleSensor.getIR(); // reading the value of
InfraRed LED

    if (checkForBeat(irValue) == true) {

        long delta = millis() - lastBeat; // Calculate the difference
between the present heartbeat and the last heartbeat

        lastBeat = millis();

        beatsPerMinute = 60 / (delta / 1000.0); // Calculate the beats
per minute
    }
}

```

```

if (beatsPerMinute < 255 && beatsPerMinute > 20) {

    rates[rateSpot++] = (byte)beatsPerMinute; // Array which
    store the measurements

    rateSpot %= RATE_SIZE; // Determines in which position of
    the array the new measurement will be stored and increase by 1


// Calculation of Average BPM

beatAvg = 0;

for (byte x = 0; x < RATE_SIZE; x++) {

    beatAvg += rates[x];

}

beatAvg /= RATE_SIZE;

}

// Print Values

Serial.print("IR=");

Serial.print(irValue);

Serial.print(", BPM=");

Serial.print(beatsPerMinute);

Serial.print(", Avg BPM=");

Serial.print(beatAvg);




if (irValue < 50000) {

    Serial.print(" No finger?");

}


```

```

Serial.println();

// Sending data every 8 seconds to ESP8266

unsigned long currentMillis = millis();

if (currentMillis - previousMillis >= interval) {

    previousMillis = currentMillis;

    String data = String(beatAvg);

    Serial.println("Sending data: " + data);

}

// Send data via Serial to ESP8266

Serial1.println(data);

}

```

B1.1 Κώδικας ESP-8266 κυκλώματος μέτρησης παλμών

Επειτα, ακολουθεί ο κώδικας για το **ESP-8266**, ώστε να μπορεί να συνδεθεί με διαδίκτυο, να στέλνει δεδομένα μέσω MQTT σε έναν MQTT broker και σειριακά να συνδέεται με Arduino. Η βιβλιοθήκη που χρησιμοποιήθηκε είναι «PubSubClient» και η «ESP8266WiFi». Κάποια στοιχεία του παραδείγματος «mqtt_esp8266» χρησιμοποιήθηκαν για την υλοποίηση του κώδικα [77].

```

// libraries

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

```

```

// Network imformation

const char* ssid = "Odyn";

const char* password = "ody123456";


// MQTT broker information

const char* mqtt_server = "test.mosquitto.org"; // IP
142.93.191.12

const char* HeartRate_topic = "sensor/heartRate"; //topic

const int mqtt_port = 1883; // Port


WiFiClient espClient; // Class TCP communication to MQTT
server

PubSubClient client(espClient); // class for Publish and subscribe
using TCP communication


void setup() {

    Serial.begin(115200);

    setup_wifi(); // Call function

    client.setServer(mqtt_server, mqtt_port); // Set for MQTT
    Server

}

//Connect to wifi

void setup_wifi() {

    delay(200);

```

```

Serial.println();
Serial.print("Connecting to ");
Serial.println(ssid);

WiFi.begin(ssid, password); // starting the connection

while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}

Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}

// Function for reconnecting

void reconnect() {
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        if (client.connect("ESP8266Client")) {
            Serial.println("connected");
        } else {
            Serial.print("failed, rc=");
        }
    }
}

```

```

Serial.print(client.state());

Serial.println(" try again in 5 seconds");

delay(5000);

}

}

}

void loop() {

if (!client.connected()) {

reconnect();

}

client.loop();

}

if (Serial.available() > 0) {

String data = Serial.readStringUntil('\n'); // Receive data from
arduino

Serial.print("Received data: ");

Serial.println(data);

}

if (client.publish(HeartRate_topic, data.c_str())) { // sending
data via MQTT

Serial.println("Data sent to MQTT broker");

} else {

Serial.println("Failed to send data to MQTT broker");

}

```

```
}
```

```
}
```

B2 Κώδικας Arduino μέτρηση ποσοστού οξυγόνου στο αίμα

Η μέτρηση ποσοστού οξυγόνου στο αίμα γίνεται με των παρακάτω κώδικα.

Η βάση του κώδικα για το **Arduino** είναι από την βιβλιοθήκη του SparkFun_MAX3010x_Pulse_and_Proximity_Sensor_Library το παράδειγμα οκτώ.

```
// Libraries
```

```
#include <Wire.h>
```

```
#include "MAX30105.h"
```

```
#include "spo2_algorithm.h"
```

```
MAX30105 particleSensor; // Call class
```

```
#define MAX_BRIGHTNESS 255
```

```
// For serial connection
```

```
unsigned long previousMillis = 0;
```

```
const long interval = 8000;
```

```
uint32_t irBuffer[100]; // infrared LED sensor data
```

```
uint32_t redBuffer[100]; // red LED sensor data
```

```
int32_t bufferLength; // data length
```

```

int32_t spo2; // SPO2 value

int8_t validSPO2; // indicator to show if the SPO2 calculation is
valid

uint8_t readLED = 4; // PWM pin for onboard LED indication

void setup()
{
    Serial.begin(115200); // initialize serial communication at
115200 bits per second:

    Serial1.begin(115200); // serial ESP

    while (!Serial);

    pinMode(readLED, OUTPUT);

    // Initialize sensor

    if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) // Use
default I2C port, 400kHz speed

    {
        Serial.println(F("MAX30105 was not found. Please check
wiring/power."));

        while (1);
    }

    uint8_t ledBrightness = 60; // Options: 0=Off to 255=50mA

    uint8_t sampleAverage = 4; // Options: 1, 2, 4, 8, 16, 32

```

```

    uint8_t ledMode = 2;      // Options: 1 = Red only, 2 = Red +
IR, 3 = Red + IR + Green

    uint8_t sampleRate = 100; // Options: 50, 100, 200, 400, 800,
1000, 1600, 3200

    int pulseWidth = 411;    // Options: 69, 118, 215, 411

    int adcRange = 4096;    // Options: 2048, 4096, 8192, 16384

particleSensor.setup(ledBrightness, sampleAverage, ledMode,
sampleRate, pulseWidth, adcRange); // Configure sensor
}

void loop()
{
    bufferLength = 100; // buffer length of 100 stores 4 seconds of
samples running at 25sps

// Read the first 100 samples, and determine the signal range

for (uint8_t i = 0 ; i < bufferLength ; i++)
{
    while (particleSensor.available() == false) // Do we have new
data?

    particleSensor.check(); // Check the sensor for new data

    redBuffer[i] = particleSensor.getRed();

    irBuffer[i] = particleSensor.getIR();
}

```

```

particleSensor.nextSample(); // We're finished with this sample
so move to next sample

// Print ir and red values

Serial.print(F("red="));

Serial.print(redBuffer[i], DEC);

Serial.print(F(", ir="));

Serial.println(irBuffer[i], DEC);

}

// Calculate SpO2 after the first 100 samples

maxim_heart_rate_and_oxygen_saturation(irBuffer,
bufferLength, redBuffer, &spo2, &validSPO2, nullptr, nullptr);

// Continuously take samples from MAX30102 and recalculate
SpO2 every 1 second

while (1)

{

// Dump the first 25 sets of samples in memory and shift the
last 75 sets to the top

for (uint8_t i = 25; i < 100; i++)

{

redBuffer[i - 25] = redBuffer[i];

irBuffer[i - 25] = irBuffer[i];

}

// Take 25 sets of samples before recalculating SpO2

```

```

for (uint8_t i = 75; i < 100; i++)
{
    while (particleSensor.available() == false) // Do we have new
    data?

    particleSensor.check(); // Check the sensor for new data

    digitalWrite(readLED, !digitalRead(readLED)); // Blink
    onboard LED with every data read

    redBuffer[i] = particleSensor.getRed();
    irBuffer[i] = particleSensor.getIR();
    particleSensor.nextSample(); // Move to next sample

    // Send SpO2 calculation result to terminal program through
    UART
    Serial.print(F("red="));
    Serial.print(redBuffer[i], DEC);
    Serial.print(F(", ir="));
    Serial.print(irBuffer[i], DEC);

    Serial.print(F(", SPO2="));
    Serial.print(spo2, DEC);

    Serial.print(F(", SPO2Valid="));
    Serial.println(validSPO2, DEC);
}

```

```

    }

// After gathering 25 new samples recalculate SpO2

    maxim_heart_rate_and_oxygen_saturation(irBuffer,
bufferLength, redBuffer, &spo2, &validSPO2, nullptr, nullptr);

// Send SpO2 data via Serial to ESP8266 every 8 seconds

unsigned long currentMillis = millis();

if (currentMillis - previousMillis >= interval) {

    previousMillis = currentMillis;

    String data = String(spo2);

    Serial.println("Sending data: " + data);

// Send data via Serial to ESP8266

    Serial1.println(data);

}

}

}

```

B2.1 Κώδικας ESP-8266 κυκλώματος μέτρησης οξυγόνου

Για το **ESP8266** είναι παρόμοιο με τις προηγούμενη εφαρμογή, το μόνο που αλλάζει είναι το όνομα του θέματος.

```

// libraries

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

```

```

// Network imformation

const char* ssid = "Odyn";

const char* password = "ody123456";


// MQTT broker information

const char* mqtt_server = "test.mosquitto.org"; // IP
142.93.191.12

const char* HeartRate_topic = "sensor/Spo2"; //topic

const int mqtt_port = 1883;

WiFiClient espClient; // Class TCP communication to MQTT
server

PubSubClient client(espClient); // class for Publish and subscribe
using TCP communication


void setup() {

    Serial.begin(115200);

    setup_wifi(); // Call function for the Connection to wifi

    client.setServer(mqtt_server, mqtt_port); // Set-up for MQTT
    Server

}

//Connect to wifi

void setup_wifi() {

    delay(200);

    Serial.println();

```

```

Serial.print("Connecting to ");
Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}

Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}

// starting the connection

void reconnect() {
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        if (client.connect("ESP8266Client")) {
            Serial.println("connected");
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
        }
    }
}

```

```

    Serial.println(" try again in 5 seconds");

    delay(5000);

}

}

}

void loop() {

if (!client.connected()) {

    reconnect();//reconnect

}

client.loop();

}

if (Serial.available() > 0) {

    String data = Serial.readStringUntil('\n'); // receive data from
arduino

    Serial.print("Received data: ");

    Serial.println(data);

}

if (client.publish(HeartRate_topic, data.c_str())) { // sending
data via MQTT

    Serial.println("Data sent to MQTT broker");

} else {

    Serial.println("Failed to send data to MQTT broker");

}

}

```

}

B3	Κώδικας	Arduino	απεικόνισης
ηλεκτροκαρδιογραφήματος			

Στη συνέχεια αναγράφεται ο κώδικας του αισθητήρα AD8232 για την απεικόνιση του ηλεκτροκαρδιογραφήματος.

Για το Arduino χρησιμοποιήθηκε η βιβλιοθήκη ECG από το κανάλι Viral science [43].

```
// Declaration varable
```

```
const long interval = 100;
```

```
unsigned long previousMillis = 0;
```

```
void setup() {
```

```
    // Initialize the serial communication:
```

```
    Serial.begin(9600);
```

```
    Serial1.begin(9600);
```

```
    pinMode(10, INPUT); // Setup for leads off detection LO +
```

```
    pinMode(11, INPUT); // Setup for leads off detection LO -
```

}

```
void loop() {
```

```
    if((digitalRead(10) == 1)||(digitalRead(11) == 1)){
```

```
        Serial.println('!');
```

}

```

else{

    // send the value of analog input 0:

    Serial.println(analogRead(A0));

}

//Wait for a bit to keep serial data from saturating

delay(1);

unsigned long currentMillis = millis();

if (currentMillis - previousMillis >= interval) {

    previousMillis = currentMillis;

    int data = analogRead(A0);

    //Serial.println("Sending data: " + data);

    // Send data via Serial to ESP8266

    Serial1.println(data);

}

}

```

B3.1 Κώδικας για το ESP8266 εφαρμογής ηλεκτροκαρδιογραφήματος

Για το ESP8266 ο κώδικας είναι παρόμοιος με την προηγούμενη εφαρμογή της μέτρησης καρδιακών παλμών. Οι διαφορά είναι ότι η σειριακή επικοινωνία γίνεται στα 9600 και αλλάζει το θέμα (Topic).

// libraries

```

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

// Network imformation

const char* ssid = "Odyn";

const char* password = "ody123456";

// MQTT broker information

const char* mqtt_server = "test.mosquitto.org"; // IP
142.93.191.12

const char* HeartRate_topic = "sensor/heartPloter"; // Topic

const int mqtt_port = 1883; // Port

WiFiClient espClient; // Class TCP communication to MQTT
server

PubSubClient client(espClient); // class for Publish and subscribe
using TCP communication

void setup() {

    Serial.begin(9600);

    setup_wifi(); //Connect to wifi

    client.setServer(mqtt_server, mqtt_port); // Set for MQTT
    Server

}

void setup_wifi() {

```

```

delay(200);

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password); // starting the connection

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

    delay(500);

    Serial.print(".");
}

Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());

}

// starting the connection

void reconnect() {

    while (!client.connected()) {

        Serial.print("Attempting MQTT connection...");

        if (client.connect("ESP8266Client")) {

            Serial.println("connected");

        } else {

```

```

        Serial.print("failed, rc=");
        Serial.print(client.state());
        Serial.println(" try again in 5 seconds");
        delay(5000);
    }

}

}

void loop() {
    if (!client.connected()) {
        reconnect();//reconnect
    }
    client.loop();

    if (Serial.available() > 0) {
        String data = Serial.readStringUntil('\n'); // receive data from
        arduino
        Serial.print("Received data: ");
        Serial.println(data);

        if (client.publish(HeartRate_topic, data.c_str())) { // sending
            data via MQTT
            Serial.println("Data sent to MQTT broker");
        } else {
            Serial.println("Failed to send data to MQTT broker");
        }
    }
}

```

```
    }  
}  
}
```

B4 Κώδικας Arduino μοντέλου αναγνώρισης στάσης

Ο κώδικας εξήχθη από την εφαρμογή Edge Impulse, ενώ η διαδικασία και τα επιμέρους βήματα αναλύονται αναλυτικά στο Παράρτημα A6.

```
/* Edge Impulse ingestion SDK  
* Copyright (c) 2022 EdgeImpulse Inc.  
*  
* Licensed under the Apache License, Version 2.0 (the  
* "License");  
* you may not use this file except in compliance with the License.  
* You may obtain a copy of the License at  
* http://www.apache.org/licenses/LICENSE-2.0  
*  
* Unless required by applicable law or agreed to in writing,  
* software  
* distributed under the License is distributed on an "AS IS"  
* BASIS,  
* WITHOUT WARRANTIES OR CONDITIONS OF ANY  
* KIND, either express or implied.  
* See the License for the specific language governing  
* permissions and  
* limitations under the License.  
*  
*/
```

```

/* Includes -----
 */

#include <CV_Detection_inferencing.h>

#include <Arduino_OV767X.h> //Click here to get the library:
https://www.arduino.cc/reference/en/libraries/arduino\_ov767x/

#include <stdint.h>
#include <stdlib.h>

/* Constant variables -----
--- */

#define EI_CAMERA_RAW_FRAME_BUFFER_COLS 160
#define EI_CAMERA_RAW_FRAME_BUFFER_ROWS 120

#define DWORD_ALIGN_PTR(a) ((a & 0x3) ?(((uintptr_t)a + 0x4) & ~uintptr_t(0x3)) : a)

/*
** NOTE: If you run into TFLite arena allocation issue.
**
** This may be due to memory fragmentation.
** Try defining "-DEI_CLASSIFIER_ALLOCATION_STATIC" in boards.local.txt (create
** if it doesn't exist) and copy this file to

```

```

**<ARDUINO_CORE_INSTALL_PATH>/arduino/hardware/<m
bed_core>/<core_version>`.

** See

** (https://support.arduino.cc/hc/en-us/articles/360012076960-
Where-are-the-installed-cores-located-)

** to find where Arduino installs cores on your machine.

** If the problem persists then there's not enough memory for
this model and application.

*/

```

```

/* Edge Impulse -----
--- */

class OV7675 : public OV767X {

public:

    int begin(int resolution, int format, int fps);

    void readFrame(void* buffer);

private:

    int vsyncPin;

    int hrefPin;

    int pclkPin;

    int xclkPin;

    volatile uint32_t* vsyncPort;

```

```

        uint32_t vsyncMask;

        volatile uint32_t* hrefPort;

        uint32_t hrefMask;

        volatile uint32_t* pclkPort;

        uint32_t pclkMask;

        uint16_t width;

        uint16_t height;

        uint8_t bytes_per_pixel;

        uint16_t bytes_per_row;

        uint8_t buf_rows;

        uint16_t buf_size;

        uint8_t resize_height;

        uint8_t *raw_buf;

        void *buf_mem;

        uint8_t *intrp_buf;

        uint8_t *buf_limit;

    void readBuf();

    int allocate_scratch_buffs();

    int deallocate_scratch_buffs();

};

typedef struct {

    size_t width;

    size_t height;

```

```

} ei_device_resize_resolutions_t;

/**

 * @brief    Check if new serial data is available
 *
 * @return   Returns number of available bytes
 */

int ei_get_serial_available(void) {

    return Serial.available();

}

/**

 * @brief    Get next available byte
 *
 * @return   byte
 */

char ei_get_serial_byte(void) {

    return Serial.read();

}

/* Private variables -----
 - */

static OV7675 Cam;

static bool is_initialised = false;

/*

```

```

** @brief points to the output of the capture

*/

static uint8_t *ei_camera_capture_out = NULL;

uint32_t resize_col_sz;
uint32_t resize_row_sz;
bool do_resize = false;
bool do_crop = false;

static bool debug_nn = false; // Set this to true to see e.g. features
generated from the raw signal

/* Function definitions -----
----- */

bool ei_camera_init(void);
void ei_camera_deinit(void);

bool ei_camera_capture(uint32_t img_width, uint32_t
img_height, uint8_t *out_buf);

int calculate_resize_dimensions(uint32_t out_width, uint32_t
out_height, uint32_t *resize_col_sz, uint32_t *resize_row_sz,
bool *do_resize);

void resizeImage(int srcWidth, int srcHeight, uint8_t *srcImage,
int dstWidth, int dstHeight, uint8_t *dstImage, int iBpp);

void cropImage(int srcWidth, int srcHeight, uint8_t *srcImage,
int startX, int startY, int dstWidth, int dstHeight, uint8_t
*dstImage, int iBpp);

/***
* @brief Arduino setup function

```

```

*/



void setup()
{
    // put your setup code here, to run once:
    Serial.begin(115200);

    // comment out the below line to cancel the wait for USB
    // connection (needed for native USB)

    while (!Serial);

    Serial.println("Edge Impulse Inferencing Demo");




    // summary of inferencing settings (from model_metadata.h)
    ei_printf("Inferencing settings:\n");
    ei_printf("\tImage resolution: %dx%d\n",
EI_CLASSIFIER_INPUT_WIDTH,
EI_CLASSIFIER_INPUT_HEIGHT);

    ei_printf("\tFrame size: %d\n",
EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE);

    ei_printf("\tNo. of classes: %d\n",
sizeof(ei_classifier_inferencing_categories) /
sizeof(ei_classifier_inferencing_categories[0]));

}

/**



* @brief Get data and run inferencing
*
* @param[in] debug Get debug info if true
*/
void loop()

```

```

{

    bool stop_inferencing = false;

    while(stop_inferencing == false) {

        ei_printf("\nStarting inferencing in 2 seconds...\n");

        // instead of wait_ms, we'll wait on the signal, this allows
        threads to cancel us...

        if (ei_sleep(2000) != EI_IMPULSE_OK) {

            break;
        }

        ei_printf("Taking photo...\n");

        if (ei_camera_init() == false) {

            ei_printf("ERR: Failed to initialize image sensor\r\n");

            break;
        }

        // choose resize dimensions

        uint32_t resize_col_sz;
        uint32_t resize_row_sz;
        bool do_resize = false;

        int             res      =
calculate_resize_dimensions(EI_CLASSIFIER_INPUT_WIDT
H,      EI_CLASSIFIER_INPUT_HEIGHT,      &resize_col_sz,
&resize_row_sz, &do_resize);
    }
}

```

```

    if (res) {

        ei_printf("ERR: Failed to calculate resize dimensions
(%d)\r\n", res);

        break;

    }

void *snapshot_mem = NULL;

uint8_t *snapshot_buf = NULL;

snapshot_mem = ei_malloc(resize_col_sz*resize_row_sz*2);

if(snapshot_mem == NULL) {

    ei_printf("failed to create snapshot_mem\r\n");

    break;

}

snapshot_buf = (uint8_t
*)DWORD_ALIGN_PTR((uintptr_t)snapshot_mem);

if (ei_camera_capture(EI_CLASSIFIER_INPUT_WIDTH,
EI_CLASSIFIER_INPUT_HEIGHT, snapshot_buf) == false) {

    ei_printf("Failed to capture image\r\n");

    if (snapshot_mem) ei_free(snapshot_mem);

    break;

}

ei::signal_t signal;

signal.total_length = EI_CLASSIFIER_INPUT_WIDTH *
EI_CLASSIFIER_INPUT_HEIGHT;

```

```

    signal.get_data = &ei_camera_cutout_get_data;

    // run the impulse: DSP, neural network and the Anomaly
    algorithm

    ei_impulse_result_t result = { 0 };

    EI_IMPULSE_ERROR ei_error = run_classifier(&signal,
&result, debug_nn);

    if (ei_error != EI_IMPULSE_OK) {

        ei_printf("Failed to run impulse (%d)\n", ei_error);

        ei_free(snapshot_mem);

        break;

    }

    // print the predictions

    ei_printf("Predictions (DSP: %d ms., Classification: %d ms.,
Anomaly: %d ms.):\n",
            result.timing.dsp, result.timing.classification,
            result.timing.anomaly);

    #if EI_CLASSIFIER_OBJECT_DETECTION == 1

        ei_printf("Object detection bounding boxes:\r\n");

        for (uint32_t i = 0; i < result.bounding_boxes_count; i++) {

            ei_impulse_result_bounding_box_t bb =
result.bounding_boxes[i];

            if (bb.value == 0) {

                continue;

            }

        }

    
```

```

    ei_printf(" %s (%f) [ x: %u, y: %u, width: %u, height:
%u ]\r\n",
        bb.label,
        bb.value,
        bb.x,
        bb.y,
        bb.width,
        bb.height);

}

// Print the prediction results (classification)

#else

ei_printf("Predictions:\r\n");
for (uint16_t i = 0; i < EI_CLASSIFIER_LABEL_COUNT;
i++) {
    ei_printf(" %s: ", ei_classifier_inferencing_categories[i]);
    ei_printf("%0.5f\r\n", result.classification[i].value);
}
#endif

// Print anomaly result (if it exists)

#if EI_CLASSIFIER_HAS_ANOMALY
ei_printf("Anomaly prediction: %.3f\r\n", result.anomaly);
#endif

#if EI_CLASSIFIER_HAS_VISUAL_ANOMALY
ei_printf("Visual anomalies:\r\n");

```

```

        for (uint32_t i = 0; i < result.visual_ad_count; i++) {
            ei_impulse_result_bounding_box_t bb =
result.visual_ad_grid_cells[i];
            if (bb.value == 0) {
                continue;
            }
            ei_printf(" %s (%f) [ x: %u, y: %u, width: %u, height:
%u ]\r\n",
bb.label,
bb.value,
bb.x,
bb.y,
bb.width,
bb.height);
        }
#endif

while (ei_get_serial_available() > 0) {
    if (ei_get_serial_byte() == 'b') {
        ei_printf("Inferencing stopped by user\r\n");
        stop_inferencing = true;
    }
    if (snapshot_mem) ei_free(snapshot_mem);
}
ei_camera_deinit();
}

```

```

/**
 * @brief Determine whether to resize and to which dimension
 *
 * @param[in] out_width width of output image
 * @param[in] out_height height of output image
 * @param[out] resize_col_sz pointer to frame buffer's
 * column/width value
 * @param[out] resize_row_sz pointer to frame buffer's
 * rows/height value
 * @param[out] do_resize returns whether to resize (or not)
 *
 */

int calculate_resize_dimensions(uint32_t out_width, uint32_t
out_height, uint32_t *resize_col_sz, uint32_t *resize_row_sz,
bool *do_resize)

{
    size_t list_size = 2;

    const ei_device_resize_resolutions_t list[list_size] = { {42,32},
{128,96} };

    // (default) conditions

        *resize_col_sz = EI_CAMERA_RAW_FRAME_BUFFER_COLS;
        *resize_row_sz = EI_CAMERA_RAW_FRAME_BUFFER_ROWS;
        *do_resize = false;
}

```

```

        for (size_t ix = 0; ix < list_size; ix++) {

            if ((out_width <= list[ix].width) && (out_height <=
list[ix].height)) {

                *resize_col_sz = list[ix].width;

                *resize_row_sz = list[ix].height;

                *do_resize = true;

                break;

            }

        }

        return 0;

    }

}

/* @brief Setup image sensor & start streaming

 *

 * @retval false if initialisation failed

 */

bool ei_camera_init(void) {

    if (is_initialised) return true;

    if (!Cam.begin(QQVGA, RGB565, 1)) { // VGA downsampled
to QQVGA (OV7675)

        ei_printf("ERR: Failed to initialize camera\r\n");

        return false;

    }

    is_initialised = true;

```

```

        return true;

    }

/***
 * @brief      Stop streaming of sensor data
 */
void ei_camera_deinit(void) {
    if (is_initialised) {
        Cam.end();
        is_initialised = false;
    }
}

/***
 * @brief      Capture, rescale and crop image
 *
 * @param[in]  img_width    width of output image
 * @param[in]  img_height   height of output image
 * @param[in]  out_buf      pointer to store output image, NULL
 *                         may be used
 *                         when full resolution is expected.
 *
 * @retval     false if not initialised, image captured, rescaled or
 *             cropped failed
 *
 */

```

```

bool    ei_camera_capture(uint32_t    img_width,    uint32_t
img_height, uint8_t *out_buf)

{
    if (!is_initialised) {
        ei_printf("ERR: Camera is not initialized\r\n");
        return false;
    }

    if (!out_buf) {
        ei_printf("ERR: invalid parameters\r\n");
        return false;
    }

    // choose resize dimensions

    int res = calculate_resize_dimensions(img_width, img_height,
&resize_col_sz, &resize_row_sz, &do_resize);

    if (res) {
        ei_printf("ERR: Failed to calculate resize dimensions
(%d)\r\n", res);
        return false;
    }

    if ((img_width != resize_col_sz)
        || (img_height != resize_row_sz)) {
        do_crop = true;
    }
}

```

```

Cam.readFrame(out_buf); // captures image and resizes

if (do_crop) {

    uint32_t crop_col_sz;
    uint32_t crop_row_sz;
    uint32_t crop_col_start;
    uint32_t crop_row_start;
    crop_row_start = (resize_row_sz - img_height) / 2;
    crop_col_start = (resize_col_sz - img_width) / 2;
    crop_col_sz = img_width;
    crop_row_sz = img_height;

    //ei_printf("crop   cols:  %d,   rows:  %d\r\n",
    crop_col_sz,crop_row_sz);

    cropImage(resize_col_sz, resize_row_sz,
              out_buf,
              crop_col_start, crop_row_start,
              crop_col_sz, crop_row_sz,
              out_buf,
              16);

}

// The following variables should always be assigned
// if this routine is to return true
// cutout values
//ei_camera_snapshot_is_resized = do_resize;

```

```

//ei_camera_snapshot_is_cropped = do_crop;

ei_camera_capture_out = out_buf;

return true;

}

/***
* @brief    Convert RGB565 raw camera buffer to RGB888
*
* @param[in] offset      pixel offset of raw buffer
* @param[in] length     number of pixels to convert
* @param[out] out_buf    pointer to store output image
*/
int ei_camera_cutout_get_data(size_t offset, size_t length, float
*out_ptr) {

    size_t pixel_ix = offset * 2;
    size_t bytes_left = length;
    size_t out_ptr_ix = 0;

    // read byte for byte
    while (bytes_left != 0) {
        // grab the value and convert to r/g/b
        uint16_t pixel = (ei_camera_capture_out[pixel_ix] << 8) |
ei_camera_capture_out[pixel_ix+1];
        uint8_t r, g, b;
        r = ((pixel >> 11) & 0x1f) << 3;
        g = ((pixel >> 5) & 0x3f) << 2;

```

```

    b = (pixel & 0x1f) << 3;

    // then convert to out_ptr format
    float pixel_f = (r << 16) + (g << 8) + b;
    out_ptr[out_ptr_ix] = pixel_f;

    // and go to the next pixel
    out_ptr_ix++;
    pixel_ix+=2;
    bytes_left--;
}

// and done!
return 0;
}

// This include file works in the Arduino environment
// to define the Cortex-M intrinsics
#ifndef __ARM_FEATURE SIMD32
#include <device.h>
#endif

// This needs to be < 16 or it won't fit. Cortex-M4 only has SIMD
for signed multiplies
#define FRAC_BITS 14
#define FRAC_VAL (1<<FRAC_BITS)
#define FRAC_MASK (FRAC_VAL - 1)

```

```

// Resize

// Assumes that the destination buffer is dword-aligned

// Can be used to resize the image smaller or larger

// If resizing much smaller than 1/3 size, then a more robust
algorithm should average all of the pixels

// This algorithm uses bilinear interpolation - averages a 2x2
region to generate each new pixel

// Optimized for 32-bit MCUs

// supports 8 and 16-bit pixels

void resizeImage(int srcWidth, int srcHeight, uint8_t *srcImage,
int dstWidth, int dstHeight, uint8_t *dstImage, int iBpp)

{
    uint32_t src_x_accum, src_y_accum; // accumulators and
fractions for scaling the image

    uint32_t x_frac, nx_frac, y_frac, ny_frac;

    int x, y, ty, tx;

    if (iBpp != 8 && iBpp != 16)

        return;

    src_y_accum = FRAC_VAL/2; // start at 1/2 pixel in to account
for integer downsampling which might miss pixels

    const uint32_t src_x_frac = (srcWidth * FRAC_VAL) /
dstWidth;

    const uint32_t src_y_frac = (srcHeight * FRAC_VAL) /
dstHeight;

```

```

const uint32_t r_mask = 0xf800f800;
const uint32_t g_mask = 0x07e007e0;
const uint32_t b_mask = 0x001f001f;
uint8_t *s, *d;
uint16_t *s16, *d16;
uint32_t x_frac2, y_frac2; // for 16-bit SIMD
for (y=0; y < dstHeight; y++) {
    ty = src_y_accum >> FRAC_BITS; // src y
    y_frac = src_y_accum & FRAC_MASK;
    src_y_accum += src_y_frac;
    ny_frac = FRAC_VAL - y_frac; // y fraction and 1.0 - y
fraction
    y_frac2 = ny_frac | (y_frac << 16); // for M4/M4 SIMD
    s = &srcImage[ty * srcWidth];
    s16 = (uint16_t *)&srcImage[ty * srcWidth * 2];
    d = &dstImage[y * dstWidth];
    d16 = (uint16_t *)&dstImage[y * dstWidth * 2];
    src_x_accum = FRAC_VAL/2; // start at 1/2 pixel in to
account for integer downsampling which might miss pixels
    if (iBpp == 8) {
        for (x=0; x < dstWidth; x++) {
            uint32_t tx, p00,p01,p10,p11;
            tx = src_x_accum >> FRAC_BITS;
            x_frac = src_x_accum & FRAC_MASK;
            nx_frac = FRAC_VAL - x_frac; // x fraction and 1.0 - x
fraction
            x_frac2 = nx_frac | (x_frac << 16);

```

```

src_x_accum += src_x_frac;

p00 = s[tx]; p10 = s[tx+1];

p01 = s[tx+srcWidth]; p11 = s[tx+srcWidth+1];

#ifndef __ARM_FEATURE SIMD32

    p00 = __SMLAD(p00 | (p10<<16), x_frac2,
FRAC_VAL/2)>> FRAC_BITS; // top line

    p01 = __SMLAD(p01 | (p11<<16), x_frac2,
FRAC_VAL/2)>> FRAC_BITS; // bottom line

    p00 = __SMLAD(p00 | (p01<<16), y_frac2,
FRAC_VAL/2)>> FRAC_BITS; // combine

#else // generic C code

    p00 = ((p00 * nx_frac) + (p10 * x_frac) + FRAC_VAL/2)
>> FRAC_BITS; // top line

    p01 = ((p01 * nx_frac) + (p11 * x_frac) + FRAC_VAL/2)
>> FRAC_BITS; // bottom line

    p00 = ((p00 * ny_frac) + (p01 * y_frac) + FRAC_VAL/2)
>> FRAC_BITS; // combine top + bottom

#endif // Cortex-M4/M7

*d++ = (uint8_t)p00; // store new pixel

} // for x

} // 8-bpp

else

{ // RGB565

for (x=0; x < dstWidth; x++) {

    uint32_t tx, p00,p01,p10,p11;

    uint32_t r00, r01, r10, r11, g00, g01, g10, g11, b00, b01,
b10, b11;

    tx = src_x_accum >> FRAC_BITS;

```

```

x_frac = src_x_accum & FRAC_MASK;

nx_frac = FRAC_VAL - x_frac; // x fraction and 1.0 - x
fraction

x_frac2 = nx_frac | (x_frac << 16);

src_x_accum += src_x_frac;

p00 = __builtin_bswap16(s16[tx]); p10 =
__builtin_bswap16(s16[tx+1]);

p01 = __builtin_bswap16(s16[tx+srcWidth]); p11 =
__builtin_bswap16(s16[tx+srcWidth+1]);

#ifndef __ARM_FEATURE SIMD32

{
    p00 |= (p10 << 16);

    p01 |= (p11 << 16);

    r00 = (p00 & r_mask) >> 1; g00 = p00 & g_mask; b00 =
    p00 & b_mask;

    r01 = (p01 & r_mask) >> 1; g01 = p01 & g_mask; b01 =
    p01 & b_mask;

    r00 = __SMLAD(r00, x_frac2, FRAC_VAL/2) >>
    FRAC_BITS; // top line

    r01 = __SMLAD(r01, x_frac2, FRAC_VAL/2) >>
    FRAC_BITS; // bottom line

    r00 = __SMLAD(r00 | (r01<<16), y_frac2, FRAC_VAL/2)
    >> FRAC_BITS; // combine

    g00 = __SMLAD(g00, x_frac2, FRAC_VAL/2) >>
    FRAC_BITS; // top line

    g01 = __SMLAD(g01, x_frac2, FRAC_VAL/2) >>
    FRAC_BITS; // bottom line

    g00 = __SMLAD(g00 | (g01<<16), y_frac2,
    FRAC_VAL/2) >> FRAC_BITS; // combine
}

```

```

    b00 = __SMLAD(b00, x_frac2, FRAC_VAL/2) >>
FRAC_BITS; // top line

    b01 = __SMLAD(b01, x_frac2, FRAC_VAL/2) >>
FRAC_BITS; // bottom line

    b00 = __SMLAD(b00 | (b01<<16), y_frac2,
FRAC_VAL/2)>>FRAC_BITS; // combine

}

#else // generic C code

{

    r00 = (p00 & r_mask) >> 1; g00 = p00 & g_mask; b00 =
p00 & b_mask;

    r10 = (p10 & r_mask) >> 1; g10 = p10 & g_mask; b10 =
p10 & b_mask;

    r01 = (p01 & r_mask) >> 1; g01 = p01 & g_mask; b01 =
p01 & b_mask;

    r11 = (p11 & r_mask) >> 1; g11 = p11 & g_mask; b11 =
p11 & b_mask;

    r00 = ((r00 * nx_frac) + (r10 * x_frac) + FRAC_VAL/2)
>> FRAC_BITS; // top line

    r01 = ((r01 * nx_frac) + (r11 * x_frac) + FRAC_VAL/2)
>> FRAC_BITS; // bottom line

    r00 = ((r00 * ny_frac) + (r01 * y_frac) + FRAC_VAL/2)
>> FRAC_BITS; // combine top + bottom

    g00 = ((g00 * nx_frac) + (g10 * x_frac) + FRAC_VAL/2)
>> FRAC_BITS; // top line

    g01 = ((g01 * nx_frac) + (g11 * x_frac) + FRAC_VAL/2)
>> FRAC_BITS; // bottom line

    g00 = ((g00 * ny_frac) + (g01 * y_frac) + FRAC_VAL/2)
>> FRAC_BITS; // combine top + bottom

```

```

        b00 = ((b00 * nx_frac) + (b10 * x_frac) + FRAC_VAL/2)
        >> FRAC_BITS; // top line

        b01 = ((b01 * nx_frac) + (b11 * x_frac) + FRAC_VAL/2)
        >> FRAC_BITS; // bottom line

        b00 = ((b00 * ny_frac) + (b01 * y_frac) + FRAC_VAL/2)
        >> FRAC_BITS; // combine top + bottom

    }

#endif // Cortex-M4/M7

r00 = (r00 << 1) & r_mask;

g00 = g00 & g_mask;

b00 = b00 & b_mask;

p00 = (r00 | g00 | b00); // re-combine color components

*d16++ = (uint16_t)__builtin_bswap16(p00); // store new
pixel

} // for x

} // 16-bpp

} // for y

} /* resizeImage() */

//



// Crop

//



// Assumes that the destination buffer is dword-aligned

// optimized for 32-bit MCUs

// Supports 8 and 16-bit pixels

//


void cropImage(int srcWidth, int srcHeight, uint8_t *srcImage,
int startX, int startY, int dstWidth, int dstHeight, uint8_t
*dstImage, int iBpp)

```

```

{
    uint32_t *s32, *d32;
    int x, y;

    if (startX < 0 || startX >= srcWidth || startY < 0 || startY >=
srcHeight || (startX + dstWidth) > srcWidth || (startY + dstHeight)
> srcHeight)
        return; // invalid parameters

    if (iBpp != 8 && iBpp != 16)
        return;

    if (iBpp == 8) {
        uint8_t *s, *d;
        for (y=0; y<dstHeight; y++) {
            s = &srcImage[srcWidth * (y + startY) + startX];
            d = &dstImage[(dstWidth * y)];
            x = 0;
            if (((intptr_t)s & 3 || (intptr_t)d & 3) { // either src or dst
pointer is not aligned
                for (; x<dstWidth; x++) {
                    *d++ = *s++; // have to do it byte-by-byte
                }
            } else {
                // move 4 bytes at a time if aligned or alignment not enforced
                s32 = (uint32_t *)s;
                d32 = (uint32_t *)d;
                for (; x<dstWidth-3; x+= 4) {

```

```

*d32++ = *s32++;
}

// any remaining stragglers?

s = (uint8_t *)s32;
d = (uint8_t *)d32;

for (; x<dstWidth; x++) {

    *d++ = *s++;
}

}

}

} // for y

} // 8-bpp

else

{

    uint16_t *s, *d;

    for (y=0; y<dstHeight; y++) {

        s = (uint16_t *)&srcImage[2 * srcWidth * (y + startY) +
startX * 2];

        d = (uint16_t *)&dstImage[(dstWidth * y * 2)];

        x = 0;

        if (((intptr_t)s & 2 || (intptr_t)d & 2) { // either src or dst
pointer is not aligned

            for (; x<dstWidth; x++) {

                *d++ = *s++; // have to do it 16-bits at a time

            }

        } else {

            // move 4 bytes at a time if aligned or alignment no enforced

            s32 = (uint32_t *)s;

```

```

d32 = (uint32_t *)d;

for (; x<dstWidth-1; x+= 2) { // we can move 2 pixels at a
time

    *d32++ = *s32++;

}

// any remaining stragglers?

s = (uint16_t *)s32;

d = (uint16_t *)d32;

for (; x<dstWidth; x++) {

    *d++ = *s++;

}

}

}

} // for y

} // 16-bpp case

} /* cropImage() */

#ifndef           !defined(EI_CLASSIFIER_SENSOR)      ||
EI_CLASSIFIER_SENSOR          !=      ||
EI_CLASSIFIER_SENSOR_CAMERA

#error "Invalid model for current sensor"

#endif

// OV767X camera library override

#include <Arduino.h>

#include <Wire.h>

```

```

#define digitalPinToBitMask(P) (1 << (digitalPinToPinName(P)
% 32))

#define portInputRegister(P) ((P == 0) ? &NRF_P0->IN :
&NRF_P1->IN)

//  

// OV7675::begin()  

//  

// Extends the OV767X library function. Some private variables  

// are needed  

// to use the OV7675::readFrame function.  

//  

int OV7675::begin(int resolution, int format, int fps)  

{  

    pinMode(OV7670_VSYNC, INPUT);  

    pinMode(OV7670_HREF, INPUT);  

    pinMode(OV7670_PLK, INPUT);  

    pinMode(OV7670_XCLK, OUTPUT);  

    vsyncPort      =  

portInputRegister(digitalPinToPort(OV7670_VSYNC));  

    vsyncMask = digitalPinToBitMask(OV7670_VSYNC);  

    hrefPort      =  

portInputRegister(digitalPinToPort(OV7670_HREF));  

    hrefMask = digitalPinToBitMask(OV7670_HREF);  

    pclkPort      =  

portInputRegister(digitalPinToPort(OV7670_PLK));  

    pclkMask = digitalPinToBitMask(OV7670_PLK);

```

```

// init driver to use full image sensor size

bool ret = OV767X::begin(VGA, format, fps);

width = OV767X::width(); // full sensor width

height = OV767X::height(); // full sensor height

bytes_per_pixel = OV767X::bytesPerPixel();

bytes_per_row = width * bytes_per_pixel; // each pixel is 2

bytes

resize_height = 2;

buf_mem = NULL;

raw_buf = NULL;

intrp_buf = NULL;

//allocate_scratch_buffs();

return ret;

} /* OV7675::begin() */

int OV7675::allocate_scratch_buffs()

{

//ei_printf("allocating buffers..\r\n");

buf_rows = height / resize_row_sz * resize_height;

buf_size = bytes_per_row * buf_rows;

buf_mem = ei_malloc(buf_size);

if(buf_mem == NULL) {

```

```

    ei_printf("failed to create buf_mem\r\n");
    return false;
}

        raw_buf          =      (uint8_t
*)DWORD_ALIGN_PTR((uintptr_t)buf_mem);

//ei_printf("allocating buffers OK\r\n");
return 0;
}

int OV7675::deallocate_scratch_buffs()
{
    //ei_printf("deallocating buffers...\r\n");
    ei_free(buf_mem);
    buf_mem = NULL;

    //ei_printf("deallocating buffers OK\r\n");
    return 0;
}

// 
// OV7675::readFrame()
//
// Overrides the OV767X library function. Fixes the camera
// output to be
//
// a far more desirable image. This image utilizes the full sensor
// size

```

```

// and has the correct aspect ratio. Since there is limited memory
on the

// Nano we bring in only part of the entire sensor at a time and
then

// interpolate to a lower resolution.

//

void OV7675::readFrame(void* buffer)

{

    allocate_scratch_buffs();

    uint8_t* out = (uint8_t*)buffer;

    noInterrupts();

    // Falling edge indicates start of frame

    while ((*vsyncPort & vsyncMask) == 0); // wait for HIGH

    while ((*vsyncPort & vsyncMask) != 0); // wait for LOW

    int out_row = 0;

    for (int raw_height = 0; raw_height < height; raw_height +=

buf_rows) {

        // read in 640xbuf_rows buffer to work with

        readBuf();

        resizeImage(width, buf_rows,

raw_buf,

resize_col_sz, resize_height,

&(out[out_row]),


```

```

16);

        out_row += resize_col_sz * resize_height * bytes_per_pixel;
        /* resize_col_sz * 2 * 2 */

    }

    interrupts();

    deallocate_scratch_buffs();

} /* OV7675::readFrame() */


//  

// OV7675::readBuf()  

//  

// Extends the OV767X library function. Reads buf_rows VGA  

rows from the  

// image sensor.  

//  

void OV7675::readBuf()  

{  

    int offset = 0;  

    uint32_t ulPin = 33; // P1.xx set of GPIO is in 'pin' 32 and above  

    NRF_GPIO_Type * port;  

    port = nrf_gpio_pin_port_decode(&ulPin);

```

```

for (int i = 0; i < buf_rows; i++) {

    // rising edge indicates start of line

    while ((*hrefPort & hrefMask) == 0); // wait for HIGH


    for (int col = 0; col < bytes_per_row; col++) {

        // rising edges clock each data byte

        while ((*pclkPort & pclkMask) != 0); // wait for LOW


        uint32_t in = port->IN; // read all bits in parallel

        in >>= 2; // place bits 0 and 1 at the "bottom" of the register

        in &= 0x3f03; // isolate the 8 bits we care about

        in |= (in >> 6); // combine the upper 6 and lower 2 bits


        raw_buf[offset++] = in;

    }

    while ((*pclkPort & pclkMask) == 0); // wait for HIGH

}

while ((*hrefPort & hrefMask) != 0); // wait for LOW

}

} /* OV7675::readBuf() */

```

B5 Κώδικας Arduino μοντέλου αναγνώρισης και μέτρησης παλμών

Πρόκειται για τον κώδικα του μοντέλου αναγνώρισης και μέτρησης παλμών, ο οποίος αποτελεί συνδυασμό των παραπάνω κωδίκων: του μοντέλου αναγνώρισης και της μέτρησης

καρδιακών παλμών, με επιπλέον προσθήκη την εμφάνιση μηνύματος προειδοποίησης για χαμηλούς ή υψηλούς παλμούς.

```
#include <CV_Detection_inferencing.h>
#include <Arduino_OV767X.h>
#include <stdint.h>
#include <stdlib.h>
///////////////////////////////
//Max30102
#include <Wire.h>
#include "MAX30105.h"
#include "heartRate.h"

MAX30105 particleSensor; // Class to set the Max30102 sensor

const byte RATE_SIZE = 4; // Increase this for more averaging.
4 is good.

byte rates[RATE_SIZE]; // Array of heart rates
byte rateSpot = 0;
long lastBeat = 0; // Time at which the last beat occurred

float beatsPerMinute;
int beatAvg;
unsigned long lastHeartRateRead = 0;

void readHeartRate(MAX30105 &particleSensor){
```

```

long irValue = particleSensor.getIR(); // reading the value of
InfraRed LED

if (checkForBeat(irValue) == true) {

    long delta = millis() - lastBeat; // Calculate the difference
    between the present heartbeat and the last heartbeat

    lastBeat = millis();

    beatsPerMinute = 60 / (delta / 1000.0); // Calculate the beats
    per minute

    if (beatsPerMinute < 255 && beatsPerMinute > 20) {

        rates[rateSpot++] = (byte)beatsPerMinute; // Array which
        store the measurements

        rateSpot %= RATE_SIZE; // Determines in which position of
        the array the new measurement will be stored and increase by 1

        // Calculation of Average BPM

        beatAvg = 0;

        for (byte x = 0; x < RATE_SIZE; x++) {

            beatAvg += rates[x];

        }

        beatAvg /= RATE_SIZE;

    }

}

```

```

// Print Values

Serial.print("IR=");
Serial.print(irValue);
Serial.print(", BPM=");
Serial.print(beatsPerMinute);
Serial.print(", Avg BPM=");
Serial.print(beatAvg);

Serial.println();

}
////////////////////////////////////////////////////////////////////////

```

```

unsigned long readingCamera = 0;

/* Constant variables -----
--- */

#define EI_CAMERA_RAW_FRAME_BUFFER_COLS 160
#define EI_CAMERA_RAW_FRAME_BUFFER_ROWS 120

#define DWORD_ALIGN_PTR(a) ((a & 0x3) ?(((uintptr_t)a +
0x4) & ~uintptr_t)0x3) : a)


```

```

/* Edge Impulse -----
--- */


```

```

class OV7675 : public OV767X {
```

```
public:  
    int begin(int resolution, int format, int fps);  
    void readFrame(void* buffer);  
  
private:  
    int vsyncPin;  
    int hrefPin;  
    int pclkPin;  
    int xclkPin;  
  
    volatile uint32_t* vsyncPort;  
    uint32_t vsyncMask;  
    volatile uint32_t* hrefPort;  
    uint32_t hrefMask;  
    volatile uint32_t* pclkPort;  
    uint32_t pclkMask;  
  
    uint16_t width;  
    uint16_t height;  
    uint8_t bytes_per_pixel;  
    uint16_t bytes_per_row;  
    uint8_t buf_rows;  
    uint16_t buf_size;  
    uint8_t resize_height;  
    uint8_t *raw_buf;  
    void *buf_mem;
```

```

        uint8_t *intrp_buf;
        uint8_t *buf_limit;

    void readBuf();
    int allocate_scratch_buffs();
    int deallocate_scratch_buffs();
};

typedef struct {
    size_t width;
    size_t height;
} ei_device_resize_resolutions_t;

/***
 * @brief      Check if new serial data is available
 *
 * @return     Returns number of available bytes
 */
int ei_get_serial_available(void) {
    return Serial.available();
}

/***
 * @brief      Get next available byte
 *
 * @return     byte
*/

```

```

        */

    char ei_get_serial_byte(void) {
        return Serial.read();
    }

    /* Private variables -----
    - */

    static OV7675 Cam;
    static bool is_initialised = false;

    /*
    ** @brief points to the output of the capture
    */

    static uint8_t *ei_camera_capture_out = NULL;
    uint32_t resize_col_sz;
    uint32_t resize_row_sz;
    bool do_resize = false;
    bool do_crop = false;

    static bool debug_nn = false; // Set this to true to see e.g. features
    generated from the raw signal

    /* Function definitions -----
    ---- */

    bool ei_camera_init(void);
    void ei_camera_deinit(void);

```

```

    bool    ei_camera_capture(uint32_t    img_width,    uint32_t
    img_height, uint8_t *out_buf);

    int calculate_resize_dimensions(uint32_t out_width, uint32_t
    out_height, uint32_t *resize_col_sz, uint32_t *resize_row_sz,
    bool *do_resize);

    void resizeImage(int srcWidth, int srcHeight, uint8_t *srcImage,
    int dstWidth, int dstHeight, uint8_t *dstImage, int iBpp);

    void cropImage(int srcWidth, int srcHeight, uint8_t *srcImage,
    int startX, int startY, int dstWidth, int dstHeight, uint8_t
    *dstImage, int iBpp);

    /**
     * @brief    Arduino setup function
     */
void setup()
{
    Serial.begin(115200);

    Serial1.begin(115200); // Serial connection for ESP

    // comment out the below line to cancel the wait for USB
    // connection (needed for native USB)

    while (!Serial);

    /////////////////////////////////
    Serial.println("Initializing...");

    // Initialize sensor Max30102

    if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) {

        Serial.println("MAX30105 was not found. Please check
wiring/power.");
        while (1);
    }
}

```

```
}
```

```
particleSensor.setup();

particleSensor.setPulseAmplitudeRed(0x0A); // Turn Red LED
to low to indicate sensor is running

particleSensor.setPulseAmplitudeGreen(0); // Turn off Green
LED
```

```
////////////////////////////////////////////////////////////////////////
```

```
Serial.println("Edge Impulse Inferencing Demo");
Serial.println("Initializing...");

// summary of inferencing settings (from model_metadata.h)
ei_printf("Inferencing settings:\n");

    ei_printf("\tImage      resolution:      %dx%d\n",
EI_CLASSIFIER_INPUT_WIDTH,
EI_CLASSIFIER_INPUT_HEIGHT);

    ei_printf("\tFrame      size:      %d\n",
EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE);

    ei_printf("\tNo.      of      classes:      %d\n",
sizeof(ei_classifier_inferencing_categories) /
sizeof(ei_classifier_inferencing_categories[0]));

}
```

```
/**
```

```
* @brief Get data and run inferencing
```

```
*
```

```

* @param[in] debug Get debug info if true
*/
void loop()
{
    //////////////////////////////////////////////////

    bool stop_inferencing = false;

    while(stop_inferencing == false) {
        //////////////////////////////////////////////////

        //MAX30102
        for (int i = 0; i < 500; i++){
            readHeartRate(particleSensor); // runs 500 times the pulse
sensor
        }
        //////////////////////////////////////////////////

        for (int y = 0; y < 1; y++){ // runs 1 time the diagnosis model
ei_printf("\nStarting inferencing in 2 seconds...\n");
        }
        // instead of wait_ms, we'll wait on the signal, this allows
threads to cancel us...
        if (ei_sleep(2000) != EI_IMPULSE_OK) {

```

```

        break;

    }

ei_printf("Taking photo...\n");

if (ei_camera_init() == false) {

    ei_printf("ERR: Failed to initialize image sensor\r\n");

    break;

}

// choose resize dimensions

uint32_t resize_col_sz;

uint32_t resize_row_sz;

bool do_resize = false;

int res = calculate_resize_dimensions(EI_CLASSIFIER_INPUT_WIDTH,
H, EI_CLASSIFIER_INPUT_HEIGHT, &resize_col_sz,
&resize_row_sz, &do_resize);

if (res) {

    ei_printf("ERR: Failed to calculate resize dimensions
(%d)\r\n", res);

    break;

}

void *snapshot_mem = NULL;

uint8_t *snapshot_buf = NULL;

```

```

snapshot_mem      =
ei_malloc(resize_col_sz*resize_row_sz*2);

if(snapshot_mem == NULL) {

    ei_printf("failed to create snapshot_mem\r\n");

    break;

}

snapshot_buf      =      (uint8_t
*)DWORD_ALIGN_PTR((uintptr_t)snapshot_mem);

if (ei_camera_capture(EI_CLASSIFIER_INPUT_WIDTH,
EI_CLASSIFIER_INPUT_HEIGHT, snapshot_buf) == false) {

    ei_printf("Failed to capture image\r\n");

    if (snapshot_mem) ei_free(snapshot_mem);

    break;

}

ei::signal_t signal;

signal.total_length = EI_CLASSIFIER_INPUT_WIDTH *
EI_CLASSIFIER_INPUT_HEIGHT;

signal.get_data = &ei_camera_cutout_get_data;

// run the impulse: DSP, neural network and the Anomaly
algorithm

ei_impulse_result_t result = { 0 };

EI_IMPULSE_ERROR ei_error = run_classifier(&signal,
&result, debug_nn);

if (ei_error != EI_IMPULSE_OK) {

```

```

    ei_printf("Failed to run impulse (%d)\n", ei_error);

    ei_free(snapshot_mem);

    break;

}

// print the predictions

ei_printf("Predictions (DSP: %d ms., Classification: %d ms.,
Anomaly: %d ms.):\n",
          result.timing.dsp, result.timing.classification,
          result.timing.anomaly);

String detectedLabel = "";

#if EI_CLASSIFIER_OBJECT_DETECTION == 1

ei_printf("Object detection bounding boxes:\r\n");

for (uint32_t i = 0; i < result.bounding_boxes_count; i++) {

    ei_impulse_result_bounding_box_t bb =
result.bounding_boxes[i];

    if (bb.value == 0) {

        continue;

    }

    ei_printf(" %s (%f) [ x: %u, y: %u, width: %u, height:
%u ]\r\n",
              bb.label,
              bb.value,
              bb.x,
              bb.y,
              bb.width,
              bb.height);
}

```

```

///////////////////////////////
String data;

// print a message

if (beatAvg <= 40){

    ei_printf("Attention!! Position: %s, Low Heart Rate =
%d\n", bb.label, beatAvg);

    data = String("Position: ") + String(bb.label) + ", Low
Heart Rate = " + String(beatAvg);

}

else if (beatAvg >= 100){

    ei_printf("Attention!! Position: %s, High Heart Rate =
%d\n", bb.label, beatAvg);

    data = String("Position: ") + String(bb.label) + ", High
Heart Rate = " + String(beatAvg);

}

else{

    ei_printf("Position: %s, Heart Rate = %d\n", bb.label,
beatAvg);

    data = String("Position: ") + String(bb.label) + ", Heart
Rate = " + String(beatAvg);

}

delay(500);

}

// Print the prediction results (classification)

#else

ei_printf("Predictions:\r\n");

```

```

        for (uint16_t i = 0; i < EI_CLASSIFIER_LABEL_COUNT;
i++) {
    ei_printf(" %s: ", ei_classifier_inferencing_categories[i]);
    ei_printf("%.5f\n", result.classification[i].value);
}

#endif

// Print anomaly result (if it exists)

#if EI_CLASSIFIER_HAS_ANOMALY
ei_printf("Anomaly prediction: %.3f\n", result.anomaly);
#endif

#if EI_CLASSIFIER_HAS_VISUAL_ANOMALY
ei_printf("Visual anomalies:\r\n");
for (uint32_t i = 0; i < result.visual_ad_count; i++) {
    ei_impulse_result_bounding_box_t bb =
result.visual_ad_grid_cells[i];
    if (bb.value == 0) {
        continue;
    }
    ei_printf(" %s (%f) [ x: %u, y: %u, width: %u, height:
%u ]\r\n",
bb.label,
bb.value,
bb.x,
bb.y,
bb.width,

```

```

        bb.height);

    }

#endif

while (ei_get_serial_available() > 0) {

    if (ei_get_serial_byte() == 'b') {

        ei_printf("Inferencing stopped by user\r\n");

        stop_inferencing = true;

    }

}

if (snapshot_mem) ei_free(snapshot_mem);

beatAvg = 0;

}

}

ei_camera_deinit();

}

// end void loop

/**

 * @brief Determine whether to resize and to which dimension

 *

 * @param[in] out_width width of output image

 * @param[in] out_height height of output image

```

```

* @param[out] resize_col_sz      pointer to frame buffer's
column/width value

* @param[out] resize_row_sz      pointer to frame buffer's
rows/height value

* @param[out] do_resize      returns whether to resize (or not)

*

*/
int calculate_resize_dimensions(uint32_t out_width, uint32_t
out_height, uint32_t *resize_col_sz, uint32_t *resize_row_sz,
bool *do_resize)

{
    size_t list_size = 2;

    const ei_device_resize_resolutions_t list[list_size] = { {42,32},
{128,96} };

    // (default) conditions

        *resize_col_sz      =
EI_CAMERA_RAW_FRAME_BUFFER_COLS;

        *resize_row_sz      =
EI_CAMERA_RAW_FRAME_BUFFER_ROWS;

        *do_resize = false;

    for (size_t ix = 0; ix < list_size; ix++) {

        if ((out_width <= list[ix].width) && (out_height <=
list[ix].height)) {

            *resize_col_sz = list[ix].width;

            *resize_row_sz = list[ix].height;

            *do_resize = true;

```

```

        break;

    }

}

return 0;

}

/***
 * @brief Setup image sensor & start streaming
 *
 * @retval false if initialisation failed
 */

bool ei_camera_init(void) {
    if (is_initialised) return true;

    if (!Cam.begin(QQVGA, RGB565, 1)) { // VGA downsampled
        to QQVGA (OV7675)
        ei_printf("ERR: Failed to initialize camera\r\n");
        return false;
    }

    is_initialised = true;

    return true;
}

/***

```

```

* @brief Stop streaming of sensor data
*/
void ei_camera_deinit(void) {
    if (is_initialised) {
        Cam.end();
        is_initialised = false;
    }
}

/***
* @brief Capture, rescale and crop image
*
* @param[in] img_width width of output image
* @param[in] img_height height of output image
* @param[in] out_buf pointer to store output image, NULL
may be used
* when full resolution is expected.
*
* @retval false if not initialised, image captured, rescaled or
cropped failed
*
*/
bool ei_camera_capture(uint32_t img_width, uint32_t
img_height, uint8_t *out_buf)
{
    if (!is_initialised) {
        ei_printf("ERR: Camera is not initialized\r\n");

```

```

        return false;
    }

    if (!out_buf) {
        ei_printf("ERR: invalid parameters\r\n");
        return false;
    }

    // choose resize dimensions
    int res = calculate_resize_dimensions(img_width, img_height,
    &resize_col_sz, &resize_row_sz, &do_resize);
    if (res) {
        ei_printf("ERR: Failed to calculate resize dimensions
(%d)\r\n", res);
        return false;
    }

    if ((img_width != resize_col_sz)
        || (img_height != resize_row_sz)) {
        do_crop = true;
    }

    Cam.readFrame(out_buf); // captures image and resizes

    if (do_crop) {
        uint32_t crop_col_sz;
        uint32_t crop_row_sz;

```

```

        uint32_t crop_col_start;
        uint32_t crop_row_start;
        crop_row_start = (resize_row_sz - img_height) / 2;
        crop_col_start = (resize_col_sz - img_width) / 2;
        crop_col_sz = img_width;
        crop_row_sz = img_height;

        //ei_printf("crop    cols:    %d,    rows:    %d\r\n",
        crop_col_sz,crop_row_sz);
        cropImage(resize_col_sz, resize_row_sz,
                  out_buf,
                  crop_col_start, crop_row_start,
                  crop_col_sz, crop_row_sz,
                  out_buf,
                  16);

    }

// The following variables should always be assigned
// if this routine is to return true
// cutout values
//ei_camera_snapshot_is_resized = do_resize;
//ei_camera_snapshot_is_cropped = do_crop;
ei_camera_capture_out = out_buf;

return true;
}

```

```

/**

* @brief    Convert RGB565 raw camera buffer to RGB888

*

* @param[in] offset      pixel offset of raw buffer
* @param[in] length     number of pixels to convert
* @param[out] out_buf    pointer to store output image

*/

int ei_camera_cutout_get_data(size_t offset, size_t length, float
*out_ptr) {

    size_t pixel_ix = offset * 2;
    size_t bytes_left = length;
    size_t out_ptr_ix = 0;

    // read byte for byte

    while (bytes_left != 0) {

        // grab the value and convert to r/g/b

        uint16_t pixel = (ei_camera_capture_out[pixel_ix] << 8) |
ei_camera_capture_out[pixel_ix+1];

        uint8_t r, g, b;

        r = ((pixel >> 11) & 0x1f) << 3;
        g = ((pixel >> 5) & 0x3f) << 2;
        b = (pixel & 0x1f) << 3;

        // then convert to out_ptr format

        float pixel_f = (r << 16) + (g << 8) + b;
        out_ptr[out_ptr_ix] = pixel_f;
    }
}

```

```

        // and go to the next pixel

        out_ptr_ix++;
        pixel_ix+=2;
        bytes_left--;
    }

    // and done!

    return 0;
}

// This include file works in the Arduino environment
// to define the Cortex-M intrinsics

#ifndef __ARM_FEATURE SIMD32
#include <device.h>
#endif

// This needs to be < 16 or it won't fit. Cortex-M4 only has SIMD
for signed multiplies

#define FRAC_BITS 14
#define FRAC_VAL (1<<FRAC_BITS)
#define FRAC_MASK (FRAC_VAL - 1)

//
// Resize

//
// Assumes that the destination buffer is dword-aligned
// Can be used to resize the image smaller or larger

```

```

// If resizing much smaller than 1/3 size, then a more robust
algorithm should average all of the pixels

// This algorithm uses bilinear interpolation - averages a 2x2
region to generate each new pixel

//

// Optimized for 32-bit MCUs

// supports 8 and 16-bit pixels

void resizeImage(int srcWidth, int srcHeight, uint8_t *srcImage,
int dstWidth, int dstHeight, uint8_t *dstImage, int iBpp)

{
    uint32_t src_x_accum, src_y_accum; // accumulators and
fractions for scaling the image

    uint32_t x_frac, nx_frac, y_frac, ny_frac;

    int x, y, ty, tx;

    if (iBpp != 8 && iBpp != 16)
        return;

    src_y_accum = FRAC_VAL/2; // start at 1/2 pixel in to account
for integer downsampling which might miss pixels

    const uint32_t src_x_frac = (srcWidth * FRAC_VAL) /
dstWidth;

    const uint32_t src_y_frac = (srcHeight * FRAC_VAL) /
dstHeight;

    const uint32_t r_mask = 0xf800f800;
    const uint32_t g_mask = 0x07e007e0;
    const uint32_t b_mask = 0x001f001f;
    uint8_t *s, *d;
    uint16_t *s16, *d16;
}

```

```

    uint32_t x_frac2, y_frac2; // for 16-bit SIMD

    for (y=0; y < dstHeight; y++) {

        ty = src_y_accum >> FRAC_BITS; // src y

        y_frac = src_y_accum & FRAC_MASK;

        src_y_accum += src_y_frac;

        ny_frac = FRAC_VAL - y_frac; // y fraction and 1.0 - y
        fraction

        y_frac2 = ny_frac | (y_frac << 16); // for M4/M4 SIMD

        s = &srcImage[ty * srcWidth];

        s16 = (uint16_t *)&srcImage[ty * srcWidth * 2];

        d = &dstImage[y * dstWidth];

        d16 = (uint16_t *)&dstImage[y * dstWidth * 2];

        src_x_accum = FRAC_VAL/2; // start at 1/2 pixel in to
        account for integer downsampling which might miss pixels

        if (iBpp == 8) {

            for (x=0; x < dstWidth; x++) {

                uint32_t tx, p00,p01,p10,p11;

                tx = src_x_accum >> FRAC_BITS;

                x_frac = src_x_accum & FRAC_MASK;

                nx_frac = FRAC_VAL - x_frac; // x fraction and 1.0 - x
                fraction

                x_frac2 = nx_frac | (x_frac << 16);

                src_x_accum += src_x_frac;

                p00 = s[tx]; p10 = s[tx+1];

                p01 = s[tx+srcWidth]; p11 = s[tx+srcWidth+1];

            #ifdef __ARM_FEATURE SIMD32

```

```

        p00 = __SMLAD(p00 | (p10<<16), x_frac2,
FRAC_VAL/2)>> FRAC_BITS; // top line

        p01 = __SMLAD(p01 | (p11<<16), x_frac2,
FRAC_VAL/2)>> FRAC_BITS; // bottom line

        p00 = __SMLAD(p00 | (p01<<16), y_frac2,
FRAC_VAL/2)>> FRAC_BITS; // combine

#else // generic C code

    p00 = ((p00 * nx_frac) + (p10 * x_frac) + FRAC_VAL/2)
>> FRAC_BITS; // top line

    p01 = ((p01 * nx_frac) + (p11 * x_frac) + FRAC_VAL/2)
>> FRAC_BITS; // bottom line

    p00 = ((p00 * ny_frac) + (p01 * y_frac) + FRAC_VAL/2)
>> FRAC_BITS; // combine top + bottom

#endif // Cortex-M4/M7

*d++ = (uint8_t)p00; // store new pixel

} // for x

} // 8-bpp

else

{ // RGB565

for (x=0; x < dstWidth; x++) {

    uint32_t tx, p00,p01,p10,p11;

    uint32_t r00, r01, r10, r11, g00, g01, g10, g11, b00, b01,
b10, b11;

    tx = src_x_accum >> FRAC_BITS;

    x_frac = src_x_accum & FRAC_MASK;

    nx_frac = FRAC_VAL - x_frac; // x fraction and 1.0 - x
fraction

    x_frac2 = nx_frac | (x_frac << 16);
}

```

```

src_x_accum += src_x_frac;

p00 = __builtin_bswap16(s16[tx]); p10 =
__builtin_bswap16(s16[tx+1]);

p01 = __builtin_bswap16(s16[tx+srcWidth]); p11 =
__builtin_bswap16(s16[tx+srcWidth+1]);

#ifndef __ARM_FEATURE SIMD32

{
    p00 |= (p10 << 16);
    p01 |= (p11 << 16);

    r00 = (p00 & r_mask) >> 1; g00 = p00 & g_mask; b00 =
    p00 & b_mask;

    r01 = (p01 & r_mask) >> 1; g01 = p01 & g_mask; b01 =
    p01 & b_mask;

    r00 = __SMLAD(r00, x_frac2, FRAC_VAL/2) >>
    FRAC_BITS; // top line

    r01 = __SMLAD(r01, x_frac2, FRAC_VAL/2) >>
    FRAC_BITS; // bottom line

    r00 = __SMLAD(r00 | (r01<<16), y_frac2, FRAC_VAL/2)
    >> FRAC_BITS; // combine

    g00 = __SMLAD(g00, x_frac2, FRAC_VAL/2) >>
    FRAC_BITS; // top line

    g01 = __SMLAD(g01, x_frac2, FRAC_VAL/2) >>
    FRAC_BITS; // bottom line

    g00 = __SMLAD(g00 | (g01<<16), y_frac2,
    FRAC_VAL/2) >> FRAC_BITS; // combine

    b00 = __SMLAD(b00, x_frac2, FRAC_VAL/2) >>
    FRAC_BITS; // top line

    b01 = __SMLAD(b01, x_frac2, FRAC_VAL/2) >>
    FRAC_BITS; // bottom line
}

```

```

        b00 = __SMLAD(b00 | (b01<<16), y_frac2,
FRAC_VAL/2)>> FRAC_BITS; // combine

    }

#else // generic C code

{

    r00 = (p00 & r_mask) >> 1; g00 = p00 & g_mask; b00 =
p00 & b_mask;

    r10 = (p10 & r_mask) >> 1; g10 = p10 & g_mask; b10 =
p10 & b_mask;

    r01 = (p01 & r_mask) >> 1; g01 = p01 & g_mask; b01 =
p01 & b_mask;

    r11 = (p11 & r_mask) >> 1; g11 = p11 & g_mask; b11 =
p11 & b_mask;

    r00 = ((r00 * nx_frac) + (r10 * x_frac) + FRAC_VAL/2)
>> FRAC_BITS; // top line

    r01 = ((r01 * nx_frac) + (r11 * x_frac) + FRAC_VAL/2)
>> FRAC_BITS; // bottom line

    r00 = ((r00 * ny_frac) + (r01 * y_frac) + FRAC_VAL/2)
>> FRAC_BITS; // combine top + bottom

    g00 = ((g00 * nx_frac) + (g10 * x_frac) + FRAC_VAL/2)
>> FRAC_BITS; // top line

    g01 = ((g01 * nx_frac) + (g11 * x_frac) + FRAC_VAL/2)
>> FRAC_BITS; // bottom line

    g00 = ((g00 * ny_frac) + (g01 * y_frac) + FRAC_VAL/2)
>> FRAC_BITS; // combine top + bottom

    b00 = ((b00 * nx_frac) + (b10 * x_frac) + FRAC_VAL/2)
>> FRAC_BITS; // top line

    b01 = ((b01 * nx_frac) + (b11 * x_frac) + FRAC_VAL/2)
>> FRAC_BITS; // bottom line

```

```

        b00 = ((b00 * ny_frac) + (b01 * y_frac) + FRAC_VAL/2)
        >> FRAC_BITS; // combine top + bottom

    }

#endif // Cortex-M4/M7

r00 = (r00 << 1) & r_mask;

g00 = g00 & g_mask;

b00 = b00 & b_mask;

p00 = (r00 | g00 | b00); // re-combine color components

*d16++ = (uint16_t)__builtin_bswap16(p00); // store new
pixel

} // for x

} // 16-bpp

} // for y

} /* resizeImage() */

//



// Crop

//



// Assumes that the destination buffer is dword-aligned

// optimized for 32-bit MCUs

// Supports 8 and 16-bit pixels

//


void cropImage(int srcWidth, int srcHeight, uint8_t *srcImage,
int startX, int startY, int dstWidth, int dstHeight, uint8_t
*dstImage, int iBpp)

{
    uint32_t *s32, *d32;
    int x, y;
}

```

```

    if (startX < 0 || startX >= srcWidth || startY < 0 || startY >=
srcHeight || (startX + dstWidth) > srcWidth || (startY + dstHeight)
> srcHeight)

        return; // invalid parameters

    if (iBpp != 8 && iBpp != 16)

        return;

if (iBpp == 8) {

    uint8_t *s, *d;

    for (y=0; y<dstHeight; y++) {

        s = &srcImage[srcWidth * (y + startY) + startX];

        d = &dstImage[(dstWidth * y)];

        x = 0;

        if (((intptr_t)s & 3 || (intptr_t)d & 3) { // either src or dst
pointer is not aligned

            for (; x<dstWidth; x++) {

                *d++ = *s++; // have to do it byte-by-byte

            }

        } else {

            // move 4 bytes at a time if aligned or alignment not enforced

            s32 = (uint32_t *)s;

            d32 = (uint32_t *)d;

            for (; x<dstWidth-3; x+= 4) {

                *d32++ = *s32++;

            }

            // any remaining stragglers?

        }

    }

}

```

```

s = (uint8_t *)s32;

d = (uint8_t *)d32;

for (; x<dstWidth; x++) {

    *d++ = *s++;

}

}

}

// for y

} // 8-bpp

else

{

    uint16_t *s, *d;

    for (y=0; y<dstHeight; y++) {

        s = (uint16_t *)&srcImage[2 * srcWidth * (y + startY) +
startX * 2];

        d = (uint16_t *)&dstImage[(dstWidth * y * 2)];

        x = 0;

        if (((intptr_t)s & 2 || (intptr_t)d & 2) { // either src or dst
pointer is not aligned

            for (; x<dstWidth; x++) {

                *d++ = *s++; // have to do it 16-bits at a time

            }

        } else {

            // move 4 bytes at a time if aligned or alignment no enforced

            s32 = (uint32_t *)s;

            d32 = (uint32_t *)d;

            for (; x<dstWidth-1; x+= 2) { // we can move 2 pixels at a
time

```

```

*d32++ = *s32++;
}

// any remaining stragglers?

s = (uint16_t *)s32;
d = (uint16_t *)d32;

for (; x<dstWidth; x++) {

    *d++ = *s++;
}

}

}

} // for y

} // 16-bpp case

} /* cropImage() */

#ifndef           !defined(EI_CLASSIFIER_SENSOR)      ||
EI_CLASSIFIER_SENSOR          !=      ||
EI_CLASSIFIER_SENSOR_CAMERA

#error "Invalid model for current sensor"

#endif

// OV767X camera library override

#include <Arduino.h>

#include <Wire.h>

#define digitalPinToBitMask(P) (1 << (digitalPinToPinName(P)
% 32))

#define portInputRegister(P) ((P == 0) ? &NRF_P0->IN :
&NRF_P1->IN)

```

```

//  

// OV7675::begin()  

//  

// Extends the OV767X library function. Some private variables  

// are needed  

// to use the OV7675::readFrame function.  

//  

int OV7675::begin(int resolution, int format, int fps)  

{  

    pinMode(OV7670_VSYNC, INPUT);  

    pinMode(OV7670_HREF, INPUT);  

    pinMode(OV7670_PLK, INPUT);  

    pinMode(OV7670_XCLK, OUTPUT);  

    vsyncPort      =  

    portInputRegister(digitalPinToPort(OV7670_VSYNC));  

    vsyncMask = digitalPinToBitMask(OV7670_VSYNC);  

    hrefPort       =  

    portInputRegister(digitalPinToPort(OV7670_HREF));  

    hrefMask = digitalPinToBitMask(OV7670_HREF);  

    pclkPort      =  

    portInputRegister(digitalPinToPort(OV7670_PLK));  

    pclkMask = digitalPinToBitMask(OV7670_PLK);  

    // init driver to use full image sensor size  

    bool ret = OV767X::begin(VGA, format, fps);

```

```

width = OV767X::width(); // full sensor width

height = OV767X::height(); // full sensor height

bytes_per_pixel = OV767X::bytesPerPixel();

bytes_per_row = width * bytes_per_pixel; // each pixel is 2

bytes

resize_height = 2;

buf_mem = NULL;

raw_buf = NULL;

intrp_buf = NULL;

//allocate_scratch_buffs();

return ret;

} /* OV7675::begin() */

```

```

int OV7675::allocate_scratch_buffs()

{

//ei_printf("allocating buffers..\r\n");

buf_rows = height / resize_row_sz * resize_height;

buf_size = bytes_per_row * buf_rows;

buf_mem = ei_malloc(buf_size);

if(buf_mem == NULL) {

ei_printf("failed to create buf_mem\r\n");

return false;

}

```

```

        raw_buf          =      (uint8_t
*)DWORD_ALIGN_PTR((uintptr_t)buf_mem);

//ei_printf("allocating buffers OK\r\n");

return 0;

}

int OV7675::deallocate_scratch_buffs()
{
    //ei_printf("deallocating buffers...\r\n");
    ei_free(buf_mem);
    buf_mem = NULL;

    //ei_printf("deallocating buffers OK\r\n");

    return 0;
}

//  

// OV7675::readFrame()  

//  

// Overrides the OV767X library function. Fixes the camera  

// output to be  

// a far more desirable image. This image utilizes the full sensor  

// size  

// and has the correct aspect ratio. Since there is limited memory  

// on the

```

```

// Nano we bring in only part of the entire sensor at a time and
then

// interpolate to a lower resolution.

//

void OV7675::readFrame(void* buffer)

{

allocate_scratch_buffs();

uint8_t* out = (uint8_t*)buffer;

noInterrupts();

// Falling edge indicates start of frame

while ((*vsyncPort & vsyncMask) == 0); // wait for HIGH

while ((*vsyncPort & vsyncMask) != 0); // wait for LOW

int out_row = 0;

for (int raw_height = 0; raw_height < height; raw_height +=

buf_rows) {

    // read in 640xbuf_rows buffer to work with

    readBuf();

    resizeImage(width, buf_rows,

    raw_buf,

    resize_col_sz, resize_height,

    &(out[out_row]),

    16);
}

```

```

        out_row += resize_col_sz * resize_height * bytes_per_pixel;
        /* resize_col_sz * 2 * 2 */

    }

interrupts();

deallocate_scratch_buffs();

} /* OV7675::readFrame() */

//  

// OV7675::readBuf()  

//  

// Extends the OV767X library function. Reads buf_rows VGA  

rows from the  

// image sensor.  

//  

void OV7675::readBuf()  

{  

    int offset = 0;  

    uint32_t ulPin = 33; // P1.xx set of GPIO is in 'pin' 32 and above  

    NRF_GPIO_Type * port;  

    port = nrf_gpio_pin_port_decode(&ulPin);  

    for (int i = 0; i < buf_rows; i++) {  

        // rising edge indicates start of line

```

```

        while ((*hrefPort & hrefMask) == 0); // wait for HIGH

    }

    for (int col = 0; col < bytes_per_row; col++) {

        // rising edges clock each data byte

        while ((*pclkPort & pclkMask) != 0); // wait for LOW

        uint32_t in = port->IN; // read all bits in parallel

        in >>= 2; // place bits 0 and 1 at the "bottom" of the register

        in &= 0x3f03; // isolate the 8 bits we care about

        in |= (in >> 6); // combine the upper 6 and lower 2 bits

        raw_buf[offset++] = in;

        while ((*pclkPort & pclkMask) == 0); // wait for HIGH

    }

    while ((*hrefPort & hrefMask) != 0); // wait for LOW

}

} /* OV7675::readBuf() */

```

Βιβλιογραφία

- [1] <https://www.e-trikala.gr/portfolio/telecare/?id=1012>
- [2] <https://www.arduino.cc/en/Guide/Introduction>
- [3] <https://wiring.org.co/>
- [4] <https://processing.org/>
- [5] <https://dl.acm.org/doi/abs/10.5555/1406766>
- [6] https://projecthub.arduino.cc/abid_hossain/air-quality-monitor-14f9b4?utm_source=chatgpt.com
- [7] https://projecthub.arduino.cc/angadiameya007/smart-street-light-6ad038?utm_source=chatgpt.com

[8]https://www.youtube.com/watch?v=ZMC_o2gjbVc&ab_channel=HVSTechnologies

[9]https://docs.arduino.cc/resources/datasheets/ABX00031-datasheet.pdf?_gl=1*w0r5yp*_up*MQ..*_ga*MjEwNTUzOTgyNS4xNzM0MTA2Mzgz*_ga_NEXN8H46L5*MTczNDEwNjM4MS4xLjAuMTczNDEwNjM4MS4wLjAuODkyNTg5Nzk2

[10]https://docs.arduino.cc/resources/pinouts/ABX00031-full-pinout.pdf?_gl=1*nkfpd0*_up*MQ..*_ga*MjEwNTUzOTgyNS4xNzM0MTA2Mzgz*_ga_NEXN8H46L5*MTczNDEwNjM4MS4xLjAuMTczNDEwNjM4MS4wLjAuODkyNTg5Nzk2

[11] <https://docs.arduino.cc/hardware/nano-33-ble-sense/>

[12]<https://www.vectornav.com/resources/inertial-navigation-primer/hardware/synccomm>

[13]<https://support.arduino.cc/hc/en-us/articles/360018922239>About-the-AREF-pin>

[14] <https://www.geeksforgeeks.org/what-is-flash-memory/>

[15]https://docs.arduino.cc/tutorials/nano-33-ble-sense/imu-accelerometer/?_gl=1*1ukzthw*_up*MQ..*_ga*MTYyNDIwNjQxNy4xNzM0MzU3NTQ1*_ga_NEXN8H46L5*MTczNDM1NzU0NC4xLjAuMTczNDM1NzU0NC4wLjAuMzg4NDU1NzQ0

[16]<https://docs.arduino.cc/tutorials/nano-33-ble-sense/microphone-sensor/>

[17]https://docs.arduino.cc/tutorials/nano-33-ble-sense/gesture-sensor/?_gl=1*tvty3d*_up*MQ..*_ga*MTkyODY1MDcxNy4xNzM0MzUxNDM5*_ga_NEXN8H46L5*MTczNDM1MTQzOC4xLjAuMTczNDM1MTQzOC4wLjAuMTY3NDU3NzIyNQ..

[18]<https://docs.arduino.cc/tutorials/nano-33-ble-sense/humidity-and-temperature-sensor/>

[19]<https://docs.arduino.cc/tutorials/nano-33-ble-sense/barometric-sensor/>

[20]<https://lastminuteengineers.com/wemos-d1-mini-pinout-reference/>

[21]https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

[22]<https://ardustore.dk/error/Manuel%20-%20D1%20Mini.pdf>

[23]<https://docs.arduino.cc/learn/microcontrollers/analog-output/>

[24]<https://learnelectronics.gr/to-%CF%80%CF%81%CF%89%CF%84%CF%8C%CE%BA%C E%BF%CE%BB%CE%BB%CE%BF-%CE%B5%CF%80%CE%B9%CE%BA%CE%BF%CE%B9%CE%BD%CF%89%CE%BD%CE%AF%CE%B1%CF%82-i2c/>

[25]<https://www.geeksforgeeks.org/what-is-serial-peripheral-interface-spi/>

[26] <https://docs.arduino.cc/learn/communication/one-wire/>

[27]https://www.wemos.cc/en/latest/d1/d1_mini.html

[28]<https://lastminuteengineers.com/max30100-pulse-oximeter-heart-rate-sensor-arduino-tutorial/>

[29]<https://www.analog.com/media/en/technical-documentation/data-sheets/MAX30102.pdf>

[30] ΙΩΑΝΝΗΣ ΤΣΑΚΝΑΚΗΣ

91_%CE%98%CE%B5%CF%89%CF%81%CE%AF%CE%B1%CE%9F%CF%85%CF%81%CE%AC.pdf

[31]<https://www.geeksforgeeks.org/interrupts/>

[32]<https://www.analog.com/media/en/technical-documentation/data-sheets/MAX30100.pdf>

[33]<https://my.clevelandclinic.org/health/diagnostics/23429-heart-rate-monitor>

[34]https://openaccess.thecvf.com/content_cvpr_2014/papers/Li_Remote_Hart_Rate_2014_CVPR_paper.pdf

[35]<https://en.wikipedia.org/wiki/Photoplethysmogram>

[36]<https://el.wikipedia.org/wiki/%CE%91%CE%B9%CE%BC%CE%BF%CF%83%CF%86%CE%B1%CE%B9%CF%81%CE%AF%CE%BD%CE%B7>

[37]<https://www.metropolitan-hospital.gr/el/metropolitan-blog/%CE%BA%CF%85%CE%BA%CE%BB%CE%BF%CF%86%CE%BF%CF%81%CE%BA%CE%BF%8C-%CF%83%CF%8D%CF%83%CF%84%CE%B7%CE%BC%C%E%B1/2474-%CE%BA%CE%B1%CF%81%CE%B4%CE%B9%CE%B1%CE%BA%CE%AD%CF%82-%CE%B1%CF%81%CF%81%CF%85%CE%B8%CE%BC%C%E%AF%CE%B5%CF%82-%CF%84%CE%B9-%CF%80%CF%81%CE%BF%CE%BA%CE%B1%CE%BB%CE%BF%CF%8D%CE%BD-%CF%83%CF%84%CE%BF%CE%BD-%CE%BF%CF%81%CE%B3%CE%B1%CE%BD%CE%B9%CF%83%CE%BC%CF%8C-%CE%BC%CE%B1%CF%82-%CE%BA%CE%B1%CE%B9-%CF%80%CF%8E%CF%82-%CE%BC%CF%80%CE%BF%CF%81%CE%BF%CF%8D%CE%BC%CE%B5-%CE%BD%CE%B1-%CF%84%CE%B9%CF%82-%CE%B1%CE%BD%CF%84%CE%B9%CE%BC%CE%B5%>

CF%84%CF%89%CF%80%CE%AF%CF%83%CE%BF%CF
%85%CE%BC%CE%B5

[38]<https://my.clevelandclinic.org/health/diseases/17727-hypoxemia>

[39]https://www.researchgate.net/profile/Arni-Markom/publication/352264532_IoT_Health_Monitoring_Device_of_Oxygen_Saturation_SpO2_and_Heart_Rate_Level/links/62504e67b0cee02d695b8aaf/IoT-Health-Monitoring-Device-of-Oxygen-Saturation-SpO2-and-Heart-Rate-Level.pdf

[40]<https://www.heartfoundation.org.nz/your-heart/how-the-heart-works>

[41]<https://my.clevelandclinic.org/health/body/21704-heart>

[42]<https://en.wikipedia.org/wiki/Electrocardiography>

[43]<https://www.mayoclinic.org/tests-procedures/ekg/about/pac-20384983>

[44]https://www.researchgate.net/figure/Sample-of-visualization-of-input-data-3-ECG-leads-upper-curves-and-3-channel-motion_fig1_242375541

[45]<https://www.hopkinsmedicine.org/health/treatment-tests-and-therapies/electrocardiogram>

[46]<https://ia601608.us.archive.org/7/items/guyton-and-hall-textbook-of-medical-physiology-14ed/Guyton%20and%20Hall%20Textbook%20of%20Medical%20Physiology,%2014ed.pdf>

[47]<https://diec.unizar.es/~laguna/personal/publicaciones/libroWiley.pdf>

[48]<https://www.doctor33.it/cont/download-center-files/17519/cap-electrocardiography-x20968allp1.pdf>

[49]<https://cdn.sparkfun.com/datasheets/Sensors/Biometric/AD8232.pdf>

- [50]<https://repository-api.ihu.gr/server/api/core/bitstreams/90a74663-b886-43a3-88a9-b696fdb3bad2/content>
- [51]<https://www.pulseai.io/blog/why-electrodes-matter-electrode-electrolyte-interface>
- [52]https://en.wikipedia.org/wiki/Common-mode_signal
- [53]https://en.wikipedia.org/wiki/Pull-up_resistor
- [54]<https://www.emqx.com/en/blog/the-easiest-guide-to-getting-started-with-mqtt>
- [55]<https://www.paessler.com/it-explained/mqtt>
- [56]<https://www.fortinet.com/resources/cyberglossary/tcp-ip>
- [57]https://en.wikipedia.org/wiki/Artificial_intelligence
- [58]<https://www.ibm.com/think/topics/artificial-intelligence>
- [59]<https://dac.digital/what-are-the-practical-applications-of-modern-computer-vision-technology/>
- [60] <https://en.wikipedia.org/wiki/Eigenface>
- [61] <https://en.wikipedia.org/wiki/AlexNet>
- [62]<https://www.ibm.com/think/topics/computer-vision>
- [63]https://en.wikipedia.org/wiki/Computer_vision
- [64]<https://www.algotive.ai/blog/what-is-computer-vision-and-how-does-it-work-with-artificial-intelligence>
- [65]https://medium.com/@azadsingh_35336/introduction-to-machine-learning-20c919899efe
- [66]<https://www.ibm.com/think/topics/machine-learning>
- [67]<https://www.ibm.com/think/topics/neural-networks>
- [68][https://en.wikipedia.org/wiki/Neural_network_\(machine_learning\)](https://en.wikipedia.org/wiki/Neural_network_(machine_learning))
- [69]<https://www.deepinstinct.com/glossary/deep-learning>

[70]<https://builtin.com/machine-learning/deep-learning>

[71]<https://www.ibm.com/think/topics/deep-learning>

[72]<https://www.arduino.cc/en/software>

[73]<https://test.mosquitto.org/>

[74]<https://arduino.github.io/arduino-cli/1.1/installation/>

[75]<https://docs.edgeimpulse.com/docs/edge-ai-hardware/mcu/arduino-nano-33-ble-sense>

[76]https://github.com/sparkfun/SparkFun_MAX3010x_Sensor_Library

[77]https://github.com/knolleary/pubsubclient/blob/master/examples/mqtt_esp8266/mqtt_esp8266.ino